БИБЛИОТЕКА ПРОГРАММИСТА

Стивен Холзнер

XSLT

- исчерпывающее описание XSLT 1.0
- рекомендации по работе с XSLT 1.1
- требования и детали реализации XSLT 2.0
- применение XSLT для разработки страниц ASP, JSP и сервлетов JAVA

New Riders

ПИТЕР

Steven Holzner

Inside XSLT



i

www.newriders.com B
201 West 103rd Street, Indianapolis, Indiana 46290
An Imprint of Pearson Education
Boston • Indianapolis • London • Munich • New York • San Fransico

Стивен Холзнер

БИБЛИОТЕКА ПРОГРАММИСТА

XSLT

ПИТЕР®

Москва • Санкт-Петербург • Нижний Новгород - Воронеж
Ростов-на-Дону • Екатеринбург • Самара
Киев »Харьков • Минск
2002

Стивен Холзнвр

XSLT библиотека программиста

Перевел с английского А. Войтенко

Главный редактор	*Е. Строганова*
Заведующий редакцией	*И. Корнеева*
Руководитель проекта	*А. Пасечник*
Научный редактор	*А. Пасечник*
Корректоры	*Н. Лукина, А. Моносов*
Верстка	*Р. Гришанов*

ББК 32.988-018
УДК 681.3.06

Холзнер С.

X71 XSLT библиотека программиста, — СПб.: Питер, 2002. — 544 е.: ил.

ISBN 5-94723-271-5

Эта книга посвящена XSLT — мощным и удобным системам преобразования документов. В ней не только рассматривается весь синтаксис XSLT, от основ до специальных аспектов, но и подробно излагаются современные способы использования XSLT.

Вы найдете сотни практических примеров, таких как осуществление преобразований XSLT на web-сереерах, подключение к базам данных и использование браузеров для осуществления преобразований на лету.

Книга рассчитана на всех, кто хочет изучать XSLT и использовать его на практике.

© New Riders Publishing, 2002

© Перевод на русский язык, ЗАО Издательский дом «Питер», 2002

© Издание на русском языке, оформление, ЗАО Издательский дом «Питер», 2002

Права на издание получены по соглашению с New Riders Publishing.

Вое права защищены. Никакая часть данной книга не может быть воспроизведена в какой бы то ни было форме без письменного разрешения владельцев авторских прав.

Информация, содержащаяся в данной книге, получена из источников, рассматриваемых издательством как надежные. Тем не менее, имея в виду возможные человеческие или технические ошибки, издательство не может гарантировать абсолютную точность и полноту приводимых сведений и не несет ответственности за возможные ошибки, связанные с использованием книга.

ISBN 5-94723-271-5

ISBN 0-7357-1136-4 (англ.)

ООО «Питер Принт». 196105, Санкт-Петербург, ул. Благодатная, д. 67в.
Лицензия ИД № 05784 от 07.09.01.
Налоговая льгота - общероссийский классификатор продукции ОК 005-93, том 2; 953005 - литература учебная.
Подписано в печать 29.09.02. Формат 70x100/16. Усл. п. л. 43,86. Тираж 3000 экз. Заказ № 1468.
Отпечатано с готовых диапозитивов в ФГУП «Печатный двор» им. А. М. Горького
Министерства РФ по делам печати, телерадиовещания и средств массовых коммуникаций.
197110, Санкт-Петербург, Чкалова ий пр., 15.

Краткое содержание

Об авторе . . . 13
О технических редакторах . . . 14
Благодарности . . . 15
Введение . . . 16
От издательства . . . 21
Глава 1 • Основы XSLT . . . 22
Глава 2 • Создание и применение таблиц стилей . . . 68
Глава 3 • Создание и применение шаблонов . . . 111
Глава 4 • Создание образцов выбора . . . 147
Глава 5 • Принятие решений и сортировка данных . . . 187
Глава 6 • Преобразование в XML, HTML, XHTML, RTF . . . 228
Глава 7 • Работа с XPath . . . 271
Глава 8 • Работа с функциями XSLT и XPath . . . 305
Глава 9 • Именованные шаблоны, параметры и переменные . . . 354
Глава 10 • Работа с API процессоров XSLT . . . 388
Глава 11 • Применение XSLT для создания документов XSL-FO . . . 425
Глава 12 • Документы XSL-FO . . . 469
Приложение А • DTD XSLT . . . 508
Приложение Б • Форматирующие свойства XSL-FO . . . 516
Алфавитный указатель . . . 534

Содержание

06 авторе . . . 13

О технических редакторах . . . 14

Благодарности . . . 15

Введение . . . 16

- Материал книги . . . 17
- Для кого предназначена эта книга . . . 19
- На каком уровне написана книга . . . 19
- Используемые соглашения . . . 20

От издательства . . . 21

Глава 1 • Основы XSLT . . . 22

- XSL = XSLT + XSL-FO . . . 22
- Краткая историческая справка . . . 23
  - Преобразования XSLT—XSL . . . 24
  - XSL-FO: форматирующие объекты XSL . . . 24
  - Спецификации W3C . . . 25
  - Версии XSLT . . . 26
- Документы XML . . . 27
  - Хорошо сформированные документы XML . . . 29
  - Допустимые документы XML . . . 29
- Как XML выглядит в браузере? . . . 32
- XSLT-преобразования . . . 33
- Что нужно для XSLT-преобразования . . . 35
- Использование отдельных процессоров XSLT . . . 36
  - Использование процессора XSLT на Java . . . 38
  - XT Джеймса Кларка . . . 39
  - Saxon . . . 40

Oracle XSLT . . . 41
Xalan . . . 41
Преобразование документов XML при помощи браузеров . . . 42
Работа с XSLT и JavaScript в Internet Explorer . . . 45
XSLT-преобразования на web-серверах . . . 47
Преобразования из XML в XML . . . 49
Преобразования из XML в XHTML . . . 51
Ресурсы XSLT . . . 54
Спецификации, руководства и примеры XSLT . . . 54
Редакторы XSLT . . . 55
Утилиты XSLT . . . 58
Форматирующие объекты XSL: XSL-FO . . . 59
Ресурсы XSL-FO . . . 59
Форматирование документа XML . . . 60
Таблица стилей XSLT . . . 61
Преобразование документа в представление с форматирующими объектами . . . 63
Создание форматированного документа . . . 65

Глава 2 • Создание и применение таблиц стилей . . . 68

Деревья и узлы . . . 68
Символ-разделитель . . . 71
Модель информационного множества против модели дерева XSLT . . . 73
Работа с элементами XSLT . . . 74
Инструкция обработки <?xsl:stylesheet?> . . . 75
Элемент <xsl:stylesheet> . . . 77
Пространство имен XSL . . . 78
Элементы таблиц стилей высокого уровня . . . 79
Элемент <xsl:template> . . . 80
Тела шаблонов . . . 81
Инструкции XSLT . . . 81
Элементы расширения . . . 82
Элементы буквального результата . . . 82
Совпадающие элементы в шаблонах . . . 83
Элемент <xsl:apply-templates> . . . 85
Доступ к значениям узлов . . . 87
Поддержка XML Base . . . 92
Выбор методов вывода . . . 93
Метод вывода: HTML . . . 94
Метод вывода: XML . . . 96
Метод вывода: текст . . . 97
Упрощенные таблицы стилей . . . 99
Встроенные таблицы стилей . . . 100
Элемент <xsl:include> . . . 103
Элемент <xsl:import> . . . 104
Элемент <xsl:apply-imports> . . . 105
Преобразование документов XML при помощи Internet Explorer . . . 107

Глава 3 • Создание и применение шаблонов . . . 111

Создание шаблона . . . 111
Обработка дочерних узлов . . . 113
Доступ к значениям узлов . . . 114

Создание образцов выбора . . . 115
Выбор применяемого шаблона . . . 116
Чтение значений атрибутов . . . 119
Элемент <xsl:text> . . . 121
Отключение вывода эскейп-последовательностей . . . 123
Написание значений атрибутов . . . 126
Шаблоны значений атрибутов . . . 127
Обработка символов-разделителей . . . 129
Элементы <xsl:strip-space> и <xsl:preserve-space> . . . 130
Автоматическое выравнивание . . . 132
Правила по умолчанию в шаблоне . . . 135
Удаление содержимого . . . 137
Разрешение конфликтов шаблонов . . . 139
Элемент <xsl:copy> . . . 141
Элемент <xsl:copy-of> . . . 144
Элемент <xsl:message> . . . 145

Глава 4 • Создание образцов выбора . . . 147
Выбор корневого узла . . . 148
Выбор элементов . . . 149
Выбор дочерних элементов . . . 149
Выбор потомков элемента . . . 149
Выбор атрибутов . . . 150
Формальное определение образцов выбора . . . 152
Образцы шага, часть 1: оси образца . . . 154
Сокращенный синтаксис . . . 156
Образцы шага, часть 2: условия узла . . . 157
Выбор комментариев . . . 157
Выбор узлов при помощи node() . . . 158
Выбор текстовых узлов при помощи text() . . . 160
Выбор инструкций обработки . . . 160
Образцы шага, часть 3: предикаты . . . 161
Создание предикатов . . . 163
Предикаты: наборы узлов . . . 164
Предикаты: логические значения . . . 166
Предикаты: числа . . . 169
Строки . . . 170
Предикаты: фрагменты результирующего дерева . . . 173
Сокращенный синтаксис предикатов . . . 173
Выбор по ID . . . 173
Выбор по ключу . . . 174
Применение операции Or . . . 176
Примеры образцов . . . 178

Глава 5 • Принятие решений и сортировка данных . . . 187
Элемент <xsl: If> . . . 187
Элементы <xsl:choose>, <xsl:when> и <xsl:otherwise> . . . 191
Элемент <xsl:for-each> . . . 198
Сортирующие элементы . . . 201
Сортировка по нескольким критериям . . . 205
Элемент <xsl:number> . . . 206

Одноуровневая нумерация . . . 207
Нумерация на произвольных уровнях . . . 210
Многоуровневая нумерация . . . 213
Расширяемость XSLT . . . 214
Функции расширения . . . 215
Элемент <xsl:script> . . . 218
Применение функции function-available . . . 222
Внешние объекты . . . 223
Элементы расширения . . . 223
Применение функции element-available . . . 226
Элемент <xsl:fallback> . . . 226

Глава 6 • Преобразование в XML, HTML, XHTML, RTF . . . 228

Элемент <xsl:output> . . . 229
Метод вывода: HTML . . . 231
Метод вывода: XML . . . 234
Метод вывода: текст . . . 239
Вывод в формате XHTML . . . 241
Изменение структуры документа на основе входных данных . . . 244
Элемент <xsl:element>: создание новых элементов на этапе выполнения . . . 245
Элемент <xsl:attribute>: создание новых атрибутов . . . 247
Элемент <xsl :comment>: создание комментариев . . . 249
Элемент <xsl:processing-instiuction>: создание инструкций обработки . . . 250
Элемент <xsl:document>: создание нескольких выходных документов . . . 251
Элемент <xsl:namespace>: создание объявлений пространств имен . . . 254
Элемент <xsl:attribute-set>: создание наборов атрибутов . . . 254
Пропуск объявления XML и создание фрагментов XML . . . 256
Создание уникальных идентификаторов при помощи generate-Id . . . 258
Создание разделов CDATA . . . 262
Установка кодировки символов . . . 263
Режимы: форматирование в зависимости от контекста . . . 264

Глава 7 • Работа с XPath . . . 271

Изучаем XPath . . . 272
Типы данных XPath . . . 275
Наборы узлов XPath . . . 275
Числа XPath . . . 278
Строки XPath . . . 279
Логические значения XPath . . . 282
Создание путей расположения XPath . . . 284
Шаги расположения XPath, часть 1: оси . . . 285
Шаги расположения XPath, часть 2: условия узлов . . . 286
Шаги расположения XPath, часть 3: предикаты . . . 286
Применение осей XPath . . . 287
Применение оси ancestor . . . 287
Применение оси ancestor-or-self . . . 288
; Применение оси descendant . . . 289
Применение оси descendant-or-self . . . 290
Применение оси following . . . 291
Применение оси following-sibling . . . 292
Применение оси namespace . . . 293

Применение оси parent 294
Применение оси preceding 295
Применение оси preceding-sibling 297
Применение оси self 298
Примеры путей расположения 299
Сокращенный синтаксис XPath 300
Проверка выражений XPath 301
XPath 2.0 302

Глава 8 • Работа с функциями XSLT и XPath 305

Функции XSLT 306
current() 307
document() 308
element-available() 311
format-number() 311
function-available() 314
generate-id() 315
key() 318
system-property() 319
unparsed-entity-uri() 320
Функции XPath для работы с наборами узлов 323
count() 323
id() 324
last() 325
local-name() 326
name() 327
namespace-uri() 327
position() 328
Функции XPath для работы со строками 329
concat() 330
contains() 331
normalize-space() 332
starts-with() 334
string() 336
string-length() 338
substring() 339
substring-after() 341
substring-before() 341
translated) 341
Функции XPath для работы с числами 341
ceiling() 342
floor() 342
number() 344
round() 344
sum() 344
Логические функции XPath 345
boolean() 345
false() 345
lang() 345
not() 347
true() 348

Элемент <xsl:decimal-format>: создание числовых форматов 349
Новые функции в XSLT 2.0 и XPath 2.0 352

Глава 9 • Именованные шаблоны, параметры и переменные 354
Элемент <xsl:variable>: создание переменных 355
Область видимости переменной 356
Работа с переменными 359
Элемент <xsl:call-template>: применение именованных шаблонов 363
Элементы <xsl:param> и <xsl:with-param>: создание параметров 366
Рекурсивный вызов шаблонов 372
Шаблон: значение по умолчанию 374
Как задавать значения шаблона в командной строке 375
Элемент <xsl:key>: выбор по ключу 375
Элемент <xsl:document>: создание нескольких результирующих документов . . 381
Элемент <xsl:namespace-alias>: генерация таблиц стилей 385

Глава 10 • Работа с API процессоров XSLT 388
XSLT и JavaScript в Internet Explorer 389
Обработка ошибок разбора 390
Internet Explorer и динамические стили 393
Internet Explorer и участки данных XML 399
Вызов Java непосредственно из процессоров XSLT 401
Работа с API XSLT для Java 402
Взаимодействие Xalan с Java 403
Взаимодействие Saxon с Java 405
Взаимодействие процессора XSLT Oracle с Java 408
Взаимодействие XT с Java 412
Преобразование XML в реляционную базу данных 414
Применение XSLT с активными серверными страницами 418
Применение XSLT с Java Server Pages 421
Применение XSLT с сервлетами Java 422

Глава 11 • Применение XSLT для создания документов XSL-FO . . . 425
Форматирование XSL 425
Форматирующие объекты XSL-FO 426
Форматирующие свойства XSL-FO 430
Форматирование документа XML 438
Преобразование в XSL-FO при помощи таблицы стилей XSLT 440
Создание корня документа: <fo:root> 446
Создание схемы набора шаблонов: <fo:layout-master-set> 447
Создание шаблона страницы: <fo:simple-page-master> 447
Создание областей 448
Создание последовательностей страниц: <fo:page-sequence> 450
Создание объектов потока: <fo:flows> 451
Создание содержимого уровня блока: <fo:block> 452
Создание таблиц 456
Создание таблиц: <fo:table> 461
Создание столбцов таблицы: <fo:table-column> 463
Создание тел таблицы: <fo:table-body> 464
Создание строк таблицы: <fo:table-row> 465
Создание ячеек таблицы: <fo:table-cell> 466

Глава 12 • Документы XSL-FO 469
Создание списков XSL-FO 469
Создание списка: <fo:list-block> 473
Создание элементов списка: <fo:list-item> 474
Создание меток элемента списка: <fo:list-item-label> 475
Создание тел элементов списка: <fo:list-item-body> 476
Размещение текста при помощи контейнеров блоков: <fo:block-container> . . . 477
Встроенные форматирующие объекты 482
Создание встроенных областей: <fo:inline> 483
Обработка отдельных символов: <fo:character> 484
Создание номеров страниц: <fo:page-number> 486
Вставка графики: <fo:external-graphic> 487
Форматирование первых строк: <fo:initial-property-set> 489
Создание сносок: <fo:footnote> и <fo:footnote-body> 490
Создание ссылок: <fo:basic-Iink> 493
Создание колонок 496
Последовательности страниц и нумерация страниц 498

Приложение А • DTD XSLT 508

Фрагмент DTD для таблиц стилей XSLT 508

Приложение Б • Форматирующие свойства XSL-FO 516

Алфавитный указатель 534

Об авторе

Стивен Холзнер (Steven Holzner) — широко известный автор, пишущий на темы XML, в том числе и XSLT, с момента появления этих технологий. Его перу принадлежат 67 книг по программированию; всего их было продано более миллиона Экземпляров. Эти книги переведены на 16 языков мира, среди них немало бестселлеров. Стивен ранее работал редактором журнала PC Magazine; он окончил Массачусетский технологический институт (MIT) и получил степень доктора философии (Ph. D.) в институте Корнелла (Cornell), работал на факультетах Массачусетркого технологического института и института Корнелла, а в настоящее время ведет семинары на предприятиях по всей стране.

о технических редакторах

Огромный опыт технических редакторов был неоценим на протяжении всего периода работы над книгой. По мере написания книги ими велась высокопрофессиональная работа над техническим содержанием, организацией и компоновкой текста. Их отзывы были необходимым условием того, что книга будет соответствовать потребностям читателей в технической инфбрмации самого высокого качества.

Джейсон А. Басс (Jason A. Buss) — программист-аналитик, разработчик собственных программных решений для лидирующего производителя авиационного оборудования на Среднем Западе. Он женат, у него четверо детей. В свободное время любит играть с детьми, модернизировать домашний компьютер и работать на нем (сейчас он совершенствует свои знания во FreeBSD), читать, играть на гитаре и слушать живую музыку. Связаться с ним можно по адресу: jabuss@worldnet.att.net.

Дарин Бишоп (Dam'n Bishop) работает консультантом в компании Levi, Ray and Shoup, Inc., расположенной на Среднем Западе. Сейчас он занят в роли архитектора продукта в крупном проекте, использующем технологии XML. В компетенцию этой должности входят управление проектированием, выбор и исследование технологий, а также технические аспекты работы над проектом. Ранее Дарин занимался разработкой web-приложений для агентства недвижимости, приложений для программ пенсионного обеспечения и другими проектами. Главная область его интересов — связывание отдельных систем при помощи HTTP и XML.

Благодарности

Книга подобного рода — плод работы многих людей, а не одного только автора. Работа сотрудников издательства New Riders была выше всяких похвал, и я хотел бы поблагодарить замечательного исполнительного редактора Стефани Уолл (Stephanie Wall), редакторов Лори Лайонс (Lori Lyons) и Марго Кэтс (Margo Catts) и, наконец, технических редакторов Джейсона Басса и Дарина Бишопа, проверивших весь материал. Спасибо всем за ваш нелегкий труд.

Введение

Добро пожаловать в XSLT, книгу, посвященную языку преобразований расширенной таблицы стилей (Extensible Stylesheet Language Transformations). Книга писалась так, чтобы быть настолько исчерпывающей (и доступной), насколько это возможно для единственной книги по XSLT. XSLT представляет собой средство преобразования документов XML в другие документы XML или в документы совершенно другого типа; сейчас эта тематика очень популярна. Я постарался написать наиболее полную книгу по XSLT, и я считаю, что мне это удалось.

Многие книги по XSLT описывают только преобразования XML-HTML, но здесь мы собираемся пойти гораздо дальше. Все большее значение приобретают преобразования XML-XML, и в скором времени с ростом числа приложений на различных диалектах XML они станут более важны, чем преобразования XML-HTML.

В этой книге мы рассмотрим преобразования на языке XSLT из XML в XML, в HTML, в XHTML, в RTF (Rich Text Format, документы форматированного текста), в простой текст, в сценарии JavaScript, в написанные на SQL базы данных и в XSL-FO (Extensible Stylesheet Language Formatting Object, документы форматированных объектов языка расширенной таблицы стилей).

Чтобы полностью раскрыть тему XSLT, в книге описываются все возможные элементы XSLT один за другим, а также их приложения. Кроме того, мы рассмотрим все функции XSLT и XPath, которые могут быть использованы в таблицах стилей XSLT.

И вам предстоит познакомиться с сотнями работающих примеров. Без сомнений, это лучший способ разобраться в том, как работает XSLT. Если мне не удастся объяснить что-либо в тексте, вы всегда сможете посмотреть соответствующий пример, поскольку буквально для каждой подтемы в книге приводится завершенный работающий код.

Все примеры приводятся для того, чтобы их запускать. Я использовал самые популярные процессоры XSLT, их все можно бесплатно загрузить из Интернета: Xalan, XT, Saxon, процессор XSLT фирмы Oracle и даже процессор, встроенный в Microsoft Internet Explorer. В книге по мере изложения будет указываться, где в Интернете можно найти все это программное обеспечение и как с ним работать.

Аналогично написанию кода XML, написание кода XSLT не обычная и не монотонная задача: этот процесс требует искусства, страстной увлеченности и оригинальности мышления, не говоря уже о том, что он часто вызывает разочарование и раздражение. Я постараюсь вести изложение в таком духе и приложу все усилия, чтобы передать в книге мощь и замечательные возможности XSLT.

Материал книги

Книга призвана дать вам столько материала, посвященного XSLT, сколько может содержать одна книг^/Мы не только рассмотрим весь синтаксис XSLT от основ до всех специальных аспектов, но и подробно изучим различные современные способы использования XSLT.

Как я уже говорил, в книге рассмотрены сотни практических примеров, таких как осуществление преобразований XSLT на web-серверах, подключение к базам данных и использование браузеров для осуществления преобразований на лету.

Ниже перечислен ряд тем книги. Заметьте, что каждая тема включает в себя несколько подтем (слишком многочисленных, чтобы приводить их здесь):

- рекомендации по работе с XSLT 1.0;
- рабочий проект XSLT 1.1;
- требования XSLT 2.0;
- рекомендации по работе с XPATH 1.0;
- требования XPath 2.0;
- рекомендации по работе с пробной версией XSL 1.0;
- работа с процессорами Xalan, Saxon, XT и Oracle;
- создание выходных данных в форматах XML, HTML, RTF и простого текста;
- рекурсивная обработка шаблонов;
- правила шаблона по умолчанию;
- обработка пустого пространства;
- отключение вывода;
- выбор применяемого шаблона;
- создание шаблонов совпадений;
- использование предикатов в шаблонах совпадений;

- поиск совпадений элементов, дочерних и производных элементов, атрибутов и т. д.;
- поиск по идентификатору;
- синтаксис, связанный с XPath;
- условия if;
- конструкции for-each;
- реализация конструкций условного выбора;
- простая сортировка элементов;
- обработка множественного выделения;
- вложенные шаблоны;
- параметры таблицы стилей;
- упрощенные таблицы стилей;
- написание таблиц стилей для Internet Explorer;
- создание правил таблицы стилей;
- использование элементов расширения;
- использование функций расширения;
- создание элементов с результатом в виде литерала;
- использование шаблонов со значениями атрибутов;
- удаление пустого пространства;
- создание новых элементов, атрибутов, комментариев и Т. д.;
- режимы XSLT (форматирование по контексту);
- фрагменты дерева результатов;
- форматы данных XPath;
- выражения XPath;
- все функции XSLT и XPath;
- создание наборов атрибутов;
- вызов именованных шаблонов;
- единственные и множественные ключи;
- создание документов с множественным выводом;
- использование процессоров Oracle, Saxon, XML4Java, XT в коде Java при помощи вызовов API;
- использование JavaScript для XSLT в Internet Explorer;
- Formatting Objects (форматирующие объекты) XSL-FO;
- Formatting Properties (форматирующие свойства) XSL-FO;
- преобразования XSLT нд стороне сервера при помощи Java Server Pages (JSP), Active Server Pages (ASP) и сервлетов Java.

В книге полностью описаны официальные спецификации XSLT, установленные консорциумом World Wide Web Consortium (W3C: рекомендации по работе с XSLT 1.0, рабочий проект XSLT 1.1 и требования XSLT 2.0. Объяснение этих терминов приводятся в главе 1). Как мы рассмотрим в главе 1, консорциум W3C объявил, что рабочий проект (working draft) XSLT 1.1 не будет иметь статуса официальных рекомендаций, поскольку к выпуску готовится XSLT 2.0. Сейчас XSLT 2.0 имеет статус требований (requirements), который предшествует статусу рабочего проекта. Тем не менее, поскольку содержание рабочего проекта XSLT 1.1 будет включено в XSLT 2.0 и поскольку во многих процессорах XSLT будет реализована поддержка XSLT 1.1, я полностью описываю в этой книге и рабочий проект XSLT 1.1, специально отмечая относящийся к этому материал.

При создании таблицы стилей XSLT необходимо указать версию XSLT. Поскольку версия 1.1 никогда не выйдет из стадии рабочего проекта и не будет официально выпущена, я придерживаюсь в примерах этой книги версии 1.0, за исключением обсуждения рабочего проекта XSLT 1.1, — в этом случае я явно указываю версию 1.1.

Кроме того, сегодня в Интернете можно найти огромное количество материала по XSLT, так что я дополняю книгу десятками адресов URL таких ресурсов, в том числе и адресами, с которых можно бесплатно загрузить используемое в книге программное обеспечение. Тут, однако, стоит отметить одно затруднение: адреса в Интернете меняются очень часто, поэтому не удивляйтесь, если некоторые из приведенных в книге адресов изменятся к тому моменту, когда вы попробуете их использовать.

В Интернете по адресу издательства New Riders, http://www.newriders.com, можно найти полные тексты всех листингов книги.

Для кого предназначена эта книга

Эта книга рассчитана на всех, кто хочет изучать XSLT и способы его использования на практике. Единственное требование — вы должны представлять, как создавать документы на XML и HTML. Нет необходимости быть экспертом в XML, однако знание соответствующих тем будет полезно[1]. Больше никакой специальной подготовки не требуется.

Если вы хотите выполнять приведенные в книге примеры в процессорах XSLT, которые загружаются бесплатно, в вашей системе должна быть установлена Java (это бесплатно), более подробно мы рассмотрим эту тему в главе 1.

На каком уровне написана книга

Эта книга написана на нескольких различных уровнях, от начального до продвинутого, поскольку область применения XSLT очень широка. Я постараюсь охватить

[1] **См. выпущенные издательством книги «XML и Java 2, библиотека программиста» и «XML, справочник». — *Примеч. ред.***

так много материала, касающегося XSLT, как это только возможно, поэтому почти весь материал, от начального до профессионального уровня, изложен в книге. Как уже говорилось, я считаю, что у вас есть представление о XML и HTML

Навыков программирования при чтении книги не требуется — за исключением главы 10, специально посвященной преимуществам использования процессоров XSLT вместе с Java. Если вы хотите изучить эту главу, вам необходим некоторый опыт программирования на Java.

Поскольку сейчас вариантов использования XSLT очень много, в книге упоминаются несколько программных пакетов; их все можно бесплатно загрузить из Интернета (далее я укажу, откуда).

Используемые соглашения

Примите во внимание ряд используемых в книге соглашений.

Самое важное правило: когда я добавляю новый раздел кода, я помечаю обсуждаемые строки кода полужирным шрифтом:

```
<xsl:tempi ate match-"PLANET">
    <xsl:value-of select-"NAME"/>
    <xsl:1f test-"pos1tion()!-last()">, </xsl:1f>
    <xsl:1f test-"pos1tion()-last()-1">and </xsl:1f>
    <xsl:1f test-"pos1t1on()-last()">.</xsl:1f>
</xsl:tempiate>
```

Отметим также, что нумерованные листинги можно загрузить с сайта издательства: http://www.newriders.com. Загрузив этот код, вы сэкономите много времени, если вы хотите выполнить или изменить код примеров книги.

В тех местах, на которые требуется обратить особое внимание, или где требуется дополнительная информация для обсуждения, я буду помещать врезку, например:

УСТАНОВКА НАЧАЛЬНОГО НОМЕРА СТРАНИЦЫ

Еще один совет: чтобы установить начальный номер страницы, можно воспользоваться свойством initial-page-number элемента <fo:page-sequence>. Это даст вам возможность, например, отдельно отформатировать главы, начиная каждую главу с нужного номера страницы.

Теперь можно начинать чтение. Предположим, у вас появятся замечания — я приветствую, если вы пришлете их мне или в издательство New Riders. Эта книга задумывалась как новый стандарт в охвате темы XSLT, более полный и более доступный, чем раньше. Пожалуйста, не стесняйтесь уведомлять меня о всех возможных улучшениях. Если вы считаете, что в книге чего-либо недостает — я добавлю этот материал, поскольку я хочу, чтобы эта книга всегда оставалась бестселлером.

От издательства

Ваши замечания, предложения, вопросы, касающиеся русского издания этой книги, отправляйте по адресу электронной почты comp@piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На web-сайте издательства http://www.piter.com вы найдете подробную информацию о наших книгах.

1

Основы XSLT

Добро пожаловать в мир языка преобразований расширенной таблицы стилей, XSLT (Extensible Stylesheet Language Transformations). Эта книга послужит вам путеводителем в огромном мире XSLT, который каждую минуту расширяется непредсказуемым образом. Мы хотим, чтобы этот мир стал и вашим миром. Нам нужно охватить весьма большой материал, поскольку в наши дни XSLT используется в очень интересных местах и очень интересными способами. В этой книге вы увидите, как это все работает.

Собственно XSLT представляет собой средство обработки и форматирования содержимого документов XML. XML уже стал очень популярным, теперь настала очередь XSLT. XML дает вам возможность структурировать данные в документах, а XSLT позволяет работать с содержимым документов XML — оперировать содержимым и создавать другие документы (например, при сортировке XML записей базы данных сотрудников или при сохранении данных в документ HTML, а также при детальном форматировании данных).

С содержимым документов XML можно работать, написав собственную программу, реализующую интерфейс с приложениями разборщика (parser) XML, однако при этом приходится писать код программы самостоятельно. При помощи XSLT вы можете выполнять задачи подобного же рода, не прибегая к программированию. Вместо того чтобы писать код обработки содержимого документов XML на Java, Visual Basic или C++, можно просто указать при помощи XSLT, что вы хотите сделать, и процессор XSLT сделает все остальное. Именно для этих целей и предназначен XSLT, и в мире XML он выходит на ключевые позиции.

XSL = XSLT + XSL-FO

Сам XSLT в действительности является частью более крупной спецификации — расширенного языка таблиц стилей, Extensible Stylesheet Language, или XSL. XSL предназначен для задания точного, до миллиметра, формата документов.

Форматирующая часть XSL, представляющая гораздо более крупную спецификацию, чем XSLT, основана на специальных форматирующих объектах (formatting objects) — эта часть XSL часто называется XSL-FO (или XSL:FO, или XSLFO). XSL-FO — сложная тема, поскольку задание стилей документов при помощи форматирующих объектов может оказаться весьма запутанным процессом. Фактически XSLT изначально был добавлен в XSL для того, чтобы проще осуществлять преобразование документов XML в документы, основанные на форматирующих объектах XSL-FO.

Эта книга посвящена XSLT, но рассматривается также и введение в XSL-FO, в том числе способ использования XSLT для преобразования документов в форму XSL-FO; в конце концов, XSLT впервые был представлен для упрощения работы с XSL-FO. В начале данной главы будет приведен обзор как XSLT, так и XSL-FO.

Краткая историческая справка

XSL был создан консорциумом World Wide Web Consortium (W3C , www.w3.org) — объединением групп, первоначально возглавшимся Тимом Бернерс-Ли (Tim Berners-Lee). W3C — это комитет, выпускающий спецификации, — такие, как спецификация для XSL, используемая в данной книге. Именно спецификации делают XML и XSL тем, чем они являются.

W3C И ЯЗЫКИ СТИЛЕЙ

Вы можете прочитать об истории работы комитета W3C с языками стилей по адресу www.w3.org/Style/History. Интересно посмотреть, какая объемная работа была проделана и как много языков стилей сменилось за это время.

W3C первоначально, в 1980-х, разработал «дедушку» XML, SGML (Standard Generalized Markup Language, стандартный обобщенный язык разметки), однако он был слишком сложен для того, чтобы завоевать широкую популярность, и в действительности XML (как и HTML) представляет собой упрощенную версию SGML. W3C также создал для работы совместно с SGML язык стилей DSSSL (Document Style Semantics and Specification Language, язык семантики стиля и спецификации документа) — и так же, как XML был произведен от SGML, XSL основан на исходном DSSSL. Как утверждает W3C: «Модель, используемая XSL для отображения документов на экране, базируется на многих годах работы над сложным языком стилей по стандарту ISO, который называется Document Style Semantics and Specification Language (DSSSL)».

Однако исходная часть XSL, то есть XSL-FO, все равно оказалась не настолько простой, чтобы найти широкое распространение, поэтому XSLT был представлен как средство упрощения преобразования документов XML в форму XSL-FO. Как оказалось, именно XSLT был взят на вооружение, поскольку он представляет собой законченный язык преобразований, позволяющий работать с содержимым документов XML без написания программного кода, преобразовывать эти документы в другие документы XML, формат HTML или другие основанные на текстах форматы. Большой успех XSLT удивил даже W3C.

Преобразования XSLT—XSL

XSLT позволяет работать непосредственно с содержимым документов XML. Например, у вас может быть огромный документ XML, содержащий всю бейсбольную статистику за последний сезон, однако вас может интересовать только статистика для питчеров. Чтобы извлечь только эти данные, можно написать программу на Java, Visual Basic или C++, которая будет работать с разборщиками XML. Разборщики представляют собой специальные программные пакеты, читающие документы XML и передающие все данные документа последовательно в ваш код. Затем можно создать новый документ XML, pitchers.xml, содержащий только данные о питчерах.

Этот способ работает, однако при этом необходимо довольно много программировать и тратить много времени, в том числе и на тестирование. XSLT разрабатывался для решения подобных проблем. XSLT могут читать процессоры XSLT, обрабатывающие документы XML: все, что вам нужно сделать, — это создать таблицу стилей XSLT, определяющую правила, которые вам требуется применить для преобразования одного документа в другой. Никакого программирования не понадобится, что является несомненным достоинством в глазах многих людей, даже опытных программистов. В примере с бейсболом вам нужно написать таблицу стилей XSLT, определяющую необходимые действия, и предоставить процессору XSLT довершить все остальное.

Помимо преобразования одного документа XML в другой документ XML, можно также преобразовывать документы XML в документы разных типов — таких, как HTML, документы форматированного текста (RTF), документы, использующие XSL-FO и другие. Можно также преобразовать документы XML в иные основанные на XML языки — такие, как MathML, MusicML, VML, XHTML и другие — все это можно осуществить, не прибегая к программированию.

Во многих случаях язык XSLT может работать аналогично языку баз данных, такому как SQL (Structured Queiy Language, язык структурированных запросов, — известный язык доступа к базам данных), поскольку он позволяет извлекать требуемые данные из документов XML во многом похоже на то, как инструкция SQL извлекает данные из базы данных. Иногда даже говорят о XSLT как о SQL в Web, и если вы знакомы с SQL, это даст вам некоторое представление о безграничных возможностях XSLT. Например, при помощи таблицы стилей XSLT можно извлечь подмножество данных из документа XML, создать оглавление для большого документа, найти все элементы, удовлетворяющие определенному условию (например, поставщиков с определенным индексом) и т. д. И все это — за один шаг!

XSL-FO: форматирующие объекты XSL

Другая часть XSL — это XSL-FO, форматирующая часть языка XSL, с которой мы также познакомимся в этой книге. При помощи XSL-FO можно определить способ представления данных в документах XML, вплоть до размера полей, шрифтов, выравнивания, размера заголовка и подзаголовка, а также ширины

страницы. При форматировании документа XML приходится думать о множестве вещей, и, соответственно, XSL-FO значительно объемистее, чем XSLT.

С другой стороны, из-за излишней сложности XSL-FO не очень популярен, и его нельзя сравнивать с XSLT в этом отношении. Существует не так много программ, поддерживающих XSL-FO, и ни одна из них не реализует достаточно полное приближение к стандарту. Так же как самый распространенный способ применения XSLT — это преобразование XML в HTML, самый распространенный способ применения XSL-FO — преобразование XML в текст в формате PDF (Portable Data Format, переносимый формат данных), используемый в Adobe Acrobat. Пример такого преобразования приведен в конце этой главы, а также в главе 11.

Спецификации W3C

W3C выпускает спецификации как для XML, так и для XSL, и именно с ними мы будем работать на протяжении всей книги. Спецификации W3C не называются стандартами, поскольку, по международным соглашениям, стандарты могут создаваться только уполномоченными государством комитетами. Вместо этого W3C начинает с выпуска *требований* (requirements) для новой спецификации. Требования представляют собой список целей и обзоров спецификации, но сама спецификация на этот момент еще не написана. Далее W3C выпускает спецификации: сначала в виде *рабочих проектов* (working drafts), которые могут комментировать все заинтересованные лица, затем в виде *рекомендаций-кандидатов* (candidate recommendations), которые еще могут быть изменены; и, наконец, в виде окончательных *рекомендаций* (recommendations), которые уже нельзя изменять.

В следующем списке приведены имеющие отношения к XSLT спецификации W3C, которые мы будем использовать в книге, и адреса, по которым их можно найти:

- полная рекомендация-кандидат XSL www.w3.org/TR/xsl/, Это большой документ, определяющий все аспекты XSL;
- рекомендация XSL Transformations 1.0 по адресу www.w3.org/TR/xslt. Предназначение XSLT состоит в преобразовании содержимого документов XML в другие документы, и именно поэтому XSL стал таким популярным;
- рабочий проект XSLT 1.1 по адресу www.w3.org/TR/xsltll. Это рабочий проект XSLT 1.1, который не будет далее разрабатываться до рекомендации — W3C планирует добавить всю функциональность XSLT 1.1 в XSLT 2.0;
- требования XSLT 2.0 по адресу www.w3.org/TR/xslt20req. W3C выпустил группу целей для XSLT 2.0, включая дополнительную поддержку схем XML;
- спецификация XPath 1.0 по адресу www.w3.orgДR/xpath. XPath используется для нахождения и указания на определенные разделы и элементы в документах XML так, чтобы можно было с ними работать;
- требования XPath 2.0 по адресу www.w3.org/TR/xpath20req. XPath в данный момент обновляется — добавляется дополнительная поддержка XSLT 2.0.

Версии XSLT

Спецификации XSLT разрабатывались значительно активнее, чем спецификации для всего XSL. Рекомендация XSLT 1.0 была окончательно принята 16 ноября 1999 г., и эта версия является сегодня основной версией XSLT.

Затем появился рабочий проект XSLT 1.1 и, хотя первоначально он рассматривался в качестве пролога новой рекомендации, ряд сотрудников W3C начал работать над XSLT 2.0 — и через некоторое время W3C решил прекратить работу над рекомендацией XSLT 1.1. Это означает, что рабочий проект XSLT 1.1 не будет развиваться далее — он навсегда останется в виде рабочего проекта и никогда не станет рекомендацией. Иными словами, не будет выпущено официальной версии 1.1 для XSLT.

Однако консорциум W3C также утверждает, что он планирует включить большую часть того, что было сделано в рабочем проекте XSLT 1.1, в XSLT 2.0, и поэтому в данной книге я кратко рассмотрю рабочий проект XSLT 1.1. Я обязательно отмечу материал как «только для рабочего проекта XSLT 1.1* при обсуждении нового материала, представленного в рабочем проекте XSLT 1.1.

Ниже перечислены изменения в XSLT 1.0, сделанные в рабочем проекте XSLT 1.1; заметьте, что этот список приведен здесь только в качестве справочного материала, так как большая часть материала вряд ли пока что-нибудь для вас значит:

- исключен поддерживаемый в XSLT 1.0 тип данных фрагмента результирующего дерева;
- метод вывода больше не может произвольно добавлять узлы пространства имен, поскольку процесс установки пространства имен применяется автоматически;
- была добавлена поддержка для XML Base;
- теперь поддерживаются несколько выходных документов при помощи элемента <xsl:document>;
- элемент <xsl:apply-imports> теперь может иметь параметры;
- функции расширения теперь можно определять при помощи функции <xsl:script>;
- функции расширения теперь могут возвращать внешние объекты, не соответствующие никаким типам данных XPath.

В этой книге рассматривается рекомендация XSLT 1.0, а также рабочий проект XSLT 1.1. В развитие данной темы W3C и выпустил требования для XSLT 2.0, которые также рассматриваются в книге под именем XSLT 2.0. В следующем списке приведен обзор целей XSLT 2.0:

- добавить дополнительную поддержку для работы с содержимым схемы XML при помощи XSLT;
- упростить работу со строками;
- упростить работу с XSLT;

- улучшить поддержку различных языков;
- сохранить обратную совместимость с XSLT 1.0;
- поддерживать повышенную эффективность процессора.

Хотя XSLT 2.0 еще некоторое время не будет выпущен в окончательном варианте, я рассмотрю все, что о нем известно, при обсуждении имеющих к нему отношение тем. Например, разработанный W3C последователь HTML -т это основанный на XML язык XHTML. В XSLT 1.0 и в рабочем проекте XSLT 1.1 нет прямой поддержки преобразований из XML в XHTML, поэтому нам придется создать это преобразование с нуля. Однако такая поддержка входит в состав XSLT 2.0, и я отмечу этот факт при обсуждении XHTML.

Таким образом, мы рассмотрели краткий обзор тематики; давайте перейдем к работе. XSL предназначен для работы с документами XML, поэтому сначала я хочу рассмотреть структуру документов XML. Вы будете работать с документами XML, но сами таблицы стилей XSL также являются документами XML, что следует помнить при их написании. В этой книге считается, что вы обладаете знаниями и XML, и HTML.

Документы XML

Вы должны понимать, как работают документы,XML, поэтому проверьте по материалам этого раздела, готовы ли вы к работе. Давайте посмотрим на примере документа XML:

```
<?xml version-"1.Л" encoding="UTF-8"?>
<D0CUMENT>
    <GREETING>
        Hello From XML                                                    *
    </GREETING>
    <MESSAGE>
        Welcome to the wild and woolly world of XML.
    </MESSAGE>
</D0CUMENT>                                                               *
```

Вот как работает этот документ: я начал с инструкции обработки XML <?xml version-" 1.0" encoding="UTF-8"?> (все инструкции обработки XML начинаются с <? и заканчиваются ?>), означающей, что используется XML версии 1.0, единственная определенная сейчас версия, и кодировка символов UTF-8, то есть используется восьмибитная сжатая версия Unicode:

```
<?xml version-''Г.О" encoding-"UTF-8"?>
<D0CUMENT>
    <GREETING>
        Hello From XML
    «/GREETING* ◆>
    <MESSAGE>
        Welcome to the wild and woolly world of XML.
    </MESSAGE>      -                              Л
«/DOCUMENT»                                  •;,?>/   yTt          v   •
```

Затем я создаю новый тег (tag) с именем <DOCUMENT>. Для тега можно использовать любое имя, не обязательно DOCUMENT; необходимо только, чтобы имя начиналось с буквы или символа подчеркивания (_), а последующими символами были буквы, цифры, символы подчеркивания, точки (.) или дефисы (-), но не пробелы. В XML теги всегда начинаются с < и заканчиваются >.

Документы XML образованы из элементов XML; последние начинаются с открывающего тега, такого как <DOCUMENT> (за которым следует содержимое (content) элемента, если оно есть, — например, текст или другие элементы), и завершаются закрывающим тегом, парным тегу открытия (он начинается с символов </ — например, </DOCUMENT>). Весь документ целиком, за исключением инструкций обработки, заключается в один элемент, называемый корневым (root) элементом; в нашем случае это элемент <DOCUMENT>:

```
<?xml version-"1.0" encoding="UTF-8"?>
<DOCUMENT>
    .
    .
    .
«/DOCUMENT»
```

Теперь я добавлю новый элемент, <GREETING>, в который заключено текстовое содержимое (здесь «Hello From XML», привет от XML) этого XML-документа:

```
<?xml version-"1.0" encoding-"UTF-8"?>
<DOCUMENT>
    <GREETING>
        Hello From XML
    </GREETING>
    .
    .
    .
</DOCUMENT>
```

После этого я могу добавить еще один элемент, <MESSAGE>, также образующий оболочку над текстовым содержимым:

```
<?xml version-"1.0" encoding-"UTF-8"?>
<DOCUMENT>
    <GREETING>
        Hello From XML
    </GREETING>
    <MESSAGE>
        Welcome to the wild and woolly world of XML.
    </MESSAGE>
</DOCUMENT>
```

Сейчас корневой элемент <DOCUMENT> содержит два элемента — «GREETING» и «MESSAGE», в каждом из которых имеется текст. Таким образом, я создал новый документ XML.

Однако следует добавить еще кое-что: документы XML могут быть также *хорошо сформированными* (well-formed) и *допустимыми* (valid).

Хорошо сформированные документы XML

Для того чтобы быть хорошо сформированным, документ XML должен следовать правилам синтаксиса, установленным для XML консорциумом W3C в рекомендации XML 1.0 (которую можно найти по адресу \лм\л/.\л/3.огдДЯ/ИЕС-хт1). Неформально «хорошо сформированный» означает главным образом то, что документ должен содержать один или более элементов, и один из них, корневой, обязан включать в себя все остальные элементы. Кроме того, для каждого элемента должны соблюдаться правила вложенности. Например, следующий документ не будет хорошо сформированным, потому что закрывающий тег </GREETING> встречается после открывающегося тега <MESSAGE> для следующего элемента:

```
<?xml version-"1.0" encod1ng-"UTF-8"?>
<D0CUMENT>
    <GREETING>
        Hello From XML
    <MESSAGE>
    «/GREETING*
        Welcome to the wild and woolly world of XML.
    </MESSAGE>
</D0CUMENT>
```

Допустимые документы XML

Большинство браузеров XML проверяют документы на то, являются ли они хорошо сформированными. Некоторые браузеры могут также проверить, допустим ли документ. Документ XML допустим (valid), если с ним связаны *объявление* типа документа (*DTD* — Document Type Declaration) или схема XML, и если документ удовлетворяет этим DTD или схеме. То есть DTD или схема задает набор правил для внутренней целостности самого документа, и если браузер может подтвердить, что документ удовлетворяет этим правилам, он является допустимым.

XML-схемы завоевывают популярность, и значительно большая поддержка схем присутствует в XSLT 2.0 (фактически поддержка XML-схем — одна из побудительных сил к переходу на XSLT 2.0), однако объявления DTD все еще остаются наиболее часто используемым средством проверки на допустимость. Объявления DTD могут быть сохранены в отдельном файле или в самом документе в элементе <!D0CTYPE>. Ниже этот элемент добавлен в разработанный нами пример:

```
<?xml version-"1.0" encoding-"UTF-8"?>
<?xml-stylesheet type="text/css" href-"first css"?>
<!D0CTYPE DOCUMENT [
    <!ELEMENT DOCUMENT (GREETING. MESSAGE)»
    <!ELEMENT GREETING (#PCDATA)>
    <!ELEMENT MESSAGE (#PCDATA)>
J>
<D0CUMENT>
    <GREETING>
        Hello From XML
    </GREETlNG>
    <MESSAGE>
```

```
        Welcome to the wild and woolly world of XML.
      </MESSAGE>
    </DOCUMENT»
```

В этой книге не рассматриваются объявления DTD, но из них видно, что элемент <DOCUMENT> — корневой, а элементы <GREETING> и <MESSAGE> могут, во-первых, находиться внутри него, а во-вторых — содержать текст.

[4] В документах XML могут присутствовать все виды иерархий, когда один элемент заключен в другой элемент и так далее на много уровней вложенности. Элементам можно также задавать атрибуты, например: <CIRCLE C0L0R«"btue">, где атрибут COLOR содержит значение "blue". При помощи таких атрибутов удобно хранить дополнительные данные об элементах. Еще в документы XML разрешается включать комментарии, поясняющие определенные элементы текстом внутри тегов <!— и —>.

Ниже, в листинге 1.1, приведен пример XML-документа planets.xml, в котором эти возможности используются для хранения данных о планетах Меркурии (Mercury), Венере (Venus) и Земле (Earth) — таких как масса, длительность дня, плотность, расстояние от Солнца и т. д. Мы будем работать с этим документом во всей книге, поскольку в нем в компактной форме содержится большая часть возможностей XML, которые могут вам потребоваться.

Листинг 1.1. planets.xml

```
<?xml version-"1.0"?>
<PLANETS>
    <PLANET>
        <NAME>Mercury</NAME>
        <MASS UNITS-"(Earth - 1)">.0553</MASS>
        <DAY UNITS="days">58.65</DAY>
        <RADIUS UNITS-"miles">1516</RADIUS>
        «DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
        «DISTANCE UNlTS-"million mi1es"»43.4«/DISTANCE»«!--B перигелии--»
    «/PLANET»

    «PLANET»
        <NAME»Venus«/NAME»
        «MASS UNITS-"(Earth - 1) "».815</MASS>
        «DAY UNITS-"days"»116.75</DAY»
        «RADIUS UNITS«"mnes">3716</RADIUS>
        «DENSITY UNITS-"(Earth - 1)"» 943</DENSITY>
        «DISTANCE UNITS-"million miles"»66.8</DISTANCE><!-B перигелии-»
    «/PLANET»

    «PLANET»
        <NAME»Ea rth«/NAME»
        «MASS UNITS-"(Earth - 1)">1«/MASS>
        «DAY UNITS-"days"»l</DAY>
        «RADIUS UNITS"miles">2107«/RADIUS>
        «DENSITY UNITS-"(Earth - 1)"»1«/DENSITY>
        «DISTANCE UNITS-"milli6n miles''»128.4«/DISTANCE>«!-B перигелии-»
    «/PLANET»
«/PLANETS»
```

Вам нужно также усвоить ряд определений XML, принятых в этой книге:

- CDATA. Простые символьные данные (то есть текст, не содержащий какой-либо разметки);
- ID. Корректное имя XML, которое должно быть уникальным (то есть не использоваться в каких-либо других атрибутах типа ID);
- IDREF. Содержит значение атрибута ID некоторого элемента, как правило, отличного от элемента, с которым связан текущий элемент;
- IDREFS. Несколько идентификаторов (ID) элементов, разделенных пробелами;
- NAME Символ. Буква, разряд, точка, дефис, символ подчеркивания или двоеточие;
- NAME. Имя XML, котрое должно начинаться с буквы, символа подчеркивания или двоеточия, за которыми, возможно, следуют дополнительные символы имени;
- NAMES. Список имен, разделенных символом-разделителем;
- NMTOKEN. Лексема, образованная из одной или более букв, цифр, дефисов, символов подчеркивания, двоеточий и точек;
- NMTOKENS. Несколько корректных имен XML в списке, разделенных символом-разделителем;
- NOTATION. Название нотации (которое должно быть объявлено в DTD);
- PCDATA. Разобранные символьные данные. PCDATA не содержит какой-либо разметки, и любые ссылки на сущности уже были в PCDATA раскрыты.

Теперь у нас есть представление о документах XML, в том числе о том, что такое хорошо сформированный и допустимый документ. Если вы еще не чувствуете себя достаточно уверенно в этом материале, прочитайте другую книгу по этой теме. Вы можете также просмотреть на некоторые ресурсы XML в Web:

- http://www.w3c.org/xml. Основной web-узел консорциума World Wide Web Consortium, посвященный XML, начальная точка для всех аспектов XML;
- http://www.w3.org/XML/1999/XML-in-10-points, «XML за 10 шагов» (в действительности только семь) — обзор XML;
- Ьйр://^м^3.огдДР/РЕС-хт1 Официальная рекомендация W3C для XML 1.0, текущая (и единственная) версия. Не слишком просто для чтения;
- http://www.w3.org/TR/xml-stylesheet/. Все о работе таблиц стилей и XML;
- http://www.w3.org/TR/REC-xml-names/. Все о пространствах имен XML;
- http://www.w3.org/XML/Activity.html. Обзор текущей деятельности по XML в W3C;
- http://www.w3.orgДR/xmlschema-0/, http://www.w3.orgДR/xmlschema-l/ и http://www.w3.org/TR/xmlschema-2/. XML схемы, альтернатива объявлениям DTD;

- http://www.w3.org/TR/xlink/. Спецификация XLinks;
- http://www.w3.org/TR/xptr. Спецификация XPointers;
- http://www.w3.org/TR/xhtml1/. Спецификация XHTML 1.0;
- http://www.w3.org/TR/xhtml11/, Спецификация XHTML 1.1;
- http://www.w3.org/D0M/. W3C Объектная модель документа, DOM (Document Object Model).

Теперь мы научились создавать документы XML. Каковы они в наглядном виде?

Как XML выглядит в браузере?

Можно непосредственно отображать XML-документы при помощи браузера, такого, как Microsoft Internet Explorer версии 5 или более поздней. Например, если сохранить созданный нами XML-документ под именем greeting.xml и открыть его в Internet Explorer, он будет выглядеть так, как показано на рис. 1.1.



Рис. 1.1. XML-документ в Internet Explorer

На рис. 1.1 можно видеть документ XML полностью. Нет никакого форматирования, документ XML выглядит в Internet Explorer точно так же, как если бы

вы вывели его на печать. (Для отображения экрана, показанного на рис. 1.1, Internet Explorer использовал свою таблицу стилей по умолчанию. Она преобразует XML в формат Dynamic HTML, с которым Internet Explorer умеет работать.) А что, если нам требуется представить данные в другом виде? Допустим, мы захотим представить данные из planets.xml в документе HTML в виде HTML-таблицы?

Тут-то нам и понадобятся преобразования XSLT. В этой главе мы с ними познакомимся, а в конце главы рассмотрим другую сторону XSL — XSL-FO.

XSLT-преобразования

XSLT обладает большими возможностями для оперирования данными в документах XML Например, при помощи таблицы стилей (stylesheet) XSLT я мог бы отформатировать данные из planets.xml в таблицу HTML. *Таблицы стилей* содержат правила, установленные для преобразования XML-документа, и большая часть книги посвящена созданию таблиц стилей и объяснению того, как они работают. Вот как выглядит таблица стилей XSLT planets.xsl (листинг 1.2), преобразующая данные из planets.xml в таблицу HTML (мы проанализируем ее в главе 2).

Листинг 1.2. planets.xsl

```
<?xml version-"l.Q"?>
<xsl:stylesheet version-"1 0"
xmlns:xsl-"http://www.w3.org/1999/XSl/Transform">

    <xs1:tempiate match-"/PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <B0DY>
                <H1>
                    The Planets Table
                </Hl>
                <TABLE BORDER-"2">
                    <TR>
                        <TD>Name</TD>
                        <TD>Mass</TD>
                        <TD>Radius</TD>
                        <TD>Day</TD>
                    </TR>
                    <xsl:apply-templates/>
                </TABLE>
            </B0DY>
        </HTML>
    </xsl tempiate>

    <xsltemplate match-"PLANET">
       <TR>
          <TD><xal:value-of select-"NAME "7></TD>
          <r0><xs1 ;apply-templates select-"MASS "/x/TO
```

продолжение &

Листинг 1.2 (продолжение)

```
        <TDxxsl:apply-templates select-"RADIUS"/></TD>
        <TDxxsl:apply-templates se1ect-"DAV'7></TD>
      </TR>
  </xsl tempiate>

   <xsl:template match-"MASS">
       <xsl:value-of select»"."/>
       <xsl:text> </xsl:text>
       <xsl:value-of select~"@UNITS"/>
   </xsl:tempiate>

   <xsl:tempiate match-"RAOIUS">
       <xsl:value-of select-".7>
       <xsl:text> </xsl:text>
       <xsl:value-of select-"@UNITS"/>
   </xsl:tempiate>
...
   <xsl:tempi ate match-"DAY">
       <xsl:value-of select-"."/>
       <xsl:text> </xsl:text>
       <xsl:value-of select-"@UNITS'7>
   </xsl:tempiate>

</xsl:stylesheet»
```

Как видите, эта таблица стилей XSLT выглядит как XML-документ — и для этого есть основания, потому что она им и является. Все таблицы стилей XSLT являются также и документами XML, а раз так, то они должны быть хорошо форматированными документами XML С этими двумя документами — ptanets.xml (листингом 1.1) и связанной с ним таблицей стилей, planets.xsl (листинг 1.2) — мы будем работать на протяжении всей книги, рассматривая различные возможные преобразования XSLT.

Как вы свяжете эту таблицу стилей с XML-документом planets.xml? Как мы увидим в следующей главе, одним из способов это сделать является инструкция обработки XML <?xml-stylesheet?>, использующая два атрибута. Первый атрибут — type, который следует установить в «text/xml», чтобы указать, что используется таблица стилей XSLT. (Чтобы использовать таблицы стилей другого типа — каскадирующие таблицы стилей (CSS, cascading stylesheets), обычно использующиеся с HTML — следует задать «text/ess».) Второй атрибут — href, которому следует присвоить значение URI (вспомните, что XML использует не адреса URL, а идентификаторы URI, Uniform Resource Identifier) таблицы стилей:

```
<?xml version="1.0"?>
<?xml-stylesheet type-"text/xml" href»"planets.xsl"?>
<PLANETS>

    <PLANET>
        <NAME>Mercury</NAME>
        <MASS UNITS?"(Earth - 1}">.0553</MASS>
        <DAY UNITS-"days">58.65</DAY>
        <RADIUS UNITS-"miles">1516</RADIU$>
        <DENSITY UNITS-"(Earth - 1)B>.983</DENSITY>
        «DISTANCE UNITS-"mi 111on roiles">43.4</DISTANCE><!--В перигелии~>
```

```
</PLANET>
```

Теперь при помощи *процессора* (processor) XSLT можно применить таблицу planets.xsl к planets.xml и создать новый документ, planets.html. Процессор XSLT создает новый файл planets.html, который вы можете увидеть на рис. 1.2.

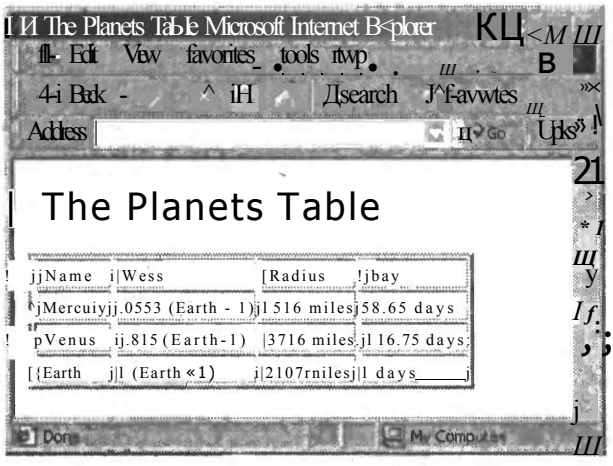


Рис. 1.2. HTML-документ, созданный процессором XSLT

Как можно видеть на рис. 1.2, процессор XSLT считывает данные из planets.xml, применяет к ним правила из planets.xsl и создает HTML-таблицу в planets.html. Это наш первый пример XSLT-преобразования.

Что здесь на самом деле происходит? У нас были XML-документ, planets.xml, и таблица стилей XSLT, planets.xsl. Но каким образом они объединились для создания planets.html?

Что нужно для XSLT-преобразования

Для осуществления XSLT-преобразования, такого как преобразование planets.xml в planets.html, необходим XSLT-процессор. Для преобразований документов XML можно использовать XSLT тремя способами:

- используя отдельные программы, называемые процессорами XSLT. Существует несколько программ, осуществляющих XSLT-преобразования, обычно основанных на Java, — с некоторыми из них мы познакомимся в этой главе;
- на стороне клиента. Клиентская программа, например браузер, может осуществлять преобразование, считывая таблицу стилей, заданную в инструкции обработки <?xml-stylesheet?>. В частности, Internet Explorer может осуществлять преобразования подобного рода;

- на стороне сервера. Серверная программа, например сервлет Java, может преобразовывать документ автоматически при помощи таблицы стилей и отправлять его клиенту.

В книге мы рассмотрим все три способа осуществления XSLT-преобразований. Уже в этой главе будет предоставлен обзор всех этих трех различных способов.

Использование отдельных процессоров XSLT

Отдельные процессоры XSLT — один из самых распространенных способов провести XSLT-преобразования. Существует весьма много таких процессоров, хотя и не все из них могут обрабатывать все возможные таблицы стилей XSLT. Для того, чтобы использовать процессор XSLT, достаточно запустить его из командной строки (в окне DOS в Windows), задать имя исходного документа XML, имя используемой таблицы стилей и имя документа, который вы хотите создать.

Ниже приведен список некоторых доступных в Интернете процессоров XSLT в алфавитном порядке — большинство (но не все) из них бесплатны:

- 4XSLT, http://Fourthought.com/4Suite/4XSLT. XSLT-процессор для Python;
- EZ/X, http://www.activated.com/products/products.html. Пакет Java одновременно для разбора XML и обработки XSLT;
- iXSLT, http://www.infoteria.com/en/contents/download/index.htmL Процессор XSLT в командной строке;
- Koala XSL Engine, http://www.inria.fr/koala/XML/xslProcessor. XSLT-процессор для Java, использующий простой API для XML, Simple API for XML (SAX 1.0) и API объектной модели документов, Document Object Model (DOM 1.0) API;
- LotusXSL, http://www.alphaworks.ibm.com/tech/LotusXSL. LotusXSL фирмы IBM реализует XSLT-процессор на Java и может взаимодействовать с интерфейсами API, удовлетворяющими спецификации Document Object Model (DOM) Level 1 Specification. Известный процессор XSLT, но сейчас он вытесняется Xalan 2.0;
- MDC-XSL, http://mdc-xsl.sourceforge.net. Процессор XSLT на C++, его можно использовать как отдельную программу;
- Microsoft XML Parser, http://msdn.microsoft.com/downloads/ webtechnology/ xml/msxml.asp. Разборщик XML фирмы Microsoft, высокопроизводительный, доступный в виде COM-компонента. Его можно использовать для реализации поддержки XSLT в приложениях;
- Sablotron, http://www.gingerall.com/charlie-bin/get/webGA/acV'sablotron.act. Sablotron — быстрый, компактный и переносимый процессор XSLT. На настоящий момент поддерживает подмножество рекомендации XSLT. Его можно использовать вместе с C или Perl;
- SAXON, http://users.iclway.co.uk/mhkay/saxon/index.html. Такой процессор XSLT полностью реализует XSLT 1.0 и XPath 1.0, а также ряд расширений

упомянутых спецификаций. Обратите внимание на то, что в этом выпуске имеется также поддержка рабочего проекта XSLT 1.1;

- Transformiix, http://www.mozilla.org. Transformiix — XSLT-компонент фирмы Mozilla, в настоящее время частично реализованный в Netscape 6.0;
- Unicorn XSLT processor (UXT), http://www.unicorn-enterprises.com. Этот процессор XSLT поддерживает XSLT-преобразования и написан на C++;
- Xalan C++, http://xml.apache.org/xalan-c/index.html. Реализация рекомендаций W3C для XSLT и XML Path Language (XPath). Версия известного процессора Apache Xalan на C++;
- Xalan Java, http://xml.apache.org/xalan-j/index.htmL Реализация рекомендаций W3C для XSLT и XML Path Language (XPath) на Java. Версия известного процессора Apache Xalan на Java. Включает также функции расширения для доступа к базам данных на SQL через JDBC и многое другое;
- xesalt, http://www.inlogix.de/products.htmL Этот процессор XSLT существует в виде web-серверного модуля (как для web-сервера Apache, так и для IIS), в виде расширения Netscape 4.x и в виде процессора командной строки;
- XML parser for C, http://technet.oracle.com/tech/xml/parser_c2. Процессор XSLT фирмы Oracle. Поддерживает рекомендацию XSLT 1.0, разработан для использования с C;
- XML parser for Java, http://technet.oracle.com/tech/xml/parserjava2. Процессор XSLT фирмы Oracle. Поддерживает рекомендацию XSLT 1.0, разработан для использования с Java;
- XML parser for PL/SQL, http://technet.oracle.com/tech/xml/ parser_plsqL Процессор XSLT фирмы Oracle. Поддерживает рекомендацию XSLT 1.0, разработан для использования с PL/SQL;
- XMLrXSLT, http://xmlxslt.sourceforge.net. Этот разборщик XSLT написан на Perl. Частично реализует рекомендацию XSLT;
- Xport, http://www.timelux.lu. Процессор XSLT-преобразования, доступен в виде COM-объекта;
- XSL:P, http://www.clc-marketing.com/xslp/download.htmL Современный процессор XSLT;
- XT, http://www.jclark.com/xml/xt.html. XT — известная реализация рекомендации XSLT на Java.

В следующих разделах мы подробно рассмотрим четыре из вышеперечисленных процессоров XSLT: XT, Saxon, XSLT-процессор фирмы Oracle и Xalan. Все Они бесплатно доступны в Интернете, и на них можно проверить все приведенные в книге примеры. Если вы хотите изучать примеры, загрузите один из этих процессоров (лучше всего известный и широкоиспользуемый — процессор Xalan). Для осуществления XSLT-преобразований я буду пользоваться этими процессорами при дальнейшем изложении.

Выбранные процессоры основаны на Java, поэтому необходимо, чтобы у вас была установлена Java. Если у вас она еще не установлена, ее можно загрузить

бесплатно с web-узла фирмы Sun. Самое последнее издание, на момент написания книги Java 2 версии 1.3 доступно по адресу http://java.sun.eom/j2se/l.B. Все, что нужно сделать, — загрузить Java для вашей операционной системы и следовать инструкциям по загрузке на соответствующих страницах.

Хотя для запуска этих XSLT-процессоров нужна Java, не стоит беспокоиться, если вы не программист — никакого программирования не требуется. В главе 10 будут приведены примеры программирования на Java — для того чтобы продемонстрировать вам, как создавать XSLT-преобразования в коде — но процессоры XT, Saxon, XSLT от Oracle и Xalan можно запускать из командной строки.

Если вы работаете в Windows, то для вас существует еще более простой способ использовать XT и Saxon — эти процессоры поставляются в виде файлов .exe (xt.exe и saxon.exe), которые можно выполнять непосредственно в Windows — и вам совсем не понадобится Java. Этот способ работы также рассмотрен в книге.

Использование процессора XSLT на Java

Для работы с основанным на Java процессором XSLT достаточно загрузить и распаковать (unzip) его — и он готов к работе. Прочитайте, конечно, сопроводительные указания; но, как правило, нужно осуществить только два шага.

Во-первых, необходимо уведомить Java, как ей найти XSLT-процессор, который хранится в файле архива Java, или JAR (Java Archive). Чтобы сообщить Java о том, где искать JAR-файл, нужно установить переменную окружения classpath в путь этого JAR-файла. Например, в любой версии Windows следует открыть окно DOS и выполнить команду, устанавливающую переменную classpath в путь к JAR-файлу процессора XSLT от Oracle, xmlparserv2.jar, который в этом случае хранится в каталоге c:\oraclexml\lib:

```
C:\>set dasspath=c:\oradexml\lib\xmlparserv2.jar
```

Теперь можно перейти ко второму этапу — запустить процессор XSLT; при этом будет выполняться класс Java, поддерживающий данный процессор. Для XSLT-процессора Oracle этот класс называется oracle.xml.parser.v2.oraxsl. В Windows, например, вы можете перейти в каталог, в котором расположены файлы planets.xml и planets.xsl, и выполнить класс oracle.xml.parser.v2.oraxsl при помощи Java таким образом:

```
C:\planets>java oracle.xml.parser.v2.oraxsl planets.xml planets.xsl planets.html
```

При этом файл planets.xml будет преобразован в planets.html при помощи planets.xsl. Заметьте: в этом примере предполагается, что файл java.exe, в котором исполняется Java-код, расположен в вашем каталоге Windows. Если его там нет, можно явно задать его месторасположение, каталог Java bin, например c:\jdkl.3\bin (JDK— сокращение от Java Development Kit, а Java 2 версии 1.3 устанавливает себя по умолчанию в каталог c:\jdkl.3), как в следующей команде:

```
C:\planets>c:\jdkl.3\bin\java oracle.xml.parser.v2.oraxsl
planets.xmlplanets.xslplanets.html
```

Можно объединить оба этапа (установка переменной classpath и запуск процессора XSLT) в один, если вы используете ключ -cp вместе с Java, указывая, какую переменную classpath использовать:

```
C:\planets>c:\jdkl.3\bin\java -cp c:\oraclexml\lib\xmlparserv2.jar
oracle.xml.parser.v2.oraxsl planets.xml planets.xsl planets.html
```

Это весьма длинные команды, что может на первый взгляд показаться вам излишне сложным. Тем не менее есть веская причина, по которой большинство процессоров XSLT написаны на Java: Java поддерживается на многих платформах, от Macintosh до UNIX, а поэтому XSLT-процессор также может выполняться на всех этих платформах.

Конечно, все становится намного проще, если вы работаете в Windows и используете скомпилированную версию процессора XT (xt.exe) либо Saxon (saxon.exe). Ниже приведен пример работы с xt.exe в Windows для осуществления того же преобразования (здесь предполагается, что xt.exe расположен в текущем каталоге):

```
C:\planets>xt planets.xml planets.xsl planets.html
```

Мы рассмотрели процесс кратко; теперь я хочу описать четыре процессора XSLT (XT, Saxon, процессор XSLT от Oracle и Xalan) подробно, продемонстрировав, как работать с каждым из них. Обратите внимание на два момента: программное обеспечение, реализующее языки XML и XSL, меняется очень быстро, так что когда вы читаете эти строки, некоторые процессоры уже могут оказаться устаревшими; хотя все процессоры должны поддерживать весь стандартный XSLT, в некоторых случаях они приводят к различным результатам.

XT Джеймса Кларка

XT Джеймса Кларка (James Clark) можно загрузить с www.jclark.com/xml/xt.html. Помимо самого процессора XT, вам понадобится разборщик XML, при помощи которого XT будет читать ваш XML-документ. В файлах загрузки XT есть еще файл sax.jar, содержащий разборщик XML Джеймса Кларка; можете впрочем воспользоваться в этих же целях и его разборщиком XP, загрузив его с адреса www.jclark.com/xml/xp/index.html.

Я лично предпочитаю использовать разборщик XML Xerces от Apache Project, который расположен по адресу http://xml.apache.org. (Ко времени написания книги текущую версию, Xerces 1.3.0, можно было озагрузить с адреса http://xml.apache.org/dist/xerces-j/ в формате zip для UNIX в файле Xerces-J-bin.l.3.0.tar.gz и формате для Windows в файле Xerces-J-bin. 1.3.0.zip.)

Сам процессор XT — это приложение Java, включенное в пакет загрузки XT в файле JAR, xt.jar. Для использования xerces.jar и xt.jar вы должны включить и тот, и другой файлы в переменную classpath, как показано в следующем примере для Windows (соответственно измените места нахождения этих файлов):

```
C:\>set classpath-C:\xerces-l_3_0\xerces.jar;C:\xt\xt.jar
```

Затем вы можете воспользоваться классом преобразования XT, com.jclark.xsl. sax.Driver.class. Нужно предоставить имя разборщика, которое вы хотите использовать (в данном случае это org.apache.xerces.parsers.SAXParser в архиве xerces.jar), установив переменную com.jclark.xsLsax.parser в нужное имя в командной строке. Вот как я использую XT для преобразования в Windows (считая, что файлы planets.xml и planets.xsl содержатся в каталоге c:\planets, ajava.exe расположен в вашем текущем каталоге):

```
C:\planets>java -Dcom.jclark.xsl.sax.parser-org.apache.xerces.parsers.SAXParser
com.jclark.xsl.sax.Driver planets.xml planets.xsl planets.html
```

Эта строка довольно длинна, и хорошо, что XT также упакован в исполняемый файл Win32 xt.exe. Для работы с xt.exe, однако, необходимо, чтобы на компьютере была установлена виртуальная машина Java от Microsoft, Microsoft Java Virtual Machine (VM), — она входит в поставку Internet Explorer. Вот пример в Windows, осуществляющий то же преобразование, что и предыдущая строка (предполагается, что xt.exe расположен в текущем каталоге):

```
C:\planets>xt planets.xml planets.xsl planets.html
```

Если xt.exe расположен не в текущем каталоге, вы можете задать его месторасположение явно, как в следующем примере (если xt.exe находится в c:\xt):

```
C:\planets>c:\xt\xt planets.xml planets.xsl planets.html
```

Saxon

Saxon Майкла Кэя (Michael Kay) — один из самых первых процессоров XSLT, его можно бесплатно загрузить с http://users.iclway.co.uk/mhkay/saxon/. Все, что нужно сделать, — это загрузить файл saxon.zip и распаковать архив zip; будет создан требуемый файл Java JAR, saxon.jar.

Для осуществления XSLT-преобразования вначале нужно удостоверится, что saxon.jar находится в переменной classpath. Например, в Windows, считая, что saxon.jar находится в c:\saxon, установить переменную classpath можно следующим образом:

```
C:\>set classpath-c:\saxon\saxon.jar
```

Теперь для выполнения преобразования можно использовать класс Saxon XSLT, com .icl.saxon .StyleSheet.class:

```
C:\planets>java com.icl.saxon.StyleSheet planets.xml planets.xsl
```

По умолчанию Saxon выводит результаты на экран — не совсем то, что нужно, если мы хотим создать файл planets.htmL Чтобы создать файл planets.html, можно воспользоваться символом перенаправления вывода UNIX или DOS >:

```
C:\planets>java com.icLsaxon.StyleSheet planets.xml planets.xsl > planets.html
```

При работе в Windows несложно сразу же вызвать Saxon в виде исполнимой программы Win32, saxon.exe, которая загружается с http://users.iclway.co.uk/mhkay/saxon/. В Windows его можно запустить следующим образом (часть -o planets.html задает имя выходного файла):

```
C:\planets>saxon -o planets.html planets.xml planets.xsl
```

Oracle XSLT

Корпорация Oracle также предоставляет бесплатный процессор XSLT, который можно загрузить с http://technet.oracle.com/tech/xml/, — правда, для этого придется пройти весьма длительную процедуру регистрации. Ко времени написания книги для загрузки процессора XSLT достаточно было щелкнуть на ссылке для XDK для Java http://technet.oracle.com/tech/xml/.

После распаковки архива zip, загруженного с web-узла Oracle, требуемый JAR-файл (на момент написания книги) будет называться xmlparserv2.jar. Его можно поместить в вашу переменную classpath в Windows следующим образом:

```
C:\>set classpath-c:\oraclexml\lib\xmlparserv2.jar
```

Требуемый класс Java — oracle.xml.parser.v2.oraxsl, для преобразования planets.xml в planets.html вы можете использовать его таким образом:

```
C:\planets>java oracle.xml.parser.v2.oraxsl planets.xml planets.xsl planets.html
```

Xalan

Скорее всего, самый широко используемый отдельный процессор XSLT — это Xalan фирмы Apache Project (Apache — широко распространенный web-сервер). Версию Xalan на Java можно загрузить с http://xml.apache.org/xalan-j/index.html — достаточно щелкнуть на требуемом вам файле архива zip (сейчас это xalan-j_2_0_0.zip для Windows или xalan-j_2_0_0.tar.gz для UNIX).

При распаковке загруженного файла вы получите как процессор XSLT, xalan.jar, так и разборщик XML, xerces.jar. Оба эти JAR-файла в Windows можно включить в classpath следующим образом (измените пути к файлам в соответствии с вашей системой):

```
C:\>set classpath«c:\xalan-j_2_0_0\bin\xalan.jar;c:\xalan-j_2_0_0\bin\xerces.jar
```

Чтобы затем использовать planets.xsl для преобразования planets.xml в planets, html, запустите на выполнение класс Java org.apache.xalan.xslt.Process:

```
C:\planets>java org.apache.xalan.xslt.Process
-IN planets.xml -XSL planets.xsl -OUT planets.html
```

Заметьте, что для указания входного файла используется -IN, для указания выходного —OUT, а для указания таблицы стилей XSLT —XSL. Чаще всего мы будем работать именно с процессором Xalan, поэтому ниже Приведены дополнительные подробности. Следующий список содержит все лексемы, которые можно использовать с классом org.apache.xalan.xslt.Process, как их выводит сам Xalan:

- -CR. Использовать в выходных данных только символы возврата каретки (carriage return) — по умолчанию используется CR/LF;
- -DIAG. Вывод диагностики времени;
- -EDUMP [optional] FileName. Копировать стек при ошибке;
- -HTML, Использовать формат HTML;
- -IN inputXMLURL;

- -INDENT. Количество пробелов для выравнивания (indent) каждого уровня в выходном дереве — по умолчанию 0;
- -LF. Использовать в выходных данных только символы перевода строки (linefeed) — по умолчанию используется CR/LF;
- -OUT outputFileName;
- -PARAM name value. Устанавливает параметр таблицы стилей;
- -Q. Режим с минимальным выводом (quiet);
- -QC. Quiet Pattern Conflicts Warnings, предупреждения конфликтов шаблонов;
- -TEXT. Использовать средство форматирования простого Текста;
- -TG. Отслеживать (trace) каждое событие генерации результирующего дерева;
- -TS. Отслеживать каждое событие выделения;
- -TT. Отслеживать шаблоны по мере их вызова;
- -TTC. Отслеживать «детей» шаблона по мере их обработки;
- -V. Информация о версии;
- -VALIDATE. Проверить входной XML и XSL (по умолчанию проверка отключена);
- -XML. Использовать средства форматирования XML и добавить заголовок XML;
- -XSL XSLTransformationURL.

В книге мы рассмотрим все эти процессоры; как уже говорилось, чаше всего я буду использовать Xalan. (Причина в том, что этот процессор стал самым популярным и широко распространенным процессором XSLT.) Разумеется, вы можете использовать любой процессор XSLT, если только он удовлетворяет спецификации XSLT консорциума W3C.

На этом мы завершаем рассмотрение отдельных процессоров XSLT. Существует другой способ преобразовывать документы XML, не прибегая к отдельной программе: для этих целей можно использовать такую клиентскую программу, как браузер.

Преобразование документов XML при помощи браузеров

Поддержка XSLT включена и в Microsoft Internet Explorer, и в Netscape Navigator. Из этих двух браузеров Internet Explorer обладает гораздо большей поддержкой XSLT, и здесь я буду использовать версию 5.5 этого браузера. О поддержке XSLT в Internet Explorer вы можете прочитать по адресу http://msdn.microsoft.com/xml/XSLGuide/.

Internet Explorer 5.5 и ранних версий по умолчанию не поддерживает в точности синтаксис XSLT, поэтому нам нужно провести ряд модификаций в файлах

planets.xml и planets.xsl. (В следующей главе мы рассмотрим эту тему подробнее. Там приведены адреса, с которых можно загрузить пакеты обновленной поддержки XSLT.) Когда печаталась эта книга появился уже Internet Explorer 6.0. Я установил его и проверил работу; похоже, он поддерживает стандартный синтаксис XSLT (за тем исключением, что все еще требуется использовать тип «text/xsU для таблиц стилей — таких как <?xml-stylesheet type-"text/xsl" href= "planets.xsl"?» вместо «text/xml»). Если вы все еще используете IE 5.5 или более ранний, вам нужно будет провести приведенные здесь и в следующей главе изменения. Если вы хотите избежать этого, используйте IE 6.0: кажется, этот браузер поддерживает полный синтаксис XSLT.

Для работы с planets.xml в IE (в том числе и версии 6.0) мне пришлось преобразовать атрибут type в инструкции обработки <?xml-stylesheet?> из «text/xml» в «text/xsU (предполагается, что файл planets.xsl расположен в том же каталоге, что и planets.xml, как указывает атрибут href в листинге 1.3).

Листинг 1.3. Версия planets.xml для Microsoft Internet Explorer

```
<?xml version-"! 0"?>
<?xml-stylesheet type-"text/xsl" href-"planets.xsl"?>
<PLANETS>

    <PLANET>
        <NAME>Mercury</NAME>
        <MASS UNITS-"(Earth - 1)">.0553</MASS>
        <DAY UNITS-"days">58.65</DAY>
        <RADIUS UNITS-"miles">1516</RADIUS>
        <DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
        «DISTANCE UNITS-"mill1on miles">43.4</DISTANCE><!--B перигелии-->
    </PLANET>

    <PLANET>
        <NAME>Venus</NAME>
        <MASS UNITS-"(Earth - 1)">.815</MASS>
        <DAY UNITS-"days">116.75</DAY>

        <RADIUS UNITS-"miles">3716</RA0IUS>
        <DENSITY UNITS-"(Earth - 1)">.943</DENSITY>

        «DISTANCE UNITS-"minion miles">66.8</DISTANCE><!--B перигелии-->
    </PLANET>

    <PLANET>
        <NAME>Earth</NAME>
        <MASS UNITS-"(Earth - 1)">1</MASS>
        <DAY UNITS-"days">1</DAY>
        <RADIUS UNITS-"mi 1es">2107</RAOIUS>
        «DENSITY UNITS-"(Earth - 1)">1</DENSITY>
        <DISTANCE UNITS-"million miles">128.4</DISTANCEx!--B перигелии-->
    </PLANET>

</PLANETS>
```

Теперь, если вы используете IE версии 5.5 или более ранней, вы должны также преобразовать таблицу стилей planets.xsl (но не для версии 6-0 и старше;

в этом случает нужно только изменить атрибут type в инструкции обработки <?xml-stylesheet?> из «text/xmU в «text/xsU). Мы рассмотрим, как это сделать, в следующей главе, а вот новая версия planets.xsl, которую следует использовать:

Листинг 1.4. Версия planets.xsl для Microsoft Internet Explorer

```
<?xml version-"1.0"?»
<xsl:stylesheet xml ns: xs 1 http://www w3.org/TR'WD-xsl">

    <xsl-.template match-"/">
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <B0DY>
                <H1>
                    The Planets Table
                </Hl>
                <TABLE BORDER-"2"»
                    <TR>
                        <TD>Name</TD>
                        <TD>Mass</TD>
                        <TD>Radius</TD>
                        <TD>Day</TD>
                    </TR>
                    <xsl:apply-templates/>
                </TABLE>
            </B0DY>
        </HTML>
    </xsl:tempiate>

    «xsl:tempiate match-"PLANETS">
        <xsl:apply-tempiates/>
    </xsl:template»

    <xsltemplate match-"PLANET">
       <TR>
          <TD><xsl value-of select-"NAME''/></TD>
          <TDxxsl:apply-templates select-"MASS^M/x/TD>
          <TDxxsl :apply-templates select-"RADIUS"/x/TD>
          <TDxxs1 -.apply-templates select-"DAY"/x/TD>
       </TR>
    </xsl:tempiate>

    <xsltemplate match-"MASS">
        <xsl:value-of select-"."/>
        <xsl value-of select-"@UNITS"/>
    </xsl:template»

    <xsl:tempiate match-"RADIUS">
        <xs1:value-of select-"."/»                                    *
        <xsl:value-of select-"@UNITS7>
    </xsl:template»

    <xsl :template match^DAY"»
        <xsl value-of select-"."/»
```

```
        <xsl:value-of select="@UNITS'7>
    </xsl:template»

</xsl:stylesheet>
```

Теперь можно непосредственно открыть planets.xml в Internet Explorer, как видно на рис. 1.3.

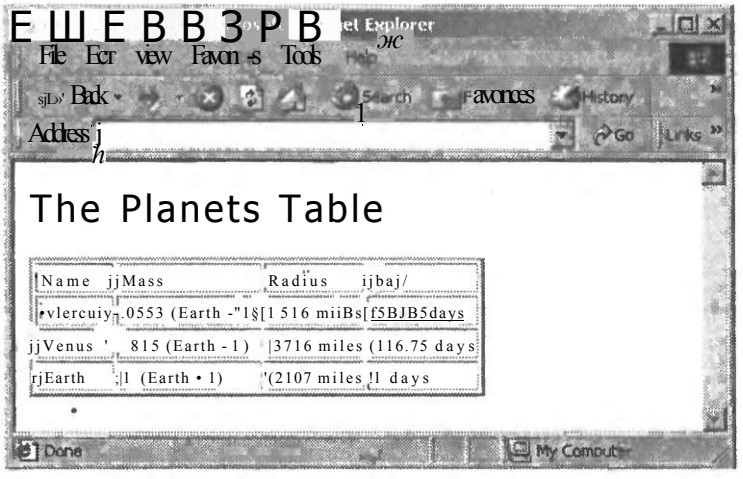


Рис. 1.3. Осуществление XSLT-преобразования в Internet Explorer

Хотя таким образом при помощи Internet Explorer можно работать с XSLT, при этом необходимо так изменять таблицу стилей, чтобы она удовлетворяла требованиям Internet Explorer. Поскольку Internet Explorer в текущий момент фактически не поддерживает XSLT, открывая документы XML при навигации, я не буду использовать этот браузер для XSLT-преобразований в книге (кроме специально оговоренных случаев). Для осуществления преобразований я стану использовать процессоры XSLT, такие как Saxon и Xalan, а если результаты будут представлятся в формате HTML, то мы будем просматривать их в Internet Explorer.

Интересно отметить, что существует способ провести настоящие XSLT-преобразования в Internet Explorer, не делая никаких специальных изменений в документах XML или XSL, даже не загружая и не устанавливая последний разборщик MSXML (как обсуждается в главе 2) — однако при этом нужно не осуществлять навигацию XML-документа, а обращаться к встроенному в Internet Explorer процессору XSLT, MSXML3, напрямую при помощи JavaScript.

Работа с XSLT и JavaScript в Internet Explorer

Процессор XSLT в Internet Explorer 5.5 является частью разборщика XML MSXML3, и если вы работаете напрямую с MSXML3 при помощи JavaScript, нет необходимости изменять исходные файлы planets.xml и planets.xsl (лис-

тинги 1.1 и 1.2), как мы это сделали в предыдущем разделе. Мы посмотрим, как это работает в главе 10, а здесь приведена web-страница xslt.html, которая при помощи JavaScript и MSXML3 осуществляет преобразование planets.xml с использованием planets.xsl и отображает результаты (заметьте, что вы можете модифицировать этот документ для использования ваших собственных документов XML и XSLT, не прибегая к написанию кода на JavaScript: достаточно заменить имена planets.xml и planets.xsl на названия ваших документов XML и XSL) (листинг 1.5).

Листинг 1.5. Преобразование в Internet Explorer при помощи JavaScript

```
<HTML>
    <HEAD> .
        <TITLE>XSLT Using JavaScript</TITLE>

        «SCRIPT LANGUAGE-"JavaScri pt">
        <!--

        function xsltO
        {
           var XMLDocument - new ActiveX0bject('MSXML2 DOMDocument 3 0'):
           var XSLDocument - new ActiveXObject('MSXML2.DOMDocument.3.0'):
           var HTMLtarget - document.allC'targetDIV']:

           XMLDocument.validateOnParse - true:
           XMLDocument.1oad('pianets.xml'):
           1f (XMLDocument.parseError.errorCode !- 0) {
                HTMLtarget.InnerHTML - "Error!"
                return false:
           }

           XSLDocument.validateOnParse - true;
           XSLDocument.1oad('pianets.xsl'):
           if (XSLDocument.parseError.errorCode !- 0) {
                HTMLtarget.innerHTML - "Error!"
                return false:
           )

           HTMLtarget.innerHTR - XMLDocument.transformNode(XSLDocument):
           return true:
        }

        //-->
        </SCRIPT>
    </HEAD>

    <B0DY onload-"xslt()">
        <DIV ID-"targetDIV">
        </DIV>
    <BODY>
</HTML>
```

В результате обработки этой web-страницы будет получен тот же результат, что и на рис. 1.3: она напрямую загружает файлы planets.xml и planets.xsl и применяет к ним разборщик MSXML3. Это в точности те же файлы planets.xml и planets.xsl, с которыми мы работали на протяжении этой главы, — в них нет изменений, необходимых при рассмотрении предыдущей темы, в которой мы не-

посредственно просматривали planets.xml в Internet Explorer. Дополнительная информация приведена в главе 10.

ИСПОЛЬЗОВАНИЕ VBSCRIPT

Если вам больше нравится VBScript, попробуйте использовать этот язык сценариев Internet Explorer: можно достичь таких же результатов.

XSLT-преобразования на web-сереерах

Можно осуществлять XSLT-преобразования на web-сервере — так чтобы XML-документ был преобразован до того, как web-сереер отправит его браузеру. Здесь самое распространенное преобразование — это преобразование документа XML в HTML, но преобразования XML-XML на сервере также становятся все более и более распространенными.

В отличие от остальных XSLT-преобразований, рассмотренных до сих пор в этой главе, при проведении XSLT-преобразований на web-сервере обычно необходимо прибегнуть к программированию. Существует три распространенных способа выполнять XSLT-преобразования на web-серверах: используя сервлеты Java, Java Server Pages (JSP) и Active Server Pages (ASP). В главе 10 мы подробно исследуем все эти три способа. Некоторые процессоры XSLT можно настроить для использования на web-серверах — вот начальный список:

- AXSL, www.javalobby.org/axsl.html. AXSL — серверный инструмент, преобразующий XML в HTML при помощи XSLT;
- Microsoft XML Parser, http://msdn.microsoft.com/downloads/ webtechnology/ xml/msxml.asp. MSXML3 обеспечивает безопасный доступ к серверу по протоколу HTTP для работы с ASP;
- mod_xslt, http://modxslt.userworld.com. Простой модуль web-сереера Apache, использующий XSLT для получения содержимого XML. Для обработки XSLT используется процессор Sablotron;
- PXSLServlet, www.pault.com/PxsL При помощи этого сервлета можно преобразовать XML в HTML с использованием XSLT. Он также позволяет читать и записывать данные в базу данных на основе SQL (через JDBC);
- xesalt, www.inlogix.de/products.html. Этот процессор XSLT существует в виде модуля для web-серверов Apache и IIS;
- XML Enabler, www.alphaworks.ibm.com/tech/xmlenabler. XML Enabler позволяет отправлять запросы сервлету, а при ответе сервлета XML Enabler может отформатировать данные при помощи различных таблиц стилей XSLT;
- XT, может быть использован как сервлет Java. Ему требуется сервлет-машина (servlet engine), реализующая, по крайней мере, версию 2.1 Java Servlet API. Класс-сервлет Java называется com.jclark!xsi.sax.XSLServlet.

Следующий пример демонстрирует страницу JSP, активизирующую на web-сервере процессор Xalan. Xalan преобразует planets.xmlB planets.html при помощи

таблицы стилей planets.xsl. В коде затем считывается файл planets.html и отправляется обратно от web-сервера в браузер:

```
<X@ page errorPage-"error.jsp" language-"java"
    contentType-"text/html" import="org.apache.xalan.xslt.*:java. io.*" *>

<3;
    try
    {
        XSLTProcessor processor - XSLTProcessorFactory.getProcessor0:
        processor.process(new XSLTInputSource("pianets.xml").
            new XSLTInputSource("pianets.xsl").
            new XSLTResultTarget("planets.html")):
    }
    catch(Exception e) {}

    FileReader filereader - new FileReader("pianets.html"):
    BufferedReader bufferedreader - new BufferedReader(filereader);
    String instring;

    while((instring - bufferedreader. readLine0) !- null) { %>
        <*- instring %>
<% }
    filereader.close0:
%>
```

Результат можно увидеть на рис. 1.4: там изображен файл planets.html, отправленный в Internet Explorer web-сервером, на котором выполняется JSP. В главе 10 приведена дополнительная информация об использовании сервлетов Java, JSP и ASP для XSLT-преобразований на стороне сервера.

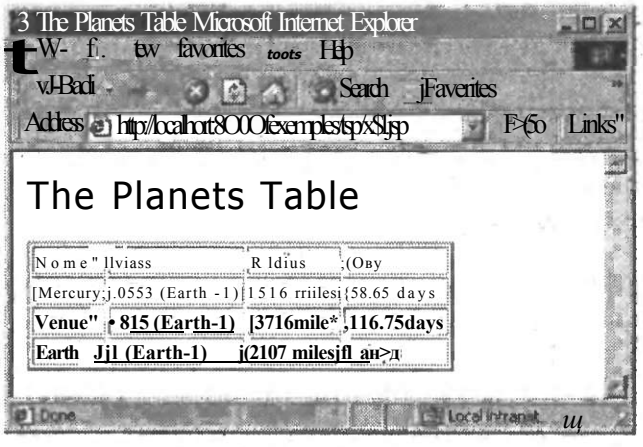


Рис. 1.4. Преобразование XML на web-сереере

Мы рассмотрели, как выполнить XSLT-преобразования при помощи отдельных процессоров XSLT в браузере Internet Explorer и на web-сервере. Однако единственное преобразование, которое мы до сих пор сделали, — это преобразование XML в HTML. Хотя сейчас это наиболее популярное преобразование, преобразования из XML в XML также становится все более распространенными.

Преобразования из XML в XML

Преобразования XML-XML иногда рассматриваются как SQL для Интернета, поскольку они позволяют оперировать запросами к базе данных в XML-документах. Ниже приведен пример. Используемый нами файл planets.xml содержит достаточно много данных о каждой планете:

```
<?xml version-"1.0"?>
<PLANETS>
    <PLANET>
        <NAME>Mercury</NAME>
        <MASS UNITS-"(Earth - 1)">.0553</MASS>
        <DAY UNITS-"days">58.65</DAY>
        <RADIUS UNITS-'miles">1516</RADIUS>
        «DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
        <01STANCE UNITS-"mi11 ion miles">43.4</DISTANCE><!--В перигелии-->
    </PLANET>

    <PLANET>
        <NAME>Venus</NAME>
        <MASS UNITS-"(Earth - 1)">.815</MASS>
        <DAY UNITS-"days">116 75</DAY>
        <RADIUS UNITS-"miles">3716</RADIUS>
        «DENSITY UNITS-"(Earth - 1)">.943</DENSITY>
        «DISTANCE UNITS-"million miles">66.8</DISTANCE><!--В перигелии-->
    </PLANET>
        .
        .
        .
```

Что, если нам нужно только подмножество этих данных — например, имя и масса каждой планеты? В терминах баз данных planets.xml представляет собой таблицу, и мы хотим создать новую таблицу, содержащую подмножество данных из первой. В базах данных для этого служит язык SQL, а для документов XML мы можем использовать XSLT.

В листинге 1.6 приведена новая версия файла planets.xsl, осуществляющая требуемое преобразование: выбираются только имя и масса каждой планеты, которые отправляются в выходной документ. В особенности обратите внимание на то, что мы осуществляем преобразование XML-XML, поэтому я использую элемент <xsl:output>, атрибут method которого установлен в «хт!» (фактически тип выходных данных обычно и есть XML, но если процессор XSLT видит тег <html>, он обычно по умолчанию генерирует HTML).

Листинг 1.6. Выбор только имени и массы

```
<?xml version-"1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

<xsl:strip-space elements-"*"/>
<xsl:output method-"xmr indent-"yes"/>

<xsl .-template match-"/">
    <xsl:apply-tempiates/>
```

продолжение &

Листинг 1.6 (продолжение)

```
</xsl:tempi ate>

    <xsl:template match-"PLANETS">
        <xsl:apply-templates/>
    </xsl template»

    <xsl  template match-"PLANET">
       <xsl:copy>
          <xsl:apply-templates/>
       </xsl:copy>
   </xsl:tempiate>

    <xsl:tempi ate match-"MASS">
        <xsl:copy>
        <xsl:value-of select-"."/»
        <xsl:value-of select-"@UNITS"/>
        </xsl:copy>
    </xsl:tempiate>

    <xsl:template match-"RADIUS">
    </xsl:tempiate>

    <xsl -.template match-"DAY">
    </xsl:tempiate>

    <xsl:tempi ate match-"DENSITY"»
    </xsl -.template»

    <xsl:template match-"DISTANCE">
    </xsl:template»

</xsl:stylesheet»
```

Далее я применяю эту новую версию planets.xsl к planets.xml, используя Xalan, чтобы создать новый документ XML, new.xml:

```
C:\planets>java org.apache.xalan.xslt.Process -IN planets.xml -XSL
planets.xsl -OUT new.xml
```

Вот как выглядит результирующий документ XML, new.xml:

```
<?xml version-'!.0" ehcoding-"UTF-8"?>
<PLANET>
    <NAME»Mercury</NAME>
    <MASS>.0553(Earth - 1)</MASS>
</PLANET»
<PLANET>
    <NAME»Venus</NAME»
    <MASS».815(Earth - 1)</MASS>
</PLANET»
<PLANET»
    <NAME»Ea rth</NAME»
    <MASS»1(Earth - 1)</MASS>
</PLANET»
```

Отметьте, что этот файл выглядит во многом похоже на исходный файл planets.xml, за тем исключением, что каждый элемент <PLANET> содержит только

элементы <NAME> и <MASS>. Таким образом, мы смогли получить подмножество данных первоначального документа XML.

Конечно, можно проводить любое число такого рода преобразований XML-XML. Можно обрабатывать данные в документе XML для создания совершенно новых XML-документов. Например, можно взять XML-документ с именами студентов и их отметками и создать новый документ, отображающий средние оценки. В XSLT есть много встроенных функций, позволяющих работать с данными подобным образом, — мы познакомимся с ними в главе 8.

В дополнение следует сказать, что многие программы используют XML для обмена данными в Интернете, и, так как обычно они форматируют свои документы XML по-разному, еще одним популярным способом использования преобразований XML-XML в Интернете является преобразование XML из формата, используемого одной программой, в формат другой программы.

Преобразования из XML в XHTML

Хотя во многих книгах рассматриваются исключительно преобразования из XML в HTML, правда состоит в том, что консорциум W3C этому не очень-то рад. Консорциум пытался сместить акцент с HTML (который они первоначально стандартизировали) на свою новую спецификацию, XHTML, представляющую собой совместимую с XML переработку HTML. Документы XHTML являются также хорошо сформированными допустимыми документами XML, поэтому преобразование из XML в XHTML в действительности есть преобразование из XML в специальный вид XML.

Хотя W3C активно продвигает XHTML, этот формат еще не стал широко распространенным. Поэтому в данной книге я в основном рассматриваю HTML; но поскольку W3C утверждает, что следует использовать XHTML, я кратко рассмотрю эту тему здесь и в главе 6. Если вы хотите узнать больше о XHTML, прочитайте рекомендацию W3C XHTML 1.0 по адресу www.w3.org/TR/xhtmll/, а также рекомендацию XHTML 1.1 по адресу www.w3.org/TR/xhtmlll/.

Пусть W3C и утверждает, что XML следует преобразовывать в XHTML, а не в HTML, я не видел ни одного работающего примера на их web-узле. Представленные ими примеры на самом деле не генерируют допустимых документов XHTML. Тем не менее, поддержка преобразований XML-XHTML предположительно будет встроена в XSLT 2.0, и желаемое W3C явление, вероятно, нас вскоре ждет.

Более подробно мы рассмотрим этот тип преобразования в главе 6, а здесь я приведу рабочую версию таблицы planets.xsl, которая создает допустимую XHTML-версию planets.html (листинг 1.7). Заметьте, что на этот раз в элементе <xsl:output> нужно использовать атрибут doctype-public, и хотя это корректный код XSLT, не все процессоры XSLT смогут его обработать.

Листинг 1.7. Преобразование XML-XHTML

```
<?xml version-"1.0"?>
<xsl: stylesheet version-"!..0"
xmlns: xsVhttp: //www. w3. org/1999/XSL /Trans form" >
```

* продолжение &

Листинг 1.7 (продолжение)

```
<xsl:output methods"xml" doctype-system
-"http://www.w3.org/TR/xhtml1/DTD/xhtmll-trans1t1onal  dtd"
    doctype-public-"-//W3C//DTD XHTML 1.0 Transitional//EN" indent-"yes'7>

    <xsl:template match-"/PLANETS">
        <html>
            <head>
                <title>
                    The Planets Table
                </title>
            </head>
            <body>
                <hl>
                    The F1anets Table
                </hl>
                <table>
                    <tr>
                        <td>Name</td>
                        <td>Mass</td>
                        <td>Radius</td>
                        <td>Day</td>
                    </tr>
                    <xsl:apply-templates/>
                </table>
            </body>
        </html>
    </xsl:tempiate>

    <xsl:tempi ate match-"PLANET">
       <tr>
          <td><xsl  value-of select-"NAME"/x/td>
          <td><xsl:apply-templates  select-"MASS"/></td>
          <td><xsl:apply-templates  select-"RADIUS"/></td>
          <tdxxsl :apply-templates select-"DAY"/></td>
       </tr>
   </xsl:template>

    <xsl:tempiate match-"MASS">
        <xsl:value-of select-"."/>
        <xsl:text> </xsl:text>
        <xsl:value-of select-"@UNITS"/>
    </xsl:tempiate>

    <xsl:template match-"RADIUS">
        <xsl:value of select-"."/>
        <xsl:text> </xsl:text>
        <xsl:value-of select-"@UNITS"/>
    </xsl:tempiate>

    <xsl:tempiate match-"0AY">
        <xsl;value-of select-"."/>
        <xsl:text> </xsl:text>
        <xsl :value-of select-''@UNITS'7>
    </xsl:tempiate>

</xsl:stylesheet>
```

Я преобразую planets.xml в допустимый документ XHTML, planets.html, используя этот новый вариант planets.xsl и процессор XSLT XT. Сначала нужно корректно установить переменную classpath:

```
C:\>set classpath-c:xerces\xerces-l_3_0\xerces.jar c:\xt\xt.jar;
```

Затем я провожу преобразование:

```
C:\p1anets>java - Dcom.jclark.xsl sax.parser-org.apache.xerces.parsers SAXParser
com.jclark xsl.sax.Driver planets xml planets.xsl planets.html
```

В результате будет получен файл XHTML, planets.html:

```
<?xml version-"l 0" encoding="UTF-8"?>
<!D0CTYPE html PUBLIC "-//W3C//DTD XHTML 1 0
Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-1rans i 11onal.dtd">
<html>
    <nead>
        <title>
            The Planets Table
        </t1tle>
    </head>

    <body>
        <hl>
            The Planets Table
        </hl>

        <tab1e>
            <tr>
                <td>Name</td>
                <td>Mass</td>
                <td>Radius</td>
                <td>Day</td>
            </tr>

            <tr>
                <td>Mercury</td>
                <td>.0553 (Earth - l)</td>
                <td>1516 miles</td>
                <td>58.65 days</td>
            </tr>

            <tr>
                <td>Venus</td>
                <td>.815 (Earth - l)</td>
                <td>3716 miles</td>
                <td>116.75 days</td>
            </tr>

            <tr>
                <td>Earth</td>
                <td>l (Earth - l)</td>
                <td>2107 miles</td>
                <td>l days</td>
            </tr>
```

```
        </table>
    </body>
</html>
```

Этот документ, planets.html, действительно является хорошо сформированным и допустимым документом формата XHTML 1.0 (самый популярный вид XHTML) в соответствии с программой проверки допустимости HTML и XHTML консорциума W3C. Средство проверки HTML/XHTML можно найти в Интернете по адресу http://validator.w3.org/file-upload.html. В главе 6 представлена дополнительная информация о преобразованиях XML-XHTML.

К настоящему моменту мы провели обзор работы XSLT, рассмотрев преобразования из XML в HTML, XML и XHTML. Далее в книге мы познакомимся с такими преобразованиями XSLT, как преобразования из XML в RTF (форматированный текст, Rich Text Format), в простой текст, XSL-FO, JavaScript, в базы данных на основе SQL, а также и другие. Кроме того, существует много дополнительного материала по XSLT, о котором вам следует знать, и теперь мы освоимся в разнообразных видах ресурсов XSLT, которые можно найти в Интернете.

Ресурсы XSLT

В Интернете можно найти огромное количество нужных нам сведений. Имейте в виду, что все приведенные далее адреса URL легко могут измениться; списки будут актуальны столь долго, сколько этого будут хотеть люди, занимающиеся поддержкой перечисленных web-узлов.

Спецификации, руководства и примеры XSLT

Точкой отсчета для ресурсов XSLT, безусловно, является сам W3C. Ниже приведены адреса URL для используемых в книге спецификаций W3C:

- www.w3.org/Style/XSL/. Основная страница W3C по XSL;
- www.w3.orgДR/xslt. Спецификация XSLT 1.0;
- www.w3.orgДR/xsltll. Рабочий проект XSLT 1.1, упрощающий работу с расширениями XSLT и добавляющий поддержку рекомендации W3C XBase;
- www.w3.org/TR/xslt20req. Требования XSLT 2.0, в которых приводится обзор XSLT 2.0, включающий дополнительную поддержку схем XML;
- www.w3.org/TR/xsl/. Форматирующие объекты XSL;
- www.w3.org/Style/2000/xsl-charter.html. Цели комитета XSL;
- www.w3.org/TR/xpath. Рекомендация XPath 1.0;
- www.w3.org/TR/xpath20req. Требования XPath, в которых приводится обзор XPath 2.0, включающего дополнительную поддержку XSLT 2.0;
- http://lists.w3.org/Archives/Public/www-xml-stylesheet-comments/. Список таблиц стилей XML консорциума W3C.

Многие руководства по XSLT и примеры доступны также и из других источников — вот начальный список:

- http://http.cs.berkeley.edu/~wilensky/CS294/xsl-examples.html. Ряд примеров по XSLT;
- http://msdn.microsoft.com/xml/reference/xsl/Examples.asp. Примеры шаблонов XSLT, используемых в совпадающих элементах;
- http://msdn.microsoft.com/xml/XSLGuide/xsl-overview.asp Начальное описание работы с XSLT;
- www.lists.ic.ac.uk/hypermail/xml-dev/xml-dev-Nov-1999/0371.html. Руководство по XSLT в виде презентации PowerPoint;
- www.mulberrytech.com/xsl/xsl-list/. Открытый список, посвященный обсуждению XSL;
- www.nwalsh.com/docs/tutorials/xsl/xsl/slides.html. Руководство по XSLT;
- www.oasis-open.org/cover/xsl.html. Описание текущих событий вокруг XSLT;
- www.w3.org/Style/Activity. Внушительный список страниц, посвященных таблицам стилей W3C;
- www.xmll01.com/xsl/. Полноценный набор справочников по XSLT;
- www.xslinfo.com. Полезный набор ресурсов по XSLT, собранных Джеймсом Таубером (James Tauber);
- www.2von.org/xxl/XSLTutorial/Books/Bookl/bookIn0ne.html. Справочники по XSLT, XPath, XML, WML и другим языкам.

Мне известна однако только одна группа новостей Usenet по XSLT, и она поддерживается Microsoft — microsoft.public.xsL Со временем появятся и другие. Для вас может также представлять интерес список почтовой рассылки по XSL — www.mulberrytech.com/ xsl/xsl-list

Помимо спецификаций, руководств и примеров W3C, в Интернете можно найти большое число редакторов создания таблиц стилей XSLT.

Редакторы XSLT

Для создания используемых в книге документов XML и XSL достаточно обычного текстового редактора, такого как vi, emacs, pico, Windows Notepad или Windows WordPad. По умолчанию предполагается, что документы XML и XSL должны быть написаны в Unicode, хотя на практике их можно создавать и в ASCII,

РАБОТА С WORDPAD

У текстовых редакторов Windows, таких как WordPad, есть раздражающая особенность — если они не узнают данное вами файлу расширение, они добавляют к имени файла расширение .txt. Для файлов .xml и .xsl это не проблема, поскольку WordPad понимает эти расширения, но если вы попытаетесь сохранить документы, созданные при работе над книгой, с расширением, не распознаваемым WordPad, ко всем ним редактор добавит расширение .txt. Чтобы этого не происходило, при сохранении документа заключайте имя файла в кавычки: "file.abc".

и практически все документы до сих пор были написаны именно в ASCII. Следите только за тем, чтобы при создании документа сохранять его в формате простого текста вашего редактора.

Будет, однако, намного проще использовать настоящий редактор XML, специально предназначенный для обработки XML-документов. Вот список программ редактирования документов XML:

- Adobe FrameMaker, www.adobe.com. Adobe представляет прекрасный, но дорогой редактор FrameMaker с поддержкой XML;
- XML Pro, www.vervet.com/. Дорогой, но мощный редактор XML;
- XML Writer, на диске, XMLWriter http://xmlwriter.net/. Выделение ключевых слов цветом, удобный интерфейс;
- XML Notepad, msdn.microsoft.com/xml/notepad/intro.asp. Бесплатный редактор XML от Microsoft, немного неудобный в работе;
- eNotepad, www.edisys.com/Products/eNotepad/enotepad.asp. Аналог WordPad, имеющий хорошие средства работы с XML и удообный пользовательский интерфейс;
- XMetal from SoftQuad, www.xmetal.com. Дорогой, но очень мощный редактор XML, любимый редактор многих авторов;
- XML Spy, www.xmlspy.com/. Имеет хороший пользовательский интерфейс и прост в работе;
- Arbortext's Epic, www.arbortext.com/. Мощный редактор, дорогой, обладает богатыми возможностями настройки.

Редактор XML Spy изображен на рис. 1.5, XML Writer — на рис. 1.6, а XML Notepad — на рис. 1.7.

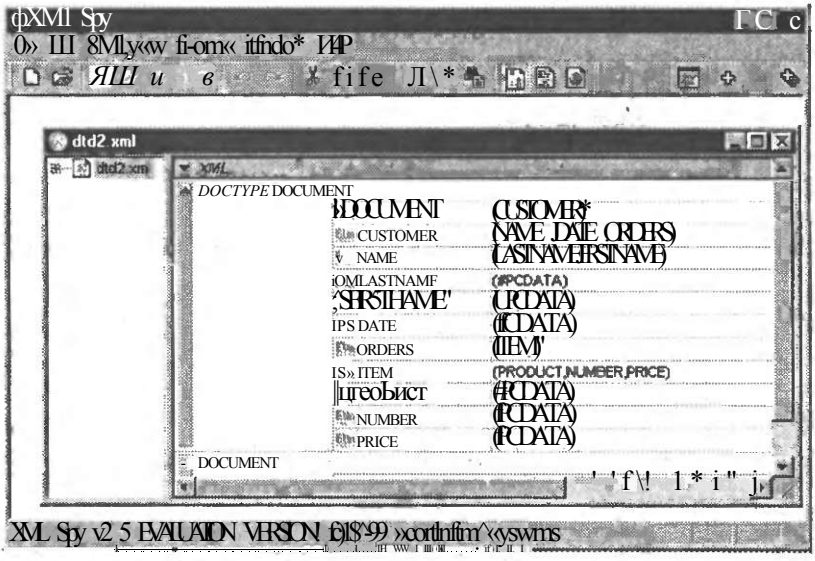


Рис. 1.5. Редактирование XML в XML Spy

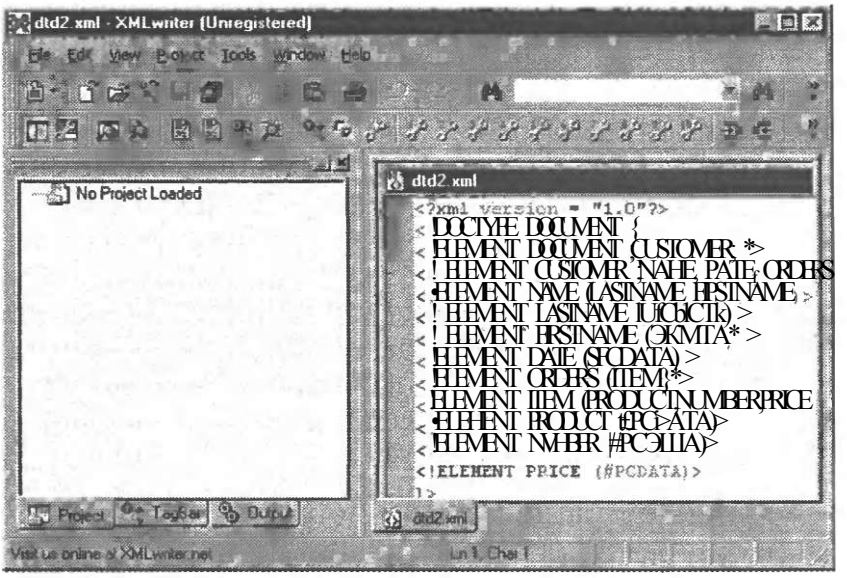


Рис. 1.6. Редактирование XML в XML Writer

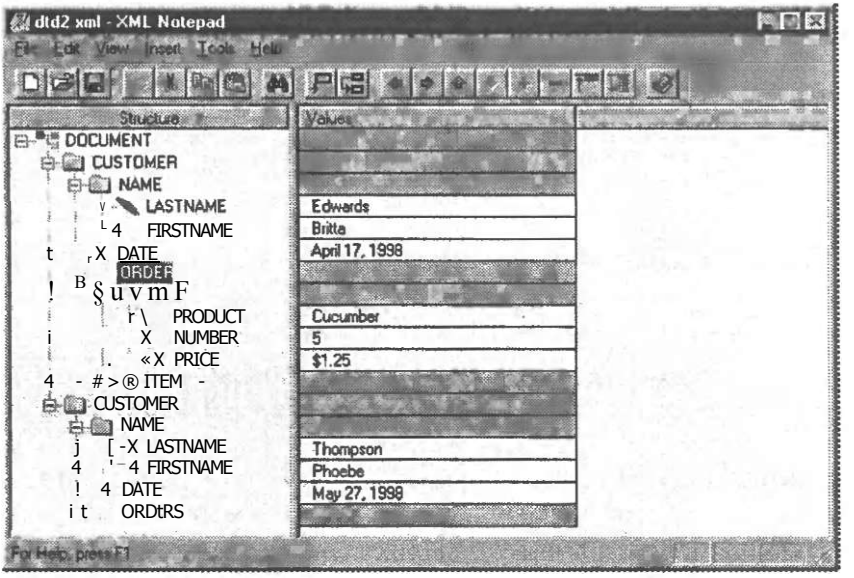


Рис. 1.7. Редактирование XML в XML Notepad

Существуют и специальные редакторы XSLT. Вот начальный список:

- http://lists.w3.org/Archives/Public/xsl-editors/. Список обсуждения редакторов XSL на web-узле W3C;
- IBM XSL Editor, www.alphaworks.ibm.com/tech/xsleditor. Редактор таблиц стилей XSLT на Java, обладающий визуальным интерфейсом для написания

таблиц стилей и выражений выбора и совпадения. Однако необходимо, чтобы была установлена поддержка Java 2 version 1.1 (не 1.2 или 1.3);

- Stylus, www.exceloncorp.com/products/excelon_stylus.html. Stylus включает редактор таблиц стилей XSLT;
- Visual XML Transformation Tool, www.alphaworks.ibm^com/aw.nsf/techmain/visualxmltools. Visual XML Transformation Tool генерирует для вас XSLT для преобразования исходных документов в результирующие;
- Whitehill Composer, www.whitehill.com/products/prod4.html. WYSIWYG средство генерации таблиц стилей XSLT с поддержкой перетаскивания;
- XL-Styler, www.seeburger.de/xmL Включает выделение ключевых слов, завершение тегов, предварительный просмотр HTML и многое другое;
- XML Cooktop, http://xmleverywhere.com/cooktop/. Этот редактор только что выпущен и выглядит прилично. Включены средства разработки и проверки таблиц стилей XSLT;
- XML Spy, www.xmlspy.com/. При помощи этого редактора XML можно также редактировать XSLT;
- XML Style Wizard, www.infoteria.com/en/contents/download. Средство генерации файлов XSLT при помощи мастера, который изучает данные XML и задает вопросы пользователю;
- xslide, www.mulberrytech.com/xsl/xslide. Поддерживает режим редактирования XSLT для Emacs;
- XSpLit, www.pefcussion.com/xmlzone/technology.htm. Позволяет вам разделять документы HTML на определения DTD XML и таблицы стилей XSLT.

Утилиты XSLT

В Интернете существует также много утилит XSLT — в следующем списке перечислены наиболее известные:

- Microsoft XSL API Extension, http://msdn.microsoft.com/downloads/webtechnology/xml/xslisapi.asp. Упрощает задачу выполнения XSLT-r еобразований на стороне сервера;
- Microsoft XSL-to-XSLT Converter, http://msdn.microsoft.com/downloads/webtechnology/xml/xsltconv.asp. Преобразует XSL в XSLT;
- XSL Lint, www.nwalsh.com/xsl/xslint. XSL Lint осуществляет проверку синтаксиса XSLT, позволяя обнаружить ошибки многих типов;
- XSL Trace, www.alphaworks.ibm.com/tech/xsltrace. Этот продукт позволяет пользователю проходить по шагам XSLT визуально;
- XSLT Compiler, www.sun.com/xml/developers/xsltc. Прообразует файлы XSLT в классы Java Для преобразования файлов XML;
- XSLT test tool, www.nètcrucible.com/xslt/xslt-tool.htm. Этот инструмент дает возможность запускать XSLT в разл] ечных популярных процессорах, что

позволяет проверить, корректно ли ваше преобразование работает на всех системах. Существует возможность вызывать из командной строки MSXML3 от Microsoft, как и любой другой процессор XSLT;

- XSLTC, www3.cybercities.com/x/xsltc. Компилирует таблицы стилей XSLT в код на C++. Основан на Transformix, процессоре XSLT фирмы Mozilla;
- XSLTracer, www.zvon.org/xxl/XSLTracer/Output/introduction.htmL XSLTracer — средство на языке Perl, демонстрирующее процесс обработки файлов XML при помощи таблиц стилей XSLT.

На этом завершается обзор XSLT в данной главе. Как видите, нас ждет огромный набор сведений. В оставшейся части главы будет представлен обзор XSL-FO.

Форматирующие объекты XSL: XSL-FO

Самая популярная часть XSL — это XSLT-преобразования, с которыми мы уже познакомились в этой главе. Другая, существенно большая часть — это форматирующие объекты XSL, XSL-FO (XSL Formatting Objects).

При помощи XSL-FO можно с точностью до миллиметра задать форматирование и способ отображения XML-документа. Для документов можно задать все, что угодно: шрифт текста, место, выравнивание, цвет, индексы, размер полей и многое другое. Работа с XSL-FO похожа на процесс создания вручную текстового процессора, и из-за сложности XSL-FO некоторые избегают их использовать. В главах 11 и 12 мы изучим то, что могут предложить нам XSL-FO, и способы работы с ними.

Ресурсы XSL-FO

Ряд ресурсов XSL-FO доступен в Интернете, но их гораздо меньше, чем ресурсов XSLT. Вот основные:

- www.w3.org/TR/xsl. Основная рекомендация-кандидат XSL, включающая также XSL-FO;
- http://lists.w3.org/Archives/Public/www-xsl-fo/. Список заметок W3C для XSL-FO.

Так же, как существуют процессоры XSLT, существуют и процессоры XSL-FO. Ни один из них, однако, не приближается к тому, чтобы реализовать стандарт полностью. Вот начальный список процессоров XSL-FO:

- FOP, http://xml.apache.org/fop. Приложение Java, которое считывает дерево форматирующих объектов XSL (создаваемое разборщиком XML) и создает документ PDF;
- PassiveTeX, http://users.ox.ac.uk/~rahtz/passivetex. Пакет TeX, форматирующий вывод XSL-FO в PDF. Использует разборщик XML xmltex Дэвида Карлайла (David Carlisle);

- SAXESS Wave, http://www.saxess.com/wave/index.htnil. Конвертер XML-Shockwave/ Flash;
- TeXML, http://www.alphaworks.ibm.com/tech/texml. Преобразует документы XML в формат TeX;
- Unicorn Formatting Objects (UFO), http://www.unicorn-enterprises.com. Процессор форматирующих объектов XSL, написанный на C++. Может генерировать вывод в форматах PostScript, PDF и других форматах, поддерживаемых драйверами DVI TeX;
- XEP, http://www.renderx.com/F02PDF.htmL Процессор XSL-FO на Java, преобразующий форматирующие объекты XSL в PDF или PostScript.

В этой книге я буду пользоваться FOP (formatting objects processor, процессор форматирующих объектов) — вероятно, самым распространенным процессором XSL-FO. Основанный на Java процессор XSL-FO берет XML-документ, написанный для использования форматирующих объектов XSL-FO, и преобразует его в формат PDF, который можно просмотреть в Adobe Acrobat. Хотя XSLT-преобразования часто производятся в HTML, для XSL-FO это работать не будет, потому что в этом случае указывается каждый аспект формата представления документа вплоть до мельчайших деталей, а для этих целей гораздо лучше подходит формат PDF.

Форматирование документа XML

Для форматирования planets.xml в planets.pdf мы можем воспользоваться фор' матирующими объектами XSL-FO, представленными в главе 12. Например, вот как мы можем отобразить название первой планеты, Меркурия, при помощи форматирующих объектов XSL-FO flow и block:

```
<fo:page-sequence master-name-"page">
    <fo:flow flow-name="xsl-region-body">
<fo:block font-family-"sans-serif" line-height-"48pt"
            font-size-"36pt" font-weight-"bold">
            Mercury
        </fo:block>
```

Однако создание всего документа целиком при помощи форматирующих объектов XSL — непростая задача, за исключением коротких документов. W3C предвидел эти трудности, и это одна из главных причин, по которой консорциумом был представлен язык преобразований, XSLT. В частности, можно создать таблицу стилей и при помощи XSLT преобразовать XML-документ так, чтобы он использовал форматирующие объекты XSL.

На практике преобразования практически всегда выполняются при помощи таблиц стилей, и именно так мы поступим в главах 11 и 12. Все, что вам нужно сделать, это предоставить таблицу стилей XSLT, при помощи которой

можно преобразовать ваш документ с использованием форматирующих объектов. При таком способе всю работу берет на себя процессор XSLT, преобразуя документ из представления, с которым вам удобно работать, в представление с форматирующими объектами, которое затем можно вставить в программу, способную обрабатывать форматирующие объекты и отображать форматированный результат.

Чтобы пояснить все вышесказанное, ниже приведен пример работы с XML-документом, с которым мы уже встречались в этой главе, planets.xml:

```
<?xml version-"].. 0"?>
<PLANETS>
    <PLANET>
        <NAME>Mercury«/NAME>
        <MASS UNITS-"(Earth - 1)">.0553</MASS>
        <DAY UNITS="days">58.65</DAY>
        «RADIUS UNITS-"miles">1516</RADIUS>
        «DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
        «DISTANCE UNITS-"million nriles">43.4</DISTANCE>«!--В перигелии-->
    </PLANET>

    <PLANET>
        «NAME>Venus«/NAME>
        <MASS UNITS-"(Earth - 1)">.815</MASS>
        <DAY UNITS-"days">116.75</DAY>
        «RADIUS UNITS-"miles">3716«/RADIUS>
        «DENSITY UNITS-"(Earth - 1)">.943</DENSITY>
        «DISTANCE UNITS-"million miles">66.8</DISTANCE>«!--В перигелии-->
    </PLANET>

    <PLANET>
        <NAME>Ea rth«/NAME>
        «MASS UNITS-"(Earth - 1)">1«/MASS>
        «DAY UNITS-"days">l«/DAY>
        «RADIUS UNITS-"miles">2107«/RADIUS>
        «DENSITY UNITS-"(Earth - 1)">1«/DENSITY>
        «DISTANCE UNITS-"million m1les">l28.4</DISTANCE>«!--В перигелии-->
    </PLANET>
</PLANETS>
```

В этом примере я воспользуюсь таблицей стилей XSLT — мы посмотрим, как ее создавать, в главе 11 — для преобразования planets.xml так, чтобы он использовал форматирующие объекты. Затем при помощи процессора FOP я преобразую новый документ в файл PDF. Мы также посмотрим, как форматированный документ выглядит в Adobe Acrobat.

Таблица стилей XSLT

В листинге 1.8 приведена таблица стилей planetsPDF.rcl, которая берет данные из planets.xml и форматирует их в файл PDF, planets.pdf. В этом случае для текста я использую крупный шрифт — 36 пунктов.

Листинг 1.8. Преобразование XML-XSL-FO

```
<?xml version-'1 0'?>
<xs1:stylesneet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fo-"http://www.w3.org/1999/XSL/Format"
    version-'1.0'>

    <xsl:template match-"PLANETS">
        <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
            <fo:1ayout-master-set>
                <fo:simple-page-master master-name-"page"
                    page-hei ght-"400mm" page-wi dth-"300mm"
                    margi n-top»"lOrmi" ma rgi n-bottom-"10mm"
                    margin-1eft«"20mm" margin-right-"20mm">

                    <fo:region-body
                        margin-top="0mm" margin bottom-"10mm"
                        margin-left-"0mm" margin-right-"Omm"/>

                    <fo:region-after extent-"10mm"/>
                </fo:simple-page-master>
            </fo:1ayout-ma ster-set>

            <fo:page-sequence master-name-"page">
                <fo: f1 ow f1ow-name-"xs1 -regi on-body">
                    <xsl:apply-templates/>
                </fo flow>
            </fo page-sequence»
        </fo root>
    </xsl:tempiate>

    <xsl:tempi ate match-"PLANET/NAME">
        <fo:block font-weight-"bold" font-size-"36pt"
            1i ne-height-"48pt" font-fami 1y-"sans-seri f">
            Name:
            <xsl:apply-templates/>
        </fo:b1ock>
    </xsl:tempiate>

    <xsl:tempi ate match-"PLANET/MASS">
        <fo:block font-size-"36pt" line-height-"48pt"
            font-family="sans-serif">
            Mass (Earth - 1):
            <xsl:apply-temp!ates/>
        </fo:block>
    </xsl:tempiate>

    <xsl:template match-"PLANET/DAY">
        <fo:block font-size-"36pt" line-height="48pt" font-family-"sans-serif">
            Day (Earth - 1):
            <xsl:apply-templates/>
        </fo block>
    </xsl:template»

    <xsl:tempiate match-"PLANET/RADIUS">
        <fo:block font-size-"36pt" line-height-"48pt" font-family-"sans-serif">
            Radius (in miles):
            <xsl:apply-templates/»
```

```
        </fo:block»
    </xsl template»

    <xsl template match-"PLANET/DENSITY">
        <fo:block font-size-"36pt" line-height="48pt" font-fami1y-"sans-serif">
            Density (Earth - 1):
            <xsl:apply-templates/>
        </fo:block»
    </xsl:tempi ate»

    <xsl template match-"PLANET/DISTANCE"»
        <fo:block font-size-"36pt" line-height="48pt" font-family-"sa ns-ser i f"»
            Distance (million miles):
            <xsl:apply-templates/»
        </fo:block»
    </xsl:tempi ate»
</xsl:stylesheet»
```

Преобразование документа в представление с форматирующими объектами

Для того чтобы преобразовать planets.xml в документ, использующий форматирующие объекты, который я назвал planets.fo, достаточно всего лишь применить таблицу стилей planetsPDF.xsl. Это можно сделать при помощи техник XSLT, уже рассмотренных в этой главе.

Например, чтобы создать planets.fo при помощи Xalan, в Windows сначала нужно установить classpath:

```
C:\>set classpath«c:\xalan\xalan-j_2_0_0\bin\xalan.jar;
c:\xalan\xalan-j_2_0_0\bin\xerces.jar
```

Затем применить planetsPDF.xsl к planets.xml для генерации planets.fo:

```
C:\planets>java org.apache.xalan.xslt.Process
-IN planets.xml -XSL planetsPDF.xsl -OUT planets.fo
```

Для задания форматирования документ planets.fo использует форматирующие объекты XSL. Вот как выглядит файл planets.fo (листинг 1.9):

Листинг 1.9. planets.fo

```
<?xml version-"1.0" encoding-"UTF-8"?>
<fo root xmlns:fo-"http.//www.w3.org/1999/XSL/Format"»

<fo:1ayout-master-set>
<fo:simple-page-master margin-right="20mm" margin-left-"20mm"
        ma rgi n-bottom-"lOmrn" margi n-top-"1Omm"
        page-width-"300mm" page-height-"400mm" master-name-"page">
<fo-region-body margin-right-"0mm" margin-left-"Omm"
            margin- bottom-*" 1 Omm" margin-top-"Omm"/>
        <fo:regi on-after extent-"lOnrn"/>
    </fo:simple-page-master»
</fo:1ayout-master-set»
```

продолжение &

Листинг 1.9 (продолжение)

```
<fo:page-sequence master-name-"page">
    <fo:flow flow-name«"xsl-region-body">
<fo:block font-family-*"sans-serif" line-height-"48pt"
            font-size-"36pt" font-weight-"bol d">
            Name:
            Mercury
        </fo:block>
        <fo:block font-fam1ly-"sans-serif" line-height-"48pt"
            font-size-"36pt">
            Mass (Earth = 1):
            .0553
        </fo-block>
        <fo block font-family-"sans-serif" line-height-"48pt"
            font-size-"36pt">
            Day (Earth - 1):
            58 65
        </fo:block>
        <fo:block font-family-"sans-serif" line-height«"48pt"
            font-size-"36pt">
            Radius (in miles):
            1516
        </fo:block>
        <fo:block font-family-"sans-serif" line-height-"48pt"
            font-size-"36pt">
            Density (Earth -1):
            .983
        </fo block>
        <fo block font-family-"sans-ser1f" line-height-"48pt"
            font-size-"36pt">
            Distance (million miles):
            43.4
        </fo:block>

<fo:block font-family-"sans-serif" line-height-"48pt"
            font-size-"36pt" font-weight-"bold">
            Name:
            Venus
        </fo block>
        <fo block font-family-"sans-serif" line-height»"48pt"
            font-size-"36pt">
            Mass (Earth - 1):
            .815
        </fo:block>
        <fo:block font-family-"sans-serif" line-height-"48pt"
            font-size-"36pt">
            Day (Earth - 1):
            116.75
        </fo block>
        <fo:block font-family-"sans-serif" line-height-"48pt"
            font size-"36pt">
            Radius (in miles):
            3716
        </fo:block>
        <fo:block font-family-"sans-serif" line-height-"48pt"
            font-size-"36pt">
```

```
            Density (Earth - J):
            .943
        </fo:block>
        <fo:block font-family*"sans-serif" line-height-"48pt"
            font-sizP-"36pt"H>
            Distance (million miles):
            66.8
        </fo-block»

<fo:block font-family-"sans-$erif" line-height="48pt"
            font-size-"36pt" font-weight~"bold">
            Name:
            Earth
        </fo:block»
        <fo:block font-family-"sans-serif" line-height-"48pt"
            font-s1ze-"36pt">
            M^ss (Earth - 1):
            1
        </fo:block>
        <fo;Mock font-family="sans-serif" line-height-"48pt"
  . ,v|7    font-size-"36pt">
.   •       Day (Earth - 1):
            l</fo:block»
        <fo:block font-family-"sans-serif" line-height-"48pt"
            font-size="36pt">
            Radius On miles):
            2107
        </fo:block»
        <fo:block font- familyv'sans-serif" line~height-"48pt"
            font-size«"36pt">
            Density (Earth - 1):
            l</fO:blOCk>
        <fo:block font-family="sans-serif" line-height-"48pt"
            font-size="36pt">
            Distance (million miles):
            128.4
        </fo block>
    </fO:flow»
</fo:page-sequence»

</fo:root>
```

Итак, мы успешно создали planets.fo. Как нам теперь с его помощью создать форматированный файл PDF?

Создание форматированного документа

Чтобы обработать planets.fo и создать форматированный (formatted) документ, я воспользуюсь процессором FOP Джеймса Таубера Games Tauber), который был подарен им проекту Apache XML Project.

Главная страница процессора — http://xml.apache.org/fop; в настоящий момент FOP можно загрузить с http://xml.apache.org/fop/download.htmL Пакет FOP, включая документацию, поставляется в формате zip, поэтому сначала его нужно рас-

паковать FOP реализован как JAR-файл Java, fop.jar, здесь я буду использовать FOP версии 0.15.

FOP можно запустить из командной строки, используя класс Java, на момент написания книги называвшийся org.apache.fop.apps.CommandLine. Нужно предоставить разборщик XML — я буду использовать разборщик Xerces на Java в файле xerces.jar (он поставляется вместе с Xalan). Давайте рассмотрим, как в Windows, используя Java, при помощи FOP преобразовать planets.fo в planets.pdf: в данном случае я указываю переменной classpath ключом -cp включить файл xerces.jar, а также два необходимых JAR-файла, входящих в состав загружаемого пакета FOP — fop.jar и w3c.jar. (В этом примере предполагается, что все файлы fop.jar, xerces.jar и w3c.jar расположены в каталоге C:\planets; если это не так, укажите полные пути к ним.)

```
C:\planets>java -cp fop.jar:xerces.jar:w3c.jar
org.apache.fop apps.CommandLine planets.fo planets.pdf
```

Получившийся файл, planets, pdf, можно просмотреть в средстве чтения файлов PDF Adobe Acrobat Reader, как изображено на рис. 1.8. (Acrobat PDF Reader можно бесплатно загрузить с www.adobe.com/products/acrobat/readermain.html.) Документ planets.xml изображен на рисунке отформатированным в соответствии с таблицей стилей planetsPDF.xsl.

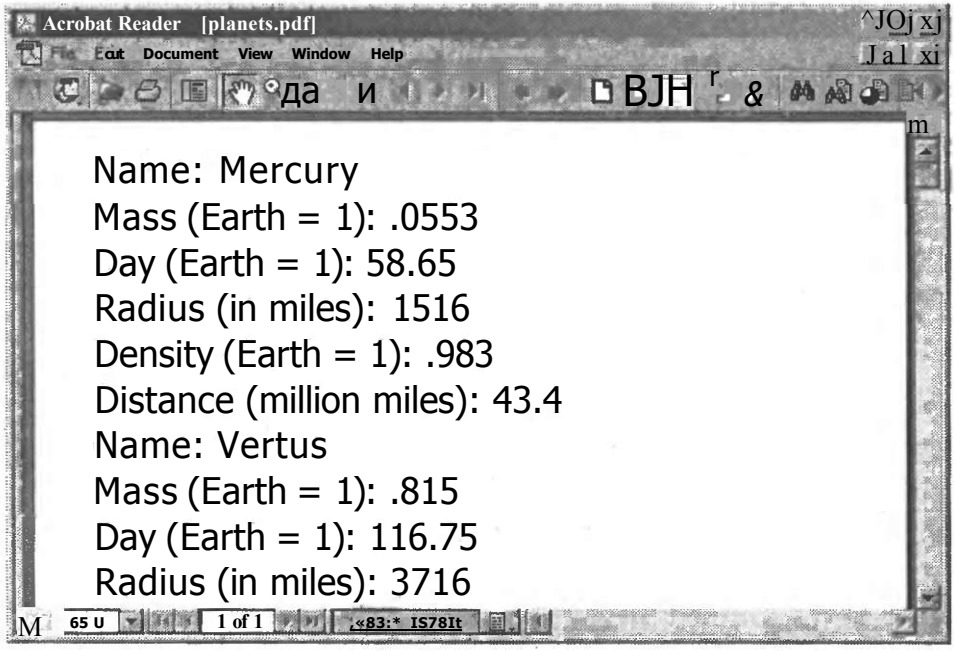


Рис. 1.8. Документ PDF, созданный при помощи форматирующих объектов

Формат PDF — хороший формат для вывода форматирующих объектов, хотя он и обладает рядом ограничений — например, он не способен обрабатывать

динамические таблицы, которые могут разворачиваться или сворачиваться по щелчку мыши, или интерактивные многоадресные ссылки (а ведь и те, и другие входят в спецификацию форматирующих объектов). Хотя в основных браузерах XSL-FO поддерживается довольно слабо, ожидается, что в будущем ситуация изменится, и браузеры будут поддерживать XSL-FO.

На этом я завершаю наш обзор. Далее в книге мы рассмотрим все вышеперечисленные темы по XSLT, а также вы познакомитесь с введением в XSL-FO. Теперь самое время начать подробно изучать XSLT, — прямо со следующей главы.

Создание и применение таблиц стилей



В предыдущей главе был представлен обзор XSLT. В этой главе мы начинаем детальную работу с ним. Мы собираемся научиться рассматривать документы в терминах XSLT, структурировать таблицу стилей XSLT и встраивать в документы таблицы стилей, В этой главе будет дано введение в шаблоны таблиц стилей, которые образуют основу таблиц стилей XSLT. Шаблоны реализуют фактические правила, которые вам требуется применить к данным; более подробно мы рассмотрим их в следующей главе.

Глава открывается систематическим рассмотрением таблиц стилей, и, несомненно, начинать следует с представления документа XML с точки зрения таблицы стилей.

Деревья и узлы

При работе с XSLT следует перестать мыслить в терминах документов и начать — в терминах деревьев. Дерево представляет данные в документе в виде множества узлов — элементы, атрибуты, комментарии и т. д. трактуются как узлы — в иерархии, и в XSLT структура дерева соответствует рекомендации XPath W3C (www.w3.org/TR/xpath). В данной главе мы рассмотрим деревья и узлы концептуально, а в главах 3 и 4 я дам формальное введение в XPath и его связь с XSLT. Выражения XPath, записываемые в терминах деревьев и узлов, используются для поиска данных в XML-документах.

В действительности, в соответствии с рекомендацией XSLT процессоры XSLT не обязаны уметь работать с документами; формально XSLT-преобразования принимают в качестве ввода исходное дерево и производят в качестве вывода результирующее дерево. В большинстве процессоров, тем не менее, существует дополнительная поддержка работы с документами.

Таким образом, с точки зрения XSLT документы представляют собой образованные из узлов деревья; XSLT распознает семь типов узлов:

- Корневой узел. Это самое начало документа. Этот узел представляет для процессора XSLT весь документ. Важно: не путайте корневой узел с корневым элементом, который также называется элементом документа (подробнее об этом мы поговорим позже в этой главе);
- Узел атрибута. Содержит значение атрибута после того, как были раскрыты ссылки на сущности и отброшены окружающие символы-разделители;
- Узел комментария. Содержит текст комментария, не содержащий символов <! и >;
- Узел элемента. Состоит из части документа, заключенной в открывающий и соответствующий ему завершающий теги, или единственный пустой элемент-тег, например <br/>;
- Узел пространства имен. Представляет объявление пространства имен. Обратите внимание: этот узел добавляется к каждому элементу, к которому применяется это пространство имен;
- Узел инструкции обработки. Содержит текст инструкции обработки, не содержащий символов <? и ?>. Скажем, объявление <?xml version-" 1.0"?> не является инструкцией обработки, несмотря на то, что выглядит таковой. Процессор XSLT отбросит его автоматически;
- Текстовый узел. Текстовые узлы содержат последовательности символов, То есть текст PCDATA. Текстовые узлы по умолчанию в XSLT подвергаются нормализации, то есть смежные текстовые узлы объединяются.

Как мы увидим в главе 7, для работы с деревьями и узлами используются выражения XPath. Выражение XPath возвращает единственный удовлетворяющий выражению узел, или множество узлов, если таких узлов несколько. XPath проектировался как средство навигации по деревьям — и, разобравшись с XPath, вы разберетесь с большей частью XSLT.

Важно помнить следующее: корневой узел дерева 5£SLT представляет весь документ. Это не то же самое, что корневой элемент. Взгляните, например, на следующий документ — в терминах XSLT корневой узел представляет документ г целиком, а корневым элементом является <library>:

```
<?xml version-"1.0"?>
<library>
   <book>
        <title>
            Earthquakes for Lunch
        </title>
        <tltle>
            Volcanoes for Dinner
        </title>
   </book>
</Hbrary>
```

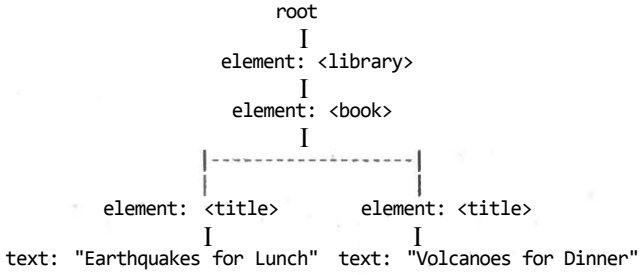
Термин *корневой элемент* (root element) прищел из рекомендации XML„ и, щн скольку его легко спутать с корневым узлом (root node;) XStjT„ пришедшим из <

рекомендации XPath, некоторые авторы, пишущие на тему XSLT, называют корневой элемент элементом документа. Очень жаль, что существует такое перекрытие терминов.

Вам следует также знать, что процессоры XSLT нормализуют текстовые узлы дерева, то есть объединяют два соседних текстовых узла в один большой текстовый узел для упрощения работы со структурой дерева документа. Поэтому, например, между двумя смежными узлами элементов никогда не будет более одного текстового узла, если изначально между этими узлами элементов располагался только текст.

В XSLT узлы могут иметь имена — так же, как *дочерние узлы* (child node) и *родительские узлы* (parent node). Иными словами, узлы элементов, атрибутов, пространств имен и инструкций обработки могут иметь имена; каждый узел элемента и корневой узел могут иметь дочерние узлы; и все узлы, за исключением корневого, имеют родителей.

Например, вот как выглядит рассмотренный нами ранее XML-документ в процессоре XSLT в виде дерева, состоящего из узлов:



Как видим, корневой узел расположен на самом верху дерева, за которым следует узел корневого элемента, ему соответствует элемент <library>. За ним следует узел <book>, у которого есть два дочерних узла <title>. Эти два узла **<title>** являются внуками элемента <library>. Родители, дедушки и прадедушки узла, назад до и включая корневой узел, являются *предками* (ancestor) элемента. Узлы, производные от узла (дети, внуки, правнуки и т. д.), называются его *потомками* (descendant). Узлы одного уровня называются *братьями* (sibling).

При помощи этой модели в виде дерева можно представить любой хорошо сформированный XML-документ. Но XSLT не ограничивается работой только с такими документами. В хорошо сформированных документах должен существовать один элемент, содержащий все остальные, но в соответствии с рекомендацией XSLT это не обязательно. В XSLT корневой узел может иметь столько детей, сколько их может иметь элемент, — например, несколько узлов элементов или текстовых узлов. Таким образом, XSLT может работать с фрагментами документа, д не только с хорошо сформированными документами.

ФРАГМЕНТЫ РЕЗУЛЬТИРУЮЩЕГО ДЕРЕВА

Помимо работы с фрагментами входного дерева, процессоры могут включать в вывод специальный тип данных, в XSLT 1.0 называемый фрагментом результирующего дерева (result tree fragment). Этот тип данных, однако, не был включен в рабочий проект XSLT 1.1 (см. главу 7), поэтому он, скорее всего, не будет входить в состав XSLT 2.0.

В действительности, рассмотренная только что диаграмма дерева не дает полной картины того, как она выглядит с точки зрения процессора XSLT. Я исключил один тип узлов, который вызывает большую путаницу при изучении XSLT — текстовые узлы, содержащие только символ-разделитель (whitespace). Теперь самое время заняться ими.

Символ-разделитель

Пример XML-документа, с которым мы до сих пор работали, выровнен так, чтобы показать иерархическую структуру его элементов:

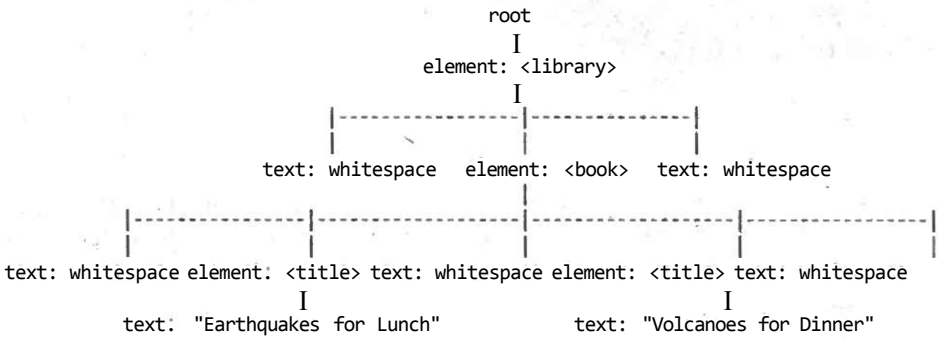
```
<?xml version»"1.0"?>
<library>
   <book>
        <title>
            Earthquakes for Lunch
        </title>
        <title>
            Volcanoes for Dinner
        </title>
   </book>
</library>
```

Однако, с точки зрения XSLT, символы-разделители, используемые для выравнивания элементов, в действительности являются текстовыми узлами. Это означает, что по умолчанию эти пробелы будут скопированы в выходной документ. Понимание принципов работы XSLT с разделителями всегда вызывает большую путаницу, поэтому мы кратко рассмотрим здесь, как обрабатывать символы-разделители, и подробно займемся этим в следующей главе.

В XSLT существует четыре символа-разделителя: пробел, возврат каретки, перевод строки и табуляция. Все эти символы трактуются как разделители. Таким образом, с точки зрения процессора XSLT, входной документ выглядит так:

```
<?xml version="1.0"?>
<library>
    <book>
        <title>
            Earthquakes for Lunch
        </title>
        <title>
            Volcanoes for Dinner
        </title>
... </book>
</library>
```

Все разделители между элементами трактуются в XSLT как текстовые узлы, содержащие символ-разделитель. Это означает, что в нашу диаграмму нам нужно добавить пять текстовых узлов с символом-разделителем: один перед элементом <book>, один после элемента <book>, и точно так же один перед элементом <title>, один после и один между элементами:

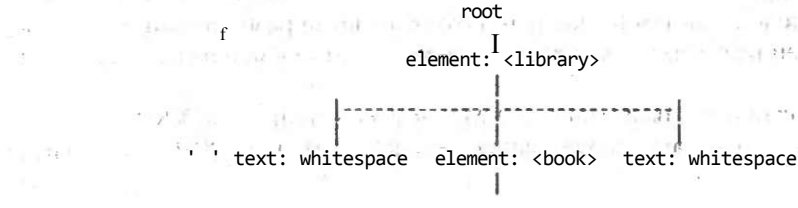


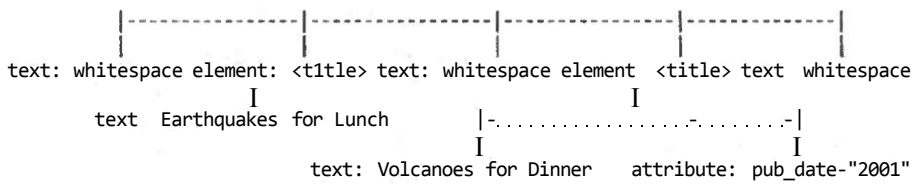
Такие узлы-разделители, как эти, представляют собой текстовые узлы, не содержащие ничего, кроме символа-разделителя. Поскольку процессоры XSLT по умолчанию сохраняют эти разделители, вас не должно удивлять их появление в результирующих документах. Такие дополнительные разделители обычно не представляют проблемы в документах HTML, XML и XHTML, и здесь в тексте результирующих документов я их не отображаю — для того, чтобы правильно показать выравниванием структуру документа. Мы рассмотрим, как процессоры XSLT могут удалять узлы-разделители из документов, а также как процессоры могут выравнивать результирующие документы. Заметьте, что текстовые узлы, содержащие символы, отличные от символов-разделителей, не считаются узлами-разделителями, поэтому они никогда не будут удалены из документов.

Следует отметить еще один момент: сами атрибуты трактуются как узлы. Хотя узлы-атрибуты не считаются дочерними узлами тех элементов, в которых они появляются, элемент считается их родительским узлом. (В этом отличие данной модели от модели XML DOM, в которой атрибуты не являются детьми и не имеют родителей.) Если я добавлю атрибут в такой элемент:

```
<?xml version-"1.0"?>
<library>
   <book>
        <title>
            Earthquakes for Lunch
        </t1tle>
        <title pub_date-"2001">
            Volcanoes for Dinner
        </title>
   </book>
</library»
```

то в дереве документа он отобразится следующим образом:





У каждого узла есть ряд установленных свойств, связанных с ним в XSLT. В следующем списке перечислены виды свойств, которые создатели процессоров XSLT отслеживают для каждого узла:

- Имя. Имя узла;
- Строка-значение. Текст узла;
- Базовый URI. Базовый URJ узла (XML-вариант URL);
- Дети. Список дочерних узлов; ноль, если детей нет;
- Родитель. Узел-родитель данного узла;
- Имеет атрибут. Определяет атрибуты узла элемента, если таковые имеются;
- Имеет пространство имен. Определяет узлы пространства имен узла-элемента.

При работе с деревьями следует принять во внимание еще одно соображение; процессоры XSLT работают поверх разборщиков XML, и так как правила для разборщиков XML и процессоров XSLT слегка различаются, это может привести к проблемам. В некоторых случаях данный аспект может быть важен, поэтому в следующем разделе он кратко рассмотрен.

Модель информационного множества против модели дерева XSLT

Разборщики XML передают только определенную информацию: как задается основной спецификацией информационного множества (Information Set) XML — которую можно найти по адресу www.w3.org/TR/xml-infoset, в то время как процессоры XSLT придерживаются модели дерева XSLT. Эти модели, а также элементы, считающиеся в них важными, различны, что может привести к проблемам.

Вот, например, два элемента XML, входящих в основное информационное множество, но невозможные в XSLT: примечания и пропущенные ссылки на сущности (ссылки на сущности, которые разборщик XML предпочел не раскрывать). На практике это означает, что даже если разборщик XML передал информацию для этих пунктов, процессор XSLT ничего не сможет с ней сделать. Однако примечания используются редко, и только отдельные разборщики XML генерируют пропущенные ссылки на сущности, поэтому это не является значимой проблемой.

С другой стороны, разборщики XML *могут* удалить из XML-документов комментарии, о чем вам следует знать, так как в модели XSLT комментарии включаются.

Кроме того, информация об объявлениях DTD *не* передается от разборщика XML в процессор XSLT (возможно, потому, что W3C планирует более широко использовать схемы XML в XSLT 2.0, хотя пока еще нет официального механизма связывания схем XML с документами XML). Как правило, это не образует проблемы, поскольку проверка документа XML входит в задачи разборщика XML, за исключением одного случая: когда атрибут объявлен с типом ID. В XML с типом ID можно объявить атрибут с любым именем, поэтому процессор XSLT не знает, какие атрибуты принадлежат к этому типу, если только у процессора нет доступа к DTD. Это важно при использовании таблиц стилей, встроенных в документы XML, поскольку в этом случае процессор XSLT должен быть способен распознать, в каком элементе документа содержится таблица стилей, нужная для преобразования документа. В этом случае некоторые процессоры XSLT, например, Saxon, идут дальше рекомендации XSLT и проверяют DTD, если они есть, чтобы узнать, какие атрибуты принадлежат к типу ID.

Еще несколько моментов могут представлять для вас интерес. Например, модель обработки XSLT открывает доступ к префиксам пространств имен во входном дереве, но предоставляет очень небольшие возможности управления ими в выходном дереве, где они обрабатываются автоматически. К тому же, в модели обработки XSLT для каждого узла в дереве определяется базовый URI, представляющий собой URI внешней сущности, от которой был произведен узел. (В рабочем проекте XSLT 1.1 эти возможности были расширены для поддержки спецификации XML Base, как вы увидите ближе к концу этой главы.) Однако в информационном множестве XML базовые URI считаются второстепенными, а это означает, что разборщик XML может не передать эту информацию в процессор XSL.

В общем, вам следует знать, что процессоры XSLT читают документы XML при помощи разборщиков XML, и что совместная работа этих программных единиц не бесшовна. Если вы обнаружите, что некоторая необходимая информация при преобразовании XSLT пропала, следует вспомнить этот момент. Фактически различие между информационным множеством XML и моделью дерева XSLT — это одна из проблем, которую призван исправить XSLT 2.0. Помимо прочего, в XSLT 2.0 должно быть проще восстанавливать ID и ключевую информацию из исходного документа, а также восстанавливать информацию из объявления XML исходного документа — например, версию XML и кодировку.

Работа с элементами XSLT

Для того чтобы создавать таблицы стилей XSLT, вы должны хорошо знать элементы XSLT, такие как <xsl:template> и <xsl:stylesheet>. Эти элементы поддерживают большое число атрибутов, и W3C выработал ряд формальных определений типов данных, которые можно присваивать этим атрибутам. Вот ряд определений XSLT; их необходимо знать:

- NCNameChar. Буква, цифра, точка, дефис или символ подчеркивания;
- NCName. Буква или символ подчеркивания, за которым (не обязательно) следуют данные типа NCNameChars. То есть это имя XML, не содержащее двоеточий (определение имен XML приводится в главе 1);

- QName. Полностью определенное (qualified) имя. Оно формируется из префикса (должен принадлежать к типу NCName), за которым следуют двоеточие и локальная часть (которая также должна быть типа NCName);
- NameTest. Имя (например, -«book») или обобщенное имя с символами подстановки (как, например, «book*» или «*»).

Теперь можно начать создавать таблицы стилей XSLT. Первым элементом будет элемент для связывания таблиц стилей с документами XML <?xst:stylesheet?>.

Инструкция обработки <?xsl:stylesheet?>

Когда у нас есть таблица стилей XSL, которую нужно применить к документу XML, требуется каким-то образом связать эту таблицу стилей с документом, — для чего часто используется инструкция обработки <?xsl:stylesheet?>. У этой инструкции есть несколько возможных атрибутов:

- href (обязательный). URI таблицы стилей. Может быть полный URI, либо фрагмент из #new_style. Устанавливается в URI;
- type (обязательный). Тип MIME таблицы стилей. Обычно «text/xml» или «application/xml». Для Internet Explorer используйте «text/xsl». Устанавливается в допустимый тип MIME;
- title (необязательный). Заголовок позволяет различить несколько различных элементов <?xsl:stylesheet?>. Некоторые разборщики XML позволяют задать, какой из них использовать. Устанавливается в строковое значение;
- media (необязательный). Описание средств вывода, например «print» или «aural»(звуковой). Устанавливается в одно из значений, перечисленных в спецификации W3C HTML 4.0;
- charset (необязательный). Устанавливает кодировку символов. Заметьте, что таблицы стилей XSLT устанавливают свои собственные кодировки, поскольку они являются документами XML, так что практической пользы этот атрибут не имеет. Устанавливается в кодировку символов — например в «UTF-8»;
- alternate (необязательный). Принимает либо значение «yes» (да), которое является признаком альтернативной таблицы стилей, либо «по» (нет), означающее предпочитаемую таблицу стилей.

Инструкция обработки <?xsl:stylesheet?> добавляется в документ XML, а не в таблицу стилей, и указывает процессору XSLT, какую таблицу стилей применять к данному документу.

На практике <?xsl:stylesheet?> используется главным образом с браузерами, поскольку для отдельных процессоров таблица стилей, как правило, задается непосредственно — как, например, при использовании процессора XSLT Oracle:

```
C:\planets>java oracle.xml.parser.v2.oraxsl planets.xml planets.xsl planets.html
```

Возможно, вас это удивит, но <?xsl:stylesheet?> не является частью рекомендации XSLT. У этой инструкции обработки существует своя собственная рекомендация только для нее одной, которую можно найти по адресу www.w3c.org/TR/xml-stylesheet. Помимо прочего, это означает, что процессоры XSLT не обязаны поддерживать данную инструкцию обработки, и большинство отдельных процессоров не поддерживают ее.

Ниже приведен пример. В главе 1 мы рассмотрели файл planets.xml — хорошо сформированный документ XML, содержащий данные о трех планетах: Меркурии, Венере и Земле. Можно применить инструкцию обработки <?xml-stylesheet?> к planets.xml для задания используемой таблицы стилей XSLT. В инструкции следует установить тип атрибута в «text/xml» (W3C также позволяет «application/xml», а Internet Explorer требует «text/xsl») и атрибут href — в URI таблицы стилей XSLT, например planets.xsl:

Листинг 2.1. planets.xml

```
<?xml version-"l 0"?>
<?xml-sty!esheet type-"text/xml"  href«"planets.xsl"?>
<PLANETS>

    <PLANET>
        <NAME>Mercury</NAME>
        «MASS UNITS-"(Earth - 1)">.0553</MASS>
        <DAY UNITS-"days">58.65</DAY>
        <RADIUS UNITS-"m1les">l5l6</RADIUS>
        «DENSITY UNITS*"(Earth - 1)"> 983«/DENSITY>
        «DISTANCE UNITS-"million m1les"<@062>43.4</DISTANCE><!--B перигелии-»
    «/PLANET»

    «PLANET»
        «NAME»Venus«/NAME>
        «MASS UNITS-"(Earth - 1)">.815</HASS>
        <DAY UlfITS-"days">ll6,75*/DAY>
        «RADIUS UNITS4*"mi>es"»3716</RADIUS>
        «DENSITY UNITS-"(Earth - 1)">.943</DENSITY>
        «DISTANCE UNITS-"million miles">66.8</DISTANCEx!--B перигелии--»
    ' «/PLANET»

    <PLANET>
        <NAHE>Earth«/NAME>
        «MASS UNITS-"(Earth = 1)">1«/MASS>
        «DAY UNItS-"days">l</DAY>
        «RADIUS LNºIITS-"m1les">2107«/RADIUS>
        «DENSITY UNITS-"(Earth - 1)">1«/DENSITY>
        «DISTANCE UNITS-"million miles">128.4</DISTANCE>«!--B перигелии--»
    «/PLANET»
        •          )>                    ij-.Ji'i.i (Й                    > •!
«/PLANETS»
```

Йот Как следует работать с элементом <?xml-stylesheet?>; теперь можно начинать писать саму таблицу стилей. Я сделаю это, создав файл planets.xsl.

Элемент <xsl:stylesheet>

Таблицы стилей XSL начинаются с объявления XML, так как они являются хорошо сформированными документами XML, — поэтому planets.xsl начинается с такого же объявления:

```
<?xml version-"! 0"?>
    .
    .
    .
```

Однако объявление XML немедленно исключается процессором XSLT, и мы не будем на нем останавливаться в нашем обсуждении XSLT. Первый только XSL-элемент таблицы стилей XSL — это элемент <xsl:stylesheet> (не путайте его с инструкцией обработки <?xml-stytesheet?> из XML документа). Встречались и возражения против такого названия элемента: обычно он используется в преобразованиях XSLT, поэтому W3C позволяет в то же время ссылаться на этот элемент как на <xsl:transform>.

Вот пример использования этого элемента:

```
<?xml vers1on-"l.0"?>
<xsl:stylesheet version-'l.O"
xmlns:xsl-"http;//www.w3.org/1999/XSL/Transform">
    .
    .
    .
```

В следующем списке перечислены атрибуты элемента <xsl:stylesheet>:

- id (необязательный). Идентифицирует таблицу стилей. Устанавливается в имя XML;
- version (обязательный). Определяет версию XSLT, необходимую для работы с таблицей стилей. Сейчас это значение обычно равно «1.0». Хотя можно установить это значение и в «1.1»; так как XSLT 1.1 не выйдет из стадии рабочего проекта, это значение, вероятно, не будет «корректным» значением с точки зрения W3C;
- extension-element-prefixes (необязательный). Задает расширения в таблице стилей, используемые для идентификации элементов расширения. Принимает значение списка имен NCName, разделенных символами-разделителями;
- exclude-result-prefixes (необязательный). Задает пространства имен таблицы стилей, которые не будут копироваться в выходные данные (если только они явно не используются в выходном документе). Принимает значение списка имен NCName, разделенных символами-разделителями.

Содержимое этого элемента может содержать любой из следующих элементов XSL высокого уровня: <xsl:attribute-set>, <xsl:decimal-format>, <xsl:import>, <xsl:include>, <xsl:key>, <xsl:namespace-alias>, <xsl:output>, <xsl:param>, **<xsl:preserve-**space:», <xsl:strip-space>, <xsl:template> или <xsl:variable>. XSLT добавляет к списку <xsl:script>.

Заметьте, что атрибут id этого элемента можно использовать в том случае, когда требуется сослаться на определенную таблицу стилей (заметьте также, что в этом случае необходим процессор XSLT, который умеет читать объявления DTD или схемы XML).

Атрибут задания версии обязателен, в данный момент его можно установить в «1.0». Можно также установить его в «1.1», определяя рабочий проект XSLT 1.1; однако, так как XSLT 1.1 не собирается выходить из стадии рабочего проекта, значение «1.1», скорее всего, не будет считаться корректным значением этого атрибута в долгосрочной перспективе. Везде в книге я буду устанавливать этот атрибут в значение «1.0», поскольку это текущая рекомендация XSLT W3C, а версия 1.1, видимо, останется только на стадии рабочего проекта. Как говорилось в главе 1, W3C также выпустил требования для XSLT 2.0, которая и будет следующей версией.

ПРОЦЕССОРЫ XSLT И СОВМЕСТИМОСТЬ ВПЕРЕД

Если процессор XSLT не распознает версию XSLT, то, в соответствии с утверждениями W3C, процессор должен считать, что все новые элементы входят в новую версию XSLT, и не завершать работу, — это W3C называет совместимостью вперед (forward compatibility). Все нераспознанные элементы не должны отвергаться процессором, если только таблица стилей не попытается создать экземпляр этого элемента и не найдет дочернего элемента <xsl:fallback> (см. главу 5). Таким образом, вы можете установить версию XSLT в 2.0 даже в процессорах XSLT, написанных для XSLT 1.0, и это не приведет к проблемам, если вы не будете использовать возможности, существующие только в XSLT 2.0. (Исключением является MSXML3, который в данный момент генерирует ошибки, если установить значение версии, отличное от 1.0.)

Пространство имен XSL

Заметьте, что элементы XSLT, такие как <xst:stylesheet>, используют Префикс пространства имен (namespace) xsl, который теперь, после стандартизации XSLT, всегда установлен в «http://www.w3,org/1999/XSL/Transform». Это пространство имен обычно устанавливается с атрибутом xmlns (это не атрибут XSL, а, скорее, атрибут любого элемента XML) в корневом элементе таблицы стилей, <xsl:stylesheet>:

```
<xsl:stylesheet version-"1.0" xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">
.
.
.
```

Таблицы стилей XSL используют свое собственное пространство имен, чтобы избежать конфликта с другими используемыми в таблице элементами. Например, вам может потребоваться указать, какая таблица стилей использовалась для преобразования документа; в таком случае вы можете создать свой собственный элемент <stylesheet> — это не создает проблемы, поскольку новый Элемент не будет конфликтовать с элементом XSL <xsl:stylesheet>.

В таблицах стилей для элементов XSLT обычно используется префикс пространства имен xsl. Формально можно использовать для элементов XSLT любой префикс пространства имен (или даже вообще никакого), но практически но-

всюду применяется xsl, поскольку это пространство имен фигурирует в рекомендации XSLT.

На практике это означает, что все элементы XSLT, с которыми мы станем работать, будут начинаться с префикса пространства имен xsl, как в <xsl:stylesheet>. Следовательно, надо начинать таблицу стилей XSLT с элемента <xsl:stylesheet>. (Есть одно исключение: в «упрощенных» таблицах стилей этот элемент опускается, как мы увидим позже в этой главе.)

Как и любое другое приложение XML, XSLT имеет хорошо разработанный набор правил допустимости таблицы стилей XSLT. W3C даже определил для XSLT псевдо-DTD, где перечислены все правила синтаксиса, — эти объявления DTD приведены в приложении A, которое послужит вам хорошим справочником в случае затруднений с синтаксисом XSLT. Как показано в приложении A, элемент <xsl:stylesheet> может содержать несколько других элементов XSLT, называемых элементами верхнего уровня.

ОБРАБОТКА ПРОСТРАНСТВ ИМЕН ПО УМОЛЧАНИЮ

Работа с пространствами имен может оказаться не слишком простой. Например, иногда присваивают пространство имен по умолчанию всему элементу <xsl:stylesheet>, как в <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/ 1999/XSL/Transform" xmlns="mydefault">, ожидая, что <xsl template match="mysymbol"> выберет mysymbol в пространстве имен "mydefault" исходного документа. Однако это не будет работать. Обработка пространств имен по умолчанию, как в таком случае, — одна из проблем, которую призван решить XSLT 2.0.

Элементы таблиц стилей высокого уровня

В XSL определен ряд элементов высокого уровня, которые могут быть прямыми дочерними элементами <xsl:stylesheet>:

- <xsl:attribute-set>;
- <xsl:decimal-format>;
- <xsl:import>;
- <xsl:include>;
- <xsl:key>;
- <xsl:namespace-alias>;
- <xsl:output>;
- <xsl:param>;
- <xsl: preserve-space»;
- <xsl:strip-space>;
- <xsl:template>;
- <xsl:variable>.

Рабочий проект XSLT 1.1 добавляет еще один элемент высокого уровня:

- <xsl:script>, >'

В книге мы поработаем со всеми этими официальными элементами XSLT высокого уровня.

Заметьте, что, в дополнение ко всем элементам высокого уровня, вы можете также повсюду использовать обычные комментарии XML, поскольку таблицы стилей XSLT являются также документами XML:

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xs1-"http://www.w3.org/1999/XSL/Transform"»
    <!-Этот шаблон содержит все элементы PLANETS -•>
    <xsl:tempiate match-"/PLANETS"»
        <HTML>
            <HEAD>
            .
            .
            .
```

В дополнение к официальным элементам высокого уровня, некоторые процессоры XSLT определяют свои собственные элементы высокого уровня, использующие пространства имен, отличающиеся от пространства имен XSL. Функциональность таких элементов полностью определяется разработчиком процессора XSLT.

Вы также можете определять собственные элементы высокого уровня, называемые «элементами высокого уровня, определенными пользователем». Эти элементы должны иметь пространство имен, отличное от XSL и элементов, определенных разработчиком процессора. Процессор XSLT игнорирует определенные пользователем элементы высокого уровня; но, поскольку вы способны сами получить доступ ко всему документу при помощи функции документа (как мы увидим в главе 8), вы можете самостоятельно работать с этими элементами.

Из всех элементов высокого уровня самым популярным является элемент <xsl:template>:

```
<?xml version-"1.0">
<xsl:stylesheet version-"1.0" xmlns:xsl-"http://www w3.org/1999/XSL/Transform">
    <xsl:template»
    .
    .
    .
    </xsl:template»
    .
    .
    .
```

Создание шаблонов и работа с ними — суть XSLT, и сейчас мы рассмотрим этот элемент, продолжая создавать таблицу planets.xsl

Элемент <xsl:template>

Шаблоны XSL позволяют задать способ выполнения вашего преобразования. Каждый элемент <xsl:template> устанавливается для выбора одного узла (кото-

рый может содержать другие узлы) или ряда узлов в исходном документе, а также точного указания способа преобразования этого узла.

В следующем списке перечислены атрибуты <xsl:template>:

- match (необязательный). Задает шаблон, выбирающий обрабатываемые узлы. Устанавливается в допустимый шаблон;
- name (необязательный). Содержит имя шаблона, что разрешает его вызов. Если вы не используете этот атрибут, вы должны использовать атрибут match. Принимает значение типа QName;
- priority (необязательный). Положительное или отрицательное целое или действительное число, задающее приоритет шаблона. Используется, когда один и тот же узел удовлетворяет нескольким шаблонам. Устанавливается в число;
- mode (необязательный). Если вы применяете <xsl:apply-templates> к множеству узлов, будут использоваться только шаблоны с совпадающим режимом (mode). Принимает значение типа QName.

Каждый такой элемент <xsl:template> называется правилом (rule). В общем случае элемент <xsl:template> может содержать ноль или более элементов <xsl:param> (как мы увидим в главе 9), за которыми следует тело шаблона, задающего способ осуществления преобразования.

Тела шаблонов

Шаблоны формируются по жестко заданным правилам. Они способны содержать элементы <xsl:param>, за которыми следует тело шаблона; в последнем могут содержаться данные PCDATA, инструкции XSLT, элементы расширения и элементы буквального результата.

Инструкции XSLT

В теле шаблона может присутствовать ряд элементов XSLT, называемых инструкциями:

- <xsl:apply-imports>;
- <xsl:apply- templates»;
- <xsl:attribute>;
- <xsl:call-template>;
- <xsl:choose>;
- <xsl:comment>;
- <xsl:copy>;
- <xsl:copy-of>;
- <xsl:element>;

- <xsl:fallback>;
- <xsl:for-each>;
- <xsl:if>;
- <xsl:message>;
- <xsl:number>;
- <xsl:processing-instruction>;
- <xsl:text>;
- <xsl:value-of>;
- <xsl:variable>.

В теле шаблона больше никакие элементы XSLT непосредственно появиться не могут. Как вы увидите в главе 9, элемент <xsl:param> может появиться в шаблоне перед телом, но он не называется инструкцией XSLT. В шаблонах могут также появляться и другие элементы XSLT, такие как <xsl:sort>, <xsl:otherwise> и <xsl:with-param>, но только в определенных местах, поэтому W3C не называет их инструкциями. Далее в книге мы рассмотрим, как использовать каждую из этих инструкций.

Элементы расширения

Элементы расширения рассматриваются в главе 5; эти элементы, расширяющие XSLT, определяются пользователем или процессором XSLT* Во многих процессорах XSLT были определены собственные расширения — и это стало одной из причин, по которым W3C представил рабочий проект XSLT 1,1, где механизм расширений был более регламентирован. Вероятно, эта функциональность будет включена в XSLT 2.0.

Элементы буквального результата

Если элемент в теле шаблона не является инструкцией XSL или элементом расширения, процессор XSLT должен рассматривать его в качестве элемента буквального результата. Это означает, что элемент должен трактоваться буквально и копироваться в результирующее дерево (то есть копироваться в выходное дерево, созданное процессором XSLT).

Например, в следующем теле шаблона элемент <ID> является элементом буквального результата, который будет скопирован в выходной документ:

```
<xsl:tempiate match-"RADIUS">
    <TD>RADIUS</TD>
</xsl:template»
```

Элементы буквального результата могут сами по себе иметь содержимое, которое затем трактуется как еще одно тело шаблона и разбирается процессором XSLT. Позже в настоящей главе мы посмотрим, как работает этот механизм.

Элементы буквального результата могут также обладать атрибутами, которые интерпретируются процессором XSLT. Например, при помощи атрибута version

можно указать, что все элементы XSLT внутри элемента буквального результата должны быть элементами XSLT версии 1.0:

```
<xsl template match="RADIUS">
    <TD xsl:versi on="1.0">RADIUS</TD>
</xsl:template»
```

В следующем списке перечислены все возможные атрибуты элемента буквального результата:

- Attribute Value Templates, шаблон значений атрибута (необязательный). Любые выражения XPath в фигурных скобках вычисляются, и результат в виде строкового значения копируется в атрибут в результирующем дереве. Устанавливается в шаблон значений атрибута (см. главу 3);
- xsl:exclude-result-prefixes (необязательный). Определяет, какие пространства имен не будут скопированы в результирующее дерево. Принимает значения списка префиксов пространств имен, разделенных символами-разделителями;
- xsl:extension-element-prefixes (необязательный). Заставляет процессор XSLT трактовать дочерние элементы элемента буквального результата в перечисленных пространствах имен как элементы расширения, а не элементы буквального результата;
- xsl:use-attribute-sets (необязательный). Атрибуты в перечисленных наборах атрибутов добавляются в элемент буквального результата и копируются в результирующее дерево. Принимает значение списка QName, идентифицирующих поименованные элементы <xsl:attribute-set>;
- xslrversion (необязательный). Устанавливает версию элементов XSL в элементе буквального результата. Принимает численное значение.

Теперь можно начать работать с этой информацией.

Совпадающие элементы в шаблонах

Для указания того, с каким узлом или узлами вы хотите работать в шаблоне, в XSLT имеются разнообразные способы поиска или выбора узлов. Следует установить атрибут match элемента <xsl:template> в *образец* (pattern), определяющий имя узла или узлов, с которыми вы хотите работать. В главе 3, посвященной шаблонам, будет показано, как создавать образцы. Например, образец «/» соответствует корневому узлу; образец «*» задает любой узел элемента; образцу «PLANET» удовлетворяют все узлы элемента <PLANET> и т. д.

Для начала я создал короткий пример, заменяющий корневой узел, — а следовательно, и весь документ — на HTML-странице. Первое, что Я сделал, — создал шаблон с элементом <xsl:template>, установив атрибут match в образец для совпадения «/»:

```
<?xrtl version-"!„0">
<xsl:stylesheet version-"1.0" xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">
    <xs1:tempiate match-"/"»
```

```
    .
    .
    </xsl:template*
</xsl:stylesheet»
```

Когда устанавливается соответствие с корневым узлом, шаблон применяется к самому узлу. В данном случае (листинг 2.2) я хочу заменить корневой узел документом HTML, поэтому я только включу этот документ HTML непосредственно в качестве содержимого элемента <xsl:template>.

Листинг 2.2. Простейшее преобразование

```
<?xml version-"]..0"?>
<xsl:stylesheet version-"l 0" xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

    <xsl:tempiate match-"/">
        <HTML>
            <HEAD>
                <TITLE>
                    A trivial transformation
                </TITLE>
            </HEAD>
            <BODY>
                This transformation has replaced
                the entire document.
            </BODY>
        </HTML>
    </xsl:template»

</xsl:stylesheet»
```

Результат: при помощи элемента <xsl:template> я установил правило в таблице стилей. Когда процессор XSL считывает документ, первым узлом является корневой узел. Это правило находит данный корневой узел, и от того процессор XSL копирует литералы в результирующее дерево, что даст нам HTML doc и заменит его документом HTML, генерируя следующий результат:

```
<HTML>
    <HEAD>
        <TITLE>
            A trivial transformation
        </TITLE>
    </HEAD>
    <B0DY»
        This transformation has replaced
        the entire document.
    </B0DY>
</HTML>
```

Рассмотренный пример иллюстрирует первое, устаревшее преобразование. Была создана простая таблица стилей с единственным элементом <xsl:template>, который содержит только элемент буквального результата. Все, что сделано в примере, — замена всего документа XML на документ HTML, что не очень впечатляет. Далее мы увидим, как работает рекурсивная обработка с использованием элемента <xsl:apply-templates>.

Элемент <xsl:apply-templates>

В уже написанном нами основном шаблоне корневой узел искался по выражению "/" и заменялся на элемент буквального вывода. Однако, когда мы ищем корневой узел, обычно нам нужно обработать и всю оставшуюся часть документа, что мы сделаем при помощи элемента <xsl:apply-templates>.

В следующем списке перечислены атрибуты этого элемента:

- select (необязательный). Набоор обрабатываемых узлов. Если атрибут опущен, автоматически обрабатываются все потомки узла. Устанавливается в выражение;
- mode (необязательный). Устанавливает режим обработки. К этому узлу применяются правила шаблона с режимом выбора. Принимает значение типа QName.

Элемент <xsl:apply-templates> может содержать ноль или более элементов <xsl:sort>, или ноль или более элементов <xsl:with-param>.

В следующем примере шаблон ищет корневой узел и замещает его элементом буквального результата <HTML>:

```
<?xш1 version="1.0">
<xsl:stylesheet version-"1.0" xmlns:xsl="http://ww.w3.org/1999/XSL/Transform">
    <xsl:tempiate match-"/">
        <HTML>
        </HTML>
    </xsl:template»
```

С другой стороны, мы только нашли корневой узел, а дерево данных planets.xml имеет ряд узлов под корневым узлом:

```
<?xml version-"1.0"?>
<?xml-stylesheet type="text/xml" href-"planets.xsl"?>
<PLANETS>

    <PLANET>
        <NAME>Mercury</NAME>
        <MASS UNITS»"(Earth = 1)">.0553</MASS>
        <DAY UNITS-"days">58.65</DAV>
        <RADIUS UNITS="miles">1516</RADIUS>
        <DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
        «DISTANCE UNITS-"million miles">43.4</DISTANCE><!--В перигелии-->
    </PLANET>
```

Для обработки не только одного корневого узла можно использовать <xsl:af)ply-templates>, добавив этот элемент следующим образом:

```
<?xml version-"1.0">
<xsl: styl esheet version-"!.0" xml ns: xsl-"http: //ww. w3. org/i999/XSmfansforrrv">'
```

```
<xsl:tempi ate match-"/">
    <HTML>
        <xsl:apply-templates/»
    </HTML>
</xsl:template»
    .
    .
    .
```

Этот элемент дает указание процессору XSLT просмотреть все дочерние узлы корневого узла и попытаться найти шаблон, которому эти узлы удовлетворяют. Например, вам может потребоваться заменить все элементы <PLANET> на <P>Planet</P>. Элементы <PLANET> — дочерние узлы элемента <PLANETS>, поэтому сначала я добавил новый шаблон для <PLANETS>, что говорит процессору о том, что следует продолжать поиск дочерних узлов:

```
<?xml version-"1.0">
<xsl: styl esheet vers ion-"1.0" xmlns:xsl-"http //wwww3.org/1999/XSL/Transform"»

    <xsl template match="/">
        <HTML>
            <xsl:apply-templates/»
        </HTML>
    </xsl:tempi ate»

    <xsl .-template match-"PLANETS"»
            <xsl:apply-tempiates/»
    </xsl:tempiate»
        .
        .
        .
```

Теперь можно добавить еще один шаблон для следующего уровня, включающего элементы <PLANET>. В этом случае я просто заменю каждый элемент <PLANET> элементом буквального результата <P>Planet</P> (листинг 2.3).

Листинг 2.3. Использование <xsl:apply-templates/>

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0" xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"»

    <xsl:template match-"/"»
        <HTML>
            <xsl:apply-tempiates/»
        </HTML>
    </xsl:tempiate»

    <xsl:tempi ate match-"PLANETS"»
            <xsl:apply-templates/»
    </xsl:tempi ate»

    <xsl:tempi ate match-"PLANET"»
        <P>
            Planet
        </P>
    </xsl:tempi ate»

</xsl:stylesheet»
```

Вот результат применения этой таблицы стилей:

```
<HTML>
    <P>
        Planet
    </P>
    <P>
        Planet
    </P>
    <P>
        Planet
    </P>
</HTML>
```

Как видите, от элемента <PLANETS> ничего не осталось. Все, что осталось, — три элемента буквального результата <P>Planet</P>, которые заменили три элемента <PLANET>.

ПРОПУСК АТРИБУТА SELECT

Если опустить атрибут select, будут обрабатываться только дочерние узлы текущего узла, не включающие атрибуты или узлы пространств имен, так как они не считаются дочерними. Если вы хотите обрабатывать узлы этих видов, необходимо использовать атрибут select, как будет рассмотрено в главе 3.

Все это занимательно, но малопригодно практически. Было бы гораздо лучше, если бы мы, например, могли обратиться к фактическому значению каждого элемента (такому как имя каждой планеты) и работать с этими данными. И это, конечно, возможно.

Доступ к значениям узлов

Получить доступ к значению узла можно при помощи элемента <xsl:value-of>, у которого есть два возможных атрибута:

- select (обязательный). Выходное значение. Устанавливается в выражение;
- disable-output-escaping (необязательный). Указывает, что символы, такие как >, будут отправляться в выходной поток как есть, не изменяясь на &gt. Значения этого атрибута: yes или по.

Элемент <xsl:value-of> всегда пуст.

При помощи атрибута select можно указать, значение какого узла требуется получить. Например, вам может потребоваться значение узла <NAME> в каждом элементе <PLANET>, то есть текст, заключенный в этом узле. Это можно сделать следующим образом (листинг 2.4).

Листинг 2.4. Использование <xsl:value-of>

```
<?xml version-"1.0">

<xsl:stylesheet version-"1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <xsl:template match-"/">
```

продолжение &

Листинг 2.4 (продолжение)

```
        <HTML>
            <xsl:apply-templates/>
        </HTML>
    </xsl:template»

    <xsltemplate match-"PLANETS">
        <xsl:apply-templates/»
    </xsl:template»

    <xsl:tempiate match«"PLANET">
        <P>
            <xsl:value-of select-"NAME"/»
        </P>
    </xsl -.template»

</xsl:stylesheet»
```

Значение узла, содержащего текст, будет просто текстом, поэтому результат применения этой таблицы стилей к planets.xml следующий:

```
<HTML>
    <P»Mercury</P>
    <P»Venus</P>
    <P»Earth</P>
</HTML»
```

АТРИБУТ DISABLE-OUTPUT-ESCAPING

Атрибут disable-output-escaping элемента <xsl:value-of> более подробно рассмотрен в главе 3.

Предположим, нам нужно осуществить нечто более сложное — например, преобразовать данные из planets.xml в HTML-таблицу в новом файле planets.html (рис. 2.1), как мы видели в главе 1. Теперь это можно сделать при помощи <xsl:value-of>.

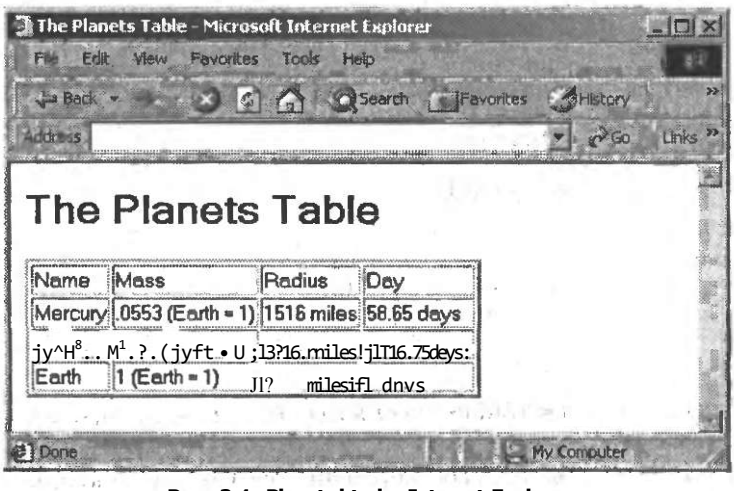


Рис. 2.1. Planets.html в Internet Explorer

Здесь важно учесть один момент. В planets.xml формально не задан порядок элементов <MASS>, <RADIUS>, <DAY> и <DISTANCE>, однако важно, чтобы эти элементы обрабатывались в определенном порядке в соответствии с заголовками таблицы. Поэтому я буду использовать элементы <xsl:value-of> в том порядке, в котором они требуются в таблице HTML.

Таким образом, чтобы создать HTML-таблицу, изображенную на рис. 2.1, я сначала ищу элемент <PLANETS> и затем заменяю его на HTML для создания самой HTML-таблицы. Элемент <PLANETS> — дочерний элемент корневого узла, и поскольку на корневой узел можно сослаться через «/», на элемент <PLANETS> можно сослаться непосредственно через "/PLANETS", без необходимости предварительно применять шаблон для корневого узла. Это пример выражения XPath, большое число подобных мы также увидим в главе 4.

Ниже приведен пример того, как я начал создавать HTML-таблицу путем выбора элемента <PLANETS> непосредственно как "/PLANETS" — заметьте, что для применения шаблонов к любым дочерним узлам <PLANETS> я использовал <xsl:apply-templates>:

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmln^:xsl-"http://www.w3.org/1999/XSL/Transforni">

    <xsl:template match-"/PLANETS"»
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <BODY>
                <H1>
                    The Planets Table
                </H1>
                <TABLE BORDER«"2">
                    <TR>
                        <TD>Name</TD>
                        <TD>Mass</TD>
                        <TD>Radius</TD>
                        <TD>Day</TD>
                    </TR>
                    <xsl:apply-templates/»
                </TABLE>
            «/BODY»
        </HTML>
    </xsl:tempiate»
```

Каждый дочерний узел <PLANET> имеет дочерние узлы <NAME>, <MASS>, <RADIUS> и <DAY>, и я хочу обраб; бывать их именно в таком порядке — для гого чтобы они добавлялись в HTML-таблицу в соответствии с заголовками таблицы. Для задания порядка их обработай "я поместил элементы <xsl:value-of> (листинг 2.5).

Листинг 2.5. planets.xsl

```
<?xml version-"1.0"?>
<xsl:stylesheet version«"1.0*' xmlns:xsl="http://www.w3.org/1999/XSL/Trdnsforni">

    <xsl:tempi ate match~'7PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <B0DY> '
                <H1>
                    The Planets Table
                </Hl>
                <TABLE BORDER-"2">
                    <TR>
                        <TD>№me</TD>
                        <TD>Mass</TD>
                        <TD>Rad1us</TD>
                        <TD>Day</TD>
                    </TR>
                    <xsl:apply-tempiates/>
                </TABLE>
            </BODY>
        </HTHL>
    </xsl :terrplate>

    <xsl:tempiate match-"PLANET">
        <TR>
            <TD><xsl:value-of select="NAME"/></TD>
            <TD><xsl ivalue-of select-"MASS"/></TD>
            <TD><xsl:value-of select-"RADIUS"/></TD>
            <TD><xsl rvalue-of select=»"DAY"/></TD>
        </TR>
    </xsl:template*
</xsl:stylesheet»
```

Это все, что нам нужно; вот результат:

```
<HTML>
    <HEAD>
        <TITLE>
            The Planets Table
        </TITLE>
    </HEAD>

    <BODY>
        <H1>
            The Planets Table
        </Hl>

        <TABLE BORDER-"2">
            <TR>
                <TD>Name</TD>
                <TD>Mass</TD>
                <TD>Radius</TD>
```

```
            <TD>Day</TD>
        </TR>
        <TR>
            <TD>Mercury</TD>
            <TD>.0553</TD>
            <TD>1516</TD>
            <TD>58.65</TD>
        </TR>
        <TR>
            <TD>Venus</TD>
            <T0>.815</TD>
            <TD>3716</TD>
            <TD>116.75</TD>
        </TR>
        <TR>
            <TD>Earth</TD>
            <T0>1</TD>
            <TD>2107</TD>
           <TD>1</TD>
       </TR>
    </TABLE>
  </B0DY>
</HTML>
```

Это практически то, что требовалось. Если взглянуть на рис. 2.2, видно, что в этом HTML-файле не выведены значения атрибута UNITS, который присутствует у каждого элемента (кроме атрибута <NAME>) в planets.xml:

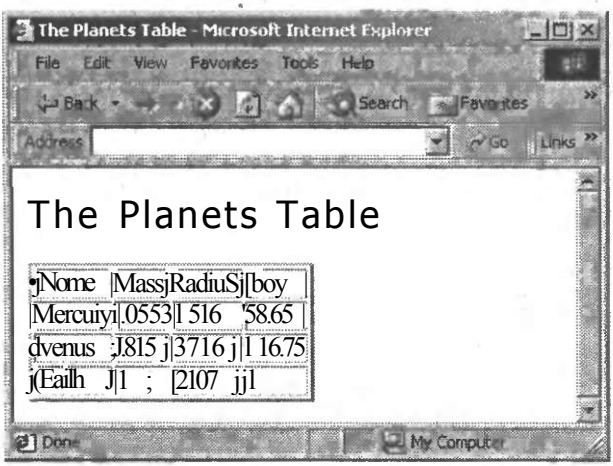


Рис. 2.2. Planets.html без атрибутов в Internet Explorer

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml" href°"planets.xsl"?>
<PLANETS>

    <PLANET>
        <NAME>Mercury</NAME>
        <MASS UNITS-"(Earth - 1)">.0553</tfASS>
        <DAY UMTS-"days">58.65</DAY>
```

```
        <RA0IUS UNITS-"miles">1516</RADIUS>
        <0ENSITY UNITS="(Earth - 1)">.983</DENSITY>
        <DISTANCE UNITS-"minion mi 1 es">43. 4</DISTANCEx | - -В перигелии-->
    </PlANET>
```

В главе 3, в которой шаблоны рассмотрены более подробно, будет показано, как извлекать значения атрибутов из элементов XML

Однако перед тем, как начать углубляться в работу с шаблонами, необходимо еще изучить достаточно много материала о таблицах стилей в общем. Например, в рабочий проект XSLT 1.1 включена поддержка рекомендации XML Base, а это значит, что она также появится и в XSLT 2.0.

Поддержка XML Base

Одним из пунктов, добавленных в рабочий проект XSLT 1.1, была поддержка спецификации W3C XML Base. На момент написания книги спецификация XML Base существует в форме предлагаемой рекомендации (Proposed Recommandation), датированной 20 декабря 2000 г.; текущую версию документа можно найти по адресу www.w3.org/TR/xmlbase/.

Эта спецификация позволяет обеспечивать для документов XML и XSL базовый идентификаторы URI аналогично элементу HTML <BASE>. (Фактически, благодаря элементу HTML <BASE> существует XBase: W3C взял на себя обязательство дать XML всю мощь ссылочной модели HTML 4.0 и затем развивать ее.) Как вы помните, одним из свойств элементов XSL является их базовый URI, и теперь его можно устанавливать при помощи XML Base. Однако пока ни один из известных мне процессоров XSLT не поддерживает XML Base.

Здесь будет приведен только обзор работы с XML Base: для установки базового URI XML-документа можно использовать атрибут xmlibase. Другие идентификаторы URI документа затем вычисляются, используя это значение в качестве базы. Заметьте, что xmltbase использует пространство Имен xml, которое предопределено в XML как "http://www.w3.org/XML/1998/namespace". В следующем примере используются ссылки XML, XML links (или XLinks):

```
. <?xml version-"1.0"?>
 <MOVIE_REVIEW xmlns-.xlink - "http //www.w3.org/1999/xlink"
     xmT:base-"http://www.starpowder.com/"
     xlinkrtype - "simple"
     xlink:show - ".new"
     xlink:href - "reviews.xml">
     Mr. Blandings Builds his Dream House
 </MOVIE_REVlEW>
```

Используя значение, присвоенное атрибуту xml-.base, URI атрибута xlink:href, "reviews.xml" раскрывается в полный URI "http://www.starpowder.com/reviews.xmr. Подобным образом можно при помощи xml:base задать базовый URI для документа или конкретного элемента* I. ;

В рабочем проекте XSLT 1.1 с каждым узлом связан его URI, называемый базовым URI, который используется для раскрытия значений атрибута, представляющих относительные URI, в абсолютные URI. Вот как следует определять базовый URI:

- базовый URI корневого узла — это URI документа;
- базовый URI узла элемента— это базовый URI, заданный в элементе атрибутом xmlrbase (если таковой существует), или базовый URI элемента-родителя элемента в документе, или внешняя сущность (если существует), или базовый URI сущности документа или внешней сущности, содержащей элемент;
- базовый URI для узла инструкции обработки — это URI, который будет применен к ссылке URI в содержимом инструкции обработки. В соответствии со спецификацией XML Base, базовым URI для ссылки URI, появляющейся в содержимом инструкции обработки, является базовый URI родительского элемента инструкции обработки (если таковой имеется) в пределах сущности документа или внешней сущности, или базовый URI сущности документа или внешней сущности, содержащей инструкцию обработки;
- базовым URI для текстового узла, узла комментария или узла атрибута является базовый URI родителя этого узла;
- базовый URI для узла пространства имен, однако, зависит от реализации.

Установка базовых URI документов и элементов может оказаться удобной, если вам приходится работать с очень большим множеством документов При реорганизации этого множества документов вам нужно будет переустановить только один базовый URI, а не все индивидуальные URI. Однако, как я уже говорил, из известных мне процессоров XSLT ни один пока не поддерживает XML Base.

Выбор методов вывода

Еще один важный аспект создания таблиц стилей — выбор метода вывода: XML, HTML, текст (то есть любой текстовый документ, не являющийся документом XML или HTML) и т. д. Другими словами, метод вывода (output method) определяет тип создаваемого документа. По умолчанию методом вывода является XML, хотя большинство процессоров при встрече элемента <HTML> создают документы HTML. (Некоторые процессоры действуют подобным образом, если расширение создаваемого файла документа .html.)

Этот материал подробно изложен в главе 6, но сейчас мы также кратко его рассмотрим. Если только вы полностью не уверены, что правила вывода вашего процессора XSLT по умолчанию делают именно то, что нужно, зачастую можно посоветовать явно установить тип вывода в требуемый вам вид документа при помощи элемента <xsl:output>. Выходной тип может задать, например, будет ли процессор XSLT записывать инструкцию обработки XML, <?xmlversion«"1.0"?>,

в начало документа, а также указать тип MIME (такой, как «text/xml» или «text/html») документов, отправляемых процессором XSLT с web-сервера браузеру. Кроме того, если вы установите тип вывода в HTML, большинство процессоров XSLT распознают, что не всем элементам в HTML требуются закрывающие и открывающие теги, и т. д.

Преобразованию из XML в другие типы документов посвящена глава 6, но здесь мы также кратко рассмотрим элемент <xsl:output>, поскольку это важно для общего понимания работы таблиц стилей. В следующем списке перечислены атрибуты <xsl:output>:

- cdata-section-elements (необязательный). Задает названия тех элементов, чье содержимое вы хотите вывести в виде разделов CDATA. Принимает значение списка QName, разделенных символами-разделителями;
- doctype-public (необязательный). Определяет открытый идентификатор, который будет использоваться в объявлении <!DOCTYPE> вывода. Устанавливается в строковое значение;
- doctype-system (необязательный). Определяет системный идентификатор, который будет использоваться в объявлении <!DOCTYPE> вывода;
- encoding (необязательный). Определяет кодировку символов. Устанавливается в строковое значение;
- indent (необязательный). Определяет, будет ли вывод Выровнен с отображением структуры вложенности. Устанавливается в «yes» или «no»;
- media-type (необязательный). Определяет тип MIME вывода. Устанавливается^ в строковое значение;
- method (необязательный). Определяет формат вывода. Устанавливается в «xml», «html», «text» или допустимое имя типа QName;
- omit-xml-declaration (необязательный). Определяет, будет ли включено в вывод объявление XML. Устанавливается в «yes» или «no»;
- standalone (необязательный). Определяет, будет ли включено в вывод отдельное объявление, и если да — устанавливает его значение. Устанавливается в «yes» или «no»;
- version (необязательный). Задает версию вывода. Устанавливается в допустимую лексему типа NMToken.

Чаще всего используется атрибут method элемента, потому что с его помощью устанавливается требуемый тип дерева вывода. Самые распространенные значения — «html», «xml» и «text».

Метод вывода: HTML

В нашей таблице стилей planets.xsl элемент <xsl:output> не используется; это значит, что для этой таблицы я полагался на правила вывода по умолчанию. Типом вывода по умолчанию является XML, если только процессор XSLT не встретит тег <HTML> или <html>. (Заметьте, что это не формальное требование, а только со-

глашение, которому не обязаны следовать все процессоры XSLT.) В planets.xsl я использовал тег <HTML> следующим образом:

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

    <xsl:template match-"/PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
```

Однако, если удалить этот тег:

```
<?xml version-"1,0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

    <xsltemplate match-"/PLANETS">

            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
```

то такой вид вывода вы получите от процессора XT Джеймса Кларка. Обратите внимание на инструкции обработки XML в начале:

```
<?xml version-"1.0" encoding»"utf-8"?>
    <HEAD>
        <TITLE>
            The Planets Table
         </TITLE>
    </HEAO>
```

С другой стороны, можно явно указать тип вывода HTML при помощи элемента <xsl:output>, даже не прибегая к элементу <HTML>:

```
<?xml version«"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http  //www.w3.org/1999/XSL/Transform">
    <xsl:output method-"html"/>

    <xsl:tempiate match-'7PLANETS">
            <HEAD>
```

```
        <TITLE>
            The Planets Table
        </TITLE>
    </HEAD>
```

Вот вывод XT в данном случае — только фрагмент HTML, никаких инструкций обработки XML:

```
<HEAD>
    <TITLE>
        The Planets Table
     </TITLE>
</HEAD>
```

ЭЛЕМЕНТЫ <META>, АВТОМАТИЧЕСКИ ДОБАВЛЯЕМЫЕ В HTML

Если вы явно используете элемент <xml:output method="html7>, некоторые процессоры XSLT, такие как Saxon, добавляют в элемент <head> выходного документа элемент <meta> подобным образом: cmeta http-equiv= "Content-Type" content="text/html; charset=utf-8">.

В общем случае процессоры XSLT должны учитывать, что в HTML определенные элементы, такие как **
, <img>, <frame>** и т. д., пусты. Также пробелы и другие символы в значениях атрибута URI преобразуются в соответствии со спецификацией HTML (пробел становится «%20» и т. п.), инструкции обработки завершаются символом >, а не ?>, и учитывается тот факт, что отдельным атрибутам значение не присваивается.

Метод вывода: XML

В этом разделе я собираюсь рассмотреть пример, который подробно изучается в главе 6. Я хочу зайти немного вперед и использовать элемент <xsl:copy>, с которым мы познакомимся в главе 3, для создания таблицы стилей, которая создает копию любого XML-документа.

Я воспользуюсь образцом совпадения «*», которому, как говорилось ранее, удовлетворяет любой элемент, и применю элемент <xsl:copy> для копирования текущего элемента в выходной документ. Вот как выглядит новая таблица стилей, копирующая исходный документ в результирующий:

```
<?xml version="1.0"?!>
<xsl:stylesheet version-"1.0"
xmlns:xsl«"http://www.w3.org/1999/XSL/Transform">
    <xsl:tempiate match-"*">
        <xsl:copy>
            <xsl:apply-templates/>
        </xsl:copy>
    </xsl:template*
</xsl:stylesheet»
```

Поскольку эта таблица стилей предназначена для копирования в новый документ XML любого документа XML даже документов XHTML, представляющих собой XML-документы, использующие тег <html>, — я явно указываю, что здесь методом вывода является XML. Если бы я этого не сделал, скопированные документы XHTML не начинались бы с объявления XML:

```
<?xml version-"1 0"?>
<xsl:stylesheet version«"1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method-"xml"/>
    <xsl:tempiate match-"*">
        <xsl:copy>
            <xsl:apply-templates/>
        </xsl:copy>
    </xsl:tempiate>
</xsl:stylesheet>
```

В этом примере в результирующий документ копируются только элементы, но не текстовые узлы, комментарии или атрибуты. Более законченную версию этой же таблицы стилей мы рассмотрим в главе 4.

Помните, что XML — это метод вывода по умолчанию, если только ваш входной документ не содержит тег <HTML> или <html>. Тем не менее, даже если вы выполняете преобразование одного XML-документа в другой, зачастую удобно использовать элемент <xsl:output> — для того, чтобы, например, задать кодировку символов (по умолчанию это обычно UTF-8, восьмибитовое подмножество Unicode) или выравнивание выходного документа (это описывается в главе 3).

РАБОТА С ФРАГМЕНТАМИ XML

Можно работать не только с документами XML целиком, но и с их фрагментами. В этом случае следует установить атрибут omit-xml-declaration в «yes»—для того, чтобы опустить объявление XML в начале дерева вывода, как обсуждается в главе 6.

Когда вы используете метод вывода XML, дерево вывода представляет собой хорошо сформированный XML (однако он не обязан быть допустимым). Не обязательно оно должно быть хорошо сформированным XML-документом, это может быть общая разобранная внешняя сущность XML. Содержимое вывода может включать символьные данные, разделы CDATA, ссылки на сущности, инструкции обработки, комментарии и элементы. Вывод дожен также удовлетворять объявлению пространств имен XML.

Метод вывода: текст

Метод текстового вывода предназначен не только для создания простого текста: он применяется для любого основанного на тексте формата, не являющегося XML или HTML. Например, с его помощью можно создавать документы в формате RTF (Rich Text Format). В этом формате для задания форматирования документов используются встроенные текстовые коды — их можно поместить в документы самостоятельно при помощи метода текстового вывода.

В листинге 2.6 приведен пример таблицы стилей (которую мы увидим в главе 6), преобразующей planets.xml в planets.rtf.

Листинг 2.6. Таблица стилей RTF

```
<?xml version="1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www w3.org/1999/XSL/Transform">
    <xsl:output method-"text"/>
    <xsl:strip-space elements-"*"/»
<xsl: tempi ate match-"/PLANETS"»{\rtfl\ansi\deffO{\fonttbl {WfcharsetO
Courier New:}}
\viewkind4\ucl\pard\langl033\b\ The Planets Table\par
\b0
NameVtab MassXtab Rad.\tab Day\par
<xsl:apply-templates/»
\par
}</xsl:template>
<xsl:tempiate match-"PLANET"»
<xsl:value-of select-"NAME"/>
\tab
<xsl:value-of select-"MASS"/>
\tab
<xsl:value-of select-"RADIUS"/>
\tab
<xsl:value-of select-"DAY"/>
\tab
\par
</xsl:template»
</xsl:stylesheet»
```

Результирующий документ RTF, planets.rtf, изображен на рис. 2.3 в Microsoft Word 2000. Обратите внимание на то, что я установил метод вывода в текст в элементе <xsl:output method«"texty>:

```
<?xml version-" 1.0"?»
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">
    <xsl:output method-"text"/>
    <xsl:tempiate
match-'7PLANETS"»{\rtfl\ansi\deffO{\fonttbl(WfcharsetO Courier New:}}
\viewkind4\ucl\pard\langl033\b\ The Planets Table\par
```

Вы также могли заметить, что я принялся записывать коды RTF сразу же после элемента <xsl:template> — я сделал так потому, что в документах RTF коды RTF должны появляться с самого начала. Если бы я решил вставлять коды RTF на следующей строке — например, так:

```
<?xml version-'! 0">»                                    „ л х ^ т Щ щ и л
<xsl stylesheet versioft-" 1.0"                  .чям с-м  -н.элзил-эдн*
xmlns xsVhttp://www.w3 org/1999/XSL/Transform">          rt>
    <xsl:output method»"text"/>
    <xsl:tempiate match-"/PLANETS"»
{\rtfl\anst\deffO{\fonttbl{WfcharsetO Courier New;})
Xviewkind4\ucl\pard\langl033\b\ The Planets TableXpar л\\ к-щ
```

то выходной файл RTF начинался бы с символа новой строки, что вызвало бы ошибку приложения для работы с файлами RTF — такого, как Microsoft Word. Более подробно RTF и другие форматы будут рассмотрены в главе 6.

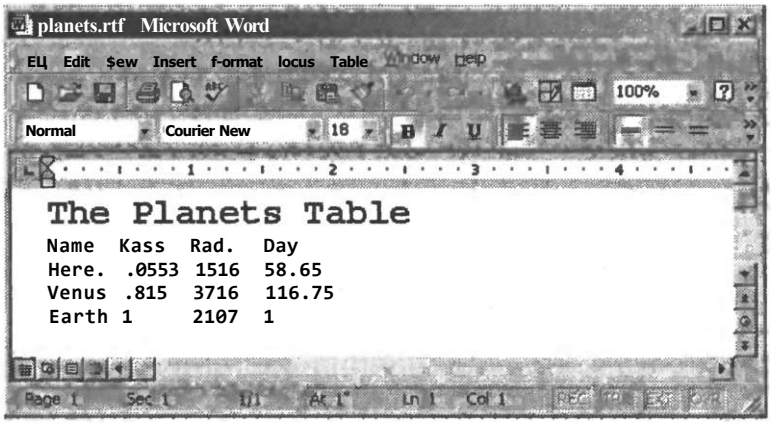


Рис. 2.3. Planets.rtf в Microsoft Word

Упрощенные таблицы стилей

Как можно заметить на основе рассмотренного до сих пор материала, создавать таблицы стилей XSLT не так-то просто. W3C попытался облегчить эту процедуру, разработав упрощенные таблицы стилей (simplified stylesheets), в которые не нужно — а на самом деле и невозможно — включать элемент <xsl:stylesheet> или какие-либо еще элементы высокого уровня.

Фактически упрощенная таблица стилей представляет собой результирующий документ, содержащий несколько элементов XSL невысокого уровня. W3C называет такую таблицу «элемент буквального результата как таблица».

В листинге 2.7 я осуществлю преобразование planets.xml в planets.html, но теперь я сделаю это при помощи упрощенной таблицы стилей. В упрощенных таблицах стилей вы не можете применять элементы высокого уровня, такие как <xsl:template>, позволяющие рекурсивную обработку всех элементов в исходном документе. Поэтому здесь я зайду немного вперед и воспользуюсь элементом <xsl:for-each> (рассматриваемым в главе 5), который, не являясь элементом высокого уровня, позволяет обработать в цикле сразу несколько узлов.

Мне также нужно каким-либо образом выбрать все элементы <PLANET> в исходном документе, и вы можете подумать, что это невозможно без использования шаблонов нескольких уровней — например, один для корневого узла, затем один для выбора на следующем уровне вниз, корневом элементе <PLANETS>, и затем еще один уровень вниз для самих элементов <PIANET>. Однако, используя XPath,

1можно при помощи выражения "//PLANET" найти все узлы элемента <PLANET>, производные от корневого узла (см. главу 4). Это значит, что я могу написать упрощенную таблицу стилей следующим образом.

Листинг 2.7. Упрощенная таблица стилей

```
<HTMl xmlns:xsl-"http://www.w3.org/1999/XSL/Transform" xsl:vers1on="l.0">
    <HEAD>
        <TITLE>
           The Planets Table
        </TITLE>
    </HEAD>
    <B0DY>
        <H1>
           The Planets Table
        </Hl>
        <TABLE BORDER-"2">
           <TR>
                <TD>Name</TD>
                <TD>Mass</TD>
                <TD>Rad1us</TD>
                <TD>Day</TD>
           </TR>
           <xsl:for-each  select-"//PLANET">
               <TR>
                   <TD><xsl:value-of  select-"NAME"/x/TD>
                   <TDxxsl :value-of  select-"MASS"/x/TD>
                   <TDxxsl:value-of  select-"RADIUS"/></TD>
                   <TD><xslrvalue-of  select-"DAY"/></TD>
               </TR>
            </xsl:for-each>
        </TABLE>
    </BODY>
</HTML>
```

Данная версия работает точно так же, как и предыдущая версия planets.xsl, при этом совсем не используя элементы высокого уровня. Упрощенные таблицы стилей, такие как эта, были представлены в помощь авторам HTML при осуществлении преобразований в XSL, однако их применимость весьма ограничена. Очевидно, все равно нужно знать, как работать с элементами XSL, а отсутствие возможности использовать <xsl:template> в данном случае только затруднило работу. Однако вы должны знать, что упрощенные таблицы стилей существуют и включены в спецификацию XSLT.

ОБРАБОТКА ПО УМОЛЧАНИЮ БЕЗ ЭЛЕМЕНТА <XSL:STYLESHEET>

Если процессор XSLT не может найти в таблице стилей элемент <xsl:stylesheet>, он трактует таблицу как упрощенную таблицу стилей.

Встроенные таблицы стилей

Рекомендация XSLT также поддерживает встроенные таблицы стилей, embedded stylesheets (вслед за использованием встроенных таблиц стилей и элементов стиля в HTML), но, как и упрощенные таблицы стилей, они не очень распространены.

Встроенные таблицы стилей могут обрабатывать не все процессоры XSLT, но некоторые — в частности, Saxon — могут. Давайте рассмотрим пример. В этом случае я включил весь элемент таблицы стилей целиком из planets.xsl в planets.xml для создания нового документа, embedded.xml. В новом документе будут содержаться все данные и вся таблица стилей. Заметьте, что для того, чтобы быть хорошо сформированным XML, embedded.xml должен иметь только один корневой элемент, поэтому я сделал таблицу стилей (то есть элемент <xsl:stylesheet>) родительским элементом корневого элемента <PLANETS>.

Чтобы указать, какой элемент будет трактоваться как встроенная таблица стилей, я задал элементу <xsl:stylesheet> ID "stylesheet", установив атрибут id в это имя:

```
<xsl:stylesheet version-"1.0" id-"stylesheet"
mlns:xs 1 -"http://www.w3.org/1999/XSL/Transform">
```

В начале документа я также присваиваю это имя, "stylesheet", атрибуту href элемента <?xml-stylesheet?>:

```
<?xml-stylesheet type-"text/xml" href="#stylesheet"?>
```

Теперь процессору XSLT известно, какой элемент я хочу использовать в качестве таблицы стилей — элемент с ID "stylesheet". Но что это за элемент? Элементы XML формируют элементы типа ID в объявлениях DTD или схемах XML, и, как вы помните, информация объявлений DTD и схем пока еще не передается процессору XSLT.

Некоторые процессоры XSLT, такие как Saxon, читают объявление DTD, если оно есть, для определения атрибутов, обладающих типом ID, поэтому я включил DTD в embedded.xml (листинг 2.8).

Листинг 2.8. planets.xml со встроенной таблицей стилей

```
<?xml version-"1.0"?>
<?xml-stylesheet type-"text/xml" href-"#stylesheet"?>
<!DOCTYPE PLANETS [
<!ELEMENT PLANET (CUSTOMER)*»
<1ELEMENT CUSTOMER (NAME.MASS.RADIUS.DAY)>
<!ELEMENT NAME (#PCDATA)>
<1ELEMENT MASS (#PCDATA)>
<!ELEMENT RADIUS (#PCDATA)>
<!ELEMENT DAY (#PCDATA)>
<!ELEMENT xsl:stylesheet (xsl:template)*»
<1ELEMENT xsl:tempiate (#PCDATA)>
<!ATTLIST xsl:stylesheet
    1d ID ^REQUIRED
    version CDATA #IMPLIED>
]>
<PLANETS>
    <PLANET>
        <NAME>Mercury</NAME>
        <MASS UNITS-"(Earth - 1)">.0553</MASS>
        <DAY UNITS-"days">58.65</DAY>
        <RADIUS UNITS-"miles">1516</RADIUS>
        «DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
        «DISTANCE UNITS-"million miles">43.4</0ISTANCE><!--В перигелии--»
    </PLANET>
```

продолжение &

Листинг 2.8 (продолжение)

```
    <PLANET>
        <NAME»Venus</NAME>
        <MASS UNITS-"(Earth - 1)">.815</MASS>
        <OAY UNITS»"days">116.75</DAY>
        «RADIUS UNITS-"miles">3716</RADIUS>
        <DENSITY UNITS-"(Earth - 1)"».943</DENSITY»
        «DISTANCE UNITS-"million miles"»66.8</DISTANCE><!--B перигелии--»
    </PtANET>

    «PLANET»
        <NAME>Earth</NAME>
        <MASS UNITS-"(Earth - 1)">1</MASS>
        <DAY UNlTS«"days">l</DAY>
        <RADIUS UNITS-"miles"»2107«/RADIUS»
        «DENSITY UNITS-"(Earth - 1)">1</DENSITY>
        «DISTANCE UNITS""million miles">128 4</DISTANCEx!--B перигелии-»
    </PLANET>
<xsl-.stylesheet version»" 1.0" id-"stylesheet"
xmlns:xsl="http //www w3 org/1999/XSL/Transform">

    <xsl:tempi ate match-"/PLANETS"»
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <B0DY»
                <H1>
                    The Planets Table
                </H1»
                <TABLE BORDER""2"»
                    <TR>
                        <TD»Name</TD»
                        <TD»Mass</TD>
                        <TD»Radius</TD>
                        <TD>Day</TD>
                    </TR>
                    <xsl:apply-templates/»
                </TABLE>
            </B0DY>
        </HTML>
    </xsl template»

    <xsl:tempiate match-"PLANET"»
       <TR>
          <TD»<xsl :value-of select-"NAME"/x/TD>
          <TDxxsl :value-of select-"MASS"/»</TD>
          <TDxxsl:value-of select-"RADIUS"/»</TD>
          <TDxxsl:value-of select-"DAY"/x/TD>
       </TR>
   </xsl:tempi ate»

    <xsl template match-"xsl-. styl esheet"x/xsl: tempi ate»

</xsl stylesheet»

</PLANE7S>
```

Следует отметить еще одно: включив всю таблицу стилей в файле embedded.xml. в элемент <xsl:stylesheet>, я должен был предоставить шаблон таблицы стилей для элемента <xsl:styLesheet>. (Если бы я этого не сделал, текст из текстовых узлов таблицы стилей был бы скопирован в результирующий документ, — это обсуждается в главе 3 в разделе, посвященном правилам по умолчанию для шаблонов.) Я оставил этот элемент пустым, поместив в конце таблицы стилей в embedded.xml следующую строку, поэтому из самой таблицы стилей в результирующий документ ничего не копируется:

```
<xsl:template match»"xsl:stylesheet"></xsl:template»
```

Теперь в Saxon я могу из embedded.xml создать planets.html. В Windows для указания того, что используется встроенная таблица стилей, в Saxon служит параметр -a:

```
C:\planets>saxon -a embedded.xml > planets.html
```

Элемент <xsl:include>

Другой способ вставить таблицы стилей внутрь других документов — использовать элемент <xsl:include>, позволяющий включить содержимое файла в определенное место в таблице стилей. У этого элемента только один атрибут:

- href (обязательный). URI таблицы стилей, которую вы хотите включить.

Этот элемент пустой и не обладает никаким содержимым.

Рассмотрим листинг 2.9. В этом случае я помещу часть таблицы стилей из planets, xsl в новый документ, rules.xml Затем я смогу включить rules.xml в planets.xsl.

Листинг 2.9. Включение таблицы стилей

```
<?xml version-"1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

    <xsl:i nclude href-"rules.xsl"/>

    <xsltemplate match-"/PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <BODY>
                <H1>
                    The Planets Table
                </Hl>
                <TABLE B0RDER-"2">
                    <TD>Name</TD>
                    <TD>Mass</TD>
                    <TD>Radius</TD>
                    <TD>Day</TD>
```

продолжение ^

Листинг 2.9 (продолжение)

```
                    <xsl:apply-templates/»
                </TABLE>
            </B0DY>
        </HTML>
    </xsl:tempiate>

</xsl:stylesheet>
```

А вот как выглядит rules.xsl (листинг 2.10). Обратите внимание на то, что это полный документ XSL с объявлением XML и элементом <xsl:stylesheet>.

Листинг 2.10. rules.xsl

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

    <xsl template match-"PLANET">
       <TR>
          <T0xxsl:value-of select°"NAME"/></TD>
          <TDxxsl:value-of select-"MASS"/></TD>
          <TDxxsl :value-of select-"RADIUS"/></TD>
          <TDxxsl:value-of select-"DAY"/x/TD>
       </TR>
   </xsl:template»

</xsl:stylesheet>
```

Вот как это работает. Кроме <xsl:include>, для вставки таблиц стилей или фрагментов таблицы стилей можно также применять <xsl:import>.

НОВШЕСТВА В XSLT 2.0

Один из аспектов, которые явно войдут в XSLT 2.0, следующий — включаемые документы могут использовать свои собственные таблицы стилей. Например, если вы включите документ, написанный на XML языке MathML, этот включенный документ должен иметь возможность использовать свою собственную таблицу стилей.

Элемент <xsl:import>

Так же как и <xsl:include>, <xsl:import> позволяет вставить таблицу стилей или фрагмент таблицы стилей в другую таблицу стилей. И так же, как <xsl:include>, у <xsl:import> есть только один атрибут:

- href (обязательный). URI включаемой таблицы стилей.

И так же, как <xsl:include>, <xsl:import> пуст и не имеет содержимого. В чем же тогда разница между ними? Разница заключается в *старшинстве импорта* (import precedence), w.

Порядок включения дает процессору XSLT способ разрешения; конфликтов, которые могут возникнуть, например, когда двум правилам соответствует один и тот же узел. Старшинство импортируемой таблицы стилей или фрагмента таблицы меньше, чем старшинство таблицы стилей, которая ее импортирует. Если вы импортируете несколько таблиц стилей или фрагментов таблицы стилей,

первая из них будет обладать меньшим старшинством, чем импортируемая вслед за ней, которая в свою очередь будет обладать меньшим старшинством, чем следующая, и т. д.

В остальном, однако, импорт таблицы стилей или фрагмента таблицы практически аналогичен их включению, хотя вы и используете <xsl:import> вместо <xsl:include>:

Листинг 2.11. Импорт таблицы стилей

```
<?xml version-"1 0"?>
<xsl:stylesheet version-"l O"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

    <xsl:Import href-"rules.xsl"/>

    <xsltemplate match-"/PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <B0DY>
                <H1>
                    The Planets Table
                </H1>
                <TABLE B0RDER-"2">
                    <T0>Name</TD>
                    <TD>Mass</TD>
                    <TD>Radius</TD>
                    <TD>Day</TD>
                    <xsl:apply-templates/>
                </TABLE>
            </B0DY>
        </HTML>
    </xsl:tempiate>

</xsl:stylesheet>
```

Элемент <xsl:apply-imports>

Если вы импортируете таблицу стилей с шаблоном, например, для элемента <PLANET>, и затем определите свой собственный элемент <PLANET>, импортируемая версия будет перекрыта. Как можно получить доступ к перекрытой версии? Этому служит элемент <xsl:apply-imports>.

В XSLT 1.0 у этого элемента нет атрибутов и нет содержимого. В рабочем проекте XSLT 1.1 элемент <xsl:apply-imports> может обрабатывать параметры, поэтому он может содержать ноль или более элементов <xsl:with-param> (подробнее о параметрах см. главу 9).

В качестве примера я модифицирую только что рассмотренный нами пример <xsl:import>. В этом случае я Добавлю еще один столбец в генерируемую в этом

примере (листинг 2.12) HTML-таблицу, названную DATA, путем перекрытия шаблона <PLANET> в rules.xsl новым шаблоном <PLANET> из planets.xsl. Новый шаблон просто добавляет в таблицу новый столбец и затем применяет старый шаблон <PLANET> к оставшимся данным. Доступ к старому шаблону осуществляется при помощи <xsl:apply-im ports».

Листинг 2.12. Использование <xsi:apply-imports>

```
<?xml version="1.(r?>
<xsl:stylesheet version-"1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <xsl:1mport href-"rules.xSl"/>

    <xsl:template match-"/PLANETS"»
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <B0DY>
                <H1>
                    The Planets Table
                </Hl>
                <TABLE BORDER-"2">
                    <TR>
                        <TD>Date</TD>
                        <TD>Name</TD>
                        <TD>Mass</TD>
                        <TD>Radius</TD>
                        <TD>Day</TD>
                    <xsl apply-templates/>
                    </TR>
                </TABLE>
            <BODY>
        </HTML>
    </xsl:tempiate>

    <xsl:tempiate match-KPLANET">
        <TR>
            <TD>4/l/2002</TD>
            <xsl:apply-Imports/»
        </TR>
    </xsl:template»

</xsl:stylesheet»
```

А вот как выглядит новая версия rules.xsl (листинг 2.13). *

Листинг 2.13. Новая версия rules.xsl

```
<?xml versi0n-"1.0"?»
<xsl:stylesheet version-"1.0"
xmlns xsl-"http://www.w3.org/1999/XSL/Transform">

    <xs!: tempi ate matcfi-"PLANET''>
          <TD»<xsl :value-0f select-"NAME''/»</TD>
```

```
        <TDxxs I:value-of select- "MASS"/></TD>
        <TDxxsl:value-of select-"RADIUS"/></TD>
        •itDxXsl: value-of select="DAY"/x;/TD>
    </xsl:template»

</xsl:stylesheet»
```

Результаты можно видеть на рис. 2.4. Я создал один шаблон при помощи другого, что является ближайшим аналогом в XSLT для наследования из объектно-ориентированной парадигмы.

В рабочем проекте XSLT 1.1 с элементом <xsl:apply-imports> можно использовать параметры таблицы стилей, что означает, что вы можете использовать элементы <xsl:with-param> в качестве содержимого <xsl:apply-imports>. Подробное обсуждение параметров и элемента <xsl:with-param> приводится в главе 9.

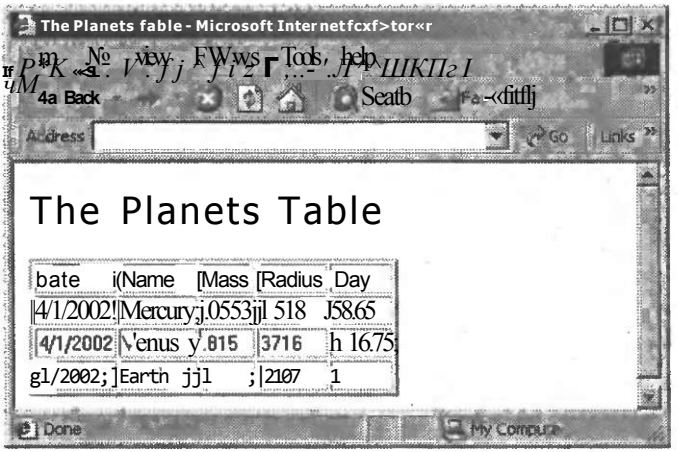


Рис. 2.4. Использование <xsl:apply-imports>

Преобразование документов XML при помощи Internet Explorer

В нашем обзоре таблиц стилей есть еще одна тема для обсуждения: как использовать таблицы стилей в Internet Explorer. Как мы видели в главе 1, для считывания документов XML и XSL можно использовать JavaScript, и осуществлять преобразование при помощи разборщика MSXML3. (Дополнительная информация по этой теме приведена в главе 10. Документацию по Internet Explorer можно также прочитать по адресу http://msdn.microsoft.com/xml/XSLGuide/.)

Однако, если вам нужно открыть XML-документ непосредственно в Internet Explorer при навигации (например, набрав URI в адресной строке), в использовании элементов <?xml-stytesheet?> и <xsl:stylesheet> вы полагаетесь на браузер, а это означает, что для IE версии 5.5 и более ранних необходимо провести ряд изменений.

INTERNET EXPLORER 6.0 И ПОЛУЧЕНИЕ И УСТАНОВКА РАЗБОРЩИКА MSXML

Обратите внимание: IE 6.0 только что вышел, когда книга готовилась к печати, и он поддерживает полный синтаксис XSLT (за исключением того, что вы все еще должны использовать тип «text/xsl* для таблиц стилей, как в <?xml-stylesheet type="text/xsl" href="planets.xsl"?>, вместо «text/xml»). Если вы используете IE 5.5 или более ранних версий, вы такюе можете загрузить и установить последнюю версию разборщика MSXML прямо от Microsoft вместо предыдущей версии, используемой Internet Explorer. Если это сделать, вам не нужно будет проводить изменения, перечисленные в этом разделе. Дополнительную информацию можно получить по адресу http://msdn.microsoft.com/ xml/general/xmlparser.asp. Загрузить разборщик сейчас можно по адресу http://msdn.microsoft.com/downloads/default.asp?URL=/code/sample.asp?url=/msdn-files/027/000/541/msdncompositedoc.xml. (Учтите, однако, что Microsoft, кажется, реорганизует свой web-узел примерно каждые пятнадцать минут.) Если вы работаете с IE 5.5 или более ранней версии, я настоятельно рекомендую вам загрузить MSXML, чтобы вам не пришлось изменять все ваши таблицы стилей XSLT для их использования в IE, или обновить браузер до версии 6.0 или более поздней.

Для IE версии 5.5 или более ранней необходимо внести изменения и в planets.xml, и в planets.xsl. Для того чтобы использовать planets.xml в IE, необходимо преобразовать атрибут type в инструкции обработки <?xml-stylesheet?> из «text/xml» в «text/xsl» (листинг 2.14).

Листинг 2.14. Версия planets.xml для Internet Explorer

```
<?xml version-"!..D"?>
<?xml-stylesheet type-"text/xsl" href="pianets.xsl"?>
<PLANETS>

    <PLANET>
        <NAME>Mercury</NAME>
        <MASS UNITS-"(Earth - 1)">.0553</MASS>
        <DAY UNITS-"days">58.65</DAY>
        <RADIUS UNITS-"miles">1516</RADIUS>
        <DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
        <DISTANCE UNITS-'million miles">43.4</DISTANCE><!--В перигелии-->
    </PLANET>

    <PLANET>
        <NAME>Venus</NAME>
        <MASS UNITS»"(Earth - 1)">.815</MASS>
        <DAY UNITS-"days">116.75</DAY>
        <RADIUS UNITS-"mi 1es">3716</RADIUS>
        OENSITY UNITS-"(Earth - 1)">.943</DENSITY>
        «DISTANCE UNITS="mill ion miles">66.8</DISTANCE><!--В перигелии->
    </PLANET>

    <PLANET>
        <NAME>Earth</NAME>
        <MASS UNITS-"(Earth - 1)">1</MASS>
        <DAY UNITS-"days">1</DAY>
        <RADIUS UNITS-"miles">2107</RADIUS>  ,r:
        <DENSITY UNITS-"(Earth - 1)">1</DENSITY>
        <DISTANCE UNITS-"million miles">128 4</DISTANCEx'--В перигелии-->
    </PLANET>

</PLANETS>
```

Для работы в IE версии 5.5 или младше необходимо также преобразовать таблицу стилей planets.xsl. Главное отличие между рекомендацией W3C XSL и реализацией XSL в IE состоит в том, что в IE версии 5.5 и младше не реализованы никакие правила XSL по умолчанию — см. главу 3 (заметьте, что IE версии 6.0 на момент выхода этой книги в свет не обременена такой проблемой). Это значит, что для IE версии 5.5 или младше я должен включать правило XSL для корневого узла документа, который задается при помощи «/*. Я также должен использовать в таблице стилей другое пространство имен XSL, «http://www.w3.org/TR/WD-xsb, и опустить атрибут версии в элементе <xsl:style$heet> (листинг 2.14).

Листинг 2.15. Версия planets.xsl для Internet Explorer

```
<?xml version-"]..0"?>
<xsl:stylesheet xmlns:xsl-"http://www w3.org/TR/WD-xsl">

    <xsltemplate match-"/">
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <B0DY>
                <H1>
                    The Planets Table
                </Hl>
                <TABLE B0RDER-"2">
                    <TR>
                        <TD>Name</TD>
                        <TD>Mass</TD>
                        <TD>Radius</TD>
                        <TD>Day</TD>
                    </TR>
                    <xsl:apply-templates/>
                </TABLE>
            </BODY>
        </HTML>
    </xsl:tempiate>

    <xsl:tempi ate match-"PLANETS">
        <xsl:apply-templates/>
    </xsl:tempiate>

    <xsl:template match-"PLANET">
       <TR>
          <TD><xsl:value-of select-"NAME"/></TD>
          <TD><xsl value-of select-"MASS''/></TD>
          <TD><xsl:value-of select-"RADIUS"/></TD>
          <TDxxsl value-of select-"DAY"/></TD>
       </TR>
   </xsl:tempiate>

</xsl:stylesheet>
```

Вот и все! Теперь мы успешно изменили planets.xml и planets.xsl для прямого просмотра в Internet Explorer. Такие изменения необходимо провести для навигации непосредственно по XML-документам с таблицами XSL в этом браузере.

На этом завершается обзор работы с таблицами стилей в XSL. В следующей главе мы более подробно познакомимся с основной частью таблиц стилей — шаблонами.

Создание и применение шаблонов



Эта глава полностью посвящена созданию шаблонов и работе с ними как с основой таблиц стилей XSLT. Каждый шаблон образует правило, которое процессор XSLT пытается применить к исходному документу.

В главе 2 мы кратко рассмотрели таблицы стилей и основные шаблоны, при помощи которых наши примеры таблиц стилей в действительности могли что-то сделать. В этой главе мы собираемся подробно изучить шаблоны, а в следующей мы увидим, какие виды выражений можно применять для создания образцов выбора в шаблонах, позволяющих находить требуемые узлы. Образцы выбора (match pattern) являются подмножеством всего языка XPath и достаточно сложны для того, чтобы посвятить им отдельную главу.

В этой главе мы сначала рассмотрим, как работают основные шаблоны, и затем перейдем к таким темам, как правила шаблонов по умолчанию, выбор шаблона для работы, обработка атрибутов, создание мелких и глубоких копий элементов, завершение обработки шаблона и многое другое.

Создание шаблона

В главе 2 для выбора узлов в planets.xml и преобразования этого документа в HTML я создал основной шаблон. Шаблоны в таблицах стилей создаются при помощи элементов <xsl:template>, задающих правила для требуемых преобразований. Мы создали шаблон, находивший корневой элемент <PLANETS> по образцу "/PLANETS", которому соответствуют все элементы <PLANETS>, дочерние для корневого узла:

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl»"http://www.w3.org/1999/XSL/Transform">

    <xsl.'template match-"/PLANETS">
```

```
    </xsl:tempiate>

</xsl:stylesheet>
```

Когда процессор XSLT находит узел, удовлетворяющий образцу вашего шаблона, этот узел становится контекстным узлом шаблона, то есть все операции производятся над этим узлом. На текущий узел можно ссылаться при помощи выражения XPath «.». Другие выражения XPath мы рассмотрим в этой главе и в главе 7.

УСОВЕРШЕНСТВОВАНИЯ В XSLT 2.0

В XSLT 1.0 возникали трудности при выборе элементов или атрибутов с нулевыми значениями, что должно быть исправлено в XSLT 2.0.

Внутри шаблона создайте разметку HTML, начинающую требуемую таблицу, — такая прямая вставка разметки называется элементом буквального (literal) результата. Когда процессор встречает литерал, он копирует его в результирующее дерево:

```
<?xml version-"1.0"?>
<xsl;stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

    <xsl:tempi ate match-"/PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <BODY>
                <H1>
                    The Planets Table
                </H1>
                <TABLE BORDER-"2">
                    <TR>
                        <TD>Name</TD>
                        <TD>Mass</TD>
                        <TD>Radius</TD>
                        <TD>Day</TD>
                    </TR>
                </TABLE>
            </BODY>
        </HTML>
    </xsl:tempiate>

</xsl:stylesheet>
```

Однако это правило обрабатывает только элемент <PLANETS>, но не его дочерние элементы <PLANET>, содержащие сами данные. В соответствии с правилами шаблона по умолчанию, элементы <PLANET> будут обрабатываться, если установить

шаблон для их выбора. Это, кстати, нежелательно, поскольку я хочу вставить результат обработки элементов <PLANET> в определенное место в создаваемом файле HTML. Для этого мне потребуется применить элемент <xsl:apply-templates>.

Обработка дочерних узлов

Элемент <xsl:apply-templates> дает указание процессору XSLT обрабатывать все совпадающие шаблоны для дочерних узлов контекстного узла. Элемент <xsl:apply-templates> дает возможность явно указать, когда следует закончить обработку дочерних узлов, а это имеет решающее значение при вставке их данных в нужное место в HTML-таблице.

Один важный момент часто вызывает затруднения: элемент <xsl:apply-templates> по умолчанию применяет шаблоны только к дочерним узлам контекстного или выбранного узла или набора узлов. Это выглядит довольно безобидно, но многие люди забывают о том, что атрибуты *не* считаются дочерними узлами элементов и также ими не являются объявления пространств имен. Это значит, что для обработки как элементов, так и атрибутов следует осуществить еще один или два дополнительных шага, что мы увидим далее в этой главе.

В следующем примере мы поместим элемент <xsl:apply-templates> туда, куда нам требуется вставить данные из элементов <PLANET> в таблицу HTML. Я также добавил новый шаблон для обработки элементов <PLANET>:

```
<?xml vers1on-"l.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

    <xsltemplate match-"/PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <BODY>
                <H1>
                    The Planets Table
                </H1>
                <TABLE BORDER-"2">
                    <TR>
                        <TD>Name</TD>
                        <TD>Mass</TD>
                        <TD>Radius</TD>
                        <TD>Day</TD>
                    </TR>
                    <xsl:apply-tempiates/>
                </TABLE>
            </BODY>
        </HTML>
    </xsl:tempiate>

    <xsl:tempi ate match-"PLANET">
```

```
    «/xsl:template»
</xsl:stylesheet>
```

В новом шаблоне, обрабатывающем элементы <PLANET>, я должен извлечь данные из каждого элемента <PLANET>, то есть восстановить значения в дочерних элементах элемента <PLANET>, таких как <MASS>, <DAY> и т. д.:

```
<?xml version-"1.0"?>
<?xml-stylesheet type="text/xml" href-"planets.xsl"?>
<PLANETS>

    <PLANET>
        <NAME>Hercury</NAME>
        «MASS UNITS-"(Earth - 1)">.0553«/MASS>
        <DAY UNITS""days">58.65</DAY>
        <RADIUS UNITS-"m11es">1516</RADIUS>
        «DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
        «DISTANCE UNITS-"mill1on m1les"">43.4«/DISTANCE><!—В периТелии—>
    </PLANET>
```

Это можно сделать при помощи элемента <xsl:value-of>.

Доступ к значениям узлов

Элемент <xsl:value-of> записывает в результирующий документ строковое значение выражения; в частности, с его помощью можно возвратить значение узла, которым для элемента будет заключенный в элемент текст. Атрибуту выбора элемента <xsl:value-of> можно присвоить выражение XPath, задающее узел или набор узлов. В шаблоне, задающем элементы <PLANET>, обратиться к дочернему элементу <MASS> можно при помощи выражения XPath "child::MASS". Как мы увидим в главе 4, выражения XPath можно писать разными способами: например, вместо "child::MASS" можно просто написать "MASS". Таким образом, получить данные дочерних элементов, таких как <MASS>, <DAY> и т. д., можно следующим способом (листинг 3.1).

Листинг 3.1. Полная версия planets.xsl

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl="http://vjww.w3.org/1999/XSL/Transform">

  <xsl:tempi ate match-"/PLANETS">
      <HTML>
          <HEAD>
              <TITLE>
                  The Planets Table
              </TITLE>
          «/HEAD>
          <B0DY>
```

```
            <H1>
                The Planets Table
            </H1»
            <TABLE B0RDER»"2">
                <TR>
                    <TD>Name</TD>
                    <TD>Mass</TD>
                    <TD>Rad1us</TD>
                    <TD>Day</TD>
                </TR>
                <xsl:apply-templates/>
            </TABLE>
        </BODY>
    </HTML>
</xsl:template»

<xsl:tempi ate match-"PLANET">
   <TR>
      <TDxxsl :value-of select-"NAME"/></TO>
      <TD><xslrvalue-of select-"MASS"/»</TD>
      <TD><xsl :value-of select-"RADIUS"/x/TD>
      <TDxxsl:value-of select-"DAY"/></TD>
   </TR>
</xsl:template»
</xsl -.stylesheet»
```

Создание образцов выбора

Как вы могли ожидать, исходя из того, что "child::MASS" можно представить в виде "MASS" и применения шаблонов, таких как"/", "/PLANETS" и т. д., вплотную познакомиться с созданием образцов выбора не так-то просто — этому посвящена вся глава 4.

Образцы выбора (match pattern) являются подмножеством полного языка XPath, их можно использовать в элементах <xsl:template>, <xsl:key> и <xsl:number>. В частности, в образец можно установить атрибут match у <xsl:template> и <xsl:key>, и атрибуты count и from элемента <xsl:number>. В следующем списке приведен ряд примеров образцов выбора, много других примеров будет показано в главе 4 при подробном обсуждении использования XPath для выбора узлов и атрибутов:

- "/" выбирает корневой узел;
- "*" выбирает элементы узлов (но не всех узлов, как зачастую ошибочно полагают);
- "PLANET" выбирает элементы <PLANET>;
- "PLANET/MASS" выбирает все элементы <MASS>, дочерние для элемента <PLANET>;
- "//PLANET" выбирает все элементы <PLANET>, производные от корневого узла;
- "." выбирает текущий узел (технически это не образец выбора, а выражение XPath, как мы увидим в главе 7).

Образцы можно также использовать в атрибуте select элементов <xsl:apply-templates>, <xsl:value-of>, <xsl:for-each>, <xsl:copy-of> и <xsl:sort>; фактически атри-

бут select этих элементов может содержать полные выражения XPath, а не только образцы выбора. Атрибут select элемента <xsl:value-of> определяет дочерний узел, значение которого нужно получить:

```
<xsl:template match-"PLANET">
      <TR>
         <TD><xsl;value-of  select«"NAME"/></TD>
         <TDxxsl: val ue-of select-'HASS"/></TD>
         <TD><xsl:value-of  select-"RADIUS"/></TD>
         <TD><xsl:value-of  select-"DAY'7»</TD>
      </TR>
  </xsl:template»
```

Теперь пора воспользоваться атрибутом select элемента <xsl:apply-templates>, поскольку это даст нам возможность указать, какой шаблон использовать и когда.

Выбор применяемого шаблона

Вплоть до нынешнего момента я использовал только версию элемента <xsl:apply-templates> по умолчанию, как, например:

```
<TABLE BORDER-"2"»
    <TR>
        <TD>Name</TD>
        <TD>Mass</TD>
        <TD>Radius</TD>
        <TD>Day</TD>
    </TR>
    <xsl:apply-tempiates/>
</TABLE>
```

Простое использование только <xsl:apply-templates/> указывает процессору XSLT осуществлять поиск всех шаблонов, выбирающих дочерние узлы контекстного узла, — это использование по умолчанию. Однако иногда это не лучший вариант, поскольку может потребоваться применять шаблоны в определенном порядке или иным образом выбрать применяемые шаблоны (последнее можно сделать при помощи атрибута select элемента <xsl:apply-templates/>).

Например, до сих пор мы только получали значение каждого элемента <MASS>, <DAY> и <RADIUS> при помощи <xsl:value-of>:

```
<?xml version-"1.0"?>
<xsl:stylesheet  version-"!..0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

    <xsl:tempiate match-"/PLANETS"»
        <HTML»
            <HEAD>
                <TITLE>                ч
                    The Planets Table      „                    ,и
                </TITLE>
            </HEAD>
            <BODY»
                <H1>
                    The Planets Table
```

```
                </H1>
                <TABLE BORDER-"2">
                    <TR>
                        <TD>Name</TD>
                        <TD>Mass</TD>
                        <TD>Radius</TD>
                        <TD>Day«/TD>
                    </TR>
                    <xsl:apply-templates/>
                </TABLE>
            </BODY>
        </HTML>
    </xsl.template»

    <xsl: tempi ate match-"PLANET">
        <TR>
            <TD><xsl:value-of  select-"NAME"/></TD>
            <TDxxsl:value-of  select-"MASS'7></TD>
            <TD><xsl rvalue-of  sel ect-"RADIUS'7x/TD>                    i
            <TDxxsl:value-of  select-"DAY'7»«/TD>
        </TR>
    </xsl:template»
</xsl:stylesheet»
```

Здесь только извлекается «сырое» строковое значение каждого узЛа, которое помещается в HTML-таблицу. Однако может потребоваться дополнительная обработка каждого элемента — например, получить также значения атрибутов UNITS каждого элемента и отобразить их:

```
<?xml  version-"1.0"?>
<?xml-stylesheet  type-"text/xml"  href-"planets.xsl"?»
<PLANETS>

    <PLANET>
        <NAME»Mercury«/NAME>
        <MASS  UNITS-"(Earth  -  1)">.Q553«/MA$S>
        <DAY UNITS-"days"»58.65</DAY»
        <RADIU$ UNITS-"m11es">1516</RADIUS»
        «DENSITY  UNITS-"(Earth  -  1)">.983«/DENSITY»
        «DISTANCE  UNITS-'mllHon  m1les">43.4«/DISTANCE»«l--B  перигелии-•>
    «/PLANET»                                                    -
```

Для этого нельзя просто применить элемент <xsl:value-of>, поскольку он вернет значение узла только как текст, а не как значения атрибута. Вместо этого нужно создать новый набор шаблонов, по одному для каждого интересующего нас элемента: <MASS>, <RADIUS> и <DAY>. У элемента <NAME> нет никаких атрибутов, поэтому для него не нужны шаблоны — нужно только значение узла. Каждому из этих новых шаблонов требуется получить значение элемента, а также значение атрибута UNITS.

Чтобы быть уверенным в том, что эти новые шаблоны применяются в правильном порядке, соответствующем заголовкам HTML-таблицы, я явно перечислю все новые шаблоны, выбирая их один за другим при помощи атрибута select:

```
<2xml version«"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns xsl-"http://www.w3.org/1999/XSL/Transform">

    <xsl:tempi ate matph="/PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <BODY>
                <H1>
                    The Planets Table
                </H1>
                <TABLE BORDER="2">
                    <TD>Name</TD>
                    <TD>Mass</TD>
                    <TD>Radi us</TD>
                    <TD>Day</TD>
                    <xsl:apply-templates/>
                </TABLE>
            <BODY>
        </HTML>
    </xsl:tempiate>

    <xsl:tempi ate match-"PLANET'>
        <TR>
            <TD><xsl:value-of select-"NAME"/></TD>
            <TD><xsl:apply-tempiates select-"MASS"/></TD>
            <TD><xsl:apply-tempiates select-"RADIUS"/></TD>
            <TD><xsl:apply-tempiates select-"DAY"/></TD>
        </TR>
    </xsl:tempiate>
</xsl:stylesheet>
```

ИСПОЛЬЗОВАНИЕ ИМЕНОВАННЫХ ШАБЛОНОВ

К шаблонам можно также обращаться по имени. Этот вопрос будет рассмотрен в главе 9.

Теперь к каждому элементу <MASS>, <RADIUS> и <DAY> применяется новый шаблон: мы не просто получаем строковое значение узла при помощи <xsl:value-of>, мы можем дополнительно обработать каждый элемент — например, прочитать значения атрибута UNITS. Сначала я получу строковые значения каждого из элементов <MASS>, <RADIUS> и <DAY>. Теперь, когда у нас есть шаблон для каждого из ?тих узлов-элементов ц каждый узел является контекстным узлом в своем шаблоне, вместо того, чтобы ссылаться на узел элемента по имени, мы можем сослаться на него как на контекстный узел при помощи выражения XPath"." (листинг 3.2):

Листинг 3.2. Версия planets.xsl с выбором

```
<?xml version-"1.0"?>
<xsl:stylesheet version="l.O"
xmlns:xslhttp://www.w3.org/1999/XSL/Trans form">

    <xsl:template match-"/PLANETS">
f. /. " "jHTML>          9              . •          i:          K'-u /
```

```
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <BODY>
                <H1>
                    The Planets Table
                </H1>
                <TABLE BORDER-"2">
                    <TD>Name</TD>
                    <TD>Mass</TD>
                    <TD»Radius</TD>
                    <TD>Day</TD>
                    <xsl:apply-templates/>
                </TABLE>
            </BODY>
        </HTML>
    </xsl -.template»

    <xsl:tempiate match-"PLANET">
        <TR>
            <TD><xsl:value-of select«'NAME'/></TD>
            <TD><xsl:apply-templates se 1 ect-" MASS' / x / TD>
            <TD><xsl:apply-templates select-"RADIUS7></TD>
            <TD><xsl:apply-templates sel ect-"DAY" /x/TD»
        </TR>
    </xsl:template»

    <xsl:tempiate match-"MASS">
        <xsl:value-of select-"."/>
    </xsl:tempi ate»

    <xsl:template matcfo-"RADIUS"»
        <xsl:value-of select-"."/»
    </xsl:tempiate»

    <xsl: tempi ate matchp«'DAY'>
        <xsl:value-of select-"."/>
    </xsl:tempiate»

</xsl stylesheet»
```

Этот код только воспроизводит то, что мы уже делали ранее в предыдущей версии planets.xsl, использующей <xsl:value-of select-"MASS">, <xsl:value-of select» "RADIUS"> и т. д., то есть читающей и отображающей данные каждого элемента <MASS>, <RADIUS> и <DAY>. Теперь же, когда у нас есть отдельный шаблон для каждого из этих элементов, можно осуществить дополнительную обработку — например чтение значений атрибутов.

Чтение значений атрибутов

Чтобы получить доступ к значению атрибута при помощи XPath, нужно добавить к имени атрибута префикс ⌒. например: "@src", "@height", "©width" и т. д.

Для выбора любого атрибута можно применить выражение Чтобы сослаться на атрибут UNITS в каждом элементе <MASS>, <RADIUS> и <DAY>, следует использовать выражение "@UNITS".. Следовательно, получить значения и отобразить единицы (unit) каждого измерения в planets.xml можно так (листинг 3.3).

Листинг 3.3. Чтение значений атрибутов

```
<?xml version-"1.0"?»
<xsl stylesheet version-"!.6"
xmlns:xs1-"http://www.w3.org/1999/XSL/Tra ns form">

    <xsl template match-"/PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <BODY>
                <H1>
                    The Planets Table
                </H1>
                <TABLE BORDER-"2"»
                    <TD>Name</TD>
                    <TD»Mass</TD>
                    <TD>Radius</TD>
                    <TD>Day</TD>
                    <xsl:apply-templates/>
                </TABLE>
            </BODY>
        </HTML»
    </xsl:template»

    <xsltemplate match-"PLANET"»
        <TR>
            <TDxxsl :value-of select-"NAME"/x/TD»
            <TDxxsl:apply-tempiates select="MASS"/x/TD>
            <TD><xsl:apply-tempiates seleCt»"RADIUS"/></TD>
            <TQxxsl:apply-tempi ates select-"DAY"/></TD>
        </TR»
    </xsl:template»

    <xsl:tempiate match-"MASS">
        <xsl:value-of select-"."/»
        <xs1rvalue-of select-"PUNITS"/>
    </xsl:tempi ate»

    <xslrtemplate match-"RADIUS">
        <xsl:value-of select-"."/>
        <xsl;value-of select-"0UNITS"/>
    </xslrtemplate»

    <xsl -.template match-"DAY'>
        <xsl:value-of select-"."/>
        <xslrvalue-of select-"0UNITS"/>
     </xsl:tempiate»

</xsl:stylesheet»
```

Результаты работы показаны на рис. 3.1, Как видите, теперь мы извлекли строковое значение атрибута UNITS и отобразили его.

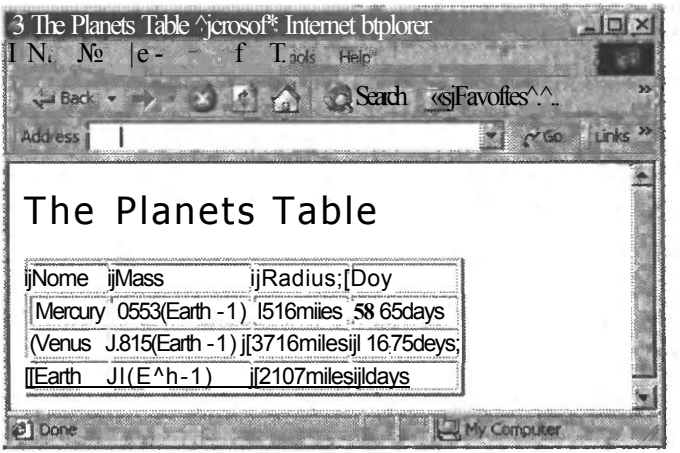


Рис. 3.1. Вывод значений атрибутов, первый вариант

Рисунок 3.1 не совсем точен: обратите внимание на то, что между значением и соответствующей единицей измерения в таблице нет пробела. Процессор XSLT просто поместил в результирующее дерево текст без каких-либо разделителей между текстовыми узлами. Хотя это в точности соответствует требованиям рекомендации XSLT, нам бы хотелось, чтобы элементы таблицы выглядели как «1516 miles», а не «1516miles». Как нам добавить этот дополнительный пробел?

Элемент <xsl:text>

Работа с пробелами всегда обсуждается при рассмотрении XSLT, и в данной главе я уделю этой теме некоторое время. Вставить единственный пробел несложно при помощи элемента <xsl:text>, который используется для вставки символьного текста непосредственно в дерево вывода. У этого элемента только один атрибут:

- disable-output-escaping. Устанавливается в yes для того, чтобы такие символы, как < и >, выводились буквально, а не как < и >. По умолчанию установлен в по.

Этот элемент может содержать только текстовый узел.

Текстовые узлы создаются при помощи элемента <xsl;text>, позволяющего по ходу дела замещать элементы целиком на текст. Одна из целей применения <xsl: text> — сохранить символы-разделители, как в следующем примере (листинг 3.4), в котором элемент <xsl:text> используется для вставки пробелов.

Листинг 3.4. Вставка пробелов в таблицу стилей

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
```

продолжение xP

Листинг 3.4 (продолжение)

```
xmlns:xs1-"http://www.w3.org/1999/XSL/Trans form">

    <xsl:template match»"/PLANETS"»
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <BODY>
                <H1>
                    The Planets Table
                </H1>
                <TABLE>
                    <TD>Name</TD>
                    <T0>Mass</TD>
                    <TD>Radius</TD>
                    <TD»Day</TD>
                    <xsl:apply-templates/>
                </TABLE>
            </BODY>
        </HTML>
    </xsl:template»

    <xsl:tempiate match-"PLANET">
        <TR>
         <TD><xsl:value-of select-"NAME"/><7TD>
         <TD><xsl:apply-templates select""MASS"/></TD>
         <TD><xsl:apply-templates select-"RADIUS"/></TD>
        </TR>
    </xsl:template»

    <xsl template match-"MASS">
      <xsl:value-of select-"."/>
      <xsl:text> </xsl:text>
      <xsl:value-of select-"@UNITS"/>
    </xsl:tempiate»

    <xsl:template match-"RADIUS">
      <xsl:value-of select-"."/»
      <xsl:text> </xsl:text>
      <xsl:value-of select-"@UNITS"/>
    </xsl:tempi ate»

    <xsl:template match-"DAY">
      <xsl:value-of select-"."/>
      <xsl:text> </xsl:text>
      <xsl:value-of select-"@UNITS"/>
    </xsl:tempiate>

</xsl:stylesheet»
```

Новый результат виден на рис. 3.2, на котором между числовыми значениями и их единицами измерения вставлены пробелы.

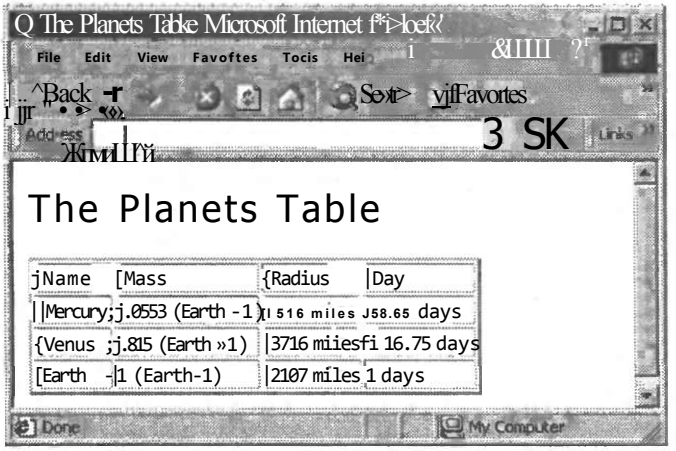


Рис. 3.2. Вывод значений атрибутов, второй вариант

Как видите, элемент <xsl:text> весьма удобен. Тем не менее вам следует знать вот что: по умолчанию элементы <xsl:text> заменяют символы, которые мохут входить в разметку, на эскейп-последовательности. Например, <xsl:text>Here is a greater-than sign: x/xsl:text> будет выведено как «Here is a greater-than sign: >,», а не как «Here is a greater-than sign: >». А если попытаться использовать < внутри элемента <xsl:text>, процессор XSLT посчитает, что вы пытаетесь открыть элемент внутри элемента <xsl:text>, что неверно. Как же вывести значащие символы, такие как < и >, если есть необходимость? Это можно сделать путем отключения вывода ESC-последовательностей (disabling output escaping).

УСОВЕРШЕНСТВОВАНИЯ В XSLT 2.0

Один из вопросов, которые призван решить XSLT 2.0, это упрощение импорта подобного текста без разбора из других файлов.

Отключение вывода эскейп-последовательностей

При помощи <xsl:text> можно включать в выходной документ непосредственно символы < и &, а не последовательности < и &атр;. Для этого следует установить атрибут disable-output-escaping элемента <xsl:text> в yes («да», по умолчанию он установлен в по, «нет»). В приведенном ниже примере я при помощи <xsl:text> непосредственно пишу текст "<PLANET/>" в выходной документ:

```
<?xml version="l 0"?>
<xsl:stylesheet version»"10" xml ns: xs 1 =*"http://www w3.org/1999/XSL/Transform"»

<xsl:template match-"PLANETS">
    <HTML>
        <HEAD>
            <TITLE>
                Planets
```

```
            </TITLE>
        </HEAD>
        <BODY>
            <xsl:apply-templates select-"PLANET"/>
        </BODY>
    </HTML>
</xsl:template»

<xsl:template match-"PLANET"»
    <xsl:text disable-output-escaping - "yes"»
        &lt:PLANET/&gt:
    </xsl:text»
</xsl:tempi ate»

</xsl:stylesheet»
```

Результат следующий:

```
<HTML>

    <HEA0>
        <TITLE»
            Planets
        </TITLE>
    </HEAD»

    <BODY>
      <PLANET/>
      <PLANET/>
      <PLANET/>
  </BODY>

</HTML»
```

Разумеется, не обязательно выводить <PLANET/> при помощи <xsl:text>: этот элемент можно было поместить непосредственно в элемент буквального результата. Но что делать в тех случаях, когда процессор XSLT не сможет распознать нужный нам в выходном документе элемент как фактический элемент? Например, в преобразованных документах XHTML вам потребуется <!DOCTYPE html PUBLIC "-//W3C//DTD XHIML 1.0 Transitional//EN">, но процессор XSLT сочтет, что это не хорошо сформированный XML. Как поместить этот элемент в выходные данные?

Можно попробовать поместить этот элемент в раздел <![CDATA[ ]>, как мы увидим в главе 6, и попытаться трактовать его как простые символьные данные, но процессоры XSLT все равно заменят < на <, а > на >.

Правильный способ добавить в вывод элемент <!DOCTYPE> — использовать атрибут doctype-public элемента <xsl:output>, как мы увидим в главе 6, но в качестве демонстрации для этих целей здесь я отключу вывод ESC-последовательностей в элементе <xsl:text> (хотя этот способ не рекомендован для создания элементов <!DOCTYPE>). Вот как это выглядит:

```
<?xml version-"1.0"?»
<xsl:stylesheet version-"!.0"
```

```
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">
    <xsl.output method-"xml"/>

    <xsl:tempi ate match-"/PLANETS">
        <xsl:text d1sable-output-escaping="yes">
            &lt;IDOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"&gt:
        </xsl:text>
        <HIML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <BODY>
                <H1>
                    The Planets Table
                </HI>
                <TABLE BORDER-"2">
                    <TO>Name</TD>
                    <TD>Mass</TD>
                    <TD>Radius</TD>
                    <TD>Day</TD>
                    <xsl:apply-templates/>
                </TABLE>
            </BODY>
        </HIML>
    </xsl:template»

    <xsltemplate match-"PLANET"»
       <TR>
          <TD»<xsl value-of select-"NAME"/></TD»
          <TDxxsl: apply-tempi ates select-"MASS"/»</TD>
          <TD»<xsl apply-templates select-"RADIUS"/»</TD>
          <TD»<xsl:apply-templates select-"DAY"/»</TD»
       </TR»
   </xsl:template»

    <xsl-.template match-"MASS">
        <xsl:value-of select-"."/»
        <xsl:text> </xsl text»
        <xsl:value-of select-"@UNITS"/>
    </xsl:tempi ate»

    <xsl;tempiate match-"RADIUS">
        <xsl:value-of select-"."/>
        <xsl text» </xsl:text>
        <xsl:value-of select-"@UNITS"/>
    </xsl:tempiate»

    <xsl:tempi ate match-"DAY">
        <xsl value-of select-"."/»
        <xsl:text> </xsl:text>
        <xsl value-of select-"@UNITS"/>
    </xsl:tempiate»

</xsl:stylesheet»
```

А вот результат:

```
<?xml version»"1.0" encoding-"UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN">
    <HTML>
        <HEAD>
            <TITLE>
                The Planets Table
            </TITLE>
        </HEAD>

        <B0DY>
            <H1>
                The Planets Table
                .
                .
                .
```

Далее в книге будут приведены другие примеры использования <xsl:text>, в том числе — при обсуждении символов-разделителей далее в этой главе.

Написание значений атрибутов

В XSLT есть несколько способов написать значения атрибутов в выходные документы, и наиболее мощный — это создать атрибуты с нуля при помощи элемента <xsl:attribute>, как мы увидим в главе 6.

Однако можно также во многих случаях использовать шаблоны значений атрибутов, с которыми мы познакомимся в этой главе.

Предположим, например, что нам требуется преобразовать текст в таких элементах, как <MASS>, <DAY> и <NAME>, в атрибуты элементов <PLANET>, преобразуя planets.xml в следующий вид:

```
<?xml version-"l 0" encoding»"UTF-8"?>
<PLANETS>
    <PLANET DAY-"58.65 days" RADIUS-"1516 miles"
        MASS-".0553 (Earth - 1)" NAME-"Mercury"/>
    <PLANET DAY-"116.75 days" RADIUS-"3716 miles"
        MASS-" 815 (Earth - 1)" NAME-"Venus"/>
    <PLANET DAY-"1 days" RADIUS="2107 miles"
        MASS-"1 (Earth - 1)" NAME-"Earth"/>
</PLANETS>
```

Чтобы создать преобразование, нельзя просто использовать следующее выражение, в котором я взял значения элементов <NAME>, <MASS> и <DAY> и попытался сделать их значениями атрибутов:

```
<xsl -.template match-"PLANET">
    <PLANET NAME-"<xsl:value-of select""NAME"/>"
        MASS-"<xsl:value-of select-"MASS"/>"
        DAY-"<xsl:value-of select="DAY"/>"
    />
```

Это не будет работать, поскольку нельзя использовать < внутри значений атрибутов, как я сделал в предыдущем примере. В XSLT для этого есть несколько способов. Один из них — использовать шаблоны значений атрибутов.'

Шаблоны значений атрибутов

Шаблон имени значения атрибута не имеет ничего общего с теми шаблонами, с которыми мы до сих пор работали, — то есть с шаблонами для создания правил таблицы стилей. Напротив, применение шаблона значения атрибута (attribute value template) означает лишь, что значение атрибута может быть установлено во время выполнения.

В этом случае атрибут можно установить в значение выражения XPath (более подробно эта тема изучается в главе 4), если заключить выражение в фиэдшые скобки. Например, чтобы установить атрибут NAME в строковое значение элемента <DESCRIPTION>, дочернего элемента контекстного узла, это значение можно присвоить следующим образом: NAME-{DESCRIPTION}.

В листинге 3.5 приведен правильный код XSLT, в котором значения элементов <NAME>, <MASS>, <RADIUS> и <DAY> присваиваются атрибутам элемента <PLANET> с теми же именами.

Листинг 3.5. Работа с шаблонами значений атрибутов

```
<?xml version^'1.O"?»
<xsl:stylesheet versi on-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Tr$nsform">
<xsl:output method-"xml"/>

<xsl:tempi ate match-"PLANETS">
    <xsl:copy>
        <xsl:apply-templates select-"PLANET"/>
    </xsl:copy>
</xsl:template»
<xsl .-template match-"PLANET">
    <PLANET NAME-"{NAME}"
        MASS-"{MASS}"
        RADIUS-B{RADIUS}"
        DAY-"{DAY}H
    />
</xsl:template»

</xsl:stylesheet»
```

Вот как это делается; посмотрите теперь на результирующий документ, в котором значения в различных элементах были преобразованы в атрибуты:

```
<?xml version-"1.0" encod1ng-"UTF-8"?>
<PLANETS>
    <PLANET DAY-"58.65 " RADIUS-"1516 " MASS-".0553 " NAME-"Mercury"/>
    <PLANET DAY-"116.75 " RADIUS-"3716 " MASS-".815 " NAME-"Venus"/>
    <PLANET DAY-"1 " RADIUS-"2107 " MASS-"1 " NAME-"Earth"/»
</PLANETS»
```

Предположим теперь, что нам также нужно включить все единицы измерения. Каждый элемент <MASS>, <NAME> и <RADIUS> содержит атрибут UNITS, задающий единицы измерения, и можно извлечь эти значения. Контекстным узлом является элемент <PLANET>, поскольку шаблон установлен для выбора этого элемента, поэтому на дочерние элементы <MASS>, <NAME> и <RADIUS> можно ссылаться как

"MASS", "NAME" и "RADIUS". Для обращения к атрибуту UNITS этих элементов можно использовать синтаксис "MASS/@UNITS", "NAME/@UNITS", и "RADIUS/@UNITS":

```
<?xml version-"1.0"?»
<xsl:stylesheet version="1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">
<xsl output method-"xml"/>

<xsl:template match-"PLANETS">.
    <xsl:copy>
        <xsl:apply-templates select-"PLANET"/>
    </xsl:copy>
</xsl:template»

<xsl:tempiate match-"PLANET">
    <PLANET NAME-"{NAME}"
        MASS-"{MASS} -{MASS/@UNITS}"
        RADIUS-"{RADIUS} -{RADIUS/0JNITS}"
        DAY»"{DAY} -{DAY/PUNITS}"
    />
</xsl:template»

</xsl:stylesheet»
```

И вот результат, включающий единицы измерения:

```
<?xml version-'! 0" encoding-"UTF-8[4]?>
<PLANETS>
    <PLANET DAY-"58.65 days" RADIUS-"1516 miles"
        MASS   0553 (Earth = 1)" NAME-"Mercury"/>
    <PLANET DAY-"116.75 days" RADIUS-"3716 miles"
        MASS-".815 (Earth - 1)" NAME-"Venus"/»
    <PLANET DAY-""1 days" RADIUS-"2107 miles"
        MASS-"1 (Earth = 1)" NAME-"Earth"/»
</PLANETS»
```

Заметьте, что в шаблонах значений атрибутов нельзя использовать вложенные фигурные скобки, и в выражении, использующем фигурные скобки, — таком как function printHello {cout«'Hello';} — фигурные скобки необходимо удваивать, для того чтобы процессор XSLT их игнорировал: function printHello {{cout«'Hello';}}.

Шаблоны значений атрибутов всегда работают с контекстным узлом. Тем не менее, нельзя использовать шаблоны значений атрибутов в произвольном месте таблицы стилей, что зачастую и вызывает затруднения у XSLT-разработчиков. Шаблоны значений атрибутов можно использовать только в следующих местах:

- элементы буквального результата;
- элементы расширения (см. главу 5);
- <xsl:attribute>. Здесь можно использовать атрибуты name и namespace (см. главу 6);
- <xsl:element>. Здесь можно использовать атрибуты name и namespace (см. главу 6);
- <xsl:number>. Здесь можно использовать атрибуты format, lang, letter-value, grouping-separator и grouping-size (см. главу 4);

- <xsl:processing-instruction>. Здесь можно использовать атрибут name (см. главу 6);
- <xsl:sort>. Здесь можно использовать атрибуты Lang, data-type, order и case-order (см. главу 5).

В главе б эта тема рассмотрена более подробно: мы узнаем, как создавать атрибуты (и новые элементы) с нуля. Дополнительная информация об использовании выражений XPath в шаблонах значений атрибутов приведена в главе 7.

Обработка символов-разделителей

Поначалу символы-разделители (whitespace) доставляют авторам XSLT много хлопот. В главе 2 объяснялось, что «чистые» узлы-разделители — это текстовые узлы, содержащие только символы-разделители (пробелы, возвраты каретки, переводы строки и символы табуляции). Эти узлы по умолчанию копируются из исходного документа.

Заметьте, что в таблице стилей также могут быть узлы-разделители:

```
<xsl:tempiate match-"PLANETS">
    <xsl:copy>
        <xsl:apply-tempiates select-"PLANET"/>
    </xsl:copy>
</xsl:template»
```

В нашем случае пробелы используются для выравнивания элементов таблицы стилей, а возвраты каретки — для разрежения кода. Чистые узлы-разделители, такие, как этот, не копируются из таблицы стилей в выходной документ. Заметьте, однако, что разделитель из следующего элемента <TITLE> копируется в выходной документ, так как это не чистый узел-разделитель (он также содержит текст «The Planets Table»):

```
<xsl:template match-"/PLANETS"»
    <HTML>
        <HEAD>
            <TITLE>
                The Planets Table
            </TITLE>
```

Если вы хотите убрать этот разделитель и сохранить выравнивание, можно использовать пустые элементы <xsl:text>, так чтобы символы-разделители стали чистыми узлами-разделителями:

```
<xsl:template match""/PLANETS"»
    <HIM>
        <HEAD>
            <TITLE»
                <xsl:text/>The Planets Table<xs1:text/>
            </TITLE>
```

Чистые узлы-разделители не копируются йз таблицы стилей в выходной документ, ейли'+олько они не находятся ййутри элёмента <xsl:text>, иЛиуэлёмента, $ который они вложены, атрибут xml:space не убтановлейь «preserve» (сохранить).

С другой стороны, по умолчанию XSLT сохраняет текстовые узлы' разделители в исходном документе и копирует их в результирующий документ. Возьмите уже рассмотренную нами копирующую таблицу стилей, которая копирует все элементы из исходного документа в результирующий:

```
<?xml version-"1.0"?>
<xsl stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">
    <xsl:output methocK'xml"/>
    <xslrtemplate match-"*">
        <xsl:copy>
            <xsl:apply-templates/>
        </xsl:copy>
    </xsl:tempiate>
</xsl:stylesheet»
```

и примените ее к planets.xml; все символы-разделители будут также скопированы в результирующий документ:

```
<?xml version-"1.0"?>
<?xml-stylesheet type-"text/xml" href-"planets.xsl"?>
<PLANETS>

    <PLANET>
        <NAME>Mercufy</NAME>
        <MASS UNITS-"(Earth - 1)">.0553</MASS>
        <DAY UNIT5-"days"*58.65</DAY>
        <RADIUS UNITS-"miles">1516</RADtUS>
        <DENSITY UNITS-"(Earth - ir>.983</DENSITY>
        <DISTANCE UNITS-"million miles">43.4</DISTANCE><!-B перигелии--»
    </PLANET>
```

Тем не менее, иногда требуется удалить разделители, используемые при форматировании входных документов — это Можно сделать при помощи элемента <xsl:strip-space>.

Элементы <xsl:strip-space> и <xsl:preserve-space>

Элемент <xsl:strip-space> дает указание процессору XSLT убрать бее чистые узлы-разделители (также называемые «потребляемыми», expendable, узлами-разделителями) из. выходного документа. Чистый узел-разделитель состоит только

из символов-разделителей и не содержит текст какого-либо другого вида. У этого элемента только один атрибут:

- elements (обязательный), Задает элементы, из,которых ну?кно убрать символы-разделитрщ. Пре^стадляет срб^й Qn^cpK радд^л^нрщ я^^вдаци-разделителями NameTesrt. (именами .^^оО^^вд^р»^^ Я^^ШЙяда^ЭД

Элемент не включает содержимого.

Например, чтобы удалить все узлы-разделители из planets.xml, можно применить <xsl:strip-space elements-"* "/> следующим образом:

```
<?xml version-"1.0"?»
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"»
    <xsl:strip-space elements-"*"/»
    <xsl:output method-"xml"/>
    <xsltemplate match-"*"»
        <xsl:copy>
            <xsl:apply-templates/»
        </xsl:copy>
    </xsl -.template»
</xsl:stylesheet»
```

Вот результирующий документ, полученный после применения этой таблицы стилей к planets.xml. Обратите внимание на то, что убраны все символы-разделители, в том числе все символы новой строки:

```
<?xml version-"1.0" encoding="utf-8"?><PLANETS><PLANET»<NAME»Mercury</NAME>
<MASS>.0553
</MASS><DAY»58.65</DAY><RADIUS>1516</RADIUS><DENSITY>.983</DENSITY»
<DISTANCE>43.4</DISTANCE></PLANETxPLANET»<NAME>Venus</NAME><MASS> ,315</MASS>
<DAY»116.75</DAY><RADIUS»3716</RADIUS><L)ENSITY> 943</DENSITY><DISTANCE»66.8</DISTANCE>
</PLANET><PLANET»<NAME>Earth</NAME><MASS>I</MASS><DAY>I</DAY><RADIUS>21Q7</RAOIUS>
<DENSITY»1</DENSITY»<DISTANCE»128.4</DISTANCE></PLANET»</PLANETS» '
```

Заметьте, что таким образом удаляются только чистые узлы-разделители. Например, текст элемента <TITLE>Volcanoes for Dinner</TITLE> не содержит чистых текстовых узлов-разделителей, поэтому текст «Volcanoes for Dinner» будет сохранен в выходном документе вместе с пробелами. Так будет даже тогда, когда текст будет содержать несколько пробелов подряд, как в «Volcanoes for < Dinner».

В некоторых ситуациях может потребоваться не удалять все узлы-разделители из всего документа; задать элементы, в которых следует сохранить узлы-разделители, можно при помощи элемента <xsl: preserve-space». У этого элемента такой же атрибут, что и у <xsl:strip-space>:

- elements (обязательный). Задает элемент^, в которых нужно сохранить символы-разделители. Представляет собой список, раздоенных символа-^ми-разделителями NameTest (именами или обобщенными именами с символами подстановок),

Фактическй элемент <xsl:preserve-space> является элементом по умолчанию для всех элементов в XSLT. Если ®ы использовали элемент *sxsl:strip-space>, все

равно можно указать, в каком элементе или элементах нужно сохранить узлы-разделители, установив атрибут elements элемента <xsl:preserve-space> в список этих элементов:

```
<?xml version-"1.0"?»
<xsl:stylesheet vers1on-"l.0"
xmlns:xsl""http://www w3,org/1999/XSL/Transform">
    <xsl:strip-space elements-"*"/»
    <xsl:preserve-space elements-"MASS RADIUS"/»
    <xsl:output method-"xml"/>
    <xsl:tempiate match-"*»
        <xsl copy»
            <xsl:apply-templates/»
        </xsl:Copy»
    </xsl template»
</xsl:stylesheet»
```

Обсужденные средства удаления и сохранения разделителей могут показаться слишком сложными для форматирования выходных документов выравнивающими пробелами, но, к счастью, существует простой способ: атрибут indent элемента <xsl:output> позволяет автоматически выровнять выходной документ.

Автоматическое выравнивание

Элемент <xsl:output> поддерживает атрибут indent который устанавливается в «yes» или «по», и указывает процессору XSLT, нужно ли выравнивать результирующий документ. Как правило, выравнивание результирующего документа не имеет большого значения, поскольку с ним работает приложение, которому все равно, выровнен документ или нет, как мы видели в примерах преобразований XML-XML и XML-HTML. Однако иногда требуется представить результирующий документ в виде простого текста, и в таких случаях выравнивание документа для отображения иерархической структуры может оказаться удобным.

Способ работы процессора XSLT с переменной выравнивания не регламентируется W3C и зависит от процессора, поэтому для получения требуемого результата нужно экспериментировать. Пусть, например, у нас есть версия planets.xml без какого-либо выравнивания:

```
<?xml version-"1.0"?>
<?xml-stylesheet type-"text/xml" href-"planets.xsl"?>
<PLANETS>

<PLANET>
<NAME»Mercury</NAME»
<MASS UNITS-"(Earth - 1)"».0553</MASS>
<DAY UNITS-"days">58.65</DAY>
<RADIUS UNITS-"miles"»1516</RADIUS>
«DENSITY UNITS-"(Earth - 1)"».983</DENSITY>
«DISTANCE UNITS-"million miles">43.4</DISTANCE><!--B леригелии-->
</PLANET»

<PLANET>
<NAME»Venus</NAME»
<MASS UNITS-"(Earth - 1)"».815</MASS»
```

```
<DAY UNIT$-"days"»116.75«/DAY>
«RADIUS UNITS-"miles">3716«/RADIUS>
«DENSITY UNITS-"(Earth * 1)">.943</DENSITY>
«DISTANCE UNITS-"million mi 1es">66.8«/DISTANCE><!—B перигелии—»
«/PLANET»

«PLANET»
<NAME»Ea rth«/NAME»
«MASS UNITS-"(Earth - 1)"»1«/MASS»
«DAY UNITS-"days"»l«/DAY>
«RADIUS UNITS-"miles"»2107«/RADIUS»
«DENSITY UNITS-"(Earth - 1)"»1«/DENSITY>
«DISTANCE UNITS-"million miles">128.4«/DISTANCE>«!—B перигелии—»
«/PLANET»

«/PLANETS»
```

При помощи элемента <xsl:output indent-"yes"/> можно указать процессору XSLT осуществлять выравнивание документа при преобразовании его в HTML (листинг 3.6).

Листинг 3.6. Таблица стилей, задающая выравнивание

```
<?xml version-"1.0"?>
«xsl:stylesheet vers1on-"l.O"
xmlns xsl-"http //www w3 org/1999/XSL/Transform">
    «xsl:output 1ndent-"yes"/>

    «xsl -.template match""/PLANETS"»
«HTML»
«HEAD»
«TITLE»
          The Planets Table
      «/TITLE»
«/HEAD»
«BODY»
<H1>
          The Planets Table                                    r>'-
      </Hl>                                                  .»,
«TABLE BORDER-"2"»
«TD>Name«/TD>
<TD>Ma$s«/TD>
«TD»Radius«/TD»
<TD»Day«/TD>
«xsl:apply-templates/»
«/TABLE»
«/BODY»
</HTML>
«/xsl:tempiate»

«xslrtemplate match-"PLANET"»
«TR>
<TD»«xsl :value-of select-"NAME"/x/TD>
<TD»«xsl:value-of select-"MASS"/»«/TD>
<TD»«xsl:value-of select-"RADIUS"/>«/TD>
<TD»«xsl:value-of select-"DAY"/x/TD>
</TR>
«/xsl template»
</xsl:stylesheet»
```

Результат применения таблицы с использованием процессора Saxon (в повтором особенно хорошо реализовано выравнивание) с требуемым выравниванием:

```
<HTML>
  <HEAD>
    <meta http-equ1                                   ΠNS6C| Π

    <TITLE>
      The Planets Table

    </TITLE>
  </HEAD>
  <BODY>
    <H1>

      The Planets Table

    </H1>
    <TABLE BORDER="2">
      <TD>Name</TD>
      <TD>Mass</TD>
      <TD>Radius</TD>
      <TD>Day</TD>

      <TR>
        <TD>Mercury</TD>
        <TD>.0553</TD>
        <TD>1516</TD>
        <TD>58 65</TD>
      </TR>

      <TR>
;i!   .4>ilt;   , <TQ>Vepu5</TD>                      \n . o          ,.,
,       <T0>.815<(TD>
      * <TD>3716</TD>                                                 •
        <TH>116  75</TD>                                i           nAJ%
      </TR>

      <TR>
        <TD>Earth</TD>
        <TD>1</TD>
        <TD>2107</TD>
        <TD>1</TD>
      </TR>

    </TABLE>
  </BODY>                                                  4
</HTML>
```

Как видите, в XSLT обработке символов-разделителей приходится уделять достаточное внимание, но процедура упрощается, если вы знаете, что происходит.

ВЫРАВНИВАНИЕ ДОКУМЕНТОВ В ЭТОЙ КНИГЕ

Способ выравнивания документов зависит от конкретного процессора XSLT. В этой книге документы выровнены для удобочитаемости, даже если в действительности документы не были выровнены процессором XSLT.

Правила по умолчанию в шаблоне

Взгляните на следующую таблицу стилей XSLT — в ней заданы правила для выбора корневого узла, узлов <PLANETS> и узлов <PLANET>:

```
<?xml version="l.0"?>
<xsl:stylesheet version-"l.O"
xmlns:xsl-"http.//www w3.org/1999/XSL/Transform"»

    <xsl:tempi ate match-"/">
        <HIML>
            <xsl:apply-templates/>
        </HIML>
    </xsl template»

    <xsltemplate match-"PLANETS">
        <xsl:apply-templates/>
    </xsl.template»

    <xsl:template match-"PLANET"»
        <P>
            <xsl:value-of select-"NAME"/>
        </P>
    </xsl:template»

</xsl:stylesheet»
```

Обратите внимание на правило для элемента <PLANETS>: в нем просто используется элемент <xsl:apply-templates> для применения шаблонов ко всем дочерним узлам. Однако при обработке шаблона существует правило по умолчанию: если для элемента не задано правило, автоматически вызывается <apply-templates/>. Таким образом, следующая таблица стилей, в которой опущено правило для <PLANETS>, делает в точности то же, что и предыдущая:

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http //www w3.org/1999/XSL/Transform"»

    <xsltemplate match-"/"»
        <HIML>
            <xsl:apply-templates/»
        </HIML>
    </xsl:tempiate»

    <xsl template match-"PLANET"»
        <P»
            <xsl:value-of select-"NAME"/»
      ► </P> •  . C
    </xsl template»

</xsl:stylesheet»
```

В этом случае я воспользовался преимуществом правил по умолчанию для шаблона. Ниже приведены правила для каждого вида узлов, которые будут применены, если не задать правило для узла явно:

- **Корневой узел.** По умолчанию вызывается <xsl:apply-templates/>;
- **Узлы элементов.** По умолчанию вызывается <xsl:apply-templates/>;
- **Узлы атрибутов.** Копирует в результирующий документ значение атрибута, однако копирует его как текст, но не как атрибут;
- **Текстовые узлы.** Копирует в результирующий документ текст;
- **Узлы комментариев.** Нет обработки **XSLT,** ничего не копируется;
- **Узлы инструкций обработки.** Нет обработки XSLT, ничего не копируется;
- **Узлы пространств имен.** Нет обработки XSLT, ничего не копируется.

Наиболее важное правило по умолчанию применяется к элементам и может быть выражено следующим образом:

```
<xsl:template match-"*">
    <xsl:apply-templates/»
</xsl:template»
```

Это правило приведено здесь только для гарантии того, что каждый элемент, от корня и ниже, будет обрабатываться при помощи <xsl:apply-templates/>, если не предоставить другого правила, которое перекроет правило по умолчанию.

Правило по умолчанию для текстовых узлов можно выразить следующим образом: функция XSLT **text** выбирает текст узла, так что текст текстового узла добавляется в выходной документ:

```
<xsl:tempi ate match-"text()">
    <xsl:value-of select-"."/>
</xsl:template»
```

Правило по умолчанию того же вида применяется к атрибутам, которые добавляются в выходной документ при помощи следующего правила, где выражение "@*" выбирает любой атрибут:

```
<xsl:tempi ate match-"®*"»
    <xsl:value-of select-"."/»
</xsl:tempiate»
```

По умолчанию инструкции обработки не вставляются в выходной документ, поэтому их правило по умолчанию можно выразить просто при помощи следующей функции-инструкции обработки XSLT, которая выбирает инструкции обработки (как мы увидим В главе 8):

```
<xsl:template match-"processing-instruct1on()'7»
```

То же верно для комментариев — их правило по умолчанию может быть выражено при помощи функции XSLT comment, которая также будет рассмотрена в главе 8:

```
<xsl: tempi ate match-"comment ()"/»
```

Подведем итоги рассмотрения правил по умолчанию: если вообще не задать никаких правил, все разбираемые символьные данные входного документа будут вставлены в выходной документ. Вот как выглядит таблица стилей, в которой не задано никаких явных правил:

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsb"http://www.w3.org/1999/XSL/TranSform">
</xsl:stylesheet»
```

А вот результат применения этой таблицы стилей к planets.xml. Заметьте, что правило по умолчанию для атрибутов не применялось, потому что они не являются дочерними по отношению к другим узлам:

```
<?xml version-"1.0" encoding-"UTF-8"?>

    Mercury
    .0553
    58.65
    1516
    .983
    43.4

    Venus
    .815
    116.75
    3716
    .943
    66.8

    Earth
    1
    1
    2107
    1
    128.4
```

ПРАВИЛА ПО УМОЛЧАНИЮ И INTERNET EXPLORER

Одна из проблем при работе с XSLT в Internet Explorer 5.5 или младше состоит в том, что браузер не предоставляет никаких правил по умолчанию. Необходимо задавать все правила самостоятельно, если только не установлен процессор MSXML3 в режиме замены (подробнее см. главу 2) или вы не обновили браузер до Internet Explorer 6.0.

Кроме того, узлы-разделители исходного документа сохраняются, поэтому можно считать, что следующее правило также является правилом по умолчанию: <xsl:output preserve-space-"*"/*.

Удаление содержимого

Если ваше правило для узла не работает, то есть оно пусто, содержимое выбранного узла не будет скопировано в выходной документ. Таким способом при создании выходного документа можно выборочно удалять содержимое из исходного документа.

Предположим, нам нужно удалить из planets.xml все данные о планетах, за исключением их названий и данных о массе. Следующая таблица стилей выполняет данную задачу.

Листинг 3.7. Удаление содержимого

```
<?xml version-"1.0"?>                                                    к
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3 org/1999/XSL/Transform">
<xsl:strip-space elements-"*"/»
<«sliotitput method-"xml" Indent-"yes"•»

<xsl template match-"/"»
        <xsl:apply-templates/»
</xsl:tempi ate»
                                  O i "4 < "> | 4 O i к« f VT Я^ ' ' 7 > » ( * 1l C S\ I U Л
    <xsl: tempi ate match-" PLANETS"»  * ^        "  ^  '  '     * ' '  "   ~     '
        <xsl.apply-templates/»
    </xsl:template»

    <xslrtemplate match-"PLANET"»
        <xsl:copy>
            <xsl:apply-templates/»
        </xsl:copy>
    </xsl template»

    <xsl:tempi ate match-"NAME">
        <xsl:copy>
            <xsl apply-tempiates/»
        </xsl:copy»
    </xsl:tempi ate»

    <xslrtemplate matchv'MASS">
        <xsl:copy>
        <xsl:value-of select-"."/»
        <xsl:value-of select-"@UNITS"/>
        </xsl:copy>
    </xsl:tempi ate»
    <xsl: tempi ate matcfrK'RADIUS"»
    </xsl:tempiate»

    <xsl:tempi ate match-"DAY">
    </xsl:tempiate»

    <xsl rtemplate matchr"DENSITY"»
    </xslrtemplate»

    <xsl rtemplate match-*"DISTANCE">
    </xslrtemplate»
</xsl:stylesheet»
```

А вот результирующий документ (отметьте, что я сохранил только элементы <NAME> и <MASS>):

```
<?xml version-"1.0" encoding-"UTF-8"?>
<PLANET>
    <NAME»Mercury</NAME>
    <MASS».0553(Earth * 1)</MASS>
</fLANET*
```

```
<PLANET>
    <NAME>Venus</NAME>
    <MASS> 815(Earth - 1)</HASS>
</PLANET>

<PLANET>                                                    omw. .    .>NHf»iSE\! Л.£ 1HHTЭMД
    <NAME>Ea rth</NAME>
    <MASS>1(Earth - 1)</MASS>
</PLANET>
```

Таким способом можно фильтровать XML-документЫ) создавая новые XML-документы только с требуемыми данными.

Разрешение конфликтов шаблонов

Еще одним важным аспектом работы с шаблонами является разрешение конфликтов. Если двум шаблонам удовлетворяют один и тот же узел или набор узлов, для определения применяемого шаблона XSLT учитывает их приоритет.

У каждого шаблона есть приоритет по умолчанию, основанный на значении атрибута select. Как правило, чем более сужающим является правило выбора или выражение (например, "PLANET" и "*"), тем выше его приоритет. В главе 4 мы рассмотрим, как процессор определяет приоритеты и как он работает с шаблонами, имеющими одинаковый приоритет.

Приоритет шаблона можно установить при помощи атрибута priority. В приведенном ниже листинге 3.8 правило, созданное элементом <xsl:template priority» "1"/>, имеет меньший приоритет, чем правило, созданное элементом <xsl:template priority-"2"/>.

Листинг 3.8. Установка приоритета шаблона

```
<?xml version-"1.0"?>
<xsl:stylesheet version="1.0"
xml ns: xsl-"http://www.w3 org/1999/XSL/Transform">                    илвни– $ft»l дов. ><•

    <xsl:tempiate match-"/PLANETS">
        <HTML>
            <HEAD>                                          r;vj                [«;*;
                <TITLE>                                                         ;*• -л •
                    The Planets Table
                </TITLE>....
            </HEAD>
            <B0DY>
                <H1>
                    The Planets Table
                </HI>
                <TABLE BORDER-"2">
                    <TR>
                        <TD>Name</TD>                                 4 "
                        <TD>Mass</TD>
                        <TD>Radius</TO>
                        <TD>Day</TD>
                    </TR>                                   ,
```

продолжение &

Листинг 3.8 (продолжение)

```
                    <xsl:apply-templates/>
                </TABLE>
            </BODY>
        </HTML>
    </xslrtemplate»

    <xsl template match-"PLANET">
        <TR>
            <TDxxsl: value-of sel ect-" NAME "/></TD>
            <TD><xsl: apply-templ ates select-"MASS'7x/TD>
            <TDxxsl: apply-templ ates select-"RADIUS"/></TD>
            <TD><xsl:apply-templates select-"DAY"/></TD>
        </TR>
    </xsl tempiate>

    <xslrtemplate match-/"MASS' prior1ty-"2">
        <xsl:value-of select»"."/>
        (<I>Very</I> heavy)
    </xsl:template»

    <xslrtemplate match-'MASS" pr1or1ty-"l">
        <xslr value-of select-"."/>
    </xslrtemplate*

    <xsl:tempiate match-"RADIUS">
        <xsl:value-of select-"."/»
        <xsl:text» </xsl:text>
        <xsl:value-of select-"@UNITS"/>
    </xsl:template»

    <xsl template match-"OAY">
        <xsl:value-of select-"."/»
        <xsl;text» </xsl:text>
        <xsl:value-of $elect-"@UNITS"/»
    </xsl:template>

</xsl:stylesheet»
```

Процессор XSLT выбирает шаблон с наивысшим приоритетом, который добавляет текст "(<I>Very</I> heavy)" после каждого значения массы. В следующем примере использован шаблон с наивысшим приоритетом:

```
<HIML>

    <HEAD>
        <TITLE>
            The Planets Table
        </TITLE>
    </HEAD>

    <BODY>
        <H1>
            The Planets Table
        </H1»

        <TABLE BORDER-"2">
            <TR» '
```

```
            <TD>Name</TD>
            <TD>Mass</TD>
            <TD>Radius</TD>
            <TD>Day</TD>

        </TR>

        <TR>
            <TD>Mercury</TD>
            <TD>.0553(<I>Very</I> heavy)</TD>
            <TD>1516 miles</TD>
            <TD>58.65 days</TD>
        </TR>
        <TR>
            <TD>Venus</TD>
            <TD>.815(<I>Very</I> heavy)</TD>
            <TD>3716 miles</TD>
            <TD>116.75 days</TD>
        </TR>

        <TR>
            <TD>Earth</TD>
            <TD>1(<I>Very</I> heavy)</TD>
            <TD>2107 miles</TD>
            <TD>1 days</TD>
        </TR>

    </TABLE>
  </BODY>
</HTML>
```

УСОВЕРШЕНСТВОВАНИЯ В XSLT 2.0

Вопрос приоритета шаблонов должен быть учтен в XSLT 2.0. В частности, W3C рассматривает возможность добавления нового элемента с предварительным названием <xsl:next-match/>, который позволит выбирать для шаблона второй лучше всего подходящий элемент.

О приоритетах полезно знать еще одно: если двум шаблонам удовлетворяет один и тот же узел, и этим шаблонам не были присвоены приоритеты, процессор XSLT выберет шаблон с более узким правилом выбора. Например, условию "PLANET" будет отдано предпочтение перед обобщенным условием "*".

Элемент <xsl:copy>

Элемент <xsl:copy> позволяет скопировать узел из исходного дерева в выходное. Заметьте, однако, что это поверхностное (shallow) копирование, при котором не копируются потомки и атрибуты узла. У элемента есть один атрибут:

- use-attribute-sets. Задает названия наборов атрибутов, которые нужно применить к создаваемому элементу. Принимает значение списка QName, разделенных символами-разделителями. Этот атрибут можно использовать только в том случае, когда контекстный узел является элементом. Дополнительная информация о наборах атрибутов приведена в главе 6.

Этот элемент может содержать тело шаблона, которое используется только когда копируется корневой узел или элемент. Заметьте, что при применении к корневому узлу элемент <xsl:copy> Не задействован, поскольку узел выходного документа создается автоматически.

Приведенная в листинге 3.9 таблица стилей впервые появилась в главе 2; все, что она делает, — копирует все элементы из исходного документа в результируя> щий.

Листинг 3.9 Таблица стилей, копирующая элементы

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"l 0"
xm!ns:xsl-"http://vyww.w3.org/1999/XSL/Transform">
    <xsl.output method-"xml"/>

    <xsltemplate match-"*"»
        <xsl:COpy>
            <xsl:apply-tempiates/>
        </xsl:copy>
    </xsl:template»
</xsl:stylesheet»
```

Однако <xsl:copy> не копирует атрибуты — вот результат применения этой таблицы стилей к planets.xml:

```
<?xml version-"1.0" encoding-"UTF-8"?>
<PLANETS>

    <PLANET>
        <NAME»Mercury</NAME>
        <MASS».0553</MASS>
        <DAY>58.65</DAY>
        <RADIUS»1516</RADIUS»
        <DENSITY>.983</0ENSITY>
        <DISTANCE>43.4</DISTANCE>
    </PLANET»

    <PLANET»
        <NAME»Venus</NAME»
        <MASS».815</MASS>
        <DAY>116.75</DAY>
        <RADIUS»3716</RADIUS»
        <DENSITY>.943</DENSITY>
        <DISTANCE»66.8</DISTANCE»
    </PLANET»

    <PLANET>
        <NAME»Earth</NAME>
        <MASS»1</MASS>
        <DAY»1</DAY>
        <RADIUS»2107</RADIUS>
        <DENSITY»1</DENSITY>
        <DISTANCE»128.4</DISTANCE>
    </PLANET>
. . . ,>  I/,  A  M  .  i.  !<«»i{  .  i к  !
</PLANETS»                                                    .«lu-yqn  ,rx>
```

Копирование атрибутов несколько сложнее, потому что нужно найти какой-либо способ применить <xsl:copy> к каждому атрибуту элемента. Это можно сделать, например, при помощи элемента <xsl:for-each>, о котором пойдет речь в главе 5,

Листинг 3.10. Копирование атрибутов

```
<?xml veMbn--i.ID^          тщчьл г . лп          ,;гЧ          .мчытщП
<xsV:$ty1esheetWei^^"l:0'K< bMino>c )ii и;   гнул-л »       п;нюм    тчы я ':«о оти
xmlns: xsl -"http: //www. w3. org/ 1999/XSL/Transform">                    i ч; • «
    <xsl:output method-"xml"/>
                              WTHvMi— •   SJi'      4    <  r  • •  ':0ч  Ч с "tiNT IWf*
    <xsl:template match-"*"»
        <xsl:copy>
            <xsl:for-each select-"®*"»
                <xsl:copy/»
            </xsl:for-each»
            <xsl:apply-tempiates/»
        </xsl:copy»
    </xsl:tempiate»
</xsl:stylesheet»
```

А вот результат — заметьте, что на этот раз атрибуты не затронуты:

```
<?xml version="1.0" encoding-"UTF-8"?>
<PLANETS>

    <PLANET>
        <NAME»Mercury</NAME>
        <MASS UNITS-"(Earth - 1)"».0553</MASS>
        <DAY UNITS-"days"»58.65</DAY>
        <RADIUS UNITS-"miles"»1516</RADIUS>
        <DENSITY UNITS-"(Earth - 1)"».983</DENSITY>
        <DISTANCE UNITS-"million miles"»43.4</DISTANCE>
    </PLANET»

    <PLANET>
        <NAME»Venus</NAME>
        <MASS UNITS-"(Earth - 1)"».815</MASS>
        <DAY UNITS-"days"»116.75</DAY>
        <RADIUS UNITS-"miles"»3716</RADIUS>
        <DENSITY UNITS-"(Earth - 1)"».943</DENSITY>
        <DISTANCE UNITS-"million miles"»66.8</DISTANCE>
    </PLANET»

    <PLANET»
        <NAME»Earth</NAME>
        <MASS UNITS-"(Earth - 1)">1</MASS>
        <DAY UNITS-"days"»l</DAY>
        <RADIUS UNITS-"miles"»2107</RADIUS>
        «DENSITY UNITS-"(Earth - 1)"»1</DENSITY>
        <DISTANCE UNITS-"million miles"»128.4</DISTANCE>
    </PLANET»

</PLANETS>
```

Но есть более простой путь проверить, что копируются все дочерние узлы, атрибуты и другие потомки узлов: вместо элемента <xsl:copy> можно применить <xsl:copy-of>.

ГЛУБОКОЕ КОПИРОВАНИЕ

Пример использования <xsl:copy> для осуществления глубокого копирования документа будет приведен в главе 4, в которой описывается функция узла и объясняется, как рекурсивно вызывать один и тот же шаблон.

Элемент <xsl:copy-of>

Элемент <xsl:copy-of> позволяет осуществлять глубокое копирование узлов, при котором копируется не только узел, но и все его атрибуты и потомки. У этого элемента единственный атрибут:

- select (обязательный). Узел или набор копируемых узлов.

Этот элемент пуст и не имеет содержимого.

Ниже приведен пример работы этого элемента; в этом случае я заменил в листинге 3.10 элемент <xsl:for-each> элементом <xsl:copy-of>, который явно выбирает для копирования все атрибуты контекстного элемента.

Листинг 3.11. Применение <copy-of>

```
<?xml version-"1.0"?»
<xsl:stylesheet version-"!.O"
xmlns•xsl-"http://www.w3.org/1999/XSL/Transform">
    <xsl:output method-Hxml"/>
    <xslrtemplate match-"*"»
        <xsl:copy>
            <xsl:copy-of select-"®*"/»
            <xsl:apply-tempiates/>
        </xsl:copy>
    </xsl:template»
</xsl.styTesheet»
```

Этот код работает так же, как и предыдущий пример, копируя все элементы и атрибуты. С другой стороны, можно вообще обойтись без каких-либо изменений в листинге 3.10, — я могу просто использовать <xsl:copy-of> для того, чтобы скопировать весь документ, выбрав корневой узел и скопировав всех его потомков:

```
<?xml version-"IQ"?»
<xsl:stylesheet version~"I 0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"»
    <xsl:output method-"xml"/»
    <xslrtemplate match-"/"»
        <xslrcopy-of select-"*"/»
    </xslrtemplate»
</xsl:stylesheet»
```

При помощи <xsl:copy-of> можно также копировать конкретные узлы и их потомков вместо того, чтобы задавать подстановку *. Например, следующее правило копирует все элементы <MASS> и их потомков:

```
<xslrtemplate match-'MASS"»
        <xsl:copy-of select-"."/»
    </xsl template»
```

По этой причине я могу заменить элемент <MASS> на элемент <DAY>:

```
<xsl;template match-"MASS">
    <xsl:copy-of select-"DAY"/>
</xsl.-template»
```

Элемент <xsl: message>

При помощи элемента <xsl:message> можно дать указание процессору XSLT отобразить сообщение и, по выбору, прекратить обработку таблицы стилей. У элемента <xsl:message> один атрибут:

- terminate (необязательный). Значение «yes» прекращает обработку. По умолчанию установлено «по».

Куда на самом деле будет отправлено сообщение, зависит от процессора XSLT. Для процессоров, основанных на Java, сообщение обычно отправляется в выходной поток ошибок Java, которому соответствует экран компьютера, если процессор XSLT был вызван из командной строки. Другие процессоры XSLT могут выводить сообщения во всплывающие окна или в web-страницы, отправляемые в браузеры.

В приведенном ниже листинге 3.12 я прекращаю обработку XSLT, когда процессор XSLT пытается преобразовать элемент <DAY> в planets.xml, выводя сообщение "Sorry, DAY information is classified." (Извините, информация о параметре «ДЕНЬ» засекречена.).

Листинг 3.12. Применение <xsl:message>

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"I 0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"»

    <xsl:template match-"/PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <BODY>
                <Hl>
                    The Planets Table
                </Hl>
                <TABLE BORDER-"2">
                    <TD>Name</TD>
                    <TD>Mass</TD>
                    <TD>Radius</TD>
                    <TD>Day</TD>
                    <xsl:apply-templates/>
                </TABLE>
            </BODY>
        </HTML>
    </xsl:template»

    cxsl:template match-"PLANET">
```

продолжение &

Листинг 3.12 (продолжение)

```
        <TR>
          <TDxxsl value-of select-"NAME'7»</TD>
          cTDxxsl :anplv-tpmplatp<; <plpt«="MAS;"/></In>
          <TD><xsl:apply-templates select="RADIUS"/></TD>
          <TD><xsl:apply-templates select="DAY"/></TD>
        </TR>
    </xsl:template>

    <xsl:template match="MASS">
        <xsl:value-of select=". "/>
        <xsl:text> </xsl:text>
        <xsl:value-of select="@UNITS'7>
    </xsl:template»

    <xsl.template match-"RADIUS">
        <xsl:value-of select-"."/>
        <xsl:text> </xsl:text>
        <xsl:value-of select-"@UNITS"/»
    </xsl:template»

    <xslrtemplate match-"DAY">
        <xslrmessage terminate-"yes">
            Sorry. DAY information is classified.
        </xsl:message terminate-"yes">
    </xslrtemplate»

</xsl:stylesheet»
```

Вот результаты применения этой таблицы стилей в Xalan:

```
C \planets»java org.apache.xalan xslt.Process -IN planets.xml
-XSL message.xsl -OUT planets.html
file:///C:/XSL/messages/message.xsl; Line 49: Column 38;
Sorry. DAY Information is classified.
XSLT Error (javax.xml.transform.TransformerException):
Stylesheet directed termination
```

При помощи элемента <xsl:message> можно выводить информацию о том, что происходит при обработке таблицы стилей, что может пригодиться не только для вывода ошибок и предупреждений, но и для отладки таблиц стилей.

Далее в книге будут рассмотрены и другие посвященные шаблонам темы — такие, как вызов именованных шаблонов и использование параметров. В следующей главе мы перейдем к большой и важной теме — созданию образцов выбора, При помощи которых в XSLT можно выбирать, какке узел или узлы вам нужны для работы. Мы уже затрагивали тему работы с образцами выбора, теперь же пора перейти к их систематическому рассмотрению.

Создание образцов выбора



До сих пор материал книги был достаточно очевиден, за исключением одного: несколько загадочных образцов выбора (match pattern). Мы работали с различными образцами выбора, такими как «/PLANETS» в элементах <xsl: template», не предлагая систематического объяснения того, как в действительности работают эти образцы, — как в этом случае:

```
<xslrtemplate match-"/PLANETS">
    <HTML>
        <HEAD>
            <TITLE>
                The Planets Table
            </TITLE>
        </HEAD>
        <BODY>


        </BODY>
    </HTML>
</xsl:template»
```

В этой главе мы рассмотрим все необходимое, что нужно знать для создания образцов выбора в XSLT. Образцы выбора применяются в элементах <xsl:template>, <xsl:key> и <xsl:number>; с элементом <xsl:template> мы работали с самого начала книги, элемент <xsl:key> мы увидим в главе 9 и элемент <xsl:number> — в главе 5. В частности, для выбора по образцу у элементов <xsl:template> и <xsl:key> используется атрибут match, а у элемента <xsl:number> — атрибуты count и from.

Образцы выбора можно также применять в атрибуте select таких элементов, как <xsl:apply-templates>, <xsl:value-of>, <xsl:for-each> и <xsl:copy-of>. Здесь важно отметить один момент: атрибут select этих элементов обладает большими возможностями, чем атрибуты match, count и from элементов <xsl:template>, <xsl:key>

и <xsl:number>, потому что в select можно использовать полные выражения XPath, а не только образцы выбора.

Образцы выбора являются подмножеством выражений XPath, то есть все образцы выбора являются допустимыми выражениями XPath, но не все выражения XPath являются образцами выбора. Единственные выражения XPath, которые могут быть образцами, — это выражения, возвращающие набор узлов (даже набор, состоящий из одного узла) и использующие пути, которые задают только дочерние узлы или узлы атрибутов.

Образцы выбора определены в самой рекомендации XSLT, в то время как выражения XPath определены в рекомендации XPath (www.w3.org/TR/xpath); тем не менее, эти рекомендации совместимы, потому что все образцы выбора являются одновременно выражениями XPath.

СОЗДАНИЕ ПОЛНЫХ ВЫРАЖЕНИЙ XPATH

В главе 7 «Работа и изучение Xpath» показано, как создавать полные выражения XPath. Полные выражения XPath можно применять в XSLT в следующих местах: в атрибуте select элементов <xsl:apply-templates>, <xsl:value-of>, <xsl:for-each>, <xsl:param>, <xsl:variable>, <xsl:wit-param>, <xsl:copy-of> и <xsl:sort>; в значениях шаблонов атрибутов; в атрибуте test элементов <xsl:if> и <xsl:when>; в значении атрибута элемента <xsl:number> и в предикатах образцов выбора.

Чтобы еще больше все усложнить, следует сказать, что выражения XPath можно использовать в специальной, необязательной части (и только в этой части) образцов выбора: в *предикатах*. Как мы увидим в этой главе, предикаты — это выражения XPath, которые вычисляются либо в значения «истина/ложь», либо в числа, заключаемые в квадратные скобки, [и]. Например, образец PLANET[NAME="Venus"] выбирает дочерние узлы <PLANET> контекстного узла, у которых есть дочерние узлы <NAME> с текстом «Venus». Выражения внутри [и] представляют собой настоящие выражения XPath с известными ограничениями, которые будут рассмотрены в этой главе.

Безусловно, для создания образцов выбора необходим опыт, поэтому в данной главе приводится много примеров.

MICROSOFT И НЕСТАНДАРТНЫЕ ОБРАЗЦЫ ВЫБОРА

Microsoft поддерживает образцы выбора в своем процессоре XML MSXML3, но есть еще кое-что, о чем вам следует знать: с образцами выбора Microsoft использует также весьма много нестандартного, не используемого W3C синтаксиса. В этой главе я собираюсь придерживаться официальной, W3C, версии, и если вам доведется читать документацию Microsoft об образцах выбора, имейте в виду, что многое из этой документации относится только к реализации Microsoft.

Выбор корневого узла

Как вы уже видели, выбрать корневой узел можно при помощи образца выбора «/», как, например:

```
<xsl:template match="/">
    <HTML>
        <xsl:apply-templates/>
    </HTML>
</xsl:template>
```

Выбор элементов

Можно выбирать элементы, просто задавая их имена, как мы уже видели. Следующий шаблон выбирает элементы <PLANETS>:

```
<xslrtemplate match-'PLANETS"»
  <HTML>
    <xsl:apply-templates/>
  </HTML>
</xsl:template»
```

Выбор дочерних элементов

При доступе к дочернему узлу определенного узла для разделения имен элементов можно использовать операцию шага /. Пусть, например, требуется создать правило, которое должно применяться только к тем элементам <NAME>, которые являются дочерними для элементов <PLANET>. Для этого можно задать выражение "PLANET/NAME". Вот правило, окружающее текст таких элементов в элемент HTML <H3>:

```
<xsl:tempi ate match-"PLANET/NAME">
    <H3>
        <xsl:value-of select-" "/>
    </H3>
</xsl:template»
```

Можно также использовать символ * в качестве символа-подстановки, что соответствует любому элементу. (* может выбирать только элементы, однако образец @* выбирает любой атрибут.) Например, следующее правило применяется ко всем элементам <NAME>, которые являются внуками элементов <PLANET>:

```
<xsl:tempi ate match-"PLANET/*/NAME">
    <H3>
        <xsl:value-of select-"."/»
    </H3>
</xslrtemplate»
```

Выбор потомков элемента

В предыдущем разделе **При** помощи выражения "PLANET/NAME" я выбирал все элементы <NAME>, являющиеся прямыми потомками элементов <PLANET>, а при помощи выражения "PLANET/*/NAME" — все элементы <NAME>, являющиеся внуками элементов <PLANET>. Есть, однако, более простой способ выполнить оба выбора — применить выражение "PLANET//NAME", выбирающее все элементы <NAME>, находящиеся внутри элементов <PLANET>, независимо от глубины вложенности (соответствующие элементы называются потомками элемента <PLANET>). Иными словами, "PLANET//NAME" выбирает "PLANET/NAME", "PLANET/7NAME", "PLANET/*/*/NAME" **и** т. д.:

```
<xslrtemplate match-"PLANETS//NAME">
    <H3>
        <xsl:value-of select-"."/>
    </H3>
</xsl:template»
```

Выбор атрибутов

Как было показано в главе 3, «Создание и применение шаблонов», можно выбирать атрибуты, если предварять их имена префиксом _ Вы уже работали с атрибутом UNITS, который поддерживают большинство детей элементов <PLANET>:

```
<PLANET>
        <NAME>Earth</NAME>
        <MASS UNITS-"(Earth - 1)">1</MASS>
        <DAY UNITS""days">1</DAY>
        <RADIUS UNITS-"miles">2107</RADIUS>
        «DENSITY UNITS-"(Earth - 1)">1</DENSITY>
        «DISTANCE UNITS-"m1llion miles"»128.4</DISTANCE»<!--B перигелии-->
</PLANET>
```

Чтобы извлечь единицы измерения и отобразить их вместе со значениями массы и т. п., можно выбрать атрибут UNITS при помощи @UNITS (листинг 4.1).

Листинг 4.1. Выбор атрибутов

```
<?xml version-"l 0"?>
<xsl:stylesheet versi on-"1.0"
xml ns:xsl-"http://www.w3.org/1999/XSL/Transform">

    <xsl:tempi ate match-"/PLANETS"»
        <HTML>
            <HEAD»
            .
            .
            .
            </HEAD>

            <BODY>
            .
            .
            .
            </BODY>
        </HTML>
    </xsl:tempi ate»

    <xsl:template match-"PLANET"»
        <TR>
            <TD»<xsl:value-of select-"NAME"/x/TD>
            <TD»<xsl;apply-tempiates select-"MASS"/»</TD>
            <TD»<xsl:apply-tempiates select-"RADIUS"/></TD>
        </TR>
    </xsl:tempi ate»

    <xsl:tempiate match-"MASS"»
        <xsl:value-of select-"."/»
        <xsl:text» <1 xsl:text»
```

```
        <xs1:value-of select-"GUNITS"/»
    </xsl:template»

    <xsl:tempiate match-"RADIUS">
        <xsl:value-of select-" "/>
        <xsl:text> </xsl text»
        <xsl:value-of select-"@UNITS"/>
    </xsl:template»

    <xslrtemplate match-"DAY"»
tii     <xsl;value-of select-" "/>
        <xslitext» </xsl:text»
        <xsl:value-of select-"@UNITS"/»
    ^/xsl template»

</xsl:stylesheet»
```

Теперь результирующая HTML-таблица включает не только значения, но и их единицы измерения:

```
<HTML>
    <HEAD>
        <TITLE>
            The Planets Table
        </TITLE>
    </HEAD>

    <BODY> -
        <H1>
            The Planets Table
        </H1>

        <TABLE>
            <TR>
                .
                .
                .
            <TR>
                <TD»Mercury</TD»
                <TD».0553 (Earth - 1)</TD»
                <TD»1516 miles</TD»
            </TR>

            <TR»
                <TD»Venus</TD>
                <TD».815 (Earth - 1)</TD»
                <TD»3716 miles</TD>
            </TR>
                .
                .
                .
        </TABLE>
    </BODY>

</HTML>
```

Для выбора всех атрибутов элемента можно также использовать подстановку Например, "PLANET/@*" выбирает все атрибуты элементов <PLANET>.

Формальное определение образцов выбора

Определение образцов выбора приводится также в рекомендации XSLT W3C. Образцы выбора определяются в терминах выражений XPath следующим образом:

«Синтаксис для образцов является подмножеством для выражений [XPath]. В частности, пути расположения, удовлетворяющие определенным ограничениям, могут использоваться как образцы. Выражение, в то же время являющееся образцом, всегда вычисляется в объект типа набора узлов. Узел удовлетворяет образцу, если узел является членом результата вычисления образца как выражения по отношению к возможному контексту; возможный контекст — это контекст, контекстный узел которого был выбран, или один из его предков».

Самое важное предложение в предыдущем абзаце — последнее. Суть в том, что узел X удовлетворяет образцу тогда и только тогда, когда существует узел X или предок X, такой, что при применении к этому узлу образца как выражения XPath, результирующий набор узлов будет включать X.

Что в действительности это означает? Это значит, что когда нужно проверить, удовлетворяет ли узел образцу, сначала следует применить образец как выражение XPath к самому узлу, затем применить его последовательно ко всем его предкам, вплоть до корневого узла. Если какой-либо полученный при этом набор узлов будет содержать сам узел, узел удовлетворяет образцу. Такой порядок действий имеет смысл потому, что образцы выбора пишутся для применения к текущему узлу или его дочерним узлам.

СЛЕДСТВИЯ ФОРМАЛЬНОГО ОПРЕДЕЛЕНИЯ ОБРАЗЦОВ ВЫБОРА

Приведенное определение образцов в терминах выражений XPath довольно очевидно, но существуют следствия, которые сразу не видны. Например, хотя функция node() определена как функция, выбирающая любой узел, при использовании ее в качестве образца, "nodeQ", в действительности она представляется как "child: :node()", как вы увидите позже в этой главе. Помимо прочего, это означает, что образец "node(>" может выбирать только дочерние узлы — он никогда не выберет корневой узел. Отметьте также, что нет образцов, которые бы могли выбрать узлы объявлений пространств имен.

W3C дает формальное определение образцов выбора в нотации расширенных форм Бэкуса-Наура (РБНФ), при помощи которой написана и спецификация XML. Объяснение этой грамматики можно найти по адресу www.w3.org/TR/REC-xml (раздел 6). Здесь я привожу формальное определение образцов только для справки. (Разъяснению этого формального определения посвящена целая глава.) В следующем списке приведены используемые здесь лексемы нотации РБНФ:

- ::- означает «определяется как»;
- + означает «один или больше»;
- * означает «ноль или больше»;
- | означает «или»;
- - означает «не»;
- ? означает «необязательно»,

Далее приведено настоящее, формальное определение образцов выбора W3C; когда элемент заключен в одиночные кавычки, как 'child' или ' ', это значит, что элемент должен появиться в образце буквально (как "child::NAME"), — такие элементы называются литералами, Literal:

```
Pattern  ::-LocationPathPattern | Pattern 'LoqationPathPattern

LocationPathPattern ::-  '/' RelatlvePathPattern?
    I  IdKeyPattern (V  |  '//') RelatlvePathPattern)?
    j  '//'? RelatlvePathPattern

IdKeyPattern ::-  'id'  'C Literal  ')'   |  'key'  '('  Literal      Literal  ')'

RelativePathPattern ::-  StepPattern  |  RelativePathPattern  '/'  StepPattern
     |  RelativePathPattern  '//'       StepPattern

StepPattern  ::-  ChildOrAttributeAxisSpecifier  NodeTest  Predicate*

ChiIdOrAttributeAxisSpecitier         AbbreviatedAxisSpecifier
     |  ('child'  |  'attribute')  '::'
```

Определения *NodeText* (текстового узла) и *Predicate* (предиката) приводятся в спецификации XPath (Expr соответствует выражению XPath, а NCName и QName были определены в начале главы 2, «Создание и применение таблиц стилей»):

```
NodeTest  ::-NameTest  |  NodeType  'C  ')'  |  'processing-instruction'  'C Literal  ')'

Predicate  ::-  '['  PredicateExpr  ']'

PredicateExpr  ::-  Expr

AbbreviatedAxisSpecifier        '@'?

NameTest  :: =  '*'  |  NCName  ':'  '*'  |  QName

NodeType  :'comment'  |  'text'  |  'processing-instruction'  |  'node'
```

Как вы можете видеть, все это больше походит на какой-то код. Давайте начнем его расшифровывать. Во-первых, *образец* (pattern) состоит из одного (или более) *образца пути расположения* (location path pattern). Образец пути расположения, в свою очередь, состоит из одного или нескольких *образцов шага* (step pattern), разделенных / или //, или одним (несколькими) образцом шага в объединении с функциями id и key (выбирающими элементы с определенными идентификаторами или ключами).

Образцы шага являются строительными блоками шаблонов: в одном пути можно использовать несколько шагов, разделяя их символами / или //, как в образце "PLANET/*/ NAME", в котором три шага: "PLANET", "*" и "NAME". Если вы начнете сам образец с символа /, он будет называться *абсолютным,* так как вы указали образец от корневого узла (как в "/PLANETS/PLANET" или "//PLANET"); иначе образец называется *относительным* и применяется начиная с контекстного узла (как в "PLANET").

Затем образец шага состоит из оси, условия узла и предикатов (которых может и не быть). Например, в выражении child::PLANET[position() « 5], child — это

имя оси, PLANET — условие узла, а [position() = 5] — это предикат. (Предикаты всегда заключены в квадратные скобки.) Образцы можно создавать при помощи одного или более образцов шага, как, например, образец /child::PLANET/child::NAME, который выбирает элементы <NAME>, дочерние по отношению к родителю <PLANET>.

Таким образом, чтобы понять работу образцов, вам необходимо понять работу образцов шага, поскольку образцы состоят из одного или более образцов шага, в таких выражениях, как "step-patternl/step-pattern2/step-pattern3... ". А чтобы понять работу образца шага, необходимо понять работу деятельности трех составных частей — осей, условий узлов и предикатов, которыми мы и займемся в следующих разделах.

Образцы шага, часть 1: оси образца

Оси — первая часть образцов шага. Например, в образце шага child::NAME, ссылающемся на элемент <NAME>, дочерний по отношению к контекстному узлу, child называется осью. У образцов две оси:

- ось attribute содержит атрибуты контекстного узла;
- ось child содержит детей контекстного узла. Если ось явно не задана, ось child будет осью по умолчанию.

При помощи осей можно задать шаг расположения (location path) или путь, как в следующем примере, в котором ось child используется для задания выбора дочерних узлов контекстного узла, элемента <PLANET>:

```
<xsl:tempi ate match«"PLANET">
    <HTML>
        <CENTER>
            <xs1:value-of  select-'child::NAME"/>
        </CENTER>
        <CENTER>
            <xsl:value-of  select-"child::MASS"/>
        </CENTER>
        <CENTER>
            <xsl :value-of sel ect-*" child: :DAY"/>
        </CENTER>
    </HTML>
</xsl:template»
```

Рассмотрим ряд примеров применения осей:

- child::PLANET. Возвращает дочерние элементы <PLANET> контекстного узла;
- child::*. Возвращает все дочерние элементы контекстного узла (* выбирает только элементы);
- attribute::UNIT. Возвращает атрибут UNITS контекстного узла;
- child::*/child::PLANET. Возвращает всех внуков <PLANET> контекстного узла.

Хотя, судя по этим примерам, кажется, что можно применять только оси детей и атрибутов, на практике это не совсем так. Когда требуется указать детей, возможности оси child несколько ограничены, потому что необходимо указывать каждый уровень, который необходимо выбрать — например "child::PLANETS/child::PLANET/

child::MASS" выбирает элемент <MASS>, дочерний по отношению к элементу <PLANET>, который, в свою очередь, дочерний по отношению к <PLANETS>. Если вам требуется выбрать все элементы <MASS>, появляющиеся в любом месте элемента <PLANETS>, детей, внуков, правнуков и ъ д< кажется, что нет способа сделать это в одном образце. В XPath это можно сделать при помощи выражения наподобие [4]cbUd::PtANETS/descendant::MASS", но в образцах нельзя использовать ось потомков (descendant). Помните, однако^ что в этих же целях можно применить опера* цию //; Например, образец "child::PLANETS//child::MASS" выбирает все элементы <MASS> в любом месте внутри элемента <PLANETS>.

Следующий пример (листинг 4.2) демонстрирует работу этого[1] образца, заменяя текст во всех элементах <MASS> независимо от того, где они находятся внутри элемента <PLANETS>, на текст "Very heavy!". Для того чтобы скопировать в результирующий XML-документ все остальные узлы planets.xml, я также установил правило, выбирающее любой узел при помощи условия узла (node test) node, с которым мы познакомимся позже. Заметьте, что, хотя образец, выбирающий любой узел, также выбирает все элементы <MASS>, образец "chi'ld::PLANETS//chi4d::MASS" гораздо более специален — поэтому, как объяснялось в главе 3, процессор XSLT задаст ему более высокий приоритет для элементов <MASS>.

Листинг 4.2. Выбор элементов <MASS>

```
<?xml version-"!. 0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Trans form">
<xsl-.output method-"xmrv>

    <xsl:tempiate match-"©*|node()">
        <xsl:copy>
          <xsl:apply-templates select-"@*|node()"/>
        </xsl:Copy>
    </xsl:tempiate>

    <xsl:tempiate match-"ch1ld::PLANETS//child::MASS">
        <MASS>
            Very heavy!
        </MASS>
    </xsl:tempiate>

</xsl:stylesheet>
```

А вот результирующий XML-документ:

```
<?xml version="1.0" encoding-"UTF-8"?>
<?xml-stylesheet type="text/xml" href-"planets.xsl"?>
<PLANETS>

    <PLANET>
        <NAME>Mercury</NAME>
        <MASS>
            Very heavy!
        </MASS>
    1 <DAY UNITS-h'days">58.65</0AY>
      «RADIUS UNITS~e'm1les"M5l6</RADIUS*                    -j
!И\     DENSITY UNIJ^-^a/t^ na^^^^/D^SIjTY^  „„          >iuv
```

```
    <DISTANCE UNITS='million miles">43.4</DISTANCE><!--В перигелии->
</PLANET>

<PLANET>
    <NAME>Venus</NAME>
    <MASS>
        Very heavy!
    </MASS>
    <DAY UNITS^"days">116.75</DAY>
    <RADIUS UNITS-"miles">3716</RADIUS>
    <DENSITY UNITS-"(Earth - 1)">.943</DENSITY>                    »
    <DISTANCE UNITS-"million miles">66.8</DISTANCE><!--В перигелии-->
</PLANET>

<PLANET>
    <NAME>Earth</NAME>
    <MASS>
        Very heavy!
    </MASS>
    <DAY UNITS-"days">l</DAY>
    <RADIUS UNITS-"miles">2107</RADIUS>
    <DENSITY UNITS-"(Earth - 1)">1</DENSITY>
    <DISTANCE UNITS-"million miles">128.4</DISTANCE><!--В перигелии-->
</PLANET>

</PLANETS>
```

При задании осей в образцах можно воспользоваться рядом сокращений, применяемых практически повсеместно.

Сокращенный синтаксис

Для образцов существует два правила сокращения осей:

- child::childname может быть сокращено как childname;
- attribute::childname может быть сокращено как @childname.

В следующем списке перечислен ряд примеров образцов с сокращенным синтаксисом; в конце главы вы увидите много других.

- PLANET. Выбирает дочерние элементы <PLANET> контекстного узла;
- *. Выбирает все дочерние элементы контекстного узла;
- @UNITS. Выбирает атрибут UNITS узла;
- @* Выбирает все атрибуты контекстного узла;
- */PLANET. Выбирает всех внуков <PLANET> контекстного узла;
- //PLANET. Выбирает всех потомков <PLANET> корня документа;
- PLANETS/PLANET. Выбирает все элементы <PLANET>, являющиеся потомками дочерних элементов <PLANETS> контекстного узла;
- //PLANET/NAME. Выбирает все элементы <NAME>, дочерние по отношению к <PLANET>;
- PLANET[NAME]. Выбирает детей <PLANET> контекстного узла, у которых есть дочерние элементы <NAME>.

В таком образце, как "child::PLANET", "child" является осью, а "PLANET" — условием узла, что представляет собой вторую часть образцов шага.

Образцы шага, часть 2: условия узла

Условия узла (node test) составляют вторую часть образцов шага. В качестве условий узла можно использовать названия узлов или символ подстановки * для выбора и узлов, и их типов. Например, выражение child::*/child::NAME выбирает все элементы <NAME>, являющиеся правнуками контекстного узла.

Помимо названий узлов и символа подстановки, можно применять также следующие условия узлов:

- comment() выбирает узлы комментария;
- node() выбирает узел любого типа;
- processing-instruction() выбирает узел инструкции обработки. В скобках можно указать название выбираемой инструкции обработки;
- textQ выбирает текстовый узел.

В следующих разделах мы изучим эти условия узлов и рассмотрим примеры их применения.

Выбор комментариев

Текст комментариев можно выбрать при помощи образца comment(). Разумеется, не следует хранить данные, которые попадут в выходной документ, в комментариях входного документа. Тем не менее, вам может потребоваться преобразовать комментарии из формы <!-comment~> в какую-то другую форму, используемую другим языком разметки, — например, элемент <COMMENT>.

В следующем примере я извлеку комментарии из planet.xml и включу их в полученные выходные данные.

```
<PLANET>
    <NAME>Venus</NAME>
    <MASS UNITS»"(Earth - 1)">.815</MASS>
    <DAY UNITS-"days">116.75</DAY>
    «RADIUS UNITS""mi 1es">3716</RADIUS>
    «DENSITY UNITS-"(Earth - 1)">.943</DENSITY>
    «DISTANCE UNITS-"million m1les">66.8</DISTANCE><!—B перигелии-•>
</PLANET>
```

Чтобы извлечь комментарии и поместить их в элементы <COMMENT>, я включил правило только для комментариев (листинг 4.3).

Листинг 4.3. Выбор комментариев

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

    <xsltemplate match-"PLANETS">
```

продолжение &

Листинг 4.3 (продолжение)

```
            <HTML>
                <xsl:apply-templates/>
            </HTML>
        </xsl:tempiate>

    <xsl:template match-"comroent()">
        <C0MMENT>
            <xsl:value-of  select»"."/>
        </C0MMENT>
    </xsl:template»

    </xsl:stylesheet»
```

Вот результат для Венеры, в котором комментарий преобразован в элемент <COMMENT>:

```
Venus
.815
116.75
3716
.943
668<COMMENT»В перигелии</COMMENT»
```

Обратите внимание: здесь текст для других элементов в элементе <PLANET> также включается в выходной документ, потому что так установлено в соответствии с правилом по умолчанию для каждого элемента. Поскольку для элементов я не предоставил какого-либо правила, их текст просто включается в выходной документ.

Выбор узлов при помощи node()

В образце условие узла node выбирает любой узел, за исключением корневого узла — помните, что в действительности это child::node(). Предположим, мы хотим создать таблицу стилей, копирующую произвольный документ XML, используя <xsl:copy>. (В главе 3 для этого применялся элемент <xsl:copy-of>.) Можно начать так, как показано в следующем примере. В этом случае в применяемом шаблоне для выбора любого элемента или любого атрибута используется операция OR, с которой мы познакомимся позже в этой главе (этот шаблон фактически выбирает себя — для того чтобы продолжать копирование на много уровней вглубь):

```
<?xml  version-"1.0"?>
<xsl:stylesheet  version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"»
<xsl:output  method-"xmr/>

    <xslrtemplate match-"@*|*">
        <xsl:copy»
            <xsl:apply-templates  select-"P*|*"/»
        </xsl:copy»
    </xsl:tempiate»

</xsl:stylesheet»
```

Однако посмотрите на результат — обратите внимание на то, что в этой версии, выбирающей только элементы и атрибуты (@*|*), не копируются узлы-разделители и текстовые узлы:

```
<?xml version-"1.0" encoding="UTF-8"?>
<PLANETS><PLANET><NAME/><MASS UNITS-"(Earth - 1)"/><DAY UNITS-"days"/><RADIUS
UNITS-"mi 1 es"/><DENSITY UNITS-"(Earth = 1)"/><DISTANCE UNfTS^milHon nriles'7^
</PLANET><PLANETxNAME/xMASS UNITS-"(Earth - 1)"/><DAY UNITS-"days"/><RADFTL5
UNITS-"mi1es"/xDENSITY UNITS-"(Earth - 1)"/XDISTANCE UNITS-"million miles"/»
</PLANET><PLANET><NAME/xMASS UNITS-"(Earth - 1)"/XDAY UNITS-"days"/xRADIUS
UN ITS- "mi 1 es" /xDENSITY UNITS-"(Earth - 1)"/XDISTANCE UNITS="million miles'7>
</PLANETX/PLANETS>
```

Это, конечно, неполно. Если я, с другой стороны, буду выбирать по образцу "@*|node()" вместо "@*|*", новое правило шаблона выберет все узлы за исключением корневого узла (который создается в результирующем дереве автоматически), поэтому символы-разделители будут скопированы, так же как и текст (листинг 4.4).

Листинг 4.4. Копирующая таблица стилей

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">
<xsl.output method-"xml"/>
  <xsl:tempiate match-"©*|node()">
    <xsl:copy>
      <xsl:apply-templates select-"©*|node()"/>
    </xsl:copy>
  </xsl:template»

</xsl:stylesheet»
```

Новый результат:

```
<?xml vers1on-"L0" encpding«"UTF-8"?>
<?xml-stylesheet type=r "text/xml" href-"planets.xsl"?>
<PLANETS>

    <PLANET? • •
        <NAME>Mexgury</NAME>
        <MASS UNITS-"(Earth - 1)"».0553</MASS>
' 5 '   <DAY UNITS-"days">58.65</DAY>
  .ф  <RADIUS UNITS-"miles">1516</RADIUS>
irjMM'i-  «DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
        <DISTANCE UNITS-"million miles">43.4</DISTANCE><!--B перигелии--»
    </PLANET>

    <PLANET>
        <NAME>Venus</NAME>
        <MASS UNITS-"(Earth - 1)">.815</MASS»
        <DAY UNITS-"days">116.75</DAY>
        <RADIUS UNITS-"miles">3716</RADIUS>
        <DENSITY UNITS-"(Earth - 1)">.943</0ENSITY>
        «DISTANCE UNITS-"million miles">66.8</DISTANCE><!--B перигелии'-.-»
    </PLANET>
```

Выбор текстовых узлов при помощи text()

Выбрать текст узла можно при помощи образца "text()[n]". Как правило, нет особых причин применять условие узла text. В XSLT существует правило по умолчанию, в соответствии с которым текст текстового узла включается в выходной документ, если этот узел не выбирается какими-либо другими правилами. Если нужно сделать это правило по умолчанию явным, можно поступить, например, так:

```
<xsl:tempiate match="text()">
    <xsl:value-of select»"."/>
</xsl:template»
```

Можно перекрыть это правило, не отправляя текст из текстовых узлов в выходной документ, — в том числе так:

```
<xsl:tempiate match»"text()">
</xsl:template»
```

Потребность в применении условия текстового узла возникает, например, когда нужно выбрать узлы с определенным текстом. Предикат "NAME[text() - 'Venus']" выбирает элементы <NAME>, в которых содержится имя "Venus". (Будьте внимательны с вложением кавычек, чтобы процессор XSLT не ошибся, — например, такой предикат не работает: "NAME[text() - "Venus"]".) Еще одна причина для использования условия текстового узла появляется, когда требуется применить к текстовым узлам некоторое условие при помощи строковых функций XPath (которые будут рассмотрены позже в этой главе). Например, текстовый узел "Earth" в <NAME>Earth</NAME> выбирается образцом "text()tstarts-with(.,'E')]".

КАК УБРАТЬ КОММЕНТАРИИ

Ранее мы видели, что образец "@*|node()" (в котором используется операция OR, обсуждаемы позже) выбирает из файла planets.xml все, включая комментарии. Если вы хотите убрать комментарии, воспользуйтесь образцом "@*|*|text()", который сохраняет только элементы, атрибуты и текстовые узлы.

Выбор инструкций обработки

Для выбора инструкций обработки используйте образец processing-instruction():

```
<xslrtemplate match-"/processing-1nstructi on()">
    <I>
        Found a processing instruction.
    </I>
</xsl:tempi ate»
```

Можно также указать, какую именно инструкцию обработки вы хотите выбрать, задав имя инструкции (исключая <? и ?>), — как в следующем примере, в котором выбирается инструкция обработки <?xmt-include?>:

```
<xsl template match="/processing-instruction(xml-include)"»
    <I>
        Found an xml-include processing instruction.
    </I>
</xslrtemplate»
```

РАЛИЧИЕ МЕЖДУ КОРНЕВЫМИ УЗЛАМИ И КОРНЕВЫМИ ЭЛЕМЕНТАМИ

Одна из основных причин различия корневого узла в самом начале документа и Корневого элемента заключается в том, что при этом у вас есть доступ к инструкциям обработки и другим узлам в прологе документа.

На этом мы заканчиваем обсуждение возможных условий узлов в образцах шага. Третья и последняя часть образцов шага — предикаты.

Образцы шага, часть 3: предикаты

Предикаты, третья часть образцов шага, содержат выражения XPath. Предикат можно заключить в операцию [] и проверить, верно ли заданное условие.

Например, можно проверить:

- значение атрибута в заданной строке;
- значение элемента;
- содержит ли элемент определенного ребенка, атрибут или другой элемент;
- позицию узла в дереве узлов.

Подробнее выражения XPath обсуждаются в главе 7, но здесь будет представлено введение в тему, так как эти выражения можно применять в предикатах образцов.

Выражения XPath более сложны, чем образцы выбора. Если при их создании у вас возникнут затруднения, вам может помочь удобная программа-пример ApplyXPath.java из пакета Xalan, при помощи которой можно применить к документу выражение XPath и посмотреть на результаты Например, если применить выражение XPath "PLANET/NAME" к planets.xml, будут отображены значения всех элементов <NAME>, дочерних по отношению к элементам <PLANET> (открывающий и закрывающий теги <output> добавляются программой ApplyXPath):

```
C:\>java ApplyXPath planets.xml PLANET/NAME
<output>
<NAME>Mercury</NAME>
<NAME>Venus</NAME>
<NAME>Earth</NAME>
</output>
```

Если предикат имеет числовое значение, последнее представляет *условие* позиции (position test). Например, NAME[1] выбирает первого ребенка <NAME> контекстного узла. Условия позиции W3C, а также условия позиции в Xalan, Oracle, XT, Saxon и MSXML3 (XML процессор от Microsoft, подразумевающий использование JavaScript, с которым вы встречались в главе 1 и еще встретитесь в главе 10, «Работа с API процессоров XSLT») основаны на 1, поэтому первый ребенок — это ребенок 1. Условия позиции в XML-документах, которые используют таблицы стилей XSL и загружаются в текущую версию Internet Explorer (версию 5.5, на смену которой приходит 6.0), основаны на 0 (и в предикатах можно использовать только очень сокращенную форму выражений XPath) — и, следовательно, так же считается в большей части документации по XSL на web-узле Microsoft. В другом

случае значением предиката должна быть ложь или истина, что называется *логическим условием* (Boolean test). Например, предикат [@UNITS~ "million miles"] выбирает элементы, у которых имеются атрибуты UNITS со значением "million miles".

Предикаты являются полными выражениями XPath, хотя на предикаты, используемые в образцах, накладывается два ограничения:

- когда образец используется в атрибуте match, предикат не должен содержать никаких ссылок на переменные XSL (которые обсуждаются в главе 9). Это ограничение не применяется к предикатам, используемым в элементах <xsl:number>;
- образцы не могут использовать в предикатах функцию XPath current, возвращающую текущий узел. Ее применение ограничено, поэтому обработка зависит от реализации и не зависит от текущего состояния обработки.

В следующем примере обазец выбирает элементы <PLANET> с дочерними элементами <;NAME>:

```
<xsl:tempiste match - "PLANET[NAME]">
    .
    .
    .
</xsl template»
```

Этот образец выбирает любой элемент с дочерним элементом <NAME>:

```
«xsl:template match - "*[NAME]">
    .
    .
    .
</xsl:template»
```

Теперь я задал элементам <PLANET> в planets.xml новый атрибут, COLOR, устанавливающий цвет планеты:

```
<?xml version-"1.0"?>
<?xml-stylesheet type-"text/xml" href-"planets.xsl"?»
<PLANETS>

  <PLANET COLOR-"RED"»
    <NAME»Mercu ry</NAME»
    «MASS UNITS-"(Earth - 1)"».0553</MA5S»
    «DAY UNITS-"days">58,65</QAY>
    «RADIUS UNITS-"miles">1516</RADIUS>
    <DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
    «DISTANCE UNITS-"million miles">43 4</DISTANCE><!-B перигелии-->
  </PLANET>

  <PLANET COLOR-"WHITE">
    <NAME>Venus</NAME>
    <MASS UNITS-" (Earth - 1) ">. 815</MASS>
    <DAY UNITS-"days">116 75</DAY>
    «RADIUS UNITS-"mi 1 es">3716</RADIUS>
    «DENSITY UNITS-"(Earth - !.)">.943</DENSITY>
    «DISTANCE UNITS-"mi LLIon miles^66'. 8</DISTANCEx! - -B перигелии—>
  <i/PLANET>
```

```
    «PLANET COLOR-"BLUE">
      <NAME>Earth«/NAME>
      «MASS UNITS-"(Earth - D">I</MASS>
      «DAY UNITS-"days">1«/DAY>
      «RADIUS UNIT$-"miles"»2107<;/RADIUS>
      «DENSITY UNITS-"(Earth - 1)"»1«/DENSITY>
      «DISTANCE UNITS-"ratilion miles">128.4«/DISTANCE><!--В перигелии-->
    </PLANET>

<   «/PLANETS»
```

Следующее выражение выбирает элементы <PLANET> с атрибутом COLOR:

```
    «xsl.template match-"PLANET[@COLOR]"»

    f I( · „ c
    «/xsl:template»
И /' 41                  J
```

А что, если нам требуется выбрать планеты, у которых атрибут COLOR имеет значение "BLUE" (голубой)? Это можно сделать при помощи операции · как показано в листинге 4.5.

Листинг 4.5. Применение операции =

```
«'xml version-"1 0"?>
«xsl-.stylesheet version-"I 0"
xmlns:xsl-"http://www.w3.prg/1999/XSL/Transform">

    «xsltemplate match-"PLANETS">
        <HTML>
            «xsl:apply-templates/>
        </HTML>
    «/xsl:template»

    «xsl:tempiate «match-"PLANET[@COLOR - 'BLUE1]">
            The <xslrvalue-of select-"NAME"/> is blue,
    «/xsl:template»

    «xsl:template match-"text()">
    «/xslrtemplate»

«/xsl:stylesheet»
```

Таблица стилей из листинга 4.5 отбирает планеты с голубым цветом и убирает остальные, выключая правило по умолчанию для текстовых узлов. Результат следующий:

```
«HTML»
   The Earth is blue.
«/HTML»
```

Создание предикатов

Предикаты — настоящие выражения XPath, и XPath гораздо ближе к настоящему языку, чем образцы; к примеру, выражения XPath мрхут возвращать не только списки узлов, но также логические, строковые и числовые значения. Выраже-

ния XPath могут работать не только с текущим узлом или дочерними узлами: можно работать с родительскими узлами, узлами-предками и другими узлами.

Глава 7 полностью посвящена XPath, но имеет смысл предоставить введение в предмет здесь, при обсуждении образцов, потому что часть предиката образца обладает наибольшими возможностями. В предикатах могут быть все виды выражений; в следующем списке перечислен ряд возможных типов, которые будут изучены в следующих разделах:

- наборы узлов;
- логические выражения;
- числа;
- строки.

Предикаты: наборы узлов

Набор узлов (node set), как понятно из названия, представляет собой просто совокупность узлов (и может содержать только один узел). Выражение child::PLANET возвращает набор узлов, состоящий из всех элементов <PLANET>. Выражение child::PLANET/child::NAME возвращает список узлов, состоящий из всех элементов <NAME>, дочерних по отношению к элементам <PLANET>. Для выбора узла или узлов из набора узлов воспользуйтесь следующими функциями для работы с наборами узлов в предикатах:

- last(). Возвращает количество узлов в наборе узлов;
- position(). Возвращает позицию контекстного узла в контекстном наборе узлов (начиная с 1);
- count(node-set). Возвращает количество узлов в наборе. Если опустить node-set, функция будет применена к контекстному узлу;
- id(string ID). Возвращает набор узлов, содержащий элемент с ID, удовлетворяющим переданной функции строке, или пустой набор узлов, если такой элемент отсутствует. Можно перечислить несколько идентификаторов, разделенных символами-разделителями, — тогда функция вернет набор узлов, состоящий из элементов с этими идентификаторами;
- local-name(node-set). Возвращает локальное имя первого узла в наборе узлов. Если опустить node-set, функция будет применена к контекстному узлу;
- namespace-uri(node-set). Возвращает URI пространства имен первого узла в наборе узлов. Если опустить node-set, функция будет применена к контекстному узлу;
- name(node-set). Возвращает полностью определенное имя первого узла в наборе узлов. Если опустить node-set, функция будет применена к контекстному узлу.

В листинге 4.6 я перенумеровал элементы в выходном документе при помощи функции positionQ.

Листинг 4.6. Применение функции position

```
<?xml version""1.0"?>
<xsl:stylesheet version-"1.0"
xml ns:xsl-"http://Www.w3.org/1999/XSL/Transfornf >

    <xslrtemplate match-"PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets
                </TITLE>
            </HEAD>
            <BODY>
                <xsl:apply-templates select-"PLANET'7>
            </BODY>
        </HTML>
    </xsl:tempiate>

    <xsl template match-"PLANET">
        <P>
            <xsl:value-of select^positlonO"/».

            <xsl:text> </xsl:text>
            <xsl :value-of select-"NAME7>
        </P>
    </xsl:template>

</xsl:stylesheet*
```

Вот результат. Как видите, планеты перенумерованы:

```
<HTML>
    <HEAD>
        <TITLE>
            The Planets
        </TITLE>
    </HEAD>

    <BODY>
        <P>
            1. Mercury
        </P>
        <P>
            2. Venus
        </P>
        <P>
            3. Earth
        </P>
    </BODY>
</HTML>
```

Можно также применять функции для работы с наборами узлов в предикатах, как, например, PLANETfposition() - last()], выбирающая последнего ребенка <PLANET> контекстного узла.

Предикаты: логические значения

В выражениях XPath можно также использовать логические (Boolean) значения. Для чисел ноль прнимается за ложь (false), другие значения — за истину (true). Пустая строка, "", также считается ложью, все остальные строки — истиной.

Для вычисления логических результатов true/false можно применять следующие логические операции XPath:

- !« означает «не равно»;
- < означает «меньше, чем» (в документах XML или XSL используйте <);
- <- означает «меньше или равно» (в документах XML или XSL используйте <=);
- • означает «равно» (программисты на C, C++, Java и JavaScript, обратите внимание: эта операция пишется как один знак =, а не два);
- > означает «больше, чем»;
- >= означает «больше или равно».

ИСПОЛЬЗОВАНИЕ СИМВОЛА <

Особенно обратите внимание на то, что непосредственно в документах XML или XSL нельзя использовать символ с, необходимо использовать ссылку на сущность <.

Для связи логических выражений логическими операциями And и Or используются ключевые слова and и or; слово not инвертирует логический смысл выражения — с истины на ложь или со лжи на истину.

В листинге 4.7 я определяю элемент <PLANET> Земли и помещаю в таблицу строки "Earth", "needs", "по" и "introduction" вместо числовых данных Земли. Я определяю, которая из планет есть Земля, при помощи предиката "[NAME—'Earth']", проверяющего значение элемента <NAME>, которое, в свою очередь, представляет собой заключенный в элементе текст. Я также предоставил шаблон Для других планет, удовлетворяющих предикату "[NAMEI-Earth']".

Листинг 4.7. Определение планеты Земля

```
<?xml version-"1.0"?>
<xsl stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

    <xsl:tempiate match-"/PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <BODY>
                <H1>
                    The Planets Table
                </H1>
                <TABLE BORDER="2">
                    <TR>
                        <TD>Name</TD>
```

```
                    <TD>Mass</TD>
                    «TD*Radius</TD>
                    <TD*Day«/TD*
                </TR>
                «x'sl ^apply-templ ates/*
            «/TABLE*    .
          </BODY>
        </HTML>
    </xsl:tempiate>

    <xsl:tempiate match-"PLANET[NAME-'Earth']">
      <TR>
        <TD>Earth</TD>
        <TD*needs</TD*
        <TD>no</TD>
        <T0>i ntroducti on.</TD>
      </TR>
    </xsl:template*

    <xsl -.template match-"PLANET[NAME!-'Earth']">
      <TR>
        <TD><xslrvalue-of select-"NAME"/*«/TD*
        <TD><xslrapply-tempiates select-"MASS"/></TD>
        <TD*«xslrapply-tempiates select-"RADIUS"/></TD>
        <TD><xsl:apply-tempiates select-"DAY"/></TD>
      </TR>
    «/xsl:template*

    <xslrtemplate match-"MASS">
      «xslrvalue-of select-"."/*
      <xsl - text* «/xsl r'text*
      <ksl:value-of select-'GUNITS"/»
    </xs1:tertiplate>

    <xslrtemplate match«"RADIUS">
      «xsl:value-of select-"."/>
      <xsl text* </xslrtext*
      <xsl:value-of select-"@LINITS"/>
    </xslrtemplate*

    <xsl template match-"DAY"*
      <xslrvalue-of select-"."/*
      <xslrtext* </xslrtext*
      <xslrvalue-of select-"@UNITS"/>
    </xslrtemplate*

</xslrstylesheet*
```

Вот результат:

```
<HTML>
  <HEAD*
    «TITLE*
      The Planets Table
    «/TITLE*
  «/HEAD*
  «BODY*
```

```
        <H1>
            The Planets Table
        </H1>

        <TABLE BORDER="2">
            <TR>
                <TD>Name</TD>
                <TD>Mass</TD>
                <TD>Radi us</TD>
                <T0>Day</TD>
            </TR>
            .
            .
            .
            <TR>
                <TD>Earth</TD>
                <TD>needs</TD>
                <TD>no</TD>
                <TD>i ntroducti on.</TD>
            </TR>
        </TABLE>
    <BODY>
</HTML>
```

Результат можно увидеть на рис. 4.1.

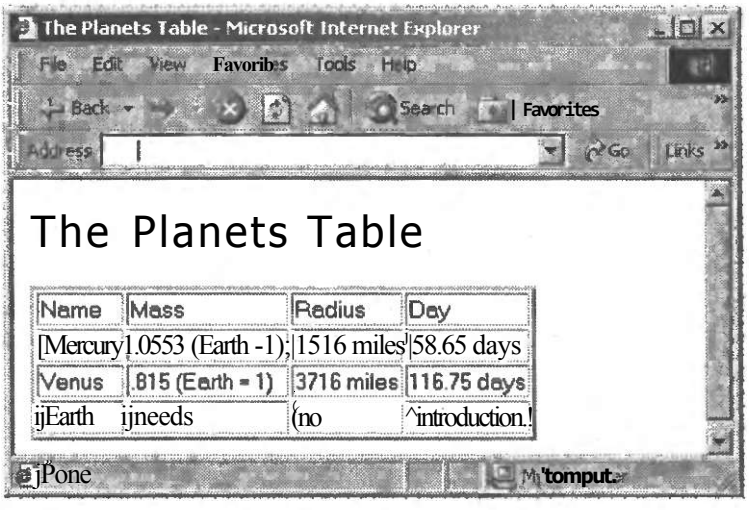


Рис. 4.1. Применение предикатов XPath

В следующем примере используется логическая операция >. Это правило применяется ко всем элементам <PLAN£T> после позиции 5:

```
<xslrtemplate match-"PLANET[position() > 5]">
    <xslrvalue-of  select-"."/>
</xsl:tempiate>
```

Имеется также функция true, всегда возвращающая значение true, и функция false, всегда возвращающая значение false. Функция not инвертирует логический

смысл выражения, как в следующем случае, где я выбираю все элементы <PLANET>, кроме последнего:

```
<xslrtemplate match-"PLANET£not(posit1on() - lastC))]">
    <xslrvalue-of select-"."/>
</xsl:tempiate>
```

Наконец, функция lang возвращает истину или ложь в зависимости от того, является ли язык контекстного узла (определяемый атрибутами xmlitang) таким же, как язык, который передан в эту функцию.

Предикаты: числа

В XPath числа хранятся в формате числа с плавающей точкой двойной точности. (Технически все числа XPath хранятся в 64-разрядном формате IEEE числа с плавающей точкой двойной точности, floating-point double.) Все числа хранятся как числа с двойной точностью — даже целые числа, как 5 в рассматриваемом примере:

```
<xslrtemplate match-"PLANET[position() > 5]">
    <xslrvalue-of select-","/>
</xslrtempiate>
```

Над числами можно производить ряд операций:

- + сложение;
- - вычитание;
- * умножение;
- div деление (символ /, соответствующий делению в других языках, в XML, XSL и XPath уже занят);
- mod возвращает значение деления по модулю двух чисел (остаток после деления первого числа на второе).

Например, элемент <xsl:value-of select-"180 + 420"/> вставит в выходной документ строку "600". В следующем примере выбираются все планеты, у которых отношение дня (измеренного в днях Земли) к массе (где масса Земли принята за 1) больше 100:

```
<xslrtemplate match="PLANETS">
    <HTML>
        <BODY>
            <xslrapply-templates select-"PLANET[0AY div MASS > 100]"/>
        </BODY>
    </HTML>
</xsl:tempiate>
```

XPath также поддерживает следующие функции работы с числами:

- ceiling(). Возвращает наименьшее целое, большее, чем переданное функции число;
- floor(). Возвращает наибольшее целое, меньшее, чем переданное функции число;

- roundQ. Округляет переданное число до ближайшего целого;
- sum(). Возвращает сумму переданных функции чисел.

Например, среднюю массу планет в planets.xml можно найти так, как в листинге 4.8:

Листинг 4-Э- Вычисление средней массы планет

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-'http://www.w3.org/1999/XSL/Transform">
<xsl:output method-"xml"/>

<xsl:tempi ate match^'PLANETS'»
    <HIML>
        <BODY>
            The average planetary mass is:
<xsl:value-of select-"sum(child::PLANET/ch1ld::HASS) div count(child::PLANET)"/>
        </BODY>
    </HIML>
</xsl•tempiate>

</xsl stylesheet>
```

Строки

В XPath строки формируются из символов Unicode, как можно было предположить. Ряд функций специально предназначен для работы со строками:

- string(object objectl). Преобразует объект в строку;
- starts-with (string stringl, string string2). Возвращает истину, если первая строка начинается (starts with) со второй строки;
- contains(string stringl, string string2). Возвращает истину, если первая строка содержит (contains) вторую строку;
- substring (string stringl, number offset number length). Возвращает length символов из строки, начиная со смещения offset;
- substring-before(string stringl, string string2). Возвращает часть строки stringl до первого вхождения строки string2;
- substring-after(string stringl, string string2). Возвращает часть строки stringl после первого вхождения string2;
- string-length(string stringl). Возвращает количество символов в строке stringl;
- normalize-space(string stringl). Возвращает строку stringl после отбрасывания лидирующих и завершающих символов-разделителей и замены нескольких последовательных разделителей на один пробел;
- translate(string stringl, string string2, string string3). Возвращает строку stringl, в которой все вхождения символов в строке string2 заменены на соответствующие символы в строке string3;
- concat(string stringl, string string2,.,,,). Возвращает конкатенацию (объединение) всех строк.

Есть еще одна строковая функция» о которой вам следует знать, входящая не в XPath, а в XSLT:

- format-number(number numberl, string string2, string string3), Эозвращает строку, содержащую число numberl в виде форматированной строки, используя string2 в качестве форматирующей строки (форматирующие строки создаются так же, как для метода Java java.text.DfeeimalFormat) й strings как возможную строку локализации.

В листинге 4.9 я выбираю текстовые узлы, в которых текст начинается с 'E', чтобы выбрать Earth (Земля), и добавляю текст '(the World)' (мир), получая 'Earth (the World)'. Для этого я применяю предикат "text()[starts-with(., 'E')]".

Листинг 4.9. Применение функции starts-with

```
<?xml version="1.0"?>
<xsl:stylesheet version»"1.0"
xmlns xsl-"http://www.w3 org/1999/XSL/Transform">

    <xs1:tempi ate match-"/PLANETS">
        <HTML>
            <HEAD>

            </BODY>
        </HTML>
    </xs1:tempiate>          f/

    <xsl-rtemplate match-"PLANET">
        <TR>
            <TD><xsl:apply-templates select-"NAME"/></TD>
            <TD><xsl:apply-templates select-"MASS'7></TD>
            <TD><xsl: apply-templ ates seTect-"RADIUS"/x/TD>
            <Tp><xs1:apply-templates select="DAY"/></TD>
        </TR>
    </xsl:tempiate>
                                                i      .      .          *
    <xslrtemplate match-"text()[starts-with(,.  'E')]">
        <xsl:text>(the World)</xsl:text>
    </xsl:template»                                  > ь

    <xsl template match*-'NAME'>
        <xsl:value-of select-"."/?»
        <xsl:text> </xsl:text>
        <xsl:value-of select="@UNITS"/>
•' i.'  <xsl apply-templ ates/*
    </xsl:tempiate>

    <xsl:template match-"DAY">
        <xsl:vanue-of select-".*7>
        <xsl:text> </xsl:text>
        <xsl:value-of select-'GUNITS"/»
    </xsl:template»
</xsl:stylesheet»
```

А вот результат — заметьте, что заголовок для Земли стал "Earth (the World)":

```
<HTML>
    <HEAD>
        <TITLE>
            The Planets Table
        </TITLE>
    </HEAD>

    <BODY>
        <H1>
            The Planets Table
        </H1>

        <TABLE BORDER="2">
            <TR>
                <TD>Name</TD>
                <TD>Mass</TD>
                <TD>Radius</TD>
                <TD>Day</TD>
            </TR>
            .
            .
            .
            <TR>
                <TD>Earth (the World)</TD>
                <TD>1 (Earth - 1)</TD>
                <TD>2107 miles</TD>
                <TD>1 days</TD>
            </TR>
        </TABLE>
    </BODY>
</HTML>
```

Этот документ показан на рис. 4.2.

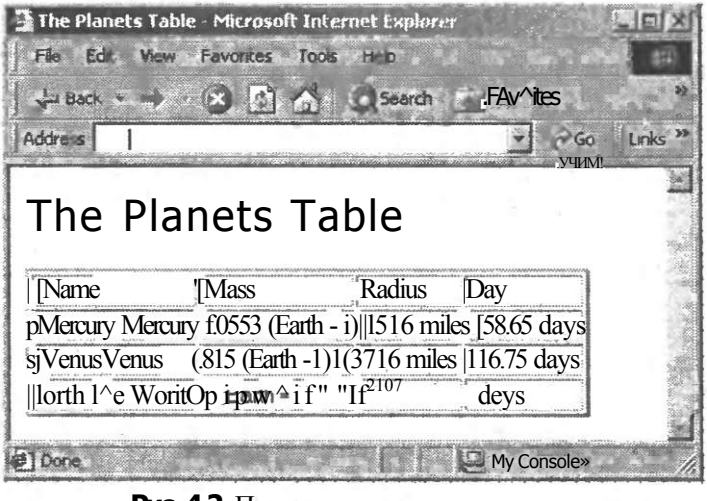


Рис. 4.2. Применение текстовых предикатов

Предикаты: фрагменты результирующего дерева

XSLT 1.0 добавляет к поддерживаемым XPath типам данных фрагменты результирующего дерева. Это фрагменты дерева, которые можно присваивать переменным XSLT, они не очень широко распространены. Практически все, что можно с ними сделать, — это вычислить их строковое значение. В рабочем проекте XSLT 1.1 их поддержка была удалена, поэтому, видимо, в XSLT 2.0 их уже не будет.

Сокращенный синтаксис предикатов

Выражения предикатов можно сокращать, опуская "position() =". Например, [position() - 3] становится [3], [position() •• last()] становится [last()] и т. д. С использованием сокращенного синтаксиса применять выражения XPath в предикатах становится существенно проще. Вот ряд примеров:

- PLANET[2]. Возвращает второго ребенка <PLANET> контекстного узла;
- PLANET[last()]. Возвращает последнего ребенка <PLANET> контекстного узла;
- /PLANETS/PLANET[2]/NAME[1], Возвращает первый элемент <NAME> второго элемента <PLANET> элемента <PLANETS>;
- PLANET[5][@UNITS - "million miles"]. Возвращает пятого ребенка <PLANET> контекстного узла, только если у него имеется атрибут UNITS со значением "million miles". Это выражение можно также написать как PLANET[@UNITS» "million miles"] [5].

На этом мы заканчиваем рассмотрение трех частей образцов шага: осей, условий узлов и предикатов. Это строительные блоки образцов выбора. Лучше всего изучить создание образцов на примере, и многие примеры мы вскоре рассмотрим. Сначала, однако, важно рассмотреть две небольшие темы. Как вы помните из формального определения образцов выбора, можно помимо образцов шага, создавать образцы, выбирающие элементы по идентификатору (ID) или ключу.

Выбор по ID

В дополнение к созданию образцов из образцов шага, задающих ось, условие узла и предикат, можно применять и образец id() для выбора элементов с определенным значением ID. Для работы с этим образцом необходимо задать элементам атрибут ID, который должен быть объявлен с типом ID, что можно сделать в DTD или схеме документа. В следующем примере правило добавляет текст всех элементов, имеющих ID "favorite":

```
<xsl:template match - "idCfavorite')">
    <H3><xslrvalue-of select-"."/></H3>
</xsl:template>
```

Вот как может выглядеть объявление DTD для planets.xml, в котором объявляется ID и его значение устанавливается в "favorite":

```
<?xml version-"1.0"?>
<?xml-stylesheet type-"text/xml" href-"#stylesheet"?>
<IDOCTYPE PLANETS [
<!ELEMENT PLANET (CUSTOMER)*>                                    ,n
<!ELEMENT CUSTOMER (NAME,MASS,RADIUS.DAY)>
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT MASS (#PCDATA)>
<!ELEMENT RADIUS (#PCDATA)>
<!ELEMENT DAY (#PCDATA)>
<!ATTLIST PLANET
    id ID #REQUIRED>
]>
<PLANETS>
    <PLANET id-"favorite">
        <NAME>Mercury</NAME>
        <MASS UNITS-"(Earth - 1)">.0553</MASS>
        <DAY UNITS-"days">58.65</DAY>
        <RADIUS UNITS-"mi 1es">1516</RADIUS>
        <DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
        <DISTANCE UNITS-"million miles">43.4</DISTANCE><!--B перигелии-->
    </PLANET>
```

Ряд процессоров XSLT не может осуществлять выбор по ID, потому что они не читают DDS или схему XML. (Возможность доступа к информации ID должна быть включена в XSLT 2.0.) Но есть альтернатива: можно осуществлять выбор по ключу.

ВОЗМОЖНАЯ ПОДДЕРЖКА IDREF

Помимо упрощения работы с ID, в XSLT 2.0 W3C даже рассматривает возможность включения поддержки IDREF, В частности, по заданному ID процессор XSLT может предоставить список всех элементов с атрибутом IDREF или IDREFS, ссылающихся на этот ID. (Заметьте, что сейчас это тоже можно сделать при помощи элемента <xsl:key> и образца "keyQ".)

Выбор по ключу

Ключи дают простой способ идентифицировать элементы; конкретные ключи можно выбрать при помощи образца "key()". Работа с ключами подробно обсуждается в главе 9, но здесь я также приведу небольшой пример.

Для создания ключа служит элемент <xsl:key>. Это элемент верхнего уровня, поэтому он используется вне шаблонов и как дочерний элемент <xsl:stylesheet>. В следующем примере я при помощи ключа выбираю планеты, чей атрибут COLOR (цвет) установлен в "BLUE" (голубой), что означает Землю:

```
<?xml version="l 0"?>
<?xml-stylesheet type-"text/xml" href="planets.xsl"?>
<PLANETS>

    «PLANET COLOR="BLUE">
```

```
        <NAME>Earth</NAME>
        <MASS UNITS-"(Earth - 1)'>1</MASS>
        <DAY UNITS-"days">l</DAY>
        <RADIUS UNITS-"miles">2107«/RADIUS>
        «DENSITY UNITS-"(Earth - 1)">1</DENSITY>
<$[bold>        «DISTANCE UNITS-"million miles">128.4</DISTANCEx!"B перигелии-^
    </PLANET>

</PLANETS>
```

Теперь я могу создать ключ с именем COLOR, который выбирает элементы <PLANET>, проверяя их атрибут COLOR:

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http.//www.w3.org/1999/XSL/Transform">

    <xsl:key name-"COLOR" match-"PLANET" use-'eCOLOR"/»
```

Теперь при помощи образца "key()" можно выбрать элементы <PLANET> с атрибутом COLOR со значением "BLUE":

```
«?xml versior^"1.0"?>
<xsl:stylesheet version-"l.0"
xmlns:xsl-"http://www.^3.org/1999/XSL/Transfortp">

    <xsl:key name-"COLOR" match-"PLANET" use-'@COLOR'/>

    <xsl:tempi ate match-"/PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <BODY>
                <HI>
                    The Planets Table
                </HI>
                «TABLE BORDER-"2">
                    <TR>
                        <TD>Name«/TD>
                        <TD>Mass«/TD>
                        <TD>Radius«/TD>
                        <TD>Day«/TD>
                    </TR>
                    <xsl:apply-tempiates select-"key('COLOR'.  'BLUE*)"/>
                </TABLE>
            </BODY>
        «/HTML>
    «/xsl:tempiate>
```

А вот результат — как видите, единственной планетой, удовлетворявшей используемому образцу, была Земля (Earth).

```
<HTML>
    <HEAD>
        <TITLE>
            The Planets Table
        </TITLE>
    </HEAD>

    <BODY>
        <H1>
            The Planets Table
        </H1>

        <TABLE BORDER*"2">
            <TR>
                <TD>Name</TD>
                <TD>Mass</TD>
                <TD>Radius</TD>
                <TD>Day</TD>
            </TR>

            <TR>
                <TD>Earth</TD>
                <TD>1 (Earth - 1)</TD>
                <TD>2107 miles</TD>
                <TD>1 days</TD>
            </TR>
        </TABLE>
    </BODY>
</HTML>
```

Применение операции Or

При помощи операции Or (или), |, можно осуществлять выбор по нескольким возможным образцам, что очень удобно с ростом сложности документов. В следующем примере я хочу отобразить элементы <NAME> и <MASS> полужирным, что задается тегом HTML <B>. Для выбора либо элементов <NAME>, либо <MASS> в новом правиле я использую операцию Or:

```
<?xml version-"1 0"?>
<xsl:stylesheet version-"l 0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

    <xslrtemplate match-"/PLANETS">
        <HTML>
            <HEAD>


            </BODY>
        </HTML>
```

```
    </xs1rtemplate»

    <xslrtemplate match-"PLANET">
       <TR>
          <TD><xsl:apply-templates select-"NAME"/></TD>
          <TDxxsl :apply-templates select="MASS"/></TD>
          <TD><xsl:apply-templates select-"RADIUS'7></TD>
          <TD><xsl:apply-templates select="DAY'7></TD>
       </TR>
    </xsltemplate»

    <xslrtemplate match-"NAME | MASS">
       <B>
          <xsl:apply-templates/»
       </B>
    </xsl .-template»

    <xslrtemplate match-"RADIUS">
       <xsl.value-of select-"."/>
       <xsl:text> </xsl:text>
       <xslrvalue-of select-"@UNITS"/>
    </xslrtemplate»

    <xslrtemplate match-"DAY">
       <xslrvalue-of select-"."/>
       <xslrtext> </xsl:text>
       <xsl rvalue-of select-"@UNITS"/>
    </xsl template»

</xsl.stylesheet»
```

Вот результаты; заметьте, что значения имени и массы заключены в элементы <B>:

```
<HIML>
   <HEAD>
      <TITLE>
         The Planets Table
      </TITLE>
   </HEAD>

   <BODY>


      <TR>
         <TD><B>Mercury</B></TD>
         <TD><B>.0553</B></TD>
         <TD>1516 miles</TD>
         <TD>58.65 days</TD>
      </TR>

      <TR>
         <TD><B>Venus</B></TD>
         <TD><B>.815</B></TD>
         <TD>3716 miles</T0>
         <TD>116.75 days</TD>
      </TR>
```

```
            <TR>
                <TD><B>Earth</B></TD>
                <TD><B>1</B></TD>
                <TD>2107 miles</TD>
                <TD>1 days</TD>
            </TR>
        </TABLE>
    </BODY>
</HTML>
```

Операцию | можно применять к любым допустимым образцам — например, в таких выражениях, как "PLANET | PLANET//NAME"; можно Использовать несколько операций | — например, "NAME | MASS | DAY" и т. п.

Примеры образцов

Изучать образцы лучше всего на примерах. Предположим, что нам требуется преобразовать planets.xml в planets.html, но сохранить только первую планету, Меркурий. Это можно сделать при помощи предиката [position() < 2], так как позиция первой планеты равна 1, следующей — 2, и т. д. Заметьте, однако, что <; является управляющим символом для процессоров XSLT, начинающим разметку, поэтому вместо < необходимо использовать &lt.;. И отметьте, что для того чтобы убрать из planets.xml другие элементы, для них нужно предоставить пустой шаблон, что можно сделать при помощи предиката {positionQ >= 2] (листинг 4.10).

Листинг 4.10. Выбор только Меркурия

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"!.O"
xmlns; xsl -"http: / /www. w3. org/1999/XSL/Transform">

    <xslrtemplate match-"/PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <BODY>
                <H1>
                    The Planets Table
                </H1>
                <TABLE BORDER-"2">
                    <TR>
                        <TD>Name</TD>
                        <TD>Mass</TD>
                        <TD>Radius</TD>
                        <TD>Day</TD>
                    </TR>
                    <xsl;apply-templates/>
                </TABLE>
            </BODY>
        </HTML>
```

```
    </xsl:template»

    <xslrtemplate match-"PLANET[posit1on() &lt.: 2]">
       <TR>
          <TDxxsl: val ue - of sel ect-"NAME" /x/TD>
          <TDxxsl: apply-tempi ates select«"MASS"/»</TD>
          <TDxxsl: appl y- tempi ates sel ect-"RADIUS" /></TD>
          <TD»<xsl: apply-templ ates select«"DAY"/»</TD»
       </TR>
   </xslrtemplate»

    <xslrtemplate match-"PLANET[position() >- 2]">
   </xslrtemplate»
<xslrtemplate match-"MASS">
        <xslrvalue-of  select-"."/>
        <xslrtext» </xslrtext»
        <xsl:value-of  select-"@UNITS"/>
    </xsl  template»

    ^xsTrtemplate match-"RADIUS">
        <xs-l:value-of  select-"."/»
        <xsl,: text» </xs 1: text»
        <xsl:value-of  select-"@UNIT$'7>
    </xslrtemplate»

    <xslrtemplate match-"0AY">
        <xslrvalue-of  select-"."/>
        <xslrtext» </xslrtext»
        <xsl rvalue-of select-"(aUNITS"/>
    </xslrtemplate»

</xsl:stylesheet»
```

Вот результирующий документ — отметьте, что сохранилась только первая планета, Меркурий:

```
    <HTML»
        <HEAD>
            <TITLE>
                The Planets Table
            </TITLE>
        </HEAD»

        <BODY>
            <H1>
                The Planets Table
            </H1>

            <TABLE BORDER-"2"»
                <TR>
                    <TD»Name</TD>
                    <TD»Mass</TD>
                    <TD»Radius</TD>
                    <TD»Day</TD>
                </TR>

                <TR>
```

```
                <TD>Mercury</TD>
                <TD>.0553 (Earth - 1)</TD>
                <TD>1516 miles</TD>
                <TD>58.65 days</TD>
            </TR>
        </TABLE>
    </BODY>
</HTML>
```

В следующем примере в элемент <PLANET> Земли были добавлены атрибуты COLOR (цвет) и POPULATED (населена):

```
<PLANET COLOR»"blue" POPULATED-"yes">
    <NAME>Earth</NAME>
    <MASS UNITS-"(Earth - 1)">1</MASS>
    <DAY UNITS-"days">l</DAY>
    <RADIUS UNITS-"miles">2107</RADIUS>
    <DENSITY UNITS-"(Earth = 1)">1</DENSITY>
    <DISTANCE UNITS-"million miles">128 4</DISTANCE><!—B перигелии—»
</PLANET>
```

Как выбрать только элементы, имеющие оба атрибута, COLOR и POPULATED? Можно применить предикат "[(©COLOR and (©POPULATED]". Чтобы убрать другие элементы — так, чтобы правило по умолчанию не поместило их текст в результирующий документ, — можно применить предикат "[not(@C0L0R) or not(@P0PULATED)]", как показано в листинге 4.11.

Листинг 4.11. Выбор только элементов с двумя атрибутами COLOR и POPULATED

```
<?xml version="1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

    <xslrtemplate match-"/PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    Colorful. Populated Planets
                </TITLE>
            </HEAD>
            <BODY>
                <H1>
                    Colorful. Populated Planets
                </H1>
                <TABLE BORDER-"2">
                    <TR>
                        <TD>Name</TD>
                        <TD>Mass</TD>
                        <TD>Radius</TD>
                        <TD>Day</TD>
                    </TR>
                    <xsl:apply-templates/>
                </TABLE>
            </BODY>
        </HTML>
    </xsl:tempiate>

    <xslrtemplate match-"PLANET[@C0L0R and ^POPULATED]"»
```

```
        <TR>
          <TD*<xsl:value-of select-"NAME"/*</TD*
          <TD><xsl: appl y- tempi ates sel ect- "MASS" /x/TD>
          <TDxxsl: apply-tempi ates select-"RADIUS"/x/TD>
          <TDxxsl -.apply-tempi ates select-"DAY"/></TD>
        </TR>
    </xsl:template*

    <xslrtemplate match-"PLANET[not(PCOLOR) or not«aPOPULATED)]">
    </xsl:template*

    <xslrtemplate match-"MASS">
        <xsl:value-of select-"."/*
        <xsl:text> </xsl:text>
        <xsl:value-of select-"@UNITS'7>
    </xsl:template*

    <xsl:template match-"RADIUS">
        <xslrvalue-of select-"."/*
        <xsl:text* </xslrtext*
        <xsl:value-of select-"@UNITS'7>
    </xslrtemplate*

    <xslrtemplate match-"DAY">
        <xsl:value-of select-"."/>
        <xsl:text* </xsl:text*
        <xslrvalue-of select-"@UNITS'7*
    </xsl rtemplate?-

</xslrstylesheet*
```

А вот результат:

```
<HTML>
    <HEAD>
        <TITLE*
            Colorful. Populated Planets
        </TITLE*
    </HEAD*

    <BODY>
        <Hl*
            Colorful. Populated Planets
        </Hl*

        <TABLE BORDtR-"2">
            <TR>
                <TD*Name</TD*
                <TD*Mass</TD>
                <TD*Radius</TD*
                <TD*Day</TD*
            </TR*

            <TR>
                <TD*Earth</TD>
                <TD*1 (Earth - 1)</TD*
                <TD*2107 miles</TD*
                <TD>1 days</TD
```

```
            </TR>
          </TABLE>
        </BODY>
      </HTML>
```

Этот документ показан на рис. 4.3.

В следующем примере я копирую planets.xml в новый XML-документ и изменяю текст в элементе <NAME> Венеры на "The Planet of Love" (планета любви). Для этого я сначала копирую в результирующий документ все узлы и атрибуты:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">
<xs1.output methoch"xml"/>

  <xsl:tempiate match-"@*|node()">
    <xsl:copy>
      <xsl:apply-tempiates select-"@*|node()"/>
    </xsl:copy>
  </xsl:template*
```

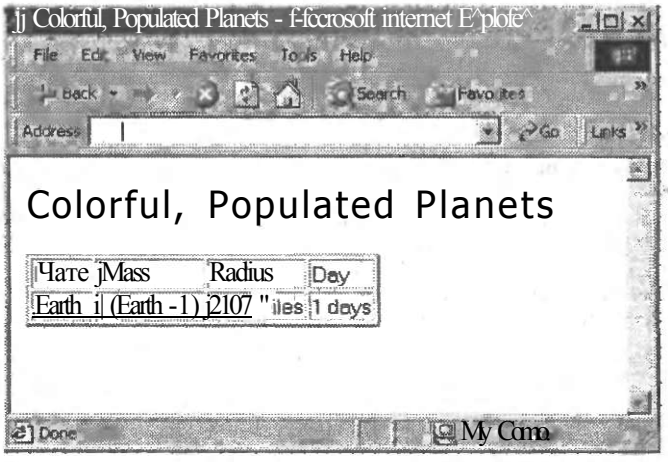


Рис. 4.3. **Применение предикатов XPath для проверки атрибутов**

Теперь я добавлю новое правило, выбирающее элементы <NAME> с текстом "Venus" (Венера) по образцу "NAME[text() = 'Venus']". Хотя элементы <NAME> удовлетворяют обоим правилам этой таблицы стилей, правило с образцом "NAME[text() = 'Venus']" осуществляет более узкий выбор, поэтому для элемента <NAME> Венеры процессор XSLT применит его:

```
,<?xml versi.on="1.0"?>
 <xsl:stylesheet version-"1.0"
 xml ns: xsVhttp: //www. w3. org/1999/XSL/Transfbrm">
 <xsl:output method="xm1"/>

   <xslrtemplate match="@*|node()">
```

```
    <xsl:copy>
      <xsl:apply-templates select-"@*|node()"/>
    </xsl:Copy>
  </xsl template*

    <xsl:tempiate match-"NAME[text() - 'Venus']">
        <NAME>
            The Planet of Love
        </NAME>
    </xslrtemplate*

</xsl:stylesheet*
```

И вот результат:

```
<?xml version-"1.0" encoding-"utf-8"?*
<?xml-stylesheet type-"text/xml" href-"planets.. xsl "?*
<PLANETS*

    <PLANET>
        <NAME*Mercu ry</NAME*
        <MASS UNITS-"(Earth - 1)"*.0553</MASS>
        <DAY UNITS-"days">58.65</DAY>
        <RADIUS UNITS="miles">1516</RADIUS>
        <DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
        «DISTANCE UNITS-"million miles">43.4</DISTANCE><!--B перигелии-->
    </PLANET>

    <PLANET>
        <NAME>
            The Planet of Love
        </NAME>
        <MASS UNITS-"(Earth - 1)"*.815</MASS>
        <DAY UNITS-"days"*116.75</DAY*
        <RADIUS UNITS-"mi 1es">3716</RADIUS*
        <DENSITY UNITS-"(Earth - 1)">.943</DENSITY>
        <DISTANCE UNITS-"million miles">66 8</DISTANCE><!--B перигелии-->
    </PLANET>

    <PLANET>
        <NAME>Eath</NAME>
        <MASS UNITS-"(Earth - 1)">1</MASS>
        <DAY UN ITS^" days" >1</DAY>
        <RADIUS UNITS-"miles">2107</RADIUS>
        <DENSITY UNITS-"(Earth - 1)">1</DENSITY>
        <DISTANCE UNITS-"million miles">128.4</DISTANCE><!--B перигелии--*
    </PLANET>

</PLANETS*
```

В действительности, в выражениях XPath можно ссылаться на контекстный узел при помощи ".", и значением по умолчанию для узла является его текст, поэтому следующее правило работает точно так же:

```
<?xml version-"1.0"?*
<xsl stylesheet version-"1.0"
xmlns:xsl'-"http://www.w3.org/1999/XSL/Transform">
```

```
<xsl:output method-"xml "/>

  <xsl template match-"@*|node()"*
    <xsl:copy*
      <xsl:apply-templates select-"@*|node()"/>
    </xsl:copy>
  </xsl:template*

    <xslrtemplate match-"NAME[. - 'Venus']"*
        <NAME*
            The Planet of Love
        </NAME*
    </xslrtemplate*

</xsl stylesheet*
```

Имеет смысл привести как можно больше примеров — примеров XPath или образцов выбора никогда не бывает слишком много. Ниже приведен содержательный ряд примеров образцов выбора:

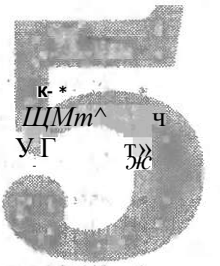
- PLANET выбирает дочерние элементы <PLANET> контекстного узла;
- /PLANETS выбирает корневой элемент (PLANETS) документа;
- * выбирает все дочерние элементы контекстного узла;
- PLANET[3] выбирает третьего ребенка <PLANET> контекстного узла;
- PLAN ET[last()] выбирает последнего ребенка <PLANET> контекстного узла;
- PLANET[NAME] выбирает детей <PLANET> контекстного узла, имеющих детей <NAME>;
- PLANET[DISTANCE]/NAME выбирает все элементы <NAME> элементов <PLANET>, содержащих по крайней мере один элемент <DISTANCE>;
- PLANET[DISTANCE]/PLANET[DAY] выбирает все элементы <PLANET> элементов <PLANET>, в которых элемент <PLANET> содержит по крайней мере один элемент <DISTANCE>, и элемент <PLANET> содержит по крайней мере один элемент <DAY>;
- P L AN ETS [P L A N ET/D AY] выбирает все элементы <PLANETS>, содержащие элементы <PLANET> как минимум с одним элементом <DAY>;
- PLANET[DISTANCE][NAME] выбирает все элементы <PLANET>, имеющие Элементы <DISTANCE> и <NAME>;
- PLANETS/PLANET[last()] выбирает последний элемент <PLANET> в каждом элементе <PLANETS>;
- */PLANET выбирает всех внуков <PLANET> контекстного узла;
- /PLANETS/PLANET[3]/NAME[2] выбирает второй элемент <NAME> третьего элемента <PLANET> элемента <PLANETS>;
- //PLANET выбирает всех потомков <PLANET> корня документа;
- PLANETS//PLANET выбирает потомков элемента <PLANET> детей элемента <PLANETS> контекстного узла;

- //PLANET/NAME выбирает все элементы <NAME>, дочерние по отношению к родителю <PLANET>;
- P L A N ETS//P L A N ET/DIST A N C E//P E RIH E LI0 N выбирает элементы <PERIHELION> везде внутри элемента <DISTANCE> элемента <PLANET>, везде внутри элемента <PLANETS>;
- @UNITS выбирает атрибут UNITS контекстного узла;
- @* выбирает все атрибуты контекстного узла;
- *[@UNITS] выбирает все элементы с атрибутом UNITS;
- DENSITY/@UNITS выбирает атрибут UNITS в элементах <DENSITY>;
- PLANET[not(@C0L0R) or not(@SIZE)] выбирает элементы <PLANET>, не имеющие обоих атрибутов COLOR и SIZE;
- PLANETS[@STAR-"Sun"]//DENSITY выбирает любой элемент <DENSITY> с элементом-предком <PLANETS>, имеющим атрибут STAR со значением "Sun";
- PLANET[NAME-f"Venus"] выбирает детей <PLANET> контекстного узла, имеющих детей <NAME> с текстом "Venus";
- PLANET[NAME[1] - "Venus"] выбирает все элементы <PLANET>, у которых в первом элементе <NAME> содержится текст в "Venus";
- DAY[@UNITS !- "million miles"] выбирает все элементы <PLANET>, атрибут UNITS которых не равен "million mites";
- PLANET[@UNITS « "days"] выбирает всех детей <PLANET> контекстного узла, имеющих атрибут UNITS со значением "days";
- PLANET[6][@UNITS." "days"] выбирает шестого ребенка <PLANET> контекстного узла, только если у этого ребенка есть атрибут UNITS со значением "days", — что также можно записать как PLANET[@UNITS = "days"][6];
- PLANET[@C0L0R and @UNITS] выбирает всех детей <PLANET> контекстного узла, имеющих оба атрибута COLOR и UNITS;
- *[1][NAME] выбирает любой элемент <NAME>, являющийся первым ребенком своего родителя;
- *[position() < 5] выбирает первые пять детей контекстного узла;
- *[position() < 5][@UNIT] выбирает первые пять детей контекстного узла с атрибутовм UNITS;
- text() выбирает все дочерние текстовые узлы контекстного узла;
- text()[starts-with(;, "In the course of human events")] выбирает все дочерние текстовые узлы контекстного узла, начинающиеся с "In the course of human events";
- /PLANETS[@UNITS - "million miles"] выбирает все элементы PLANETS, у которых значение атрибута UNITS равно "million miles";
- **PLANET[/PLANETS/@UNITS - ©REFERENCE]** выбирает все элементы <PLANET>, у которых значение атрибута REFERENCE такое же, как значение атрибута UNITS элемента PLANETS в корне документа;

- PLANET/* выбирает все дочерние элементы элементов PLANET;
- PLANET/*/DAY выбирает все элементы DAY—правнуки элементов PLANET, являющиеся детьми контекстного узла;
- */* выбирает элементы-внуки текущего элемента;
- astrophysics:PLANET выбирает элемент PLANET в пространстве имен «astrophysics»;
- astrophysics:* выбирает любые элементы в пространстве имен «astrophysics»;
- PLANET[DAY and DENSITY] выбирает все элементы <PLANET>, у которых есть по крайней мере один элемент <DAY> и один элемент <DENSITY>;
- PLANET[(DAY or DENSITY) and MASS] выбирает все элементы <PLANET>, у которых есть по крайней мере один элемент <DAY> или один элемент <DENSITY>, а также по крайней мере один элемент <MASS>;
- PLANET[DAY and not(DISTANCE)] выбирает все элементы <PLANET>, у которых есть по крайней мере один элемент <DAY> и нет элементов <DISTANCE>;
- PLANET[MASS - /STANDARD/REFERENCE/MASS] выбирает все элементы <PLANET>, у которых значение элемента <MASS> равно значению элемента /<STANDARD>/<REFERENCE>/<MASS>.

На этом мы завершаем в данный момент рассмотрение образцов выбора; связанный материал приводится в главе 7 при рассмотрении выражений XPath. Глава 5 начинается с изучения способов работы с данными в XML-документах путем сортировки и принятия решения на основе значений данных.

Принятие решений и сортировка данных



г.

Эха глава посвящена принятию решений,, сортировке и раз /ичным способам обработки данных в XM1--документах. Мы рассмотрим элементу <xsl:if>j <xsl:choose>, <xsl:when>, <xsl:otherwise>, <xsl:for-each> и <xsl:sort>, при помощи которых вы сможете обрабатывать данные и управлять выполнением преобразования в зависимости от значения данных.

Однако эти элементы не предоставляют такой точности, как в языках программироианйя. Поэтому я также представлю в этой главе расширения XSLT, в том числе элемент рабочего проекта XSLT 1.1 <xsl:script>. Этот элемент был предназначен для упрощения применения Java и JavaScript с процессором XSLT. (Для чтения этой книги нет необходимости владеть Java или JavaScript, но если вы знаете эти языки, вам будет приятно удостовериться, что некоторые процессоры XSLT дают возможность использовать их при преобразованиях XML.) Нечто похожее на этот элемент обязательно появится в XSLT 2.0. При помощи расширений вы можете расширять спецификацию XSLT, добавляя в XSLT новые элементы и функции какого-либо производителя или свои собственные.

Кроме того, в этой главе мы также рассмотрим, как перенумеровать элементы в документе, что делать в случае, когда ваш процессор XSLT не поддерживает определенное расширение, и многое другое. Я начну с наиболее часто используемого элемента из рассматриваемых в данной главе: <xsl:if>.

Элемент <xsl:if>

При помощи элемента <xsl:if> осуществляются проверки условия и принимаются действия на основе результата проверки. Он во многом похож на оператор if в языках программирования. У элемента <xsl:if> один атрибут:

- test (обязательный). Устанавливается в значение логического (Boolean, true/false) условия, которое вы хотите проверить.

Элемент заключает в себе тело шаблона.

Вот как это работает: вы включаете тело шаблона внутрь элемента <xsl:if>, проверяющего какое-то выражение. Если это выражение истинно, тело шаблона используется, если ложно — игнорируется:

```
<xsl:if test="expression">
    <!--template body-->
</xsl:if>
```

Можно проверять любое выражение XPath. Для преобразования его в значения true/false в элементе <xsl:if> применяйте следующие правила:

- если выражение вычисляется в набор узлов, оно трактуется как true, когда набор узлов содержит хотя бы один узел;
- выражение-строка считается true, если строка не пуста;
- фрагмент результирующего дерева трактуется как true, если содержит узлы;
- если результат выражения — число, он считается true, когда отличен от нуля.

Элемент <xsl:if> во многом похож на оператор if-then в языках программирования. Однако не существует оператора <xsl:else> для формирования конструкций if-then-else — для этого служит элемент <xsl:choose>.

В листинге 5.1 я перечисляю планеты в planets.xml одну за другой и добавляю горизонтальное правило HTML, элемент <HR> (horizontal rule), после последнего элемента — но только после последнего. При помощи <xsl:if> это можно сделать так.

Листинг 5.1. Применение :xsl:if>

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"i>

<xsltemplate match="PLANETS">
    <HIML>
        <HEAD>
            <TITLE>
                Planets
            </TITLE>
        </HEAD>
        <BODY>
            <xsl:apply-templates select-"PLANET"/>
        <4BODY>
    </HTML>
</xsl:tempiate>

<xslrtemplate match-"PLANET">
    <P>
    <xsl :value-of select«"NAME7>
    is planet number <xsl:value-of select-"posi ti on O"/> from the sun.
    </P>
    <xsl:if test-"pos1tion() - 1astC)"><HR/>c/xsl:if>
</xsl:template»

</xsl:stylesheet>
```

Вот результат — как видите, элемент <HR> появляется только после последней перечисленной планеты:

```
<HTML>
    <HEAD>
        <TITLE>
            Planets
        </TITLE>
    </HEAD>

    <BODY>
        <P>
            Mercury is planet number 1 from the sun.
        </P>
        <P>
            Venus is planet number 2 from the sun.
        </P>
        <P>
            Earth is planet number 3 from the sun.
        </P>
        <HR>
    </60DY>
</HTML>
```

Рассмотрим еще один пример — преобразование XML-XML, в котором перечисляются планеты из planets.xml. Однако я хочу, чтобы выводилось не просто «The first three planets are: Mercury Venus Earth» (первые три планеты: Меркурий Венера Земля), а «The first three planets are: Mercury, Venus, and Earth». Необходимые знаки пунктуации можно добавить, определяя текущий элемент при помощи функции position и проверяя позицию при помощи <xsl:if> (листинг 5.2).

Листинг 5.2. Второй пример применения <xsl:if>

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">
<xsl:output method-"xml"/>
<xsl .'template match-"PLANETS">
<DOCUMENT>
    <TITLE>
        The Planets
    </TITLE>
    <PLANETS>
        The first three planets are: <xsl:apply-templates select-"PLANET"/>
    </PLANETS>
</DOCOMENT>
</xsl:tempiate>

<xsl:template match="PLANET">
    <xsl:value-of select-"NAME"/>
    <xsl:1f test-"position()!-last()">, </xsl:if>
    <xsl:i f te3t«"position()-last()-l">and </xsl:if>
    <xsl:if test^pos1tion()»last()">.</xsl:1f>
</xsl:template»

</xsl;stylesheet»
```

Вот результат:

```
<?xml versi on—" 1.0" encoding="UTF-8"?>
<DOCUMENT>

    <TITLE>
        The Planets
    </TITLE>

    <PLANETS>
        The first three planets are: Mercury. Venus, and Earth.
    </PLANETS>

</DOCUMENT>
```

Как видите, я смог добавить правильные знаки пунктуации, определяя место в документе при помощи <xsl:if>.

При помощи <xsl:if> можно также обнаруживать ошибки во время преобразования. Например, при помощи <xsl:if> можно вывести сообщение, есть ли в planets.xmt элемент <NAME> (листинг 5.3).

Листинг 5.3. Обнаружение ошибок при помощи <xsl:if>

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"»

<xsl:output method - "xml"/>
<xs1:tempiate match-"PLANETS">
<DOCUMENT>
    <TITLE>
        The Planets
    </TITLE>
    <PLANETS>
        The first three planets are: <xsl:apply-templates select-"PLANET"/>
    </PLANETS>
</DOCUMENT>
</xsl:template»

<xsl:template match-"PLANET">
    <xsl:1f test-"NAME[not(text())]">
        <xsl:message terminate-"yes">
            Each planet must have a name!
        </xsl:message»
    </xsl:if»
    <xsl:value-of select-"NAME"/»
    <xsl:if test-"position()!-last()">. </xsl:if>
    <xsl:if test-"posit-ton()»last()-l">and </xsl .if»
    <xsl:if test-"position()-last()"».</xsl:if»
</xsl:tempiate»

</xsl:stylesheet»
```

Чтобы проверить работу этого кода, я задал один из элементов <NAME> пустым:

```
<?xml version-"1.0"?>
<?xml-stylesheet type-"text/xml" href-"planets.xsl"?»
<PLANETS>
```

```
<PLANET»
    <NAME»Mercury</NAME>
    <MASS UNITS-"(Earth - 1)">.0553</MASS>
    <DAY UNITS-"days">58.65</DAY>
    <RADIUS UNITS-"miles"»1516</RADIUS>
    <DENSITY UNITS-"(Earth - 1)"> 983</DENSITY»
    <DISTANCE UNITS-"million miles">43.4</DISTANCE><!--B перигелии-->
</PLANET»

<PLANET>
    <NAME></NAME>
    <MASS UNITS-"(Earth - 1)"».815</MASS»
    <DAY UNITS-"days"»116.75</DAY»
    <RADIUS UNITS-"miles"»3716</RADIUS>
    «DENSITY UNITS-"(Earth - 1)"».943</DENSITY»
    <DISTANCE UNIT$-"miVlion miles"»66.8</DISTANCE»<!--B перигелии-->
</PLANET»
```

Вот как происходит обработка примера в Xalan:

```
C:\planets»java org.apache.xalan.xslt.Process -IN planets.xml -XSL errors.xsl -OUT new.xml
file:///C:/XSL/w.xsl: Line 18: Column 38: Each planet must have a name!

XSLT Error (javax.xml.transform.TransformerException): Stylesheet directed termination
```

Если вы знакомы с конструкцией if в языках программирования, вы знаете, что инструкция if обычно сопровождается инструкцией else, которая выполняется при ложности условия в if. Но в XSLT нет элемента <xsl:else>. Для задания альтернативных ветвей выполнения XSLT служит элемент <xsl:choose>.

Элементы <xsl:choose>, <xsl:when> и <xsl:otherwise>

Элемент <xsl:choose> похож на оператор Java switch, который позволяет сравнивать значение условия с несколькими возможными вариантами.

У элемента <xsl:choose> нет атрибутов. Он содержит один или более элементов <xsl:when> и (не обязательно) один элемент <xsl:otherwise>, который в случае применения должен стоять последним.

Вот как это работает: в элемент <xsl:choose> заключаются элементы <xsl:when>, каждый с условием true/false. Применяется тело шаблона в первом элементе <xsl:when>, чье условие имеет значение true, все остальные не применяются. Последним элементом внутри элемента <xsl:choose> может быть элемент <xsl:otherwise> тело шаблона внутри этого элемента применяется, если ни одно из предыдущих условий <xsl:when> не имело значения true:

```
<xs1:choose»
    <xsl:when test-"expressionl">
        <!--template-body 1-->
    </xsl:when>
```

```
    <xsl:when test="expression">
        <! - - tempi ate - body 2--»
    «/xsl:when>
    <xsl:when test-"expression3">
        <!--template-body 3-->
    </xsl:when»
    <xsl:otherwise»
        <!--template body 4-->
    </xsl:otherwise»
</xsl:choose»
```

В предыдущем разделе для осуществления этого преобразования нам потребовалось три элемента <xsl:if>:

```
<?xml version-"1.0"?>
<xs1:stylesheet vers i on-"1.0"
xmlns:xsl-"http://www w3 org/1999/XSL/Transform">
<xsl:output method-"xml/»
<xsl:tempi ate match*"PLANETS">
«DOCUMENT»
    <TITLE>
        The Planets
    </TITLE>
    <PLANETS>
        The first three planets are: <xsl:apply-templates select-"PLANET"/»
    </PLANETS»
«/DOCUMENT»
</xsl:tempiate»

<xsl:template match-"PLANET"»
    <xsl:if test-"NAME[not(text())]"»
        <xsl:message terminate-"yes">
            Each planet must have a name!
        </xsl:message»
    </xsl:if>
    <xsl:value-of select-"NAME"/»
    <xsl:if test-"pos1t1on()!-last()"». </xsl:1f»
    <xsl :if test-"pos1tion()«last()-r>and </xsl:if»
    <xsl:if test»"position()-last()"».</xsl:if»
</xsl:tempi ate»

</xsl;stylesheet»
```

Теперь то же самое можно сделать при помощи единственного элемента <xsl:choose>:

```
<?xml version-"1.0"?»
«xsl;stylesheet version-"1.0"
xmlns:xsl-"http://www w3.org/1999/XSL/Transform"»
<xml:output method-"xml"/»
<xsl:output method*"xml"/>

<xsltemplate match-"PLANETS"»
«DOCUMENT»
    «TITLE»
        The Planets
    «/TITLE»
    «PLANETS»
```

```
        The first three planets are: <xsl:apply-templates select-"PLANET'7>
    </PLANETS>
</DOCUMENT>
</xsl:template>

<xsl:tempi ate match«"PLANET">
    <xsl:if test-"NAME[not(text())]">
        <xsl:message terminate-"yes">
            Each planet must have a name!
        </xsl:message>
    </xsl:if>
    <xsl:value-of select-"NAME"/>
    <xsl:choose>
        :
        :          f
        .
    </xsl:choose>
</xsl:tempiate>

</xsl:stylesheet>
```

Нам нужно проверить, в каком месте документа мы находимся, при помощи включения нескольких элементов <xsl:when>. У этого элемента только один атрибут:

- test (обязательный). Принимает логическое (Boolean) значение (true/false) проверяемого условия.

Элемент <xsl:when> содержит тело шаблона.

Атрибут проверки принимает значение true/false выражения, определяющего, будет ли применяться заключенное в элементе <xsl:when> тело шаблона или нет. Например, вот как я добавил элементы <xsl:when> с соответствующими знаками пунктуации для всех планет, кроме последней:

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"I 0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">
<xsl:output method-"xmr/>
<xsl:tempi ate match-"PLANETS">
<DOCUMENT>
    <TITLE>
        The Planets
    </TITLE>
    <PLANETS>
        The first three planets are: <xsl apply-templates select="PLANET7>
    </PLANETS>
</DOCUMENT>
</xsl:template»

<xsltemplate match-"PLANET">
    <xsl if test-"NAME[not(text())]">
        <xsl:message terminate-"yes">
            Each planet must have a name!
        </xsl message>
    </xsl:1f>
    <xsl:value-of select-"NAME"/>
    <xsl:choose>
        <xsl:when test-"posit1on()Mast()">, </xsl:when>
```

```
            <xs1:when test-"pos1tion()-last()-l">and </xsl:when>
            .
            .
            .
        </xsl:choose>
    </xsl tempiate>

    </xsl stylesheet>
```

Эти два элемента <xsl:when> выбирают все элементы <PLANET> за исключением последнего, поэтому элемент <xsl:otherwise> можно применить для последнего элемента <PLANET>. Тело шаблона в этом элементе будет применено, если ни в одном элементе <xsl:when> в элементе <xsl:choose> условие не примет значение true.

У элемента <xsl:otherwise> нет атрибутов, и он содержит тело шаблона. Вот как это выглядит в листинге 5.4.

Листинг 5.4. Применение <xsl:choose>

```
<?xml version-"1 0"?>
<xsl:stylesheet version-"1.0"
xmlns:xs1-"http://www.w3.org/1999/XSL/Trans form">
<xsl:output method«"xml"/>

<xsl:template match-"PLANETS">
<DOCUMENT>
    <TITLE>
        The Planets
    </TITLE>
    <PLANETS>
        The first three planets are: <xsl:apply-templates select-hPLANET"/>
    </PLANETS>
</DOCUMENT>
</xsl:tempiate>

<xsl -.template match-"PLANET">
    <xsl:if test-"NAMEtnot{text())]">
        <xsl:message term1nate-"yes">
            Each planet must have a name!
        </xsl:message>
    </xsl:if>
    <xsl:value-of select-"NAME"/>
    <xsl:choose>
        <xsl:when test-"position()!-last()">. </xsl:when>
        <xsl:when test-"pos1tion()-last()-l">and </xsl:when>
        <xsl:otherwise>.</xsl:otherwise»
    </xsl:choose>
</xsl:tempiate>

</xsl:stylesheet>
```

Вот как это работает; этот код дает тот же результат, что и код, проверяющий позицию элементов <PLANET> при помощи <xsl:if>:

```
<?xml version-"1.0" encoding-"UTF-8"?>
<DOCUMENT>

    <TITLE>
        The Planets
```

```
    </TITLE>

    <PLANETS>
      The first three planets are: Mercury. Venus, and Earth.
    </PLANETS>

</DOCUMENT>
```

Вот еще один пример преобразования XML-XML. В этом случае я преобразую planets.xml в новый XML-документ, сохраняя только название каждой планеты и добавляя описание:

```
<?xml version-"!.O" encoding-"UTF-8"?>
<DOCUMENT>

    <TITLE>
        The Planets
    </TITLE>

    <PLANETS>
        <PLANET>
            <NAME>Me rcu ry</NAME>
            <DESCRIPTION>Hottest</DESCRIPTION>
        </PLANET>
        <PLANET>
            <NAME>Venus</NAME>
            <DESCRIPTION>Hot</DESCRIPTION>
        </PLANET>
        <PLANET>
            <NAME>Ea rth</NAME>
            <DESCRIPTION>OK</DESCRIPTION>
        </PLANET>
    </PLANETS*

</DOCUMENT>
```

Это преобразование можно реализовать, выбирая значение каждого элемента <NAME>, то есть заключенный в нем текст (заметьте, что такого рода строки в XSLT учитывают регистр) (листинг 5.5).

Листинг 5.5. Второй пример <xsl:choose>

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.O"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

<xsl:output method-"xml"/>
<xsltemplate match-"PLANETS">
<DOCUMENT>
    <TITLE>
        The Planets
    </TITLE>
    <PLANETS>
        <xsl:apply-templates select-"PLANET"/>
    </PLANETS>
</DOCUMENT>
</xs1:template»
<xsl.template match-"PLANET">
```

продолжение &

Листинг 5.5 (продолжение)

```
    <xsl:if test-"NAME[not(text())]">
        <xsl:message terminate-"yes">
            Each planet must have a name!
        </xsl:message>
    </xsl:if>
    <PLANET>
        <NAME>
            <xsl:value-of select-"NAME"/>
        «/NAME*
        «DESCRIPTION*
            «xsl:choose*
                <xsl -.when test-"NAME-'Mercury' ">Hottest</xsl :when>
                <xsl:when test-"NAME-'Venus"'>Hot</xsl:when>
                <xsl -.when test-"NAME«'Earth' "*0K«/xsl :when>
            </xsl:choose>
        «/DESCRIPTION*
    </PLANET*
</xsl:tempiate*

</xsl:stylesheet*
```

Вот и все.

Предположим теперь, что нам нужно добавить в каждый элемент <PLANET> атрибут COLOR:

```
<?xml version-"1.0"?>
<?xml-stylesheet type-"text/xml" href-"planets.xsl"?*
«PLANETS*

  «PLANET COLOR--RED"*
    <NAME*Mercury«/NAME>
    «MASS UNITS-"(Earth - 1)"* 0553«/MASS*
    «DAY UNITS-Hdays"*58.65«/DAY*
    «RADIUS UNITS-"miles"*1516«/RADIUS*
    «DENSITY UNITS-"(Earth - 1)"*.983«/DENSITY*
    «DISTANCE UNITS-"mill1on miles">43.4«/DISTANCE><!--B перигелии--*
  «/PLANET*

  «PLANET COLOR-"WHITE">
    <NAME*Venus«/NAME*
    «MASS UNITS-"(Earth - 1)"*.815«/MASS*
    «DAY UNITS-"days"*116.75«/DAY*
    «RADIUS UNITS-"mi1es">3716«/RADIUS*
    «DENSITY UNITS-"(Earth - 1)">.943«/DENSITY>
    «DISTANCE UNITS-"million miles"*66.8«/DISTANCE*<!--B перигелии-*
  «/PLANET*

  «PLANET COLOR-'BLUE"*
    «NAME*Ea rth«/NAME*
    «MASS UNITS-"(Earth - 1)"*1</MASS*
    «DAY UNITS-"days"*l«/DAY*
    «RADIUS UNITS-"miles"*2107«/RADIUS*
    «DENSITY UNITS-"(Earth - 1)"*1</DENSITY*
    «DISTANCE UNITS-"million miles">128.4«/DISTANCE><!--B перигелии-->
  </PLANET>

</PLANETS>
```

Отобразить названия различных планет при помощи элемента <xsl:choose>, отформатированные по-разному при помощи тегов HTML <B>, <I> и <U> в зависимости от значения атрибута COLOR, можно следующим образом (листинг 5.6).

Листинг 5.6. Форматирование при помощи <xsl:choose>

```
<?xml version-"1.0"?>
<xs1:stylesheet versi on-"1.0"
xmlns:xsl-"http: //ww.w3 .Org/1999/XSL/Transform">

<xsl:tempiate match-"PLANETS">
    <HTML>
        <HEAD>
            <TITLE>
                Planets
            </TITLE>
        </HEAD>
        <BODY>
            <xsl:apply-templates select-"PLANET"/>
        </BODY>
    «/HTML*
</xsl:tempiate>

<xsl:template match-"PLANET"*
    <xsl;choose*
        «xsl:when test-"@C0L0R - 'RED'">
            <B>
                <xsl:value-of select-"NAME"/>
            </B>
        </xsl:when>
        <xsl:when test-"@C0L0R - 'WHITE'">
            <I>
                <xsl:value-of select«"NAME7>
            </I>
        </xsl:when>
        <xsl:when test-"0COLOR - 'BLUE'"»
            <U>
                <xsl:value-of se1ect-"NAME"/>
            «/U*
        </xsl:when>
        <xsl:otherwise*
            <PRE>
                <xsl:value-of select-"."/>
            </PRE>
        </xsl:otherwise*
    </xsl:choose*
</xsl:tempi ate*

</xsl:stylesheet*
```

Вот результирующий документ:

```
<HTML*

    <HEAD*
        <TITLE>
            Planets
        </TITLE>
    «/HEAD*
```

```
    <BODY>
        <B>Mercury</B>
        <I>Venus</I>
        <U>Earth</U>
    </BODY>

</HTML>
```

Как вы видели, при помощи <xsl:if> можно проверять единственное условие, а при помощи <xsl:choose> — несколько; аналогичные конструкции присутствуют в большинстве языков программирования. Кроме подобных этим условных операторов, в большей части языков программирования существуют также операторы цикла, и в XSLT содержится нечто похожее — элемент <xsl:for-each>.

Элемент <xsl:for-each>

Элемент <xsl:for-each> позволяет применять тело шаблона в цикле снова и снова для всех элементов набора узлов. С технической точки зрения, он работает с набором узлов, который возвращает выражение XPath и выполняет одно и то же действие с каждым узлом в наборе. При каждом шаге цикла тело шаблона применяется к следующему узлу из набора узлов, что дает возможность легко обрабатывать несколько узлов.

<XSL:FOR-EACH> ПРОТИВ <XSL:APPLY-TEMPLATES>

Вы могли заметить, что это описание практически такое же, как и у элемента <xsl:apply-templates>, и я сравню элементы <xsl:for-each> и <xsl:apply-templates> через несколько страниц.

У элемента <xsl:for-each> один атрибут:

- select (обязательный). Принимает значение выражения XPath, возвращающее набор узлов, который нужно обработать в цикле.

Элемент может содержать ноль или более элементов <xsl:sort>, за которыми следует тело шаблона. Работу с элементом <xsl:sort> мы изучим позже в этой главе.

В теле шаблона функция position возвращает позицию текущего узла в наборе узлов, а last возвращает число узлов в наборе. Если <xsl:sort> не используется, узлы обрабатываются в порядке документа (в порядке, в котором они перечислены в документе); если же используется элемент <xsl:sort>, набор узлов будет сначала отсортирован в порядке, заданном этим элементом.

Предположим, нам нужно отформатировать все названия планет, заключив их в элементы HTML <P>, — это можно сделать следующим образом:

```
<xsltemplate match-"PLANET">
    <P>
        <xsl:va1ue-of select-"NAME'7>
    </P>
</xsl:tempiate>
```

Но что делать, если у некоторых планет по два названия, как, например:

```
<PLANET>
        <NAME>Mercury</NAME>
        <NAME>Closest planet to the sun</NAME>
```

```
        <MASS UNITS-"(Earth - 1)">.0553</MASS>
        <DAY UNITS-"days">58.65</DAY>
        <RADIUS UNITS-"miles">1516</RADIUS>
        <DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
        <D I STANCE UNITS-"million miles">43.4</DISTANCEx!--В перигелии-->
    </PLANET>
```

Это проблема, поскольку атрибут select элемента <xsl:value-of> сам по себе выберет только первый элемент <NAME>. Чтобы пройти в цикле все возможные варианты, вместо него следует применить элемент <xsl:for-each> (листинг 5.7).

Листинг 5.7. Применение <xsl:for-each>

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transforni">

    «xsl:template match-"PLANETS">
        <HTML>
            <xsl:apply-templates/>
        </HTML>
    </xsl:tempiate>

<xsltemplate match="PLANET">
    <xsl:for-each select-"NAME">
        <P>
            <xsl:value-of select-"."/>
        </P>
    </xsl:for-each>
</xsl:tempiate>

</xsl:stylesheet>
```

Эта таблица стилей охватывает все элементы <NAME>, помещает их значения в элемент <P> и добавляет их в выходной документ следующим образом:

```
<HTML>

    <P>Mercury</P>
    <P>Closest planet to the sun</P>
    <P>Venus</P>
    <P>Earth</P>

</HTML>
```

Вот еще один пример, впервые появившийся в главе 3, «Создание и применение шаблонов», где при помощи элемента <xsl:for-each> fe цикле перебирались все атрибуты элемента:

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http.7/www.w3.org/1999/XSL/Transform">
    <xsl:output method-"xml"/>
    <xsl:tempi ate match-"*">
        <xs1:copy>
            <xsl:for-each select-"!?*"»
                <xs1:copy/>
            </xsl:for-each>
            <xsl:apply-templates/»
```

```
        </xsi:copy>
    </xsl:template»
</xsl stylesheet»
```

Следующий пример появился в главе 2, «Создание и применение таблиц стилей». Это упрощенная таблица стилей, в которой нельзя использовать какие-либо элементы высокого уровня, то есть нельзя использовать <xsl:temptate> или <xsl:apply-templates>, однако можно пройти по узлам в цикле при помощи <xsl:for-each>:

```
<HIML xmlns xsl-"http://www.w3 org/1999/XSL/Transform" xsl:version-"1 0"»
    <HEAD>
        <TITLE>
            The Planets Table
        </TITLE>
    </HEAD>
    <BODY>
        <Hl>
            The Planets Table
        </Hl>
        <TABLE BORDER-"2">
            <TR>
                <TD>Name</TD>
                <TD>Mass</TD>
                <TD>Radius</TD>
                <TD>Day</TD>
            </TR>
            <xsl:for-each select-"//PLANET">
                <TR>
                    «TDxxsl rvalue-of sel ect-"NAME "/></TD>
                    <TD><xsl:value-of select-"MASS"/></TD>
                    <TD><xs1:value-of select-"RADIUS"/></TD>
                    <TD><xsl:value-of select-"DAY"/x/TD>
                </TR>
            </xsl:for-feach>
        </TABLE>
    </BODY>
</HIML>
```

Эта упрощенная таблица стилей форматирует planets.xml в planets.html практически так же хорошо, как и шаблон, использующий <xsl:apply-templates>, в связи с чем появляется интересный вопрос: когда следует для прохода по узлам применять <xsl:for-each>, а когда <xsl:apply-templates>?

Как правило, <xsl:apply-templates> хорошо применять в тех случаях, когда организация дочерних узлов неизвестна, и вы хотите применить различные шаблоны к потомкам разных видов — независимо от количества уровней, на которые углубляется их структура. С другой стороны, если дочерние узлы обладают регулярной, хорошо определенной организацией, можно задать <xsl:for-each> для обработки всех этих узлов.

Элемент <xsl:for-each> работает во многом так же, как и <xsl:apply-templates>; можно даже вкладывать шаблоны при помощи <xsl:for-each>, как это делается при помощи последовательных элементов <xsl:apply-templates>. В листинге 5.8 я прохожу в цикле по каждому элементу <PLANET>, а затем во вложенном в него цикле по всем элементам, содержащимся в элементе <PLANET>, перечисляя их данные из элементов <DATA> следующим образом.

Листинг 5.8. Второй пример <xsl:for-each>

```
<?xml version-"1.0"?>
<xst:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Trans form">
    <xsl:output method-"xml"/>
    <xsltemplate match-"PLANETS">
        <PLANETS>
            <xsl:for-each select-"PLANET">
                <PLANET>
                    <xsl:for-each select-"*"»
                        <DATA>
                            <xsl:value-of select-"."/>
                        </DATA>
                    </xsl:for-each>
                </PLANET>
            </xsl:for-each>
        </PLANETS>
    </xsl -.template»
</xsl:stylesheet»
И вот результат:
<?xml version-"1.0" encod1ng-"UTF-8"?>
<PLANETS>
    <PLANET>
        <DATA>Mercury</DATA>
        <DATA>.0553</DATA>
        <DATA>58.65</DATA>
        <DATA>1516</DATA>
        <DATA>. 983</DATA>
        <DATA>434</DATA>
    </PLANET>
    <PLANET>
        <DATA>Venus</DATA>
        <DATA> 815</DATA>
        <DATA>116.75</DATA>
        <DATA>3716</DATA>
        <DATA>. 943</DATA>
        OATA>66. 8</DATA>
    </PLANET>
    <PLANET>
        <DATA>Earth</DATA>
        <DATA>1</DATA>
        <DATA>1</DATA>
        <DATA>2107</DATA>
        <DATA>1</DATA>
        <DATA>1284</DATA>
    </PLANET>
</PLANETS>
```

Сортирующие элементы

При помощи элемента <xsl:sort> можно сортировать узлы. Этот элемент устанавливает порядок обработки узлов для <xsl:apply-templates> и <xst:for-each>. В следующем списке перечислены атрибуты <xsl:sort>:

- select (необязательный). Принимает значение выражения XPath, возвращающего набор узлов для сортировки. По умолчанию — *string(.)»;

- order (необязательный). Задает порядок сортировки, устанавливается в «ascending» (по возрастанию) или «descending» (по убыванию);
- case-order (необязательный). Определяет, будут ли буквы в верхнем регистре располагаться перед буквами в нижнем регистре. Устанавливается в «upper-first» (сначала верхний) или «lower-first» (сначала нижний);
- lang (необязательный). Задает язык, чьи соглашения о сортировке будут применяться. Устанавливается в код языка, допустимый в атрибуте xml:lang;
- data-type (необязательный). Определяет, будет ли сортировка вестись в алфавитном или числовом порядке. Устанавливается в «text» (текст), «number» (число) или в QName.

Этот элемент не включает содержимое. Его следует применять внутри элементов <xsl:apply-templates> или <xsl:for-each> для сортировки наборов узлов, с которыми работают эти элементы.

В листинге 5.9 я только отсортирую элементы <PLANET> в planets.xml в возрастающем алфавитном порядке по их именам, используя <xsl:for-each> в упрощенной таблице стилей.

Листинг 5.9. Сортировка данных

```
<HTML xmlns:xsl-"http://www.w3.org/1999/XSL/Transform" xsl:version-"1.0">
    <HEAD>
        <TITLE>
            The Sorted Planets Table
        </TITLE>
    </HEAD>
    <B0DY>
        <H1>
            The Sorted Planets Table
        </H1>
        <TABLE BORDER-"2">
            <TR>
                <TD>Name</TD>
                <TD>Mass</TD>
                <TD>Radius</TD>
                <TD>Day</TD>
            </TR>
            <xsl:fOr-each select-"//PLANET">
                <xsl:sort/>
                <TR>
                    <TDxxsl:value-of select-"NAME"/></TD>
                    <TDxxsl:value-of select-"MASS"/></TD>
                    <TDxxsl: val ue-of sel ect="RADIUS7></TD>
                    <TDxxsl: val ue-of select-"DAY"/x/TD>
                </TR>
            </xsl:for-each>
        </TABLE>
    </BODY>
</HTML>
```

А вот результат. Обратите внимание на то, что планеты действительно отсортированы как Earth, Mercury и затем Venus:

```
<HTML>
    <HEAD>
```

```
        <TITLE>
          The Sorted Planets Table
        </TITLE>
      </HEAD>

      <BODY>
        <H1>
          The Sorted Planets Table
        </H1>

        <TABLE BORDER-"2">
          <TR>
            <TD>Name</TD>
            <TD>Mass</TD>
            <TD>Radius</TD>
            <TD>Day</TD>
          </TR>
          <TR>
            <TD>Earth</TO>
            <TD>1</TD>
            <TD>2107</TD>
            <TD>1</TD>
          </TR>
          <TR>
            <TD>Mercury</TD>
            <TD>.0553</TD>
            <TD>1516</TD>
            <TD>58.65</TD>
          </TR>
          <TR>
            <TD>Venus</TD>
            <TD>.815</TD>
            <TD>3716</TD>
            <TD>116.75</TD>
          </TR>
        </TABLE>
      </BODY>
    </HTML>
```

Вид документа показан на рис. 5.1.

При помощи атрибута select можно указать, что нужно сортировать. Например, таким образом можно отсортировать планеты по плотности (листинг 5.10).

Листинг 5.10. Сортировка планет по плотности

```
<?xml version-"1.0"?>
<xsl:stylesheet vers1on-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

  <xsl:tempi ate match-"PLANETS">
    <HTML>
      <HEAD>
        <TITLE>
          Planets
        </TITLE>
      </HEAD>
      <BODY>
        <H1>P1anets sorted by density</Hl>
```

продолжение &

Листинг 5.10 (продолжение)

```
                <TABLE>
                    <TR>
                        <TD>Planet</TD>
                        <TD>Mass</TD>
                        <TD>Day</TD>
                        <TD>Density</TD>
                    </TR>
                    <xsl:apply-tempiates>
                        <xsl:sort select®"DENSITY"/>
                    </xsl:apply-tempiates>
                </TABLE>
            </BODY>
        </HTML>
    </xsl:template»

    <xsl:tempi ate match«"PLANET">
        <TR>
            <TD><xsl:apply-templates select-"NAME"/></TD>
            <TD><xsl apply-templates select-"MASS"/></TD>
            <TD><xsl:apply-templates select-"DAY"/></TD>
            <TD><xsl:apply-tempiates select«"DENSITY"/></TD>
        </TR>
    </xsl tempiate>
</xsl:stylesheet>
```

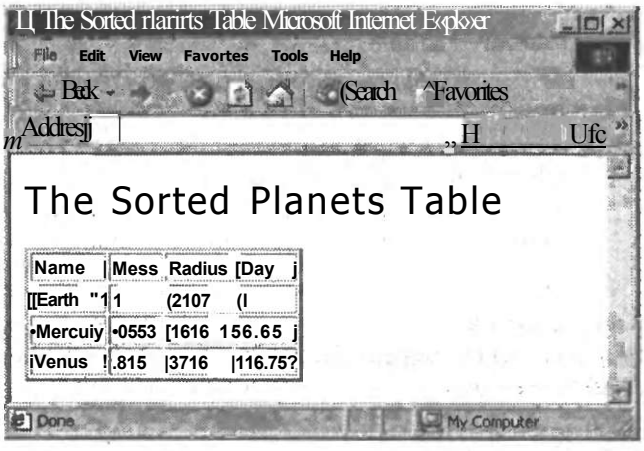


Рис. 5.1. Сортировка при помощи упрощенного шаблона

Вот результаты этого преобразования:

```
<HTHL>
    <HEAD>
        <TITLE>
            Planets
        </TITLE>
    </HEAD>

    <BODY>
        <H1>
            Planets sorted by density
```

```
        </H1>
        <TABLE>
            </TR>
                <TD>Planet</TD>
                <TD>Mass</TD>
                <TD>Day</TD>
                <TD>Density</TD>
            </TR>

            <TR>
                <TD>Venus</TD>
                <TD>.815</TD>
                <TD>116.75</TD>
                <TD>.943</TD>
            </TR>

            <TR>
                <TD>Mercury</TD>
                <TD> 0553</TD>
                <TD>58.65</TD>
                <TD>.983</TD>
            </TR>

            <TR>
                <TD>Earth</TD>
                <TD>1</TD>
                <TD>1</TD>
                <TD>1</TD>
            </TR>
        </TABLE>
    </BODY>
</HTML>
```

По умолчанию <xsl:sort> производит сортировку в алфавитном порядке, что означает, что «10» располагается перед «2». Для осуществления сортировки в числовом порядке установите атрибут типа данных в «number»:

```
<xsl:sort data-type«"number" select-^DENSITY"/»
```

Убывающий порядок сортировки задается установкой у элемента <xsl:sort> атрибута order в «descending». Можно также сортировать по значениям атрибута, например:

```
<xs1: apply-tempi ates select-"PLANETS>
    <xsl:sort select-"@SIZE"/>
</xsl:apply-templates>
```

НОВОЕ В XSLT 2.0

Одним из важных преимуществ XSLT 2.0 является поддержка схем XML, и W3C планирует внести в XSLT 2.0 возможность сортировки по любому типу данных, определенному в схеме документа, так же, как сейчас можно сортировать по строкам или числам.

Сортировка по нескольким критериям

Стоит отметить, что при сортировке можно применять несколько критериев — для этого просто примените несколько элементов <xsl:sort>. Первый элемент <xsl:sort>

отсортирует данные по основному критерию, следующий элемент — по следующему основному критерию и т. д. Вот, например, как внутри элемента <xsl:apply-templates> можно отсортировать планеты сначала по расстоянию, а затем по плотности (при этом планеты с одинаковым расстоянием от Солнца будут отсортированы по плотности):

```
<xs1:apply-tempiates>
    <xsl:sort  select-"DISTANCE"/>
    <xsl:sort  select*="DENSITY"/>
</xsl:apply-templates»
```

На этом мы заканчиваем обсуждение сортировки, и я перехожу к сопутствующей теме: нумерации.

Элемент <xsl:number>

При помощи элемента <xsl:number> узлу в результирующем документе присваивается очередной номер. Например, можно перенумеровать абзацы в контракте или строфы в поэме. Можно даже перенумеровать части документа несколькими уровнями, как, например, «Paragraph 3.2.5.1.» и т. п.

Элемент <xsl:number> обладает следующими атрибутами:

- level (необязательный). Определяет, как будут присваиваться последовательные числа. Устанавливается в «single» (один), «multiple» (несколько) или «апу» (любой). Значение по умолчанию — «single»;
- count (необязательный). Определяет, какие узлы нужно подсчитывать. Устанавливается в образец;
- from (необязательный). Определяет точку начала отсчета. Устанавливается в образец;
- value (необязательный). Форматируемое число;
- format (необязательный). Определяет формат вывода. Устанавливается в шаблон значений атрибута, возвращающий строку форматирования;
- Lang (необязательный). Определяет язык, чьи соглашения следует использовать для нумерации. Устанавливается в код языка, который можно применять в атрибуте xmlilang;
- letter-value (необязательный). Позволяет выбрать различные схемы нумерации. устанавливается в «alphabetical» (алфавитная) или «traditional» (обычная);
- grouping-separator (необязательный). Символ для разделения групп разрядов — например, запятая. Устанавливается в шаблон значений атрибутов, возвращающий единственный символ;
- grouping-size (необязательный). Количество разрядов в каждой группе — определяет место применения разделителя групп разрядов. Устанавливается в шаблон значений атрибутов, возвращающий число.

СОВЕТ ПО НУМЕРАЦИИ

Как можно видеть из этого списка атрибутов, существует весьма много возможных схем нумерации. Операции нумерации могут стать довольно сложными, поэтому есть один прием: если нумерация будет становиться слишком сложной и запутанной, я просто выведу результирующий документ без нумерации и затем воспользуюсь второй таблицей стилей, которая применит нумерацию.

Существует три основных способа нумерации, в зависимости от установки атрибута уровня (level): «single», «multiple» или «any». В следующих разделах мы по очереди рассмотрим каждую из этих схем, начав с одноуровневой нумерации, которая установлена по умолчанию.

Одноуровневая нумерация

Одноуровневая нумерация — это простая нумерация, когда перенумеровываются узлы-братья на одном уровне. Этот тип нумерации установлен по умолчанию. В листинге 5.11 при помощи одноуровневой нумерации перенумеровываются планеты в planets.xml.

Листинг 5.11. Одноуровневая нумерация

```
<?xml version-"1.0M?>
<xsl:stylesheet version-"1.0"
xmlns:xs1-"http://www.w3.org/1999/XSL/Transformi">

    <xsl template match-"/PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <BODY>
                <H1>
                    The Planets Table
                </H1>
                <TABLE BORDER-"2">
                    <TR>
                        <TD>Name</TD>
                        <TD>Mass</TD>
                        <TD>Radius</TD>
                        <TD>Day</TD>
                    </TR>
                    <xsl:apply-tempiates/>
                </TABLE>
            </BODY>
        </HTML>
    </xsl:tempiate>

    <xsl:template match-"PLANET">
        <TR>
            <TD><xsl:number/>. <xsl:value-of select-"NAME"/></TD>
            <TDxxs1 ;apply-tempi ates sel ect-"MASS" /x/TD>
            <TD><xsl apply-tempiates select-"RADIUS"/></TD>
```

продолжение &

Листинг 5.11 (продолжение)

```
            <TD><xs1:apply-templates select-"DAY"/»</TD>
        </TR>
    </xsl:template»

    <xsl:template match-"MASS">
        <xs1:value-of select-"."/»
        <xsl:text> </xsl:text»
        <xsl :value-of select-'GUNITS"/»
    </xslrtemplate»

        .
        .
        .

    <xsl:tempi ate match-"DAY">
        <xsl:value-of select-"."/»
        <xsl:text» </xsl:text»
        <xsl:value-of select-"@UNITS"/>
    </xsl:tempi ate»

</xsl:stylesheet»
```

А вот результат:

```
<HTML»
    <HEAD>
        <TITLE>
            The Planets Table
         </TITLE>
    </HEAD>

    <BODY>
        <H1>
            The Planets Table
        </HI>

        <TABLE BORDER-"2"»
            <TR»
                <TD»Name</TD>
                <TD»Mass</TD>
                <TD»Radius</TD>
                <TD»Day</TD»
            </TR>

            <TR>
                <TD»1. Mercury</TD»
                <TD>.0553 (Earth - 1)</TD>
                <TD»1516 miles</TD>
                <TD»58.65 days</TD>
            </TR>

            <TR>
                <TD»2. Venus</TD»
                <TD».815 (Earth - 1)</TD>
                <TD»3716 miles</TD>
                <TD»116 75 days</TD>
            </TR>

            <TR>
```

```
            <TD>3. Earth</TD>
            <TD>1  (Earth  -  1)</TD>
            <TD>2107 miles</TD>
            <TD>1 days</TD>
        </TR>
    </TABLE>
</BODY>
</HTML>
```

Этот результат показан на рис. 5.2.

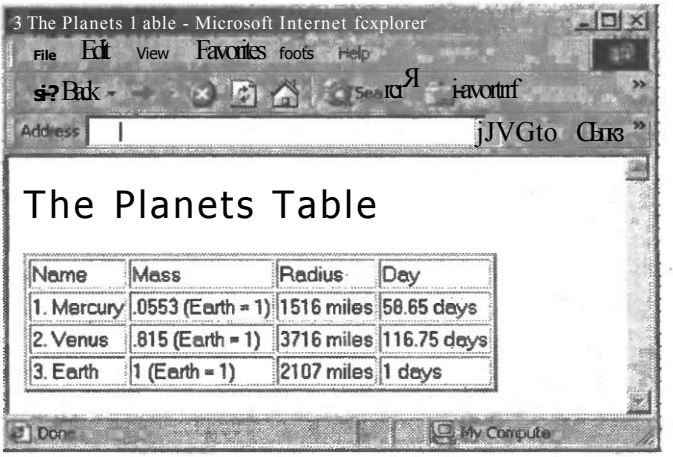


Рис. 5.2. Одноуровневая нумерация элементов

По умолчанию при нумерации используются числа, но есть и другие возможности. Например, если бы я использовал <xsl:number format="a"/>, планетам были бы присвоены буквы a, b и c:

```
<HTML>
    <HEAD>
        <TITLE>
            The Planets Table
        </TITLE>
    </HEAD>

    <BODY>
        <H1>
            The Planets Table
        </H1>

        <TABL E BORDER-"2">
            <TR>
                <TD>Name</TD>
                <TD>Mass</T0>
                <TD>Radi us</TD>
                <TD>Day</TD>
            </TR>

            <TR>
                <TD>a  Hercury</TD>
```

```
        <TD>.0553 (Earth - 1)</TD>
        <TD>1516 miles</TD>
        <TD>58.65 days</TD>
    </TR>

    <TR>
        <TD>b. Venus</TD>
        <TD>.815 (Earth - 1)</TD>
        <TD>3716 miles</TD>
        <TD>116.75 days</TD>
    </TR>
```

Ниже приведены возможные в атрибуте format лексемы и соответствующие им виды «нумерации:

- 1 порождает последовательность 1, 2, 3...;
- 01 порождает последовательность 01, 02, 03...;
- другие числа Unicode в других системах счисления порождают последовательности аналогично двум предыдущим правилам;
- а порождает последовательность a, b, c...aa, ab...;
- A порождает последовательность A, B, C...AA, AB...;
- i порождает последовательность i, ii, iii, iv...ix, x, xi, xii...;
- I порождает последовательность I, II, III, IV...IX, X, XI, XII...

Нумерация на произвольных уровнях

Иногда может потребоваться просто подсчитать узлы определенного типа независимо от их уровня в документе. Например, в различных местах иерархии документа не исключено наличие нескольких элементов <NAME>, и вам может быть нужно просто трактовать документ как поток данных, подсчитывая встречающиеся элементы <NAME>.

Вот как примерно это будет выглядеть, если в planets.xml элементы <NAME> встречаются на различных уровнях:

```
<?xml version-"1.0"?>
<?xml-stylesheet type-"text/xml"  href="planets.xsl"?>
<PLANETS>
    <TITLE>
        <NAME>Planets Table</NAME>
    </TITLE>
    <PLANET>
        <NAME>Mercury</NAME>
        <MASS UNITS-"(Earth - 1)">.0553</MASS>
        <DAY UNITS-"days">58.65</DAY>
        <RADIUS UNITS-"miles">1516</RADIUS>
        <DENSITY UNITS-"(Earth - 1)">.983</0ENSITY>
        <DISTANCE UNITS-"million miles">43.4</DISTANCE><!--В перигелии-->
    </PLANET>
```

```
    <PLANET>
        <NAME>Venus</NAME>
        <MASS UNITS-"(Earth - 1)">.815</MASS>
        <DAY UNITS-"days">116.75</DAY>
        <RADIUS UNITS-"miles">3716</RADIUS>
        OENSITY UNITS-"(Earth - 1)">.943</DENSITY>
        «DISTANCE UNITS-"m1llion ml1es">66.8</DISTANCEx!--B перигелии-->
    </PLANET>
        .
        .
        .
```

Чтобы сосчитать общее число элементов <NAME>, в таблице стилей можно установить атрибут level в «апу» (листинг 5.12).

Листинг 5.12. Нумерация на произвольных уровнях

```
<?xml vers1on-"l.0"?>
<xsl:stylesheet version-"1.0"
xml ns:xsl-"http://www.w3 org/1999/XSL/Transform">

    <xsl template match-"/PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <BODY>
                <H1>
                    The Planets Table
                </H1>
                <xsl:apply-tempiates select»"TITLE"/>
                <TABLE BORDER-"2">
                    <TR>
                        <TD>Name</TD>
                        <TD>Mass</TD>
                        <TD>Radius</TD>
                        <TD>Day</TD>
                    </TR>
                    <xsl:apply-tempiates select-"PLANET"/>
                </TABLE>
            </BODY>
        </HTML>
    </xsl -.tempi ate>

    <xslrtemplate match-"PLANET">
        <TR>
            <TD><xsl:apply-templates select-"NAME"/></TD>
            <TD><xsl:apply-templates select-"MASS"/></TD>
            <TD><xsl:apply-templates select-"RADIUS"/></TD>
            <TD><xsl:apply-templates select-"DAY"/></TD>
        </TR>
    </xsl:tempiate>

    <xs1rtemplate match-"TITLE">
        <xsl:apply-tempiates/>
    </xsl:tempiate>
```

продолжение тР

Листинг 5.12 (продолжение)

```
<xslrtemplate match-"NAME">
    <xsl:number level-"any" count-"NAME"/>. <xsl;value-of select-"."/>
</xsl:tempiate>

<xsl:tempi ate match-"MASS">
    <xslI:value-of select-"."/>
    <xsl:text> </xsl:text>
    <xsl:value-of select-"@UNITS"/>
</xsl:tempiate>

.
.
.

<xslrtemplate match-"DAY">
    <xsl:value-of select-"."/>
    <xsl:text> </xsl;text>
    <xsl:value-of select-"@UNITS"/>
</xsl:tempiate>

</xsl;stylesheet>
```

Получаем результат (заметьте, что текст каждого элемента <NAME>, независимо от его уровня в документе, пронумерован):

```
<HTML>
    <HEAD>
        <TITLE>
            The Planets Table
        </TITLE>
    </HEAD>

    <BODY>
        <H1>
            The Planets Table
        </HI>
        1, Planets Table
       <TABLE BORDER-"2">
            <TR>
                <TD>Name</TD>
                <TD>Mass</TD>
                <T0>Radi us</TD>
                <TD>Day</TD>
            </TR>

            <TR>
                <TD>2. Mercury</TD>
                <TD>.0553 (Earth - 1)</TD>
                <TD>1516 miles</TD>
                <TD>58.65 days</TD>
            </TR>

            <TR>
                <TD>3. Venus</TD>
                <TD>.815 (Earth - 1)</TD>
                <TD>3716 miles</TD>
                <TD>116.75 days</TD>
            </TR>
```

```
            <TR>
                <TD>4. Earth</TD>
                <TD>1 (Earth - 1)</TD>
                <TD>2107 miles</TD>
                <TD>1 days</TD>
            </TR>

        </TABLE>
    </BODY>
</HTML>
```

При помощи атрибута from можно указать, с какого узла-предка начинать отсчет; например, если установить узел-предок в элемент <PLANET> так:

```
<xsl-number level-"any" count-"NAME" from-"PLANET"/>
```

то процессор XSLT осуществит обратный просмотр только до первого предка <PLANET> и начнет нумерацию с этой точки документа.

Многоуровневая нумерация

Элемент <xsl:number> также поддерживает многоуровневую нумерацию — такую как 3.1.2.5 и т. п. Для работы с ней нужно установить атрибут level в «multiple». При помощи атрибута count можно указать, узлы какого типа вы хотите нумеровать, установив этот атрибут в образец, например: "PART|CHAPTER|PARAGRAPH". При обработке элементов <xsl:number> процессор XSLT нумерует узлы в соответствии с иерархией документа.

В примере я нумерую каждый уровень в иерархии элементов planets.xml, установив атрибут count в «*» для выбора всех элементов. Можно также указать формат нумерации при помощи атрибута format. При многоуровневой нумерации атрибут format задает формат для различных уровней, например «1.1.1.> задает нумерацию 1., 2.,... и т. д. для узлов верхнего уровня, 1.1., 1.2.,... и т. д. для узлов уровнем ниже и 1.2.1., 1.2.2., ... и т. д. для следующего уровня вниз. Вот как выглядит таблица стилей для этого примера в листинге 5.13.

Листинг 5.13. Многоуровневая нумерация

```
<?xml vers1on="l.0"?>
<xsl:stylesheet version»"1.0"
xmlns xsl-"http://www w3 org/1999/XSL/Transform">
<xsl: output method-"xmT7>

  <xsl template match-"node()">
    <xsl:copy>
      <xsl:number format-"1.1.1. " level-"multiple" count-"*"/>
      <xsl apply-templates select-"node()"/>
    </xsl:copy>
  </xsl:tempiate>

</xsl stylesheet>
```

Вот результат преобразования planets.xml в новый XML-документ, в котором перенумерованы все уровни элементов в соответствии с иерархией документа:

```
<?xml version»"1.0" encod1ng-"utf-8"?>
<?xml-stylesheet type-"text/xml" href="planets.xsl"?>
<PLANETS>1.

    <PLANET>1.1.
        <NAME>1.1.1. Mercury</NAME>
        <MASS>1.1.2. .0553</MASS>
        <DAY>1 1.3. 58.65</DAY>
        <RADIUS>1.1.4. 1516</RADIUS>
        <DENSITY>1.1.5. ,983</DENSITY>
        <DISTANCE>1.1.6. 43.4</DISTANCE><!—B перигелии—>
    </PLANET>

    <PLANET>1.2 -
        <NAME>1.2.1. Venus</NAME>
        <MASS>1.2.2. ,815</MASS>
        <DAY>1.2.3. 116.75</DAY>
        <RADIUS>1.2.4. 3716</RADIUS>
        <DENSITY>1.2.5. .943</DENSITY>
        <DISTANCE>1.2.6 66.8</DISTANCE><!—B перигелии—>
    </PLANET>

    <PLANET>1.3.
        <NAME>1.3.1. Earth</NAME>
        <MASS>1.3.2. 1</MASS>
        <0AY>1.3.3. 1</DAY>
        <RADIUS>1.3.4. 2107</RADIUS>
        <DENSITY>1.3.5. 1</DENSITY>
        <DISTANCE>1.3.6. 128.4</DISTANCE><!—B перигелии—>
    </PLANET>

</PLANETS>
```

На этом мы завершаем рассмотрение нумерации документов и переходим к последней теме этой главы — расширяемости XSLT.

Расширяемость XSLT

Несмотря на кажущуюся сложность XSLT, он во многих отношениях ограничен по сравнению с языками программирования, и в процессорах XSLT сразу же начали появляться расширения XSLT. Например, Saxon представил элемент <saxon:while>, реализуя в XSLT стандартный для программирования цикл while (до тех пор, пока). Xalan представил такие элементы, как <redirect:write>, для поддержки вывода нескольких документов. А процессор MSXML3 от Microsoft позволяет писать функции на языках таких сценариев, как JavaScript, и **Затем** вызывать их и выполнять их код.

Можно представить, с каким беспокойством на это смотрит W3C. Его работа, в принципе, заключается в стандартизации работы таких языков, как XSLT, но производители постоянно представляли свои собственные, нестандартные расширения в виде новых элементов и функций. С другой стороны, W3C не может предугадать все новые элементы и функции, поэтому консорциум начал рабо-

тать над стандартизацией способов включения функций расширения И элементов в XSLT. Расширения должны удовлетворять ряду общих правил:

- расширения должны использовать пространства имен во избежание конфликтов с элементами XSL;
- процессор XSLT должен быть в состоянии распознать применение расширения — и в случае ошибки расширения реагировать хорошо определенным способом;
- таблица стилей должна быть в состоянии проверить, доступно ли определенное расширение, и если нет, вернуться назад.

НОВОЕ В XSLT 2.0

Легко представить сложности W3C даже с этими общими правилами, и комитет XSLT 2.0 собирается исследовать возможность реализации всех расширений на «чистом» XSLT, вообще не прибегая к каким-либо внешним языкам программирования.

W3C разрешил расширения двух видов, главным образом, потому, что они уже были приняты де-факто — функции расширения и элементы расширения. Хотя они пользуются популярностью, это весьма неясная область, поскольку различные производители представили разные способы их реализации.

В XSLT 1.0 проверить доступность функции расширения можно при помощи функции function-available, а доступность элемента расширения — при помощи функции element-available.

XSLT 2.0 готовится определить стандартные средства связывания XSLT с элементами расширения и, вероятно, они будут гораздо лучше проработаны, чем имеющиеся сейчас.

Давайте посмотрим на все это в работе. В следующих разделах мы рассмотрим и функции, и элементы расширения, начав с функций.

ИНИЦИАТИВА EXSLT

Теперь, после того, как механизмы расширения в рабочем проекте XSLT 1.1 были отложены до XSLT 2.0, роль других разнообразных попыток стандартизации расширений XSLT значительно повысилась. Познакомьтесь, например, с EXSLT на www.exslt.org. EXSLT — это инициатива открытого сообщества, работающего над стандартизацией расширений XSLT.

Функции расширения

В XSLT 1.0 W3C определил способ отделения функций расширения от встроенных функций, установив требование, чтобы для обращения к функциям расширения использовались имена с заданным пространством имен, как в starpowder: calculate(). В XSLT 1.0 также имеется функция function-available() для проверки наличия функции по ее имени.

В рабочем проекту XSLT 1.1 на функции расширения были наложены некоторые дополнительные ограничения:

- функции расширения должны работать как встроенные функции;
- для Java и ECMAScript должны быть реализованы привязки к языку;

- механизм должен позволять естественное расширение для поддержки других языков в будущем;
- для реализации переносимой привязки функции расширения для любого конкретного языка не должен быть нужен процессор;
- процессор, реализующий функции расширения для любого языка, чья привязка обеспечивается спецификацией XSLT, должен соответствовать этим языкам;
- должны быть разрешены как встроенные реализации функций расширения, так и внешние;
- в функции расширения должно быть возможно передавать аргументы всех типов данных XPath;
- функции расширения должны иметь возможность возвращать в качестве результата все типы данных XPath;
- функции расширения должны иметь возможность создавать и возвращать наборы узлов фрагментов XML;
- должна иметься возможность включать или импортировать функции расширения из другой таблицы стилей;
- при неоднозначности выбора реализации функции расширения процессор должен выдать ошибку и прекратить работу;
- процессор должен преобразовывать аргументы способом, согласованным со встроенными функциями;
- функции расширения должны быть способны вернуть объект любого типа основного языка;
- должна существовать возможность передать в функцию расширения объект любого типа основного языка.

Вплоть до недавнего времени процессоры XSLT полностью самостоятельно определяли способ реализации функций расширения. Например, в Saxon и Xalan существует возможность непосредственно выполнять код Java, если определить пространство имен, задающее класс Java в качестве последней части URI. Я поступил так в следующем случае, определив пространство имен Date, соответствующее классу Java Date:

```
<?xml version-"1.0"?>
<xsl stylesheet version-"1.0"
xmlns xsl-http://www.w3.org/1999/XSL/Transform
xmlns:Date-"http://www.saxon.com/java/java.ut1l.Date">
```

После этого я могу воспользоваться такими функциями класса Date Java, как toString и new, для того чтобы заключить текущую дату в элементы заголовка <H1> HTML в выходном документе (листинг 5.14).

Листинг 5 14. Применение функций bate Java

```
<?xml version-*}..o*1?»
<xsl:stylesheel -ersiOn-"1.0"
xfdl ns:xsl-*"http Л м * * \:3. org/1999/XSL/Transfdi mv
xmlns:Date "httpr//ww*.saxon.coш/java/java.ut i1 Date">

    <xsl:template match-"/PLANETS"»
        <HIMLJ>
            <HEAD>
                <TITLE>
                    The Planets Table .
                </TITLF>
            </HEAD>
            <BODY>
                <HL>
                    The Planets Table
                • </HI>                                                    i
                <BR/>
                <HL>
                    <xsl:value-of select-"Date:toStrlng(Date:new(»"/>
                </HI>
                <tABLE B0RDER-"2">
                    <TD>flam:o</TD>
                    <TD>Mass</TD>
                    <TD>Radius</TD>
                    <TD:Day</TO> '
                    <xsl:apply-templates/>
                </TABLE>
            </BODY>
        </HTML>
    </xsl:tempiate>

    <xsl:tempiate match-"PLANET">
        <TR>
            <TCfxxsl :value-of select- 'NAME"/></TD>
            <TDxxsl: apply-tempi ates select-"MASS"7x/TD>
            <TDxxsl :apply-templates sel ect-"RADIUS" /></TD>
            <TUxxsT anply-templates select-"DAY"/x/TD>
        </TR>
    </xsltemplate»

    <xslrtemplate match-"MASS">
        <xsl:value-of select-"."/>
    </xsl:tempiate>

    <xsl:template match-"RADIUS">
        <xsl:value-of select-"."/>
    </xsl:template*

    <xsl:tempiate match-"DAY">
        <xsl:value-of select-"."/>
    </xsl:template»
yi;                C                .                I, ,
</xs 1: sty} eshoet>                 , Fr        v
```

Результат применения этой функции Приведён *ли* рис. 5.3. [a]

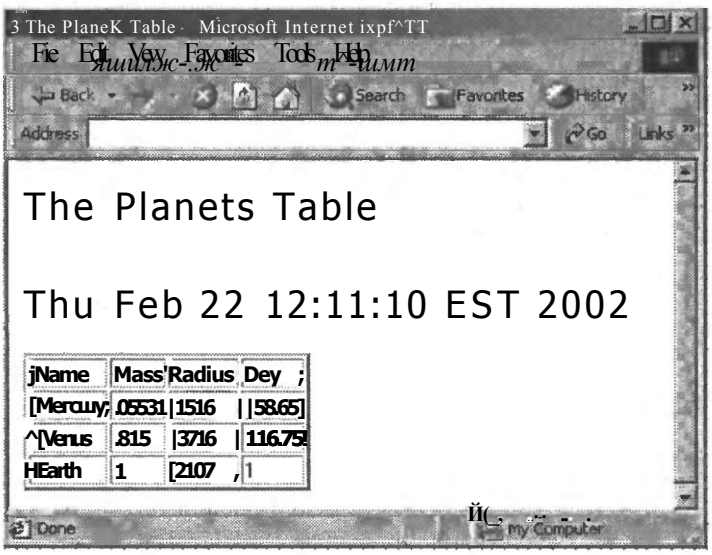


Рис. 5.3. Применение функции расширения

Вот работоспособная схема и зодно веский повод включить Java в XSLT. Тем не менее, в XSLT 1.1 был представлен элемент <xsl:script>, который, по всей видимости, будет включен и в XSLT 2.0.

Элемент <xsl:script>

Элемент <xsl:script> был определен в рабочем проекте XSLT 1.1, он предоставляет хорошо определенный способ связи функций расширения с таблицами стилей XSLT. Это элемент верхнего уровня, обладающий следующими атрибутами:

- implements-prefix (необязательный). Задает имя пространства имен функции расширения, которую реализует этот элемент. Принимает значение NCNAME;
- language (необязательный). Задает язык, используемый функцией расширения. Устанавливается в «ecmascript» (стандарт JavaScript), «javascript», «java» или QNAME, не являющееся NCNAME;
- src (необязательный). Предоставляет URI, в котором реализована функция расширения. Например, это может быть класс Java;
- arc! ve (необязательный). Задает архивы, которые необходимо загрузить перед запуском функции расширения, если они есть. Принимает значения списка URI, разделенного символами-разделителями.

Элемент содержит символьные данные (Microsoft использует раздел CDATA), реализующие функцию или функции расширения.

Как теперь связать ф[1], нкцию, определенную в элементе <xsl:script>, с вашей таблицей стилей XSLT? Сначала создайте в своей таблице стилей элемент <xsl:script> как элемент верхнего уровня, затем поместите в него функции, которые вы хоти-

те определить. В приведенном ниже примере я определяю две функции JavaScript, makeMoney (сделать деньги) и makeMoreMoney (сделать еще больше денег), в элементе <xsl:script>,]>еализующем пространство имен расширений «starpowder»:

```
<xsl:scr1pt implements prefix-"$tdrpowder" language-"javascript">
    function makeMoney(e)
    {
    .
    .
    .
    }
    function makeMoreMOney(e)
    {
    .
    .
    .
    i
</xsl:script»
```

В зависимости от вашего процессора XSLT$_r$ может оказаться хорошим решением заключить такою рода сценарии в раздел CDATA:

```
<xsl:script implements-prefix-i"sUrpowder" language "javascript"»
    <![CDATA[
        function makeMoney(e)
        {
        .
        .
        .
        }
    function makeMoreMoney(e)
    {
    .
    .
    .
    }]]>
</xsl:script»
```

Теперь при помощи пространства имен «starpowder» можно указать, что вызывается функция расширения:

```
<CASH>
    <xsl:value-of select-"starpowder:makeMoney(1000000)"/>
</CASH>
```

Вот и все (если ваш процессор XSLT это поддерживает). Если вместо сценария вы хотите указать класс Java, воспользуйтесь атрибутом sre:

```
<xsl:script implements-prefix»"starpowder• src-"java:com.MakeMoney", language*, "aava">
</xsl:script»
```

At

РАБОТА С ВНЕШНИМИ РЕСУРСАМИ

-5

Атрибут sre такюе иепользуеп я,, !сли есть архив подпрограмм JavaScript, как, например, src= "archives.js"

Из всех известных мне процессоров XSLT элемент <xsl:script> реализует только процессор Microsoft MSXML3. Информация о работе со сценариями для написания функций расширения для Internet Explorer приведена на web-узле Microsoft (в данный момент это страница по адресу http://msdn.microsoft.com/xml/xslguide/script-overview.asp, но, кажется, Microsoft меняет структуру web-узла каждые два дня или около того).

Следующий пример демонстрирует работу <xsl:script> с Internet Explorer. Я создал функцию JavaScript для преобразования данных о радиусе планет из planets, xml, приведенных в милях, в километры и последующем выводе этих данных в километрах.

Как обсуждалось в главе 2 в разделе «Преобразование документов XML при помощи Internet. Explorer», для просмотра XML-документа, использующего таблицу стилей XSL, в Internet Explorer, версии 5.5 и младше в документ необходимо внести некоторые изменения (если только вы не установили последний разборщик MSXML или не используете недавно появившуюся версию браузера 6.0, хотя и в этом случае нужно применять «text/xsl»). Для начала в таблице стилей XSL используйте тип MIME «text/xsl», а не «text/xml». Я также задал URI для таблицы стилей «kilometers.**xsbcneflyiouiHM** образом (листинг 5.15).

Листинг 5.15. Установка использования kilometers.xsl для planets.xml **в** Internet Explorer

```
<?xml version="1.0"?>
<?xml-stylesheet type-"text/xsl" href-"ki1ometers.xsl"?>
<PLANETS>

    <PLANET>
        <NAME>Mercury</NAME>
        <MASS UNITS-"(Earth - 1)">.0553</MASS>
        <DAY UNITS-"days">58.65</DAY>
        <RADIUS UNITS-"miles">1516</RADIUS>
        <DENSITY UNITS-"(Earth - 1)"> 983</DEN$ITY>
        «'DISTANCE UNITS="million mi 1es">43.4</DISTANCE><!--B перигелии-->
    </PLANET>

    <PLANET>
        <NAME>VeriUS</NAME>
        <MASS UNITS-"(Earth - 1)">.815</MASS>
        <DAY UNITS-"dayS">116.75</DAY>
        <RADIUS UNITS-"miI?s">37l6</RADIUS>
        <DENSITY UNITS-"(Earth - 1)">.943</DENSITY>
        <DISTANCE UNITS-"milTlon miles">66.8</DISTANCE>«1--B перигелии-->
    </PLANET>
        .
        .
        .
```

Для преобразования таблицы стилей kilometers.xsl для работы в IE 5.5 или младше я воспользовался пространством имен XSL, которое использует IE, и добавил элемент <xsl:script>, показав, что я собираюсь писать сценарии на JavaScript. Заметьте, однако, что элемент <xsl:script> Ь Internet Explorer не поддерживает атрибут префикса реализации, поэтому я не могу связать функции, определенные,в пространстве имен:

```
<xsl stylesheet xmlns xsl-"http://www.w3.org/TR/WD-xsl">
    <xs1:script language-"javascript"»
        .
        .
        .
    </xsl:script>
        .
        .
        .
```

В соответствии с требованиями Internet Explorer, код должен быть заключен в раздел CDATA. Здесь я определил функцию milesToKilometers, которая принимает узел, читает текст узла в свойстве text и преобразует текст в число миль при помощи функции JavaScript parselnt. Далее я умножаю число миль на 1,6, чтобы получить километры, и возвращаю результат:

```
<xsl:stylesheet xmlns:xsl-"http://www.w3.org/TR/WD-xsl">
    <xsl:script language-"javascript">
        <i[CDATA[
            function ml1esToKI1ometers(e)
            {
                miles - parseInt(e.text):
                return miles * 1.6:
            }
        ]]>
    </xsl:scr1pt>
        .
        .
        .
```

Поскольку пока в Internet Explorer нельзя связать пространство имен с функцией расширения, для их вызова используется специальный элемент Microsoft <xsl:eval>. Ниже показано, как это выглядит в таблице стилей kilometers.xsl, где я передаю в функцию milesToKilometers текущий узел <RADIUS> для преобразования миль в километры. Поскольку IE 5.5 и младше не поддерживают правила по умолчанию (хотя версия 6.0, вышедшая одновременно с подписанием этой книги в печать, поддерживает их, и вам не нужно ничего менять), для этих браузеров я предоставил правило для корневого узла (листинг 5.16).

Листинг 5.16. kilometers.xsl

```
<xsl:stylesheet xmlns:xsl-"http //www.w3.org/TR/WD-xsl">
    <xsl script language-"javascript">
        <![CDATA[
            function milesToKilometers(e)
            {
                miles - parseInt(e.text):
                return miles * 1.6:
            }
```

продолжение &

Листинг 5.16 (продолжение)

```
        ]Э>
    </xsl:script>
    <xs1:tempi ate match-"/">
        <HIML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <BODY>
                <H1>
                    The Planets Table
                </H1>
                «TABLE BORDER-"2">
                    <TR>
                        <TD>Name</TD>
                        <TD>Mass</TD>
                        <TD»Radius</TD>
                        <TD»Day</TD>
                    </TR>
                    <xsl:apply-templates/»
                </TABLE>
            </BODY>
        </HIML>
    </xsl:template»

    <xsl template match-"PLANETS">
        <xs1:apply-templates/>
    </xsl:tempiate>

    <xsltemplate match-"PLANET"»
        <TR>
            <TD»<xsl:value-of select-"NAME"/></TD>
            <TDxxsl: value-of select-"MASS'7></TO
            <TD><xsl:apply-templates match-"RADIUS"/></TD>
            <TD»<xsl:value-of select-"DAY"7»</TD>
        </TR>
    </xsl template»

    <xsl:template match-"RADIUS">
        <xsl:«val>milesToK1lometers(this)</xsl:eval>
    </xsl-.template»

</xsl:stylesheet»
```

Вот и все, результат этого преобразования приведен на рис. 5.4.'

Со временем производители будут поставлять все больше и больше функций расширения. Как можно определить, доступна ли заданная функция расширения? Для этого служит функция function-available.

Применение функции function-available

Функция XSLT 1.0 function-available служит для проверки доступности функции. В следующем примере я хочу воспользоваться функцией расширения starpowdet: calculate для математических вычислений, а если она недоступна, я отправляю

в результирующий документ текст «Sorry, can't do math today.» (Извините, сегодня математические вычислеиия не работают.), хотя можно, конечно, прекратить обработку и вывести сообщение об ошибке при помощи элемента <xsl:message>:

```
<xsl-.choose xmlns:starpowder-"http://www.starpowder.com">
    <xsl:when test»"function-available('starpowder:calculate')">
        <xsl:value-of select-"starpowder:calculate('2+2')"/>
    </xsl:when>
    <xsl:otherwise*
        <xsl:text>Sorry. can't do math today.</xsl:text>
    </xsl:otherwise*
</xsl:choose>
```

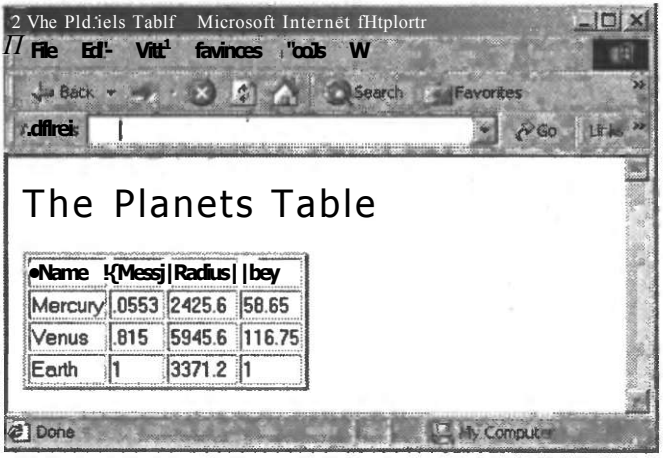


Рис. 5.4. Применение функции расширения в Internet Explorer

Внешние объекты

В рабочем проекте XSLT 1.1 для поддержки функций расширения появился новый тип данных — внешний объект (external object). Переменной XSLT, о которой пойдет речь в главе 9, может быть присвоен внешний объект — так же, как и один из четырех типов данных XPath, поддерживаемых в XSLT (строка, число, логическое значение, набор узлов). Внешний объект представляет объект, который создается внешним языком программирования, возвращается функцией расширения и не может быть преобразован в один из четырех типов данных XPath. Тип данных «external object» был добавлен в XSLT для того, чтобы предоставить вам безопасную «оболочку» для таких данных. Пока еще никто не реализовал поддержку внешних объектов, но это ожидается в скором времени.

Элементы расширения

Элементы расширения — это элементы, добавленные в XSLT пользователем или производителем. В рабочем проекте XSLT 1.1 для элементов расширения был установлен ряд правил, и в XSLT 2.0 предполагается более широкая их поддержка.

В рабочем проекте XSLT 1.1 правила определяли, что элементами расширения должны быть определенные пользователем или производителем элементы, не являющиеся элементами верхнего уровня. Они должны принадлежать к пространству имен, которое было определено как пространство имен расширений.

Для определения пространства имен расширений применяется атрибут extension-element-prefixes в элементе <xsl:stylesheet>, или атрибут xsl:extension-element-prefixes в элементе буквального результата или элементе расширения.

Ниже приведен пример. Xalan позволяет вам создать несколько выходных документов при помощи своего элемента расширения <redirect:write>. Для того чтобы применить этот элемент, я могу добавить в planets.xml элементу документа атрибут file, задав имя файла, в который будет отправлен вывод, как redirected.xml:

```
<?xml vers'on-"! ©'?>
<?xml-stylesheet type"text/xml" href»"planets.xsl"?>
«PLANETS f1le«"redire(ited.xmr>

    <PLANET>
        <NAME>Mercury</NAME>
        <MASS UNITS-"(Earth - 1)">.0553</MASS>
        <DAY UNITS-"days">58.65</DAY>
        <RADIUS UNITS-"miles">1516</RADIUS>
        «DENSITY UNITS-"(Earth = 1)">.983</DENSITY>
        «DISTANCE UNITS-"million miles">43.4</DISTANCE><!--В перигелии-->
    </PLANET>
```

Теперь в таблице стилей XSLT, которую я назвал redirect.xsl, я определяю пространство имен «redirect» так, чтобы оно соответствовало классу Java, который поддерживает ее в Xalan: org.apache.xalan.lib.Redirect. Я также устанавливаю атрибут extension-element-prefixes элемента <xsl:stylesheet> в значение пространства имен «redirect»:

```
<xsl:stylesheet xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"
    version-"1.0"
    xmlns:1xslt-"http://xml.apache.org/xslt"
    xralns:redi rect-"org.apache.xalan J1 i b.Redi rect"
    extens" on-element-pref1xes-"redirect ">
```

В этот момент мне ничто не мешает применить элемент расширения <redirect: write> для записи вывода в новый файл (в отличие от указанного в командной строке). Например, для того, чтобы отправить в другой файл отформатированное содержимое элемента <PLANET>, я могу получить имя создаваемого файла из атрибута file элемента <PLANETS> и записать данные в этот новый файл:

```
<xsl.-stylesheet xmlns:xsl="http://viMw.w3.org/1999/XSL/Transform"
    version-"1.0"
    xmlns:1xslt-"http://xml.apache.org/xsl t"
    xmlns:redi rect-"org.apache.xalan.lib.Redi rect"
    extensiOn-element-prefixes-"redirect">
```

```
<lxsIt:component prefix-"redirect" elements-"write Open close" functions-"">
  <1xslt:script 1ang-"javaclass" src-"org.apache.xalan.1ib.Redirect"/>
</1xslt:component»

<xsl:output method-"xml"/>

<xsl:template match-"/"»
    «xsl:apply-templates/>
</xsl:template»

<xsl:tempiate mat ch-'PLANEIS'>
  <red1rect:write select-"@file">
    «PLANEIS»
      «xsl:apply-tempiates/»
    «PLANEIS»
  «/redirect:write»
«/xsl:tempiate»

«xsl:tempiate match-"®*|node()"»
  «xsl:copy»
    «xsl -.apply-tempi ates select-"@*|node()"/>
  -«/xsl:copy>
«/xsl:tempiate»

«/xsl:stylesheet»
```

Операция завершена; вот как это может выглядеть при использовании Xalan в Windows:

```
C:p1anets»java org.apache.xalan.xslt.Process -IN planets.xml
-XSt redi rect. xsl -OUT new. xml
```

При этом будет создан файл redirected.xml, который выглядит следующим образом:

```
<?xml version-"1.0" encoding-"UTF-8"?>
«PLANETS»

    «PLANET»
        <NAME»Mercury</NAME>
        «MASS UNITS-,"(Earth - 1)">.0553«/MASS>
        «DAY UNITS-"days"»58.65«/DAY>
        «RADIUS UNITS-"miles"»1516</RADIUS»
        «DENSITY UNITS-"(Earth - 1)"».983</DENSITY>
        «DISTANCE UNITS-"million miles">43.4«/DISTANCE>
    «/PLANET»

    «PLANET»
        <NAME»Venus</ NAME»
        «MASS UNITS-"(Earth - 1)"».815</MASS>
        «DAY UNITS-"days"»H6.75«/DAY>
        «RADIUS UNITS-"miles"»3716</RADIUS>
        «DENSITY UNITS-"(Earth - 1)"».943«/DENSITY>
        «DISTANCE UNITS-"million miles">66.8</DISTANCE>
    «/PLANET-»
```

Применение функции element-available

Для проверки доступности элемента служит функция XSLT 1.0 element-available. В следующем примере я проверяю наличие элемента с названием <starpowder: calculate»:

```
<xsl:choose  xmlns:starpowder-"http://www.starpowder.com">
    <xsl when test' "element-available('starpowder:calculate")">
        <sta rpowder:ca1culate xs1:extens i on-element-prefi xes»"sta rpowder"/>
    </xsl:when>
    <xsl:otherwise>
        <xsl:text>S6rry. can't do math today.</xsl:text>
    </xsl:otherwise>
</xsl:choose>
```

Есть, еще один способ обработать случай отсутствия элемента расширения — элемент <xsl:fallback>.

Элемент <xsl:fallback>

При помощи элемента XSLT 1.0 <xsl:fallback> можно указать, что следует делать в случае отсутствия элемента расширения. Этот элемент заключается в элемент расширения и используется в случае его недоступности.

У элемента <xsl:fallback> нет атрибутов, он содержит тело шаблона.

В следующем примере я создам элемент <xsl:fallback> внутри элемента «redirect: write> из предыдущего примера. В случае отсутствия элемента <redirect:write> элемент <xsl:fallback> прекратит обработку и выдаст сообщение:

```
<xsl:stylesheet  xmlns:xs1-"http://www.w3.org/1999/XSL/Transform"
    version="1.0"
    xmlns Jxslt-"http://xml.apache.org/xslt"
    xmlns:redi rect-"org.apache.xalan.1i b.Redi rect"
    extens i on-element-prefi xes-"red i rect">

  <1xsltcomponent prefix-"redirect" elements-"write open close" functions-"">
    <1xslt:scri pt 1ang-"javaclass"  src="org.apache.xalan.1i b.Redirect"/>
  </lxslt:component>

  <xsl output method-"xml"/>

  <xsl:template match="/">
      <xsl:apply-templates/>
  </xsl:tempiate>

  <xsl:template match-"PLANETS">
    <redi rect:wri te select-"@fi 1e">
      <PLANETS>
        <xsl:apply-templates/>
      </PLANETS>
      <xsl:fallback*
          <xsl:message termi nate-"yes">
              <xsl:text>Could not create multiple output documents.</xsl:text>
          </xsl:message>
```

```
        </xsl:fallback»
      «/redirect:write»
    </xsl:tempiate»

    <xsl -template match-"@*|node()">
      <xsl:copy>
        <xsl:apply-temp1ates  select-"@*|node()"/>
      </xsl:copy>
    </xsl:tempiate»

  </xsl  stylesheet»
```

На этом мы заканчиваем введение в работу с данными в документах XML. В следующей главе мы разберем эту тему и узнаем, как изменять содержимое документа и создавать новые элементы, атрибуты и инструкции обработки.

Преобразование в XML, HTML, XHTML, RTF

6

i

Предположим, web-узел вашей компании использует основанное на XML программное обеспечение фирмы Commerce One, в котором для безопасной коммуникации через Интернет применяется Java Message Service (JMS). Ваша деятельность была настолько успешной, что вы только что поглотили своего конкурента. К сожалению, для своего узла в Интернете ваш бывший конкурент использует другой основанный на XML продукт, RosettaNet. Как вам теперь преобразовать заказ на покупку xCBL Commerce One, написанный на XML, в заказ на покупку RosettaNet, также написанный на XML, но совершенно на другом диалекте?

Разумеется, применить XSLT. Такого рода XML—XML преобразования становятся все более и более распространенными. Все больше компаний применяют JMS для безопасных коммуникаций через Интернет, и поскольку JMS выполняется в Java, будет разумным связать JMS с основанными на Java процессорами XSLT, такими, как Xalan или Saxon.

В этой книге мы уже рассматривали преобразование XML в HTML, XML и простой текст, но в текущей главе сделаем это более подробно. Мы также рассмотрим здесь новый тип преобразования — из XML в JavaScript. В главе 10 мы познакомимся с преобразованиями из XML в базу данных на основе SQL, а в главе 11 — с преобразованиями из XML в XSL-FO.

Основная задача XSLT состоит не просто в замене одного элемента на другой, но в полной реорганизации содержимого XML-документа. Например, вам может потребоваться реорганизовать planets.xml в терминах плотности планет при помощи XSLT для создания нового XML-документа:

```
<?xml version-" 1.0" encoding="UTP-8"?>
<DATA>

    <DENSITY>
        <VALUE>.983</VALUE>
        <NAME>Mercury</NAME>
        <MASS> 0553</MASS>
```

```
    <DAY>58.65</DAY> .
    <RADIUS>1516</RADIUS>
</DENSITY>

<DENSITY>
    <VALUE>.943</VALUE>
    <NAME>Venus</NAME>
    <MASS>.815</MASS>
    <DAY>116.75</DAY>
    <RADIUS>3716</RADIUS>
</DENSITY>

<DENSITY>
    <VALUE>1</VALUE>
    <NAME>Ea rth</NAME>
    <MASS>1</MASS>
    <DAY>1</DAY>
    <RADIUS>2107</RADIUS>
</DENSITY>
```

- </DATA>

Мы рассмотрим преобразование, которое полностью меняет содержимое planets, xml, оставляя только небольшой код HTML и код JavaScript для отображения нескольких кнопок в браузере.

До сих пор мы создавали новые элементы только при помощи элементов буквального результата, то есть рассматривая новые элементы как текст и встраивая их в таблицу стилей. Но, как мы увидим в этой главе, не всегда возможно знать имена создаваемых новых элементов. Можно состыковать вместе создаваемые элементы по ходу дел&, рассматривая их как Сырой текст, но это явная недоработка, поскольку разметка трактуется как текст. В этой главе мы начнем применять элементы XSLT <xsl:element>, <xsl:attribute>, <xsl:processing-instruction> и <xsl:comment> для создания новых элементов, атрибутов, инструкций обработки и комментариев на этапе выполнения. Хорошее знание этих элементов Необходимо при реорганизации содержимого XML.

Мы также рассмотрим использование режимов XSLT для осуществления нескольких преобразований с документом и сориентируемся, как применять только один из нескольких подходящих шаблонов.

В большей части этой главы исследуются возможности элемента <xsl:output>, с краткого обзора которого я и начну.

Элемент <xsl:output>

С элементом <xsl:output> мы впервые познакомились в главе 2 и использовали его, главным образом, для задания типа результирующего документа. Этот тип может задать, например, будет ли процессор XSLT записывать инструкцию обработки XML, <?xml version-" 1.0"?>, в начале документа, а также задать тип MIME (такой, как «text/xml» или «text/htmU) документов, отправляемых процессором XSLT из web-сервера браузеру. Кроме того, если вы установите выходной тип

в HTML, большинство процессоров XSLT смогут распознать, что не всем Элементам HTML необходимы закрывающие или открывающие теги и т.; п. и

В следующем списке перечислены атрибуты <xsl:output>:

- cdata-section-elements (необязательный). Задает имена тех элементов, чье содержимое должно выводиться как разделы CDATA. Принимает значения списка QName, разделенного символами-разделителями; 4 -«иМл.! £
- doctype-public (необязательный).[1] Задает открытый идентификатор,- который будет использован в объявлении <!DOCTYPE> в выходных данных. Устанавливается в строковое значение; >.
- doctype-system (необязательный). Задает системный идентификатор, который буДеТ использован в объявлении <!DOCTYPE> в выходных данных. Устанавливается в Строковое значение;
- encoding (необязательный). Задает кодировку символов. Устанавливается в строковое значение;
- indent (необязательный). Определяет, будет ли выходной документ выровнен с отражением структуры вложенности. Устанавливается в yes или по;
- media-type (необязательный). Задает тип MIME вывода. Устанавливается в строковое значение;
- method (необязательный). Задает формат вывода. Принимает значения «xml», «html», «text» или допустимое имя QName;
- omit-xml-declaration (необязательный) Определяет, будет ли включено в вывод объявление XML. Устанавливается в «yes» или «по»; г •
- standalone (необязательный). Определяет, будет ли включено в вывод- отдельное объявление XML, и если да — устанавливает его значение. Устанавливается в yes или по;
- version (необязательный). Задает версию вывода. Принимает значение допустимого NMToken.

Чаще, всего используется атрибут method, поскольку именно он определяет требуемый тип выходного дерева. Официально методом вывода по умолчанию является HTML, при условии, что выполняются все три следующих условия:

- корневой узел результирующего дерева имеет дочерний элемент;
- в названии элемента документа результирующего дерева присутствуй часть «html» (в любой комбинации верхнего и нижнего регистров) и пустой URI пространства имен;
- все текстовые узлы перед первым дочерним элементом корневого узла содержат только символы-разделители.

Если выполняются все три этих условия, то по умолчанию метод вывода устанавливается в HTML, В ином случае методов вывода по умолчанию является XML. s _»; .1 iv <• - ,••> ют*?. •<

Тем не менее, не стоит полагаться на установки метода вывода по умолчанию, лучше явно присвоить этому атрибуту значение. Три распространенных значения атрибута method — «html», «xml» и «text», и мы познакомимся с ними в следующих разделах.

Метод вывода: HTML

Для метода вывода HTML процессор XSLT должен предпринять определенные действия. Например, для этого метода атрибут version определяет версию HTML. Значение по умолчанию — 4.0.

Этот метод не должен добавлять завершающий тег для пустых элементов. (Для HTML 4.0 пустыми элементами являются <AREA>, <BASE>, <BASEF0NT>, <BR>, <C0L>, <FRAME>, <HR>, <IMG>, <INPUT>, <ISINDEX>, <UNK>, <META> и <PARAM>.) Метод вывода HTML должен распознавать названия элементов HTML независимо от регистра.

В соответствии с W3C, метод вывода HTML не должен скрывать содержимое элементов <SCRIPT> или <STYLE>. Например, следующий элемент буквального результата:

```
<SCRIPT>
    if (x &lt: y) {  }
</SCRIPT>
```

или следующий, использующий раздел CDATA:

```
<SCRIPT>
    <![CDATA[if (x < y) {...}]]>
</SCRIPT>
```

должен быть преобразован в:

```
<SCRIPT>
    if (x < y) {...}
</SCRIPT>
```

Метод вывода HTML не должен также подавлять символы <, встречающиеся в значениях атрибутов.

При установке метода вывода в HTML процессор может учесть атрибут выравнивания. Если этот атрибут установлен в yes, процессор XSLT может добавить (или удалить) символы-разделители для выравнивания результирующего документа,- поскольку это не влияет на отображение документа в браузере. Для метода* вывода HTML значением по умолчанию является «yes».

Как вы могли предположить, метод вывода HTML завершает инструкции обработки при помощи >, а не ?>, а также поддерживает отдельные атрибуты, как и HTML. Например, тег

```
<TD NOWRAP-'NOWRAP'>
```

будет преобразован в:

```
<TD NOWRAP>
```

Для этого метода можно установить атрибут media-type, значением по умолчанию для которого является «text/html». Метод HTML не должен убирать

символ &, который появляется в значении атрибута, если сразу за ним следует фигурная скобка. Атрибут encoding задает используемую кодировку. Если присутствует элемент <HEAD>, этот метод вывода должен добавить элемент <META> сразу же после тега <HEAD>, определяя кодировку символов:

```
<HEAD>
    <META http-equiv="Content-Type" content-"text/html; charset-utf-8">
```

При помощи атрибутов doctype-public или doctype-system можно вывести объявление типа документа непосредственно перед первым элементом, как мы увидим при преобразовании XML в XHTML.

Таковы правила вывода HTML. Ниже приведен пример преобразования из, XML в HTML с небольшими отклонениями. В этом случае таблица стилей будет фактически генерировать код JavaScript, демонстрируя создания JavaScript при помощи XSLT. В частности, мы прочитаем planets.xml и создадим новый документ HTML, отображающий три кнопки — по одной для каждой из трех планет в planets.xml. При щелчке на кнопке на странице будет выведена масса соответствующей планеты.

Все, что нам понадобится (листинг 6.1), — это два элемента <xsl:for-each>: один для прохода в цикле по трем планетам и создания для каждой кнопки HTML; и один для прохода по планетам и создания для каждой функции JavaScript. В качестве имен функций JavaScript я воспользуюсь названием планет; при вызове функция выведет массу соответствующей планеты. Заметьте, что для создания нужного кода JavaScript нужно всего лишь применить элемент <xsl:value-of> для получения названий и масс планет. Я также применю два новых элемента XSLT, <xsl:element> и <xsl:attribute-set>, которые мы рассмотрим позже в этой главе, для создания нового элемента и задания для него набора атрибутов.

Листинг 6.1. Преобразование в JavaScript

```
<?xml vers1on-"l.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">
<xsl:output method""html"/>

    <xsl-template match-"/PLANETS">
<HTML>

    <HEAD>
        <TITLE>
            The Mass Page
         «/TITLE»

        «SCRIPT LANGUAGE-'javascript'»

        <xs1:for-each select-"PLANET"»
        <xsl:text>
            function </xsl:text><xsl:value-of select-"NAME"/><xsl:text>()
            {
                dlsplay.lnnerHTML - 'The mass of </xsl:text>
                <xsl:value-of select-"NAME"/>
```

```
                <xsl:text> equals </xsl:text*
                <xsl:value-of $elect*"MASS"/>
                <xsl:text> Earth mas«es."</xsl:text*
            }
        </xsl:for-each*
        </SCRIPT>
    </HEAD>

    <BODY>
        <CENTER>
            <Hl>The Mass Page</Hl>
        </CENTER>
        <xsl:for-each select^Pi ANET">
            <P/>
            <xsl:element name""input" use-attr1hute-sets-"attr1bs"/>
        </xsl:for-each>
        <P/>
        <P/>
        <DIV ID-'display'x/DIV*
    </BODY>

</HTML>
    </xsl:template*

    <xslattribute-set name-"attr1bs">
    <xsl: attribute name=' type"*BUTTON</xsl: attribute*
    <xsl:attribute name-"value"><xsl:value-of select-"NAME"/*</xsl:attribute>
    <xsl: attribute name-"onclick"*<xsl: val ue-of select-"NAME"/*C)"S/xsl: attribute*
    </xsl:attr1bute-set*

    </xsl:stylesheet*
```

Результат, включая элемент <SIRIPT> для нового кода JavaScript (листинг 6.2). -

Листинг 6.2. Результирующий документ с JavaScript

```
<HTML>
    <HEAD>
        <TITLE*                                                    x
            The Mass Page
        </TITLE>

        <SCRIPT LANGUAGE"javascript"*
            function MercuryO
            {
                display.innerHTML •
                    'The mass of Mercury equals .0553 Earth masses.'
            }

            function VenusO
            {
                display.InnerHTML - 'The mass of Venus equals .815 Earth masses.'
            }

            function £arth()
            {
                display.InnerHTML * 'The mass of Earth equals 1 Earth masses.'
            }
```

. v продолжение &

\

Листинг 6.2 (продолжение)

```
        </SCRIPT>

    </HEAD>

    <BODY>
        <CENTER>
            <H1>The Mass Page</H1>
        </CENTER>

        <P></P>
        <input type-"BUTTON" value»"Mercury" onclick-"Mercury()">
        <P></P>
        «input type-"BUTTON" value-"Venus" onclick-"Venus()">
        <P></P>
        «input type-"BUTTON" value="Earth" onclick-"Earth()">
        <P></P>
        <P></P>
        «DIV ID-"<jisplay"></DIV>

    </BODY>
«/HTML>
```

Как видите, при помощи XSLT я написал код JavaScript для прохода в цикле по планетам. Этот документ HTML показан на рис. 6.1. При щелчке на кнопку выводится масса соотвествующей планеты.

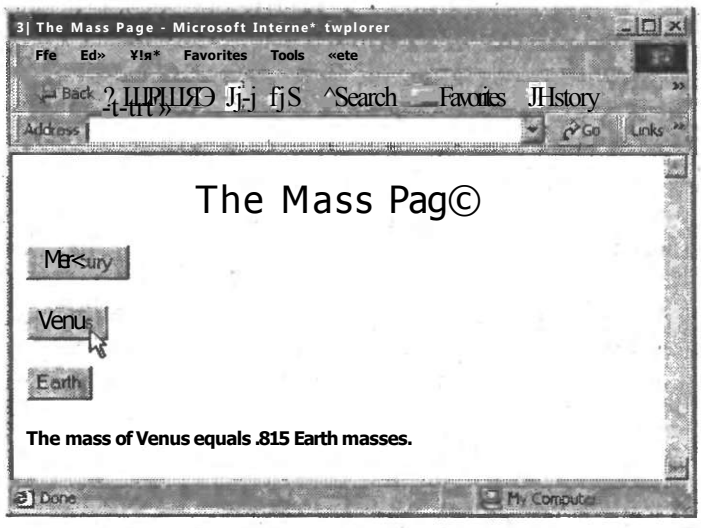


Рис. 6.1. **Преобразование XML в HTML при помощи JavaScript**

Метод вывода: XML

Формально при использовании метода вывода XML процессор XSLT создает хорошо сформированный внешний объект XML, который можно разобрать на общих условиях. Если корневой узел результирующего дерева имеет единствен-

ный дочерний узел-элемент и не имеет текстовых дочерних узлов. Тогда объект будет также хорошо сформированным документом XML.

При использовании метода вывода XML атрибут version устанавливает версию XML результата. Заметьте, что если процессор XSLT не поддерживает эту версию XML, он будет использовать ту версию XML, которую поддерживает. По умолчанию установлено значение 1.0.

Атрибут encoding устанавливает кодировку для результирующего документа. Процессоры XSLT должны поддерживать, по крайней мере, значения «UTF-8» и «UTF-16». Если процессор XSLT работает с другими значениями и не поддерживает указанную кодировку, он может сгенерировать ошибку. Если он этого не сделает, процессор должен **использовать** вместо нее UTF-8 иди UTF-16. Процессор XSLT не должен использовать кодировку, которая не была принята консорциумом W3C (см. www.w3.0rc^R/RE£-xmt). Если никакой атрибут кодировки не указан, по умолчанию процессор XSLT должен выбрать «UTF-8» или «UTF-16».

ОБРАБОТКА НЕИЗВЕСТНЫХ СИМВОЛОВ

Если результирующий документ содержит символ, который не может быть представлен в кодировке, используемой процессором XSLT для вывода, символ допустимо вывести как ссылку на символ. Если это невозможно, процессор XSLT должен сгенерировать ошибку.

Как и в случае с методом вывода HTML, если атрибут indent установлен в «yes», метод вывода XML может добавить или удалить символы-разделители в результирующее дерево для того, чтобы выровня гь результат. Значение по умолчанию — по. Заметьте^ что если символы-разделители отбрасываются, информационное множество результирующего XML-документа должно быть таким же, как если бы символы-разделители вообще не добавлялись г не удалялись для выравнивания документа.

ВЫРАВНИВАНИЕ ДОКУМЕНТОВ СО СМЕША)"н Щ СОДЕРЖИМЫМ

Для документов со смешанным содержимым лучше не устанавливать атрибут indent в «yes», поскольку это вносит путаницу в работу процессора XSLT.

Ири помощи атрибута cdata-section-elements можно задать разделенный символами-разделителями список имен элементов, чье содержимое должно трактоваться как разделы CDATA. Например, если установить атрибут cdata-section-elements'в «DATA»:

```
<xsl:output cdata-secti on-elements="DATA"/>
```

то следующий элемент буквального результата:

```
<DATA>&lt;DOCUMENT&gt;</DATA>
```

будет преобразован в:

```
<DATA><! [CDATA<[DOCUMENT>] ]></DATA>
```

Кроме того, метод вывода XML будет выводить в резулыгирутэщий документ объявление XML, если только, йтрйбут omit-xml-declaration не будет установлен в yes, Как правило,, объявление XML, помещаемое в результирующий документ,

обычно включает версию XML (что обязательно) и информацию о кодировке (хотя формально информация о кодировке в документах XML не обязательна). Если задан атрибут standalone, результирующий документ должен включать объявление отдельного документа с тем же значением, что и значение у атрибута standalone.

При использовании атрибута doctype-system процессор создает объявление типа документа непосредственно перед первым элементом. В этом случае имя, следующее за <!DOCTYPE-, будет именем корневого элемента. Заметьте, что если вы также используете атрибут doctype-public, процессор XSLT выведет «PUBLIC», вслед за ним открытый идентификатор и затем системный идентификатор. Если вы не используете атрибут doctype-public, процессор выведет «SYSTEM» и вслед за ним системный идентификатор. Теоретически атрибут doctype-public должен игнорироваться, если не задан также атрибут doctype-system, хотя большинство процессоров, кажется, не следуют этому правилу. Мы увидим, как работать с атрибутами doctype-public и doctype-system в этой главе при преобразовании XML в XHTML.

Наконец, для метода вывода XML значением по умолчанию для атрибута media-type является «text/xmU.

Вы уже встречали в этой книге многие XML-XML преобразования. Например, преобразование из главы 4 просто копировало один документ XML в другой. Обратите внимание на метод вывода, который установлен в XML:

```
<?xml version-"1.0"?>
<xsl: stylesheet version-"10"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">
<xsl:output method-"xml"/*

  <xsl:template match-"(a*|nod,e()">
/   <xsl:copy>
      <xsl:apply-templates select-"@*|node()7>
    </xsl:copy>
  </xsl:template»

</xsl:stylesheet»
```

Этот пример был приведен в начале главы, в нем planets.xml реорганизован на основе плотности планет:

```
<?xml version-"1.0" encoding-"UTF-8"?>
<DATA>

    <DENSITY>
        <VALUE>.983</VALUE>
        <NAME>Mercury</NAME>
        <MASS>.0553</MASS>
        <DAY»58.65</DAY>
        <RADIUS»1516</RADIUS>
    «/DENSITY»

    <DENSITY>
        «VALUE».943</VALUE»
        <NAME>Venus</NAME>
        «MASS>.815</MASS>
        <DAY»116 75</DAY>
```

```
            <RADIUS>3716</RADIUS>
        </DENSITY>
    • I .     • : .
        <DENSITY*
            <VALUE>1</VALUE>
            <NAME*Earth</NAME*
            <MASS*1</MASS>
            <DAY>1</DAY>
            <RADIUS*2107</RADIUS>
        </DENSITY>

    </DATA>
```

Вот таблица стилей (листинг 6.3), создающая это преобразование.

Листинг 6.3. Реорганизация planets.xml на основе плотности

```
<?xfol versiori="1.0"?>
<xsl: stylesheet version-*" 1.0"      .                «%
xmlns:xsl="http://www.w3.org/I999/XSL/Transform">
<xsl:output method-"xml" indent-"yes"/>

    <xsl:tempi ate match-"PLANETS">
        <DATA>
            <xsl:apply-templates/>
        </DATA>
    </xsl template*

    <xsl:template match-"PLANET"*
        <DENSITY>
            <VALUE>
                <xsl:value-of select-"DENSITY"/*
            </VALUE>
            <xsl:apply-templates/*
        </DENSITY*
    </xsl template*

    <xsl template match- NAME"*
        <NAME>
            <xsl:value-of select-"."/*
        </NAMF*
    </xsl:tempi ate*

    <xsl template match-"MA$S">
        <MASS>
        <xsl:value-of select-"."/*
        </MASS>
    </xsl:tempiate*

    <xsl:tempi ate match="RADIUS">
        <RADIUS>
        <xsl:value-of select-"."/*
        </RADIUS> I
    </xsl:tempi ate*

    <xsltemplate match-"DAY">
        <DAY*
        <xsl:value-of select-" ."/>
```

продолжение &

Листинг 6.3 (продолжение)

```
        </DAY>
    </xsl:template»

    <xsl template match-"DENSITY">
    </xsl:template»

    <xsl:tempiate match""DISTANCE"»
    </xsl:tempi ate»

</xsl:stylesheet»
```

Следующий пример впервые был приведен в главе 5. В этом случае я только перечислил планеты из planets.xml, но в выходном документе я хотел видеть не просто фразу «The first three planets are: Mercury Venus Earth» (первые три планеты: Меркурий Венера Земля), а «The first three planets are: Mercury, Venus, and Earth.». Для этого я применил элементы <xsl:if>:

```
<?xml version-"!..0"?>
<xsl:stylesheet version-"1.0" xmlnsxsl-"http://www w3 org/1999/XSL/Transform"»

«xsl:output method-"xml"/>
<xsltemplate match-"PLANETS"»
«DOCUMENT»
    «TITLE»
        The Planets
    </TITLE>
    <PLANETS>
        The first three planets are: <xslrapply-templates select-"PLANET"/»
    «/PLANETS»
«/DOCUMENT»
«/xsl:tempiate»

«xsl:tempiate match-"PLANET"»
    «xsl:value-of select-"NAME"/>
    <xsl:if test-"position()!»last()">. «/xsl:if>
    «xsl:1f test-"pos1tion()-last()-1">and «/xsl:1f>
    «xsl :if test^'positionQ-lastO^.c/xsl :if>
«/xsl template»

«/xsl:stylesheet»
```

И вот результат:

```
«?xml version-"1.0" encoding-»"UTF-8"?>
<DOCUMENT>

    <TITLE>
        The Planets

    «/TITLE»

    «PLANETS»
        The first three planets are: Mercury. Venus, and Earth.
    </PLANETS»
' «/DOCUMENT>
```

Хотя многие книги рассматривают главным образом преобразования из XML в HTML, важно понять, что преобразования XML—XML завоевывают все большую популярность, поэтому на них я также останавливаю ваше внимание.

Метод вывода: текст

-Этот метод вывода представляет простой текст. В данном случае выходной документ являет собой только простой текст дерева документа. То есть процессор XSLT создает результирующее дерево, выводя строковое значение каждого текстового узла, без каких-либо сокращений.

Значением по умолчанию для атрибута media-typ^ Является «text/plain». Атрибут encoding устанавливает кодировку, используемую процессором XSLT для преобразования последовательностей символов в последовательности байтоЬ. Заметьте, что есйи результирующий документ содержит символ, который не может быть представлен в выходной'кодировке, процессор XSLT должен сгенерировать ошибку.

В листинге 6.4 planets.xml преобразуется в простой текст при помощи метода текстового вывода.

Листинг 6.4. Преобразование в простой текст

```
<?xml version-"1.0"?>
<xs1:stylesheet vers ion-"1.0"                >
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"> •

    <xsl:output method="text" indent-"yes"/>

    <xsl:template match-"PLANET">
        <xsl:value-of select-"NAME"/>
        <xsl:text>'s mass is </xsl:text>
        <xsl:value-of select-"MASS"/>
        <xsl:text> Earth masses. Its rad4us is </xsl:text>
        <xsl:value-of select="RADIUS"/>
        <xsl:text> miles. Its day is </xs>:text>
        <xsl:value-of select-"DAY"/>
        <xsl:text> Earth days long.</xs! ;text*      . >.-.
  </xsl:template*

</xsl:stylesheet*
```

И вот результат — просто чистый текст, никакой разметки, никаких пропущенных символов, никаких инструкций обработки:

```
Mercury's mass is .0553 Earth masses. Its radius is 1516 miles. Its day
is 58.65 Earth days long.

Venus's mass is .815 Earth masses. Its radius is 3716 miles. Its day
is 116.75 Earth days long.

Earth's mass is 1 Earth masses. Its radius is 2107 miles. Its day
is 1 Earth days long. •                                          ^
```

- С другой стороны, метод текстового вывода предназначен не только для создания простого текста, он также применяется для любых текстовых форматов,

отличных от XML и HTML. Как мы видели в главе 2, с его помощью можно создавать форматированные документы в формате Rich Text Format (RTF). Rich Text Format использует встроенные текстовые коды, задающие формат документов, и при помощи ме+ода текстового вывода вы можете самостоятельно поместить эти коды в документы.

Следующий пример таблицы стилей первоначально был приведен в главе 2, где она использовалась с целью перевода planets.xml в формат RTF, для понимания которого вы теперь лучше вооружены. В этом случае я преобразую planets.xml в planets.rtf, используя коды RTF как элементы буквального результата:

```
<?xu1 version-"1.0"?>
<xsl:stylesheet version-"l O"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">
    <xsl:output method-"text"/>
    <xsl:strip-space elements-"*"/»
<xsl:tempi ate match="/PLANETS"»{\rtfl\ansi\deffO{\fonttbl
{WfcharsetO Courier New:}}
\viewkind4\ucl\pard\langl033\b\ The Planets Table\par
\b0                    *
Name\tab Mass\tab Rad.Ntab Day\par
<xsl:apply-templates/>
\par
}</xsl:template»
<xsltemplate match-"PLANET">
<xsl:value-of select-"NAME"/>
\tab
<xsl:value-of select-"MASS"/>
\tab
<xsl;value-of select-"RAOIUS"/>
\tab
<xsl:value-of select-"DAY"/>
\tab
\par
</xsl:template»
</xsl:stylesheet»
```

Результирующий документ RTF, planets.rtf, показан на рис. 6.2 в редакторе Microsoft Word 2000.

Заметьте, что методом вывода является текст, а не что-нибудь наподобие «rtf»:

```
<?xml vers1on-"l.O"?»
<xsl:stylesheet vers1on-"l.O"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"»
    <xsl:output method-"text"/>
<xsl:tempiate match-"/PLANETS">{\rtfl\ansi\deffO
{\fonttbl{WfcharsetO Courier New;}}
\viewkind4\ucl\pard\langl033\b\ The Planets Table\par
```

Отметьте также, что я поместил коды RTF сразу же после Элемента <xsl: template>, поскольку документы RTF должны с самого начала начинаться с кодов RTF; если бы я начал вставлять коды RTF на следующей строке, как, например:

```
<?xml version="1.0"?> i
<xs1-.stylesheet version-"1.0"
xmlns: xsl-"http: //www.w3. org/l999/XSL/Transform">
    <xsl:output met hod="text"/>
    <xsl:tempiate match-"/PLANETS">
{\rtfl\ans1\deff0{\fonttbl{WfcharsetO Courier New:}}
\viewkind4\ucl\pard\langl033\b\ The Planets Table\par
```

то выходной файл RTF начинался бы с символа новой строки, что привело бы к ошибке в приложении, работающем с файлом RTF (например, вероятно, Microsoft Word).

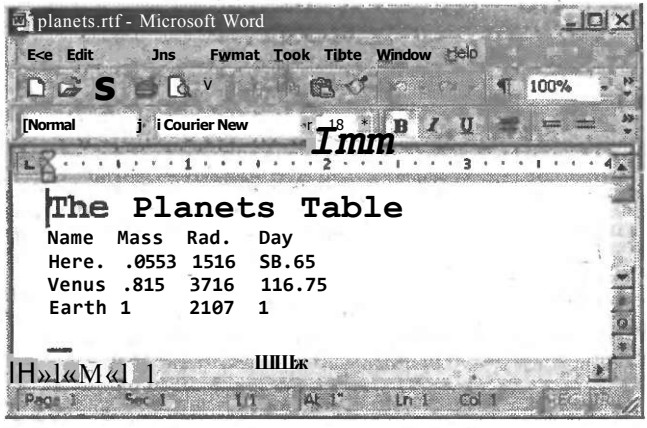


Рис. 6.2. Файл planets.rtf в Microsoft Word

Вывод в формате XHTML

W3C представил XHTML как последователя HTML, но ни в XSLT 1.0, ни в рабочем проекте XSLT 1.1 нет никакой специальной поддержки для преобразований из XML в XHTML. Предполагается, что надлежащая поддержка будет включена в XSLT 2.0. Тем не менее, при помощи процессоров XSLT все равно можно создавать документы XHTML.

ПОДРОБНЕЕ О XHTML

Если вы хотите узнать больше о XHTML, обратитесь к источникам: W3C рекомендация XHTML 1.0 по адресу www.w3.org/TR/xhtml1/, а также рекомендация XHTML 1.1 по адресу www.w3.org/TR/xhtml11/.

Помимо заботы о том, что ваш документ должен удовлетворять правилам XHTML (таким, как отсутствие отдельных атрибутов, заключение в кавычки всех значений атрибутов, использование для разметки символов в нижнем регистре, наличие для каждого открывающего тега закрывающего, проверка того, что документ является хорошо сформированным XML, и т. д.), основная задача — гарантировать, что в результирующем документе появится элемент <!DOCTYPE>.

Ниже приведены элементы <!DOCTYPE>, которые следует использовать с тремя типами XHTML 1.0 — строгим (strict), переходным (transitional) и кадровым (frameset):

```
<!DOCTYPE html
      PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!DOCTYPE html
      PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN".
      "http://www.w3.org/TR/xhtmll/DTD/xhtmll-transiti0nal .dtd">

<!DOCTYPE html
      PUBLIC "-//W3C//D!"D XHTML 1.0 Frameset//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtmll-frameset,dtd">
```

А вот элемент <!DOCTYPE> для XHTML 1.1:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
      "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

ЭЛЕМЕНТЫ <!DOCTYPE> И HTML 4.01

Строго говоря, даже документы HTML должны начинаться с элемента <!POCTYPE>. Официально существует три формы HTML 4.01: строгая (strict), переходная (transitional) и кадровая (frameset). Вот полные элементы <!DOCTYPE> для этих версий: <;!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/ htm!4/strict.dtd">, clDOCTYPE HTML PUBLIC *-//W3C// DTD HTML 4.01 Transitional//EN""http://www.w3.org^html4/loose.dtd"> и clDOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">. Если вы генерируете совершенно правильные документы HTML, не забудьте добавить в них этот элемент. Дополнительную информацию можно получить по адресу www.yv3.org/TR/html40/ struct/global.html.

При установке метода вывода в XML для создания элемента <!DOCTYPE> можно воспользоваться атрибутами doctype-system и doctype-public элемента <xsl:output>. Вот пример элемента <xsl:output>, создающего элемент <!DOCTYPE> для переходного XHTML 1.0:

```
<xsl:output method-"xml"                                                    Lx*
    doctype-system-"http://www.w3.org/TR/xhtml 1/DTD/xhtml 1-transitional ;dtd"  -  fi>.
    doctype-public-"-//W3C//DTD XHTML 1 0 Transitional//EN"  indent-"yes"/>
```

В листинге 6.5 приведена полная таблица стилей planets.html, использующая этот элемент <xsl:output> для преобразования planets.xml в допустимый документ XHTML.

Листинг 6.5. Преобразование planets.xml в XHTML

```
<?xml version="1.0"?>
<xsl: stylesheet versi on-" 1 0"                                  ....
xmlns:xsl -"http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml"                                                  «-
    doctype-system-"http://www. w3.org/TR/xhtmll/DTD/*htirt1                  У;гз;к;
    doctype-public="-//W3C7/OTD XHTML 1.0 Transitional //EN"«dndent**"yes" />

    <xsl: tempi ate match-="/PLANETS">
        <html>
            <head>                          >                  sib
```

```
                <title>
                    The Planets Table
                </title>
            </head>
            <body»
                <hl>
                    The Planets Table
                </hl>
                <table>
                    <tr>
                        <td>Name</td>
                        <td>Mass</td>
                        <td>Radius</td>
                        <td>Day</td>
                    </tr>
                    <xsl-.apply-templates/»
                </table>
            </body>
        </html>
    </xsl:template»

    <xsltemplate match-"PLANET">
        <tr>
            <tdxxsl:value-of select-"NAME"/></td>
            <tdxxsl: apply-tempi ates select-"MASS"/></td>
            <td><xsl:apply-tempiates select-"RADIUS"/></td>
            <td><xsl:apply-templates select-"DAY"/»</td>
        </tr>
    </xsl:tempiate»

    <xsl-.template match-"MASS">
        <xsl:value-of select-" ."/>
        <xsl:text> </xsl:text>
        <xsl:value-of select-"@UNITS"/>
    </xsl:template»

    <xsl:template match-"RADIUS">
        <xsl:value-of select-"."/>
        <xsl:text> </xsl:text>
        <xsl:value-of select-"0UNITS"/>
    </xsl:template»

    <xsl-.template match-"DAY">
        <xsl:value-of select-"."/>
        <xsl:text> </xsl:text>
        <xsl:value-of select-"@UNITS"/>
    </xsl:template»

</xsl:stylesheet»
```

Вот результирующий файл XHTML:

```
<?xml version-"1.0" encoding-"UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional/ZEN"
    "http://www w3.org/TR/xhtmll/DTD/xhtmll-transitional.dtd"»
<html»
    <head»
        <title>
            The Planets Table
```

```
        </title>
    </head>

    <body>                                                    -1
        <nl>
            The Planets Table
        </hl>

        <table>
            <tr>
                <td>Name</td>
                <td>Mass</td>
                <td>Radius</td>
                <td>Day</td>
            </tr>

            <tr>
                <td>Mercury</td>
                <td>.0553 (Earth = l)</td>
                <td>1516 miles</td>
                <td>58.65 days</td>
            </tr>

            <tr>
                <td>Venus</td>
                <td>.815 (Earth - l)</td>
                <td>3716 m1les</td>
                <td>116.75 days</td>
            </tr>

            <tr>
                <td>Earth</td>
                <td>l (Earth - l)</td>
                <td>2107 mi 1es</td>
                <td>l days</td>
            </tr>
        </table>
    </body>
</html>
```

Полученный документ, planets.html, является и хорошо сформированным, и допустимым документом XHTML 1.0 — в соответствии с программой допустимости W3C для HTML и XHTML, расположенной по адресу: http://validator.w3.org/file-upload.html. Заметьте, что, поскольку документы XHTML являются в то же время хорошо сформированными XML-документами, мы воспользовались методом вывода XML, — поэтому преобразование ,не было слишком сложным; единственной сложностью оказалось создание элемента <!DOCTYPE>.

Изменение структуры документа на основе входных данных

До сих пор созданные мной шаблоны основывались на определенном жестком скелете, точно задающем, что должно попасть в выходной документ и в каком порядке. Но при помощи таких элементов XSLT, как <xsl:element>, <xsl:attribute>,

<xsl:text> и т. д., можно создавать новые узлы в рабочем порядке, на основе данных входного документа.

У вас уже есть небольшой опыт в этой области, поскольку мы работали с шаблонами значений атрибутов в главе 3. Как вы помните, при помощи таких шаблонов можно присвоить атрибуту значение выражения XPath, если заключить это выражение в фигурные? скобки, { и}. Например, чтобы установить атрибут NAME в строковое значение элемента <DESCRIPTION>, дочернего элемента контекстного узла, это значение можно присвоить так: NAMf«{DESCRIPTION}. Теперь мы можем более подробно рассмотреть всю тему создания йовых элементов и атрибутов «с ходу», начав с <xsl:element>.

Элемент <xsl:element>: создание новых элементов на этапе выполнения

Новые элементы можно создавать при помощи элемента <xsl:element>, который очень удобен для задания имени нового элемента на этапе выполнения.

У этого элемента три атрибута:

- name (обязательный). Имя создаваемого элемента. Принимает значение шаблона значений атрибута, возвращающего QName;
- namespace (необязательный). URI пространства имен нового элемента. Принимает значение шаблона значений атрибута, возвращающего URI;
- use-attribute-sets (необязательный). Задает наборы атрибутов, содержащие атрибуты этого элемента. Принимает значение списка элементов QName, разделенных символами-разделителями.

Элемент <xsl:element> содержит тело шаблона.

Пусть, например, мне нужно хранить названия планет в атрибутах NAME, а не в элементе <NAME> в planets.xml:

```
<?xml ver&ion»"1.0"?>
<?xml-stylesheet type-"text/xml"  href="plan6ts.xsi"?>
<PLANETS>

  <PLANET NAME-"Mercury">
      <MASS UNITS-"(Earth - 1)">.0553</MASS>
      <DAY UNITS-"days">58.65</DAY>
      <RADIUS UNITS="miles">1516</RADIUS>
      <DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
      <DISTANCE UNITS-"million miles">43.4</DISTANCE><!--B перигелии-->
  </PLANET>                                          \

                                                  jrifj b
```

Предположим теперь, что при помощи значений этого атрибута мне нужно создать имена новых элементов в результирующем документе - такие, как <Mercury>, <Venus> и <Earth>:

```
<?xml version-"1.0" encoding="UTF-8"?>
<?xml-stylesheet type-"text/xml" href-"planets.xsl"?*
<PLANETS>

  <Mercury>
    <MASS UNITS="(Earth • 1)"*.0553</MASS*
    <DAY UNITS="days">58.65</DAY>
    <RADIUS UNITS="mi 1 es">1516</RADIUS>
    «DENSITY UNITS»"(Earth - 1)"*.983</DENSITY*
    «DISTANCE UNITS-"mi11 ion miles">43.4</DISTANCE><!--В перигелии--<@062>
  «/Mercury*
```

В этом случае я не знаю имени выходного элемента до времени выполнения, потому и не могу просто применить элемент буквального результата. Я мог бы скомпоновать новый элемент, трактуя его как текст (что и показано в Примере ниже, где я вывожу символы, подобные «<», при помощи атрибута disable-output-escaping элемента <xsl:text>):

```
<?xml versioп-"1.0"?*
<xsl:stylesheet version="1.0"
xml ns: xsl «"http://www. w3. org/1999/XSL/Transform;">
<xsl: output method="xmT"/>

  <xsltemplate match-"@*|node()"*
    <xsl:copy>
      <xsl: apply-templ ates select-"(a*|node'(rb
    </xsl:copy>
  </xsl-.template*

  <xsl -.template match-'PLANET^
    <xsl:text disable-output-escaping="yes">&11;</xsl:text>
    <xsl:value-of select-"@NAME"/>
    <xsl:text di sable-output-escapi ng="yes">&gt;</xsl:text*
    <xsl:apply-tempiates/*
    <xsl:text disable-output-escaping="yes">&lt;/x$T:text*
    <xsl:value*of select-"eNAME"/>
    <xsl:text disable-output-escaping="ye^&gt;</xsT:texts»
  </xsl: tempi ate*

</xsl:stylesheet*
```

Но это грубый способ, при котором разметка рассматривается как простой текст. С другой стороны, зная название Планеты, Я могу создать новый элемент при помощи <xsl:element> (листинг 6.6), получив название новой планеты из атрибута name следующим образом.

Листинг 6.6. Применение <xsl:elem^t>)

```
<?xml version»"1.0"?>
<xsl:stylesheet version—"1.0"
xmlns xsl-"http://www.w3.org/1999/XSL/Transform">
*xsl:output, method-"xml"/>

' <xsl:tempi ate m a t<e|node(x"-
    <xsT:copy>
```

```
      <xsl:apply-templates select-"@*|node()"/>
    </xsl:copy>
  </xsl:tempi ate»

«xsl:tempiate match-"PLANET"»
    <xsl.-element name-"{$NAHE}">
        <xsl:apply-tempiates/>
    «/xsl:element»
</xsl:template»

</xsl:stylesheet»
```

Этот способ намного чище и проще. Ниже показан результат, в котором на этапе выполнения созданы новые элементы с именами различных планет:

```
<?xml version-"1.0" encoding-"UTF-8"?>
<?xml-stylesheet type-"text/xml" href*"planets xsl"?>
<PLANETS»

    <Mercury>
        <MASS UNITS-"(Earth - 1)"».0553</MASS»
        <DAY UNITS-"days"»58.65«/DAY>
        <RADIUS UNITS-"m1les"»l516«/RADIUS»
        «DENSITY UNITS-"(Earth - 1)"».983</DENSITY»
        «DISTANCE UNITS-"a1ll1on m1les">43.4</DISTANCE><«!–B перигелии-•»
    </Mercury>

    <Venus»
        <MASS UNITS-"(Earth - 1)"».815</HASS>
        <DAY UNITS-"days"»116.75</DAY»
        «RADIUS UNITS-"m1les"»37l6«/RADIUS>
        «DENSITY UNITS-"(Earth - 1)">.943«/DENSITY>
        «DISTANCE UNITS-"m1U1on mnes"»66.8</DISTANCE»<!–B перигелии–»
    «/Venus»

    «Earth»
        «MASS UNITS-"(Earth - 1)"»1«/MASS>
        «DAY UNITS-"days"»k«/DAY»
        «RADIUS UNITS-"m1les"»2l07«/RADIUS>
        «DENSITY UNITS-"(Earth - 1)"»1«/DENSITY>
        «DISTANCE UNITS-"n1ll1on miles"»128.4«/DISTANCE»«!–B перигелии–»
    «/Earth»

«/PLANETS»
```

Таким способом можно создавать новые элементы и задавать им имя вовремя преобразования XSLT.

Элемент <xsl:attribute>: создание новых атрибутов

Аналогично тому, как вы можете создавать новые элементы при помощи <xsl: element» и устанавливать имя и содержимое элемента ца этапе выполнения, при помощи элемента <xsl:attribute> это можно делать для атрибутов.

У элемента два атрибута:

- name (обязательный). Имя нового атрибута. Принимает значение шаблона значений атрибута, возвращающего QName;
- namespace (необязательный). Пространство имен нового атрибута. Устанавливается в URI.

Этот элемент содержит в себе тело шаблона, которое устанавливает значение атрибута.

В листинге 6.7 я создаю новые элементы <PLANET> с атрибутами, которые соответствуют различным названиям планет (значения берутся из атрибута COLOR исходных элементов <PLANET>).

Листинг 6.7. Применение <xsl:attribute>

```
<?xml version-*" 1.0"?>
<xsl:stylesheet version-'!.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

<xslrtemplate matCh-"PLANETS">
    <HIML>
        <HEAD>
            <ITITLE>
                Planets
            </ITITLE>
        <HEAD>
        <BDY>
            <xsl:apply-templates select-"PLANET"/>
        <B0DY>
    <HIML>
</xsl:template»

<xslrtemplate match="PLANET">
    <PLANET>
        <xsl:attribute name-"{NAME}">
            <xsl :v$fl ue-of select-"§C0L0R"/>
        </xsl:attribute»
    <PLANET»
</xsl:temolate>

</xsl:stylesheet»
```

Как можно видеть в приведенном ниже результате, я создал новые атрибуты «с ходу», используя названия планет:

```
<HIML>
    <HEAD>
        <ITITLE>
            Planets
        </ITITLE»
    <HEAD>

    <BDY>
        <PLANET Mercury-"RED">
        <PLANET»
        <PLANET Venus-"WHITE"»
        <PLANET»
```

```
            <PLANET Earth-"BLUE">
            <"/PLANET»
  , i   </60DY>

   </HTML>
```

Элемент <xsl:comment>: создание комментариев

По ходу дела можно также создавать и комментарии при помощи элемента <xsl:comment>. Этот элемент не имеет атрибутов и содержит тело шаблона, задающего текст комментария.

В листинге 6.8 я создаю комментарии для замены элементов <PLANET>; текст комментария включает название планеты.

Листинг 6.8. Применение <xsl:comment>

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"!.O"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

<xsl template match-"PLANETS">
    <HTML>
        <HEAD>
            <TITLE>
                Planets
            </TITLE>
        </HEAD>  (
        <BODY>
            <xsl apply-templates select-"PLANET"/>
        </BODY>                                                    v
    </HTML>
</xsl:tempiate>

<xsltemplate match-"PLANET">
    <xsl:comment>This was the <xsl:value-of select-"NAME"/> element</xsl:comment»
</xsl:template»

</xs>;stylesheet»
```

Вот результат:

```
<HTML>
    <HEAD>
        <TITLE>
            Planets
        </TITLE>
    </HEAD>

    <B0DY>
        <!--This was the Mercury element-->»
        <!--This was the Venus element-->»
        <!--This was the Earth element-->»
    </BODY>

</HTML»
```

Элемент <xsl:processing-instruction>: создание инструкций обработки

При помощи элемента <xsl:processing-instruction> можно создавать новые инструкции обработки. У этого элемента один атрибут:

- name (обязательный). Задает имя инструкции обработки. Принимает значение шаблона значений атрибута, возвращающего NCName.

В следующем примере я удалил инструкцию <?xml-stylesheet?> из начала planets.xml:

```
<?xml version-"1.0"?>

<PLANETS>

    <PLANET>
        <NAME>Mercury</NAME>
        <MASS UNITS-"(Earth - 1)">.0553</MASS>
        <DAY UNITS-"days">58.65</DAY>
        <RADIUS UNITS-"m1les">1516</RADIUS>
        <DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
        <DISTANCE UNITS-"million miles">43.4</DISTANCEx!--B перигелии-->
    </PLANET>

    <PLANET>
        <NAME>Venus</NAME>
        <MASS UNITS-"(Earth - 1)">.815</MASS>
        <DAY UNITS-"days">116.75</DAY>
        <RADIUS UNITS-"miles">3716</RADIUS>
        <DENSITY UNITS-"(Earth - 1)">.943</0ENSITY>
        <DISTANCE UNITS-"million miles">66.8</DISTANCE><!--B перигелии--»
    </PLANET>
```

Для того чтобы снова добавить эту инструкцию обработки, можно применить элемент <xsl:processing-instruction> (листинг 6.9). Пункты type и href в инструкции обработки, например <?xml-stylesheettype""text/xmr href-"planets.xsl"?>, в действительности не являются атрибутами, поэтому их значения нужно устанавливать не при помощи <xsl:attribute>, а используя простой текст.

Листинг 6.9. Применение <xsl: processing-instruction >

```
<?xml version-"1 0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">
<xsl:output method-"xml"/>

  <xsltemplate match-"/">
    <xsl:process1ng-1nstruct1on name-"xml-stylesheet"»
        <xsl:text>type-"text/xml" href-"pianets.xsl"</xsl: text>
    </xsl:processing-instruct1on>
    <x$l:apply-tempiateS/>
  </xsl:tempiate>
```

```
  <xslrtemplate match="@*|node()">
    <x$1;;copy>  *    .                    ;
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:tempi ate>

</xsl:stylesheet>
```

Вот результат, где инструкция обработки <?xml-stylesheet?> снова на месте:

```
<?xml version="1.0" encoding="UTF.8"?>
<?xml-stylesheet type="text/xml" href="planets.xsl"?>
<PUANETS>

    <PLANET>
        <NAME>Mercury</NAME>
        <MASS UNITS="(Earth = 1)">.0553</MASS>
        <DAV UNITS="days">58.65</DAY>
        <RADIUS UNITS="miles">1516</RADIUS>
        <DENSITY UNITS="(Earth = 1)">.983</DENSITY>
        <DISTANCE UNITS="million miles">43.4</DISTANCE><!--В леригелии-->
    </PLANET>

    <PLANET>
        <NAME>Venus</NAME>
        <MASS UNITS-"(Earth « 1)"> 815</MASS>
        <DAY UNITS="days">116.75</DAY>
        <RADIUS UNITS="miles">3716</RADIUS>
        «DENSITY UNITS="(Earth - 1)">.943</DENSITY>
        «DISTANCE UNITS="million miles">66.8</DISTANCE><!--At perihelion-->
    </PLANET>
```

Элемент <xsl:document>: создание нескольких выходных документов

В рабочем проекте XSLT 1.1 5ыл представлен новый элемент, <xsl:document>, предназначенный для поддержки нескольких выходных документов, — и, скорее всего,.этот элемент будет добавлен в XSLT 2.0. Он имеет следующие атрибуты:

- href (обязательный). Указывает место, в которое должен быть помещен новый документ. Устанавливается в абсолютный или относительный URI,* без идентификатора фрагмента;
- method (необязательный). Устанавливает метод вывода, используемый для создания результирующего документа. Устанавливается в «xml», «html», «text» или QName, которое не является NCName;
- version (необязательный). Задает версию выходного документа. Устанавливается в NMTOKEN; .„л " * i
- encoding (необязательный). Задает кодировку выходниго документа. Устанавливается в строку; i^v

- omit-xml-declaration (необязательный). Принимает значения «yes» или «no» для того, чтобы пропускать или не пропускать объявление XML;
- cdata-section-elements (необязательный). Определяет имена тех элементов, чье содержимое вы хотите вывести как разделы CDATA. Принимает значение разделенного символами-разделителями списка QName;
- doctype-public (необязательный). Определяет открытый идентификатор, который будет использован в объявлении <!DOCTYPE> вывода. Устанавливается в строковое значение;
- doctype-system (необязательный). Определяет системный идентификатор, который будет использован в объявлении <!DOCTYPE> вывода. Устанавливается в строковое значение;
- encoding (необязательный). Задает кодировку символов. Устанавливается в строковое значение;
- indent (необязательный). Определяет выравнивание вывода для отображения структуры вложенности. Устанавливается в «yes» или «no»;
- media-type (необязательный). Задает тип MIME вывода. Устанавливается в строковое значение;
- standalone (необязательный). Определяет, нужно ли включать в вывод отдельное объявление, и если да, задает его значение. Устанавливается в «yes» или «no».

Этот элемент содержит тело шаблона.

В следующем примере, основанном на упрощенной таблице стилей, я создаю в документе HTML две рамки (frame), и два HTML-документа, которые будут в них отображаться, frame1.html и frame2.html. Первую рамку и документ, который в ней появится, frame1.html, я создам при помощи <xsl:document> следующим образом (заметьте, что здесь я устанавливаю атрибут version в «1.1», поскольку мы используем возможность, входящую только в рабочий проект XSLT 1.1, но «1.1», вероятно, не будет правильным значением атрибута version в долгосрочном периоде; если элемент <xsl:document> будет включен в XSLT 2.0, версию следует установить в «2.0»):

```
<HTML xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xsl:version="1.1">
  <HEAD>
    <TITLE>
      Two Frames
    </TITLE>
  </HEAD>

  <FRAMESET cols="50%, 50%">
    <FRAME src="frame1.html"/>

    <xsl:document href="frame1.html">
      <HTML>
        <HEAD>
          <TITLE>
            Frame 1
          </TITLE>
        </HEAD>
```

```
            <BODY>
                <H1>Th1s 1s frame 1.</H1>
            </BODY>
        </HTML>
    </xsl'document*
```

После этого я могу создать вторую рамку и документ для вывода в нее, frame2.html (листинг 6.10).

Листинг 6.10. Применение <xsl:document>

```
<HIML xmlns:xsl-"http://www.w3.org/1999/XSL/Transform" xsl:version-"l.l">
    <HEAD>                                          • »
        <TITLE>
            Two Frames
        </TITLE>
    </HEAD>

    «FRAMESET cols-"50l. 50Г*

        <FRAME src-"framel.html"/>

        <xsl•document href-"framel.html">
            <HIML>
                <HEAD>
                    <TITLE>
                        Frame 1
                    </TITLE>
                </HEAD>

                <BODY>              ,
                    <Hl>This is frame 1.</H1>
                «/BODY*
            «/HTML*
        «/xsl:document*

      «FRAME src-"frame2.htmr/>

       «xsl:document href-"frame2.html"*
            «HTML*
                «HEAD*
                    «TITLE*
                        Frame 2
                    «/TITLE*
                «/HEAD*

                «BODY*
                    «Hl*This is frame 2.</Hl*
                «/BODY*
            «/HTML*
        «/xsl:document*

    «/FRAMESET*

«/HTML*
```

ПРИМЕР ТОЛЬКО ДЛЯ XSLT 1.1

Обратите внимание на то, что этот пример предназначен только для рабочего проекта XSLT 1.1. Ни один из известных мне доступных процессоров XSLT пока не обрабатывает элемент <xsl: document».

Элемент <xsl:namespace>: создание объявлений пространств имен

В XSLT 2.0 включен еще один новый элемент: <xsl:namespace>, позволяющий добавлять в результирующий документ объявления пространств имен» Однако на текущий момент больше об этом элементе ничего не известно, так что я не могу дать здесь более подробную информацию. Следите за web-узлом W3C.

Элемент <xsl:attribute-set>: создание наборов атрибутов

Иногда при создании нового элемента возникает потребность добавить в него сразу несколько атрибутов. Существует простой способ сделать это при помощи элемента <xsl:attribute-set>. У этого элемента два атрибута:

- name (обязательный). Имя набора атрибутов. Устанавливается в QName;
- use-attribute-sets (необязательный). Имена других наборов атрибутов, которые вы хотите включить в этот набор. Устанавливается в список QName, разделенных символами-разделителями.

Элемент <xsl:attribute-set> содержит элементы <xsl:attribute>, по одному для каждого нового создаваемого атрибута. При использовании <xsl:attribute-set> для создания нового набора атрибутов элемента нужно задать имя набору. Затем при создании нового элемента для использования набора атрибутов это имя можно присваивать атрибуту use-attribute-sets элементов <xsl:copy>, <xsl:element>, <xsl:for-each> и даже самого элемента <xsl:attribute-set>.

Ранее в текущей главе мы уже рассматривали один пример применения наборов атрибутов в JavaScript при создании шаблона. В том примере я при помощи набора атрибутов задавал все атрибуты HTML-кнопок планет и затем создавал их, используя этот набор атрибутов в элементе <xsl:element>:

```
<BODY>
    <CENTER>
        <H1>The Mass Page</H1>
    </CENTER>
    <xsl:for-each select-"PLANET"»
        <P/>
        <xsl:element name»"input" use-attribute-sets«"attribs"/>
    </xsl:for-each>
    <P/>
    <P/>
```

```
<BIV ID=J'display'x/f)IV>
!                    J'V; -ir
<HIML>
... <7xs.l:template»

<xsl:attribute-set name'"'attribs">
<xsl:attribute                                   ; r                ГКЗ^ЭПб.
<xslrattribute name-"value"><xslrvalue-of select>"NAME"/></xsl:attribute>
<xsl: attribute name-"oncl1ck"*^{^«1u^f selet^-HAhE*/Ki</x$V$-attribute» tcfdQ
</xsl:attribute-set>
```

В результирующем документе такой йабор атрибутов был добавлен каждой кнОПке HTML! • ' • ..\lt:«-«rwv.. • -

```
<P></P>
<input type*'BUTION" val1ue-"Mercury" oriel ick»"Mercury()">
<P></P>
<input type-'BUTTON" value-"Venus" onclick-"Venus()">
<P></P>      . ^r     'VI      5.-.  , - ;                    '- * *"
<i nput type-"6uTror vatufe-"4V £arth" brfcti ck="Earth() ">
```

В листинге 6.11 я воспользуюсь набором атрибутов для нумерации плакеТ в planets.xml. Каждому элементу <PLANET> я добавил два атрибута: number и total В атрибуте number будет храниться номер планеты начиная с 1, а атрибут total будет содеркать общее чиСло планет в planets.xml, которое можно найти пр^ помощи функции count (мы рассмотрим ее в главе 8).

Листинг 6.11. Применение <xsl:attribute-set>

```
<xs,l:stylesheet          ••      nv
 xml ns-xsl ^"httD://www. w3. org/1999/XSU-Transform"
 verslorM'.oV'   '4

<xsl;output method-"xml"  inctent^'yes"/»                  w          ,,,if;

<xs»! «ёчйвбе: ий&ГК*"*
   !<xsbcqpy*      ,  .          5».                                  -У} K.I..
      <xsl:apply-templates/»
   </x£1  copy>
</xsl template>

<xsT:template match-'PLANET'>  .      .          ь
    <xsl r copy use-attribute-sets-"numbering"»
 nj'io».^«xslrapply-templates/» •
 >•! «/xskcopy»    '  i      «.>u.vr  hi,          • н
</xsl:template»                  .l»•>:        '.-ипк          • •>?'

<xsl:attribute-set name-"numbering">
    <xslattribute  name-"number"><xsl:number/></xsl:attribute>
    <xslrattribute  narae-"total"><xsl:value-of  select-"count(//PLANET)"/>
    </xsl:attribute»
</xsl: attri bute-set>                                  '  ..'    '<
                  >                                                  4
</xsl:stylesheet»          -Js'wirJ•sJc.jr-'iJJf»          энопп ;пэта-  (гх-
```

Обратите внимание на то, что в результирующем документе каждый элемент <PLANET> имеет атрибуты number и totaL ^

```
«'xml  version-" 1.0" encoding-"UTF8M?>
«PLANETS*

    «PLANET number-"1" total-"3">
        <NAME>Mtacury</NAME>
        <MASS*0553«/MASS>
        <DAY>5865</DAY>
        <RA0IUS>1516</RADIUS>
        <DENSITY>983</DENSITY*
        <DISTANCE>434</DISTANCE>
    </PLANET>

    «PLANET number="2" total-*3">
        <NAME>Venus«/NAME>
        <MASS>815</MASS>      /.
        <DAY>11675«/DAY>
        <RADIUS>3716</RAD IUS>
        «DENSITY*943</DENSITY>
        <DISTANCE>668</DISTANCE>
    «PLANET*

    «PLANET number-'"3" total =«'3'>
        «NAME*Ea rth</NAME*
        <MASS*1«/MASS>
        «DAY*1«/DAY*
        <RADIUS*2107«/RADIUS>
        <DENSITY*1«0ENSITY*
        <DISTANCE*128 4«/DISTANCE*
    «PLANET*

«PLANETS*
```

Пропуск объявления XML и создание фрагментов XML

При знакомстве с XSLT многих смущает объявление XML, которое всегда появляется наверху их результирующих документов, особенно если они не отдают себе OTnei в том, что методом вывода по умолчанию является XML. Разумеется, существует способ избавиться от объявления XML, что может быть полезно при создании хорошо сформированных фрагментов XML, не обязательно являющихся законченными документами. Достаточно только установить атрибут omit-xml-declaration элемента <xsl:output> в «yes»: <xsl:output method="xml" omit-xml-declaratioiWyes"/».

В следующем примере я только уберу из документа объявление XML, копируя при этом все остальное. Заметьте, что объявление XML не есть узел, поэтому ничего страшного, если оно будет удовлетворять копирующему шаблону:

```
<?xml  version-"1  O"?*
<xsl  stylesheet vers"i on*" 1 0"
xmlns:xsl-"http://www w3.org/1999/X^L/Transform"*
«xsl:output  method-"xml"  omit-xml-declaration-"yes"/>
```

```
<xs1:tempiate match-"0*|nodeO">
  <xsl:copy>
    «xsl:apply-tempiates select-"@*|node()"/>
  </xsl:cdpy>
</xsl:template*

</xsl:stylesheet*
```

Эта таблица стилей преобразует planets.xml:

```
<?xml version-"1.0"?>
«PLANETS*

    «PLANET*
        <NAME*Macury</NAME*
        «MASS UNITS-"(Earth - 1)"*.0553</MASS*
        «DAY UNITS»"days">58.65«/DAY*
        <RADIUS UNITS-"miles"*1516«/RADIUS*
        «DENSITY UNITS-"(Earth - 1)">.983«/DENSITY>
        «DISTANCE UNITS=»"million m1les"*43 4«/DISTANCE*«!–B перигелии–*
    «PLANET*

    «PLANET*
        <NAME*Venus«/NAME>
        «MASS UNITS-"(Earth - 1)"*.815«/MASS*
        «DAY UNITS-"days"*116 75</DAY*
        «RADIUS UNITS-"mileS"*3716</RADIUS>
        «DENSITY UNITS-"(Earth - 1)"*.943«/DENSITY*
        «DISTANCE UNITS-"million miles">66.8«/DISTANCE*«!–B перигелии–*
    «PLANET*
```

в новую версию без объявления XML:

```
«PLANETS*

    «PLANET*
        <NAME*Macury</NAME*
        «MASS UNITS-"(Earth - 1)"*.0553«/MASS*
        «DAY UNITS-"days">58.65«/DAY*
        «RADIUS UNITS=-'miles"*1516«/RADIUS*
        «DENSITY UNITS-"(Earth - 1)"*.983«/DENSITY*
        «DISTANCE UNITS-"million miles"*43.4«/DISTANCE*«!–B перигелии–<@062>
    «PLANET*

    «PLANET*
        <NAME*Venus«/NAME*
        «MASS UNITS-"(Earth - 1)"*.815«/MASS*
        «DAY UNITS-"days"*116 75«/DAY*
        «RADIUS UNITS-"m1les"*3716«/RADIUS*
        «DENSITY UNITS-"(Earth - 1)"*.943«/DENSITY*
        «DISTANCE UNITS-"million miles">e6.8</DI5TANCE*«!~B перигелии–*
    «PLANET*
```

Полезно знать об этом при создании фрагментов XML или выполнении другой работы. Тем не менее, заметим, что во всех законченных документах XML[1] — даже написанных самостоятельно при помощи разнообразных приложений XML, таких как WML — в начале обязательно должно быть объявление XML.

Создание уникальных идентификаторов при помощи generate-id

При существенном изменении структуры документов важно рассмотреть еще одну тему: как в результирующих документах создавать идентификаторы элементов для обращения к ним при необходимости. Представьте, например, что вам нужно при помощи таблицы стилей добавить в документ оглавление, записи в котором будут представлять собой гиперссылки, так чтобы пользователь по щелчку на них сразу переходил к нужному разделу. В этом случае потребуется какой-то способ идентификации элементов в результирующем документе, в чем поможет функция generate-id.

В следующем примере я добавляю составленное из гиперссылок оглавление в planets.html. Для создания оглавления я прохожу в цикле по всем планетам при помощи элемента <xsl:for-each>. На каждом шаге цикла я создаю гиперссылку и при помощи шаблона значений атрибута создаю атрибут HREF, который устанавливается в уникальный идентификатор для рассматриваемой планеты. Заметьте, что, несмотря на свое имя, функция generate-id создает только строковый идентификатор элемента, она не создает атрибуты ID:

```
<?xml version«"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns xsl-"http://www.w3.org/1999/XSL/Transform">

    <xsltemplate match-"/PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <BODY>
                <H1>
                    The Planets Table
                </H1>
                <xsl:for-each select-"PLANET">
                    <H2><A HREF-"#{generate-id()}">
                        <xsl: val ue-of select-"NAME"/x/A>
                    </H2>
                    <P/>
                </xsT:for-each>
```

Эта таблица стилей (листинг 6.12) генерирует для каждой планеты идентификатор и создает требуемые гиперссылки. Функция generate-id не только создает для элемента новый идентификатор, но и возвращает его при последующем применении generate-id к этому элементу. В данном случае это удобно, поскольку таким образом я могу создать закладки гиперссылки в HTML-таблице данных планет, установить по очереди атрибут <NAME> закладки в идентификатор для каждого элемента <PLANET> -у так, чтобы он стал гиперссылкой-назначением.

Листинг 6.12. Применение функции generate-id

```
<?xml version-"l Q"?>
<xsl:stylesheet version-"1.0"
xml,ns: xsl -"http; //www.. w3 .org/1999/XSL/Transform">

    <xsl template match-"/PLANEtS">
        <HIML> -          *      n
            <HEAD>
                <ITTLE>
                    the Planets Table
                </ITTLE>
            </HEAD>
            <BODY>
                <HI>
                    The Planets Table
                </HI>
                <xsl:for-each select-"PLANET">
                    <№><A HREF- "#{geherate -i d O}" >
,(                      r   <X9lvval;uerOf select-"NAME"/></A>
                    </H2>
                    <P/>
                </xsl':for-6ach>

                <TABLE BORDER-'2'>
                    <TR>
                        <TD>Name</TD>
                        <TD>Mass</TD>
                        <TD>Radius</TD>                                    г
                        <TD>Day</TD>
                    </TR>
                    <xsl:apply-templates/>
                </TABLE>
            </BODY>
        </HIML>
    </xsl:tempiate>

    <xsl:tempiate match-'PLANET'»
        <TR>
            <TDxA NAME-"{generate-id(. )}">
            <xsl:value-of select-"NAME"/></A></TD>
            <TDxxsl: apply-templ ates select-"MASS"/x/TD>
            <TD><xsl: apply-templ ates select-*RADIUS"7x/T0>
            <TDxxsl: apply-templ ates select*"DAY"/></TD>
        </TR>
    </xsl:template»

    <xsl:tempiate match-'MASS'>
        <xsl:value-of select-"."/»
```

продолжение &

Листинг 6.12 (продолжение)

```
        <xsl:text> </xsl:text>
        <xsl-.value-of select-"@UNITS"/>
    </xsl:template»

    <xsl:template match»"RADIUS">
        <xsl value-of select-"."/>
        <xsl:text> </xsl:text>
        <xsl:value-of select-'^UNITS"/»
    </xsl:template»

    <xsl:template match-"DAY"\»
        <xsl:value-of select-"."/»
        <xsl:text> </xsl:text>
        <xsl :value-of select="@UNITS"/>
    </xsl:tempiate»

</xsl:stylesheet»
```

Вот как это делается: сейчас я создал гиперссылки с атрибутом HREF, значение которого равно идентификатору элемента <PLANET>; при помощи этого же идентификатора я сделал каждый элемент <PLANET> назначением гиперссылки. Когда пользователь щелкает на гиперссылку в оглавлении, браузер прокручивает данные до соответствующей записи планеты в HTML-таблице. (Заметьте, что для того чтобы большинство браузеров осуществляли прокрутку, HTML-таблица должна быть вне пределов экрана.) Каждый процессор XSLT создает свои собственные идентификаторы; ниже приведен результат для процессора Xalan:

```
<HTML>
    <HEAD>
        <TITLE>
            The Flanets Table                                    4
        </TITLf>,
    <fHEAD> "

    <BODY>
        <Hl»
            The Planets Table
        </Hl>
        <H2»
        <A href-"#N5"»Mercury</A>
        </H2»
        <P»</P»
        <H2»
        <A href-"#N20"»Venus</A»
        </H2>
        <P»</P>
        <H2>
        <A href-"#N3B"»Earth</A»
        </H2»
        <P»</P»

        <TABLE BORDER-?'\»
            <TR>
                <TD»Name</TD»
                <TD»Mass</TD>
              4 <To>Radius</TD>
```

```
            <TD>Day</TD>
        </TR>

        <TR>
            <TD><A NAME="N5">Mercury</A>
            </TD><TD>.0553 (Earth - 1)</TD>
            <T0>1516 miles</TD>
            <T0>58.65 days</TD>
        </TR>

        <TR>
            <TD><A NAME="N20">Venus</Ax/TD>
            <TD>.815 (Earth - 1)</TD>
            <TD>3716 miles</TD>
            <TD>116.75 days</TD>
        </TR>

        <TR>
            <TD><A NAME="N3B">Earth</A></TD>
            <TD>1 (Earth - 1)</TD>
            <tD>2107 miles</TD>
            <TD>1 days</TD>
        </TR>

    </TABLE>
</BODY>
</HTML>
```

Результат показан на рис. 6.3 (в том числе — оглавление из гиперссылок). Пользователю достаточно щелкнуть на гиперссылке к соответствующей записи в таблице.

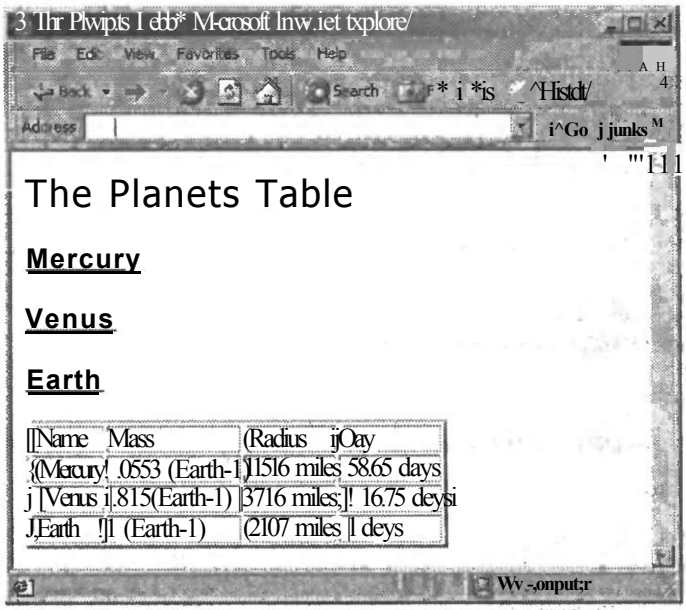


Рис. 6.3. Использование сгенерированных идентификаторов в гиперссылках

Создание разделов CPATA

При преобразованиях XML—XML может потребоваться вывести разделы CDATA. В XSLT это легко осущесвимо при помощи атрибута cdata-section-elements элемента <xst:output>. Упомянутый атрибут позволяет указать, содержимое каких элементов следует заключить в раздел CDATA. Последнее может понадобиться, например, при создании элементов сценария, когда браузер требует заключения кода сценария в раздел СБАТД.

В следующем примере я поместил содержимое элементов <NAME> И <MASS> в planets.xml в разделы CDATA:

```
<?xrrl vers ion-"1.0"?»
«xsl;stylesheet version-"l 0"
xmlns:xsl-"http://WWW.w3.org/19^9/XSL/Transform">
<xsl:output method-"xml" cdata-section-elements-"NAME MASS'\»

  «xsl: tempi ate match-"!?* | node() ">
    «xskcopy»
      <xsl:apply-templates  select-"@*|node()"/>
    </xsl:copy»
  «/xsl  template»

</xsl:stylesheet»
```

И Вот результат:

```
<^xrl verslon-^l-O"  encoding-"UTF-8"?»
<?xml-stylesheet type-"text/kiii1"  href-"planets.xsl"?>
<PLANETS»

    <PLANET»
        <NAME»<![CDATA[Mercury]]»</NAME»
        <MASS  UNITS-"(Earth  -  1)*»<![CDATA[.0553]]»</MASS>
        <DAY UNITS-"days"»58.65</DAY>
        <RADIUS UNITS="miles"»1516«/RADIUS>
        «DENSITY  UNITS-"(Earth  -  1)"»983</DENS1TY>
        «DISTANCE  UNITS-"million miles"»43  4</DISTANCE»<!-B перигелии--»
    «PLANET»

    «PLANET»
        <NAME»«![CDATA[Venus]]»</NAME»
        «MASS  UNITS-"(Earth  -  1)"»«![CDATA[.815]]»«/MASS»
        «DAY UNITS-"days"»116.75«/DAY>
        «RADIUS UNITS-"miles"»3716«/RADIUS>
        «DENSITY  UNITS-"(Earth  -  1)",» 943«/DENSITY>
        «DISTANCE  UNITS-"million mnes">66.8«/DISTANCE»«!--B перигелии--»
    «PLANET»
```

ОБРАБОТКА CDATA В ИСХОДНОМ И РЕЗУЛЬТИРУЮЩЕМ ДОКУМЕНТЕ

Рассмотренная техника создает разделы CDATA результирующего документа, она не рассматривает как CDATA какие-либо данные исходного документа. Например, если вам нужно преобразовать <script>if x < y {...}</script> в <scriptx![CDATA[if x < y {...}]]></script>, поскольку ваш браузер требует, чтобы код сценария заключался в разделы CDATA, процессор XSLT испытает трудности с символом < в «x < y». В этом случае необходимо написать <script>if x < y {...} </script>, для того чтобы процессор XSLT сгенерировал <script><![CDATA[if x < y {...}]]> </script>.

Установка кодировки символов

Для значительных переработок исходных документов важно знать, что можно устанавливать кодировку символов в результирующих документах при помощи атрибута encoding элемента <xsl:output>. Однако нет гарантии, что ваш процессор XSLT будет поддерживать выбранную кодировку, поскольку процессоры обязаны поддерживать только кодировки UTF-8 й UTF-16. С другой стороны, если вы используете символ, не поддерживаемый в используемой кодировке, процессор XSLT либо выведет символ как ссылку на сущность, либо Сгенерирует ошибку.

ЕЩЕ О КОДИРОВКАХ СИМВОЛОВ

Принятые кодировки символов указаны в рекомендации XML 1.0, www.w3.org/ TR/REC-xml.

Атрибут encoding не обязателен в объявлении XML документа; если он отсутствует, разборщики XML по умолчанию принимают кодировку UTF-8. Если вы хотите явно задать кодировку или использовать другую, ее можно задать следующим образом — там, где я задаю кодировку UTF-16:

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xslhttp://www.w3.org/1999/XSL/Transform">
<xsl:output methocK'xml" encoding-"UTF-16"/>

  <xsl:template match-"@*|node()">
    <xsl:copy>
      <xsl:apply-templates select-^lnodeO"/»
    </xsl:copy>
  </xsl:template»

</xsl:stylesheet»
```

Вот что появится в выходном документе:

```
<?xml version-"1.0" encodincp'UTF-16"?>
<?xml-stylesheet type-"text/xml" href="planets.xsl"?>
<PLANETS>

    <PLANET>
        <NAME>Macury</NAME>
        <MASS UNITS-"(Earth - 1)">.0553</MASS>
        <DAY UNITS-"days">58.65</DAY>
        <RADIUS UNITS-"m1les">1516</RADIUS>
        <0ENSITY UNITS-"(Earth - 1)">.983</DENSITY>
```

```
    <DISTANCE UNITS="nnl1ion miles">43.4</DISTANCE><!–В перигелии–>
</PLANET>

<PLANET>
    <NAME>Venus</NAME>
    <MASS UNITS-"(Earth - 1)">.815</MASS>
    <DAY UNITS-"days">116.75</DAY>
    <RADIUS UNITS-"miles">3716</RADIUS>
    <DENSITY UNITS-"(Earth - 1)">.943</DENSITY>
    «DISTANCE UNITS-"million miles">66.8</DISTANCE><!–В перигелии–»
</PLANET>
```

Режимы: форматирование в зависимости от контекста

В некоторых случаях выбор узлов может требовать буквально мастерства. Например, шаблон с условием выбора «*» выбирает и «PLANET». Если вам нужно обрабатывать элементы <PLANET> по-другому, чем все остальные элементы, вы можете задать шаблон, выбирающий «PLANET»; тогда процессор XSLT определит, что «PLANET» ближе к элементу <PLANET», чем <•*», и воспользуется правилом выбора «PLANET». Но не всегда все так просто. Например, вам может быть нужно, чтобы правило «PLANETS» выбирало не все, а лишь некоторые элементы <PLANET>. Один из способов решения этой задачи — воспользоваться *режимами* (mode).

Для задания режима обработки служит атрибут rjiode элемента <xsl:apply-templates>. При задании режима будут применяться только те шаблоны, элементы <xsl:template>, атрибут mode которых установлен в тот же режим. Таким образом, при помощи шаблона можно выбрать только один определенный узел, даже если в ином случае они бы выбирали много узлов. Это очень удобно, если, например, нам нужно обработать какие-то элементы <PLANET> отлично от других.

Рассмотрим пример. Здесь я создал новый режим обработки данных планет «fancy» (красивый), отображающий данные полужирным шрифтом. Но я хочу, чтобы полужирным выделялись только данные Земли, а не других планет. Это может составить проблему: поскольку все данные планет хранятся в одном типе элемента — <PLANET>, — постольку образец «PLANET» выберет их все. Режим решает это затруднение. В частности, если текущая планета — Земля, я могу-установить режим обработки в «fancy»:

```
<?xml version-"l 0"?>
<xsl: styl esheet version-"!..O"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

    <xsltemplate match-"/PLANETS">
        <HIML>
            <HEAD>
                <ITTLE>
                    The Planets Table
```

```
            </ITITLE>
        <HEAD>
        <BODY>
            <HI>
                The Planets Table
            </HI>
            <TABLE BORDER-'2'>
                <IR>
                    <ID>Name</ID>
                    <ID>Mass</ID>
                    <TD>Radlus</T0>
                    <ID>Day</ID>
                </IR>
                <xsl apply-templates/>
            </TABLE>
        <BODY>
    </HTML>
</xsl:template'N >r .Я                                    Ц     UA

<xsl template match-'PLANET'>
    <xsl:1f test-"NAME-'Earth'">
    <IR>
        <TD><xsl:apply-tempiates select-'NAME" mode-"fancy"/></T[>
        <TDxxsl: apply-tempi ates select-"HASS" mode-" fancy "/></TD>
        «m><xsl:apply-tempiates select-"RADIUS" mode-"fancy"/>«/TD>
        <ID><xsV: apply-tempi ates select-"DAY" roode-"fancy"/x/7D>
    </IR>
    </xsl:1 f>
    <xsl:1f test--NAHE!-'Earth'">
    <IR>
        <TD><xs1:apply-templates select-"NAME"/></TD>
        <TDxxsl: apply-tempi ates sel ect-"MASS"/></TD>
        <TD><xsl: apply-templatfes select-"RADIUS"/x/TD>
        <TDxxsl: apply- templ ates select›"DAY"/x/TD>
    </IR>
    </xsl:1f>
</xsl:template»
```

После этого я устанавливаю нужные правила шаблона (листинг 6.13), оба для режима обработки «fancy», который выделяет выходные данные полужирным, и не задаю никакого специального режима обработки.

Листинг 6.13. Применение режимов

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

    <xsl-.template match-"/PLANETS">
        <HIML>
            <HEAD>
                <ITITLE>
                    The Fancy Planets Table '
                </ITITLE>
```

продолжение &

Листинг 6.13 (продолжение)

```
            </HEAD>
            <BODY>
                <H1>
                    The Fancy P!anets Table
                </H1>
                «TABLE BORDER-'2'">
                    <TR>
                        <TD>Name</TD>
                        <TD>Mass</TD>
                        <TD>Radius</TD>
                        <TD>Day</TD>
                    </TR>
                    <xsl:apply-tempiates/>
                </TABLE>
            </BODY>
        </HTML>
    </xsl:tempiate>

    <xsltemplate match-"PLANET">
        <xsl:if test-"NAME-'Earth'">
        <TR>
            <TD><xsl:apply-templates select-"NAME" mode-"fancy"/»</TD>
            <TDxxsl: apply-templates select-"MASS" mode-"fancy'7></TD>
            <TDxxsl: apply-templ ates select-"RADIIlS" mode-"fancy'7x/T0>
            <TD><xsl:apply-tempiates select-"DAY" mode-"fancy"/></TD>
        </TR>
        </xsl:if>
        <xsl:if test-"NAME!-'Earth'">
        <TR>
            <TD><xsl:apply-tempiates select-"NAME"/></TD>
            <TDxxsl: apply-templ ates sel ect-"MASS"/x/TD>
            <TDxXsl: apply-templ ates select-"RADIUS"/x/TD>
            <TDxxsl: apply-templ ates select-"DAY'7></TD>
        </TR>
        </xsl:if>
    </xsl:template»

    <xsl:tempiate match-"NAME">
        <xsl:value-of select-"."/>
    </xsl:tempiate>

    <xsl:tempiate match-"MASS">
        <xsl:value-of select«"."/>
        <xsl:text> </xsl:text>
        <xsl:value-of select-"PUNITS'7>
    </xsl:template»

    <xslrtemplate match-'RADIUS"»
        <xsl:value-of select-"."/»
        <xsl:text> </xsl:text>
        <xsl:value-of select-"@UNITS"/>
    </xsl:tempiate>

    <xsl:tempiate match-"DAY">
        <xsl:value-of select-"."/»
```

```
        <xsl :text> </xsl :text>                                  i4          • im -nt»
        <xsl: val ue-of select-"@UNITS"/»
    </xsl:tempiate>

    <xslrtemplate match-"NAME" mode-"fancy"»
        <B><xsl:value-of  select-"."/x/B>
    </xsl:template»

    <xsl:tempiate match-"MASS" mode-"fancy"»
        <B>
            <xsl:value-of  select-"."/>
            <xsl:text> </xsl:text>
            <xsl:value-of  select="PUNITS"/>
        </B>
    </xsl:tempiate>

    <xslrtemplate match-"RADIUS" mode-"fancy'>
        <B>
            <xsl:value-of  select-"."/>
            <xsl:text> </xsl:text>
            <xsl:value-of  select-"@UNITS"/>
        </B>
    </xsl:template*

    <xslrtemplate match-"DAY" mode-"fancy"»
        <B>
            <xsl:value-of  select-"."/>
            <xsl:text> </xsl:text>
            <xsl:value-of  select-"§UNITS"/>
        </B>
    </xsl:template»

</xsl:stylesheet»
```

Вот результат. Заметьте, что одни только данные Земли выделены полужирным:

```
<HTM>
    <HEAD>
        <ITTLE»
            The Fancy Planets Table
        </ITTLE>
    <HEAD>

    <BODY»
        <Hl»
            The Fancy Planets Table
        </H1»

        <TABLE BORDER='2'>
            <TR>
                <TD»Name</T0>
                <TD»Mass</TD>
                <TD»Radius</TD>
                <TD»Day</TD>
            </TR>

            <TR>
```

v -)lчr to-aufs»

```
            <TD>Mercury«/TD>
            <TD>.0553 (Earth - 1)</TD>
            <TD>1516 mi1e$</TD>
            <TD>58.65 days</TD>
        </TR>

        <TR>
            <TD>Venus</TD>
            <TD>.815 (Earth - 1)</TD>
            <TD>3716 miles</TD>
            <TD>116.75 days</TD>
        </TR>

        <TR>
            <TD><B>Earth«/B></TD>
            <TDxB>I (Earth - 1)</B></TD>
            <TD><B>2107 m1les</B></TD>
            <TD><8>1 days</Bx/TD>
        </TR>

    </TABLE>
</BODY>
</HTML>
```

Этот документ показан на рис. 6.4.

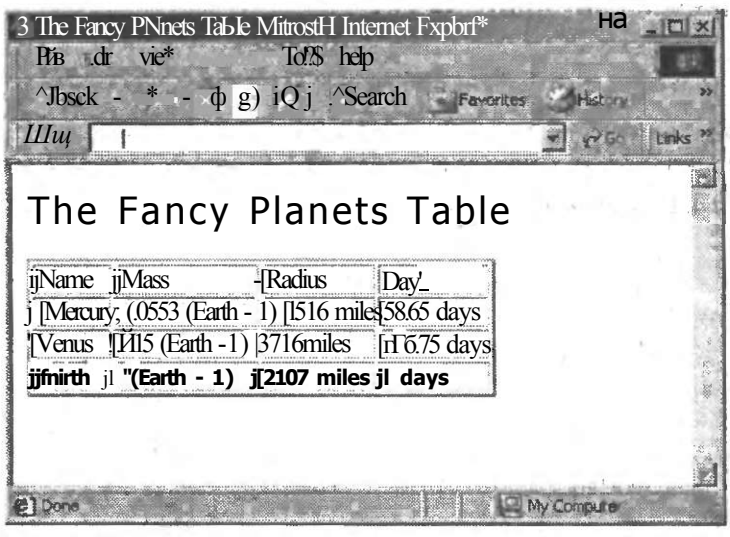


Рис. 6.4. Применение режимов обработки

Режимы также очень удобны в том случае, когда требуется обработать один и тот же документ более одного раза; классический пример гг создание оглавления.

Следующий пример демонстрирует создание оглавления. Я добавил в planets, xml оглавление в элементе <T0C> (table of contents, оглавление), имеющем три элемента <ENTRY> для каждой из планет. Заметьте, что здесь мне необходимо применить два шаблона, выбирающие элементы <PLANET>, — один для создания

оглавления и еще один для копирования всех элементов <PLANET> в результирующий документ — я использую режимы для того, чтобы их различать. Начну с установки режима в «too и применения шаблона, который поддерживает этот режим и создает оглавление:

```
<?xml vers1on-"l.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http //www.w3.org/1999/XSL/Transform">
<xsl: output method-"xmV7>

  <xsl:template match-"PLANETS">
    <PLANETS>
        <TOC>
            <xsl:apply-tempiates mode-"toc"/>
        </TOC>


    </PLANETS>
  </xsl:template»

  <xsl:tempiate match-"PLANET" mode-"toc">
    <ENTRY>
        <xsl:value-of select-"NAME"/>
    </ENTRY>
  </xslrtemplate»
```

Затем я применяю общий шаблон без каких-либо режимов обработки ко всем элементам и атрибутам, копируя их в результирующий документ (листинг 6.14).

Листинг 6.14. Создание оглавления

```
<?xml vers1on-"l.0"?»
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www w3 org/1999/XSL/Transform"»
<xsl:output method-"xml"/>

  <xsltemplate match-"PLANETS"»
    <PLANETS»
        <TOC>
            <xsl:apply-templates mode-"toc"/>
        </TOC>
        <xslrapply-tempiates select-"0*|node()"/»
    </PLANETS»
  </xslrtemplate»

  <xsltemplate match-"PLANET" mode-"toc">
    <ENTRY>
        <xsl:value-of select-"NAME"7»
    </ENTRY»
  «?/xsl: tempi ate»

  <xsl:template match-"node O"»
    <xslrcopy»
```

Листинг 6.14 (продолжение)

```
      <xsl:apply-templates select-H@*|node()"/»
    </xsl:copy>
  </xsl:template»

</xsl:stylesheet»
```

И, наконец, результат, выводящий planets.xml с оглавлением:

```
<?xml version-"1.0" encoding-"UTF-8"?>
<?xml-stylesheet type-"text/xml" href-"planets.xsl"?>
<PLANETS>
    <TOC»
        <ENTRY»Mercury</ENTRY»
        <ENTRY»Venus</ENTRY»
        <ENTRY»Earth</ENTRY»
    </TOC»

    <PLANET»
        <NAME»Mercury</NAME>
        <MASS UNITS-"(Earth - 1)">.0553</MASS>
        <DAY UNITS-"days">58.65</DAY>
        <RADIUS UNITS-"mi 1es">1516</RApIUS>
        <DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
        OLSTANCE UNITS-"million mi 1 es">43. 4</DISTA(CE»<! -В перигелии-»[1]
    </PLANET»



</PLANETS»                                                    ь
```

В этой главе мы рассмотрели большую часть базового материала, и в следующей главе я собираюсь уделить внимание созданию полных выражений Xl?ath. Хотя вы уже немного знакомы с этой темой по обсуждению образцов выбора в главе 4, нам осталось еще рассмотреть весьма значительную часть.

7

Работа с XPath

В главе 4 было объяснено, как создавать образцы выбора, которые можно использовать в атрибуте match таких элементов, как <xsl:template». Образцы являются подмножеством полного языка XPath, и в этой главе будет рассмотрена полная версия XPath.

Выражения XPath применимы в XSLT не только в образцах выбора, но и во многих других приложениях: в атрибуте select элементов <xsl:apply-templates>, <xsl:value-of>, <xsl:for-each>, <xsl:param>, <xsl:variable>, <xsl:with-param>, <xsl:copy-of> и <xsl:sort>, в шаблонах значений атрибутов, в атрибуте test элементов <xsl:if> и <xsl:when>, атрибуте value элемента <xsl:number> и в предикатах образцов выбора. На этот счет не может быть никаких сомнений: пока вы не знаете XPath, вы не знаете XSLT, и вся эта глава посвящена созданию выражений XPath и работе с ними.

Фактически мы работали с выражениями XPath начиная с главы 1, в которой мы впервые познакомились с атрибутом select в элементах <xsl:apply-temptates> и <xsl:value-of>:

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www w3 org/1999/XSL/Transform">

    <xslrtemplate match="/PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            <HEAD>
            <BODY>
                <H1>
                    The Planets Table
                </H1>
                <TABLE BORDER-'2'>
```

```
                    <TR>
                        <TD>Name</TD>
                        <TD>Mass</TD>
                        <TD>Radius</TD>
                        <TD>Day</TD>
                    </TR>
                    <xsl apply-templates/>
                </TABLE»
            </BODY>
        </HTML>
    </xsl:template»

    <xsl.template match="PLANET">
        <TR>
            <TD»<xsl:value-of select-"NAME"/></TD>
            <TD><xsl:apply-tempiates select-"MASS"/></TD>
            <TD><xsl:apply-tempiates select-"RADIUS"/></TD>
            <TD><xsl:apply-tempiates select-"DAY"/></TD>
        </TR>
    </xsl template»

    <xsl:template match»"MASS">
        <xsl:value-of select-"."/>
        <xsl:text> </xsl:text>
        <xsl:value-of select-"@UNITS"/»
    </xsl:tempi ate»

    <xsl:tempi ate match-"RADIUS">
        <xsl:value-of select-"."/>
        <xsl:text> </xsl:text>
        <xsl:value-of select-"@UNITS"/»
    </xsl .-template»

    <xsl:tempi ate match-"0AY">
        <xsl:value-of select-"."/»
        <xsl:text> </xsl:text>
        <xsl:value-of select-'GUNITS"/»
    </xsl:tempiate»

</xsl -.stylesheet»
```

Изучаем XPath

Хотя мы уже знаем, например, что для обращения к текущему узлу можно присвоить «.» атрибуту select, «.» не является допустимым образцом выбора: это сокращение XPath для self::node(). Применение образцов выбора ограничено только двумя осями: ребенка и атрибута, но в XPath тринадцать осей, включая self. В этой главе мы рассмотрим все эти оси и приемы работы с ними.

Говоря формально, XPath позволяет вам ссылаться на определенные разделы XML-документов — это язык, предназначенный для адресации различных частей таких документов. При помощи XPath вы указываете, с какой частью документа вы хотите работать. Вот что W3C говорит о XPath:

«Основная задача XPath — адресовать части документа XML. Для реализации этой первоочередной цели он также предоставляет основные средства оперирования строками, числами и логическими значениями. XPath обладает компактным, отличным от XML синтаксисом для облегчения его применения в идентификаторах URI и значениях атрибутов XML. XPath работает с абстрактной, логической структурой XML-документа, а не с его внешним синтаксисом. XPath получил свое имя благодаря тому, что для навигации по иерархической структуре XML-документа в нем используется нотация пути (path), как в идентификаторах URI».

Эта цитата взята из спецификации XPath 1.0. Заметьте, что, хотя основной целью XPath является адресация частей XML-документов, он также поддерживает синтаксис для работы со строками, числами и логическими (true/false) значениями, который, как мы увидим, сам по себе очень полезен.

В данный момент стандартом является XPath версии 1.0, но были выпущены требования для XPath 2.0. Пока еще нет рабочих проектов XPath 2.0, есть только список того, что W3C планирует туда включить. В конце главы будет приведен обзор этого списка. Первичные источники XPath вы найдете в двух местах:

- спецификация XPath 1.0. XPath применяется для поиска и указания определенных разделов и элементов в документах XML для дальнейшей работы с ними. www.w3.org/TR/xpath;
- требования XPath 2.0. XPath обновляется и в него включаются дополнительные средства поддержки XSLT 2.0 — в первую очередь поддержка схем XML. www.w3.org/TR/xpath20req.

Вас могут заинтересовать следующие руководства по XPath:

- http://www.zvon.org/xxl/XPathTutorial/General/examples.html;
- http://www.pro-solutions.com/tutorials/xpath/.

Рассмотренные нами до сих пор образцы выбора возвращали наборы узлов (node set), в которых можно было осуществить выбор или обработать набор в цикле, но XPath предоставляет более общие средства. Помимо наборов узлов, выражения XPath могут также возвращать числа, логические (true/false) значения и строки. Чтобы разобраться с XPath, необходимо разобраться с выражениями XPath. Только один вид выражения XPath (хотя и очень важный) возвращает наборы узлов, определяющие разделы документа. Как мы увидим, другие выражения XPath возвращают данные других видов.

Полный синтаксис выражений XPath описан в спецификации XPath, и я приведу его здесь для справки. Как и в случае образцов выбора, для формального определения выражений XPath W3C использует нотацию расширенных форм Бэкуса-Наура (РБНФ). (Описание этой грамматики вы можете найти по адресу www.w3.org/TR/REC-xml, раздел 6.) В следующем списке приведена нужная нам нотация РБНФ:

; i' >л t

- :> означает «определяется как»;
- + означает «один или больше»;

- * означает «ноль или больше»;
- j означает «или»;
- - означает «не»;
- ? означает «не обязательно»;

Также примите во внимание, что когда элемент заключен в одиночные кавычки, как 'ancestor' или '::', это значит, что элемент должен появиться в выражении буквально (как "ancestor::PLANET"), — такие элементы называются литералами, literals. Ниже приведено полное формальное определение выражения XPath (здесь оно называется Expr):

```
Expr      ::=    OrExpr
OrExpr            AndExpr | OrExpr "or* AndExpr
AndExpr   :=»    EqualityExpr | AndExpr 'and' EqualityExpr
EqualityExpr   :RelationalExpr | EqualityExpr '=' RelationalExpr
      | EqualityExpr '!-' Relational Expr
RelationalExpr          AdditiveExpr | RelationalExpr '<' AdditiveExpr
    | RelationalExpr '>' AdditiveExpr | RelationalExpr '<-' AdditiveExpr
    j RelationalExpr ">-' AdditiveExpr
AdditiveExpr    ::=    MultiplicativeExpr
    | AdditiveExpr '+' MultiplicativeExpr
    j AdditiveExpr     MultiplicativeExpr
MultiplicativeExpr          UnaryExpr
    ¦ MultiplicativeExpr MultiplyOperator UnaryExpr
    j MultiplicativeExpr 'div' UnaryExpr | MultiplicativeExpr 'nod' UnaryExpr
UnaryExpr          UnionExpr | '-' UnaryExpr
MultiplyOperator    ::-    '*'
UnionExpr          PathExpr | UnionExpr '|' PathExpr
PathExpr    ::-    LocationPath | FilterExpr
    | FilterExpr '/' RelativeLocationPath
    I FilterExpr '//' RelativeLocationPath
LocationPath          RelativeLocationPath | AbsoluteLocationPath
AbsoluteLocationPath    :;-    '/'RelativeLocationPath? |
Abbrevi atedAbsoluteLocati onPath
RelativeLocationPath    ::-    Step | RelativeLocationPath */' Step
      | Abbrevi atedRelativeLocationPath
AbbreviatedAbsol uteLocationPath    ::-    '//' RelativeLocationPath
AbbreviatedRelativeLocationPath    ::-    RelativeLocationPath '(t' Step
Step    ::-    AxisSpecifier NodeTest Predicate*    | AbbreviatedStep
AxisSpecifier    ::-    AxisName '::'    | AbbreviatedAxisSpecifier
AxisName    :-«    'ancestor' | 'ancestor-or-self | 'attribute' I 'child' |
'descendant'
    | 'descendant-or-self' | 'following' | 'following-sibling' | 'namespace' [ "parent"
    j 'preceding' | 'preceding-sibling' | 'self'
AbbreviatedAxisSpecifier    :=    '@'?
NodeTest    :NameTest | NodeType '(' ')'
    | "processing-instruction' '(' Literal ')'
Nam6Test          '*' | NCName ':' '*' | QName
NodeType    ::-    'comment' J 'text' | 'processing-instruction'[4] | 'node'
Predicate          'C' PredicateExpr ']*
PredicateExpr          Expr
FilterExpr    ::=    PrimaryExpr | FilterExpr Predicate
PrimaryExpr          VariableReference | '(' Expr ')'
    | Literal | Number | FunctionCall
VariableReference    ::-    't' QName                                    •;
```

```
Number       ::-    Digits ('.' Digits?)? |      Digits
Digits             [0-9]+
FunctionCall             FunctionName '(' ( Argument (      Argument )* )? ')'
FunctionName     ::-    QName - NodeType
Argument         :=E x p r
AbbreviatedStep      ::«     '.'   |   '..'
```

Как видите, спецификация весьма объемна, она включает и обращения к функциям XPath (с которыми мы познакомимся в следующей главе). Лучший способ понять» как работают выражения XPath, — рассмотреть их по возвращаемым типам данных.

Типы данных XPath

В XPath существует четыре типа данных, а не только тип набора узлов, который должны возвращать образцы выбора:

- наборы узлов;
- логические значения;
- числа;
- строки.

ФРАГМЕНТЫ РЕЗУЛЬТИРУЮЩЕГО ДЕРЕВА

В XSLT 1.0 к типам данных XPath добавляются фрагменты результирующего дерева. Как говорилось в главе 4, фрагменты результирующего дерева представляют собой просто фрагменты дерева, которые можно присваивать переменным X5LT. Их поддержка была удалена из рабочего проекта XSLT 1.1, поэтому, скорее всего, они не будут включены в XSLT 2.0. Фрагменты результирующего дерева можно рассматривать как типы данных при помощи <xsl: variable», что мы и увидим в главе 9.

В следующих разделах мы по очереди рассмотрим эти типы данных.

Наборы узлов XPath

Как следует из имени, набор узлов (node set) является просто совокупностью узлов. Набор узлов может включать несколько узлов, единственный узел или быть пустым. Поскольку главная задача XPath — определять место разделов документов, постольку возвращающие наборы узлов выражения XPath являются наиболее популярными типами выражений. Например, выражение XPath child::PLANET возвращает набор узлов из всех элементов <PLANET>, дочерних для контекстного узла. Выражение chUd::PLANET/child:: NAME возвращает набор узлов из всех элементов <NAME>, дочерних для элементов <PLANET> контекстного узла. Выражения XPath такого рода называются путями расположения, location path (W3C называет их «самой важной конструкцией» в XPath), и существенная часть этой главы будет посвящена разъяснению путей расположения.

Чтобы выбрать узел или узлы из набора узлов или обработать их, вы можете воспользоваться следующими функциями XPath для работы с наборами узлов,

которые впервые встретились нам в главе 4 (и которые мы более подробно рассмотрим в следующей главе):

- count(node-set). Эта функция возвращает количество узлов в наборе. Если опустить параметр «набор узлов», функция будет применена к контекстному узлу;
- id(string ID). Эта функция возвращает набор узлов из элемента, чей ID удовлетворяет строке, переданной функции в качестве параметра, или пустой набор узлов, если ни у какого элемента нет указанного ID. Можно указать несколько идентификаторов, разделенных символами-разделителями, — в таком случае функция вернет набор узлов из элементов с этими идентификаторами;
- lastQ. Возвращает номер последнего узла в наборе;
- local-name(node-set). Возвращает локальное имя первого узла в наборе. Если опустить параметр «набор узлов», функция будет применена к контекстному узлу;
- name(node-set). Возвращает полностью определенное имя первого узла в наборе. Если опустить параметр «набор узлов», функция будет применена к контекстному узлу;
- namespace-uri(node-set). Возвращает URI пространства имен первого узла в наборе. Если опустить параметр «набор узлов», функция будет применена к контекстному узлу;
- position(). Возвращает позицию контекстного узла в контекстном наборе узлов (начиная с 1).

В следующем примере (из главы 6) для подсчета количества узлов в наборе применяется функция count. В этом случае набор узлов состоит из всех элементов <PLANET> в planets.xml, и я получил его при помощи пути расположения «\\PLANET» (который как путь расположения также является выражением XPath):

```
<xsl:stylesheet
 xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"
 version-"1.0">
<xs1:output method-"xmr indent-"yes"/>                    1
<xsl template match-"*"»
    <xsl:Copy>
        <xsl:apply-templates/>
    </xsl:copy>
</xsl:template»

<xsl: tempi ate match-"PLANET*>
    <xsl:copy use-attr1bute-sets-"numbering"»
        <xsl:apply*templates/>
    </xsl:copy>
</xsl:template»

<xsl attribute-set name-"numbering">
    <xsl attribute name-"number"><xsl :number/x/xsl :attribute>
    <xsl -.attribute name-"total"»<xsl:value-of select-"count(//PLANET)"/>
    </xsl -.attribute»
</xsl:attribute-set»
</xsl:stylesheet»
```

Ниже показан результат; заметьте, что у каждого элемента <PLANET> есть оба атрибута, number и total, и атрибут total содержит общее число элементов <PLANET> **В** документе:

```
<?xml version-"l O" encoding-"UTF-8"?>
<PLANETS>

    <PLANET number-"Г total-43">
        <NAHE>Mearry</NAME>  . j
        <MASS>.Q553</MASS>
        <0AY>58.65</DAY>
        <RADIUS>1516</RADIUS>
        <DENSITY> 983</DENSITY>
        <DISTANCE>434</DISTANCE>
    </PLANET>

    <PLANET number-"2" total-"3">
        <NAIIE>Venus</NAME>
        <MASS>815</MASS>
        <DAY>U6.75</0AY>
        <RADIUS>3716</RADIUS>
        <DENSITY> 943</DENSITY>
        <DISTANCE>668</DISTANCE>
    </PLANET> .

    «PLANET number-"3" total-"3">
        <NAME>Eatn</NAME>
        <MASS>1</MASS>
        <DAY>1</0AY>
        <RADIUS>2107</RADIUS>
        <DENSITY>1</DENSITY>
        <DISTANCE>128 4</DISTANCE>
    </PLANET>

</PLANETS>
```

Среди функций для работы с наборами узлов в особенности обратите внимание на функции name и local-name. С их помощью можно определить имя текущего элемента: например, если текущим элементом является <DAY>, local-name вернет DAY. Следующая таблица стилей демонстрирует, для чеш это может понадобиться; в ней я использую такие теги, как <PLANETS>, <PLANET> и <DATA>, в качестве элементов буквального результата:

```
<?xml version="1.0"?>
<xsl:stylesheet  version-"]..0"
xmlns xsVhttp //www.w3.org/1999/XSL/Transform">
    <xsl:output method»"xml"/>

    <xsl:tempi ate match-"PLANETS">
        <PLANETS>
            <xs1:for-each select="PLANET">
                <PLANET>                                .«."
                    <xsl:for-each select-"*">
                    •/A    <UATA>          • —          f '          at'KSi
                            <xs1:value-of select-"."/»              •hvu
                        </DATA>
                    </xsl: for - each>                                  . J :      .
```

```
                </PLANET>
            </xsl:for-each>
        </PLANETS>
    </xsl:template»

</xsl:stylesheet»
```

Однако в таком случае разметка трактуется как простой текст. Вместо этого можно создать новые элементы при помощи <xsl:element>, определяя имена контекстных узлов через local-name:

```
<?xml version-"]. 0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method-"xml"/>

    <xslrtemplate match-'PLANETS'>
        <xsl:element name-"{local-name(.)}">
            <xsl:for-each select-"PLANET"»
                <xsl:element name-"{local-name(.)}">
                    <xsl:for-each select-"*»
                        <xsl:element name-"DATA">
                            <xsl:value-of select-"."/>
                        </xsl:element»
                    </xsl:for-each»
                </xsl:element»
            </xsl:for-each»
        </xsl:element»
    </xslrtemplate»

</xsl:stylesheet»
```

Ряд пишущих об XSLT авторов рассматривает выражения XSLT только как выражения, возвращающие наборы узлов. Но выражения XPath возвращают также логические значения, числа и строки, которые используются в элементах <xsl:param>, <xsl:with-param>, <xsl:number>, <xsl:value-of>, <xsl:sort>, шаблонах значений атрибутов и предикатах путей расположения. В предыдущем примере для вставки в документ числа я присвоил атрибуту select элемента <xsl:value-of> выражение XPath count(//PLANET), которое возвращает не набор узлов, а число. Сейчас мы как раз перейдем к обработке чисел при помощи выражений XPath.

Числа XPath

В XPath числа хранятся в формате чисел с плавающей точкой двойной точности. В соответствии с формальным определением, числа XPath должны храниться в формате 64-разрядных чисел с плавающей точкой двойной точности IEEE 754, и все числа хранятся как числа с плавающей точкой двойной точности.

В XPath можно выполнять следующие операции над числами, как мы уже видели в главе 4 при обсуждении предикатов XPath:

- + сложение;
- - вычитание;
- * умножение;

- div деление (символ /, соответствующий делению в других языках, в XML и XPath уже занят);
- mod возвращает значение деления по модулю двух чисел (остаток после деления первого числа на второе).

Например, элемент <xsl:value-of select-"15 + 75"/> вставит в выходной документ строку «90». В следующем примере выбираются все планеты, чей день (измеренный в днях Земли), умноженный на расстояние планеты от Солнца (измеренное в миллионах миль), больше, чем 60 ООО:

```
<xsltemplate match-"PLANETS">
    <HIML>
        <BODY>
            <xsl:apply-tempiates select*"PLANET[DAY * MASS > 60000]"/>
        </BODY>
    <HIML>
</xsl:tempiate>
```

XPath также поддерживает следующие функции работы с числами:

- ceiling(). Возвращает наименьшее целое, большее, чем переданное функции число;
- floor(). Возвращает наибольшее целое, меньшее, чем переданное функции число;
- round(). Округляет переданное число до ближайшего целого;
- sum(). Возвращает сумму переданных функции чисел.

Например, среднее расстояние от Солнца (в миллионах миль) планет в planets, xml можно найти таким способом:

```
<?xml version-"],.0"?>
<xsl:stylesheet vers1on="l,0"
xmlns.xsl-"http.//www.w3 org/1999/XSL/Transform">
<xsl:output method-"xml "/>

<xsl:tempiate match-"PLANETS">
    <HIML>
        <BODY>
            The average planetary distance from the Sun Is:
            <xsl:value-of select-"sum(child::PLANET/child:DISTANCE) div
            count(child::PLANET)"/>
        </BODY>
    <HIML>
</xsl:tempiate>

ji </xsl:stylesheets
```

Строки XPath

В XPath строки по умолчанию состоят из символов Unicode. Как мы уже видели в главе 4 при обсуждении выражений XPath в предикатах выбора, существует

ряд функций, специально предназначенных для работы со строками (более подробно они будут изучаться в следующей главе):

- concat(string stringl, string string2,...). Возвращает конкатенацию (объединение) всех строк;
- contains(string stringl, string string2). Возвращает true (истину), если первая строка содержит (contains) вторую строку;
- format-number(number numberl, string string2, string string3). Возвращает строку, содержащую число numberl в виде форматированной строки, используя string2 в качестве форматирующей строки (форматирующие строки создаются так же, как для метода Java java.text.DecimalFormat) и string3 как возможную строку локализации;
- normalize-space(string stringl). Возвращает строку stringl после отбрасывания лидирующих и завершающих символов-разделителей и замены нескольких последовательных разделителей на один пробел;
- starts-with(string stringl, string string2). Возвращает истину, если первая строка начинается (starts with) со второй строки;
- string-length(string stringl). Возвращает количество символов в строке stringl;
- substring(string stringl, number offset number length). Возвращает length символов из строки, начиная со смещения offset;
- substring-after(string stringl, string string2). Возвращает часть строки stringl после первого вхождения string2;
- substring-before(string stringl, string string2). Возвращает часть строки stringl до первого вхождения строки string2;
- translate(string stringl, string string2, string string3). Возвращает строку stringl, в которой все вхождения символов в строке string2 заменены на соответствующие символы в строке string3.

В листинге 7.1 я ищу слово «miles» во всех атрибутах, и если оно встречается, добавляю в результирующий документ текст «You should switch to kilometers.» (Нужно перевести в километры.).

Листинг *7.1.* Поиск текста в атрибутах

```
<?xml version»"1.0"?>
<xsl:stylesheet version-"1.{T
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

    <xsl template match»"/PLANETS">
        <HTML>
            <HEAD>
                <ITITLE>
                    The Planets Table
                </ITITLE>
            </HEAD>
            <BODY>
                <H1>                                    .!•      1L|
                    The Planets Table
                </Hl>
                <TABLE BORDER«'2'>
```

```
                    <TR>
                        <TD>Name</TQ>
                        <TD>Mass</TD>
                        <TD>Radius</TD>
                        <TD>Day</TO>
                        <TD>D\stance</TD>
                    </TR>
                    <xsl:apply-templates/>
                </TABLE>
            <BODY>
        <HTML>
    </xsl:tempiate>

    <xsl:template match="PLANET">
        <TR>
            <TO><xsl: val ue-of sel ect-" NAME "/x/TD>
            <TD><xsl:apply-templates; select-"MASS"/></TD>  . i
            <TD><xsl apply-templates selectr"RAOiUS"/x/TD>      4·      Itl)
            <TD><xsl apply-templates select-"bAY'7x/TD>
            <TD><xsl.apply-templates select-DfSTANCE"/></TD>
        </TR>
    </xsl:tempiate>

    <xsl-.template match-"MASS">
        <xsl:value-of  select-"."/>
        <xsl:text>  </xsl:text>
        <xsl: val ue-of select-"@UNITS"/>
    </xsl:template»

    <xslrtemplate match-"RAOIUS">
        <xsl:value-of  select-"."/>
        <xsl:text>  </xsl:text>
        <xsl  value-of  select-"@UNITS"/>
    </xsl:tempiate>

    <xslrtemplate match-"DAY">
        <xslrvalue-of  select-"."/>
        <xsl:text>  </xsl:text>
        <xsl:value-of  select-"@UNITS"/>
    «/xsl:template»

    <xslrtemplate match-"DISTANCE">
        <xsl:value-of  select-"."/>
        <xsl:text>  </xsl:text>
        <xsl rvalue-of  select-"(aUNITS"/>
    </xslrtempiate>

    <xsltemplate match-"//*[conta1ns(@UNITS.  'miles')]">
        <xsl:value-of  select-"."/>
        <xsl:text>  </xsl:text>
        <xsl:text>You  should  switch  to  k1lometers.</xsl:text>
    </xsl:template»

</xsl:stylesheet*
```

Вот результирующий документ:

```
<HIML>
    <HEAD>                                                    jjOAi.
```

```
        <TITLE>
            The Planets Table
        </TITLE>
    </HEAD>

    <BODY>
        <H1>
            The Planets Table
        </H1>

        <TABLE BORDER="2">
            <TR>
                <TD>Name</TD>
                <TD>Mass</TD>
                <TD>Radius</TD>
                <TD>Day</TD>
                <TD>Distance</TD>
            </TR>

            <TR>
                <TD>Mercury</TD>
                <TD>0.553 (Earth - 1)</TD>
                <TD>1516 You should switch to kilometers.</TD>
                <TD>58.65 days</TD>
                <TD>43.4 You should switch to kilometers.</TD>
            </TR>

            <TR>
                <TD>Venus</TD>
                <TD>.815 (Earth - 1)</TD>
                <TD>3716 You should switch to kilometers.</TD>
                <TD>116.75 days</TD>
                <TD>66.8 You should switch to kilometers.</TD>
            </TR>

            <TR>
                <TD>Earth</TD>
                <TD>1 (Earth - 1)</TD>
                <TD>2107 You should switch to kilometers.</TD>
                <TD>1 days</TD>
                <TD>128.4 You should switch to kilometers.</TD>
            </TR>

        </TABLE>
    </BODY>
</HTML>
```

Помимо работы с наборами узлов, числами и строками, можно работать и с логическими значениями (true/false).

Логические значения XPath

Логические (Boolean) выражения XPath вычисляются либо в истину (true), либо в ложь (false), и обычно они используются только в предикатах. Для чисел

ноль прннмается за ложь, другие значения — за истину. Пустая строка, "", также считается ложью, все остальные строки — истиной.

Для генерации логических результатов true/false в XPath можно применять ряд логических операций, как мы видели в обзоре в главе 4:

- != означает «не равно»;
- < означает «меньше, чем» (в документах XML используйте <);
- <= означает «меньше или равно» (в документах XML используйте <=);
- - означает «равно» (программисты на C, C++, Java и JavaScript, обратите внимание: эта операция пишется как один знак =, а не два);
- > означает «больше, чем»;
- >» означает «больше или равно».

Для связи логических выражений логическими операциями And и Or используются ключевые слова and и or, слово not инвертирует логический смысл выражения, как в следующем примере, где я выбираю все элементы <PLANET>, кроме первого и последнего:

```
<xsl:tempiate match-"PLAMET[not(position() - 1) and not(position() - last())]">
    <xsl value-of select-"."/>
</xsl:tempiate>
```

Следующий пример уже встречался нам в главе 5, он использует логическую операцию not и операции « и !=:

```
<xsl:template match-"PLANET">
    <xsl:1f test»"NAME Qnot(text())]">
        <xsl:message terminate="yes">
            Each planet must have a name!
        </xsl:message>
    </xsl:1f>
    <xsl:value-of select»"NAME"/>
    <xsl:choose>
        <xsl:when test-"pos1tion()!-last()">. </xsl.when>
        <xsl:when test-"position()=last()-l">and </xsl:when>
        <xsl:otherwise>.</xsl:otherwise>
    </xsl:choose>
</xsl:template»
```

Кроме того, имеется функция true, всегда возвращающая истину, и функция false, всегда возвращающая ложь. Есть также функция lang, при помощи которой вы можете проверить язык, установленный в атрибуте документа xmlrlang: эта функция возвращает истину, если язык, который вы передали в эту функцию, такой же, как и установленный в документе язык.

Как вы видели, существуют все виды выражений XPath, в том числе возвращающие узлы, числа, строки и логические значения. Наиболее важным типом выражений XPath является путь расположения; для создания путей расположена XPatJt* первоначально и. за [умывался, поэтому оставшаяся часть главы будет посвящена работе с ними. .,

Создание путей расположения XPath

Вы уже знакомы со взглядом на документы с позиции XPath; например, в XPath определено семь типов узлов документа:

- Корневой узел. Самое начало документа. Этот узел представляет в XPath весь документ;
- Узлы элементов. Представляют элементы в деревьях XPath, обозначенные открывающим и соответственным закрывающим тегами или единственным пустым тегом элемента;
- Узлы атрибутов. Значение атрибута после раскрытия ссылок на Сущность и отбрасывания окружающих символов-разделителей;
- Узлы комментариев. Текст комментариев, не включая <!-- и -->;
- Узлы пространств имен. Объявление пространства имен. В XPath узел пространства имен добавляется во все элементы для каждого активного пространства имен, включая пространства имен по умолчанию;
- Узлы инструкций обработки. Содержат текст инструкции обработки, не включая <? и ?>;
- Текстовые узлы. Текст PCDATA. Текстовые узлы по умолчанию в XPath нормализованы, поэтому расположенные рядом текстовые узлы немедленно объединяются.

Для задания узла или набора узлов в XPath служит *путь расположения* (location path). Путь расположения, в свою очередь, состоит из одного или более шагов расположения (location step), также называемых просто шагами, разделенных / или //. Если путь расположения начинается с /, он называется *абсолютным путем расположения,* поскольку путь задается от корневого узла; иначе путь называется относительным, начинаясь от контекстного узла.

Шаги расположения, строительные блоки путей расположения, во многом похожи на образцы шага (step patterns), формирующие образцы выбора, которые мы видели в главе 4. В частности, шаг расположения образован из оси (axis), условия узла (node test) и предикатов (которых может и не быть) по такому образцу: axis::nodetest[predicate]. Например, в выражении ancestor::NAME[position() > 100], ancestor — это имя оси, NAME — условие узла и [position() >100] — предикат. (Предикат сам содержит законченное выражение XPath, обычно возвращающее логическое значение.) Пути расположения создаются при помощи одного или более шагов расположения, таких как /descendant::PLANET/child::NAME, который выбирает все элементы <NAME> с родителем <PLANET>.

Шаги XPath похожи на образцы шага из главы 4, так как общий их вид одинаков — axis::nodetest[predicate] — но в данном случае нужно рассмотреть значительно Сюрьще \^а^ериала Например, осей теперь тринадцать, а не две.

Шаги расположения XPath, часть 1: оси

В пути расположения ancestor::NAME, адресующем элемент <NAME>, который является предком контекстного узла, ancestor выступает осью. XPath поддерживает много различных осей, ниже приведен полный список:

- ось ancestor содержит предков (ancestor) контекстного узла, то есть родителей контекстного узла, родителей этих родителей и т. д., вплоть до корневого узла (включая его);
- ось ancestor-or-self содержит контекстный узел и его предков;
- ось attribute содержит атрибуты контекстного узла;
- ось child содержит детей контекстного узла;
- ось descendant содержит потомков (descendant) контекстного узла, то есть его детей, детей этих детей и т. д.;
- ось descendant-or-self содержит контекстный узел и его потомков;
- ось following содержит все узлы в том же документе, такие же, как контекстный узел, встретившиеся после контекстного узла;
- ось following-sibling содержит всех последующих братьев контекстного узла «Брат» — узел, расположенный на том же уровне, что и контекстный узел;
- ось namespace содержит узлы пространств имен контекстного узла;
- ось parent содержит родителя контекстного узла;
- ось preceding содержит все узлы до контекстного узла;
- ось preceding-sibling содержит всех предшествующих «братьев» контекстного узла;
- ось self содержит контекстный узел.

В следующем примере шаблона я воспользовался осью descendant для выбора потомков контекстного узла, куда входят дочерние узлы, узлы-внуки, узлы-правнуки и т. д.:

```
<xsl template match-"PLANET">
    <DATA>
        <NAME>
            <xsl:value-of select-"descendant::NAME"/>
        <NAME>
        <MASS>
            <xsl:value-of select-"descendant::MASS"/>
        <MASS>
        <DAY>
            <xsl:value-of select-"descendant::DAY"/>
        </DAY>
    </DATA>
«/xsl;tempiate>
```

В этой главе мы рассмотрим все перечисленные оси. В данном примере осью является descendant, а имена элементов NAME, MASS и DAY — это условия узлов.

Шаги расположения XPath, часть 2: условия узлов

При создании образцов в качестве условий узлов (node test) можно использовать имена узлов или символ Подстановки * для выбора любого узла элемента. Например, выражение chiLd::*/child::NAME выбирает все элементы <NAME>, являющиеся правнуками контекстного узла. В XPath кроме имен и символа подстановки можно также применять, как и в образцах выбора, следующие условия узлов:

- условие узла comment() выбирает узлы комментария;
- условие узла node() выбирает узел любого типа;
- условие узла processing-instruction() выбирает узел инструкции обработки. В скобках можно указать название выбираемой инструкции обработки;
- условие узла text() выбирает текстовый узел

Например, в листинге 7.2 таблица стилей находит в документе все комментарии при помощи условия узла comment() и создает для каждого комментария новый, <!--Warning:comment found!—> (Внимание! Найден комментарий!).

Листинг 7.2. Выбор комментариев

```
<xuul version»"1.0"?>
<x$1:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3 org/1999/XSL/Transform">
<xsl output method"xml"/>

  <xsl:template match»"/">

  <xsl:for-each select-"descendant::comment()">
      <xsl:comment>Warningj comment found!</xsl:comment*
  </xsl:for-each>

  </xsl:template»

</xsl stylesheet>
```

При применении этой таблицы стилей к planets.xml получается следующий документ:

```
<?xml version-"1.0" encoding*»"UTF-8"?>
<!--Warning: comment found!-->
<!--Warn1ng: comment found!-->
<!--Warning: comment found!-->
```

Шаги расположения XPath, часть 3: предикаты

Предикат в шаге расположения XPath сам содержит заключенное в скобки выражение XPath, которое вычисляется в истину или ложь. Когда результатом вычисления выражения является строка, XPath считает ее истиной, если строка не пуста. Когда результат — набор узлов, XPath считает его истиной, если он не

пуст. Когда результат — число, то общий результат будет считаться истиной, если это число совпадает с контекстной позицией — например, PLANET[3] будет истиной тогда и только тогда, когда истиной будет PLANET[position() = 3].

Предикаты содержат выражения XPath наподобие тех, которые встречались нам на протяжении этой главы: редко когда выражение возвращает набор узлов, главным образом возвращаются строки, числа или логические значения. Например, путь расположения preceding-sibling::MASS[position() * 4] выбирает четыре предыдущих элемента-брата <MASS> для контекстного узла.

Применение осей XPath

К этому моменту мы рассмотрели три части шагов расположения — ось, условие узла и предикат. Вы должны быть знакомы с этими элементами по проделанной нами работе с образцами выбора, но обратите внимание на ось в предыдущем примере — preceding-sibling. До сих пор мы видели только оси, выбиравшие образцы XSLT — оси child и attribute; теперь же мы рассмотрим новые оси, возможные в полных выражениях XPath, и начнем с оси ancestor.

Применение оси ancestor

Ось ancestor (предок) содержит всех предков контекстного узла, включая родителей, дедушек, прадедушек и т. д. Эта ось всегда содержит корневой узел — если только контекстным узлом не является сам корневой узел.

Взгляните на листинг 7.3, в котором при помощи оси ancestor осуществляется поиск имен (хранимых в элементе <NAME>) всех предков элементов <MASS>.

Листинг 7.3. Применение оси ancestor

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"!..O"
xmlns:xsl-"http://www w3 org/1999/XSL/Transform">
<xsl:output method""xml"/>

<xsl :template match-"MASS">
    <xsl:for-each select-"ancestor::*">
        <xsl:value-of select-"./NAME"/>
    </xsl:for-each>
</xsl:tempiate>

<xsl:template match-"PLANET">
    <xsl apply-templates select-"MASS"/>
</xsl:template»

</xs1:$ty.lesheet>                                                    *£  j
```

Вот результат применения этой таблицы стилей к planets.xml:

```
<?xml vers1on-"l.O" encod1ng-"u,tf-8"?>
    Mercury
    Venus
m н»  Earth
```

Применение оси ancestor-or-self

Ось ancestor-or-self содержит всех предков контекстного узла, а также сам контекстный узел. Это означает, помимо прочего, что такая ось всегда содержит корневой узел.

В листинге 7.4 добавлены атрибуты AUTHOR со значением «Steve» в весь документ.

Листинг 7.4. planets.xml с атрибутами AUTHOR

```
<?xml version-"1.0"?>
<?xml-stylesheet type="text/xml"  href="planets.xsl"?>
<PLANETS AUTHOR-"Steve" >

    <PLANET AUTHOR-'Steve" >
        <NAME>Mercury«/NAME>
        <MASS AUTHOR-"Steve"  UNITS-"(Earth - 1)">.0553«/MASS>
        <DAY UNITS="days">58.65«/DAY>
        «RADIUS UNITS="miles">1516</RADIUS>
        <DENSITY  UNITS-"(Earth - 1)">.983</DENS1TY>
        <DISTANCE  UNITS-"million mi 1es">43.4</DISTANCE><!--B перигелии-->
    </PLANET>

    <PLANET AUTHOR-"Steve">
        <NAME>Venus</NAME>
        <MASS  UNITS-"(Earth - 1)">.815</MASS>
        <DAY UN I TS«" day S" >116.75</DAY>
        <RADIUS UNITS»"mi 1es">3716</RADIUS>
        «DENSITY  UNITS-"(Earth - 1)">.943</DENSITY>
        «DISTANCE  UNITS-"million miles">66.8«/DISTANCE><«!--B перигелии-^
    </PLANET>

    <PLANET>
        <NAME>Earth</NAME>
        <MASS  UNITS-"(Earth - 1)">1</MASS>
        <DAY UNITS-"days">1 </DAY>
        «RADIUS  UNITS-"miles">2107</RADIUS>
        «DENSITY  UNITS-"(Earth - 1)">1«/DENSITY>
        «DISTANCE  UNITS="million miles">128.4</DISTANCEx!--B перигелии-->
    </PLANET>

</PLANETS>
```

Предположим теперь, что я хочу перечислить По имени всех предков элементов <MASS>, имеющих атрибут AUTHOR, а также текущий элемент <MASS>, если у него есть атрибут AUTHOR. Это можно сделать при помощи оси ancestor-or-self и функции local-name (листинг 7.5).

Листинг 7.5. Применение оси ancestor-or-self

```
<?xml version-"1.0"?>
«xsl:stylesheet version-"1.0"
xmlns xsl-"http://www.w3.org/1999/XSL/Transform">
«xsl:output method="xml"/>

«xsl:template match-"MASS">
```

```
    <xsl:for-each select-"ancestor-or-self::*[@AUTH0R]">
        <xsl:value-of select-"1ocal-name(.)" />
        <xsl:text> </xsl:text>
    </xsl:for-each>
</xsl:tempiate>

<xsl:template match-"PLANET">
    <xs1:apply-templates select="MASS'7>
</xsl:tempiate>

</xsl:stylesheet»
```

Вот результат; показаны выбранные предки всех трех элементов <MASS>, включая сам элемент <MASS>, при условии, что у него имеется атрибут AUTHOR:

```
<?xml version-"1.0" encoding-"UTF-8"?>

    PLANETS PLANET MASS
    PLANETS PLANET
    PLANETS
```

Применение оси descendant

Ось descendant (потомок) содержит всех потомков контекстного узла. Заметьте, что сюда не входят атрибуты или узлы пространств имен, поскольку они не считаются дочерними узлами.

В следующем примере (листинг 7.6) демонстрируется работа с этой осью. На этот раз я хочу добавить примечание к элементу <PLANET> Меркурия: <INF0>Sorry, Mercury has blown up and is no longer available.</INFO> (Извините, но Меркурий взорвался и больше не доступен.). Чтобы найти Меркурий, мне достаточно только проверить, имеет ли какой-либо потомок элемента <PLANET> строковое значение «Mercury», что я сделаю при помощи выражения XPath внутри предиката выбора.

Листинг 7.6. Применение оси descendant

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"l.O"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">
<xsl:output method-"xml"/>

  <xslrtemplate match-"PLANET[descendant::*-'Mercury']">
    <xsl:copy>
      <xsl:apply-tempiates select-"@*|node()"/>
      <INF0»Sorry. Mercury has blown up and is no longer available.</INF0>
    </xsl:copy>
  </xsl:template»

  <xslrtemplate match-"@*|node()">
    <xsl:copy>
      <xsl-apply-templates select-"@*|node()"/>
    </xsl:copy>
  </xsl:template»

</xsl:stylesheet»
```

Вот результирующий документ, дополненный новым элементом <INF0> только для Меркурия:

```
<?xml version-"1.0" encoding-"utf-8"?>
<?xml-stylesheet type-"text/xml" href="planets.xsl"?>
<PLANETS>

    <PLANET>
        <NAME>Mercury«/NAME>
        <MASS UNITS-"(Earth - 1)">.0553</MASS>
        <DAY UNITS-"days">58.65</DAY>
        <RADIUS UNITS-"miles">1516«/RADIUS>
        «DENSITY UNITS-"(Earth - 1)">.983«/DENSITY>
        «DISTANCE UNITS-"million mi 1es">43.4</DISTANCE><!—В перигелии-->
      «INFOSorry, Mercury has blown up and Is no longer available.</INFO>
    </PLANET>

    <PLANET>
        <NAME>Venus</NAME>
        <MASS UNITS-"(Earth - 1)">.815«/MASS>
        <DAY UNITS-"days">116.75</DAY>
        «RADIUS UNITS-"miles">3716«/RADIUS>
        «DENSITY UNITS-"(Earth = 1)">.943«/DENSITY>
        «DISTANCE UNITS-"million miles">66.8</DISTANCEx!--В перигелии-->
    </PLANET>



</PLANETS>
```

Применение оси descendant-or-self

Ось descendant-or-self содержит всех потомков контекстного узла и сам контекстный узел. Заметьте, однако, что она не содержит атрибутов и узлов пространств имен.

В следующем примере (листинг 7.7) демонстрируется работа с осью. В Ьтом случае я создал упрощенную таблицу стилей (подробнее об упрощенных таблицах стилей см. главу 2), которая обрабатывает все элементы с использованием потомков, генерируя уже знакомую нам HTML-таблицу данных о планетах.

Листинг 7.7. Применение оси descendant-or-self

```
«HTML xmlns•xsl-"http://www w3 org/1999/XSL/Transform" xsl version-"1.0*>
    <HEAD>
        <T1TLE>
           The Planets Table
        </TITLE>
    </HEAD>
    <BODY>
        <H1> «•        •                                                    ;
         , The Planets Table              ч                    к
        </H1>
        «TABLE BORDER-"2">
           <TR>
                <TD>Name«/TD>
                <TD>Mass«/TD>
                <TD>Radius«/TD>
```

```
            <TD>Day</TD>
        </TR>
        <xsl:for-each  select»"/descendant-or-self::node()/PLANET">
            <TR>
                <TDxxsl: val ue-of  select-"NAME"/></TD>
                <TD><xsl:value-of  select-"MASS'7></TD>
                <TD><xsl:value-of  select-"RADIUS'7></TD>
                <TD><xsl:value-of  select-"DAY'7></TD>
            </TR>
        </xsl  for-each>
    </TABLE>
</BODY>
</HTML>
```

Вот и все. Я применил здесь упрощенную таблицу стилей, чтобы подчеркнуть, что при помощи таких осей потомков, как descendant или descendant-or-self, вы можете автоматически обрабатывать все выбираемые узлы, во многом аналогично тому, как это делают элементы <xsl:for-each> или <xsl:template>.

Применение оси following

Ось following (следующий) содержит все узлы, расположенные после контекстного узла в соответствии с установленным в документе порядком (другими словами, в порядке, в котором они появляются в документе, начиная с его начала), исключая всех потомков контекстного узла, а также исключая узлы атрибутов и пространств имен.

В этом примере (листинг 7.8) я выбираю каждый элемент <PLANET> и копирую все последующие элементы в результирующий документ.

Листинг 7.8. Применение оси following

```
<?xml version-"1 0"?>
<ksl:stylesheet  version-"l.O"
xmlns:xslhttp://www.w3.org/1999/XSL/Transform">
<xsl:output  method-"xml"/>

<xsl:tempiate  match-"PLANET">
    «xsl:for-each  select-"foilowing::*">
        <xsl:copy-of  select-"."/>
    </xsl:for-each>
</xsl:template»

</xsl:stylesheet»
```

Далее показан результат. Обратите внимание на то, что, когда этот шаблон выбирает элемент <PLANET> Меркурия, он копирует все последующие элементы — то есть Венеру, затем всех потомков Венеры, далее Землю и затем всех потомков Земли. После этого он выбирает элемент <PLANET> Венеры и копирует все следующие элементы, то есть Землю и всех потомков Земли:

```
<?xml vers1on-"l.O"  encoding-"UTF-8"?>

    <PLANET>
        <NAME>Venus</NAME>
```

```
    <MASS UNITS-"(Earth - 1)">.8l5«/MASS>
    <DAY UNITS-"days">116.75</DAY>
    «RADIUS UNITS*"ml1es">3716«/RADIUS>
    «DENSITY UNITS-"(Earth - 1)">.943«/DENSITY>
    «DISTANCE UNITS-"million m1les">66.8</DISTANCE><!—B перигелии-->
</PLANET>
<NAME>Venus«/NAME>
«MASS UNITS-"(Earth - 1)">.815«/MASS>
«DAY UNITS-"days">116.75</DAY>
«RADIUS UNITS-"mi1es">3716«/RADIUS>
«DENSITY UNITS-"(Earth - 1)">.943«/DENSITY>
«DISTANCE UNITS-"million miles">66.8«/DISTANCE>
<PLANET>
    <NAME>Earth«/NAME>
    «MASS UNITS-"(Earth - 1)">1«/MASS>
    «DAY UNITS-"days">l«/DAY>
    «RADIUS UNITS-"miles">2107«/RADIUS>
    «DENSITY UNITS-"(Earth - 1)">1«/DENSITY>
    «DISTANCE UNITS-"million miles">128.4«/DISTANCE><!—B перигелии-->
«/PLANET>
<NAME>Earth«/NAME>
«MASS UNITS-"(Earth - 1)">1«/MASS>
«DAY UNITS-"days">1«/DAY>
«RADIUS UNITS-"miles">2107«/RADIUS>
«DENSITY UNITS-"(Earth - 1)">1«/DENSITY>
«DISTANCE UNITS-"million miles">128.4«/DISTANCE>
<PLANET>
    <NAME>Earth«/NAME>
    «MASS UNITS-"(Earth - 1)">1«/MASS>
    «DAY UNITS»"days">l«/DAY>
    «RADIUS UNITS«"miles">2107«/RADIUS>
    «DENSITY UNITS-"(Earth - 1)">1«/DENSITY>
    «DISTANCE UNITS-"million miles">128.4«/DISTANCE><!—B перигелии->
</PLANET>
«NAME>Earth«/NAME>
«MASS UNITS-"(Earth - 1)">1«/MASS>
«DAY UNITS-"days">l«/DAY>
«RADIUS UNITS-"miles">2107«/RADIUS>
«DENSITY UNITS-"(Earth - 1)">1«/DENSITY>
«DISTANCE UNITS-"million miles">128.4«/DISTANCE>
```

С другой стороны, при использовании оси following-sibling в результирующий документ будут скопированы только следующие братья, то есть только элементы <PLANET>, как мы увидим в следующем разделе.

Применение оси following-sibling

Ось following-sibling содержит всех последующих братьев контекстного узла.

Например, я могу выбрать каждый элемент <PLANET> и скопировать в результирующий документ все узлы в оси following-sibling следующим образом (листинг 7.9).

Листинг 7.9. Применение оси following-sibling

```
<?xml version-"1.0"?>
«xsl:stylesheet version-"!.0"
xmlns: xsl-"http: //Www. w3. O^1999/XSL/Trahsf0rm">
```

```
<xsl -.output method-"xml"/>

<xsl:tempiate match-"PLANET"»
    <xsl:for-each select-"foilowi ng-s1bl1ng::*"»
        <xsl:copy-of select-"."/»
    </xsl:for-each>
</xsl:template»

</xsl:stylesheet»
```

При этом сначала копируются два узла-брата, следующие за Меркурием (Венера и Земля), затем копируется следующий узел-брат Венеры, Земля. У самой Земли нет следующих за ней братьев, поэтому результат выглядит Так:

```
<?xml version-"I 0" encoding="UTF-8"?>

    <PLANET>
        <NAME»Venus</NAME>
        <MASS UNITS-"(Earth - 1)"».815</MASS»
        <DAY UNITS-"days"»116.75</DAY>
        <RADIUS UNITS-"m1les"»3716</RADIUS>
        <DENSITY UNITS-"(Earth - 1)"».943</DENSITY»
        «DISTANCE UNITS-"m1ll1on miles*>66.8</DISTANCEx!--B перигелии--»
    </PLANET»
    <PLANET»
        <NAME»Ea rth</NAME»
        <MASS UNITS-"(Earth - 1)"»1</MASS>
        <DAY UNITS-"days"»1</DAY»
        «RADIUS UNITS-"miles"»2107</RADIUS»
        <DENSITY UNITS-"(Earth - 1)"»1</DENSITY>
        <DISTANCE UNITS-"milHon miles"»128.4</DISTANCE»<!--B перигелии--»
    </PLANET»

    «PLANET»
        <NAME»Ea rth</NAME»
        <MASS UNITS-"(Earth - 1)"»1</MASS>
        <DAY UNITS-"days"»l«/DAY>
        <RADIUS UNITS-"miles"»2107«/RADIUS>
        «DENSITY UNITS-"(Earth - 1)"»1</DENSITY»
        «DISTANCE UNITS-"million miles">128.4</DISTANCE»<!--B перигелии--»
    «/PLANET»
```

Применение оси namespace

Ось namespace содержит узлы пространств имен контекстного узла. Заметьте, что эта ось пуста, если контекстным узлом не является элемент. У эл'емента присутствует узел пространства имен для:

- .каждого атрибута элемента, чье имя начинается с «xmlns>;
- каждого атрибута элемента-предка, чье имя начинается с «xmlns:* (конечно, если сам элемент или ближайший предок не объявит пространство имен заново);
- атрибута xmlns, если элемент или предок имеет атрибут xmlns.

В следующем примере (листинг 7.10) я хочу отобразить пространство имен элемента <PLANET> в результирующем документе, и в исходном документе я присвоил пространству имен значение «http://www.starpowder.com».

Листинг 7.10. planets.xml с объявлением пространства имен

```
<?xml version»"1.0"?>
«?xml-stylesheet type=-"text/xml" href="planets.xs1"?>
«PLANETS xmlns-"http://www.starpowder.com">

    <PLANET>
        <NAME>Mercury</NAME>
        <MASS UNITS»"(Earth - 1)">.0553</MASS>
        <DAY UNITS-"days">58.65</DAY>
        «RADIUS UNITS="rmles">1516«/RADIUS>
        «DENSITY UNITS-"(Earth - 1)">.983«/DENSITY>
        «DISTANCE UNITS-"mill1on miles">43.4«/DISTANCE><!-8 перигелии-->
    </PLANET>
        .
        .
        .
```

Вот таблица стилей (листинг 7.11), в которой я проверяю пространства имен, используемые в элементе <PLANETS>.

Листинг 7.11. Применение оси namespace **в** planets.xml

```
«?xml version-"1.0"?>
«xsl stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">
«xsl:output method-"xml"/>

«xslrtemplate match-"PLANETS">
    «xsl:value-of select="namespace::*"/>
«/xsl:template»

«/xsl stylesheet»
```

А вот результирующий документ (заметьте, что вид документа может меняться в зависимости от процессора XSLT):

```
<?xml version-"l 0" encoding-"UTF-8"?>
http://www.sta rpowder.com
```

Применение оси parent

Ось parent (родитель) содержит родителя (и только родителя) контекстного узла, если таковой имеется.

Предположим, что я хочу изменить содержимое элемента Земли <MASS> на «The mass of Earth is set to 1> (Масса Земли принимается за 1). В следующем шаблоне (листинг 7.12) для этого проверяется, содержит ли родитель <PIANET> элемента <MASS> элемент <NAME> со строковым значением «Earth». ^.3и*'

Листинг 7.12. Применение оси parent

```
«?xml version-"1.0"?>
«xsl:stylesheet version-"!.0"
xmlns:xsl-"http:.//www. w3.org/1999/XSL/Transform">
```

```
<xs 1 ^output method-"xml"/>

  <xsl template match-"@*|node()">
    <xsl:copy>
      <xsl: apply-templ ates selpct->"@*|node(>"/»
    </xsl:copy>
  </xsl:tempi ate»

  <xsl:tempiate match-"MASS[parent::node()/NAME-*Earth']">
    <MASS>The mass of Earth 1s set to 1.</MASS>
  </xsl:template»

</xsl:stylesheet»
```

И вот результат:

```
<?xml version-"!..0" encoding-"utf-8"?>
<?xml-stylesheet type="text/xml" href-"planets.xsl"?>
<PLANETS>

    <PLANET>
        <NAME»Mercury</ NAME»
        <MASS UNITS-"(Earth - 1)">.0553</MASS>
        <DAY UNITS-"days">58 65</DAY>
        <RADIUS UNITS-"miles"»1516</RADIUS»
        <DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
        <DISTANCE UNITS-"million miles"»43.4</DISTANCEx!--B перигелии--»
    </PLANET>

    <PLANET>
        <NAME»Venus</NAME»
        <MASS UNITS-"(Earth - 1)"».815</MASS>
        <DAY UNITS-"days"»116 75</DAY>
        <RADIUS UNITS-"miles"»3716</RAD[US>
        <DENSITY UNITS-"(Earth - 1)"».943</DENSITY>
        <DISTANCE UNITS-"million miles"»66.8</DISTANCE»<!--B перигелии--»
    </PLANET>

    <PLANET>
        <NAME»Earth</NAME>
        <MASS»The mass of Earth is set to 1.</MASS>
        <DAY UNITS-"days"»l</DAY>
        <RADIUS UNITS="miles"»2107</RADIUS»
        «DENSITY UNITS-"(Earth - 1)"»1</DENSITY>
        <D I STANCE UNITS-"milliori miles"»128.4</DISTANCE»<!--B перигелии-»
    </PLANET»

Fir  </PLANETS>
tt
```

^Применение оси preceding

Ось preceding содержит все узлы, расположенные перед контекстным узлом в соответствии с установленным в документе порядком, исключая всех предков контекстного узла, а также исключая узлы атрибутов и узлы пространств имен.

Пусть, например, мне нужно задать для содержимого элемента <DISTANCE> текст «This planet is farther from the Sun than Mercury.» (Эта планета расположена дальше от Солнца, чем Меркурий.), если рассматриваемая планета действительно дальше от Солнца, чем Меркурий. Один из способов сделать это — проверить, расположен ли Меркурий перед рассматриваемой планетой в соответствии с установленным в документе порядком, при помощи оси preceding (листинг 7.13).

Листинг 7.13. Применение оси preceding

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://wv<w.w3.org/1999/XSL/Transform">
<xsl:output method*"xml"/>

  <xsl:tempiate match-"DISTANCEtpreceding::•/NAME*'Mercury']">
    <DISTANCE>This planet is farther from the Sun than Mercury.</DISTANCE>
  </xsl:template*

  <xsl rtemplate match-*"@*|node()">
    <xsl:copy>
      <xsl:apply-templates select-"@*|node()"/>
    </xsl:copy>
  </xsl:tempiate>

</xsl:stylesheet>
```

Если текущая планета расположена после Меркурия, я могу вставить сообщение в ее элемент <DISTANCE>. Результат следующий:

```
<?xml version-"1.0" encoding«"utf-8"?>
<?xml-stylesheet type-"text/xml" href-"planets.xsl"?>
<PLANETS>

    <PLANET>
        <NAME>Mercury</NAME>
        <MASS UNITS-"(Earth - 1)">.0553</MASS>
        <DAY UNITS«"days">58.65</DAY>
        <RADIUS UNITS-"miles">1516</RADIUS>
        <DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
        <01STANCE UNITS-"m1llion miles">43.4</DISTANCE>
        <!--B перигелии-->
    </PLANET>

    <PLANET>
        <NAME>Venus</NAME>
        <MASS UNITS-"(Earth - 1)">.815</MASS>
        <DAY UNITS-Bdays">116.75</DAY>
        <RADIUS UNITS-"miles">3716</RADIUS>
        «DENSITY UNITS-"(Earth - 1)">.943</DENSITY>
        <DISTANCE>Th1s planet 1s farther from the Sun than Mercury.</DISTANCE>
        <!--B перигелии-->
    </PLANET>

    <PLANET>
        <NAME>Earth</NAME>
      - <MASS UNITS-"(Earth - 1)">1</MASS>
        <DAY UNITS-"days°>I</DAY>
```

```
        <RADIUS UNITS-"m1les">2107</RADIUS>
        <DENSITY UNITS-"(Earth - 1)">1</DENSITY>
        <DISTANCE>Th1s planet 1s farther from the Sun than Mercury.</DISTANCE»
        <!--В пери гелии--<@062>
      </PLANET»

  </PLANETS»
```

Применение оси preceding-sibling

Ось preceding-sibling содержит всех предшествующих братьев контекстного узла. Заметьте, что если контекстным узлом является узел атрибута или узел пространства имен, ось preceding-sibling будет пуста.

Что, если, например, вам нужно создать шаблон, который будет выбирать только элементы <DISTAWCE> в элементе <PLANET> Меркурия? Для этого можно проверить, существуют ли братья, предшествующие элементу <DISTANCE>, которые являются элементами <NAME> со строковым значением «Mercury». Если применить ось preceding-sibling (листинг 7.14), поиск будет ограничен текущим элементом <PLANET>, что означает, что Меркурий не будет выбран, если вы только не находитесь в нужном элементе <PLANET>.

Листинг 7.14. Применение оси preceding-sibling

```
<?xml version-"1.0"?»
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http //www.w3 org/1999/XSL/Transform">
<xsl:output method="xml"/>

  <xsl-.template match-"DISTANCE[preced1ng-sibling: :*-'Mercury']"»
    <DISTANCE»Th1s 1s the planet Mercury, closest to the Sun.«/DISTANCE»
  </xsl:tempiate»

  <xsl: tempi ate match»"@* | node()"»
    <xsl:copy>
      <xsl apply-templates se1ect="@*|node()"/»
    </xsl:copy»
  </xsltemplate»

</xsl:stylesheet»
```

А вот результат:

```
<?xml version-"1.0" encoding="utf-8"?>
<?xml-stylesheet type-"text/xml" href="planets.xsV?>
<PLANETS>

    <PLANET>
        <NAME»Mercury</NAME»
        <MASS UNITS-"(Earth = 1)"».0553</MASS>
        <DAY UNITS-"days"»58.65</DAY>
        <RADIUS UNITS"miles">1516</RADIUS>
        «DENSITY UNITS-"(Earth - 1)"».983</DENSITY>
        <DISTANCE»This Is the planet Mercury, closest to the Sun.«/DISTANCE»
        <!--В перигелии--»
    «/PLANET»
```

```
<PLANET>
    <NAME>Venus</NAME>
    <MASS UNITS-"(Earth - 1)">.815</MASS>
    <DAY UNITS-"days">116.75</DAY>
    <RADIUS UNITS-"miles">3716</RADIUS>
    <DENSITY UNITS-"(Earth = 1)">.943</DENSITY>
    <D I STANCE UNITS="mmion miles">66.8</DISTANCE><!--В перигелии-->
</PLANET>

<PLANET>
    <NAME>Earth</NAME>
    <MASS UNITS-"(Earth - 1)">1</MASS>
    <DAY UNITS-"days">1</0AY>
    «RADIUS UNITS-"mi 1 es">2107«/RADIU$>                                    / ft C
    «DENSITY UNITS-"(Earth - l)">i</DENSITY>
    «DISTANCE UNITS~"million m1les">l28.4</DISTANCE><!--B перигелии-->
</PLANET>                                    "

</PLANETS>
```

Применение оси self

Ось self содержит только контекстный узел. В соответствии с одним из сокращений XPath, как мы увидим дальше, вместо «self::node()» можно использовать «.».

Эту ось полезно иметь в виду, поскольку, как вы помните из главы 4, если не задать ось, осью по умолчанию будет child::, а в некоторых случаях вам может понадобиться обратиться к действующему узлу. Например, [self::PLANET] примет значение истины только если контекстным узлом будет элемент <PLANET>.

В Следующем примере я объединяю шаблоны для элементов <NAME> и <MASS> в один шаблон. Поскольку у этих элементов разный формат, я должен обращаться с ними по-разному внутри одного и того же шаблона (что можно сделать проверкой значений оси self.'NAME, которая возвращает непустой набор узлов, если контекстным узлом является элемент <NAME>, и self::MASS, возвращающей непустой набор узлов, если контекстным узлом является элемент <MASS>):

```
<xsl .-template match-"PLANET">
   <TR>
      <TD><«xsl: apply-templ ates sel ect-" NAME' /xУТ[>
      <TD><«xsl:apply-tempiates  select-"MASS'7></TD>
      <TD><«xsl:apply-templates  select-"RADIUS'7></TD>
      <TD><«xsl: apply-templ ates select-'DAY"/></T{)>
   </TR>                                    4
</xsl:template»
          >
<xslrtemplate match™"NAME | MASS'>

   <xsl: 1 f test»"self:NAME"> >•          ?,».'  .мu-«»fj  ТЗИА.        j  «
      <xsl:value- of  select-"."/>
   </xsl:ff>

   <xsl:1f  test-"self::MASS">                    )>  гоо  )И
      «xsl:value-of  select-".*7>
      <xsl:text>  </xsl:text>
```

```
        <xsl:value-of select-"@UNITS'7>
    </xsl:1f>

</xsl:tempiate>
```

На этом мы завершаем рассмотрение новых осей XPath. Давайте перейдем к примерам.

Примеры путей расположения

Мы изучили достаточно теории путей расположения. Но, понятно, лучше всего осваивать этот материал на примерах, поэтому я привожу следующий список примеров путей расположения (сокращенные варианты рассматриваются после этого списка):

- child::PLAN ET. Возвращает дочерние элементы <PLAN ET> контекстного узла;
- child::text(). Возвращает все дочерние текстовые узлы контекстного узла;
- child::node(). Возвращает всех детей контекстного узла;
- attribute::UNIT. Возвращает атрибут UNIT контекстного узла;
- descendant::PLANET. Возвращает все элементы-потомки <PLANET> контекстного узла;
- ancestor::PLANET. Возвращает всех предков <PLANET> контекстного узла;
- ancestor-or-self::PLANET. Возвращает предков <PLANET> контекстного узла. Если контекстным узлом тоже является <PLANET>, возвращает также контекстный узел; -i
- descendant-or-self::PLANET. Возвращает элементы-потомки <PLANET> контекстного узла. Если контекстным узлом тоже является <PLANET>, возвращает также контекстный узел;
- self::PLANET. Возвращает контекстный узел, если им является элемент <PLANET>;
- child::PLANET/descendant::NAME. Возвращает элементы-потомки <NAME> дочерних элементов <PLANET> контекстного узла;
- **child::*/child::PLANET.** Возвращает всех внуков **<PLANET>** контекстного узла;
- /. Возвращает корневой узел;
- /descendant::PLANET. Возвращает все элементы <PLANET> в документе;
- /descendant::PLANET/child::NAME. Возвращает все элементы <NAME> с родителем <PLANET> в документе;
- child::PLANET[position() - 3]. Возвращает третьего ребенка <PLANET> контекстного узла;

- child::PLANET[position() - last()]. Возвращает последнего ребенка <PLANET> контекстного узла;
- /descendant::PLANET[position() • 3]. Возвращает третий элемент <PLANET> в документе;
- child::PLANETS/child::PLANET[position() = 4]/child::NAME[position() - 3]. Возвращает третий элемент <NAME> четвертого элемента <PLANET> элемента <PLANETS>;
- child::PLANET[position() > 3]. Возвращает всех детей <PLANET> контекстного узла после первых трех;
- preceding-sibling::NAME[position() = 2]. Возвращает второй предыдущий элемент-брат <NAME> контекстного узла;
- child::*[self::NAME or self::MASSJ. Возвращает детей <NAME> и <MASS> контекстного узла.
- child::*[self::NAME or self::MASS][position() - last()]. Возвращает последнего ребенка <NAME> или <MASS> контекстного узла.

Как видите, синтаксис некоторых выражений достаточно запутан, и набирать их также довольно долго. Но, как и для образцов, существует сокращенная форма синтаксиса XPath.

Сокращенный синтаксис XPath

Сокращения синтаксиса XPath могут быть весьма удобными. Ниже приведены правила:

- self::node() может быть сокращено как . ;
- parent::node() может быть сокращено как ..;
- child::childname может быть сокращено как childname;
- attribute::childnartie может быть сокращено как @childname;
- /descendant-or-self::node()/ может быть сокращено как //.

Например, путь расположения .//PLANET — сокращение для self::node()/ descendant-or-self::node()/child::PLANET. Можно также сократить выражение предиката [position() - 3] как [3], [positionQ - lastQ] как [lastQ] и т. д. Работать с путями расположения XPath при помощи сокращенного синтаксиса значительно проще. В следующем списке перечислен ряд примеров путей расположения с использованием сокращенного синтаксиса:

- PLANET возвращает дочерние элементы <PLANET> контекстного узла;
- * возвращает все дочерние элементы контекстного узла;
- text() возвращает все дочерние текстовые узлы контекстного узла;
- @UNITS возвращает атрибут UNITS контекстного узла;
- @* возвращает все атрибуты контекстного узла;

- PLANET[3] возвращает третьего ребенка <PLANET> контекстного узла;
- PLANET[last()] возвращает последнего ребенка <PIANET> контекстного узла;
- */PLANET возвращает всех внуков <PLANET> контекстного узла;
- /PLANETS/PLANET[3]/NAME[2] возвращает второй элемент <NAME> третьего элемента <PLANET> элемента <PLANETS>;
- //PLANET возвращает всех потомков <PLANET> корня документа;
- PLANETS//PLANET возвращает элементы-потомки <PLANET> дочерних элементов <PLANETS> контекстного узла;
- //PLANET/NAME возвращает все элементы <NAME>, у которых есть родитель <PLANET>;
- . возвращает сам контекстный узел;
- .//PLANET возвращает элементы-потомки <PLANET> контекстного узла;
- .. возвращает родителя контекстного узла;
- ../@UNITS возвращает атрибут UNITS родителя контекстного узла;
- .//.. возвращает всех родителей потомка контекстного узла и родителя контекстного узла;
- PLANET[NAME] возвращает детей <PLANET> контекстного узла, у которых есть дети <NAME>;
- PLANET[NAME="Venus"] возвращает детей <PLANET> контекстного узла, у которых есть дети <NAME> с текстом, равным «Venus»;
- PLANET[@UNITS «= "days"] возвращает всех детей <PLANET> контекстного узла, у которых есть атрибут UNITS со значением «days»;
- PLANET[6][@ UN ITS » "days"] возвращает шестого ребенка <PLANET> контекстного узла, только если у этого ребенка есть атрибут UNITS со значением «days». Можно также написать PLANET[@UNITS - "days"][б];
- PLANET[@C0L0R and @UNITS] возвращает всех детей <PLANET> контекстного узла, у которых есть атрибут COLOR и атрибут UNITS;
- "//PLANET[not(.»preceding::PLANET)]" выбирает все элементы <PLANET>, значение которых отлично от значения любого предшествующего элемента <PLANET>;
- *[l][self::NAME] выбирает любой элемент <NAME>, который является первым ребенком своего родителя;
- *[position() It; 5][@UNITS] выбирает первых пятерых детей контекстного узла, у которых есть атрибут UNITS.

Проверка выражений XPath

В пакет Xalan входит удобная программа-пример, ApplyXPath.java, позволяющая применить выражение XPath к документу и посмотреть на результат, что очень

помогает при тестировании. Для запуска этого примера вам нужно будет скомпилировать ApplyXPath.java в ApplyXPath.class при помощи утилиты java.exe, входящей в поставку Java.

В качестве примера я применю выражение XPath «PLANET/NAME» к planets.xml при помощи ApplyXPath.class. Ниже показан результат, отображающий все элементы <NAME>, дочерние по отношению к элементам <PLANET> (теги <output> добавлены программой ApplyXPath):

```
fcjava ApplyXPath planets.xml PLANET/NAME
<output>
<NAME>Mercury</NAME><NAME>Venus</NAME><NAME>Earth</NAME>
</output>
```

XPath 2.0

XPath находится в стадии обновления, и в него включаются средства поддержки XSLT 2.0 (см. www.w3.org/TR/xpath20req). Задачи XPath 2.0 следующие:

- упрощение операций с содержимым типов, поддерживаемых схемой XML;
- упрощение операций со строковым содержимым;
- поддержка соответствующих стандартов XML;
- улучшение удобства использования;
- улучшение функциональной совместимости;
- улучшение поддержки международных языковых средств;
- сохранение обратной совместимости;
- повышенная эффективность процессора.

Следующий список дает обзор требований XPath. Главные пункты — поддержка схемы XML и регулярных выражений, чтО дает средства работы со строками и поиска в строках. (Дополнительную информацию о регулярных выражениях можно почерпнуть по адресу http://www.perldoc.eom/perl5.6/pod/perlre.html.) В соответствии с W3C, XPath 2.0: ,

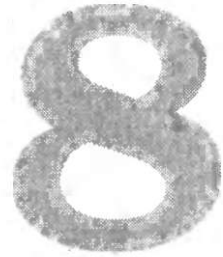
- должен поддерживать архитектуру XML W3C, хорошо взаимодействуя с другими стандартами в семействе XML;
- должен выражать свою модель данных в терминах информационного множества (infoset) XML;
- должен предоставлять общий ключевой синтаксис для XSLT 2.0 и XML Query language 1.0;
- должен поддерживать явное сравнение «for any» или .«for a1b и синтаксис равенства;
- должен расширять множество функций агрегации (например, пользователи XSLT часто требовали добавить функции min() и тах()); »
- должен сохранять обратную совместимость с XPath 1.0;

- должен предоставлять функции пересечения и разности то есть — XPath 1.0 поддерживает объединение двух наборов узлов, и к этому должны быть добавлены функции пересечения и разности;
- должен поддерживать операцию унарного плюса (поскольку в схеме XML у десятичных чисел может присутствовать лидирующий плюс);
- должен улучшать удобство использования;
- должен снизить ограничения на шаги расположения;
- должен реализовывать условную операцию, оперирующую тремя выражениями — выражением 1 (логическая операция), выражением 2 и выражением 3. Если выражение 1 принимает значение «истина», должно вычисляться выражение 2, а если выражение 1 принимает значение «ложь», должно вычисляться выражение 3;
- должен определять последовательный синтаксис для подвыражений, обрабатывающих коллекции элементов;
- должен поддерживать дополнительные строковые функции. Например, W3C рассматривает вопрос добавления средств для замены в строках, заполнения символами и преобразований регистра;
- должен поддерживать функции агрегации при применении к коллекциям. Например, некоторым пользователям XPath 1.0 требовалось применить такую функцию агрегации, как sum, к значениям выражений, примененных к наборам узлов;
- должен поддерживать регулярные выражения для поиска в строках с использованием нотации регулярных выражений, установленной в схеме XML;
- должен поддерживать элементарные типы данных схемы XML. То есть в дополнение к типам, поддерживаемым моделью данных XPath 1.0, — строке, числу, логическому значению и набору узлов — модель данных XPath 2.0 должна поддерживать элементарные типы данных схемы XML;
- должен поддерживать представления чисел с плавающей точкой одинарной и двойной точности, поддерживаемые схемой XML, которая использует научную нотацию;
- должен определять подходящий набор функций для работы пользователя с элементарными типами данных схемы XML;
- должен добавлять в XPath тип данных «список» (поскольку схема XML позволяет определять простые типы, унаследованные от списка);
- должен поддерживать доступ к значениям простых ггаюв элементов **И** атрибутов. Поскольку схемы XML представляют много новых типов, XPath 2.0 должен поддерживать доступ к собственному, простого типа, значению элемента или атрибута;
- должен определять поведение операторов для нулевых аргументов^

- должен иметь средства для выбора элементов или атрибутов на основе явного типа схемы XML;
- должен иметь средства для выбора элементов или атрибутов на основе иерархии типов схемы XML;
- должен иметь средства для выбора элементов на основе групп подстановки схемы XML;
- должен поддерживать средства поиска, основанные на уникальных ограничениях и ключах схемы.

Хотя мы подошли к концу главы, о XPath сказано еще не все. Тема будет продолжена в следующей главе, в которой мы более внимательно рассмотрим доступные в XPath функции и функции, уже встроенные в XSLT.

Работа с функциями XSLT и XPath



В этой главе мы изучим доступные вам в XSLT функции — как встроенные в XSLT, так и функции XPath. Вы уже встречались с этими функциями выборочно (например, с id, generate-id, position, count и т. д). В этой главе мы рассмотрим их все.

Некоторые функции входят в XSLT, некоторые пришли из XPath. И XSLT, и XPath обновляются до версии 2.0, и в конце главы я приведу список новых, готовящихся к выпуску функций. (Пока что никаких новых функций версии 2.0 еще не опубликовано.)

В XSLT встроены следующие функции:

- element-availableO;
- function-available();
- current();
- document();
- key();
- format-number();
- generate-id();
- system-property();
- unparsed-entity-uriQ.

Вы также можете пользоваться многими функциями XPath. Я сгруппирую их по типам данных XPath, с которыми они работают. Существует четыре типа данных: наборы узлов, строки, числа и логические значения. С наборами узлов работают следующие функции XPath:

- count();
- **id();**
- last();
- local-name();

- name();
- namespace-uri();
- position ().

Следующие функции XPath работают со строками:

- concat();
- contains();
- normalize-spaceQ;
- starts-with();
- string();
- string-length();
- substring();
- substring-after();
- substring-before();
- translate().

Для работы с числами служат следующие функции XPath:

- ceiling();
- floor();
- number();
- round();
- sum().

И эти функции XPath работают с логическими значениями:

- boolean();
- false();
- lang();
- not();
- true().

В этой главе мы познакомимся с работой всех вышеназванных функций, а начнем с функций, встроенных в XSLT. Не обязательно читать всю главу подряд — скорее ею лучше пользоваться как справочным руководством: здесь перечислены все функции в удобном для просмотра виде.

Функции XSLT

В следующем списке перечислены встроенные в XSLT функции:

- current(). Возвращает текущий (current) узел — но не контекстный узел. Текущий узел — это узел, используемый в данный момент в таких циклах, как <xsl:for-each>. Функцию current нельзя применять в образцах;

- document(). Позволяет считывать несколько документов;
- element-available(). Показывает, доступен ли элемент расширения;
- format-number(). Форматирует числа при выводе;
- Function-available(). Показывает, доступна ли функция расширения;
- generate-id(). Указывает процессору XSLT присвоить идентификатор узлу и возвратить его. При повторном применении generate-id к тому же узлу функция возвращает присвоенный ранее идентификатор;
- key(). Позволяет осуществлять поиск по ключу;
- system-property(). Позволяет проверить три системных свойства: xsl:version (версия XSLT, поддерживаемая процессором XSLT), xshvendor (производитель процессора XSLT) и xsl:vendor-url (URL производителя процессора XSLT);
- unparsed-entity-uri(). Предоставляет доступ к неразобранным сущностям, объявленным в DTD или схеме, через URI.

В следующих разделах все эти функции рассматриваются подробно и с примерами.

current()

Функция current возвращает текущий (current) — не контекстный (context) — узел. Контекстный узел шаблона — это узел в выбранном наборе узлов, к которому применяется шаблон. Текущий узел, с другой стороны, это используемый в данный момент узел в таких циклах, как <xsl:for-each>. Функция возвращает текущий узел как набор узлов с одним узлом:

```
node-set currentO
```

Обратите внимание: current нельзя применять в образцах, поскольку образцы должны быть независимы от пути обработки, а способы реализации таких структур, как циклы, в процессорах XSLT могут различаться.

Следующий пример, в котором я выбираю элементы <PLANET> при помощи шаблона (листинг 8.1), демонстрирует работу функции current Внутрь шаблона я поместил элемент <xsl:for-each>, и шаблон применяется только в том случае, если выбранный шаблоном контекстный узел в то же время является и текущим узлом в очередной итерации в элементе <xsl:for-each>.

Листинг 8.1. Применение функции current

```
<?xml version~"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

  <xsl: tempi ate match=/PLANETS>
    <HIML>

      <TABLE BORDER-*2*
```

продолжение &

Листинг 8.1 (продолжение)

```
                    <TR>
                        <TD>Name</TD>
                        <TD>Mass</TD>
                        <TD>Radius</TD>
                        <TD>Day</TD>
                    </TR>
                    <xsl:apply-templates/»
                </TABLE>
            </BODY>
        </HTML>
    </xsl:tempiate>

    <xsltemplate match-"PLANET"»
        <xsl:for-each select-"/PLANETS/*[.-current()]">
        <TR>
            <TD><xsl :value-of sel ect-" NAME' /x/lD>
            <TD><xsl -.apply-tempi ates select-"HASS"/></TD>
            <TD><xsl: apply-tempi ates select-"RADIUS"/x/TD>
            <TD»<xsl:apply-tempiates select-"DAY"/»</TD»
        </TR>
        </xsl:for-each>
    </xsl:template»

    <xslrtemplate match-"MASS">
        <xslrvalue-of select-"."/»
        <xsl:text> </xsl:text>
        <xsl:value-of select-"@UNITS'7>
    </xslrtemplate»

    <xslrtemplate match="DAY">
        <xsl;value-of select-" "/>
        <xsl:text> </xslrtext»
        <xslrvalue-of select-"@UNITS"/>
    </xslrtemplate»

</xsl:stylesheet»
```

Результат применения этой таблицы стилей будет точно таким же, как если бы элемента <xsl:for-each> вообще не было, потому что тело элемента <xsl:for-each> применяется только тогда, когда контекстный узел совпадает с текущим узлом.

document()

Функция document особенно полезна, потому что с ее помощью можно читать несколько документов и обрабатывать их содержимое. Функция document применяется следующим образом:

```
node-set document(url. base-uri?)
```

Функция принимает параметр uri, который может представлять собой URI читаемого документа или набор узлов, чьи строковые значения задают идентификаторы URI. Второй, необязательный параметр base-uri представляет собой

набор узлов, базовый URI которого используется для вычисления всех относительных URI, могущих находиться в параметре uri.

В следующем примере (листинг 8.2) я обрабатываю в процессоре XSLT один документ, planetsl.xml, и также считываю и обрабатываю второй документ, planets2.xml. Вот planetsl.xml.

Листинг 8.2. planetsl.xml

```
<?xml version-"1.0"?>
<?xml-stylesheet type-"text/xml" href-"planets.xsl"?>
<PLANETS href-"planets2 xml">
</PLANETS>
```

Вот planets2.xml — в элементе <PLANET> содержится одна планета (листинг 8.3).

Листинг 8.3. planets2.xml

```
<?xml version-"1.0"?>
<PLANET>
    <NAME>Mercury</NAME>
    <MASS UNITS-"(Earth - 1)">.0553</MASS>
    <DAY UNITS-"days">58.65</DAY>
    <RADIUS UNITS-"miles">1516</RADIUS>
    <DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
    «DISTANCE UNITS-"million miles">43.4</DISTANCE><!--В перигелии-->
</PLANET>
```

А вот таблица стилей planets.xml, которую я применяю к planetsl.xml. Таблица стилей содержит шаблон, выбирающий элемент <PLANET> в planetsl.xml, и в этом шаблоне я считываю planets2.xml при помощи <xsl:apply-templates> и функции document (листинг 8.4).

Листинг 8.4. Применение функции document

```
<?xml version-"1.0"?>
<xsl:styl esheet version-»"!.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

    <xsl:tempi ate match-'7PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLEJ>
            </HEAD>
            <BODY>
                <H1>
                    The Planets Table
                </HI>
                <TABLE BORDER-"2">
                    <TR>
                        <TD>Name</TD>
                        <TD>Mass</TD>
                        <TD>Radius</TD>
                        <TD>Day</TD>
[•,                     </TR>
                        <xsl:apply-templates select-"document(@href)"/>
            %>  </TABLE>
```

Листинг 8.4 (продолжение)

```
            </BODY>
        </HTML>
    </xsl:tempiate>

    <xsl:tempiate match-"PLANET">
        <TR>
            <TDxxsl: value-of sel ect-" NAME "/></TD>
            <TDxxsl:apply-templates select-"MASS"/></TD>
            <TDxxsl: apply-templ ates select-"RADIUS"/></TD>
            <TDxxsl: apply-templ ates select-"DAY"/></TD>
        </TR>
    </xsl:template>

    <xsltemplate match-"DAY">
        <xsl:value-of select-"."/>
        <xsl:text> </xsl:text>
        <xsl: val ue-of select-"<aUNITS"/>
    </xsl:tempiate>

</xsl:stylesheet>
```

Эта таблица стилей обрабатывает данные в planetsl.xml и также считывает и обрабатывает planets2.xml; вот полный результат — как видите, данные для planets2.xml добавились нужным образом:

```
<HTML>
    <HEAD>
        <TITLE>
            The Planets Table
        </TITLE>
    </HEAD>

    <BODY>
        <H1>
            The Planets Table
        </H1>

        <TABLE BORDER="2">
            <TR>
                <TD>Name</TD>
                <TO>Mass</TO>
                <TD>Radius</TD>
                <TD>Day</TD>
            </TR>
            <TR>
                <TD>Mercury</TD>
                <T0>.0553 (Earth - 1)</TD>
                <TD>1516 m1les</TD>
                <T0>58.65 days</TD>
            </TR>
        </TABLE>
    </BODY>
</HTML>
```

Функция document удобна, поскольку позволяет считывать на этапе выполнения такие дополнительные документы, как справка о правах владения или отказе от них, фирменный бланк и т. п.

element-available()

Функция element-available() используются для определения доступности определенного элемента расширения. Ее применяют следующем образом:

boolean element-avai1able(element-name)

Функция принимает имя искомого элемента и, если элемент доступен, возвращает истину, если недоступен — ложь.

Мы уже встречались с этой функцией в главе 5. В примере element-available той главы я проверял наличие элемента <starpowder:calculate> таким образом!

```
<xsl:choose  xmlns:starpowder-"http://www.starpowder.com">
    <xsl:when  test-"element-avai1able('starpowder:calculate")">
        <starpowder:calculate xsl:extenslon-element-prefixes»' starpowder"/>
    </xsl:when>
    <xsl:otherwise»
        <xsl:text>Sorry.  can't do math today </xsl:text>
    </xsl:otherwise»
</xsl:choose>
```

format-number()

Как можно догадаться по имени, функция format-number() служит для форматирования чисел с преобразованием их в строки. Она применяется так:

string format-number(number, format, name?)

Функция возвращает форматируемое число в виде строки. Функции передаются число number, которое нужно отформатировать, строка форматирования format и необязятельный параметр-строка name. Строка name — это имя QName, задающее формат так, как он создается элементом <xsl:decimal-format> (который будет рассмотрен в конце этой главы).

Форматирующая строка format должна соответствовать соглашениям класса Java DecimalFormat.

КЛАСС JAVA DECIMALFORMAT

На момент написания книги документация для класса Java DecimalFormat находится в Интернете по адресу: http://java.sun.com/products/jdk /1.1/docs/api/java.text.DecimalFormat.html.

Форматирующая строка (format string) состоит из следующих частей:

- format-string:- subpattern (;subpattern)?
- subpattern:-» prefix? integer (.fraction)?suffix?
- prefix:- [#xO..#xFFFD] - specialCharacters
- suffix:- [#xO..#xFFFD] - specialCharacters

- integer:-'#'* 'O'* 'O'
- fraction:-'O'*'Г*

Далее показаны специальные символы (speciaLCharacters), которые можно использовать в подчиненных образцах, subpattern (эти символы можно изменить при помощи элемента <xsl:decimal-format>, который будет рассмотрен в конце этой главы):

- **0 — на этом месте всегда должна стоять цифра;**
- # — цифра, если только это не избыточный лидирующий или завершающий ноль;
- . — разделитель десятичной части;
- , — разделитель групп разрядов;
- ; — разделяет форматы;
- — знак минуса;
- % — умножить на 100 и показать как проценты;
- ? — умножить на 1000 и показать в тысячных частях;
- Е — разделяет мантиссу и экспоненциальную часть;
- п — символ валюты (#xA4);
- ' — заключает в кавычки специальные символы.

Следующий пример (листинг 8.5) демонстрирует работу функции. Я форматирую значения из planets.xml, отображаемые в таблице HTML.

Листинг 8.5. Форматирование чисел

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="/PLANETS">

  </xsl:template>

  <xsl:template match="PLANET">
    <TR>
      <TD><xsl:value-of select="NAME"/></TD>
      <TD><xsl:apply-templates select="MASS" /></TD>
      <TD><xsl:apply-templates select="RADIUS"/></TD>
      <TD><xsl:apply-templates select="DAY"/></TD>
    </TR>
  </xsl:template>

  <xsl:template match="MASS">
    <xsl:value-of select="format-number(., '#.###')"/>
    <xsl:text> </xsl:text>
    <xsl:value-of select="@UNITS"/>
  </xsl:template>
```

```
<xs,l:template match«"RADIUS">
    4xs1:value of select'"format-numbor(.. '#.W)"/>
    «-xsl text> </xsl :text>
    «•xslrvalue-of select-"@UNITS"/>
</xsT;t;emplate>

*xs'l template match-"DAV">
    <xsl :value-ot select "format-ni'mber(., '###.#')"/:*
    <xs1 :te; :t> </xsl :text>
    <xsl r val ue-of be'iect-"@UNITS"/>
</xsl:template»

</xsl:stylesheet»
```

Вот результат, в котором выведены отформатированные числа:

```
<HIML>
    <HEAD>                                                    «Tr
        <TITLE>
            The Formatted Planets Table
        </TITLE>
    </HEAD»

    <BODY>
        <H1>
            The Formatted Planets Table
        </Hl>

        <TABLE BORDER-"2">
            <TR>
                <Tp>Name</TD>
                <TD>Mass</TD>
                <TD»Radius</TD>
                <TD>Day</TD>
            </TR>

            <TR>
                <TD>Mercury</TD>
                <TD>0,055 (Earth - 1)</TD>
                <TD>1.516 miles</TD>
                <TD>58.65 days</TD>
            </TR>

            <TR>
                <TD>Venus</TD>
                <TD>0.815 (Earth - 1)</TD>
                <TD>3.716 miles</TD>
                <TD>116.75 days</TD>
            </TR>

            <TR>
                <TD>Earth</TD>
                <TD>1 (Earth - 1)</TD>
                <TD>2.107 miles</TD>
                <TD>1 days</TD>
            </TR>
        </TABLE>
    </BODY>
</HTHL>
```

Этот результирующий документ показан на рис. 8.1. (MSXML3 и Saxon отбрасывают лидирующие нули, поэтому 0.055 выводится как .055 и т. д.)

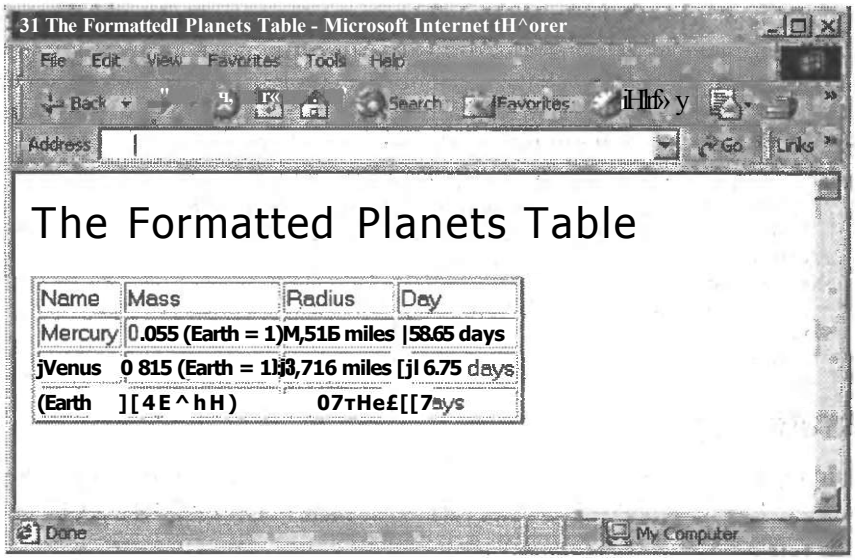


Рис. 8.1. Форматирование чисел при помощи XSLT

Следующие примеры демонстрируют способы применения форматирующих строк. Заметьте, что при помощи точки с запятой (;) можно отделять образцы для положительных и отрицательных чисел.

Число	Форматирующая строка	Результат
4567	#,###	4,567
4567.8	####.##	4567.8
4567.8	#,##0.00	4,567.80
456.789	#,##0.00	456.79
4567890	#,##0.00	4,567,890.00
4567	###0.0###	4567.0
.00045	##0.0###	0.0005
.45	#00%	45%
-4.56	#.00;(#.00)	(4.56)
-45	#,##0.00	-45

function-available()

При помощи функции XSLT 1.0 function-available можно проверить наличие функции расширения:

```
boolean function-available(function-name) .</>•-
```

Функция принимает в качестве параметра имя искомой функции и, если функция доступна, возвращает истину, если нет — ложь.

Следующий пример мы уже рассматривали в главе 5. В этом случае я хочу выполнить ряд математических операций при помощи функции расширения starpowdencalculate, и если она отсутствует, вывести в результирующий документ текст «Sorry, can't do math today.» (Извините, сегодня математические вычисления не работают.). Можно, конечно, также прекратить обработку и вывести сообщение об ошибке при помощи элемента <xsl:message>:

```
<xsl.choose xmlns:starpowder-"http://www starpowder.com">
    <xsl:when test-"funct1on-available('starpowder:calculate')">
        <xsl:value-of select-"starpowder:calculate("2+2')"/>
    </xsl:when>
    <xsl:otherwise*
        <xsl:text>Sorry, can't do math today.</xsl:text>
    </xsl:otherwise»
</xsl:choose>
```

generate-id()

Функция generate-id дает указание процессору XSLT присвоить узлу идентификатор (который возвращается как строка). Функция применяется следующим образом:

```
string generate-id(node)
```

Функции следует передать набор узлов, содержащий только узел, которому нужен идентификатор (все узлы после первого игнорируются); функция возвращает уникальный идентификатор для этого узла. Если передать функции снова этот же узел, она вернет тот же идентификатор. Заметьте, что идентификаторы различаются в зависимости от процессора XSLT.

Следующий пример взят из главы 6, но сейчас я собираюсь добавить в planets, html оглавление, составленное из гиперссылок. Для создания оглавления я прохожу в цикле по всем планетам при помощи элемента <xsl:for-each>. На каждом шаге цикла для текущей планеты я создаю гиперссылку и создаю атрибут HREF со значением, равным уникальному идентификатору, при помощи шаблона значений атрибута:

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xs1-"http://www.w3.org/1999/XSL/Transform">

    <xsl:tempiate match- /PLANETS'»
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <BODY>
                <H1>
эи                  The Planets Table
                </H1>
                <xsl:for-each select-"PLANET">
                    <H2><A HREF-"#{generate-1d()}">
```

```
            <xsl: value-of select»'NAME"/x/A>
          </H2>
          <P/>
        </xsl:for-each>
```

Этот код добавляет каждой планете идентификатор и создает нужные гиперссылки. Я могу создать закладки гиперссылки в HTML-таблице данных планет, установив по очереди атрибут <NAME> закладки в идентификатор для каждого элемента <PLANET> так, чтобы он стал гиперссылкой-назначением:

```
<?xml version-"!.. 0"?>
<xsl:stylesheet version-"!..O"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

    <xslrtemplate match-"/PLANETS">
        <HIML>

        </HIML>
    </xsl:template»

    <xslrtemplate match-"PLANET"»
        <TR>
            <ID><A NAME-*{generate-id(.)}"">
            <xsl:value-of select-"NAME"»</A»</TD»
            <T0»<xsl:apply-templates select-"MASS"/»</TD>
            <TDxxsl: apply-templ ates select-"RADIUS"/></TD>
            <TD»<xslrapply-templates select-"DAY"/></TD>
        </TR>
    </xsl:template>

    <xslrtemplate match-"MASS">
        <xsl:value-of select-"."/>
        <xsl:text> </xsl:text>
        <xslrvalue-of select-"@UNITS"/>
    </xsl:template»

    <xslrtemplate match-"DAY">
        <xslrvalue-of select-"."/»
        <xslrtext> </xsl:text>
        <xslrvalue-of select-"@UNITS"/>
    </xslrtemplate»

</xslrstylesheet»
```

Вот и все; сейчас я создал гиперссылки с атрибутом HREF, значение которого равно идентификатору элемента <PLANET>; при помощи же этого идентификатора я сделал каждый элемент <PLANET> назначением гиперссылки.

Когда пользователь щелкает на гиперссылку в оглавлении, браузер прокручивает данные до соответствующей записи планеты в HTML-таблице. Вот как выглядит результирующий документ при использовании **процессора** Xalan:

```
<HTML>
    <HEAD>
        <TITLE>
            the Planets Table
        </TITLE>
    </HEAD>

    <BODY>
        <H1>
            The Planets Table
        </H1>
        <H2>
        <A href-"#N5">Mercury</A>
        </H2>
        <P></P>
        <H2>
        <A href-"#N20">Venus</A>
        </H2>
        <px/p>
        <H2>
        <A href-"#N3B">Earth</a>
        </H2>
        <P></P>

        <TABLE BORDER-"2">
            <TR>
                <TO>Name</TD>
                <TD>Mass</tO>
                <TD>Radius</TD>
                <TD>Day</TD>
            </TR>

            <TR>
                <TD><A NAME-"N5">Mercury</A>
                </TD><TD>.0553 (Earth - 1)</TD>
                <TD>1516 miles</TD>
                <TD>58.65 days</TD>
            </TR>

            <TR>
                <TD><A NAME-"N20">Venus</A></TD>
                <TD>.815 (Earth - 1)</TD>
                <TD>3716 miles</TD>
                <TD>116.75 days</TC>
            </TR>

            <TR>
                <TD><A NAME-"N3B">Earth</A></TD>
                <TD>1 (Earth - 1)</TD>
                <TD>2107 m1les</TD>
                <TD>1 days</TD>
            </TR>

        </TABLE>
    </BODY>
</HTML>
```

key()

Функция key служит для поиска узлов с определенным значением по именованному ключу, ее следует применять следующим образом:

node-set keyname, value)

Функции передаются имя ключа name в виде строки и требуемое значение ключа value, которое нужно найти. Функция возвращает набор узлов из найденных узлов.

Ключи создаются при помощи элемента <xsl:key>. Мы уже видели следующий пример в главе 4 — здесь я при помощи ключей выбираю планеты, чей атрибут COLOR установлен в «BLUE»:

```
<?xml version-"1.0"?>
<?xml-stylesheet type-"text/xml" href-"planets.xsl"?>
<PLANETS>
        .
        .
    <PLANET COLOR-"BLUE">
        <NAME>Earth</NAME>
        <MASS UNITS-"(Earth = 1)">1</MASS>
        <DAY UNITS-"days">l</DAY>
        «RADIUS UNITS-"miles">2107</RADIUS>
        «DENSITY UNITS-"(Earth - 1)">1</DENSITY>
        «DISTANCE UNITS-"mill1on miles"»128.4</DISTANCE»<!--B перигелии-->
    </PLANET>

</PLANETS»
```

Теперь при помощи <xsl:key> я могу создать ключ COLOR, выбирающий элементы <PLANET> и проверяющий их атрибут COLOR. Вот как он выглядит:

```
<?xml version-"!.O"?»
<xsl:stylesheet version«"l.0"
xmlns:xsl-'http://www.w3.org/1999/XSL/Transforrn">

    <xsl:key name-"COLOR" match-"PLANET" use-"GCOLOR"/»
        .
        .
```

Теперь я **Moiy** применить образец **«key()»** для выбора элементов <PLANET> с атрибутом COLOR со значением «BLUE» таким образом:

```
<?xml version-"1.0"?>
<xsl stylesheet version-"1 0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"»

    <xsl:key name-"COLOR" match-'TLANET" use-"@COLOR7>
                                                        r*          '?ps*2
    <xsltemplate match-"/PLANETS"»
        <HTML» .                                                    £>
        .
        .
```

```
                <TABLE BORDER-"2">
                    <TR>
                        <TD>Name</TD>
                        <TD>Mass</TD>
                        <TD>Radius</TD>
                        <TD>Day</TO>
                    </TR>
                    <xsl: apply-tempi ates select-"key('COLOR'.  'BLUE')"/»
                </TABLE>
            </BODY>
        </HTML>
    </xsl:tempiate>
```

И вот результат — как видите, единственной планетой, удовлетворившей образцу, была Земля:

```
<HTML>
    <HEAO>
        <TITLE>
            The Planets Table
        </TITLE>
    </HEAD>

    <BODY>
        <H1>
            The Planets Table
        </H1>

        <TABLE BORDER-"2">
            <TR>
                <TD>Name</TD>
                <TD>Mass</TD>
                <TD>Radius</TD>
                <TD>Day</TD>
            </TR>

            <TR>
                <TD>Earth</TD>
                <TD>1 (Earth - 1)</TD>
                <TD>2107 m1les</TD>
                <TD>1 days</TD>
            </TR>
        </TABLE>
    </BODY>
</HTML>
```

system-propertyO

Функция system-property возвращает значение нескольких системных свойств в виде строк; вот как ей следует пользоваться:

```
stri ng system-property(property)
```

Можно проверять следующие возможные значения системного свойства property:

- xsl:versi on. Возвращает версию XSLT;
- xsl:vendor. Возвращает строку, идентифицирующую производителя процессора XSLT;
- xsl:vendor-url. Возвращает URL производителя процессора XSLT.

Вот пример — версию XSLT можно проверить, вызвав system-property ('xsl: version');

```
<?xml version="1.0"?>
<xsl:stylesheet version-"2.0"
xmlns xsl-"http.//www w3.org/1999/XSL/Transform">

.
.
.

<xsl:i f text-"system-propertyC'xsl:versi on')-2.0">
    <xsl:namespace name-"starpowder"/>
</xsl:if>

.
.
.

</xsl:stylesheet>
```

Проверка версии XSLT может понадобиться, если вы хотите воспользоваться возможностями, которые появились в последних версиях.

unparsed-entity-uri()

Функция unparsed-entity-uri() предоставляет доступ к объявлениям неразобранных сущностей в DTD или схеме исходного документа. Неразобранная сущность (unparsed entity) — это обычно двоичные данные, такие как файл с рисунком. Функция применяется следующим образом:

```
string unparsed-entity-uri(name)
```

В функцию передается имя name неразобранной сущности, и функция возвращает для нее идентификатор. В следующем примере (листинг 8.6) я добавил в planets.xml DTD и объявил три неразобранных сущности, соответствующих рисункам планет — imagel, image2 и image3, — и обратился к ним, добавив в каждый элемент <PLANET> атрибут IMAGE.

Листинг 8.6. planets.xml с неразобранными сущностями

```
<?xml version="1.0"?>
<?xml-stylesheet type-"text/xml" href-"pianets.xsl"?>
<!DOCTYPE PLANETS [
<!ELEMENT PLANET (CUSTOMER)*>
<!ELEMENT CUSTOMER (NAME.MASS.RADIUS.DAY)>
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT MASS (#PCDATA)>
<!ELEMENT RADIUS (#PCDATA)>
<!ELEMENT DAY (#PCDATA)>
<!ENTITY imagel SYSTEM "http://starpowder.com/imagel.g1f" NDATA GIF>
<!ENTITY 1mage2 SYSTEM "http://starpowder.com/image2.gif" NDATA GIF>
```

```
<!ENTITY image3 SYSTEM "http://starpowder.com/image3.gif" NDATA GIF>
]>

<PLANETS>

    <PLANET IMAGE-"imagel">
        <NAME>Mercury</NAME>
        <MASS UNITS-"(Earth - 1)">.0553«/MASS>
        <DAY UN ITS-"days ">58.65</DAY>
        <RADIUS UNITS-"miles">1516</RADIUS>
        «DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
        «DISTANCE UNITS-"million miles">43.4«/DISTANCE><!—B перигелии—>
    </PLANET>

    «PLANET IMAGE«"image2">
        <NAME>Venus«/NAME>
        «MASS UNITS-"(Earth - 1)">.815«/MASS>
        «DAY UNITS-"days">116.75</DAY>
        «RADIUS UNITS-"miles">3716«/RADIUS>
        «DENSITY UNITS-"(Earth - 1)">.943«/DENSITY>
        -DISTANCE UNITS-"million mi 1es">66.8</DISTANCE<@062><!—B перигелии—>
    </PLANET>

    <PLANET IMAGE "image3">
        <NAME5>Earth«/NAME>
        «MASS UNITS-"(Earth - 1)">1«/MASS>
        «DAY UNITS-"days">l«/DAY>
        «RADIUS UNITS-"miles">2107«/RADIUS>
        «DENSITY UNITS-"(Earth - 1)">1«/DENSITY>
        «DISTANCE UNITS-"million miles">128.4«/DISTANCE><!—B перигелии—>
    </PLANET>

</PLANETS>
```

(Заметьте, что некоторые процессоры XSLT основаны на разборщиках XML, которые проверяют, могут ли они найти сущность по заданному URI, — потому, если вы хотите опробовать этот пример, вставьте URI каких-либо настоящих рисунков.) Теперь я могу извлечь URI рисунков в таблице стилей при помощи функции unparsed-entity-uri и создать элементы HTML <IMG>, применив эту функцию в результирующем документе (листинг 8.7).

Листинг 8.7. Применение unparsed-entity-uri

```
<?xml version-"1.0"?>
«xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

    «xsl:tempiate match="/PLANETS">
        <HTML>


        </HTML>
    «/xsl:tempiate>

    «xslrtemplate match-"PLANET">
       <TR>
```

продолжение &

Листинг 8.7 (продолжение)

```
          <TD><xsl:value-of select-"NAME"/><IMG SRC-"{unparsed-entity-uri(PIHAGE)}"/></TD>
          <TD><xsl:apply-tempiates select»"MASS"7></TD>
          <TDxxsl: apply-templ ates select:"RAOIUS"/></TD>
          <TD><xsl:apply-templates select="DAY"/></TD>
        </TR>
    </xsl:tempiate>

    <xslrtemplate match-"MASS">
        <xsl:value-of se^ct-" ."/>
        <xsl:text> </xsl text>
        <xsl:value-of select="@UNITS"/>
    </xsl:template»



    <xsl:temp>ate match-"DAY">
        <xsl:value-of select-"."/>
        <xsl:text> </xsl text>
        <xsl:value-of select-"@UNITS"/>
    </xsl:tempiate>

</xsl:stylesheet»
```

И вот результирующий документ:

```
<HIML>
    <HEAD>
        <TITL£>
            The Planets Table
        </TITL£>
    </HEAO>

    <BODY>
        <H1>
            The Planets Table
        </HI>

        <TABLE BORDER-"2">
            <TR>
                <TD>Name</TD>
                <TD>Mass</TD>
                <TD>Radius</TD>
                <TD>Day</TD>
            </TR>

            <TR>
                <TD>Mercury<IMG SRC-"http://starpowder.com/imagel.gi f'></TD>
                <TD>.0553 (Earth - 1)</TD>
                <TD>1516 miles</TD>
                <TD>58.65 days</TD>
            </TR>

            <TR>                                                    i
                <TD>Venus<IMG SRC-"http-//starpowder.com/1mage2.gi f'></TD>
                <TD> 815 (Earth - 1)</TD>
                <TD>3716 miles</TD>
                <TD>116.75 days</TD>
            </TR>
```

```
            <TR>
                <TD>Earth<IMG SRO"http: //starpowder.com/image3.g1f"></TD>
                <TD>1 (Earth - 1)</TD>
                <TD>2107 miles</TD>
                <TD>1 days</TD>
            </TR>

        </TABLE>
    </BODY>
</HTML>
```

На этом мы заканчиваем рассмотрение функций XSLT. Я перехожу к функциям XPath, начиная с функций для работы с наборами узлов.

Функции XPath для работы с наборами узлов

Следующие функции XPath работают с наборами узлов:

- count(node-set). Возвращает число узлов в наборе узлов;
- id (string ID). Возвращает набор узлов элемента, ID которого совпадает с переданной функции строкой, или пустой набор узлов, если таковых элементов не оказалось;
- last(). Возвращает число узлов в наборе узлов;
- local-name(node-set). Возвращает локальное имя первого узла в наборе узлов;
- name(node-set). Возвращает полное имя первого узла в наборе узлов;
- namespace-uri(node-set). Возвращает URI пространства имен первого узла в наборе узлов;
- position(). Возвращает позицию контекстного узла в контекстном наборе узлов, начиная с 1.

count()

Функция count подсчитывает число узлов в наборе узлов.

```
number ccunt(node-set)
```

Функция принимает набор узлов и возвращает количество узлов в этом наборе.

Следующий пример применения функции count мы уже рассматривали в главе 6. В этом случае набор узлов образован из всех элементов <PLANET> в planets, xml; я получил его при помощи пути расположения «//PLANET»:

```
<xsl stylesheet
 xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"
 version="1.0">

<xsl:output method-"xml" indent-"yes"/>

<xsl template match-"*">
  '   <xsl:copy>
        <xsl;apply-templates/>
    </xsl:copy>
```

```
</xsl template»

<xslrtemplate match-"PLANET">
    <xsl:copy use-attribute-sets-"numbering">
        <xsl:apply-templates/>
    </xsl:copy>
</xslrtemplate»

<xsl:attribute-set name»"lumbering"»
    <xsl attribute name-"number"><xsl:number/></xslattribute»
    <xslattribute name-"total"><xsl:value-of select~"count(//PLANET)"/>
    </xsl:attribute»
</xsl:attribute-set»

</xsl:stylesheet»
```

Заметьте, что в приведенном ниже результате каждый элемент <PLANET> обладает и атрибутом number, и атрибутом total, а атрибут total хранит общее количество элементов <PLANET>, которое было найдено при помощи count:

```
<?xml version-"1.0" encochng-"UTF-8"?>
«PLANETS»

    <PLANET number-"1" total«"3">
        <NAME>Mercury«/NAME»
        <MASS>.0553</MASS>
        <DAY»58.65</0AY>
        <RADIUS»1516</RADIUS>
        <DENSITY».983</DENSI1Y>
        <DISTANCE»43.4</DISTANCE>
    </PLANET»

    <PLANET number-"2" total-"3">
        <NAME»Venus</NAME»
        <MASS>.815</MASS>
        <DAY»116.75</DAY>
        <RAD1US»3716</RADIUS»
        «DENSITY». 943</DENSITY>
        <DISTANCE»66,8</DISTANCE»
    «/PLANET»

    «PLANET number="3" total-"3">
        <NAME»Ea rth</NAME»
        «MASS»1«/MASS>
        <DAY»1</DAY>
        <RADIUS»2107«/RADIUS»
        «DENSITY»1</DENSITY>
        <DISTANCE»128.4«/DISIANCE»
    «/PLANET»

«/PLANETS»
```

id()

Функция id возвращает набор узлов, в котором все узлы имеют такой же ID, как и переданный в эту функцию. Функция применяется следующим образом:

```
node-set id(id-value)
```

В функцию передается значение ID, и она возвращает набор узлов с узлами, имеющими этот ID. Заметьте, что вы можете задать несколько ID, разделенных символами-разделителями, и функция вернет набор узлов с элементами, обладающими этими ID.

Следующее правило-пример выбирает текст всех элементов с ID «favorite»:

```
«xsl template match - "id('favorite')"»
    <H3><xsl:va1ue-of select»"."/></H3>
</xsl.tempiate>
```

Отметьте также, что вы должны объявлять значения ID как в DTD или схеме. Объявление DTD для planets.xml, объявляющее ID с именем id и присваивающее ему значение «favorite», может выглядеть так:

```
<?xml version-"1.0"?>
<?xml-stylesheet type-"text/xml" href-"stylesheet.xsl"?>
<!DOCTYPE PLANETS [
<!ELEMENT PLANET (CUSTOMER)*»
«!ELEMENT CUSTOMER (NAME.MASS.RADIUS.DAY)>
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT MASS (#PCDATA)>
<!ELEMENT RADIUS (#PCDATA)>
<!ELEMENT DAY (#PCDATA)>
<!ATTLIST PLANET
    id ID #REQUIRED>
]>
<PLANETS>
    «PLANET 1d-*favor1te'>
        <NAME>Mercury</NAME>
        <MASS UNITS-"(Earth - 1)">.0553</MASS>
        <DAY UNITS-"days">58.65</DAY>
        «RADIUS UNITS-"miles">1516«/RADIUS>
        «DENSITY UNITS-"(Earth - 1)">.983«/DENSITY>
        «DISTANCE UNITS="million miles">43.4«/DISTANCE><!--B перигелии-»
    «/PLANET>
```

Заметьте, что некоторые процессоры XSLT не могут осуществлять выборку по ID, потому что они не читают объявления DTD и схему XML. Это одна из задач, которые призван решить XSLT 2.0 — сделать доступной информацию об идентификаторах.

last()

Функция last возвращает число узлов в наборе узлов, поэтому ее значение равно позиции последнего узла. Она применяется следующим образом:

```
number last()
```

Следующий пример из главы 5 последовательно выводит в результирующий документ имена планет. Однако я хочу, чтобы выводилось не просто «The first three planets are: Mercury Venus Earth» (первые три планеты: Меркурий Венера

Земля), а «The first three planets are: Mercury, Venus, and Earth.» Необходимы^ знаки пунктуации можно добавить, определяя элемент, с которым мы работаем, при помощи функции position и проверяя позицию при помощи <xsl:if>:

```
<?xml version-"1.0"?>
<xsl stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">
<xml:output method-"xmV7>

<xsl template match-"PLANETS">
<DOCUMENT>
    <TITLE>
        The Planets
    </TITLE>
    <PLANETS>
        The first three planets are: <xsl:apply-tempiates select-"PLANET"/>
    </PLANETS>
</DOCUMENT>
</xsl.tempiate>

<xsl:tempi ate match-"PLANET">
    <xsl:value-of select-"NAME"/>
    <xsl:1f test^"position()!-last()">. </xsl:if>
    <xsl:1f test="posit1on()-last()•l">and </xsl:if>
    <xsl:if test-"position()=last()">.</xsl:if>
</xsl:template»

</xsl:stylesheet>
```

Вот результат:

```
<?xml version-"l 0" encoding="UTF-8"?>
<DOCUMENT>

    <TITLE*
        The Planets
    </TITLE>

    <PLANETS>
        The first three planets are: Mercury. Venus, and Earth.
    </PLANETS>

</DOCUMENT>
```

local-name()

Функция local-name возвращает локальное (неполное) имя узла. Функция применяется следующим образом:

```
string local-name(node-set?)
```

В функцию передается набор узлов с единственным узлом, и функция возвращает локальное имя этого узла. (Если в наборе узлов больше, чем один узел, берется только первый.) Если не передать никаких узлов, функция вернет локальное имя контекстного узла.

В следующем примере я создаю новые элементы при помощи <xsl:element> и определяю имена контекстных узлов при помощи local-name:

```
<?xml version-"1.0"?>
<xs1:stylesheet version-"1 O"
xmlns:xsl""http://www.w3.org/1999/XSL/Transf0rra">
    <xsl:output method-"xml"/>

    <xsl:tempi ate match-"PLANETS">
        <xsl:element name-"(1ocal-name(.)}">
            <xsl:for-each select-"PLANET">
                <xsl -.element name-"{local -name(. )}">
                    <xsl:for-each select-"*">
                        <xs1:element name-'DATA">
                            <xsl:value-of select-"."/>
                        </xsl:element>
                    </xsl:for-each>
                </xsl:element>
            </xsl:for-each>
        </xsl element>
    </xsl:tempiate>

</xsl:stylesheet»
```

name()

Функция name аналогична local-name, за тем исключением, что она возвращает полностью определенное имя узла. Вот как следует применять эту функцию:

```
string name(node-set?)
```

namespace-uri()

Функция namespace-uri возвращает строку, содержащую URI пространства имен в расширенном имени узла. Как правило, это URI в объявлении пространства имен, установленное при помощи атрибутов xmlns или xmlns:prefix. Функция применяется следующим образом (заметьте, что эту функцию можно применять только к элементам или атрибутам, результатом обработки остальных узлов будет пустая строка):

```
string namespace-uri(node-set?)
```

Например, в planets.xml я могу добавить пространство имен «star» (звезда):

```
<?xml version-"1.0"?>
<?xml-stylesheet type-"text/xml" href-"planets.xsl"?>
<star:PLANETS xmlns:star-"http://starpowder.com">

    <star:PLANET»
        <star:NAME»Mercury</star:NAME»
        <star:MASS UNITS-"(Earth - l)"».0553</star:MASS»
        <star:DAY UNITS-"dayS"»58.65</star:DAY»
        <star:RADIUS UNITS-"n)11es">1516</star:RADIUS»
        <star:DENSITY UNITS-"(Earth - l)"».983</star:DENSITY»
        <star:DISTANCE UNITS-'nrillion mi1es*»43.4</star:DISTANCEx1 - -B перигелии-->
    </star:PLANET>                                                    «»' •
```

```
<star:PLANET»
    <star:NAME>Venus</star:NAME>
    <star:MASS UNITS-"(Earth - l)"».815</star:MASS>
    <star:DAY UNITS-"days">116.75</star:DAY>
    <star:RADIUS UNITS-"miles">3716</star:RADIUS>
    <star:DENSITY UNITS-"(Earth - l)">.943</star:DENSITY>
    <star:DISTANCE UNITS-"million miles">66.8</star:DISTANCE><!--B перигелии-•>
</star:PLANET>
    .
    .
    .
```

И я могу найти URI этого пространства имен в таблице стилей при помощи namespace-uri:

```
<?xml version-'! 0"?>
<xsl:stylesheet version«"1.0"
xmlns:xs1-"http://www.w3.org/1999/XSL/Trans form"
xmlns:star-"http://www.starpowder.com">

<xslrtemplate match-"/PLANETS">
    <xsl:value-of select«"namespace-uri()"/>
</xsl:tempiate>
    .
    .
    .
```

Вот результат:

```
<?xml version-"1.0" encoding-"UTF-8"?>
http://starpowder.com
```

position()

Функция position возвращает позицию контекстного узла:

```
number positionO
```

С этой функцией мы встречались на протяжении всей книги. Такой пример вы уже видели раньше в этой главе, в нем для создания предложения «The first three planets are: Mercury, Venus, and Earth.» используются функции last и position:

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www w3.org/1999/XSL/Transform">
<xml:output method-"xml"/>

<xslrtemplate match-"PLANETS">
<DOCUMENT>
    <TITLE>
        The Planets
    </TITLE>
    <PLANETS>
        The first three planets are: <xs>:apply-templates select-"PLANET"/>
    </PLANETS>
</DOCUMENT>
</xsl:tempiate>

<xslrtemplate match-"PLANET">
```

```
    <xsl:value-of select-"NAME"/»
    <xsl:1f test-"pos1t1on()l-last()"». </xsl:if>
    <xsl:1f test-"pos1t1on()-last()-l">and </xsl:1f>
    <xsl:1f test-"pos1t1on()-last()"».«/xs1:1f>
</xsl:template»

</xsl stylesheet»
```

Полученный результат:

```
<?xml version»"1.0" encod1ng-"UTF-8"?>
«DOCUMENT»

    <TITLE>
        The Planets
    «/TITLE»

    «PLANETS»
        The first three planets are: Mercury, Venus, and Earth.
    «/PLANETS»

«/DOCUMENT»
```

Функции XPath для работы со строками

В XSLT доступны следующие функции XPath для работы со строками:

- concat(string stringl, string string2,...). Возвращает конкатенацию (объединение) всех переданных ей строк;
- contains(string stringl, string string2). Возвращает истину, если первая строка содержит (contains) вторую строку;
- normalize-space(string stringl). Возвращает строку stringl (или контекстный узел, если строки stringl нет) после отбрасывания лидирующих и завершающих символов-разделителей и замены нескольких последовательных разделителей на один пробел;
- starts-with(string stringl, string string2). Возвращает истину, если первая строка начинается (starts with) со второй подстроки;
- string (object). Преобразует объект в строку;
- string-length(string stringl). Возвращает количество символов в строке stringl;
- substring(string stringl, number offset number length). Возвращает length символов из строки, начиная со смещения offset;
- substring-after(string stringl, string string2). Возвращает часть строки stringl после первого вхождения string2;
- substring-before(string stringl, string string2). Возвращает часть строки stringl вплоть до первого вхождения строки string2;
- translate(string stringl, string string2, string string3). Возвращает строку stringl, в которой все вхождения символов в строке string2 заменены на соответствующие символы в строке string3;

В последующих разделах я рассмотрю каждую из этих функций.

concat()

Функция concat объединяет вместе все переданные ей строки и возвращает полученную строку:

```
concat(string string1. string string2. ...)
```

В качестве примера рассмотрите разработанный ранее вариант таблицы стилей planets.xsl, отображающей значения элементов и значения атрибутов UNITS, в котором шаблоны применяются следующим образом:

```
<xs1rtemplate match-"MASS">
        <xsl:value-of select-"."/>
        <xsl text> </xsl:text>
        <xsl ;value-of select-"(auNITS'7>
    </xsl:tempiate>
```

Такой код отображает строковое значение, контекстный узел, пробел и строковое значение атрибута UNITS. Но код можно значительно сократить при помощи функции concat (листинг 8.8).

Листинг 8.8. Применение функции concat

```
<?xml version="l 0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

    <xslrtemplate match-"/PLANETS">
        <HIML>
            .
            .
            .
        </HIML>
    </xslrtemplate»

    <xslrtemplate match-"PLANET">
        <TR>
            <TD><xsl:value-of select-"NAME"/></TD>
            <TDxxsl: apply-templ ates select-"MASS"/»</TD>
            <T0><xsl rapply-templates select-"RA0IUS'7x/TD>
            <T0><xslrapply-templates select-"DAY"/></TD>
        </TR>
    </xslrtempiate>

    <xslrtemplate match-"MASS">
        <xsl:value-of select-"concat(.. ' '. &UNITS)"/>
    </xsl.template»

    <xslrtemplate match-"RADIUS">
        <xsl:value-of select-"concat(,. ' '. GUNITS)"/»
    </xsl r template»

    <xsl template match-"DAY">
        <xsl:value-of select-"concat(., ' \ @UNITS)"/>
    </xslrtemplate»

</xslrstylesheet»
```

contains()

Функция contains проверяет, содержится ли одна строка внутри другой; если это так, функция возвращает истину, если нет — ложь. Функция применяется следующим образом:

```
boolean contains(container-string, contained-string)
```

Следующий пример взят из главы 7; в этом случае я хочу осуществить поиск слова «miles» во всех атрибутах, и если оно будет найдено, добавить в результирующий документ текст «You should switch to kilometers.» (Нужно перевести в километры.):

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www w3.org/1999/XSL/Transform">

    <xsl:tempi ate match-"/PLANETS">
        <HIML>
            .
            .
            .
        </HTML>
    </xsl:template»

    <xsl;template match="PLANET">
        <TR>
            <TD><xsl:value-of select="NAME"/></TD>
            <TD><xsl: apply-templ ates select="MASS"/x/TD>
            <TD><xsl:apply-templates select="RADIUS"/></TD>
            <TD><xsl apply-templates select-"DAY'7></TD>
            <TD><xsl.apply-templates select-"DISTANCE"/»</TD>
        </TR>
    </xsl:template»

    <xsltemplate match-"HASS">
        <xsl value-of select-" "/>
        <xsl:text> </xsl:text>
        <xsl:value-of select-"@UNITS"/>
    </xsl:tempi ate»
            .
            .
            .
    <xsltemplate match-"DISTANCE">
        <xsl:value-of select-"."/»
        <xsl:text» </xsl:text>
        <xsl:value-of select-'GIMTS"/»
    </xsl template»

    <xsl:template match-"//*[contains(@UNITS, 'miles')]"»
        <xsl:value-of select-"."/»
        <xsl:text» </xsl:text»
        <xsl:text»You should switch to kilometers.</xsl:text»
    </xslrtemplate»

</xsl:stylesheet»
```

Вот результирующий документ:

```
<HTML>
   <HEAD>
      <TITLE>
         The Planets Table
      </TITLE>
   </HEAD>

   <BODY>
      <H1>
         The Planets Table
      </H1>

      <TABLE BORDER-"2">
         <TR>
            <TD>Name</TD> .
            <TD>Mass</TD>
            <TD>Radius</TD>
            <TD>Day</TD>
            <TD>D1stance</TD>
         </TR>

         <TR>
            <TD>Mercury</TD>
            <TD>.0553 (Earth - 1)</TD>
            <TD>1516 You should switch to kilometers.</TD>
            <TD>58.65 days</TD>
            <TD>43.4 You should switch to kilometers.</TD>
         </TR>

         <TR>
            <TD>Venus</TD>
            <TD>.815 (Earth = 1)</TD>
            <TD>3716 You should switch to ktlometers.</TD>
            <TD>116.75 days</TD>
            <TD>66.8 You should switch to kilometers.</TD>
         </TR>

         <TR>
            <TD>Earth</TD>
            <TD>1 (Earth - 1)</TD>
            <TD>2107 You should switch to kilometers.</TD>
            <TD>1 days</TD>
            <TD>128.4 You should switch to kilometers.</TD>
         </TR>

      </TABLE>
   </BODY>
</HTML>
```

normalize-space()

Функция normalize-space удаляет лидирующие и завершающие символы-разделители и сжимает все внутренние идущие подряд разделители в один пробел, возвращая полученную строку. Функция применяется следующим образом:

string normalize-space(string?)

В следующем примере я добавил дополнительные пробелы в атрибут UNITS элемента <MASS> Меркурия:

```
<?xml version-"1.0"?»
<7xml-stylesheet type-"text/xml" href-"planets xsl"?>
<PLANETS>

    <PLANET>
        <NAME»Mercury</NAME>
        <MASS UNITS-" (Earth - 1) ">.0553</MASS>
        <DAY UNITS-"days">58 65</DAY>
        <RADIUS UNITS-"miles">1516</RADIUS>
        <DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
        <DISTANCE UNITS-"million miles">43.4</DISTANCE><!~B перигелии--»
    </PLANET>
```

Функция normalize-space поможет удалить лишние пробелы в таблице стилей:

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

    <xsl rtemplate match-"/PLANETS"»
        <HTML»

        </HTML>
    </xslrtemplate»

    <xslrtemplate match-"PLANET">
       <TR>
          <TD»<xslrvalue-of select-"NAME'7»</T0>
          <TD»<xslrapply-templates select-"MASS"/»</TD>
          <TD»<xslrapply-temp'ates select-"RADIUS"/»</TD»
          <TD»<xsl:apply-templates select-"DAY"/»</TD>
       </TR>
   </xslrtemplate»

    <xslrtemplate match="MASS">
        <xslrvalue-of select-"."/>
        <xslrtext» </xslrtext»
        <xsl:value-of select-"norma1ize-space(©UNITS)"/>
    </xslrtemplate»
```

И вот результат — обратите внимание на то, что лишние пробелы убраны:

```
<HIML>
    <HEAD>
        <TITLE»
            The Planets Table
        </TITLE>
    </HEAD>
```

```
<BODY>
    <H1>
        The Planets Table
    </H1>
    <TABLE BORDER="2">
        <TR>
            <TD>Name</TD>
            <TD>Mass</TD>
            <TD>Radius</TD>
            <TD>Day</TD>
        </TR>

        <TR>
            <TD>Mercury</TD>
            <TD>.0553 (Earth - 1)</TD>
            <TD>1516 miles</TD>
            <TD>58.65 days</TD>
        </TR>
```

starts-with()

Как можно догадаться по имени (начинается с), функция starts-with проверяет, начинается ли одна строка с другой.

```
boolean starts-with(string-to-examine. possible-start-string
```

В этом примере из главы 4 при помощи starts-with выбираются текстовые узлы, текст в которых начинается с «E», для того чтобы выбрать Earth (Землю). Затем в описание Земли добавляется текст «(the World)» (мир), и получается •Earth (the World)»:

```
<?xml version="1.0"?>
<xsl:stylesheet vers1on="l 0"
xmlnsrxsl="httpr//vMtf.w3.org/1999/XSL/Transfomf>

    <xslrtemplate match="/PLANETS">
        <HTML>
            <HEAD>


            </BODY>
        </HTML>
    </xsl:template>

    <xslrtemplate match="PLANET">
        <TR>
            <TD><xslrapply-templates select="NAME"/x/TD>
            <TD><xslrapply-templates select="MASS"/></TD>
            <TD><xslrapply-templates select="RADIUS"/></TD>
            <TD><xsl apply-templates select-"DAY"/x/TD>
        </TR>
    </xslrtempiate>
```

```
    <xsl-.template match-"text()[starts-w1th(., 'E')]">
        <xsl:text>(the World)</xsl:text>
    </xsl:template»

    <xslrtemplate match™"NAME'>
        <xsl:value-of select-"."/»
        <xsl:text> </xsl:text>
        <xsl:value-of select-"@UNITS"/>
        <xsl:apply-templates/»
    </xsl:tempiate»

    <xslrtemplate match-"DAY"»
        <xsl:value-of select-"."/»
        <xsl:text> </xsl:text>
        <xsl:value-of select-'GUNITS"/»
    </xsl template»

</xsl:stylesheet»
```

Вот результат — заметьте, что подпись для Земли теперь выглядит как «Earth (the World)»:

```
<HIM>
    <HEAD>
        <TITLE>
            The Planets Table
        </TITLE»
    </HEAD>

    <BODY>
        <HI>
            The Planets Table
        </HI>

        <TABLE BORDER-"2">
            <TR>
                <TD»Name</TD>
                <TD»Mass</TD>
                <TD»Radius</TD>
                <TD»Day</TD>
            </TR>

            <TR>
                <TD»Earth (the World)</TD»
                <TD»1 (Earth - 1)</TD»
                <TD»2107 miles</TD»
                <TD»1 days</TD»
            </TR>
        </TABLE>
    </BODY>
</HTML»
```

string()

Функция string просто преобразует переданный ей объект в строку. Функция применяется следующим образом:

```
string string(object?)
```

Как правило, необходимости в данной функции не возникает, потому что преобразования такого рода осуществляются автоматически. Мне на ум приходит всего несколько примеров, когда эта функция действительно нужна. Пусть, например, по какой-то причине вы поместили в каждый элемент <PLANET> в planets.xml по три элемента <NAME> и хотите использовать в качестве фактического названия планеты только первый элемент <NAME>:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml" href-"planets.xsl"?>
<PLANETS>

    <PLANET>
        «NAME>Mercury«/NAME>
        <NAME>Venus</NAME>
        <NAME>Earth</NAME>
        <MASS UNITS-"(Earth - 1)">.0553</MASS>
        <DAY UNITS-"days">58 65</DAY>
        «RADIUS UNITS-"miles">1516«/RADIUS>
        «DENSITY UNITS-"(Earth - 1)">.983«/DENSITY>
        «DISTANCE UNITS-"million miles">43.4«/DISTANCE><«!--B перигелии->
    </PLANET>

    <PLANET>
        «NAME>Venus</NAME>
        «NAME>Earth«/NAME>
        <NAME>Mercury«/NAME>
        <NAME>Planet of Love.«/NAME>
        «MASS UNITS*"(Earth - 1)"> 815«/MASS>
        «DAY UNITS-"days">116.75«/DAY>
        «RADIUS UNITS-"miles">3716«/RADIUS>
        «DENSITY UNITS-"(Earth - 1)">.943«/DENSITY>
        «DISTANCE UNITS-"million miles">66.8«/DISTANCE><«!--B перигелии-->
    </PLANET>

    <PLANET>
        <NAME>Earth«/NAME>
        «NAME>Mercury«/NAME>
        <NAME>Venus</NAME>
        <NAME>The planet you're standing on.«/NAME>
        «MASS UNITS-"(Earth - 1)">1«/MASS>
        «DAY UNITS-"days">l«/DAY>
        «RADIUS UNITS="miles">2107«/RADIUS>
        «DENSITY UNITS-"(Earth - 1)">1</DENSITY>
        «DISTANCE UNITS="million mi 1es">128.4</DISTANCEx!--B перигелии-->
    </PLANET>

«/PLANETS>
```

Предположим теперь, что вам нужно выбрать определенную планету, например, Венеру (Venus). Такая проверка не пройдет, потому что NAME вернет набор

узлов из всех дочерних элементов <NAME> контекстного узла, а так как у каждой планеты есть элемент <NAME> со значением «Venus», такое условие всегда будет истиной:

```
<xsl rtemplate match-"PLANET"»
<xsl:if test-"NAME-'Venus'"»
   <TR»
      <TD><xsl: value-of sel ect-" NAME "/x/TD>
      <TD><xsl:apply-templates select="MASS"/></TD>
      <TD><xsl:apply-templates select="RADIUS"/></TD>
      <TD><xsl:apply-templates select-"DAY'7»</TD>
   </TR>                                  I
</xsl:if>
</xsl:tempiate>
```

Для того чтобы проверить первый элемент <NAME> в каждом элементе <PLANET>, можно воспользоваться функцией string, поскольку она возвращает не набор узлов, а строку:

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"I.0"
xmlns:xs1-"http://www w3.org/1999/XSL/Transform">

   <xsl .-template match-"/PLANETS">
      <HIML>
         .
         .
         .
      </HTML>
   </xsl:tempiate>

   <xsl:tempi ate match-"PLANET">
   <xsl:if test-"string(NAME)-'Venus'"»
      <TR>
         <TD><xsl:value-of select-"NAME"/></TD>
         <TD><xsl:apply-templates select-"MASS"/»</TD>
         <TD><xsl:apply-templates select="RADIUS"/></TD>
         <TD><xsl:apply-templates select-"DAY'7></TD>
      </TR>
   </xsl:if>
  </xsl:tempiate>

   <xsl:tempi ate match-"MASS">
      <xsl:value-of select-"."/>
      <xsl:text> </xsl:text>
      <xsl:value-of select-"@UNITS"/>
   </xsl:template»
         .
         .
         .
   <xslrtemplate match-"DAY">
      <xsl value-of select-"."/»
      <xsl:text> </xsl:text>
      <xsl:value-of select-"@UNITS"/>
   </xsl:template»

</xsl:stylesheet»
```

Конечно, если нужно только выбрать первый дочерний элемент <NAME> контекстного узла, проще воспользоваться NAME[1].

string-lengthQ

Как можно предположить, функция string-length возвращает длину (length) переданной ей строки. Функция применяется таким образом:

```
number string-length(string?)
```

В следующем примере я определяю длину названия каждой планеты при помощи string-length:

```
<?xml vers1on="l.0"?>
<xSl:stylesheet versi on-"1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"»

    <xsltemplate match-"/PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    Length of Planet Names
                </TITLE>
            </HEAD>
            <BODY>
                <Hl>
                    Length of Planet Names
                </Hl>
                    <xsl:apply-templates/>
            </BODY>
        </HTML>
    </xsl:template»

    <xsl:tempi ate match-"PLANET"»
          <xsl:value-of select-"NAME7> is <xsl:value-of select-"str1ng-length(NAME)"/»
          characters long.
          <BR/>
   </xsl-.template»

    <xsl:template match-"*"»
    </xslrtemplate»

</xsl:stylesheet»
```

А вот результат:

```
<HTML>
    <HEAD»
        <TITLE>
            Length of Planet Names
        </TITLE>
    </HEAD>

    <BODY>
        <Hl»
            Length of Planet Names
        </Hl>
```

```
        Mercury 1s 7 characters long.
        <BR>

        Venus is 5 characters long.
        <BR>

        Earth is 5 characters long.
        <BR>

    </BODY>
</HTML>
```

Этот результирующий документ показан на рис. 8.2.

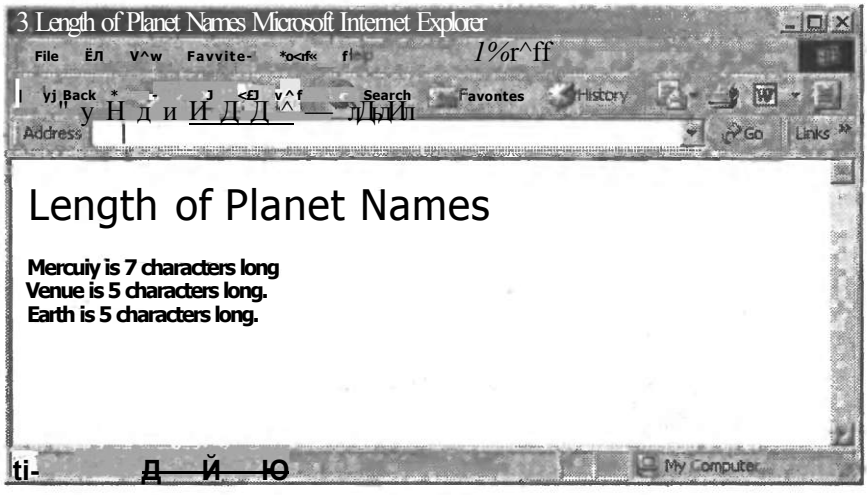


Рис. 8.2. Определение длины строк в XSLT

substring()

Функция substring возвращает подстроку переданной ей строки. Функция применяется следующим образом:

```
string substring(source-string, start-position, number-of-characters?)
```

Функция принимает параметры source-string (исходная строка), start-position (начальная позиция) и необязательный параметр number-of-characters (количество символов). Функция возвращает подстроку исходной строки с начальной позиции и до указанного количества символов или до конца строки, если это количество не задано.

Функция substring — одна из трех функций, оперирующих с подстроками: substring-before, возвращающая строку перед найденной подстрокой, сама substring, возвращающая заданную подстроку, и substring-after, возвращающая строку после найденной подстроки. В следующем примере (листинг 8.9) задействованы все три функции: в этом случае я разбил название планеты Меркурий (Mercury) на три подстроки — «Мег», «с» и «игу» — и затем объединил их снова. Вот как это можно

сделать при помощи трех рассматриваемых функций (дополнительная Информация о substring-before **и** substring_after **приведена в следующих двух разделах).**

Листинг 8.9. Применение функций substring-before, substring и substring-after

```
<?xml version«"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xsl="http.//www.w3.org/1999/XSL/Transform">

    <xsl template match»"/PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    Planetary Information
                </TITLE>
            </HEAD>
            <BODY>
                <H1>
                    Planetary Information
                </H1>
                    <xsl:apply-templates/>
            </BODY>
        </HTML>
    </xsl:tempiate>

    <xsltemplate match-"PLANET">
        <xsl:1f test-"NAME-'Mercury"">
          The first planet is
          <xsl:value-of
              select-"concat(substring-before(NAME.  'c').
              substring(NAME, 4, 1). substring-after(NAME.  'c'))"/».
          <BR/>
        </xsl:1f>
    </xsl:template»

    <xslrtemplate match-"*"»
    </xsl:template»

</xsl stylesheet»
```

Вот результирующий документ, созданный этим преобразованием:

```
<HTML>
    <HEAD>
        <TITLE>
            Planetary Information
        </TITLE>
    </HEAD>
    <BODY>
        <H1>
            Planetary Information
        </H1>

        The first planet is Mercury.
        <BR>

    </BODY>
</HTML>
```

substring-after()

Функция substring-after возвращает подстроку, которая следует за найденной строкой. В эту функцию передается строка, подстрока, которую нужно найти внутри строки, — и функция возвращает подстроку, расположенную после совпадающей части строки, если она была найдена; иначе функция возвращает пустую строку. Функция применяется следующим образом:

```
string substring-after(string, string-to-match)
```

substring-beforeQ

В функцию substring-before передается строка, подстрока, которую нужно найти внутри строки, и функция возвращает подстроку, расположенную перед совпадающей частью строки, если она была найдена; иначе функия возвращает пустую строку. Функция применяется следующим образом:

```
string substring-before(string, string-to-match)
```

В разделе substring() приведен пример применения функций substring-before, substring и substring-after.

translate()

Функция translate служит для перевода иди замены определенных символов. (Эта функция во многом похожа на операцию tr в Perl, если вы знакомы с операциями.) В функцию передаются две строки: одна задает список символов поиска, вторая задает список символов, которыми нужно заменить найденные символы. Функция применяется следующим образом:

```
string translate(string, from-characters, to-characters)
```

Например, если третий символ из from-characters будет найден в string, то третий символ из to-characters заменит его в результирующей строке.

В следующем примере такая функция translate

```
translate("steve-starpowder.com". "-". "@")
```

возвратит результирующую строку «steve@starpowder.com».

В этом примере я просто привожу строку к нижнему регистру:

```
translateC'XSLT". "ABCDEFGHIDKLMNOPQRSTUWXYZ'. "abcdefghijklmnopqrstuvwxyz")
```

Функции XPath для работы с числами

В XPath имеются следующие функции для работы с числами:

- ceiling(). Возвращает наименьшее целое, большее переданного функции числа;
- floorQ. Возвращает наибольшее целое, меньшее переданного функции числа;

- number(). Приводит переданный ей параметр к числу;
- round(). Округляет переданное функции число до ближайшего целого;
- sum(). Возвращает сумму переданных функции чисел.

В следующих разделах мы рассмотрим примеры применения каждой из этих функций.

ceiling()

Функция ceiling возвращает наименьшее целое, которое все еще больше переданного функции числа, то есть она возвращает следующее большее целое. Функция применяется следующим образом:

```
number ceiling(number)
```

Например, выражение

```
ceiling(3.1415926535)
```

возвращает 4.

floor()

Функция floor — парная к функции ceiling. Она возвращает наибольшее целое, которое все еще меньше переданного ей числа. Другими словами, функция возвращает предыдущее целое. Функция применяется следующим образом:

```
number floor(number)
```

В этом примере я при помощи floor преобразую длительность дней планет в целые числа:

```
<?xml version="1.0"?>
<xsl:stylesheet version-"1.0"
xmlns xsl-"http://www.w3.org/1999/XSL/Transform">

    <xslrtemplate match-"/PLANETS">
        <HIML>


        </HIML>
    </xsl:template»

    <xslrtemplate match-"PLANET">
        <IR>
            <TD><xslrvalue-of select="NAME"/></TD>
            <TD><xsl rapply-templates select-"MASS"/x/TD>
            <TD><xsl: apply-templ ates sel ect-"RADIUS" /x-/TD>
            <T0><xslrapply-templates select-"DAY"/></TO>
        </TR>
    </xslrtemplate»

    <xslrtemplate match-"MASS">
        <xslrvalue-of select-"."/>
```

```
        <xsl:text> </xslrtext>
        <xsl value-of select-"@UNITS"/»
    </xsl:template»

    <xslrtemplate match="RADIUS">
        <xsl value-of select"".">
        <xsl:text> </xsl:text>
        <xsl:value-of select-"@UNITS"/>
    </xslrtemplate»

    <xslrtemplate match^'DAY"»
        <xsl:value-of select-"floor(.)"/>
        <xslrtext» </xslrtext»
        <xslrvalue-of select»"@UNITS"/>
    </xslrtemplate»


</xsl:stylesheet»
```

А вот результирующий документ:

```
<HIML>
    <HEAD>
        <TITLE>
            The Planets Table
        </TITLE>
    </HEAD>

    <BODY>
1 J     <H1>
            The Planets Table
        </H1>

        <TABLE BORDER-"2"»
            <TR>
                <TD»Name</TD>
                <TD»Mass</TD>
                <TD»Radius</TD»
                <TD»Day</TD>
            </TR>

            <TR»
                <TD»Mercury</TD>
                <TD».0553 (Earth - 1)</TD>
                <TD»1516 miles</TD>
                <TD»58 days</TD>
            </TR>

            <TR»
                <TD»Venus</TD»
                <TD».B15 (Earth - 1)</TD»
                <TD»3716 miles</TD>
                <TD»116 days</TD»
            </TR>

            <TR»
                <TD»Earth</T0»
                <TD»1 (Earth - 1)</TD>
```

```
                <TD>2107 miles</TD>
                <TD>1 days</TD>
            </TR>

        </TABLE>
    </BODY>
</HTML>
```

number()

Функция number приводит свой аргумент к числу. Функция применяется следующим образом:

```
number number(object?)
```

В этом примере я передаю number строку:

```
number("456.7")
```

Это выражение возвращает число 456.7. Если вы попытаетесь преобразовать что-то, что number не сможет перевести в число, функция вернет значение XPath NaN (Not a Number, не число). NaN — корректное значение, которое можно проверять в выражениях Xpath.

round()

Функция round округляет переданное ей значение. Функция применяется следующим образом:

```
number round(number)
```

Например, round(3.1415926535) возвращает 3, round(4.5) возвращает 5 и round(-1.5) возвращает -1.

sum()

Функция sum суммирует численные значения набора узлов и возвращает результат. Функция применяется следующим образом:

```
number sum(node-set)
```

Взгляните на следующий пример из главы 4, который находит средние массы планет в planets.xml:

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xs1-"http://www.w3.org/1999/XSL/Transform">
<xsl:output method-"xml"/>

<xslrtemplate match-"PLANETS">
    <HIML>
        <BODY>
            The average planetary mass 1s:
            <xslrvalue-Of select-"sum(Child::PLANET/child::MASS) d1v
            count(chiId::PLANET/chi1d::MASS)"/>
```

```
        </BODY>
    </HTML>
</xsl:template>

</xsl:stylesheet>
```

Логические функции XPath

XPath также поддерживает следующий набор логических функций:

- boolean(). Приводит аргумент к логическому значению;
- false(). Возвращает false (ложь);
- lang(). Проверяет, совпадает ли язык, установленный в атрибуте xml:lang, с языком, переданным в функцию;
- not(). Инвертирует значение true\false аргумента;
- true(). Возвращает true (истина).

boolean()

Функция boolean преобразует аргумент в логическое значение. Функция применяется следующим образом:

```
boolean boolean(object)
```

Ниже показано, как функция работает с различными типами XPath:

- number. Если число равно нулю, результатом будет ложь; иначе результат — истина. NaN всегда возвращает ложь;
- string. Если строка не пуста, результат — истина, иначе — ложь;
- boolean. Значение не изменяется;
- node set. Пустой набор узлов дает ложь, не пустой — истину;
- Фрагмент результирующего дерева в XSLT 1.0 всегда дает истину.

false()

Функция false возвращает ложь. Она применяется следующим образом:

```
boolean falseO
```

В XPath не определены логические константы, поэтому для того чтобы присвоить переменной значение false, нужно прибегнуть к функции false. (С переменными вы познакомитесь в главе 9.)

Iang()

Функция lang проверяет, совпадает ли язык контекстного узла (определенный в атрибуте xmlrlang) с переданным ей языком. Функция применяется следующим образом:

```
boolean lang(string)
```

Функции нужно передать строку, определяющую язык в спецификации XML, как, например, «еп» для английского, <*de» для немецкого и «jp» для японского.

В следующем примере я проверяю, написан ли исходный документ на английском. Сначала я устанавливаю атрибут xmlilang в planets.xml в значение, соответствующее английскому языку:

```
<?xml version-"1.0"?>
<?xml-stylesheet type-"text/xml" href»"planets.xsl"?>
<PLANETS xml:lang-"eh">

    <PLANET>
        <NAME>Mercury</NAME>
        <MASS UNITS-"(Earth = 1)">.0553</MASS>
        <DAY UNITS-"days">58.65</DAY>
        <RADIUS UNITS="miles">1516</RADIUS>
        <DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
        <DISTANCE UNITS-'milHon miles">43.4</DISTANCE><!-B перигелии-->
    </PLANET>
        .
        .
        .
```

Теперь я удостоверяюсь, что исходный документ на самом деле написан на английском, перед тем как применять к нему шаблоны:

```
<?xuL vers1on-"l.0"?>
<xsl:stylesheet version-"1.0"
xmlns:xs1-"http://www.w3.org/1999/XSL/Transform">

    <xslrtemplate match-"/PLANETS">
        <xsl:if test-"Tang('en')">
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <BODY>
                <HL>
                    The Planets Table
                </HL>
                <TABLE BORDER-"2">
                    <TR>
                        <TD>Name</TD>
                        <TD>Mass</TD>
                        <TD>Radius</TD>
                        <TD>Day</TD>
                    </TR>
                    <xsl:apply-templates/>
                </TABLE>
            </BODY>
        </HTML>
        </xsl:if>
    </xsl tempiate>

    <xslrtemplate match-"PLANET">
```

```
        <TR>
          <TD><xs 1: value-of select-"NAME"/»</TD>
          <TD><xsl•apply-templates select-"MASS"/></TD>
          <TD><xsl:apply-templates select-"RADIUS'7></TD>
          <TD><xsl: apply-templ ates select-"DAY"/»</TD>
        </TR>
    </xsl template»
        .
        .
        .
</xsl:stylesheet»
```

not()

Функция not инвертирует логическое значение аргумента. При передаче аргумента с логическим значением true она возвращает false; при передаче аргумента со значением false функция возвращает true. Функция применяется следующим образом:

```
boolean not(boolean)
```

В следующем примере, который вы уже видели в главе 4, я хочу выбрать только те элементы, у которых есть оба атрибута — COLOR и POPULATED. Для этого я применяю предикат выбора «[@C0L0R and (©POPULATED]». Для того, чтобы убрать другие элементы (чтобы правила по умолчанию не включали их текст в результирующий документ), я использую предикат «[not(@C0L0R) or not(@POPULATED)]»:

```
<?xml version-"1.0"?>
<xsl:stylesheet version-"l.O"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

    <xsl:template match-'VPLANETS"»
        <HTML>
        .
        .
        .
        </HTML>
    </xsl:template»

    <xslrtemplate match-'PLANET[@COLOR and ©POPULATED]'»
        <TR>
          <TD»<xsl:value-of select""NAME "/»</TD>
          <TD»<xsl:apply-tempiates select-"MASS"/»</TD»
          <TD»<xslrapply-templates select-"RADIUS"/»</TD>
          <TD»<xslrapply-templates select-"DAY"/»</TD>
        </TR»
    </xslrtemplate»

    <xslrtemplate match®'PLANET[not(©COLOR) or not(©POPULATED)]'»
    </xslrtemplate»

    <xsl:tempiate match-"MASS">
        <xsl:value-of select-"."/>
        <xsl:text> </xslrtext»
```

```
        <xsl:value-of select-"@UNITS"/>
    </xslrtemplate»

    <xsl template match-"RADIUS">
        <xsl:value-of select-"."/>
        <xsl:text> </xsl text>
        <xsl:value-of seiect-"@UNITS"/>
    </xsl:tempiate>

    <xslrtemplate match-"DAY">
        <xslrvalue-of select-"."/>
        <xsl:text> </xsl:text>
        <xsl rvalue-of select-"(aUNITS7>
    </xsl:tempiate>

</xsl stylesheet»
```

Вот результат:

```
<нт>
    <HEAD>
        <TITLE>
            Colorful. Populated Planets
        </TITLE>
    </HEAD>

    <BODY>
        <HI>
            Colorful. Populated Planets
        </HI>

        <TABLE BORDER-"2">
            <TR>
                <TD>Name</TD>
                <TD>Mass</T0>
                <TD>Radi us</TD>
                <TD>Day</TD>
            </TR>

            <TR>
                <TD>Earth</TD>
                <TD>1 (Earth - 1)</TD>
                <TD>2107 mi 1es</TD>
                <TD>1 days</TD
            </TR>
        </TABLE>
    </BODY>
</HTML>
```

true()

Функция true возвращает значение true. Она применяется следующим образом:

```
boolean trueO
```

В XPath не определены логические константы, поэтому для того чтобы присвоить переменной значение true, нужно прибегнуть к функции true. (С переменными вы познакомитесь в главе 9.)

Элемент <xsl:decimal-format>: создание числовых форматов

Перед тем как закончить обсуждение функций XSLT и XPath, я хочу рассмотреть специальный элемент XSLT, <xsl:decimal-format>, единственное предназначение которого заключается в работе только с одной функцией format-number. В частности, при помощи этого элемента определяются символы и идентификаторы, которые будет использовать format-number. У этого элемента несколько атрибутов:

- name (необязательный). Имя десятичного формата. Принимает значение типа QName. Если формат не задан, используется десятичный формат по умолчанию;
- decimal-separator (необязательный). Определяет символ, разделяющий целую и дробную части числа. По умолчанию «.». Устанавливается в символ;
- grouping-separator (необязательный). Определяет символ, разделяющий группы разрядов. По умолчанию •«.«•.Устанавливается в символ;
- infinity (необязательный). Задает строку, обозначающую положительную бесконечность. По умолчанию «Infinity» (бесконечность). Устанавливается в строку;
- minus-sign (необязательный). Определяет символ, представляющий знак минуса. По умолчанию «-^.Устанавливается в символ;
- NaN (необязательный). Определяет строку, представляющую значение «Not a Number» (не число). По умолчанию «NaN». Устанавливается в строку;
- percent (необязательный). Определяет символ, представляющий знак процента. По умолчанию «%».Устанавливается в символ;
- per-mille (необязательный). Определяет символ, представляющий знак промилле, то есть тысячных долей. По умолчанию **«%o»**.Устанавливается в символ;
- zero-digit (необязательный). Определяет символ для использования в форматирующих строках для указания лидирующего или завершающего нуля. По умолчанию «О». Устанавливается в символ;
- digit (необязательный). Определяет символ для использования в форматирующих строках для указания цифры. По умолчанию «О». Устанавливается в символ;
- pattern-separator (необязательный). Определяет символ, разделяющий образцы для положительных и отрицательных чисел. По умолчанию «;».Устанавливается в символ.

Этот элемент — верхнего уровня, и он всегда пуст. При помощи этого элемента можно устанавливать форматирующие символы для функции format-number. Элемент xsUdecimal-format, у которого не установлен атрибут name, становится десятичным форматом по умолчанию. Наличие более одного элемента xsl:decimal- format по умолчанию или нескольких элементов xsl:decimal-format с одним и тем же

именем является ошибкой. В следующем примере я форматирую числа в planets, xml при помощи европейского формата числа, а именно — я использую запятую вместо десятичной точки для разделения целой и дробной частей, и точку вместо запятой для разделения групп разрядов. Для этого мне нужно только задать новое форматирование при помощи <xsl:decimal-format> и затем применить его' в format-number:

```
<?xml version-"l 0"?>
<xslr stylesheet version-"1.0"
xmlnsrxsl-"httpr//www.w3.org/1999/XSL/Transform">

    <xsl:decimal-format decimal-separator-"." grouping-separator-"."/>
    <xslrtemplate match='7PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    The Formatted Planets Table
                </TITLE>
            </HEAD>
            <BODY>
                <H1>
                    The Formatted Planets Table
                </HI>
                <TABLE BORDER-"2">
                    <TR>
                        <TD>Name</TD>
                        <TD>Mass</TD>
                        <TD>Radius</TD>
                        <TD>Day</TD>
                    </TR>
                    <xsl:apply-templates/>
                </TABLE>
            </BODY>
        </HTML>
    </xsl:tempiate>

    <xslrtemplate match-"PLANET">
        <TR>
            <TD><xsl rvalue-of sel ect-" NAME7></TD>
            <TDxxsl rapply-templates select-"MASS"/x/TD>
            <TD><xslrapply-templates select-"RADIUS"/></TD>
            <TD><xsl.apply-templates select-"DAY'7></TD>
        </TR>
    </xslrtempiate>

    <xslrtemplate match-"MASS">
        <xsl:value-of select«"format-number(., '#.##')"/>
        <xslrtext> </xslrtext>
        <xslrvalue-of select-"@UNITS"/>
    </xslrtemplate»

    <xslrtemplate match-"RADIUS">
        <xsl:value-of sel ect-" format-number (., '#.##')"/>
        <xslrtext> </xslrtext>
        <xslrvalue-of select-"@UNITS"/>
```

```
    </xsl:template»

    <xslrtemplate match-"DAY">
        <xsl:value-of select-"format-number(.. '##.#')"/>
        <xsl text> </xsl:text>
        <xsl:value-of select-"@UNITS"/>
    </xsl;template»

</xsl:stylesheet»
```

Вот результирующий документ:

```
<HTML>
    <HEAD»                    r
        <TITLE>
            The Formatted Planets Table
        </TITLE»
    </HEAD>

    <BODY»
        <H1>
            The Formatted Planets Table
        </H1»

        <TABLE BORDER-"2"»
            <TR>
                <TD»Name</TD>
                <TD»Mass</TD>
                <TD»Radius</TD»
                <TD»Day</TD>
            </TR»

            <TR>
                <TD»Mercury</TD>
                <TD>0.055 (Earth - 1)</TD>
                <TD>1 516 miles</TD>
                <TD>58.65 days</TD>
            </TR>

            <TR>
                <TD»Venus</TD>
                <TD>0.815 (Earth - 1)</TD>
                <TD>3.716 miles</TD>
                <TD>116.75 days</TD»
            </TR>

            <TR>
                <TD»Earth</TD>
                <TD>1 (Earth - 1)</TD>
                <TD>2107 miles</TD>
                <TD>1 days</TD>
            </TR»

        </TABLE>
    </BODY»
</HTML»
```

Этот документ показан на рис, 8.3.

Вот и все — теперь вы можете задавать параметры форматирования для функции format-number.

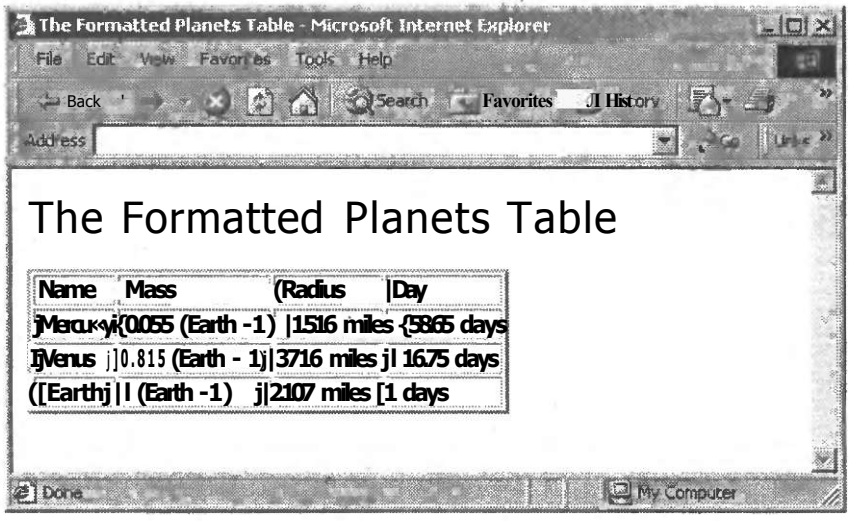


Рис. 8.3. Задание десятичных форматов

Новые функции в XSLT 2.0 и XPath 2.0

В XSLT 2.0 и XPath 2.0 планируется включить весьма много функций. В частности, для XSLT 2.0 внесены в план следующие положения:

- должны быть включены функции, которые могут работать с QName, для того, чтобы можно было применить объявления пространств имен из области видимости;
- должны быть добавлены функции форматирования даты для поддержки схемы XML;
- должна быть включена функция для преобразования относительных идентификаторов URI в абсолютные;
- должны быть включены функции, упрощающие группировку. В XSLT 2.0 должно быть возможно группировать узлы на основе их строковых значений.

В XPath 2.0 в отношении функций планируются следующие изменения:

- должен быть расширен набор агрегирующих функций — например, включены функции минимума и максимума;
- для наборов узлов должны быть реализованы функции пересечения и разности;

- должна быть включена поддержка дополнительных строковых функций, таких как функции для замены строк, заполнения символами и преобразований регистров в строках;
- должны быть включены агрегирующие функции для работы с коллекциями такими как наборы узлов;
- должны быть определены функции для работы со стандартными типами схемы XML.

На этом мы заканчиваем рассмотрение функций XSLT и XPath, применяемых при преобразованиях. В главе 9 мы изучим переменные, параметры и другие специальные темы.

Именованные шаблоны, параметры и переменные

9

В этой главе рассматривается ряд специальных возможностей XSLT, а именно шаблоны, параметры таблиц стилей и переменные. Все эти темы связаны друг с другом: параметры используются в именованных шаблонах, а переменные и параметры — это практически одно и то же, они различаются только способом создания.

Если задать шаблону имя, его можно вызывать по этому имени при помощи элемента <xsl:call-template>. Шаблон применяется при вызове, поэтому вместо того чтобы полагаться на обработку таблицы стилей, принятую в процессоре XSLT по умолчанию, вы можете явно задать, когда и какой шаблон нужно применять. Например, несколько шаблонов могут выбирать один и тот же набор узлов, и вам нужно выбрать из них один или несколько шаблонов, которые будут применены. Именованные шаблоны похожи на режимы, но дают вам больше возможностей управления.

При вызове шаблона можно настроить его работу при помощи *параметров.* Например, вам может понадобиться, чтобы текст в создаваемых шаблоном текстовых узлах был на определенном языке — таком как английский, немецкий или французский — и вы можете создать новый параметр с именем language (язык). При вызове именованного шаблона, заданного для обработки этого параметра, вы можете установить язык в «еп», *de» или «fr» и затем вызвать шаблон при помощи элемента <xsl:with-param>. В самом именованном шаблоне используемый им параметр language объявляется при помощи элемента <xsl:param>. После объявления параметра к его значению можно свободно обращаться как Slanguage и использовать его в выражениях XPath. В этой главе мы рассмотрим многочисленные примеры работы с параметрами,

-л Переменные во многом похожи на параметры, с тем лишь отличием, что они по-другому создаются. Параметры, как правило, используются в именованных шаблонах, в то время как переменные Применяются более широко, в выражениях

XPath любого вида. Как и в языках программирования, в переменных XSLT можно хранить значения и обращаться к ним позже. Но есть одна важная особенность: за исключением особых обстоятельств, вы *не можете* изменять значение, хранимое в переменной. (В связи с этим некоторые авторы, пишущие об XSLT, считают неправильным называть их переменными.)

Переменные удобны для хранения значений, создание которых занимает длительное время, но в таблице стилей они часто используются. Вместо Того,чтобы каждый раз заново создавать эти значения, сохраните их в переменной и ссылайтесь на ее значение. Как и в случае с параметрами, для получения значения переменной добавьте префикс «$». Например, для переменной с именем sandwich получить ее значение можно при помощи Ssandvvich. Как и в параметрах, в переменных можно хранить данные всех четырех типов данных XPath. Переменные также имеет смысл применять для хранения значений, которые позже в шаблоне будут изменены. Например, «.» обычно ссылается на контекстный узел шаблона, но внутри элемента <xsl:for-each> «.» ссылается на текущий обрабатываемый в элементе узел, а не на контекстный узел всего шаблона. Для того чтобы обратиться к контекстному узлу, перед входом в цикл <xsl:for-each> сохраните его в переменной contextnode и затем в теле цикла используйте это значение как Scontextnode.

Кроме четырех типов данных XPath мы также будем использовать тип данных, поддерживаемый в XSLT 1.0, но не в XSLT 1.1 — фрагменты результирующего дерева, которые создаются элементами <xsl:variable> или <xsl:with-param>. Фрагменты результирующего дерева могут быть удобны в определенных случаях, как вы увидите далее в этой главе.

Наконец, в этой главе мы также рассмотрим элемент <xsl:key>. Впервые он нам встретился в главе 4, теперь мы изучим его более подробно.

Для введения вполне достаточно; давайте перейдем к работе, и начнем мы с переменных.

Элемент <xsl:variable>: создание переменных

Для создания переменных в XSLT служит элемент <xsl:variable>, обладающий следующими атрибутами:

- паше (обязательный). Имя переменной, устанавливается в QName;
- select (необязательный). Выражение XPath, задающее значение переменной. Если опустить этот атрибут, значение переменной будет определяться содержимым <xsl:variable>.

• i ЭТот элемент может либо быть элементом верхнего уровня, либо применяться внутри тела Шаблона. Элемент может сам содержать тело шаблона, но в таком случае нельзя использовать атрибут select.

Для создания переменной присвойте ее имя атрибуту паше элемента <xsl: variable», а значение переменной <- атрибуту select, как в следующем примере, в котором я создаю переменную numberjbooks и сохраняю л ней значение 255:

```
<xsl:variable name-"number_books" select-"255"/>
```

Получить значение переменной можно, добавив к ее имени префикс $:

```
<xsl:variable name-"numberJbooks" select="255"/>
<xsl:text>There are </xsl:text>
<xsl:value-of select-"Snumber_books"/>
<xsl:text> books in my library </xsl:text>
```

Заметьте, что если вы присваиваете переменной литерал — как, например, присваивание значения «turkey» (индейка) переменной sandwich (бутерброд), — литерал необходимо заключить в кавычки, причем они должны отличаться от кавычек, в которые заключены значения атрибутов:

```
<xsl:variable name-"sandwich" select-"'turkey'"/>
```

В XSLT 1.0 нет необходимости в атрибуте select — данные можно заключить внутри самого элемента <xsl:variable>:

```
<xsl:variable name="sandwich">turkey</xsl variable»
```

Формально, однако, при пропуске атрибута select в элементах <xsl:variable> или <xsl:with-param> и задании этим элементам содержимого вы создаете фрагмент результирующего дерева, который больше не допускается в XSLT 1.1.

Стоит отметить, что имя переменной может включать префикс, как, например, star:PLANET, который должен соответствовать активному пространству имен. Сравнения осуществляются не сравнением префиксов, а проверкой фактического URI префикса — поэтому star:PLANET может быть тем же самым, что и nebula:PLANET, если пространства имен star и nebula соответствуют одному и тому же URI.

Область видимости переменной

Элемент <xsl:variable> можно использовать как элемент верхнего уровня или внутри тела шаблона для создания переменных. Переменные, созданные в элементах <xsl:variable> высокого уровня, обладают *глобальной областью видимости,* созданные в телах шаблона — *локальной.* Область видимости переменной определяет, в какой части таблицы стилей вы можете ее использовать.

Областью видимости глобальной переменной является вся таблица стилей, подразумевая и импортированные или включенные таблицы стилей. Это означает, что переменная доступна в любом месте таблицы стилей, если только она не будет перекрыта локальной переменной с тем же именем. Можно даже обращаться к глобальной переменной до ее объявления. Однако нельзя создавать циклические ссылки (то есть если вы объявили **a** через Ь, нельзя объявлять **b** через **a).**

Область видимости локальной переменной ограничена следующими за ней братьями или потомками последующих братьев. В частности это значит, что если вы объявили переменную внутри таких элементов, как <xsl:choose>, <xsl:if> или <xsl:for-each>, она не будет доступна вне этих Элементов.

Как правило, вы не можете изменять значение переменной, но вы можете перекрыть ее локальной переменной. То есть локальные переменные перекрывают глобальные в пределах области видимости локальных переменных. Пусть, например, я объявил переменную с именем movie (кинокартина):

```
<xsl:variable name-"movie" select-"'Mr. Blandings Builds His Dream House'">

<!-- здесь $mov1e - 'Mr. Blandings Builds His Dream House'●●>
```

Это элемент верхнего уровня, поэтому movie — глобальная переменная. Даже внутри шаблонов movie будет сохранять свое начальное значение, если не будет локальной переменной с таким же именем:

```
<xsl:variable name-"movie" select-"'Mr. Blandings Builds His Dream House'">

<!-- здесь $movie - 'Mr  Blandings Builds His Dream House'-->

<xsl:tempi ate match-"entertainment">

    <!-- здесь Smovie - 'Mr. Blandings Builds His Dream House'-->
```

Однако если вы объявите локальную переменную movie, в шаблоне эта версия перекроет глобальную переменную:

```
<xsl:variable name-"movie" select-"'Mr. Blandings Builds His Dream House'"»

<@060>!— здесь Smovie - 'Mr. Blandings Builds His Dream House'-->

<xsl template match-"entertainment">

    <!-- здесь Smovie -  Mr  Blandings Builds His Dream House"—<@062>

    <xsl:var1able name-Hmovie" select-"'Goldf1nger'H>

    <!-- здесь Smovie- 'Goldf1nger'-->
```

В этом случае мы перекрыли глобальную переменную при помощи глобальной. Заметьте, однако, что нельзя снова объявить одну и ту же переменную в одном шаблоне с целью попытаться изменить ее значение:

```
<xsl ^variable name-"movie" select-"'Mr. Blandings Builds His Dream House'"»

<!-- здесь Smovie -  Mr  Blandings Builds His Dream House"-->

<xsl template match-"entertainment"»

    <!-- здесь Smovie - 'Mr. Blandings Builds His Dream House'-->»
```

```
<xsl:variable name-"movie" se1ect-"'Goldfinger'">
<@060>!— здесь Smovie - 'Goldfinger'-->
<xsl:variable naee-"movie" select-'"Withnail and I' ><!,•• Запрещено ч->
```

За пределами шаблона локальная переменная невидима, и movie содержит глобальное значение:

```
<xs1variable name-"movie" select«'"Mr. Blandings Builds His Dream House\">
<!-- здесь Smovie •= 'Mr  Bindings Builds His Dream House'-->
<xsl:tempi ate match-"entertainment">
    <!-- здесь Smovie - 'Mr. Blandings Builds His Dream House'-->
    <xsl:vanable name-"mov1e" select*"'Goldfinger'">
    <!-- здесь $movie - 'Goldfinger'-->
    <xsl:variable name-"movie" select-'"Withnail and Г"><!-- Запрещено -->
</xsl tempiate>
<!-- здесь Smovie - 'Mr. Blandings Builds His Dream House'-->
```

Глобальные переменные тоже нельзя объявлять повторно:

```
<xslvariable name-'movie" select-"'Mr. Blandings Builds His Dream House'">
<@060>!— здесь Smovie - 'Mr. Blandings Builds His Dream House'—>
<xsltemplate match="entertainment">
    <!-- здесь Smovie = 'Mr. Blandings Builds His Dream House'—<@062>
    <xsl:variable name-"movie" select«'"Goldfinger'">
    <!-- здесь Smovie - 'Goldfinger'-->
    <xsl:variable name-"movie" select="'Withnail and I'"><!-- Запрещено -->
</xsl:tempiate>
<!-- здесь Smovie - 'Mr. Blandings Builds His Dream House'-->
<xsl:variable name-"movie" select»"'Goldfinger'"><!-- Запрещено,-->
```

Несмотря на все эти ограничения, вы *можете,* менять значение переменной на каждом шаге цикла <xsl:for-each>, как мы увидим в следующем разделе.

Работа с переменными

Давайте рассмотрим примеры применения переменных. В следующем примере (листинг 9.1) я присваиваю переменной copyright сообщение об авторских правах и затем с ее помощью добавляю атрибут copyright во все элементы planets.xml.

Листинг 9.1. Применение переменной

```
<?xml version^"!.0"?>
«xsl:stylesheet version-"l.1"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"»
    <xsl output method-"xml"/»

    <xsl:variable name-"copyright" select-"'(c)2002 Starpowder Inc.'"/>
    <xsl:tempi ate match="*">
        <xsl:copy>
            <xslattribute name-"copyright">
                <xsl:value-of select-"$copyr1ght"/>
            </xsl:attribute»
            <xsl:apply-templates/»
        </xsl copy»
    </xsl: ter.pl ate»
</xsl:stylesheet»
```

Вот результирующий документ, дополненный атрибутами copyright

```
<?xml version-"1.0" encoding-"utf-8"?>
<PLANETS copyright-"(c)2002 Starpowder Inc ">

    <PLANET copyright-"(c)2002 Starpowder Inc."»
        <NAME copyright-"(c)2002 Starpowder Inc."»Mercury«/NAME>
        <MASS copyright-"(c)2002 Starpowder Inc."» 0553</MASS>
        <DAY copyright-"(c)2002 Starpowder Ihc."»58.65</DAY>
        <RADIUS copyright-"(c)2002 Starpowder Inc"»1516</RADIUS»
        «DENSITY copyright-"(c)2002 Starpowder Inc."».983</DENSITY>
        «DISTANCE copyright-"(c)2002 Starpowder Inc."»43.4«/DISTANCE»
    «/PLANET»

    «PLANET copyright-"(c)2002 Starpowder Inc."»
        «NAME copyright-"(c)2002 Starpowder Inc."»Venus«/NAME»
        «MASS copyright-"(c)2002 Starpowder Inc."».815«/MASS>
        <DAY copyright-"(c)2002 Starpowder Inc H»116.75«/DAY>
        «RADIUS copyright-"(c)2002 Starpowder Inc."»3716«/RADIUS>
        «DENSITY copyright-"(c)2002 Starpowder Inc.B>.943</DENSITY>
        «DISTANCE copyright-"(c)2002 Starpowder Inc."»66.8«/DISTANCE>
    <!PLANET»
```

Переменные зачастую удобны для хранения значений, зависимых от контекста, и мы сейчас рассмотрим еще один пример, о котором я упоминал в начале главы. В этом случае я преобразую planet$.xmt в новый документ, в котором для каждой планеты будет один элемент. Каждый из этих Новых элементов будет

содержать два элемента <SIBLINGPLANET>, содержащих планеты-братья текущей планеты — например, братьями Земли будут Венера и Меркурий:

```
<?xml version-"1.0й encoding-"utf-8"?>
<Mercury>
    <SIBLINGPLANET>
        Venus
    </SIBLINGPLANET>
    <SIBLINGPLANET>
        Earth
    </SIBLINGPLANET>
</Mercury>

<Venus>
    <SIBLINGPLANET>
        Mercury
    </SIBLINGPLANET>
    <SIBLINGPLANET>
        Earth
    </SIBLINGPLANET>
</Venus>

<Earth>
    <SIBLINGPLANET>
        Mercury
    </SIBLINGPLANET>
    <SIBLINGPLANET>
        Venus
    </SIBLINGPLANET>
</Earth>
```

Для примера я поочередно выбираю каждый элемент <PLANET> и прохожу в цикле <xsl:for-each> по всем планетам, создавая элементы <SIBLINGPLANET> для всех планет, не являющихся контекстным узлом. Однако откуда мне известно внутри элемента <xsl:for-each>, какая из планет является контекстным узлом, выбранным шаблоном? Внутри элемента <xsl:for-each> о ссылается на текущий узел, с которым работает <xsl:for-each>, но не на контекстный узел шаблона. Проблему можно решить, если сохранить контекстный узел в переменной, которую я назвал contextnode:

```
<?xml vers1on-"l.0"?>
<xsl:stylesheet version-"l.l"
xmlns:xs1http://www.w3.org/1999/XSL/Transform">
<xsl:output method-"xml"/>

    <xsl:template match-"PLANETS">

        <xsl:for-each select-"PLANET">
            <xsl:element name-"{NAME}">

                <xsl:variable name-"contextnode" select-"."/»
```

Теперь для проверки в цикле <xsl:for-each> того, что текущий элемент не является контекстным узлом, я могу обратиться к контекстному узлу шаблона как $contextnode (листинг 9.2).

Листинг 9.2. Хранение в переменной информации, зависимой от контекста

```
<?xml version-"1.0"?»
<xsl:stylesheet version-"l.1"
xmlns:xsl-"http://www.w3.org/1Qgg/XSL/Transform'>
<xsl:output method""xml"/>

    <xsltemplate match-"PLANETS">

        <xsl:for-each select-"PLANET">
            <xsl:element name-"{NAME}">

                <xsl:variable name-"contextnode" select-"."/>

                <xsl:for-each select-"//PLANET"»
                    <xsl:if test-". !- $contextnode">
                        <xsl:element name-"SIBLINGPLANET">
                            <xsl:value-of select-'NAME"/»
                        </xsl:element»
                    </xsl:if»
                </xsl:for-each»

            </xsl:element»
        </xsl:for-each»

    </xsl:tempiate»

</xsl:stylesheet»
```

Теперь наша проблема решена.

Если у элемента <xsl:variable> есть тело, он создает переменную, чье значение является фрагментом результирующего дерева. В следующем примере при помощи фрагмента результирующего дерева я задаю значение по умолчанию для атрибута COLOR (цвет), если значение для него уже не задано. Значение по умолчанию я устанавливаю в «blue» (голубой):

```
<xsl:variable name="COLOR">
    <xsl:choose»
        <xsl :when test="@COLOR"»
            <xsl:value-of select-"@COLOR'7>
        </xsl:when>
        <xsl otherwise>blue</xsl:otherwise>
    </xsl:choose»
</xsl variable»
```

Строковое значение фрагмента результирующего дерева (то есть либо значение атрибута COLOR, либо значение по умолчанию, «blue») присваивается переменной COLOR. Теперь в выражениях XPath можно обращаться к значению этой переменной, $COLOR, а не к значению атрибута @COLOR, гарантированно получая при этом значение цвета, даже если у соответствующего элемента отсутствует атрибут COLOR.

Вот еще один пример фрагмента результирующего дерева. В этом случае 3 Сохраняю элемент буквального результата в переменной START_HTML: TA?/.

```
^?xml version-"! 0"?>
<xsl:stylesheet version-"l.l"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">
    <xsl:output method-"html"/>

<xsl:variable name="START_HTHL">
    <HEAD>
        <TITLE>
            Hy page
        <?/TITLE>
    </HEAD>
</xsl:variable*
```

Теперь я могу использовать этот элемент буквального результата где угодно:

```
<?xml version-"1.0"?>
<xsl stylesheet version-"l.l"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">
    <xsl:output method-"html"/>

<xslvariable name-"START_HTML">
    <HEAD>
        <TITLE>
            My page
        </TITLE>
    </HEAD>
</xsl;variable>

    <xsltemplate match-"PLANETS">

<HTML>
    <xsl:copy-of select-"$START_HTML"/>
<BODY>
<H>Welcome to my page</HI>
</B0DY>
</HTML>
    </xsl:tempiate>

</xsl:stylesheet>
```

И вот результат:

```
<HTML>
   <HEAD>
      <TITLE>
         My page
      </TITLE>
   </HEAD>
   <BODY>
      <H>Welcome to my page</HI>
   </BODY>
</HTML>
```

Однако поскольку теперь фрагменты результирующего дерева не допускаются в XSLT 1.1, этот пример работать не будет. Как же тогда сохранить весь элемент буквального результата одновременно с возможностью простого вызова? Вы можете создать именованный шаблон.

Элемент <xsl:call-template>: применение именованных шаблонов

У элемента <xsl:template> есть атрибут name, задающий имя шаблона. Предположим, у меня есть элемент буквального результата, состоящий из двух элементов
 и двух элементов <HR> HTML, при помощи которого я создаю в документах HTML вертикальный разделитель:

```
<BR/>
<HR/>
<BR/>
<HR/>
```

Тогда я могу создать шаблон с именем «separator» (разделитель), использующий этот элемент буквального результата:

```
<xsl:template name="separator">
    <BR/>
    <HR/>
    <BR/>
    <HR/>
</xsl:template>
```

Это именованный шаблон — для его создания нужно только присвоить имя атрибуту name элемента <xsl:template>.

Заметьте, что этот шаблон не установлен для выбора чего-то конкретного. Для активизации шаблона необходимо вызвать его явно. Для этого служит элемент <xsl:call-template>, обладающий только одним атрибутом::

- name (обязательный). Имя вызываемого шаблона, устанавливается в QName.

Следующий пример демонстрирует применение нашего шаблона «separator», для чего мне нужно было только вызвать его в соответствующих местах:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.1"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <xsl:template match="/PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <BODY>
                <H1>
                    The Planets Table
```

```
                </H1>
                <xsl:cal1-tempiate name-"separator"/»
                <TABLE BORDER-"2">
                    <TR>
                        <TD>Name</TD>
                        <TD>Mass</TD>
                        <TD>Radius</TD>
                        <TD>Day</TD>
                    </TR>
                    <xsl:apply-templates/>
                </TABLE>
                <xslrcall-template name-"separator"/»
            </BODY>
        </HTML>
    </xsl:template»

    <xslrtemplate match-"PLANET"»
        <TR>
            <TD><xsl:value-of select-"NAME"/></TD>
            <TD><xsl:apply-templates select-"MASS"/></TD>
            <TD»<xsl:apply-templates select="RADIUS"/»</TD>
            <TD»<xsl:apply-templates select-"DAY"/></TD>
        </TR>
    </xsltemplate»

    <xsl:tempi ate match-'MASS"»
        <xsl:value-of select-"."/>
        <xsl:text> </xslrtext»
        <xslrvalue-of select-"@UNITS"/>
    </xslrtemplate»

    <xslrtemplate match-"RADIUS">
        <xslrvalue-of select-"."/>
        <xslrtext» </xslrtext»
        <xsl:value-of select-"@UNITS"/>
    </xslrtemplate»

    <xslrtemplate match-"DAY">
        <xslrvalue-of select-"."/»
        <xslrtext» </xslrtext»
        <xslrvalue-of select-'^UNITS"/»
    </xslrtemplate»

    <xsl:tempi ate name-"separator">
        <BR/»
        <HR/>
        <BR/>
        <HR/»
    </xsl:tempiate»

</xslrstylesheet»
```

Вот результат. Обратите внимание: элементы
 и <HR> были вставлены так, как требовалось:

```
<HTN?>
    <HEAD>
        <TITLE>
```

```
            The Planets Table
        </TITLE>
    </HEAD>

    <BODY>
        <H1>
            The Planets Table
        </H1>
        <BR>
        <HR>
        <BR>
        <HR>
        <TABLE BORDER="2">
            <TR>
                <TD>Name</TD>
                <TD>Mass</TD>
                <TD>Radius</TD>
                <TD>Day</TD>
            </TR>

            <TR>
                <TD>Mercury</TD>
                <TD>.0553 (Earth - 1)</TD>
                <TD>1516 miles</TD>
                <TD>58.65 days</TD>
            </TR>

            <TR>
                <TD>Venus</TD>
                <TD>.815 (Earth - 1)</TD>
                <TD>3716 miles</TD>
                <TD>116.75 days</TD>
            </ TR>

            <TR>
                <TD>Earth</TD>
                <TD>1 (Earth - 1)</TD>
                <TD>2107 miles</TD>
                <TD>1 days</TD>
            </TR>

        </TABLE>
        <BR>
        <HR>
        <BR>
        <HR>
    </BODY>
</HTML>
```

Результирующий документ показан на **рис.** 9.1.

Таким образом, **вы** можете обращаться к элементу буквального результата по имени, легко помещая его в результирующий документ. Но это слишком статично — элемент буквального результата всегда один и тот же. Впрочем, как можно заметить, вызов именованного шаблона во многом похож на вызов подпрограммы в языке программирования. Так же, как вы передаете данные в подпрограмму, вы можете передать данные в именованные шаблоны при помощи *параметров.*

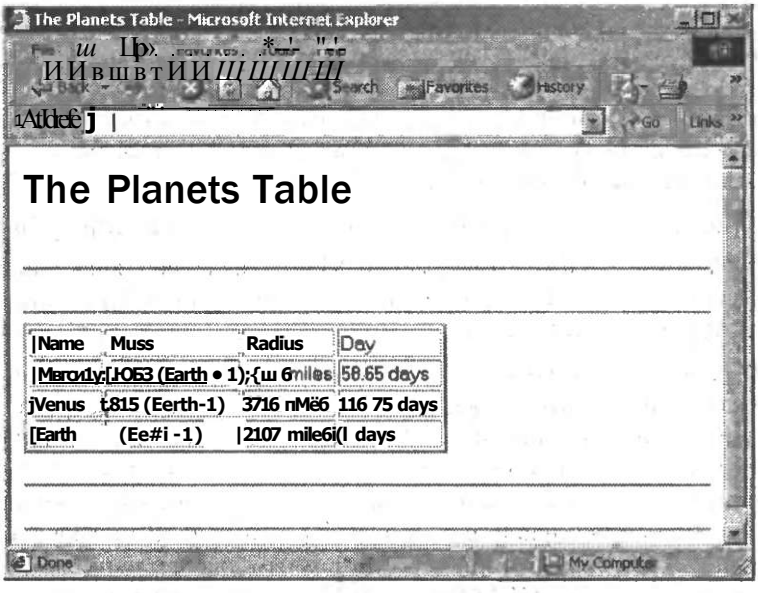


Рис. 9.1. Вызов именованного шаблона

Элементы <xsl:param> и <xsl:with-param>: создание параметров

Параметры во многом похожи на переменные — за тем исключением, что они обычно используются вместе с именованными шаблонами. Параметры позволяют вам передать в шаблон какие-то значения. Параметры создаются элементом <xsl:param>, обладающим двумя атрибутами:

- name (обязательный). Имя переменной, устанавливается в QName;
- select (необязательный). Значение параметра по умолчанию. Устанавливается в выражение XPath.

Аналогично <xsl:variable>, этот элемент может либо быть элементом высокого уровня, либо применяться внутри тела шаблона. Параметры, созданные элементами <xsl:param>, являются глобальными, а созданные внутри шаблонов — локальными. При создании параметра внутри тела шаблона, элемент <xsl:param> должен быть расположен перед какими-либо другими дочерними элементами. Элемент может также содержать необязательное тело шаблона, создающее фрагмент результирующего дерева — в XSLT 1,1 это уже не разрешено. Если элемент содержит тело, атрибут select использовать нельзя.

После объявления параметра при помощи <xsl:param> к его значению можно обращаться в шаблоне таким же способом, как и к значению переменной^ — добавляя к имени префикс «$».

Когда вы вызываете именованный шаблон при помощи <xsl:call-template> или применяете шаблоны с использованием <xsl:apply-template>, вы можете задать зна-

чение используемых в шаблоне параметров с помощью элемента <xsl:with-param>. Если самому параметру при объявлении было присвоено значение атрибутом select элемента <xsl:param>, это значение выступает в качестве значения параметра по умолчанию. Значение по умолчанию будет перекрыто, если вы зададите новое значение параметра при помощи элемента <xsl:with-param>.

У элемента <xsl:with-param> два атрибута:

- name (обязательный). Имя переменной; устанавливается в QName;
- select (необязательный). Выражение XPath, задающее значение параметра. Если опустить этот атрибут, значение переменной будет определяться содержимым <xsl:with-param>.

Элемент может также содержать необязательное тело шаблона, создающее фрагмент результирующего дерева, но в XSLT 1.1 это уже не разрешено.

В следующем примере (листинг 9,3) я создаю именованный шаблон с именем «COLORS» (цвета), добавляющий цвет в данные планет в результирующем HTML-документе. Этот именованный шаблон использует единственный параметр, COLOR, который устанавливается в требуемый цвет. В примере я устанавливаю параметр COLOR в разные цвета для разных планет, используя <xsl:with-param> и вызывая шаблон «COLORS». Вот как это реализуется на практике.

Листинг 9.3. Применение параметров таблиц стилей

```
<xsl:stylesheet versi on-"11"
xmlns: xs1-"http: //Wwww3. org/1999/ KSJL ДрарБ form" >

    <xsl template match»"/PLANETS">
        <HTML>
        .
        .
        .
        «/HTML»
    </xsl:template»

    <xslrtemplate match-"PLANET"»
        <xsl:if test-"NAME-'Mercury""»
            <xsl:call-tempi ate name='COLORS'»
                <xsl:with-param name-"COLOR" select-"'RED'"/»
            </xsl:cal1 -tempiate»
        </xsl:1f>
        <xsl:1f test-"NAME*'Venus'"»
            <xsl:cal1 -tempiate name-"COLORS">
                <xsl: with - param name-TOLOR" sel ect-"' GREEN"/»
            </xsl:cal1 -tempiate»
        </xsl:if>
        <xsl:1f test-'NAME-'Earth'"»
            <xsl:call-template name-"COLORS"»
                <xsl:with-param name*"COLOR" select»""BLUE"'/»
            </xsl:call -tempiate>      i.                    i          ? г
        </xsl:i f»                                                    l:i    ) i
    </xslrtemplate»                                                  ! i fi"H* •   КМ-Н-
```

Данная таблица вызывает шаблон «COLORS» с разными значениями параметра COLOR. Я могу воспользоваться этими цветами при форматировании данных планет. Заметьте, что я объявил параметр COLOR при помощи <xsl:param> в самом начале шаблона «COLORS»:

```
<xsl:stylesheet version-"1.1"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform">

    <xsl template match-"/PLANETS"»
        <HIML>
            .
            .
            .
        </HIML>
    </xsl:template»

    <xslrtemplate match-"PLANET"»
        <xsl:i f test-"NAME-' Mercury"'>
            <xsl:call-tempiate name-"COLQRS">
                <xsl with-param name-"COLOR" select-"'RED'"/»
            </xsl:cal1-tempiate»
        </xsl:1f»
        <xsl:1f test-"NAME"'Venus"'>
            <xsl:ca11 -tempi ate name='COLORS'>
                <xSl:with-param name»'COLOR" select-"'GREEN'"/>
            </xsl -.call-tempi ate»
        </xsl:1f>
        <xsl:if test-"NAME-'Earth"'>
            <xsl:cal1-tempiate name-"COLORS">
                <xsl with-param name="COLOR" select-'"BLUE"7>
            </xsl:call-tempi ate»
        </xsl:1f>
    </xsl template»

    <xsl:tempi ate name-"COLORS"»
       <xsl:param name-"COLOR"/»
       <TR»
          <TD><FONT COLOR-"{$COLOR}"»<xsl rvalue-of select-"NAME"/»</FONT»</TD»
          <TD><FONT COLOR-"{$COLOR} "xxsl: apply-tempi ates select-"MASS"/x/FONT»</TD»
          <TD><FONT COLOR-"{$COLOR}"»<xsl: apply-tempi ates select-"RADIUS"/»</FONT»</TD»
          <TD><FONT COLOR-"{$COLOR}"»<xsl: apply-tempi ates select-"DAY"/»</FONT»</TD>
       </TR»
   </xsl:tempiate»

    <xsl template match-"MASS">
        <xsl:value-of select-"."/»
        <xsl:text> </xsl:text>
        <xsl:value-of seiect="@UNITS"/>
    </xslrtemplate»
            .
            .
            .
    <xslrtemplate match-"DAY">
        <xslrvalue-of Select-"."/»
        <xslrtext» </xslrtext»
        <xsl rvalue-of select-"@UNITS"/>
```

```
    </xsl:template»                    v

</xsl:stylesheet»
```

И вот результат:

```
<HTML>
  <HEAD>4

    <TITLE>
      The Colorful Planets Table

    </TITLE>
  </HEAD>
  <BODY>
    <H1>

      The Colorful Planets Table

    </H1>
    <TABLE BORDER-"2"»
      <TR»
        <TD»Name</TD>
        <TD»Mass</TD>
        <TD»Radius</TD>
        <TD»Day</TD>
      <AR»
      <TR>
        <TD><FONT COLOR-"RED"»Mercury</FONT»</TD>
        <TD><FONT COLOR-"RED"».0553 (Earth - 1)</FONT»</TD»
        <TD><FONT COLOR-"RED"»1516 mi 1es</FONT></TD>
        <TD><FONT COLOR-"RED"»58.65 days</FONT»</TD>
      </TR>

      <TR>
        <TD><FONT COLOR-'GREEN'»Venus</FONT»</TD>
        <TD><FONT COLOR-'GREEN'» 815 (Earth = 1)</FONT»</TD>
        <TD><FONT COLOR='GREEN'»3716 miles</FONT></TD»
        <TD><FONT COLOR-"GREEN">116.75 days</FONTx/TD>
      </TR»

      <TR>
        <TDxFONT COLOR-"BLUE"»Earth</FONT»</TD»
        <TD><FONT COLOR-"BLUE"»1 (Earth - 1)</FONT»</TD>
        <TD><FONt COLOR-"8LUE"»2107 miles</FONT»</TD>
        <TD><FONT COLOR-"BLUE"»1 days</FONT»</TD>
      </TR>

    </TABLE>
  </BODY>
</HTML>
```

Этот результирующий документ показан на рис. 9.2 (пусть даже в черно-белом исполнении).

В следующем примере я использую параметры для локализации используемого в шаблоне языка. Я создаю новый шаблон localize с параметром language. Если language установлен в «еп», означая английский, результирующий документ

будет озаглавлен «Planets»; если language установлен в «de», немецкий, результирующий документ будет озаглавлен «Planeten», а если в cfi>, французский, результирующий документ будет озаглавлен «Planetes».

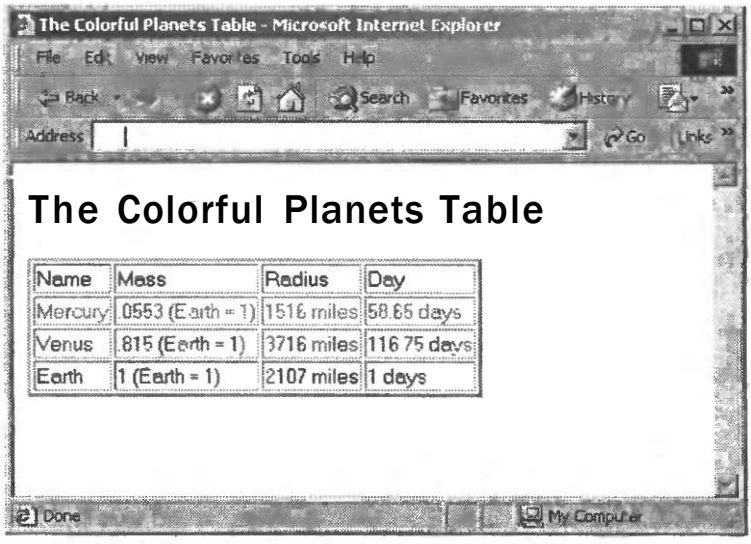


Рис. 9.2. Вызов именованного шаблона с параметрами

Здесь я вызываю шаблон localize, установив language в «6>:

```
<?xml version-"1.0"?>
<xsl:stylesheet version«"I.I"
xmlns:xs1«"http://www.w3.org/1999/XSL/Trans form">

    <xslrtemplate match-"/PLANETS">
        <HTML>
            <HEAD>
                <TITLE>
                    <xsl:call-template name-"localize">
                        <xsl:with-param name-"language" select-"'fr'"/>
                    </xsl:call -tempiate>
                </TITLE>
            </HEAD>
            <BODY>
                <H1>
                    <xsl:call-template name-"localize">
                        <xsl:with-param name-"language" select-"'fr'"/>
                    </xsl:call -tempiate>
```

А вот как выглядит параметризованный шаблон «CONORS».. Заметьте, что я объявляю параметр COLOR в шаблоне при помощи элемента <xsl:param> (и заметьте, что в HTML 4.01 символьный элемент для «е» в «Planetes», который я и использую — это è):

```
<?xml version-"l 0"?>
«эд!,: stylesheet vers ion-" 1.1"
xmlns:xsVhttp. //www w3.org/1999/XSL/Transform">

    <xslrtemplate match-'7PLANETS">
        <HIML>
            ^HEAD>
                <TITLE>
                    <xsl:call-template name-"localize">
                        <xsl:with-param name-"language" select="'fr"7>
                    </xsl:cal1-tempiate>
                </TITLE>
            </HEAD>
            <BQDY>
                <Hl>
                    <xsl:call-template name-"localize">
                        <xsl:w1th-param name-"language" select-"'fr"7>
                    </xsl:call-template»
                </Hl>
                <TABLE BORDER"'2'>
                    <IR>
                        <TD>Name</TD>
                        <TD>Mass</TD>
                        <TD>Radius</TD>
                        <TD>Day</TD>
                    </TR>
                    <xsl:apply-templates/>
                </TABLE>
            <BODY>
        </HIML>
    </xsl:template»

    <xslrtemplate name-"localize">
        <xsl:param name-"language"/>
        <xs1:1f test»"$1anguage-'en"'>
            <xsl:text>Planets</xsl:text>
        </xsl:if>
        <xsl:i f test-"$language-'de"'>
            <xsl:text>Planten</xsl:text>
        </xsl:if>
        <xsl:i f test-"$1anguage-'fr"'>
            <xsl:text>Plan&#232:tes</xsl:text>
        </xsl:if>
    </xsl:template»
```

Вот результирующий локализованный документ*.

```
<HIML>
   <HEAD>

      <TITLE>P1an&egrave;tes</TITLE>
   </HEAD>          .]- •                    i
   <BODY>
      <H1>P1an&egrave;tes</Hl>
      <TABLE BORDER-"2">
```

```
<TR>
   <TD>Name</TD>
   <TD>Mass</TD>
   <TD>Radius</TD>
   <TD>Day</TD>
</TR>
   .
   .
   .
```

Этот документ показан на рис. 9.3.

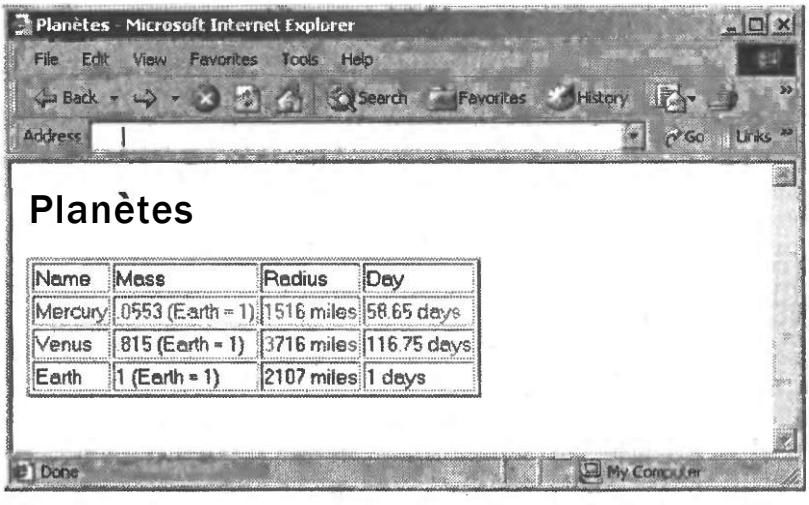


Рис. 9.3. Вызов именованного шаблона с параметрами для установки языков

Вызов шаблона во многом похож на вызов функции, что обсуждалось в главе 8, и способность передавать данные через параметры усиливает сходство. Но в отсутствие оператора присваивания, как в языках программирования, кажется, что нельзя присвоить переменной значение, возвращаемое именованным шаблоном, впрочем, поразмыслив, это можно сделать. С шаблонами можно и еще в одном случае поступить так же, как с функциями, — осуществить рекурсивный вызов.

Рекурсивный вызов шаблонов

Эта тема предназначена, главным образом, для программистов, поскольку здесь я буду пользоваться XSLT как языком программирования. В частности, я реализую вызов именованным шаблоном самого себя, то есть *рекурсивный* вызов. Классический пример рекурсии — вычисление факториала: например, факториал 6, что записывается как б!, равен 6 * 5 * 4 * 3 * 2 * 1 , или 720.

При реализации рекурсии в настоящем языке программирования создается функция — например, factorial, которая вызывается со значением б: factorial(6). Факториал 6 вычисляется как б * factorial(5), поэтому функции нужно лишь умножить на 6 результат вызова самой себя со значением 5, то есть factorial(5)..

Далее, factorial(5) — это 5 * factorial(4), поэтому функция снова вызывает сама себя, чтобы вычислить значение factorial(4). Этот процесс продолжается до вычисления factorial(l), а мы знаем, что 1! — это просто 1, поэтому factorial(l) возвращает 1. С этого момента управление последовательно возвращается на все предыдущие этапы, в результате чего будет вычислено выражение 1 * 2 * 3 * 4 * 5 * 6, или 720, что составляет б!.

Кажется, что в таком языке стилей, как XSLT, реализовать подобное невозможно. Тем не менее, это можно сделать, по Крайней мере, в XSLT 1.0. Основная идея состоит в том, что значение, возвращаемое шаблоном, можно сохранять в переменной, если шаблон вызывается внутри элемента <xsl:variable>, в котором объявляется эта переменная. Пусть, например, у нас есть именованный шаблон factorial, и мы хотим вычислить б!. Тогда шаблону можно передать значение 6 при помощи элемента <xsl:with-param> и присвоить строковое значение результата переменной result, которое я затем показываю:

```
<?xml version-"1.0И?>
<xsl:stylesheet version-"1.0"
xmlns:xs1-"http://www.w3.org/1999/XSL/Transform">

    <xslrtemplate match-"/">
        <xs1:vari able name-"result">
            <xsl: cal 1 - tempi ate name-" f actori al ">
                <xslrwith-param name-"value" select-"6"/>
            </xsl:cal1 -tempiate>
        </xsl:variable»
        61 - <xsl:value-of select-"$result"/>
    </xslrtemplate»
```

Следующий пример демонстрирует, как можно реализовать шаблон factorial, чтобы для вычисления факториала он вызывал сам себя. На языке программирования я мог бы написать рекурсивный вызов как п! «= n * factorial(п -1), но у нас нет оператора присваивания; поэтому, когда я вычисляю factorial п -1), я сохраняю это значение в новой переменной temp и на каждом шаге возвращаю значение n * $temp:

```
<?xml version-"1.0"?>
<xsl:stylesheet version"" 1.0"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"»

    <xslrtemplate match-"/"»
        <xsl:variable name-"result">
            <xsl call-template name-"factorial*»
                <xsl rwith-param name-"value" select^"6"/>
            </xsl call-tempiate»
        </xsl:variable»
        6! - <xsl rvalue-of select-"$result'V>
    </xsl-.template» '

    <xsl:tempiate name-"factorial">
        <xsl;param name="va1ue"/»
```

```
        <xsl:choose>
            <xsl:when test-" $ val ue-Г">
                <xsl:value-of  select-"l"/>
            </xsl:when>
            <xsl:otherwise>
                <xsl:variable name="temp'">
                    <xsl:call-template name-"factorial">
                        <xsl:with-param name="value" select-"lvalue - l"/>
                    </xsl:cal1 -tempiate>
                </xsl:variable>
            <xsT:value-of select-"$temp * $value"/>
        </xsl:otherwise»
    </xsl:choose>
</xsl:template»

</xsl:stylesheet»
```

Вот результирующий документ:

```
<?xml version-"!..0" encoding-"utf-8"?>
6! - 720
```

Как видите, это можно сделать, по крайней мере, в XSLT 1.0, в котором разрешены использованные здесь фрагменты результирующего дерева.

Шаблон: значение по умолчанию

Как я говорил ранее, в случае задания параметру значения при объявлении, оно может быть перекрыто, если вы зададите другое значение в элементе <xsl:with-param>. Но если другого значения не указывать, исходное значение выступит в роли значения по умолчанию.

Следующий пример видоизменяет рассмотренный ранее пример «COLORS». Шаблон имеет параметр COLOR, но я могу вызвать шаблон, не устанавливая COLOR в какое-либо определенное значение:

```
<xsl -.template match*"PLANET'\»
        <xsl:if  test^"NAME-'Mercury"'>
            <xsl:call-tempiate name-'COLORS'\»
                <xsl:w1th-param name'COLOR"  select-'"RED"'/»
            </xsl:call-template»
        </xsl:if>
        <xsl:i f test="NAME-[1] Venus"'>
            <xsl:cal1-tempiate name-'COLORS'\»
                <xsl:with-param name-'C0L0R"  select-'"GREEN"'/»
            </xsl  cal1-template»
        </xsl:if»
        <xsl:if  test-"NAME-'Earth[1]'"»
            <xsl:cal1 -tempiate name»'C0L0RS'>
            </xsl:cal1 -tempiate»
        </xsl:if»
    </xsl:template»
```

В этом случае параметр COLOR принимает значение по умолчанию «ЬИие» (голубой), заданное в элементе <xsl:param> в шаблоне «COLORS»:

```
    <xsl:template match-'PLANET'\»
        <xsl:if  test-"NAME-'Mercury"'"»
```

```
        <xsl:call-tempiate name='C0L0RS'>
            <xslrwith-param name='C0L0R" select-'"RED"'/>
        </xsl:call-template»
    </xsl:if>
    <xsl if test-»"NAME-'Venus' ">
        <xslreal1-tempiate name='C0L0RS'>
            <xslrwith-param name='C0L0R" select-'"GREEN"7>
        </xsl-call-template»
    </xsl:i f>
    <xsl:1f test-"NAME-'Earth"'>
        <xsl:call-tempiate name='C0L0RS'»
        </xsl:call-tempi ate»
    </xsl:if>
</xslrtemplate»

<xslrtemplate name-'C0L0RS'>
    <xsl:param name='C0L0R" select-'"blue"'/»
    <TR>
        <TD><FONT COLOR-"{$C0L0R}"»<xslrvalue-of select="NAME"/»</FONTx/TD>
        <TD><FONT COLOR-"{$C0L0R}"><xslrapply-templates select="MASS"/></FONT></TD>
        <TD><FONT COLOR-"{$C0L0R}"><xslrapply-templates select-"RADIUS"/></FONT»</TD>
        <TD><FONT COLOR-"{$C0L0R}"»<xslrapply-templates select-"DAY"/x/FONT»</TD>
    </TR>
</xslrtemplate»
```

Как задавать значения шаблона в командной строке

Кроме возможностей, предоставляемых элементами **<xslparam>** и **<xslwith-param>,** значёние параметров таблицы стилей во многих процессорах XSLT можно также задавать в командной строке. Способ зависит от конкретного процессора.

Следующий пример показывает, как можно присвоить параметру **paraml** значение **valuel** в командной строке, используя процессор XSLT от Oracle в Windows. Заметьте, что при установке значений параметра в командной строке его все равно нужно объявлять в таблице стилей при помощи **<xslparam>:**

```
Cr\»java oracle.xml.parser.v2.oraxsl -p paraml-'valuel' planets.xml planets.xsl output.xml
```

Вот как то же самое можно сделать при помощи Saxon:

```
Cr\»saxon source.xml stylesheet.xsl paraml-valuel » output.xml
```

при помощи Xalan:

```
Cr\»java org.apache.xalan.xsIt.Process -IN planets.xml
    •XSL planets.xsl -OUT output.xml -PARAM parmal valuel
```

и XT:

```
Cr\XSL»java -Dcom jelark xsl.sax parser-org.apache.xerces.parsers.SAXParser com.
jelark.xsl.sax.Driver planets.xml planets.xsl output.xml paraml-valuel
```

Элемент <xsl:key>: выбор по ключу

Элемент <xsl:key> позволяет создать ключ и выбрать узлы по этому ключу. Он обладает следующими атрибутами:

- пате (обязательный). Имя ключа, устанавливается в QName;
- match (обязательный). Устанавливается в образец выбора для выбора узлов, к которым применяется ключ;
- use (обязательный). Устанавливается в выражение, определяющее значение ключа.

При помощи ключей можно легко распознавать элементы; конкретные ключи можно выбрать при помощи образца «кеу()». В главе 4 мы познакомились с ключами, имеющими одно значение. В следующем примере я использую ключи для выбора планет, у которых атрибут COLOR был установлен в «BLUE» — в данном случае это Земля:

```
<?xml version-"1.0"?>
<?xml-stylesheet type-"text/xml" href-"planets.xsl"?>
<PLANETS>
        .
        .
        .
    «PLANET C0L0R-"BLUE">
        <NAME>Earth</NAME>
        <MASS UNITS-"(Earth - 1)">1</MASS>
        <DAY UNITS-"days">1</DAY>
        <RADIUS UNITS-"mi1es">2107</RADIUS>
        «DENSITY UNITS-"(Earth - 1)">1</DENSITY>
        «DISTANCE UNITS-"million miles">128.4</DISTANCE><!--В перигелии--»
    </PLANET>

</PLANETS>
```

Я создал ключ COLOR, выбирающий элементы <PLANET> путем проверки их атрибута COLOR. Ключ выглядит следующим образом:

```
<?xml version»"1.0"?>
<xsl:stylesheet version-"1.1"
xmlns: xsl «'"http://www.w3.org/1999/XSL/Transform">

    <xsl:key name-"C0L0R" match-"PLANET" use-"PC0L0R"/>
        .
        .
        .
```

После этого я применил образец «кеу()» для того, чтобы выбрать элементы <PLANET>, у которых атрибут COLOR установлен в «BLUE»:

```
<?xml version="1.0"?>
<xsl:stylesheet version-"1.l"
xmlns:xsl"http://www.w3 org/1999/XSL/Transform">

    <xsl:key name~"C0L0R" match-"PLANET" use-"(PC0L0R"/>

    <xsl:template match-"/PLANETS">
        <HTML>
            <HEAD>
                <TLTLE>
```

```
                The Planets Table
            </TITLE>
        </HEAD>
        <BODY>
            <H1>
                The Planets Table
            </H1>
            <TABLE BORDER="2">
                <TR>
                    <TD>Name</TD>
                    <TD>Mass</TD>
                    <TD>Radius</TD>
                    <TD>Day</TD>
                </TR>
                <xsl:apply-tempiates select-"key("COLOR",  "BLUE")"/>
            </TABLE>
        </BODY>
    </HTM>
</xsl:template»
    .
    .
    .
```

И вот результат — как видите, образцу удовлетворяет только Земля:

```
<HTML>
    <HEAD>
        <TITLE>
            The Planets Table
        </TITLE>
    </HEAD>

    <BODY>
        <H1>
            The Planets Table
        </H1>

        <TABLE BORDER-"2">
            <TR>
                <TD>Name</TD>
                <TD>Mass</TD>
                <TD>Radlus</TD>
                <TD>Day</TD>
            </TR>

            <TR>
                <TD>Earth</TD>
                <TD>1  (Earth  -  1)</TD>
                <TD>2107 mi 1es</TD>
                <TD>1 days</TD>
            </TR>
        </TABLE>
    </BODY>
</HTML>
```

Но в данном случае существует больше вариантов. Например, ключу могут соответствовать несколько узлов, а это означает, что использующий ключ обра-

зец вернет набор узлов. Пусть, например, все планеты будут иметь одно и то же значение атрибута COLOR, «unknown» (неизвестен):

```
<?xml version-"1 0"?>
<?xml-stylesheet type="text/xml" href="planets.xsl"?>
<PLANETS>

    «PLANET COLOR'UNKNOWN'>
        <NAME>Mecury</NAME>
        <MASS UNITS-"(Earth - 1)">.0553</MASS>
        <DAY UNITS-"days">58.65</DAY>
        <RADIUS UNITS-"miles">1516</RADIUS>
        «DENSITY UNITS-"(Earth - 1)"> 983</DENSITY>
        «DISTANCE UNITS="million miles">43.4</DISTANCE><!–B перигелии–>
    </PLANET>

    «PLANET COLOR'UNKNOWN'>
        «NAME>Venus«NAME>
        «MASS UNITS-"(Earth = 1)">.815«/MASS>
        «DAY UNITS-"days">116.75</DAY>
        «RADIUS UNITS-"mi 1es">3716</RADIUS>
        «DENSITY UNITS-"(Earth - 1)">.943«/DENSITY>
        «DISTANCE UNITS="million miles">66 8</DISTANCE><!–B перигелии–>
    </PLANET>

    «PLANET COLOR'UNKNOWN'>
        <NAME>Eath«NAME>
        «MASS UNITS*"(Earth = 1)">1</MASS>
        «DAY UNITS="days">k«/DAY>
        «RADIUS UNITS-"miles">2107«/RADIUS>
        «DENSITY UNITS-"(Earth - 1)">1«/DENSITY>
        «DISTANCE UNITS-"million miles">128 4«/DISTANCE><«!–B перигелии–>
    </PLANET>

</PLANETS>
```

Если создать теперь ключ COLOR следующим образом: <xsl:key name-"C0L0R" match="PLANET" use»"@C0L0R"/> и осуществить поиск по этому ключу по образцу «key('C0L0R', 'BLUE')», то будут выбраны все три планеты:

```
<HTML>
    <HEAD>
        <TITLE>
            The Planets Table
        </TITLE>
    </HEAD>
    <BODY>
        <H1>
            The Planets Table
        </H1>
        «TABLE BORDER-'2'>
            <TR>
                <TD>Name</TD>
                <TD>Mass«/TD>
                <TD>Radius«/TD>
                <TD>Day«/TD>
            «/TR>
```

```
            <TR>
                <TD>Mercury</TD>
                <TD>.0553 (Earth - 1)</TD>
                <TD>1516 miles</TD>
                <TD>58.65 days</TD>
            </TR>

            <TR>
                <TD>Venus</TD>
                <TD>.815 (Earth - 1)</TD>
                <TD>3716 mnes</TD>
                <TD>116.75 days</TD>
            </TR>

            <TR>
                <TD>Earth</TD>
                <TD>1 (Earth - 1)</TD>
                <TD>2107 miles</TD>
                <TD>1 days</TD>
            </TR>
        </TABLE>

    </BODY>
</HTML>
```

Кроме случая, когда ключ выбирает несколько узлов, узел также может предоставить несколько значений для одного ключа. Предположим, вы установили ключ для работы с элементом <NAME> внутри элемента <PLANET>. Но что, если каждый элемент <PLANET> содержит более одного элемента <NAME>, как в этом примере:

```
<?xml version="1.0"?>
<?xml-stylesheet type-"text/xml" href-"pianets.xsl"?>
<PLANETS>

    <PLANET>
        <NAME>Mercury</NAME>
        <NAHE>Sister Planet</NAME>
        <MASS UNITS-"(Earth - 1)">. 0553<MASS>
        <DAY UNITS-"days">58 65</DAY>
        <RADIUS UNITS-"miles">1516</RADIUS>
        <DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
        <DISTANCE UNITS-"million miles">43.4</DISTANCEx>--B перигелии-->
    </PLANET>

    <PLANET>
        <NAME>Venus</NAME>
        <NAME>Sister Planet</NAME>
        <MASS UNITS-"(Earth - 1)">.815</MASS>
        <DAY UNITS-"days">116.75</DAY>
        <RADIUS UNITS-"miles">3716</RADIUS>
        <DENSITY UNITS-"(Earth - 1)"> 943</DENSITY>
        <DISTANCE UNITS-'million miles">66 8</DISTANCE><!--B перигелии-->
    </PLANET>

    <PLANET>
        <NAME>Eath</NAME>
        <MASS UNITS-"(Earth - 1)">1</MASS>
        <DAY UNITS-"days">1</DAY>
```

```
        «RADIUS UNITS-"miles">2107</RADIUS>
        «DENSITY UNITS»"(Earth - 1)">1</DENSITY>
        «DISTANCE UNITS="million miles">128.4«/DISTAtJCE><J–B перигелии-->
    </PLANET>

</PLANETS>
```

В этом случае каждый элемент <NAME> проверяется на совпадение с ключом. Предположим, например, что я хочу выбрать элементы <NAME> с текстом «Sister Planet». Вот таблица стилей:

```
<?xml version-"1.0"?>
«xsl:stylesheet versi on-"1.1"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <xsl:key name-'NAME' match-'PLANET" use-'NAME'/>

    <xsl:template match-"/PLANETS">
        «HTML>
            <HEAD>
                <TITLE>
                    The Planets Table
                </TITLE>
            </HEAD>
            <BODY>
                <H1>
                    The Planets Table
                «/H1>
                «TABLE BORDER-'2'>
                    «TR>
                        <TD>Name</TD>
                        <TD>Mass«/TD>
                        <TD>Radius«/TD>
                        <TD>Day«/TD>
                    </TR>
                    <xslrapply-templates select-"key('NAME",  "Sister Pldnet')"/>
                «/TABLE>
            </BODY>
        </HTML>
    «/xsl:tempiate>

    «xsl:tempi ate match-'PLANET'>
        <TR>
            «TD><«xsl:value-of  select-"NAME7><«/TD>
            <TD><«xsl: apply-templ ates select-HMASS"/x/TD>
            <TD><«xsl:apply-templates  select-"RADIUS"/></TD>
            <TD><«xsl:apply-templates  select-"DAY"/><«/TD>
        </TR>
    «/xsl:tempiate>

    «xsl:tempi ate match-'MASS'>
        «xsl  valuo-o.1 se'ectr" ."/>    дь.     VA/ttbiT
        <xsl:text>  </*5i:t6xt>  '
        «xsl rvalue  of  select-'^UNITS"^
    «/xsl  tenplate>
```

./K

```
    <xslrtemplate match-"DAY">
        <xsl:value of select-"."/>
        <xsl:text> </xsl:text>
        <xslrvalue-of select="@UNITS'7>
    </xsl:tempiate>

c/xs[1]:stylesheet»
```

И вот результирующий документ:

```
<нм>
    <HAD>
        <ITTLE>
            The Planets Table
        </ITTLE>
    <HEAD>

    <BDY>
        <Hl>
            The Planets Table
        </Hl>

        «TABLE BORDER«'2'>
            <IR>
                <ID>Name</ID>
                <ID>Mass</ID>
                <ID>Radius</ID>
                <ID>Day</ID>
            </IR>

            <IR>
                <ID>Mercury</ID>
                <ID>.0553 (Earth - 1)</ID>
                <ID>1516 miles</ID>
                <ID>58.65 days</ID>
            </IR>

            <IR>
                <ID>Venus</ID>
                <ID>.815 (Earth - 1)</ID>
                <ID>3716 miles</ID>
                <ID>116.75 days =/ID>
            </IR>
        </TABLE>

    <BODY>
<HIML>
```

Элемент <xsl:document>: создание нескольких результирующих документов

Весьма часто во время преобразования требуется создать несколько результирующих документов. Например, вам может понадобиться создать отчет о прохождении преобразования или разделить входной документ на несколько результирующих документов (например, разбить рассказ на главы). Или же вам может

быть нужно создать набор результирующих документов, которые будут использоваться (как при создании кадрового (frameset) документа HTML) совместно с двумя документами, отображаемыми во фреймах.

Создание нескольких результирующих документов настолько распространенная задача, что практически все процессоры XSLT позволяют вам это сделать, даже в XSLT 1.0, где отсутствуют необходимые для этого средства. Процессоры XSLT добавляют для этого новые элементы расширения. Например, Xalan предоставляет элемент **<write>,** позволяющий написать новый результирующий документ. Для работы с этим элементом нужно создать новый префикс пространства имен — здесь я использую «xalan» — которое Xalan использует для элемента, «com.lotus.xsl.extensions.Redirect», и указать, что этот новый префикс является префиксом элемента расширения:

```
<?xml version-"1.0"?>
<xsl stylesheet version»"1.1"
    xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"
    xmlns:xalan-"com.lotus.xsl.extensi ons.Redi rect"
    extensi on-element-prefi xes-"xalan">
```

Теперь при помощи атрибута file элемента <xalan:write> можно записать новый файл:

```
«'xml version="l 0"?>
<xsl:stylesheet version="l.l"
    xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"
    xmlns xalan="com.lotus,xsl.extensions.Redirect"
    extension-element-prefixes="xalan">

<xalan:write f1le-"newdoc.txt">
    <xsl:text>Here's some text.</xsl:text>
</xalan:write>
```

В процессоре Saxon применяйте элемент <output>. Для этого элемента я использую префикс «saxon», который соответствует URI «http://icl.com/saxon»:

```
<?xml version-"1.0"?>
<xsl:sty1esheet versi on-"1.1"
    xmlns:xsl-"http://www.w3.org/J999/XSL/Transform"
    xmlns:saxon-"http://i cl.com/saxon"
    extensi on-element•prefi xes-"saxon">

<saxon:output f1le-"newdoc.txt">
    <xsl:text>Here'S some text.</xsl:text»                    *
</saxon:output*
```

То же можно сделать и в XT; в этом случае используйте пространство имен «http://www.jclark.com/xts- с элементом <document> и атрибут href для задания имени нового файла:

```
, .. <?xml vepsion-"1.0"?>
ч , <xsl,,?tyle^heet version-"1.1м
        xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"
        xmlns:xt-"http://www.Jclark.com/xt"
• f tin-: extension-el ement-pref ixes-«"xt">
  1   M      .

    <xt:document href-"newdoc.txt">
        <xsl:text>Here's some text.</xsl:text>
    </xt:document»
```

Все йто создало весьма запутанную ситуацию, поскольку каждый разрабатывал свои собственные решения. По этой причине в XSLT 1.1 для поддержки нескольких результирующих документов был представлен новый элемент, <xsl: document>, со следующими атрибутами:

- href (обязательный). Показывает, где будет помещен новый документ. Устанавливается в абсолютный или относительный URI, без идентификатора фрагмента;
- method (необязательный). Определяет метод вывода, используемый для создания результирующего документа. Устанавливается в «xml», «html», «text» или QName, не являющееся NCName;
- version (необязательный). Определяет версию выходного документа. Устанавливается в NMTOKEN;
- encoding (необязательный). Устанавливает кодировку выходного документа. Устанавливается в строку;
- omit-xml-dedaration (необязательный). Устанавливается в «yes» или «по», чтобы опустить объявление XML или не отпускать;
- cdata-section-elements (необязательный). Задает имена тех элементов, чье содержимое нужно вывести как разделы CDATA. Устанавливается в список QName, разделенных символами-разделителями;
- doctype-public (необязательный). Задает открытый идентификатор, который будет использован в объявлении <!DOCTYPE> вывода. Устанавливается в строковое значение;
- doctype-system (необязательный). Задает системный идентификатор, который будет использован в объявлении <!DOCTYPE> вывода. Устанавливается в строковое значение;
- indent (необязательный). Определяет, будет ли выходной документ выровнен для отображения структуры вложенности. Устанавливается в «yes» или «по»;
- media-type (необязательный). Устанавливает тип MIME вывода. Устанавливается в строковое значение;
- standalone (необязательный). Определяет, будет ли отдельное объявление ; включено в выходные данные, и если да, устанавливает его значение. Устанавливается в «yes» или «по». а ь"н

Этот элемент содержит тело шаблона.

Следующий пример основан на упрощенной таблице стилей. **В** этом случае **я** создал в HTML-документе две рамки (frame), а также два HTML-документа для отображения в этих рамках **(framel.html** и **frame2.html).** Первую рамку и ее документ я создал следующим образом:

```
<HIM>
    <HEAD>
        <ITTLE>
            Two Frames
        </ITTLE>
    <HEAD>

    «RAMESET cols-"50*. 50Г>
        «RAME sre-"framel.html"/*

        «xsl:document href-"framel.html">
            <HIML>
                <HEAD>
                    «ITTLE>
                        Frame 1
                    </ITTLE>
                <HEAD>

                «BODY>
                    <Hl>This is frame l.«/Hl>
                «BODY>
            «HIML>
        </xsl:document>
        .
        .
        .
```

Затем я могу создать вторую рамку и ее документ:

```
<HIML>
    <HEAD>
        <ITTLE>
            Two Frames
        «/ITTLE>
    <HEAD>

    «RAMESET cols-"503!. 50Г>

        «RAME src-"framel.html"/>

        «xsl:document href-"framel.html">
            <HIML>
                <HEAD>
                    <ITTLE>
                        Frame 1
                    «/ITTLE>
                <HEAD>

                <BODY>
                    <Hl>This is frame l.«/Hl>
                <HIML>
```

```
    </xsl: document*

  «FRAME sre-" frair э2. html" />

   <xs1:document href-"frame2.html">
        <IIII>
            <HEAD>
                «ITTLE*
                    Frame 2
                </TTTLE>
            <HEAD>

            <BODY>
                «Hl*Th1s is frame 2.</Hl>
            <BODY>
        «HTML*
    </xsl:document*

«/FRAMESET*

</HTML>
```

Заметьте, однако, что этот пример работоспособен только в XSLT 1.1.

На момент написания книги в одном процессоре XSLT элемент **<xsl:document>** был реализован: это Saxon версии 6.2.1 и старше, в котором свой элемент **<saxon: output>** был изменен на **<xsl:document>.** Но пока это единственный известный мне процессор XSLT, поддерживающий этот элемент.

Элемент <xsl:namespace-alias>: генерация таблиц стилей

Одна из основных задач XSLT состоит в преобразовании одних таблиц стилей в другие, хотя на первый взгляд это может быть неочевидно. Например, вам может потребоваться обработать длинные правила, которые нужно настроить непосредственно перед обработкой документов. И, как вы знаете, первоначально XSLT был представлен прежде всего для облегчения создания таблиц стилей объектов форматирования.

Но это порождает проблему: если вы обрабатываете таблицу стилей, полную таких элементов, как **<xsl:template>** и **<xsl:apply-templates>,** которые вы хотите видеть в результирующем документе — поскольку результирующий документ сам является таблицей стилей — как XSLT сможет отличить эти элементы буквального результата от элементов XSLT, которые он должен обработать?

Тут на помощь приходит элемент **<xsl:namespace-alias>,** поскольку он позволяет использовать новое пространство имен для элементов в исходном документе и преобразовывать это пространство имен обрашо в правильное в результирующем документе. У элемента два атрибута:

- **stylesheet-prefix** (обязательный). Префикс пространства имен, используемый в таблице стилей. Устанавливается в NCName или «#default»;

- result-prefix (обязательный). Префикс, URI которого вы хотите присвоить пространству имен в результирующем документе. Устанавливается в NCName или «#default».

Следующий пример пояснит сказанное. Представьте, что вам нужно создать такую таблицу стилей:

```
<?xml version-"1.0" encoding-"UTF-8"?>
<xsl:stylesheet
    xmlns:xsl-"http://www.w3 org/1999/XSL/Transform" vers1on-"l 1">
    <xsl:tempiate match-'PLANET">
        <TR>
            <TD>
                <xsl:value-of select-"NAME"/>
            </TD>
            <TD>
                <xsl:apply-templates select-"MASS"/>
            </TD>
            <TD>
                <xsl:apply-templates select="RADIUS"/>
            </TD>
            <T0>
                <xsl:apply-templates se1ect-"DAY"/>
            </T0>
        </TR>
    </xsl:template»
</xsl:stylesheet»
```

Заметьте, что здесь много элементов XSLT с префиксом «xsl», — значит, если вы попытаетесь создать их при помощи XSLT-преобразования, процессор XSLT попытается выполнить эти элементы. Чтобы избежать такой ситуации, я Эадал им новый префикс пространства имен, «xslt». Ниже показано, как это выглядит в таблице стилей, производящей предыдущую таблицу стилей (листинг 9.4). Заметьте, что нижеследующая таблица стилей просто выбирает корневой элемент исходного документа, чтобы она могла начать работать; она не использует исходный документ ни для каких других целей.

Листинг 9.4. Применение <xsl:namespace-alias>

```
<xsl:stylesheet version-"1.1"
    xmlns:xsl -"http://www.w3.org/1999/XSL/Transform"
    xmlns:xslt-"http://xslt"»

<xsl:tempiate match-"/"»
   <xslt:stylesheet vers1on-"1.l">
    <xslt:tempi ate match-"PLANET"»
       <TR»
          <TD»<xslt:value-of select-"NAME"/»</TD>
          <TD»<xslt: appl y - tempi ates sel ect" "MASS" /»</TD»
          <TD»<xslt:apply-tempiates select^'RADIUS"/»</TD»
          <TD»<xsl t: apply- tempi ates sel ect- "DAY" /x/TD»
       </TR>
   </xslt:tempiate»
   </xslt:stylesheet»
</xslrtemplate»
```

```
</xslStylesheet*
```

Здесь я использую пространство имен "http://xslt" для префикса "xslt", но я смогу изменить его в выходном документе на правильное пространство имен XSLT, "http://www.w3.org/ 1999/XSL/Transform", воспользовавшись элементом <xsl: namespace-alias>:

```
<xsl:stylesheet version-"1.1"
    xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"
    xmlns:xslt-"http://xslt">

<xsl:tempiate match-"/"*
   <xslt:stylesheet version-"l 1"*
    <xslt:tempiate match-"PLANET"*
       <TR>
          <TD*<xslt:value-of select="NAME"/></TD>
          <TD*<xslt:apply-templates select-"MASS"/></TD>
          <TD><xslt apply-templates select-"RADIUS"/»</TD>
          <TD><xslt apply-templates seiect-"DAY"/></TD>
       </TR>
   </xslt :tempi ate*
   </xslt:stylesheet*
</xsl:template*

<xsl.-namespace-alias stylesheet-prefix-"xslt" result-prefix-"xsl"/*

</xsl:stylesheet*
```

Вот результат. Заметьте, что здесь все еще используется префикс пространства имен «xslt», но это пространство имен теперь соответствует правильному пространству имен XSLT:

```
<?xml version-"Г,0" encoding-"UlF-8"?>
<xslt:stylesheet
    xmlns:xslt-"http://www.w3.org/1999/XSL/Transform" versi on-"1.1"*
    <xslt-.'template match-"PLANET"*
        <TR>
            <TD>
                <xslt:value-of select-"NAME"/*
            </TD*
            <TD>
                <xslt:apply-templates select-"MASS"/>
            </TD*
            <TD*
                <xs1t:apply-tempiates select«"RADIUS"/>
            </TD>
            <TD>
                <xslt:apply-templates select-"DAY"7>
            </TD*
        </TR>
    </xslt:tempi ate*
</xslt:stylesheet*
```

На этом глава завершается. В следующей главе мы рассмотрим, как работать с XSLT в коде.

Работа с API процессоров XSLT

10

Как вы видели на протяжении всей книги, при работе с XSLT вам не требовалось никакого программирования. Но все процессоры XSLT, с которыми мы работали до сих пор, — Xalan, Saxon, XT, Oracle и MSXML — разработаны так, что при желании к ним можно обращаться и из кода программы. Здесь мы рассмотрим, как работать с этими процессорами в прикладной программе. Можете спокойно пропустить эту главу, если вы не программист или не собираетесь им становиться. Но если вы не будете использовать возможности программного интерфейса, вы потеряете довольно много, в том числе поддержку XSLT на web-серверах. В этой главе будет также показано, как преобразовывать XML в реляционную базу данных.

Для взаимодействия с различными прикладными программными интерфейсами (API, Application Programming Interface) процессоров XSLT я буду применять Java и JavaScript. С различными процессорами XSLT можно работать и при помощи таких языков, как C++ или Visual Basic, но до сих пор подавляющее большинство программистов выбирало Java, а язык JavaScript удобен для обработки преобразований в Internet Explorer. Кроме того, используемые вами процессоры XSLT специально разработаны для взаимодействия с Java, хотя с процессором MSXML3 можно работать из программы как с COM-объектом. Как я говорил, с процессором MSXML в Internet Explorer можно работать при помощи JavaScript, такой пример был показан еще в главе 1. И, как я обещал в главе 1, здесь я рассмотрю его более подробно.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ РАБОТЫ С XSLT ИЗМЕНЯЕТСЯ ОЧЕНЬ БЫСТРО!

Будьте внимательны! Программное обеспечение для работы с XSLT изменяется очень быстро, зачастую ежемесячно, поэтому ко времени, когда вы будете читать книгу, оно частично может уже устареть. Туг уж ничего не поделаешь, и вам нужно за ним следить. Все примеры в этой главе были тщательно проверены по крайней мере тремя людьми – мной и двумя техническими редакторами, работавшими независимо, и на момент написания книги все примеры работали. Если у вас что-то не заработает, проверьте, конечно, такие элементы, как путь к классам Java, но также сверьтесь и с документацией по вашему процессору XSLT на предмет возможных изменений.

XSLT и JavaScript в Internet Explorer

Любите вы Microsoft или нет, нельзя отрицать тот факт, что эта фирма включает в Internet Explorer все больше средств поддержки XSLT (дополнительную информацию можно найти по адресу http://msdn.microsoft.com/xml/general/xmlparser.asp), и поэтому работа с браузером заслуживает нашего внимания. В главе 1 я представил пример создания преобразований XSLT в Internet Explorer при помощи JavaScript, и здесь мы рассмотрим этот пример подробнее. Как вы помните (см. главу 2), IE версии 5.5 и младше может осуществлять настоящие преобразования XSLT, если они реализованы на JavaScript (новый, только что вышедший IE 6.0 может разбирать непосредственно синтаксис XSLT просто путем просмотра XML-документов).

В этом случае я, пользуясь MSXML и JavaScript, преобразую planets.xml при помощи planets.xsl. Для хранения этих документов я создаю два новых объекта, XMLDocument и XSLDocument, опираясь на классы ActiveXObject и DOMDocument процессора MSXML в функции xslt. (Эта функция запускается сразу при загрузке страницы, поскольку я установил атрибут onload элемента <B0DY> в «xslt()»). Я также создам объект, соответствующий элементу <DIV>, который отображает результаты преобразования:

```
<HIML>
  <HEAD>

    <ITTLE>XSLT Using JavaScript</TITLE>

    <SCRIPT LANGUAGE-"JavaScript">
    function xslt()
    {
        var XMLDocument - new Acti veXObject('MSXML2.DOMDocument.3.0');
        var XSLDocument - new Acti veXObject('MSXML2.DOMDocument.3.0');
        var HTMLtarget - document.all['targetDIV'];
```

И planets.xml, и planets.xsl являются документами XML; и процессор MSXML может работать и как проверяющий на допустимость разборщик XML, если установить свойство validateOnParse в true. Для загрузки planets.xml и planets.xsl в объекты XMLDocument и XSLDocument служит метод load, Я также проверяю наличие ошибок, просматривая код ошибок разбора следующим образом:

```
<HIML>
  <HEAD>

    <ITTLE>XSLT Using JavaScript</TITLE>

    <SCRIP1 LANGUAGE-"JavaScript">
    function xsltO
    {
        var XMLDocument - new ActiveXObjectC'MSXML2.DOMDocument.3 0');
        var XSLDocument - new Act 1 veXOb ject CMSXML2 DOMDocument.3.0'>;
        var HTMLtarget - document.si 1 [' targetDIV' ];
        XMLDocument.validateOnParse - true:
```

```
XMLDocument.1oad('pianets.xml');
if (XMLDocument.parseError.errorCode !- 0) {
    HTMLtarget.innerHTML - "Error!"
    return false;
}

XSLDocument.validateOnParse - true;
XSLDocument.1oad('pianets.xsl');
if (XSLDocument.parseError.errorCode !- 0) {
    HTMLtarget.innerHTML - "Error!"
    return false;
}
```

Теперь, после того как оба файла, planets.xml и planets.xsl, были загружены, преобразование можно осуществить методом transform Node. Посмотрите, как я преобразую XMLDocument при помощи XSLDocument и показываю результат в элементе-приемнике <DIV>:

```
<HlML>
    <HEAD>

        <ITITLE>XSLT Using JavaScript</TITLE>

        <SCRIPT LANGUAGE-"JavaScript"*
        function xsltO
        {
            var XMLDcument - new Acti veXObject('MSXML2.DOMDocument3.0");
            var XSLDcument - new Acti veXOb ject C MSXM2 DOMDcument 3 0');
            var HTMLtarget - document.all['targetDIV']:


            HTMLtarget.1nnerHTML - XMLDocument.transformNode(XSLDocument):

            return true;
        }

        </SCRIPT>
    <HEAD>

    <BDY onload-"xslt()">
        <DIV ID-"targetDIV">
        </DIV>
    <BODY>
<HIML>
```

Эти результаты показаны на рис. 10.1.

Обработка ошибок разбора

При использовании JavaScript для загрузки документов XML и XSL и работы с ними важно знать, как сообщать об ошибках разбора. В предыдущем примере я сообщал об ошибках, выводя сообщение «Error!» в элементе-приемнике <QIV>

документа HTML, но это не очень информативно. Как получить дополнительную информацию?

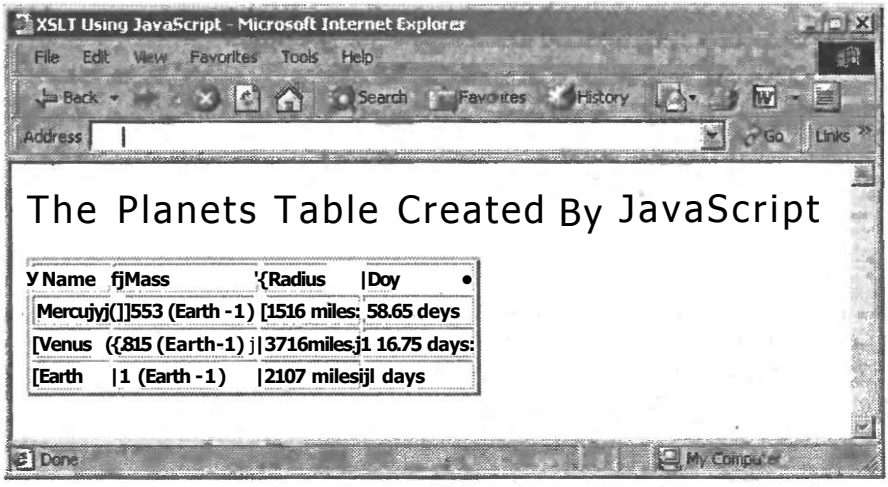


Рис. 10.1. Использование JavaScript для преобразования документа

В следующем примере я намерено создаю ошибку разбора, изменив первый тег <PLANET> в planets.xml на тег <PLANETS>:

```
<?xml version-"1.0"?>
<?xml-stylesheet type-"text/xml" href-"planets.xsl"?>
<PLANETS>

    <PLANETS>
        <NAME>Mecury</NAME>
        <MASS UNITS-"(Earth - 1)"> 0553</MASS>
        <DAY UNITS-"days">58.65</DAY>
        <RADIUS UNITS-"miles">1516</RADIUS>
        <DENSITY UNITS-"(Earth - 1)">.983</DENSITY>
        <DISTANCE UNITS-"million miles">43.4</DISTANCE><!–В перигелии–>
    </PLANET>
```

Я установил свойство validateOnParse объекта XMLDocument в true (значение по умолчанию — false, что означает отсутствие проверки), поэтому процессор **MSXML** отловит эту ошибку разбора. Объект XMLDocument содержит объект parseError, и если его свойство errorCode не равно нулю, это означает наличие ошибки. Сейчас я хочу не просто вывести сообщение «Error!», а расшифровать ошибку при помощи новой функции getError, которая?,возвращает строку с информацией о месте и причине ошибки.

С целый получёния этой дополнительной информации я использую свойства i<url, line, linepos и reason объекта parseError для того, чтобы определить вызвавший 'проблему файл, строку, позицию ошибки и ее описание (листинг 10:1).

Листинг 10.1. Создание преобразования XSLT и отображение ошибок разбора

```
<HTML>
    <HEAD>
        <TITLE>XSLT Using JavaScript</TITLE>

        «SCRIPT LANGUAGE"'JavaScript ">

        function xslt()
        i
            var XMLDocument - new ActiveXObject('MSXML2.DOMDocument.3.0"):
            var XSLDocument - new ActiveXObject("MSXML2.DOMDocument.3.0):
            var HTMLtarget - document.allC'targetDIV'];

            XMLDocument.validateOnParse = true:
            XMLDocument.1oad("pianets.xml'):
            if (XMLDocument,parseError.errorCode !- 0) {
                HTMLtarget.innerHTML - getError(XMLDocument)
                return false:
            }

            XSLDocument.validateOnParse - true;
            XSLDocument.load*"planets.xsl');
            if (XSLDocument parseError.errorCode !- 0) {
                HTMLtarget.innerHTML - getError(XSLDocument)
                return false:
            }

            HTMLtargetInnerHTML - XMLDocument.transformNode(XSLDocument):

            return true;
        }

        function getError(errorObject)
        {
            var Error • new String:
            Error - "Error. " + errorObject.parseError.url + "<BR>"
                + "Line: " + error0bject.parseError.l1ne + "<BR>"
                + "Character: " + errorObject.parseError.linepos + "<BR>"
                + "Description: " + errorObject.parseError.reason:
            return Error:
        }

        </SCRIPT>
    </HEAD>

    <&DY onload-"xslt()">
        <DIV ID-"targetDIV">
        </DIV>
    </B0DY>
</HTML>
```

Результат можно увидеть на рис. 10.2, где показаны вызвавший ошибку файл, место ошибки и описание ошибки процессором MSXML. Если вы собираетесь реализовывать преобразования XSLT в Internet Explorer, когда пользователи просматривают ваши документы случайным образом, обработка подобных ошибок разбора исключительно важна.

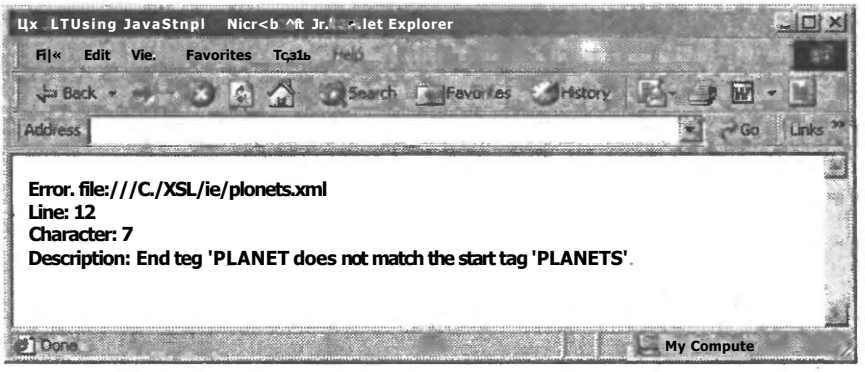


Рис. 10.2. Обработка ошибки разбора

Пока что я преобразовывал при помощи процессора MSXML только документ целиком, но здесь у вас существует значительно больше возможностей. Например, в следующем разделе я воспользуюсь XSLT в Internet Explorer для сортировки данных планет в HTML-таблицы по нажатию кнопок. Для этого я буду обращаться к отдельным узлам внутри таблицы стилей.

Internet Explorer и динамические стили

Для того чтобы продемонстрировать все возможности управления преобразованиями XSLT при использовании процессора MSXML, в этом примере я разрешу пользователю динамически сортировать таблицу Planets. Общая идея показана на рис. 10.3. Пользователю достаточно щелкнуть на кнопку, чтобы отсортировать таблицу по названию, массе, радиусу или дню.

Для сортировки таблицы я, как легко догадаться, применяю таблицу стилей с элементом <xsl:sort> (листинг 10.2). По умолчанию я сортирую планеты по названию.

Листинг 10.2. Применение динамических XSLT-преобразований

```
<?xml version-"1.0" encoding-"iso-8859-1" ?>
<xsl :stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version*"1.1">

    <xsl template match-"/">
        <HIML>
            <HEAD>
                <ITTLE>
                    The Sorted Planets Table
                </ITTLE>
            <HEAD>
            <BODY>
                <Hl>
                    The Sorted Planets Table
                </Hl>
                <TABLE BORDER~'2'>
                    <TR>
                        <TD>Name</TD>
```

продолжение &

Листинг 10.2 (продолжение)

```
                        <TD>Mass</TD>
                        <TD>Radius</TD>
                        <TD>Day</TD>
                    </TR>
                    <xsl:apply-tempiates select-"/PLANETS/PLANET">
                        <xsl:sort select-"NAME" order="ascending"/>
                    </xsl:apply-tempiates>
                </TABLE>
            </BODY>
        </HIML>
    </xsl:tempiate>

    <xsltemplate match-"PLANET">
        <TR>
            <TD><xsl:value-of select-"NAME"/></TD>
            <TD><xsl:value-of select-"MASS'7></TD>
            <TD><xsl:value-of select-"RADIUS'7></TD>
            <TD><xsl:value-of select="DAY"/></TD>
        </TR>
    </xsl:tempiate>

</xsl:stylesheet>
```

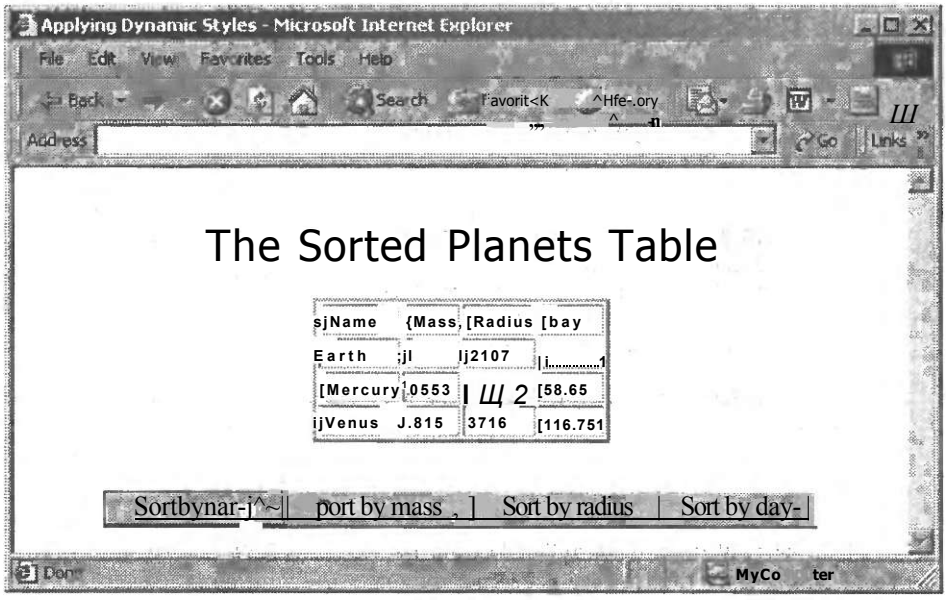


Рис. 10.3. Поддержка динамических преобразований XSLT

Эта таблица стилей сортирует цланеты в. алфавитном порядке по их названиям; но есть и другие возможности: можно позволить пользователю осуществлять сортировку и по другому критерию. Для этого выберите в данной таблице стилей атрибут select элемента <xsl:sort> и при помощи JavaScript измените его динамически с «NAME» на «MASS», «RADIUS» или на другой критерий по желанию

пользователя, и затем снова выполните преобразование XSLT При повторном осуществлении преобразования появится новая таблица с новым порядком сортировки.

В качестве первого шага в создании нужной нам HTML-страницы я загружу planets.xml и требуемую таблицу стилей и выполню сортировку по умолчанию, то есть по названию планет:

```
<HIML>
    <HEAD>
        <TITLE>
            Applying Dynamic Styles
        </TITLE>

        «SCRIPT LANGUAGE»»"JavaScript"»
            var XMLDocument:
            var XSLDocument:
            var HTMLtarget:

            function initiallzeO
            {
                XMLDocument - new ActiveXObject('MSXML2.DOMDocument.3.0'):
                XSLDocument - new Acti veXObject('MSXML2.DOMDocument.3.0'):
                HTMLtarget - document.all['targetDIV']:

                XMLDocument.validateOnParse - true:
                XMLDocument.load('planets.xml');
                if (XMLDocument.parseError.errorCode 1- 0) {
                    HTMLtarget.innerHTML - "Error!"
                    return false:
                }

                XSLDocument.validateOnParse - true:
                XSLDocument.1oad('pianets.xsl');
                1f (XSLDocument.parseError.errorCode !e 0) {
                    HTMLtarget.InnerHTML - "Error!"
                    return false:
                }
                HTMLtarget.innerHTML - XMLDocument.transformNode(XSLDocument):
            }
```

Эту часть вы уже видели раньше. Но пользователь теперь может отсортировать таблицу щелчком мыши по массе, радиусу и т. д. Для новой сортировки таблицы я написал функцию sort — ей при щелчке на кнопке передается имя узла (например, «MASS»), по которому нужно провести сортировку. Вот как создаются различные кнопки, показанные на рис. 10.3:

```
<INPUT TYPE='BUTTON' 0NCLICK-"Sort('NAME')" VALUE«'Sort by name'></INPUT>
<INPUt TYN='BUTt0N' ONCLICK-"SortCMASST VALUE»'Sort by maSS'></INPUT>
<INPUT TYPE'BUTTON' OfiCLICK*"sort('RADIUS-*F VALUE-"5ort by radiuS"></INPUT>
<INPUT TYPE'BUTTQN" ONCLICK-"sort<'DAY')" VALUE*"5ort by day></INPUT>
```

Затем в функции sort я хочу выполнить новую сортировку по имени переданного узла. Для этого я изменяю атрибут select элемента <xsl:sort> на имя нового узла, по которому нужно сортировать. Вот как теперь выглядит атрибут select:

```
<xs1:apply-tempiates select-"/PLANETS/PLANET">
                    <xsl:sort select›"NAME" order-"ascending"/>
                </xsl:apply-templates>
```

Я могу обратиться к этому узлу из таблицы стилей, теперь хранимой в объекте XSLDqcument, передав в метод selectSingleNode упомянутого объекта выражение XPath. Метод selectSingleNode возвращает объект node, и можно изменить текстовое значение узла при помощи свойства nodeValue объекта node. В таком случае я только устанавливаю атрибут select в имя нового узла, по которому будет вестись сортировка:

```
<HIML>
    <HEAD>
        <ITTLE>
            Applying Dynamic Styles
        </ITTLE>

        <SCRIPT LANGUAGE-"JavaScri pt">

            .
            .
            function initialize()
            {
            .
            .
            .
            }

            function sort(sortNode)
            {
                (XSLDocument.selectS1ngleNode("//xsl:sort/Gselect")).nodeValue =
                 sortNode:
            .
            .
            .
            }
        </SCRIPT>
            .
            .
            .
```

Теперь все, что осталось сделать, — снова выполнить преобразование и вывести результаты:

```
    </HEAD>
<HIML>
    <HEAD>
        <ITTLE>
            Applying Dynamic Styles
        </ITTLE>
        <SCRIPT LANGUAGE-"JavaScri pt">
```

```
        function initialize()
        {

        }
        function sort(sortNode)
        {
            (XSLDocument.selectSing1eNode("//xsl:sort/@select")).nodeValue -
            sortNode:
            HTMLtarget.innerHTML - XMLDocument.transformNode(XSLDocument);
        }
    </SCRIPT>
</HEAD>
```

Операция завершена. Результаты показаны на рис. 10.3. При щелчке на кнопке таблица сортируется заново по выбранному значению узла (имейте в виду, что сортировка ведется по алфавиту; сортировка по числовым значениям обсуждается в главе 5, где описывается элемент <xsl:sort>) и отображается еще раз с новым порядком сортировки. Вот вся HTML-страница (листинг 10.3).

Листинг 10.3. Применение динамических XSLT-преобразований

```
<HTML>
    <HEAD>
        <TITLE>
            Applying Dynamic Styles
        </TITLE>

        <SCRIPT LANGUAGE="JavaScript">
            var XMLDocument
            var XSLDocument
            var HTMLtarget:

            function initialize()
            {
                XMLDocument - new ActiveXObject("MSXML2.DOMDocument.3.0");
                XSLDocument - new ActiveXObject("MSXML2.DOMDocument.3.0");
                HTMLtarget - document all['targetDIV'];

                XMLDocument.validateOnParse - true;
                XMLDocument.1oad('pianets.xml');
                if (XMLDocument.parseError.errorCode != 0) {
                    HTMLtarget.innerHTML - "Error!"
                    return false:
                }

                XSLDocument.validateOnParse - true:
                XSLDocument loadCplanets.xsD:
                if (XSLDocument.parseError.errorCode !« 0) {
                    HTMLtarget.innerHTML - "Error!"
                    return false;
                }
```

продолжение &

Листинг 10.3 (продолжение)

```
            HTMLtargetInnerHTML - XMLDocument.transformNode(XSLDocument);
        }

        function sort(sortNode)
        {
            (XSLDocument.selectSingleNodeCV/xsl:sort/@select")) nodeValue -
            sortNode:
            HTMLtargetinnerHTML - XMLDocument.transformNode(XSLDocument):
        }
    </SCRIPT>
</HEAD>

<BODY ONLOAD"Ini ti ali ze()">
    <CENTER>

        <DIV ID="targetDIV"></div>
        <BR>
        <BR>

        <INPUT TYPE'BUTION" 0NCLICK*"sort('NAME')" VALUE-'Sort by name'></INPUT> ^
        <INPUT TYPE'BUTION" ONCLICK-"sort('MASS')" VALUE-'Sort by mass'></INPUT>
        <INPUT TYPE'BUTION" ONCLICK-"sort('RADIUS')" VALUE-'Sort by radius"></INPUT>
        <INPUT TYPE'BUTION" ONCLICK-"sort('DAY')" VALUE-'Sort by day"x/INPUT>

    <CENTER>
</BODY>
</HTML>
```

На самом деле загрузить документы XML и XSL в Internet Explorer можно несколькими способами. Для создания объектов XMLDocument и XSLDocument я использовал класс ActiveXObject, но эти объекты можно создавать и непосредственно обращаясь к содержащим документы XML объектам ActiveX по идентификатору класса, хранимому в реестре Windows. В следующем примере я таким образом загружу planets.xml и planets.xsl в XMLDocument и XSLDocument:

```
XMLDocument - document.all['XMLdoc'];
XSLDocument - document.all['XSLdoc'];
XMLDocument 1oad('pianets.xml');
XSLDocument 1 oad(' pi anets. xsl');

    .
    .

OBJECT ID-"XMLdoc" WIDTH-'0' HEIGHT-'0'
    CLASSID-"clSid'f5078f32-c551-lld3-89b9-0000f81fe221">
</0BJECT>

OBJECT ID-"XSLdoc" WIDTH-'0' HEIGHT‹'0'
    CLASSID-"clsid f5078f32-c551-lld3-89b9-0000f81fe221">
</0BJECT>
```

Эта техника не так надежна, как использование класса ActiveXObject, поскольку идентификаторы классов могут различаться в разных версиях Internet Explorer. (Приведены идентификаторы классов для Internet Explorer 5.5) Но есть еще один способ загрузить документы XML и XSL в Internet Explorer — при помощи *участков* (island) XML.

Internet Explorer и участки данных XML

В Internet Explorer есть специальный тег <XML>, при помощи которого можно создавать участки (island) XML. Участок XML может содержать либо сам код XML, либо ссылку на XML-документ.

Участки XML упрощают загрузку документов XML и XSL, поэтому их стоит здесь рассмотреть. В следующем примере я создаю два участка XML, sourceDocument и stylesheet, и загружаю planets.xml и planets.xsl, просто обратившись к ним через атрибут sre:

```
<HTML>
    <HEAD>
        <TITLE>
            The Planets Table
        </TITLE>

        <XML 1 d-"sourceDocument" src-"pl anets. xml "x/XML>
        <XML id-"stylesheet" src-"planets.xsl"></XML>
```

Теперь для осуществления XSLT-преобразования мне нужно только применить, как и раньше, метод transform Node и присвоить результаты элементу <DIV>, чтобы вывести их (листинг 10.4).

Листинг 10.4. Загрузка документов **XML** и **XSL** при помощи участков **XML**

```
<HTML>
    <HEAD>
        <TITLE>
            The Planets Table
        </TITLE>

        <XML id-"sourceDocument" src-"planets.xml"></XML>
        <XML id-"stylesheet" src-"planets.xsl"></XML>

        <SCRIPT F0R-"window" EVENT-"onload">
            targetDIV.innerHTML -
                sourceDocument.transformNode(stylesheet.XMLDocument):
        </SCRIPT>

    <HEAD>

    <BODY>
        <CENTER>
            <DIV id="targetDIV"x/DIV>
        <CENTER>
    </BODY>
<HTML>
```

Вот и все. Заметьте, что по умолчанию Internet Explorer 5.5 и младше использует устаревший процессор XSLT, как обсуждалось в главе 2 (если специально не установить Процессор MSXML3 в режиме замены или IE 6.0, о чем также говорилось в главе 2). Если вы работаете с IE 5.5 или младше, у вас в расспоря-

жении таблица стилей Internet Explorer прежнего образца, в которой нет правил по умолчанию и в которой вам придется использовать старое пространство имен XSL, как в этом примере (листинг 10 5).

Листинг 10.5. Таблица стилей Internet Explorer старого образца

```
<?xml vers1on-"l.0"?>
<xsl:stylesheet version-;" 1.1" xmlns:xsl-"http://www.w3.org/TR/WD-xsr>

    <xsl:tempi ate match-"/">
        <HIML>
            <HEAD>
                <ITTLE>
                    The Planets Table
                </ITTLE>
            <HEAD>
            <BODY>
                <Hl>
                    The Planets Table
                </Hl>
                <TABLE BORDER-'2'>
                    <IR>
                        <ID>Name</ID>
                        <ID>Mass</ID>
                        <ID>Radius</ID>
                        <I0>Day</ID>
                    </IR>
                    <xsl:apply-templates/>
                </TABLE>
            <BODY>
        <HIML>
    </xsl:tempiate>

    <xslrtemplate match-'PLANETS'>
        <xsl:apply-templates/>
    </xsl:template>

    <xsl:template match-'PLANET'>
        <IR>
            <TDxxsl: value-of select-"NAME'7x/TD>
            <TDxXsl value-of select-"MASS"/x/TD>
            <TDxXsl: val ue-of select-"RADIUS"/x/TD>
            <TDxxsl :value-of select-"DAY"/x/TO>
        </IR>
    </xsl:tempiate>

</xsl:stylesheet»
```

Как видите, при помощи JavaScript и XSLT в Internet Explorer можно сделать весьма многое. Дополнительную информацию можно подучить из руководства разработчика XSLT фирмы Microsoft, которое сейчас расположено по адресу http:// msdn.microsoft,com/libraiy/default.asp?URL»/library/psdk/xmlsdk/xslp8tlx.htm.

Пришло время рассмотреть взаимодействие XSLT с Java, начав с обращения к Java непосредственно из процессоров XSLT.

Вызов Java непосредственно из процессоров XSLT

Как мы говорили в главе 5, до недавнего времени процессоры XSLT могли реализовывать функции расширения любым способом, и один из этих способов включал прямой вызов функций Java. Например, в Saxon и Xalan можно выполнять код Java, если определить пространство имен, указывающее класс Java как последнюю часть своего URI — как в следующем примере, где я определил пространство имен Date, соответствующее классу Java Date:

```
<?xml version-"!..0"?>
<xsl:stylesheet version-"!.1"
xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"
xmlns:Date-"http://www.saxon.com/java/java.ut11.Date">
```

Как вы видели в главе 5, это означает, что теперь вы можете применять такие функции Java, как toString и new, для встраивания текущей даты в заголовок HTML <H1>, — например, так (листинг 10.6).

Листинг 10.6. Работа с функциями класса Date Java

```
<?xml version-"1.0"?>
<xsl.-stylesheet version-"1.1"
xmlns•xsl-"http.//www w3.org/1999/XSL/Transform"
xmlns:Date-"http://www.saxon.com/java/java.uti1.Date">

    <xsl -.template match-"/PLANETS">
        <HIML>
            <HEAD>
                <ITITLE>
                    The Planets Table
                </ITITLE>
            </HEAD>
            <BODY>
                <H1>
                    The Planets Table
                </H1>
                <BR/>
                <H1>
                    <xsl:value-of select-"Date:toStnng(Date:new())'7>
                </H1>
                <TABLE BORDER-'2'>
                    <ID>Name</ID>
                    <ID>Mass</ID>
                    <ID>Radius</ID>
                    <ID>Day</ID>
                    <xsl:apply-templates/>
                </TABLE>
            </BODY>
        </HIML>
    </xsl:tempiate>

    <xsV: tempiate match-'PLANET'>
```

продолжение^

Листинг 10.6 (продолжение)

```
        <TR>
          <TD><xsl:value-of select-"NAME"/></TD>
          <TD><xsl:apply-templates select-"MASS"/></TD>
          <TD><xsl: apply-templ ates select-"RADIUS7></TD>
          <TD><xs1:apply-tempiates select-"DAY"/></TD>
        </TR>
      </xsl:tempiate>

      <xsl:template match-"MASS">
          <xsl:value-of select="."/>
      </xsl:tempiate>

      <xsl:template match="RADIUS">
          <xsl:value-of select-"."/>
      </xsl:tempiate>

      <xsl:tempi ate matchk"DAY">
          <xsl:value-of select-"."/>
      </xsl:tempiate>

  </xsl stylesheet>
```

Такой код, безусловно, работает, но дает лишь ограниченные средства работы и зависит от нестандартных расширений. За исключением случая, когда вы осуществляете только несколько простых вызовов, обычно лучше начать работать в Java и взаимодействовать с процессором XSLT.

Каждый из процессоров XSLT Xalan, Saxon, XT и Oracle определяет API, к которому можно обращаться из Java. Достаточно только проверить, что нужные файлы JAR находятся в classpath Java. Мы уже рассматривали, как работать с файлами JAR и путями к классам, в главе 1; теперь мы перейдем к написанию кода на Java, а не просто выполнению предопределенных классов из командной строки. Код можно загрузить с http://www.newriders.com/books/title.cfm?isbn=0735711364, поэтому если вы не программист на Java, можете пропустить эту часть главы.

Работа с API XSLT для Java

Во всех основанных на Java процессорах XSLT определен объемный интерфейс, API, для связи с Java. Как правило, при помощи API вызываются методы и создаются объекты Java, как мы увидим в этой главе. Но все процессоры XSLT определяют свои API по-разному, и обычно эти API довольно велики, поскольку каждый процессор может создавать собственные иерархии классов по своему усмотрению.

В следующих разделах мы пройдем по этапам создания XSLT-преобразований из Java с использованием процессоров XSLT Xalan, Saxon, XT и Oracle. Все их возможности описаны в документации. Заметьте, что описание API большинства этих процессоров может само занять отдельную книгу, потому наше обсуждение вынужденно представляет собой лишь краткий обзор.

НАЗВАНИЯ ФАЙЛОВ JAVA

В отличие от остальных примеров книги, при создании файлов Java имеет значение имя файла. Java требует, чтобы имя файла совпадало с именем содержащегося в файле открытого класса. По этой причине для каждого листинга я включил в заголовок имя используемого файла. Например, заголовок «Листинг 10.7, xalanjava.java, взаимодействие Xalan с Java» означает, что перед тем, как использовать код листинга вместе с Java, необходимо сохранить листинг как xalanjava.java (для простоты имена всех открытых классов и файлов Java я писал в нижнем регистре). Обратите внимание: если вы загрузили код книги, этот пример будет называться 10-07.java, и перед работой с ним вам нужно будет его переименовать в xalanjava.java. Кроме того, требуемое имя файла будет приведено в комментариях в начале каждого файла Java.

Взаимодействие Xalan с Java

Несмотря на свой размер, Xalan — один из простейших процессоров XSLT для взаимодействия с Java. В качестве примера я создал новый класс Java, xalanjava, и для осуществления XSLT-преобразования к нему положено обращаться следующим образом:

```
C:\>java xalanjava planets.xml planets.xsl planets.html
```

Чтобы создать XSLT-преобразование, я начну с создания объекта класса TransformerFactory в классе xalanjava:

```
import javax.xml transform.Transformer:
import javax.xml.transform.TransformerFactory:
Import javax.xml.transform.stream.StreamSource;
import javax.xml.transform.stream.StreamResult:
import javax.xml transform.TransformerException;
import javax.xml.transform.TransformerConfigurationException:
import java.io.FileOutputStream:
import java.io.Fi1eNotFoundException:
import java.io.IOException:

public class xalanjava
{
    public static void main(String[] args)
        throws TransformerException. TransformerConfigurationException.
        Fi 1 eNotFoundException. IOException
    {
        TransformerFactory tFactory - TransformerFactory.newInstanceO:
        .
        .
        .
```

Имена документа XML, документа XSL и результирующего документа были нам переданы в аргументах соответственно как args[0], argsfl] и args[2]. На следующем шаге нужно загрузить документ XSL в новый созданный мной объект TransformerFactory. Для этого я создал объект StreamSource и передал его в метод newTransformer объекта TransformerFactory, чтобы создать новый объект Transformer:

```
0 import javax.xml transform.Transformer;
-i
-/
    public class xalanjava
```

```
{
    public static void ma1n(String[] args)
        throws TransformerException. TransformerConfigurationException,
        FileNotFoundException. IOException
    {
        TransformerFactory tFactory - TransformerFactory.newInstanceO;

        Transformer transformer -
            tFactory.newTransformer(new StreamSource(args[l]));
        .
        .
        .
```

Объект transformer реализует XSLT-преобразование. Для того чтобы выполнить такое преобразование, нужно вызвать метод transform этого объекта, передав ему соответствующий XML-документу объект StreamSource и соответствующий результирующему документу объект Stream Result:

```
import javax xml.transform Transformer;
        .
        .
        .
public class xalanjava
{
    public static void main(String[] args)
        throws TransformerException. TransformerConfigurationException.
        Fi1eNotFoundException. IOExcepti on
    {
        .
        .
        .
transformer.transform(new StreamSource(args[0]),
            new StreamResult(new Fi1eOutputStream(args[2]))):
    }
}
```

Вот полный файл Java, xalanjava.java (листинг 10.7).

Листинг 10.7. xalanjava.java, взаимодействие Xalan с Java

```
import javax.xml.transform.Transformer:
import javax.xml transform.TransformerFactory;
import javax.xml.transform.stream.StreamSource:
import javax.xml.transform.stream.StreamResult:
import javax.xml transform.TransformerException:
import javax.xml.transform.TransformerConfigurationException;
import java.io.FileOiitputStream;
import java.io FileNotFoundException:
import java.io.IOException:

public class xalanjava
{
    public static void main(String[] args)
        throws TransformerException. TransformerConfigurationException.
        Fi1eNotFoundExcepti on. IOExcept ion
    {
        TransformerFactory tFactory - TransformerFactory.newInstanceO;

        Transformer transformer -
```

```
        tFactory. newTransformer(new StreamSource(args'[ 1])):

transformer transform(new StreamSource(args[0]),
        new StreamResultCnew File0utputStream(args[2])));
    }
}
```

Чтобы скомпилировать файл xalanjava.java в xalanjava.class и выполнить этот класс, задайте classpath так, чтобы путь включал Xalan и разборщик XML, который вы обычно используете с Xalan, Xerces (о Xerces говорилось в главе 1, в том числе и о том, где его взять), — в Windows это будет выглядеть так (как всегда, пишите пути в соответствии со своей системой):

```
C:\>set classpath-.;C:\xalan\xalan.j,ar:c:\xalan\xerces.jar
```

Затем скомпилируйте xalanjava.java компилятором с Java, javac:

```
C:\>javac xalanjava.java
```

Здесь считается, что путь Kjavac.exe прописан, и его можно сразу активизировать из командной строки. (Как правило, компилятор Java javac.exe находится в каталоге Java bin — поэтому если путь к нему не задан, к нему можно обратиться, например, так: C:\>c:\jdkl.3\bin\javac xalanjava.java.) Компилятор Java создаст файл xalanjava.class, при помощи которого производится преобразование:

```
C:\>java xalanjava planets.xml planets.xsl planets.html
```

При этом из файлов planets.xml и planets.xsl будет создан файл planets.html — здесь я его создал при помощи своего собственного класса Java.

РАБОТА С ПАКЕТОМ JAXP ФИРМЫ SUN ДЛЯ XSLT

Фирма Sun, создатель Java, обладает пакетом Java для обработки XML, JAXP, который можно загрузить с http://java.sun.com/xml. JAXP способен также осуществлять XSLT-преобразования. Но я не буду отдельно рассматривать JAXP в текущей главе, поскольку этот пакет (по крайней мере, сейчас) для всех своих преобразований использует Xalan, поставляемый в xalan.jar. Значит, предыдущий пример, листинг 10.7, без всяких изменений можно использовать с пакетом JAXP.

Взаимодействие Saxon с Java

Процессор Saxon также определяет API для работы с Java, но, конечно, в деталях этот прикладной интерфейс отличается от API Xalan. Для демонстрации создания преобразований при помощи Saxon API версии 6.0.2 я создам новый класс Java saxonjava. Начать нужно с создания нового объекта XSLT Processor, вызвав метод newlnstance класса Processor в файле saxonjava .java:

```
import java.io.*;
import org.xml.sax.*:
import org.w3c.dom.*:
import com.icl.saxon.trax.*:

public class saxonjava
{

    public static void main(String args[])
```

```
        throws ProcessorException. ProcessorFactoryException.
        TransformException. SAXException. IOException
    {
        Processor processor - Processor.newInstanceC'xslt"):
        .
        .
        .
```

Затем необходимо создать объект Templates на основе таблицы стилей XSL, которую мы хотим применить, хранимой в argsfl]. Это можно сделать при помощи класса InputSource:

```
import java.io.*:
        .
        .
        .
public class saxonjava
{
    public static void main(String args[])
        throws ProcessorException, ProcessorFactoryException.
        TransformException. SAXException, IOException
    {
        Processor processor - Processor.newInstanceC'xslt"):

        Templates templates -
            processor.process(new InputSource(args[l])):
        .
        .
        .
    }
}
```

При помощи нового объекта Templates можно создать объект Transformer, который в действительности делает работу:

```
import java.io.*:
        .
        .
        .
public class saxonjava
{
    public static void main(String args[])
        throws ProcessorException, ProcessorFactoryException.
        TransformException. SAXException, IOException
    {
        Processor processor - Processor.newInstanceC'xslt"):

        Templates templates -
            processor.process(new InputSource(args[l])):

        Transformer transformer - tempi ates. newTransformerO:
        .
        .
        .
    }
}
```

Наконец, чтобы осуществить XSLT-преобразование, нужно вызвать метод transform объекта transformer, записывая результат в выходной документ при помощи объекта FileWriter (листинг 10.8).

Листинг 10.8. saxonjava .java, взаимодействие Saxon с Java

```
import java.io.*:
import org.xml.sax.*;
import org.w3c.dom.*:
import com icl.saxon.trax.*:

public class saxonjava
{

    public static void main(String args[])
        throws Processortxception. ProcessorFactoryException,
        TransformException. SAXException. IOException
    {
        Processor processor - Processor.newlnstance("xslt");

        Templates templates -
            processor.process(new InputSource(args[1])):

        Transformer transformer = tempiates.newTransformerO;

        transformer.transform(new InputSource(args[0]).
            new Result(new FileWriter(args[2])));
    }
}
```

Чтобы скомпилировать и использовать новый класс saxonjava, нужно установить classpath так, чтобы переменная включала путь к saxon.jar:

```
C;\>set cIasspath-.:c:\saxon\saxon.jar
```

Затем при помощи компилятора Java, javac, создается saxonjava.class.

Как и многие использующие API процессоры XSLT, Saxon ожидает, что ему будут переданы URL документов, с которыми вы хотите работать, — что я и делаю на следующем шаге:

```
C:\>java saxonjava http://www.starpowder com/planets.xml
http://www.starpowder.com/planets.xsl pianets.html
```

Таким образом, создается planets.html, как и раньше. Заметьте, что если документы расположены локально, можно использовать URL файла. Например, в Windows, если документ XML расположен в c:\XSL\saxonjava\planets.xml, а документ XSL в c:\XSL\saxonjava\planets.xsl, можно выполнить такую командную строку:

```
C:\>java saxonjava file:///XSL/saxonjava/planets.xml
file:///XSL/saxonjava/planets.xsl planets.html
```

ПРЕОБРАЗОВАНИЕ ИМЕН ФАЙЛОВ В URL

Если вам больше нравится писать в командной строке имена файлов, а не URL, имена можно преобразовывать в URL в коде. Для этого необходимо передать полный путь к файлу в класс Java URL, а путь можно получить методм getAbsolutePath класса File: File file = new File(filename); String fullpath = file.getAbsolutePath();.

В этом примере мы работали с Saxon версии 6.0.2, о котором на web-узле Saxon говорится как о вполне надежной версии, но на момент написания книги появилась новая, полностью не протестированная версия Saxon 6.2.2. (Не существовало версии Saxon 6.1.x.) В последней версии Saxon, кажется, собирается вернуться к той же модели API, которую использует Xalan, и код, работоспособный в версии 6.0.2, не будет работать в версии 6.2.2 (сюрприз!). Ниже приведен код saxonjava.java для версии 6.2.2 — проверьте, что при работе с этим кодом вы включили в classpath новую версию saxon.jar, и обратите внимание на то, что при его выполнении вам нужно передавать не URL файлов, а только их имена. Этот код идентичен показанному ранее xalanjava.java, за исключением имени класса, saxonjava:

```
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax xml.transform.stream.StreamSource:
import javax.xml.transform.stream.StreamResult:
import javax.xml.transform.TransformerException:
import javax.xml.transform TransformerConfigurationException:
import java.io.FileOutputStream;
import Java.io.Fi1eNotFoundException;
import java.io.IOException;

public class saxonjava
{
    public static void main(String[] args)
        throws TransformerException. TransformerConfigurationException,
        Fi1eNotFoundExcepti on. IOExcepti on
    {
        TransformerFactory tFactory - TransformerFactory.newInstanceC);

        Transformer transformer -
            tFactory.newTransformer(new StreamSource(args[l]));

transformer.transform(new StreamSource(args[0]).
            new StreamResult(new File0utputStream(args[2])));
    }
}
```

Взаимодействие процессора XSLT Oracle с Java

Несколько больше усилий потребуется для осуществления XSLT-преобразования при помощи API процессора XSLT фирмы Oracle. Новый пример, oraclejava.javg, продемонстрирует работу с этим API.

В oraclejava.java в первую очередь необходимо считывать требуемые документы при помощи объекта DOM Parser:

```
import org.w3c.dom.*;
import java.util *:
import java.io.*:
import java.net.*.
import oracle.xml.parser.v2.*;
```

```
public class oraclejava
{
    public static void main (String args[]) throws Exception
    {
        DOMParser parser;

        try
        {
            parser - new DOMParserO;
            parser.setPreserveWhitespace(true)i
        .
        .
        .
```

Затем, чтобы считать исходный документ **XML** и документ таблицы стилей **XSLT,** нужно преобразовать их **URL** в объекты URL Java при Помощи метода parse объекта parser. После этого я вызываю метод разборщика getDocument, для того чтобы извлечь и сохранить Документы **XML** и **XSLT** в объектах XMLDocument:

```
public class oraclejava
{
    public static void main (String args[]) throws Exception
    {
        DOMParser parser:

        XMLDocument xmldoc. xsldoc;
        URL xslURL;
        URL xmlURL;

        try
        {
            parser - new DOMParserO:
            parser.setPreserveWn itespace(true):

            xmlURL - new URL(args[0]):
            parser.parse(xmlURL):
            xml doc - parser.getDocumentO;

            xslURL - new URL(args[l]);
            parser.parse(xslURL);
            xsl doc - parser.getDocumentO;
        .
        .
        .
```

В этот момент planets.xml и planets.xsl заключены в объекты XMLDocument Для выполнения преобразования мне необходимы еще объекты XSLStylesheet и XSLProcessor для таблицы стилей **XSLT.** Фактическое преобразование **XSLT** осуществляется методом processXSL объекта parser, возвращающего фрагмент документа:

```
public class oraclejava
{
    public static void main (String args[]) throws Exception
    {
```

```
DOMParser parser;
.
.
.
    xslURL - new URL(args[l]):
    parser.parse(xslURL):
    xsl doc = parser.getDocumentO;

    XSLStylesheet xslstylesheet - new XSLStylesheet(xsldoc, xslURL):

    XSLProcessor processor - new XSLProcessorO:

    DocumentFragment docfragment -
        processor.processXSL(xslstylesheet. xmldoc);
    .
    .
    .
```

Этот код завершает преобразование. Теперь задача состоит в том, чтобы преобразовать данный фрагмент документа в документ XML, который можно записать на диск, — для чего я и создам новый XML-документ, newdoc, и вставлю фрагмент документа в корень нового документа:

```
import org w3c dom.*;
.
.
.
public class oraclejava
{
    public static void main (String args[]) throws Exception
    {
        DOMParser parser:
        XMLDocument xmldoc, xsldoc, newdoc:
        URL xslURL;
        URL xmlURL:

        try
        {
            .
            .
            .
            DocumentFragment docfragment -
                processor processXSL(xslstylesheet, xmldoc):

            newdoc - new XMLDocumentO;

            Element rootElement • newdoc.createElementC'root");
            newdoc.appendChild(rootElement);

            rootElement.appendChi1d(docfragment):
            ,   i          '•'  uu                    '                •)[1]
            .                                                          j. iF
            .
```

Теперь осталось только сохранить на диске новый XML-документ с именем, заданным в args[2]. В этих целях я использую объект FileOutputStream, и вот полный код (листинг 10.9). I

Листинг 10.9. oraclejava.java, взаимодействие процессора XSLT Oracle с Java

```
import org.w3c.dom.*;
import java.util *:
import java.io.*:
import java.net.'*:
import oracle xml parser.v2.*:

public class oraclejava
{
    public static void main (String args[]) throws Exception
    {
        DOMParser parser:
        XMLDocument xmldoc. xsldoc. newdoc;
        URL xslURL:
        URL xmlURL:

        try
        {
            parser - new DOMParserO;
            parser.setPreserveWhi tespace(true):

            xmlURL - new URL(args[0]);
            parser.parse(xmlURL):
            xml doc - parser.getDocumentO:

            xslURL - new URL(args[l]):
            parser.parse(xslURL);
            xsl doc - parser.getDocumentO:

            XSLStylesheet xslstylesheet - new XSLStylesheet(xsldoc. xslURL);

            XSLProcessor processor - new XSLProcessorO;

            DocumentFragment docfragment -
                processor.processXSL(xs1 sty1esheet. xmldoc) ;

            newdoc - new XMLDocumentO;

            Element rootElement - newdoc.createE1ement("root"):
            newdoc.appendChi1d(rootElement):

            rootElement appendChild(docfragment):

            OutputStream out - new File0utputStream(args[2]);
            newdoc.print(out);
            out.closeO:
        }
        catch (Exception e){}
    }
}
```

На этом oraclejava.java заканчивается. Чтобы скомпилировать пример, включите в classpath путь к разборщику XML процессора XSLT Oracle, xmlparserv2.jar:

```
C:\>set classpath-.:c:\oraclexml\lib\xmlparserv2 jar
```

Затем скомпилируйте oraclejava.java, как мы это уже делали, компилятором Java javac. Для выполнения XSLT-преобразований нужно указать URL докумен-

тов, с которыми вы хотите работать (если документы локальны, можно указать URL файлов, как и раньше):

```
C:\>java oraclejava http://starpowder.com/planets.xnil
http://starpowder com/planets.xsl planets.html
```

Взаимодействие XT с Java

Процессор XT также может работать с Java. API XT спроектирован для работы с классами, определенными в Project X TR2 фирмы Sun, которые поддерживают обработку XML. Вам будет необходим файл xmLjar фирмы Sun, который можно получить, загрузив Project X TR2. Чтобы получить xmLjar, необходимо зарегистрироваться на web-узле разработчиков Sun, httpi//developer.java.sun.com, что, к счастью, бесплатно, хотя и потребует от вас заполнения изрядного количества форм.

Файл xmLjar нужен для класса com.sun.xmLtree.XmlDocument. Этот класс поддерживает XML-документы, и я начну свой новый пример, xtjava.java, с создания нового объекта XmlDocument для исходного документа, таблицы стилей XSLT и результирующего документа:

```
import java.io.IOException:
import java io OutputStream:
import java io FileOutputStream;
import org xml.sax SAXException;
import com sun.xml.tree.XmlDocument:
import com.jclark.xsl.dom.Transform;
import com.jclark.xsl.dom.TransformEngine:
import com.jclark.xsl.dom.TransformException:
import com.jclark.xsl.dom.XSLTransformEngine:

class xtjava
{
    public static void main(String[] args)
        throws IOException. SAXException. TransformException
    {

        XmlDocument XMLdoc -
            new XmlDocumentO.createXmlDocument(args[0]):
        XmlDocument XSLdoc *
            new XmlDocumentO.createXmlDocument(args[1]):
        XmlDocument newdoc - new XmlDocumentO;
        .
        .
        .
```

После этого я создаю объект XSLTranformationEngine и вызываю его метод createTransform для создания нового объекта Transform на основе таблицы стилей XSLT:

```
import java.io.IOException:
        .
        .
```

```
class xtjava
{
    public static void main(String[] args)
        throws IOException. SAXException. TransformException
    {
        XmlDocument doc - new XmlDocumentO :

        XSLTransformEngine transformEngine - new XSLTransformEnglneO;

        Transform transform - transformEngine.createTransform(XSLdoc);
        .
        .
        .
```

Затем я могу преобразовать XML-документ в объект результирующего документа следующим способом:

```
import java.io.IOException:
        .
        .
        .
class xtjava
{
    public static void main(String[] args)
        throws IOException. SAXException. TransformException
    {
        XmlDocument XMidoc -
            new XmlDocument().createXm1Document(args[0]):
        XmlDocument XSLdoc -
            new XmlDocumentC).createXmlDocument(args[l]):
        XmlDocument newdoc - new XmlDocumentO:

        XSLTransformEngine transformEngine - new XSLTransformEngineO;

        Transform transform - transformEngine.createTransform(XSLdoc):

        transform transform(XMLdoc, newdoc);
        .
        .
        .
```

Так завершается преобразование. Осталось только записать результирующий документ, newdoc, на диск, что можно сделать при помощи объекта FUeOutputStream (листинг 10.10).

Листинг 10.10. xtjava.java, взаимодействие XT с Java

```
import java.io.IOException:
import java.io.OutputStream:
import java.io.F11eOutputStream:
'import org.xml.sax SAXException:
Import com.sun.xml.tree Xm1 Document;
Import com.jclark.xsl.dom,Transform:
import com.jclark.xs!.dom.TransformEngine:
Import com.jclark.xsl.dom.TransformException;
import com.jclark.xsl.dom.XSLTransformEngine;
```

продолжение &

Листинг 10.10 (продолжение)

```
class xtjava
{
    public static void main(String[] args)
        throws IOException. SAXException. TransformException
    {

        XmlDocument XML doc =
            new Xml Document().createXmlDocument(args[0]);
        XmlDocument XSLdoc =
            new XmlDocumentO createXml Document (args [ 1 ]);
        XmlDocument newdoc - new XmlDocumentO:

        XSLTransformEngine transformEngine - new XSLTransformEngineO:

        Transform transform - transformEngine.createTransform(XSLdoc):

        transform.transform(XMLdoc. newdoc):

        OutputStream out - new File0utputStream(args[2]):
        newdoc.write(out);
        out.closeO;
    }
}
```

Чтобы запустить пример, включите xt.jar и xmLjar в classpath:

```
C:\>set classpath*-.: d:\xt\xt jar:xml.jar
```

Скомпилируйте теперь xtjava.java компилятором Java javac и запустите на выполнение следующим образом, передав URL документов XML и XSL (можно передать и URL файлов, как было показано ранее):

```
C:\>java xtjava http://www.starpowder.com/planets.xml
http://www.starpowder.com/planets.xsl planets.html
```

Теперь все сделано.

Преобразование XML в реляционную базу данных

При обсуждении дополнительных возможностей XSLT стоит упомянуть элементы расширения SQL процессора Saxon. Через драйвер Java Database Connectivity (JDBC) можно работать с реляционными базами данных. Мы уже рассмотрели преобразования из XML в XML, в простой текст, в RTF, в JavaScript и т. д. Теперь пришла очередь преобразованию XML в реляционную базу данных.

РАБОТА С ИНТЕРПРЕТИРУЕМЫМ SAXON

Обратите внимание: для связи с базами данных через JDBC нельзя использовать исполняемый файл для Windows saxon.exe. Нужно запустить реализующий Saxon класс Java com.id.saxon. StyleSheet, как я сделаю в конце примера.

В примере я передам данные planets.xml в базу данных формата Microsoft Access planets.mdb. Если вы хотите повторить пример, создайте этот файл базы

данных, в нем — таблицу planets с четырьмя текстовыми полями Name, Mass, Radius и Day, а оставшуюся часть файла оставьте пустой. В Windows я зарегистрирую этот файл базы данных в качестве источника ODBC с именем «planets» через значок Data Sources (ODBC) (Источники данных (ODBC)) в панели управления (в Windows 2000 он расположен в панели управления в папке Administrative Tools (Администрирование)). При запуске пример считает данные планет из planets.xml и добавит их в файл базы данных, planets.mdb.

Для подключения к этой базе данных через JDBC я применил элемент <sql:connect> процессора Saxon. Префикс пространства имен sql определяется в Saxon следующим образом:

```
<xsl:stylesheet
    xmlnsrsql-
    "http://1cl.com/saxon/extensions/com.1cl.saxon.sql.SQLElementFactory"
    xmlns:xsl-"http://www.w3 org/1999/XSL/Transform" version-"!.1">
```

Для фактического подключения к источнику данных planets служит элемент расширения <sql:connect> с атрибутами database, user, password и driver. Для работы через JDBC установите атрибут driver в «sun.jdbc.odbc.JdbcOdbcDriver», атрибут database — в источник данных ODBC, «jdbc:odbc:planets», а атрибуты user и password — в имя пользователя и пароль, нужные для подключения к базе данных. Здесь нам не требуется задавать имя пользователя и пароль, но я задал этим параметрам шаблонные значения, так как они требуются в большинстве приложений баз данных:

```
<xsl:stylesheet
    xmlns:sql-
    "http //icl.com/saxon/extensions/com.icl saxon.sql.SQLElementFactory"
    xmlns:xsl-"http://www w3 org/1999/XSL/Transform" version-"1.1">

    <xsl:param name-"database" select-"'jdbc:odbc:pianets'"/>
    <xsl:param name-"user"/>
    <xsl:param name-"password"/»

    <xsl:template match-'PLANETS'>
        <sql:connect database-"{Sdatabase}" user="{$user}"
        password-"{$password}" driver-"sun.jdbc.odbc.JdbcOdbcDrlver"
        xsl:extens i on-element-prefi xes-"sql"/>
        <xsl:apply-tempiates select-"PLANET"/>
    </xsl:template»
```

Затем я подключаюсь к источнику данных planets. Я хочу вставить в базу данных данные из каждого элемента <PLANET>, для чего я создаю новый шаблон, который выбирает элементы <PLANET> и вставляет данные в таблицу planets базы данных при помощи элемента Saxon <sql:in$ert>:

```
<xsl:stylesheet
    xmlns sql-
     "http://icl.com/saxon/extensions/com.icT.saxon.sql.SQLElementFactory"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"  versi on-"1.1">

     .
     .
     .
     <xsltemplate match-'PLANETS'>
          <sql:connect  database-"{{database}"  user="{$user}"
          password»"{$password}"  driver«"sun.jdbc.odbc.JdbcOdbcDri ver"
          xsl:extension-element-prefixes-"sql"/>
          <xsl:apply-templates  select-"PLANEr"/>
     </xsl:tempiate>

     <xsl:tempiate match-'PLANET'>
          <sqlrinsert  table-"planets"  xsl:extension-element-pref1xes-"sql">
          .
          .
          .
          </sql:insert>
     </xsl:template»

</xsl:stylesheet»
```

Элемент <sql:insert> вставляет в базу данных новую запись. Чтобы присвоить ее полям новые данные, используйте элемент <sql:column>, установив его атрибут name в имя столбца, в который Вы хотите записать данные, и присвоив эти данные атрибуту select:

```
<xsl:stylesheet
     xmlns:sql-
     "http://icl .com/saxon/extensions/com.icl.saxon.sql.SQLElementFactory"
     xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"  version-"!.1">

     .
     .
     .
     <xsltemplate match-'PLANET'>
          <sql :insert  table="planets"  xsl:extension-element-prefixes«"sql">
               <sql:column  name-'Name"  select-"NAME"/>
               <sql:column  name-'Hass"  select*-"MASS"/>
               <sql:column  name-"Radius"  select-"RADIUS"/»
               <sql:column  name-"Day"  select»"DAY"/>
          </sql:insert»
     </xsi:template»

</xsl .-stylesheet»
```

В идеале этого должно быть достаточно, но при работе с самым последним драйвером Access в конце операции Saxon не сбрасывает все буфера данных. Это значит, что данные о последней планете в planets.xml, Земле, не будут отправлены в базу данных. Чтобы сбросить буферы данных, я явно вызываю шаблон <PLANET> как именованный шаблон, применяя <sql:insert> с данными-заглушкой (листинг 10.11).

Листинг 10.11. Работа с реляционной базой данных

```
<xsl:stylesheet
     xmlns:sql-
```

```
"http://icl.com/saxon/extensions/com.icl.saxon.sql.SQLElementFactory"
xml ns:xsl-"http://www w3.org/1999/XSL/Transform" version-"1.1"»

<xsl:param name-"database" select*"'jdbc:odbc:pianets'"/>
<xsl:param name^"user"/>
<xsl-param name-password*/»

<xslrtemplate match-'PLANETS'>
    <sql; connect database-"{$database}" user-"{$user}"
    password-B{$password}" driver="sun.jdbc.odbc.JdbcOdbcDriver"
    xsbe*tensTpn-element-pref ixes="sql"/>
    <xsl:apply-templat6s select-"PLANET"/>
    <xsl:CaTl-template name-"wr1ter"/>
</xsl:template»

<xsl;| template match-'PLANET" name-"writer">
    <xsl:choose»
        <xsl:when test-"NAME">
            <sql:insert table-'planets"
            xsl:extensi on-element-prefi xes-"sql">
                <sql:column name-"Name" select-"NAME"/>
                <sql:column name-"Mass" select-"MASS"/>
                <sql:column name-"Radius" select-"RADIUS"/>
                <sql:column name-"Day" select-"DAY"/>
            </sql:insert»
        </xsl:when>
        <xsl:otherwise»
            <sql:insert table-"planets"
            xsl:extension-element-prefixes-"sql"»
                <sql:column name-"Name" select-"' '"/»
                <sql:column name-"Mass" select-"' '"/>
                <sql:column name-"Radius" select-"' '"/>
                <sql:column name-"Day" select-"' '"/»
            </sql:insert»
        </xsl:otherwise»
    </xsl:choose»
</xsl:tempiate»

</xsl:stylesheet»
```

Эта таблица стилей правильно добавляет три записи в базу данных planets, mdb: по одной новой записи для каждой из планет. Как я говорил, здесь нельзя использовать готовый исполняемый файл saxon.exe, нужно применять класс Java com.icl.saxon.StyleSheet. Сначала нужно включить в Classpath файл saxon.jar:

```
C:\»set classpath-.;c:\saxon\saxon.jar
```

После этого я могу передать в класс com.icl.saxon.StyleSheet файл planets.xml и таблицу стилей из листинга 10.11:

```
C:\»java com.icl.saxon.StyleSheet planets.xml saxonsql.xsl
```

Теперь все сделано — данные планет вставлены в planets.mdb. Результаты применения этой таблицы стилей показаны на рис. 10.4, где я открыл planets.mdb в Microsoft Access. Таким образом, мы рассмотрели преобразование из XML в реляционную базу данных.

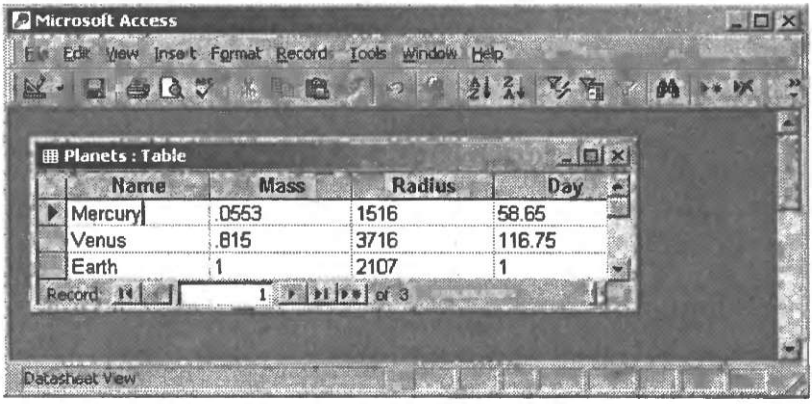


Рис. 10.4. Применение расширений SQL Saxon

Еще один аспект XSLT, в котором вам потребуется программирование, это поддержка XSLT на серверах. В качестве демонстрации далее я преобразую planets.xml при помощи planets.xsl с использованием активных серверных страниц (Active Server Pages, ASP) фирмы Microsoft, серверных страниц Java (Java Server Pages, JSP) фирмы Sun и сервлетов (servlet) Java, выполняющихся на web-сереерах и возвращающих результирующий документ браузеру пользователя.

За недостатком места мы не можем подробно рассмотреть эти технологии, и если вы с ними не знакомы, вы можете получить дополнительную информацию в Интернете (как всегда, URL могут измениться):

- ASP. http://msdn.microsoft.eom/workshop/c-frame.htm#/workshop/server/Default.asp (руководство и документация по ASP фирмы Microsoft);
- JSP. http://java.sun.com/products/jsp/ (главная страница Sun, посвященная JSP);
- Сервлеты. http://java.sun.com/products/servlet/ (главная страница Sun, посвященная сервлетам).

Применение XSLT с активными серверными страницами

Активные серверные страницы (Active Server Pages, ASP) выполняются на серверах Microsoft Windows NT или 2000, поэтому в ближайшем примере для преобразования ptanets.xmL при помощи planets.xsl и возврата результата в виде документа HTML я воспользуюсь процессором Microsoft MSXML.

Это все то же преобразование, которое нам уже много раз встречалось, — создание HTML-таблицы данных о планетах — но на этот раз преобразование будет выполнено на web-сервере, который затем отправит его пользователю. В начале сценария ASP я установил тип содержимого MIME результирующего документа в «text/html», поэтому он будет трактоваться как HTML:

```
<X@LANGUAGE"VBScript"%>
<%

    Response.ContentType     "text/html"
```

Затем, во многом аналогично предыдущим примерам в этой главе с JavaScript, я создаю два объекта документа MSXML, один для документа XML, а другой — для документа XSL:

```
SLANGUAGE-" t/BScript"3>

<%
    Response.ContentType = "text/html"

    Dim docXML
    Dim docXSL

    Set docXML « Server.CreateObject("MSXML2.DOMDocument.3,0")
    Set docXSL - Server.CreateObject("MSXML2.DOMDocument.3.0")
```

Процедура загрузки этих документов на сервере во многом похожа на процедуру с использованием JavaScript, за тем Исключением, что здесь правильные пути файлов получаются При помощи метода **MapPath** объекта Server. В данном случае я поместил planets.xml и **planets.xsl** в тот же каталог, что и сценарий ASP, поэтому документы загружаются так:

```
<t@LANGUAGE"-VBSc r i pt"S>

<%
    Response ContentType = "text/html"

    Dim docXML
    Dim docXSL

    Set docXML = Server.CreateObject("MSXML2.DOMDocument.3.0")
    Set docXSL - Server.CreateObject("MSXMI2.DOMDocument.3  0")

    docXML.ValidateOnParse =, True
    docXSL.ValidateOnParse = True

    docXML.load  Server.MapPath("pianets.xml")
    docXSL.load  Server.MapPath("pianets.xsl")


    /
```

Теперь нам осталось только применить метод transformNode (как в примере (с JavaScript раньше) для выполнения XSLT-преобразования и вывода результатов (листинг 10.12). А

Листинг 10.12. XSLT на сервере с применением ASP

```
<X@LANGUAGE="VBScript'S>
<%

    ResponseXontentType = "text/html"

    Dim docXMl
    Dim docXSL

    Set docXML - Server.CreateObject("MSXML2.DOMDocument.3.0")
    Set docXSL - Server CreateObject("MSXML2.DOMDocument.3.0")
           l                                                  /

    docXML.ValidateOnParse - True
    docXSL.ValidateOnParse - True

    docXML.load Server. MapPathC pi anets. xml")                    \
    docXSL.1oad Server.MapPath("pianets.xs1")

    strOutput - docXML.transformNode(docXSL)

    Response.Write strOutput
%>
```

Результат этого преобразовании ASP показан на рис. 10.5. Теперь вы научились выполнению XSLT-преобразований на web-сервeрах.

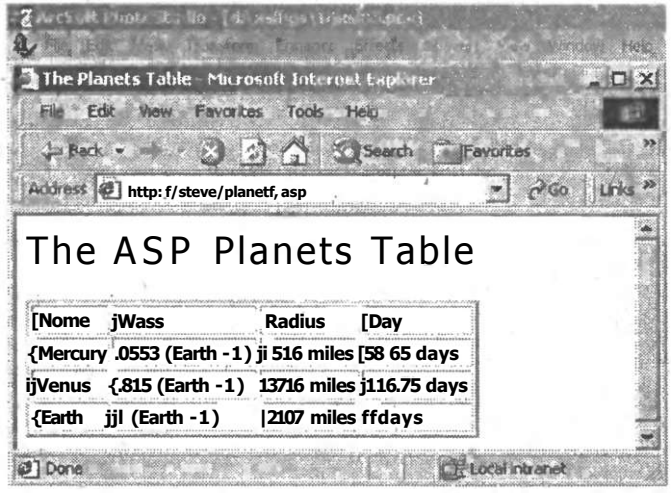


Рис. 10.5. Применение XSLT с ASP

Нужно сделать одно замечание: при выполнении XSLT-преобразований на сервере можно подумать об оптимизации ваших преобразований в зависимости от клиента, так как необязательно генерировать для карманного компьютера тот же экран, что и для браузера настольного компьютера. Например, можно настроить ответ вашего серверного сценария ASP (где я проверяю значение серверной переменной http_user_agent чтобы определить, использует ли клиент Internet Explorer) следующим образом:

```
<fc@LANGUAGE="VBScript"S>
<%

Response ContentType * "text/html"

If instr(request,servervariables("http_user_agent"). "MSIE")- - 0 then
    Response.Wrlte "Sorry, not optimized for your device."
    Response.End
End If
```

ОТДЕЛЕНИЕ ДАННЫХ ОТ ИХ ПРЕДСТАВЛЕНИЯ

Одной из преобладающих тенденций в последнее время является стремление отделить данные от представляющего их кода. Рассмотренный пример демонстрирует преимущества такого подхода: единственный набор данных можно преобразовывать в рабочем порядке для различных браузеров.

Применение XSLT с Java Server Pages

Для работы с активными серверными страницами я использовал процессор MSXML фирмы Microsoft, но серверные страницы Java (Java Server Pages, JSP) не обязаны выполняться на серверах на платформе Windows, поэтому с JSP я не использую MSXML. Для выполнения XSLT-преобразования и отправки результатов в браузер клиента я воспользуюсь API Процессора Xalan для работы с Java.

Например, при помощи Xalan можно создать planets.html на сервере как временный документ (этот код предполагает, что planets.xml и planets.xsl расположены в том же каталоге, что и сценарий JSP) таким образом:

```
<%& page errorPage-"error.jsp" language-"java"
    contentType-"text/html" import-"org.apache,xalan.xs11..*;java.io.*" %>
    <%

        try
        {
            XSLTProcessor processor - XSLTProcessorFactory.getProcessorO:
            processor.process(new XSLTInputSource("pianets.xml"),
                new XSLTInputSource("planets.xsl").
                new XSLTResultTarget("pianets.html"));
        }
        catch(Except1on e) {}
```

Теперь все, что нужно сделать, — это открыть документ и отправить его обратно клиенту (листинг 10.13).

Листинг 10.13. XSLT на сервере с применением JSP

```
<%Q page errorPage-'error.jsp" language-"java"
    contentType-"text/html" import-"org.apache.xa1 an.xslt.* java.io.*" %>

    <%
```

продолжение &

Листинг 10.13 (продолжение)

```
        try
        {
            XSLTProcessor processor - XSLTProcessorFactory.getProcessorO:
            processor.process(new XSLTInputSource("pianets.xml").
                new XSLTInputSource("pianets.xsl").
                new XSLTResultTarget("planets.html"));
        }
        catch(Exception e) {}

        FileReader filereader - new FileReaderCplanets.html");
        BufferedReader bufferedreader - new BufferedReader(filereader);
        String instring:                                                    «s

        while((instring - bufferedreader.readLineO) I- null) { *>
            <X~ instring *>
    <* }

    filereader.closeO:
    pw.closeO:

%>
```

Теперь все готово. Результаты применения этого сценария JSP показаны на рис. 10.6.

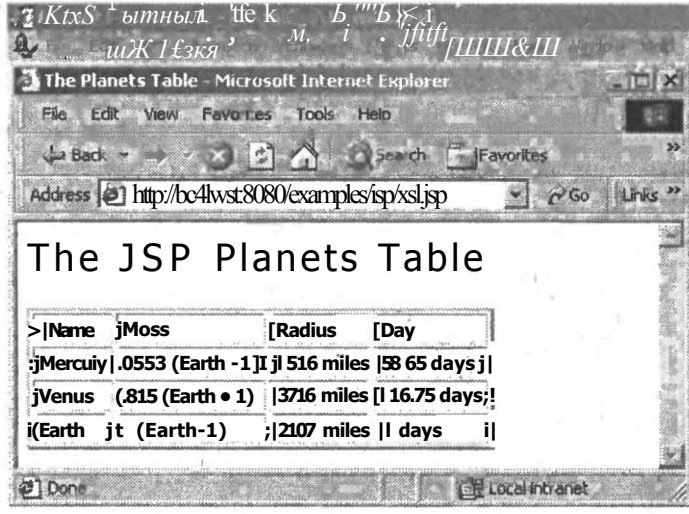


Рис. 10.6. Применение XSLT с JSP

Применение XSLT с сервлетами Java

Выполнять XSLT-преобразования на сервере можно также при помощи сервлетов (servlet) Java. Хотя многие процессоры XSLT предоставляют свои собственные версии сервлетов, я считаю, что гораздо проще самостоятельно выполнить

преобразование при помощи Xalan или другого процессора XSLT и затем обработать результирующий документ и отправить его обратно клиенту.

В следующем примере я при помощи Xalan преобразую planets.xml во временный файл planets.html, применяя planets.xsl в сервлете:

```
import java.net.*:
import java.sql.*:
import java awt.*:
import java awt.event *;
import java.io.*:
import javax.servlet.*:
import org apache xalan.xslt.*:

public class xslservlet extends GenericServlet
{

    public void service(ServletRequest request. ServletResponse response)
    {   throws ServletException. IOException

        try
        {
            XSLTProcessor processor - XSLTProcessorFactory.getProcessorO;
            processor.process(new XSLTInputSource("planets.xml").
                new XSLTInputSource("pianets.xsl").
                new XSLTResultTargetf"pianets.html"));
        }
        catch(Exception e) {}
```

Все, что осталось, — отправить HTML-документ обратно клиенту (листинг 10.14).

Листинг 10.14. Применение XSLT на сервере с сервлетами Java

```
import java.net.*:
import java.sql.*:
import java.awt.*:
import java.awt.event.*:
import java.io *:
import javax.servlet.*:
import org.apache.xalan.xslt.*:

public class xslservlet extends GenericServlet
{

    public void service(ServletRequest request. ServletResponse response)
        throws ServletException. IOException
    {
        response.setContentType("text/html"):

        PrintWriter pw - response.getWriterO;
        try
        {
            XSLTProcessor processor - XSLTProcessorFactory getProcessorO:
            processor.process(new XSLTInputSourceC'planets.xml"),
                              •>
```

продолжение.

Листинг 10Д4 (продолжение)

```
                new XSLTInputSource("pianets.xsl").
                new XSLTResultTarget("pianets.html")>;
            }
            catch(Exception e) {}

            FileReader filereader - new FileReaderCplanets.html");
            BufferedReader bufferedreader • new BufferedReader(filereader);
            String Instring;

            whi1e((instring - bufferedreader.readLineO) != null) {
                pw.println(instring):
            }

            f 11 ereader. cl oseO;
            pw.closeO;
    }
}
```

Теперь все сделано, результат показан на рис. 10.7 — HTML-документ, переданный из сервлета.

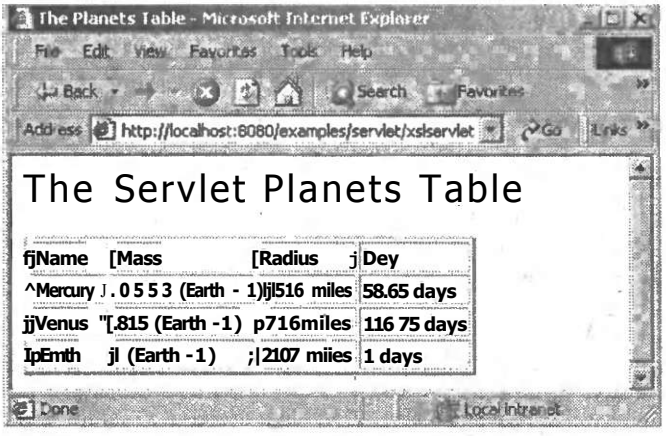


Рис. 10.7. Применение XSLT с сервлетами Java

На этом мы заканчиваем изучение работы с API процессоров XSLT в программном коде. Мы рассмотрели весьма объемный материал: как взаимодействовать с API процессоров XSLT из JavaScript и Java для процессоров MSXML, Xalan, XT, Saxon и Oracle, а также примеры выполнения XSLT-преобразований на web-серВерах при помощи ASP, JSP и сервлетов Java.

Применение XSLT для создания документов XSL-FO

11

Мы уже рассмотрели XSLT-преобразования из XML в XML, HTML, XHTML, RTF, простой текст, JavaScript и даже в реляционные базы данных. В этой и следующей главах вы познакомитесь с новым преобразованием, для которого и создавались XSLT: преобразование XML в документы, которые используют форматирующие объекты, XSL (документы XSL-FO).

Такое преобразование являлось первоочередной задачей XSLT. Как вы знаете, XSLT вырос из спецификации XSL для упрощения работы с форматирующими объектами (formatting objects). Имейте в виду, что XSL-FO гораздо объемнее XSLT — здесь я смогу представить только введение в предмет.

Форматирование XSL

Как обсуждалось в главе 1, XSLT и XSL-FO составляют спецификацию XSL. В данный момент версия 1.0 этой спецификации обладает статусом рекомендации-кандидата, с последней версией вы сможете познакомиться на http://www.w3.org/TR/xsl.

Форматирующая часть XSL, XSL-FO, позволяет устанавливать точное форматирование для всего документа, вплоть до каждого миллиметра. Например, вы можете выбрать шрифт, его размер, расположение текста на странице, цвет текста, режим подчеркивания и т. д. Работа с XSL-FO-во многом похожа на создание документов RTF, но с некоторыми дополнениями: например, в XSL-FO можно создавать ссылки для навигации по документам.

Если вы когда-либо видели огромное количество кодов, используемых в RTF для форматирования документов, вы поймете, что XSL-FO представляет собой сложную и очень большую спецификацию. (Вот почему появились XSLT — чтобы упростить работу с XSL-FO.) Таким образом, на отведенном месте я смогу только дать введение в эту тему. Однако пока что еще нет программ, полностью реали-

зующих всю спецификацию, поэтому сравнительно небольшое число людей пользуются возможностями XSL-FO, выходящими за рамки материала этой и следующей глав. В следующем списке перечислен ряд доступных процессоров XSL-FO:

- fop. http://xml.apache.org/fop. Этот популярный процессор преобразует XSL-FO в PDF. Разработан Джеймсом Таубером в Apache Software Foundation (http://xml.apache.org/);
- PassiveTeX. http://users.ox.ac.uk/~rahtz/passivetex/. Этот процессор фактически представляет собой библиотеку макросов TeX, обеспечивающих среду для тестирования XSL-FO;
- REXP. http://www.esng.dibe.unige.it/REXP. Эта ранняя версия процессора XSL-FO. Создает файлы PDF;
- Unicorn, http://www.unicorn-enterprises.com. Этот процессор (называемый UFO) распространяется бесплатно и работает в Windows NT 4.0 и Windows 95;
- XEP. http://www.renderx.com/F02PDF.html. Этот процессор, обычно называемый FOP2PDF, преобразует документы XSL-FO в формат PDF;
- XSL Formatter, http://www.antennahouse.com/xslformatter.html Процессор XSL-FO для Windows. Доступна бесплатная копия с ограниченным сроком действия (evaluation version);

Я лично предпочитаю fop, который сегодня является самым популярным процессором XSL-FO. В этой и следующей главах я буду работать с последней версией fop, 0.17. Apache Software Foundation, группа, которая создала и Xalan, осуществляет сопровождение процессора fop, преобразующего документы XSL-FO в документы в формате Portable Data Format (PDF), для просмотра которых служит Acrobat reader фирмы Adobe. Acrobat reader можно бесплатно загрузить с http://www.adobe.com/products/acrobat/readermain.html. Последнюю версию самого fop можно бесплатно загрузить с http://xml.apache.org/fop (на сегодняшний день достаточно только щелкнуть на ссылку для загрузки на этой странице).

Процессор fop работает с документами XSL-FO, и в этой и следующей главах я воспользуюсь XSLT для преобразования документов XML в формат XSL-FO и затем применю fop для преобразования документов XSL-FO в формат PDF. (Применение XSL-FO не означает, что обязательно будут создаваться документы PDF — существуют и другие форматы, — XSL-FO просто позволяет вам форматировать данные, а то, как эти форматированные данные будут представлены, зависит от используемого приложения.) Для работы с XSL-FO необходимо понимать, как работать с существующими форматирующими объектами XSL.

Форматирующие объекты XSL-FO

В документах XSLT мы работали с такими элементами, как <xsl:stylesheet>, <xsl:output> и т. д.:

```
<?xml version-"1.0" encoding="UTF-8"?>
<xsl:stylesheet version-"!.0"
```

```
xralhs:xsl-"http://www.w3.org/1999/XSL/Transform">
    <xsl output method-"xml"/>
    <xsl:tempiate match-"*"»
        <xsl: COPY>
            <xsl:apply-templates/>
            :
            :
            :
```

Документ XSL-FO написан во многом таким же способом, но вместо таких элементов XSLT, как <xsl:stylesheet>, применяются элементы, основанные на форматирующих объектах XSL-FO. Существует всего 56 таких форматирующих объектов — например, объект корня, создающий корневой узел документа XSL-FO, или объект блока, создающий область блока (прямоугольную область отображения, похожую на область, которую тег заголовка <H1> создает в документе HTML).

У форматирующих объектов есть свое собственное пространство имен, «http://www.w3.org/1999/XSL/Format», и в качестве префикса этого пространства имен практически всегда используется fo, сокращение для formatting objects. (Это соглашение было принято потому, что данный префикс пространства имен используется в спецификации XSL.) При написании документов XSL-FO используются элементы, соответствующие различным объектам XSL-FO — например, <fo:root> для корневого элемента, <fo:block> для создания блока отображения и т. д. Вот как можно начать документ XSL-FO — заметьте, что я объявляю префикс пространства имен «fo», соответствующий пространству имен XSL-FO:

```
<?xml version-"1.0" encoding-"UTF-8"?>
<fo:root xmlns:fo-"http://www.w3 org/1999/XSL/Format">

    <fo:1ayout-master-set>
        <fo:simple-page-master margin-right="20mm" margin-left="20mm"
            margin-bottom-"10mm" margin-top»"10mm" page-width="300mm"
            page-height-"400mm" master-name="page">

            <fo:region-body margin-right«"Omm" margin-left="0mm"
                margin-bottom-"1Omm" margin-top-"Omm"/>
        :
        :
        :
```

Обратите внимание: документы; XSL-FO являются документами XML, поэтому везде можно использовать стандартные комментарии XML, <!— такие как этот -->. Отметьте также, что элементы <fo:simple-page-master> и <fo:region-body> обладают такими атрибутами, как margin-right и page-height. В XSL-FO такие атрибуты называются свойствами, и их весьма много — на момент написания книги около 240. Например, вот как можно создать блок XSL-FO — прямоугольную область отображения — в котором выводится текст «Welcome to XSL formatting» (Добро пожаловать в форматирование XSL) шрифтом sans-serif размером 36 пунктов. Я использую форматирующий объект <fo:block> и свойства XSL-FO font-family — line-height (задающее высоту блока) и font-size:

```
<fo:block font-family-"sans-serif" line-height-"48pt" font-size-"36pt">
    Welcome to XSL formatting
</f0:bl0Ck>
```

В этом разделе мы рассмотрим объекты XSL-FO, а **в** следующем — свойства XSL-FO для работы с ними. При помощи объектов и свойств XSL-FO вы можете создавать документы XSL-FO, которым обычно дается расширение .fo. Процессор fop преобразует такие документы в формат .pdf.

Все форматирующие объекты XSL-FO с описанием перечислены на www.w3.org/TR/xsl/slice6.html. Во время создания книги существуют такие объекты (с работой многих из них мы познакомимся в этой и следующей главах):

- <fo:bidi-override>. Перекрывает двунаправленный алгоритм Unicode (этот алгоритм используется в документах на нескольких языках);
- <fo:block>. Создает блок отображения для формирования прямоугольных областей, используемых при выводе абзацев, заголовков, подписей к рисункам и таблицам и т. д.;
- <fo:block-container>. Создает контейнер блоков, который затем можно помещать в любом месте;
- <fo:character>. Представляет единственный символ;
- <fo:color-profile>. Создает цветовой профиль для таблицы стилей, с которым потом можно работать при помощи цветовых функций (подробности см. на www.w3.org^R/xsl/slice5.html#expr-color-functior»s);
- <fo:conditional-page-master-reference>. Задает шаблон страницы (page-master)г который следует использовать при выполнении указанных условий;
- <fo:declarations>. Создает глобальные объявления;
- <fo:external-graphic>. Добавляет в документ графику (графические данные располагаются за пределами результирующего документа, но могут быть интегрированы в документы PDF такими процессорами, как fop);
- <fo:float>. Указывает, что определенное содержимое будет форматироваться в отдельной, незакрепленной области в начале страницы или сдвинутой к одной из сторон;
- <fo:flow>. Обрабатывает поток текста, отображаемый в документе. Накладывает блокировку на «поток» встроенных областей, когда они заменяются в документе;
- <fo:footnote>. Задает ссылку на сноску, а также саму связанную с ней сноску;
- <fo:footnote-body>. Определяет содержимое сноски;
- <fo:initial-property-set>. Задает форматирующие свойства для начальной строки блока;
- <fo:inline*. Создает встроенную область (inline area). Встроенные области часто применяются для форматирования определенной части текста в блоке;
- <fo:inline-container>. Создает контейнер для встроенных объектов, позволяя обрабатывать их все вместе;
- <fo:instream-foreign-object>. Используется для ^ставки в документ встроенной графики или другого объекта;

- <fo:layout-master-set>. Задает оболочку для всех шаблонов, используемых в документе;
- <fo:leader>. Создает строку из повторяющегося символа или образец из символов для разделения двух текстовых форматирующих объектов;
- <fo:list-block>. Создает форматированный список, — с ним мы познакомимся в следующей главе;
- <fo:list-item>. Задает метку и тело элемента списка;
- <fo:list-item-body>. Задает содержимое тела элемента списка;
- <fo:list-item-label>. Задает содержимое метки элемента списка;
- <fo:marker>. Применяется вместе с <fo:retrieve-marker> для создания колонтитулов или сносок по ходу работы;
- <fo:multi-case>. Задает объекты, которые показывает или скрывает родительский элемент <fo:multi-switch>;
- <fo:multi-prioperties>. Позволяет переключаться между двумя или более наборами .свойств;
- <fo:multi-property-set>. Задает альтернативный набор свойств;
- <fo:multi-switch>. Переключает между двумя или более поддеревьями форматирующих объектов;
- <fo:multi-toggle>. Используется внутри элемента <fo:multi-case> для переключения на другой <fo:multi-case>;
- <fo:page-numben>. Указывает номер текущей страницы;
- <fo:page-number-citation>. Ссылается на номер страницы, содержащей заданный форматирующий объект;
- <fo:page-sequence>. Определяет способ создания последовательности страниц в документе;
- <fo:page-sequence-master>. Содержит последовательности шаблонов страниц, используемых для создания последовательностей страниц;
- <fo:region-after>. Обозначает область, расположенную после области <fo: region-body>;
- <fo:region-before>. Обозначает область перед областью <fo:region-body>;
- <fo:region-body>. Обозначает область в центре <fo:simple-page-master>;
- <fo:region-end>. Обозначает область в конце области <fo:region-body>;
- <fo:region-start>. Обозначает область, начинающую область <fo:region-body>;
- <fo:repeatable-page-master-alternatives>. Задает повторяющиеся экземпляры набора альтернативных шаблонов страниц;
- <fo:repeatable-page-master-reference>. Обозначает подпоследовательность повторяющихся экземпляров единственного шаблона страницы;
- <fo:retrieve-marker>. Используется вместе с <fo:marker> для создания колонтитулов или сносок в рабочем режиме;

- <fo:root>. Узел документа, отформатированного при помощи XSL;
- <fo:simple-link>. Задает начальное место в простой ссылке;
- <fo:simple-page-master>. Определяет форму страницы, которая может быть разделена на области (до пяти областей);
- <fo:single-page-master-reference>. Определяет подпоследовательность, образованную из единственного экземпляра единственного шаблона страницы;
- <fo:static-content>. Содержит последовательность форматирующих объектов, которые будут отображены в одной области или повторены в областях в одной или более страницах в последовательности страниц. Чаще всего применяется для повторяющихся колонтитулов и сносок;
- <fo:table>. Создает таблицу. Содержит такие элементы, как <fo:table-column> и <fo:table-body>;
- <fo:table-and-caption>. Содержит данные и заголовок таблицы;
- <fo:table-body>. Определяет содержимое тела таблицы. Содержит такие элементы, как <fo:table-row>;
- <fo:table-caption>. Задает форматирующие объекты уровня блока, в которых содержится заголовок таблицы;
- <fo:table-cell>. Помещает данные в ячейки таблицы;
- <fo:table-columri>. Задает свойства ячеек в одном столбце таблицы;
- <fo:table-footer>. Определяет содержимое нижнего колонтитула таблицы;
- <fo:table-header>. Определяет содержимое верхнего колонтитула таблицы;
- <fo:table-row>. Связывает ячейки таблицы в строки. Содержит элементы <fo:table-cell>;
- <fo:title>. Задает заголовок документа;
- <fo:wrapper>. Задает свойства для группы форматирующих объектов;

Каждый из этих элементов также поддерживает одно или несколько форматирующих свойств XSL-FO.

Форматирующие свойства XSL-FO

Форматирующие свойства XSL-FO настраивают действия перечисленных в предыдущем разделе форматирующих объектов; работайте с ними в документах XSL-FO так же, как с атрибутами элементов. Например, вот как я установил семейство (family), размер (size) и насыщенность (weight;) шрифта для текста в блоке:

```
<fp block color«="blue" font-family-"Times"
    font-si ze="36pt" font-weight="bold">
    Hello from XSL-FO!
</fo Ьlock>
```

Полный список свойств XSL-FO и их описание можно найти по адресу www.w3.org/TR/xsl/slice7.html. **Многие свойства** XSL-FO унаследованы из спецификации для каскадных таблиц стилей версии 2, CSS2, которую вы можете найти по адресу www.w3.org/TR/REC-CSS2.

В следующем списке перечислен ряд наиболее распространенных свойств XSL-FO. Полный список свойств (весьма длинный) приведен в Приложении Б. В списке используется следующий синтаксис, соответствующий спецификации XSL-FO:

- о обозначает единицы значений (см. таблицу 11.1) или свойства (например, <color>), которые уже были определены;
- | означает вариант выбора, использовать можно только один из них;
- || означает вариант выбора, можно использовать один или несколько вариантов в любом порядке;
- [] означает групповые инструкции, вычисляемые во многом так же, как и математические выражения;
- * означает, что предыдущий терм может встречаться ноль или более раз;
- + означает, что предыдущий терм может встречаться один или более раз;
- ? означает, что предыдущий терм необязателен;
- {} окружают пары чисел, задающие минимальное и максимальное число раз встречи терма (как в {1, 4});

Во многих свойствах XSL-FO определен определенный формат для присваиваемых им значений, как, например <color> или <angle>, — эти форматы и их описание перечислены в табл. 11.1.

Таблица 11.1. Форматы спецификации XSL-FO

Формат	Описание
<absolute-size>	Абсолютные размеры шрифта"; могут быть xx-small, x-small, small (маленький), medium (средний), large (крупный), x-large, xx-large
<angle>	Углы; могут измеряться в deg (градусах), grad (градах) или rad (радианах)
idctlpar<border-style>	Граница прямоугольника, может принимать значения попе (нет), dotted (пунктиром), dashed (штрихами), solid (непрерывная), double (двойная), groove (канавкой), ridge (с толстой кромкой), inset (вложенная) или outset (вынесенная)
<border-width>	Устанавливает ширину границы; может принимать значения thin (тонкая), medium (средняя), thick (толстая) или конкретного размера
<color>	Цвет; может быть задан предопределенным значением цвета или тройным цветовым значением RGB, как в HTML, — например, «FFFFFF» для белого цвета
<country-language>	Устанавливается в значение <language> (см. <ianguage>)
<family-name>	Название семейства шрифтов — такого, как Arial, Times New Roman или Courier

продолжение &

Таблица 11.1 (продолжение)

Формат	Описание
<frequency>	Значения частоты; единицами измерения могут быть Hz (Гц) или KHz (кГц)
<generic-family>	Обобщенные названия шрифтов, применяемых в качестве последней попытки, если браузер не может найти определенный шрифт. Например, serif (браузер должен выбрать шрифт serif), sans-serif (браузер должен выбрать семейство sans-serif) и monospace (браузер должен выбрать моноширинный шрифт)
<generic-voice>	Звук голоса; может быть male (мужской), female (женский) или child (детский)
<integer>	Стандартные целые значения
<keep>	Контекстно-зависимое свойство; принимает значение Auto, для того чтобы не удерживать вместе ограничения, или Always для жесткого удержания ограничений вместе. Дополнительная информация приведена в спецификации XSL
<language>	Определитель языка, соответствующий спецификации RFC1766 (которую можно найти по адресу www.w3.org/TR/xsl/sliceD.html#RFC1766)
<length>	Длина; может начинаться с + или -, за которыми следует число.' Число может содержать десятичную точку, за которой следует идентификатор единиц, принимающий значения em (размер шрифта соответствующего шрифта), ex (x-height, высота шрифта), px (пикселов, pixel, определяется устройством просмотра), pt (пунктов, point, 1/72 дюйма), In (дюймоЬ, Inch), cm (сантиметров), mm (миллиметров) или pc (пик, pica, 1/6 дюйма)
< length-bp-ip-direction >	Задает расстояние, разделяющее смежные границы ячейки в направлении размещения строк. Дополнительная информация приведена на www.w3.org/TR/xsl/ slice7.html
< length-conditional >	Составное значение, задающее ширину и возможный тип передней границы. Дополнительная информация приведена на www.w3.org/TR/ xsl/slice7.html#pr-section
<length-range>	Определяет диапазон длины, как определено в спецификации XSL
<margin-width>	Может принимать значения <length>, <percentage> или auto. Работа значения auto зависит от контекста, подробности приведены на www. w3.org^R/REC-CSS2/visudet.html#Computing_wldths_and_margins
<number>	Число, может включать знак и десятичную точку
<padding-width>	Устанавливается в значение <length>
<percentage>	Число, которое может содержать знак, за которым следует знак процента <%)
<relative-size>	Размер шрифта относительно родительского элемента, может быть Larger (больше) или Smaller (меньше)
<shape>	Пока может задавать только прямоугольник (shape — фигура), как, например: rect(<top> <rlght> <bottom> <left>)
<space>	Задает минимальное, оптимальное и максимальное значения для пробела. Подробности см. на www.W3.org/TR/xsl/ s!ice4.html#spacecond

Формат	Описание
<spedfic-volce>	Задает определенный голос. Подробности см. на vvvw.w3.org/TR/REC-CSS2/aural.html#propdef-voice-family
<time>	Единицы времени, указанные как число, за которым сразу же следует ms (миллисекунды) или $ (секунды)
<uri-specification>	Uniform Resource Indicator (URI); web-адрес элемента страницы — такого, как рисунок

Примите также к сведению, что свойства XSL-FO, как и свойства CSS2, могут принимать значение «inherit», что означает, что значение свойства должно быть унаследовано от его родительского элемента.

В следующем списке перечислены наиболее распространенные свойства XSL-FO. Полный список приведен в Приложении Б.

- absolute-position. Определяет, является ли позиция элемента абсолютной. Устанавливается в auto | absolute | fixed | inherit;
- background. Свойство для быстрой установки сразу всех отдельных свойств заднего фона (background-color, background-image, background-repeat, background-attachment и background-position). Устанавливается в <background-color> || •«background- image> || <background-repeat> || <background-attachment> || <background-position>]] | inherit;
- background-attachment. Определяет, будет ли прокручиваться задний фон. Устанавливается в scroll | fixed | inherit;
- background-color. Задает цвет заднего фона элемента. Устанавливается в <color> | transparent | inherit;
- background-image. Задает рисунок заднего фона элемента. Устанавливается в <uri-specification> J none | inherit;
- background-repeat. Определяет, может ли рисунок заднего фона располагаться мозаикой, и если да, то каким образом. Устанавливается в repeat | (repeat-x | repeat-y | no-repeat | inherit;
- border. Свойство для быстрой установки одинаковых ширины, цвета и стиля всех четырех границ (верхней, нижней, левой и правой) прямоугольника. Устанавливается в [<border-width> || <border-style> |] <color>] {inherit;
- border-after-color. Задает цвет задней границы области. Устанавливается в <color> | inherit;
- border-after-style. Задает стиль задней границы. Устанавливается в <border-style> | inherit;
- border-after-width. Задает ширину задней границы. Устанавливается в cborder-width> | <length-conditional> | inherit;
- border-before-color. Задает цвет передней границы. Устанавливается в <color> | inherit;
- border-before-style. Задает стиль передней границы. Устанавливается в <border-style> [inherit;

- border-before-width. Задает ширину передней границы. Устанавливается в <border-width> | <length-conditional> | inherit;
- border-bottom. Свойство для быстрой установки ширины, стиля и цвета нижней границы области блока или встроенной области. Устанавливается в [<border-top-width> || <border-style> || <color>] | inherit;
- border-bottom-color. Задает цвет нижней границы. Устанавливается в <color> | inherit;
- border-bottom-style. Задает стиль нижней границы. Устанавливается в <border-style> | inherit;
- border-bottom-width. Задает ширину нижней границы. Устанавливается в <border-width> | inherit;
- border-collapse. Задает модель границы таблицы. Устанавливается в collapse | separate | inherit;
- border-color. Задает цвет сразу всех четырех границ. Устанавливается в <color>{1,4} | transparent | inherit;
- border-end-color. Задает цвет конечной границы. Устанавливается в <color> J inherit;
- border-end-style. Задает стиль конечной границы. Устанавливается в <border-style> | inherit;
- border-end-width. Задает ширину конечной границы. Устанавливается в <border-width> | «length-conditional» | inherit;
- border-left. Свойство для быстрой установки одновременно ширины, стиля и Цвета левой границы. Устанавливается в [<border-top-width> || «border-style> || <color>] | inherit;
- border-left-color. Задает цвет левой границы. Устанавливается в <color> | inherit;
- border-left-style. Задает стиль левой границы. Устанавливается в «border-style> | inherit;
- border-left-width. Задает ширину левой границы. Устанавливается в «border-width» | inherit;
- border-right. Свойство для быстрой установки одновременно ширины, стиля и цвета правой границы. Устанавливается **в** [<border-top-width> || «border-style» || «color»] | inherit;
- border-right-color. Задает цвет правой границы. Устанавливается в <color> | inherit;
- border-right-style. Задает стиль правой границы. Устанавливается в «border-style» | inherit;
- border-right-width. Задает ширину правой границы. Устанавливается в <border-width> | inherit;

- border-spacing. Задает расстояние между смежными границами ячейки. Устанавливается в <length> «length»? | inherit;
- border-start-color. Задает цвет начальной границы. Устанавливается в <color>;
- border-start-style. Задает стиль начальной границы. Устанавливается в <border-style> | inherit;
- border-start-width. Задает ширину начальной границы. Устанавливается в <border-width> | <length-conditional> | inherit;
- border-style. Задает стиль четырех границ. Устанавливается в «border-style» {1,4} J inherit;
- border-top. Свойство для быстрой установки одновременно ширины, стиля и цвета верхней границы области блока или встроенной области. Устанавливается в [<border-top-width> || <border-style> || <color>] | inherit;
- border-top-color. Задает цвет верхней границы. Устанавливается в «color» | inherit;
- border-top-style. Задает стиль линии границы прямоугольника (непрерывная, двойная, штриховая и т. д.);
- border-top-width. Задает ширину верхней границы. Устанавливается в «border-width» J inherit;
- border-width. Свойство для быстрой установки одновременно свойств border-top-width, border-right-width, border-bottom-width и border-left-width. Устанавливается в <border-width»{1,4} | inherit;
- bottom. Определяет, насколько нижний край содержимого прямоугольника (box) смещен ввер^ от нижней границы содержащего прямоугольник блока. Устанавливается в <length> | «percentage» | auto | inherit;
- character. Задает вставляемый символ Unicode. Устанавливается в «character»;
- color. Задает цвет переднего фона текста элемента. Устанавливается в «color» | inherit;
- column-count Задает число столбцов в области. Устанавливается в «number» J inherit;
- column-number. Устанавливает номер столбца для ячеек таблицы. Устанавливается в «number»;
- column-width. Задает" ширину такого объекта, как внешняя графика. Устанавливается в auto | scale-to-fit | «length» | «percentage» | inherit;
- float. Определяет, будет ли прямоугольник прижат влево, вправо или вообще не будет прижат. Устанавливается в before | start | end | left | right | none | inherit;
- flow-name. Задает имя потока (flow). Устанавливается в <пате»;
- font. Свойство ДЛЯ быстрого задания одновременно свойств font-style, font-variant, font-weight, font-size, line-height и font-family. Устанавливается в [[«font-style» || «font-variant» || «font-weight»]? «font-size» [/ «line-height»]? «font- family»] J caption | icon | menu | message-box | small-caption | status-bar | inherit;

- font-family. Задает список имен семейств шрифтов и/или обобщенных имен семейств в порядке предпочтения. Устанавливается в [[<family-name> | <generic-family>],]* <family-name> | <generic-family>] | inherit;
- font-size. Задает размер шрифта. Устанавливается в <absolute-size> | crelative-size> | <length> | <percentage> | inherit;
- font-style. Задает стиль шрифта. Устанавливается в normal | italic | oblique | backslant | inherit;
- font-variant. Выбирает шрифты, имеющие строчные и прописные буквы (bicameral fonts). Устанавливается в normal | small-caps | inherit;
- font-weight. Задает насыщенность шрифта. Устанавливается в normal) bold | bolder | lighter | 100 j 200 j 300 | 400 | 500 | 600 | 700 | 800 \ 900 | inherit;
- format. Формат XSLT. Устанавливается в <string>;
- grouping-separator. Разделитель групп разрядов формата XSLT. Устанавливается в <character>;
- grouping-size. Размер группы формата XSLT. Устанавливается в <number>;
- height Задает высоту содержимого прямоугольников, создающихся элементами уровня блока и замещенными элементами. Устанавливается в <length> j <percentage> | auto j inherit;
- initial-page-number. Задает начальный номер страницы. Устанавливается в auto | auto-odd | auto-even j <number> | inherit;
- left. Определяет, насколько левый край содержимого прямоугольника (box) смещен вправо от левой границы содержащего прямоугольник блока. Устанавливается в: <length> | <percentage> | auto | inherit;
- linefeed-treatment. Определяет способ обработки символов перевода строки. Устанавливается в ignore j preserve | treat-as-space | treat-as-zero-width-space | inherit;
- line-height. Задает минимальную высоту каждого созданного встроенного прямоугольника. Устанавливается в normal | <length> | <number> | «percentage» | <space> | inherit;
- margin. Свойство для быстрого задания одновременно свойств полей margin-top, margin-right, margin-bottom и margin-left. Устанавливается в <margin-width> {1,4} | inherit;
- margin-bottom. Задает нижнее поле прямоугольника. Устанавливается в <margin-width> J inherit;
- margin-left. Задает левое поле прямоугольника. Устанавливается в <margin-width> | inherit;
- margin-right. Задает правое поле прямоугольника. Устанавливается в cmargin-width> | inherit;
- margin-top. Задает верхнее поле прямоугольника Устанавливается в <margin-width> | inherit;

- master-name. Задает или выбирает шаблон (master). Устанавливается в <name>;
- number-columns-spanned. Задает число столбцов, занимаемых ячейкой таблицы. Устанавливается в «number»;
- number-rows-spanned. Задает число строк, занимаемых ячейкой таблицы. Устанавливается в <number>;
- page-break-after. То же, что и свойство CSS2 с тем же именем. Устанавливается в auto | always | avoid j left | right | inherit;
- page-break-before. То же, что и свойство CSS2 с тем же именем. Устанавливается в auto | always | avoid J left | right | inherit;
- page-height. Задает высоту страницы. Устанавливается в auto j indefinite | <length> J inherit;
- page-width. Задает ширину страницы. Устанавливается в auto | indefinite | <length> | inherit;
- position. Задает используемую схему размещения. Устанавливается в static | relative | absolute | fixed | inherit;
- relative-position. То же, что и свойство CSS2 с тем же именем. Устанавливается в static | relative | inherit;
- right. Определяет, насколько край содержимого прямоугольника (box) смещен влево от правой границы содержащего прямоугольник блока. Устанавливается в <length> | «percentage» j auto | inherit;
- score-spaces. Определяет, следует ли применять свойство text-decoration к Пробелам. Устанавливается в true | false J inherit;
- space-treatment. Определяет способ обработки пробелов и других символов-разделителей, за исключением символов перевода строки. Устанавливается в ignore | preserve | ignore-if-before-linefeed | ignore-if-after-linefeed | ignore-if-surrounding-linefeed | inherit;
- span. Определяет, нужно ли поместить объект уровня блока в текущий столбец или растянуть его на все столбцы в области. Устанавливается в
- none | all | inherit;
- src Задает ссылку URI, определяющую внешний ресурс. Устанавливается в <uri-specification» | inherit;
- start-indent. Определяет расстояние от начального края прямоугольника содержимого области, содержащей блок, до начального края прямоугольника содержимого области этого блока. Устанавливается в «length» | inherit;
- starts-row. Определяет, начинает ли эта ячейка строку. Устанавливается в true | false;
- text-align. Определяет способ выравнивания встроенного содержимого блока. Устанавливается в start | center | end | justify J inside | outside | left | right | «string» j inherit;

- text-decoration. Задает параметры оформления, добавляемого тексту элемента. Устанавливается в none | [[underline | no-underline] || [overline | no-overline] || [line-through J no-line-through] || [blink | no-blink]] | inherit;.
- text-indent. Определяет выравнивание первой строки текста в блоке. Устанавливается в <length> | <percentage> | inherit;
- text-shadow. Задает разделенный запятыми список теневых эффектов, которые нужно применить к тексту элемента. Устанавливается в none | <color> || <length> <length> <length>? ,]* «color» || <length> <length> <length>?] | inherit;
- top. Определяет, насколько край содержимого смещен ниже верхнего края содержащего его блока. Устанавливается в <length> j «percentage» | auto j inherit;
- vertical-align. Определяет вертикальное расположение. Устанавливается в baseline | middle j sub | super | text-top | text-bottom [«percentage» | <length> | top | bottom | inherit;
- visibility. Определяет, будут ли отображены генерируемые элементом прямоугольники. Устанавливается в visible | hidden J collapse | inherit;
- white-space. Определяет способ обработки символов-разделителей (white-space) внутри элемента. Устанавливается в normal | pre j nowrap | inherit;
- white-space-collapse. Задает способ обработки последовательных символов-разделителей. Устанавливается в false | true | inherit;
- width. Задает ширину содержимого прямоугольников, созданных элементами уровня блока и замененными элементами. Устанавливается в <length> | «percentage» | auto | inherit;
- wrap-option. Определяет наличие оболочки строк форматирующего объекта. Устанавливается в no-wrap | wrap | inherit.

Не все свойства применимы ко всем объектам XSL-FO. В этой и следующей главах я буду явно перечислять свойства, применимые к используемым мной объектам. Заметьте также, что пока что ни в одном процессоре XSL-FO не реализованы все эти свойства.

Мы вкратце рассмотрели структуру XSL-FO, теперь пора перейти к практике: сейчас мы подробно рассмотрим работу примера.

Форматирование документа XML

Чтобы посмотреть, как XSL-FO может форматировать данные XML, я воспользуюсь planets.xml (листинг 11.1).

Листинг 11.1. planets.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type«"text/xml" href="planets.xsl"?>
<PLANETS>

  «PLANET COLOR-"RED">
    <NAME>Mercury</NAME>
```

```
  <MASS UNITS-"(Earth - 1)">10553</MASS>
  <DAY UNITS="days">58.65</DAY>
  <RADIUS UNITS-"miles">1516</RADIUS>
  <DENSITY UNITS-"CEarth - 1)">.983</DENSITY>
  <DISTANCE UNITS-"million m1les">43.4</DISTANCE><!-B перигелии-~>
</PLANET>

<PLANET C0L0R-"WHITE">
  <NAME>Venus</NAME>
  <MASS UNITS-"(Earth = 1)">.815</MASS>
  <DAY UNITS-"days">116 75</DAY>
  <RAD1US UNITS-"miles">3716</RADIUS>
  <DENSITY UNITS-"(Earth - 1)">.943</DENSITY>
  <DISTANCE UNITS-"-mil1ion mi1es">66.8</DISTANCE><!--B перигелии-->
</PLANET>

<PLANET COLOR-"BLUE">
  <NAME>Ea rth</NAME>
  <MASS UNITS-"(Earth - 1)">1</HASS>
  <DAY UNITS-"days">l</DAY>
  <RADIUS UNITS-"mi1es">2107</RADIUS>
  <DENSITY UNITS-"(Earth - 1)H>1</DENSITY>
  «DISTANCE UNITS-"million miles">128 4</DISTANCE><!--B перигелии-->
</PLANET>

</PLANETS>
```

В этом первом примере я создам таблицу стилей XSLT для форматирования файла planets.xml и преобразования его в planets.fo, использующую форматирующие объекты для задания шрифтов, стилей и цветов. Затем я воспользуюсь процессов ром fop и преобразую planets.fo в файл planets.pdf, который показан на рис. 11.1.

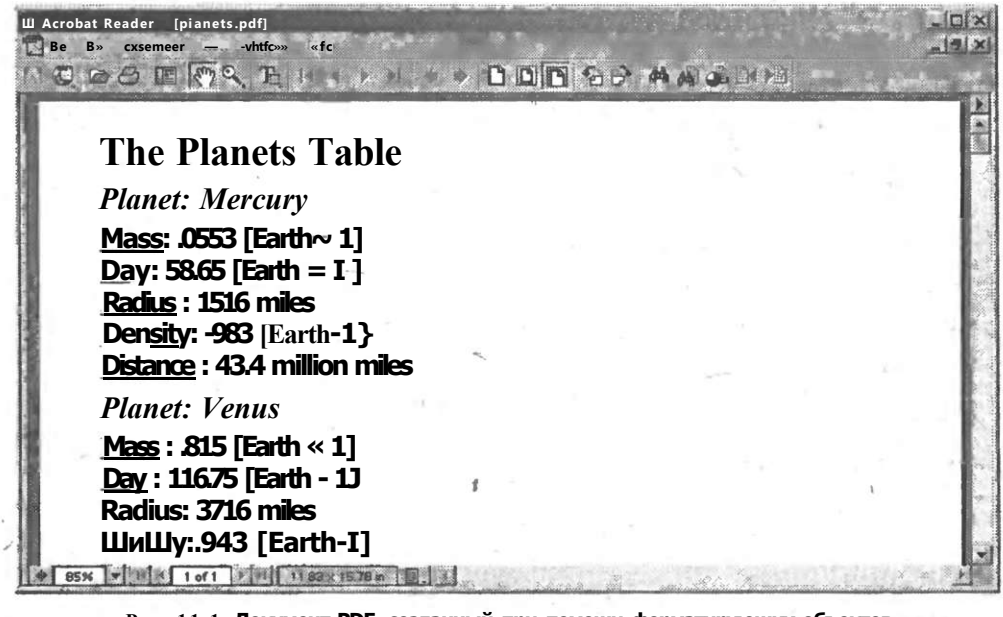


Рис. 11.1. **Документ PDF, созданный при помощи форматирующих объектов**

Как видно на рир. 11.1, в нашем первом примере я применил средства форматирования текста: установку шрифта, подчеркивание текста, выделение текста курсивом и даже установка цвета текста. (Хотя этого и не видно на рис. 11.1, заголовок «The Planets Table» выделен светло-голубым цветом.)

Первый шаг при создании документа на рис. 11.1 — применить таблицу стилей XSLT для преобразования planets.xml в planets.fo.

Преобразование в XSL-FO при помощи таблицы стилей XSLT

В этой главе я создам таблицу стилей для преобразования planets.xml в planets.fo. Я буду создавать таблицу шаг за шагом; для справки я приведу ее окончательный вид (листинг 11.2).

Листинг 11.2. planets.xsl

```
<?xml version-"1.0"?>
<xsl -.stylesheet xmlns xsl-"http://www.w3.org/1999/XSL/Transform"
    xmlns:fo-"http://www.w3.org/1999/XSL/Format"
    version»"1.0">

    <xsltemplate match-"PLANETS">
        <fo:root xmlns:fo-"http://www.w3.org/1999/XSL/Format">

            <fo:1ayout-master-set>
                <fo-simple-page-master master-name="page"
                    page-hei ght="400rnm" page-width-"300mm"
                    margin-top-"10mm" margi n-bottom-"10mm"
                    margin-left-"20mm" margin-right="20mm">

                    <fo:region-body
                      margin-top-"0mm" margi n-bottom-"10mm"
                      mar jji n -1 ef t-" Omm" ma rgi n - r i ght-" Omm" />

                    <fo:region-after extent»"10mm"/>
                </fo:simple-page-master>
            </fo:layout-master-set>

            <fo:page-sequence master-name-"page">

                <fo:flow flow-name-"xsl-region-body">
                    <fo:block font-weight-"bold" font-size="36pt"
                        1ine-height«"48pt" font-family«"Times"
                        color-"blue">
                        The Planets Table
                    </fo:block>
                    <xsl:apply-tempiates/>
                </fo:flow>
            </fo page-sequence>

        </fo:root>
    </xsl tempiate>
```

```
<xsl.rtemplate match-"PLANET/NAME">
    <fo:block font-weight="bold" font-size-"28pt"
        1ine-height-"48pt" font-family-"Times"
        font-style-"italic">
        Planet:
        <xsl:apply-templates/>
    </fo:block>
</xsl:tempiate>

<xsltemplate match="PLANET/MASS">
    <fo:block font-size-"24pt" line-height-"32pt"
        font-family-"Times">
        <fo:inline text-decoration«"underline">
            Mass
        </fo:inline>:
        <xsl:apply-templates/>
        [Earth - 1]
    </fo block>
</xsl:tempiate>

<xslrtemplate match="PLANET/DAY">
    <fo:block font-size-"24pt" line-height-"32pt"
        font-family="Times">
        <fo:inline text decorati on="under!1ne">
            Day
        </fo:inline>:
        <xsliapply-temp^ates/»
        CEarth = 1]
    </fo:block>
</xsl:tempiate>

<xslrtemplate match-"PLANET/RADIUS">
    <fo block font-size~"24pt" line-height="32pt"
        font-family-"Times">
        <fo:i n1i ne text-decorati on-"under1i ne">
            Radius
        </fo:inline>:
        <xsl:apply-templates/>
        miles                                                    i
    </fo:block>
</xsl:tempiate>

<xsl:template match-"PLANET/DENSITY">
    <fo:block font-size-"24pt" line-height-"32pt"                }
        font-family-"Times">
        <fo:i nli ne text-decorati on="under1i ne">
            Density                                  i
        </fo:inline>:
        <xsl:apply-templates/>
        [Earth - 1]
    </fo block>
</xsl template»

<xslrtemplate match-"PLANET/DISTANCE">
    <fo:block font-size-"24pt" I1ne-height-’’32pt"
        font-family-"Times">
        <forinline text-decoration-"under1ine">
```

продолжение &

Листинг 11.2 (продолжение)

```
                Di stance
            </fo:inline>:
            <xsl:apply-templates/>
            million miles
        </foblock>
    </xsl:tempiate>

</xsl:stylesheet»
```

После применения этой таблицы стилей для преобразования planets.xsl будет получен файл planets.fo, который при помощи форматирующих объектов XSL-FO создает документ с видом, показанным на рис. 11.1. Вот как выглядит planets.fo (листинг 11.3).

Листинг 11.3. planets.fo

```
<?xml version^"!..O" encoding^c"UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

    <fo:1ayout-master-set>
        <fo simple-page master margin-right="20mm" margin-left="20mm"
            margin-bottom»"10mm" margin-top-"10mm" page-width="300mm"
            page-height-^400mm" master-name="page">

            <fo:region-body margin-right-"Omm" margin-left-"0mm"
                margi n-bottom-"10mm" margi n-top="0mm"/>

            <fo:region-after extent-"10mm"/>

        </fo:simple- page-master>
    </fo:1ayout-master-set>

    <fo:page-sequence master-name-"page">

        <fo flow flow name="xsl-region-body">

            <fo:block color="blue" font-family-"Times"
                line-height-"48pt" font-size="36pt" font-weight-"bold">
                The Planets Table
            </fo:block>

           <fo:block font-style-"italic" font-family-"Times"
               line-height»"48pt" font-size-"28pt" font-weight-"bold">
1              Planet: Mercury     >«
           </fo:block>

           <fo:block font-family-"Times" line-height-"32pt"
                font-size-"24pt">
                <fo i nli ne text-decoration-"under1i ne">
                    Mass
                </fo:jnline>:
                .0553
                [Earth - 1]
            </fo:block>

            <fo:block font-family="Times" line-height="32pt"
                font-size="24pt">
                <fo:inli ne text-decorati on-"under1ine">
```

```
        Day
    </fo:inline»:
    58 65
    [Earth - 1]
</fo:block>

<fo:block font-fam1ly-"Times" line-height-"32pt"
    font-size-"24pt">
    <fo:inli ne text-decoration-"underline">
        Radius
    </fo:inline»:
    1516                                f
    miles
</fo:block>

<fo block font-family-"Times" line height-"32pt"
    font-size-"24pt"»
    <fo: inline text-decorati on="underlind">
    Density
    </fo:inline»:
    .983
    [Earth - 1]
</fO:block»                                          i

<fo block font-family-"Times" line-height-"32pt"
    font-size-"24pt">
    <fo:inline text-decoration-"underline">
        Distance
    </fo:inline>:
    43.4
    million miles
</fo:block>

<fo:block font-style-"italic" font-family-"Times"
    line-height-"48pt"
    font-size-"28pt" font-weight="bold">
    Planet:
    Venus
</fo:block>                                                                    r

<fo block font-family-"Times" line-height-"32pt"
    font-size-"24pt">
    <fo:i nline text-decoration»"underli ne">
        Mass
    </fo:inline>:
    .815
    [Earth - 1]
</fo:block>

<fo:block font-fami ly-"Times" line-height=’’32pt"                    '
    font-size-"24pt">
    <fo:i nli ne text-decoration-"underli ne">
        Day
    </fo:inline»:
    116.75
    [Earth - 1]
```

продолжение

/

Листинг 11.3 (продолжение)

```
</fo:block>

<fo:block font-family-"Times" line-height="32pt"
    font-size-"24pt">
    <fo inline text-decoration-"under1ine">
        Radius
    </fo,inline>:
    3716
    miles
</fo:block>

<fo:block font-family-"Times" line-height-"32pt"
    fdnt size-"24pt">
    <fo:i nli ne text-decoration-"under1i ne">
        Density
    </fo:inline>:
    .943
    [Earth - 1]
</fo:block>

<fo:block font-family-"Times" line-height="32pt"
    font-size-"24pt">
    <fo:inline text-decoration-"underline">
        Di stance
    </fo.inline>:
    66 8
    million miles
</fo:block>

<fo:block font-style-"italic" font-family="times"
    line-height-"48pt"
    font-s i ze-"28pt" font-wei g2ht-"bold">
    Planet:
    Earth
</fO:bIOCk>

<fo:block font-family-"Times"' line-height-"32pt"
    font-size-"24pt">
    <fo:i nline text-decoration-"underline">
        Mass
    </fo inline>:
    1
    [Earth - 1]
</fo:block>

<fo:block font-family-"Times" line-height-"32pt"
    font-size-"24pt">
    <fo:inline text-decoration-"underline">
        Day
    </fo:tnline>:
    1
    [Earth - 1]
</fo:block>

<fo:block font-family-"Tlmes" line-height-"32pt"
```

```
            font- $ize--""24pt">
            <fo:inline text-decoration="underline">
                Radius
            </fo:inline>:
            2107
            miles
        </fb:block>

        <fo.:block font-family-"Times" line-height="32pt"
            font-size-'"24\)t">r
            <fo:i nli ne text-decoration-"underli ne">
              - Density
            </fo:inline>:
            1
            [Earth, - 1]
        </fo:block>

        <fo:block font-family-Times" line-height«"32pt"
            font- size-"24pt">
            <fo: inl ine text-decoration-"underl i'ne">
                Distance
            </fo:inline>:
            128.4
            million miles
        </fo;blocfc>

    </fo:flow>

</fo:page-sequence>

</fo:root>
```

Для того чтобы обработать planets.fo и создать форматированный документ, я воспользуюсь процессором fop от Apache XML Project. Как утверждает Apache:

«FOP — первое в мире средство форматирования, управляемое форматирующим!: объектами XSL. Приложение Java считывает дерево форматирующих[1] объектов и затем преобразует его в документ PDF. Дерево форматирующих объектов может быть представлено в форме документа XML (полученного на выходе такого процессора XSLT, как XT или Xalan) или может быть передано в память как документ DOM или (в случае XT) событий SAX».

Я пользуюсь fop 0.17, — последней версией процессора на момент написания книги (похоже, что новые версии выходят практически ежемесячно). Процессор fop можно бесплатно загрузить с http://xml.apache.org/fop. Пакет загрузки fop включает три необходимых для работы файла JAR: fop.jar, w3c.jar и xerces.jar, которые нужно включить в classpath (добавьте правильные пути к этим файлам JAR в соответствии с требованиями вашей системы):

```
C:\>set classpath-.:fop.jar:xerces.jar;w3c.jar
```

**Для преобразования planets.fo в planets.pdf служит класс fop org.apache.fop.apps.CommandLine, которому в командной строке нужно передать имя входного документа, planets.fo, и имя выходного, planets.pdf: **

```
C:\>java org.apache.fop.apps.CommandLine planets.fo planets.pdf
```

Вот и все; окончательный результат, planets.pdf, в средстве просмотра Adobe Acrobat reader вы увидите, если вернетесь к рис. 11.1.

Теперь вы видели, как выполняется процедура в общем; давайте перейдем к деталям и посмотрим, как создаются документы XSL-FO. Чтобы подробно все рассмотреть, я собираюсь взять за основу листинг 11.2, таблицу стилей XSLT, создающую planets.fo.

СОЗДАНИЕ ДОКУМЕНТОВ XSL-FO С НУЛЯ

Заметьте, что не обязательно создавать таблицу стилей для преобразования документов XSL в форму XSL-FO. Я мог бы написать planets.fo так, как показано в листинге 11.3, с нуля, не прибегая к таблицам стилей XSLT. Но такой способ, как правило, годится Только для коротких документов XML. Документы с форматированием XSL-FO становятся весьма длинными очень быстро (сравните длину planets.xml с planets.fo), поэтому для создания документов XSL-FO практически всегда применяйте таблицы стилей XSLT (хотя некоторые примеры в следующей главе достаточно коротки для того,, чтобы написать их непосредственно при помощи XSL-FO).

Создание корня документа: <fo:root>

Первый форматирующий документ, который мы рассмотрим, — это <fo:root>, узел документа любого документа XSL-FO.

Детьми форматирующего объекта <fo:root> являются единственный форматирующий объект <fo:layout-master-set> и последовательность из одного или нескольких элементов <fo:page-sequence>. Форматирующий объект <fo:layout-master-set> содержит все используемые в документе шаблоны, при помощи которых вы управляете созданием каждой страницы. Каждый объект <fo:page-sequence> представляет собой последовательность страниц, отформатированных нужных вам способом. Например, каждая глава в книге может быть сформирована из своей собственной последовательности страниц, и каждой последовательности страниц вы можете задать одни и те же верхний и нижний колонтитулы: «Глава 5: Незнакомец появляется снова».

На первом шаге в таблице стилей XSLT, преобразующей planets.xml, я выбираю узел документа <PLANETS> и заменяю его на элемент <fo:root>, объявляющий префикс пространства имен «fo»:

```
<?xml version-"1.0"?>
<xsl:stylesheet  xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"
    xmlns:fo-"http://www.w3.org/1999/XSL/Format"
    version="1.0">

    <xsl:tempiate match="PLANETS">
        <fo:root  xmlns:fo»"http://www.w3.org/1999/XSL/Format">
```

Элемент <fo:root> может содержать и схемы шаблонов, и последовательности страниц. Сначала я опишу объект <fo:layout-master-set>.

Создание схемы набора шаблонов: <fo: layout-master-set>

Шаблоны (master) могут быть определены для страниц, последовательностей страниц и областей. Элемент <fo:layout-master-set> содержит все используемые в документе шаблоны, включая шаблоны последовательностей (sequence master), страниц (page master) и областей (region master).

Шаблоны страниц формируют отдельные страницы, шаблоны последовательностей страниц — последовательности страниц; шаблоны областей позволяют форматировать определенные области страницы. В следующем примере я создам единственный шаблон при помощи <fo:simple-page-master>.

Шаблоны, которые вы хотите применить в документе, должны быть перечислены в элементе <fo:layout-master-set>, поэтому теперь я добавляю его в planets.xsl:

```
<?xml version«"1 0"?>
<xsl:stylesheet xmlns:xsl-"http//www w3.org/1999/XSL/Transform"
    xmlns:fo-"http://www.w3.org/1999/XSL/Format"
    version-"1.0">

    <xsl:tempi ate match-"PLANETS"»
        <fo:root xmlns:fo-"http://www.w3.org/1999/XSL/Format">

            <fo:1ayout-master-set>
            .
            .
            .
            </f o:1ayout•master•set>
```

Этот элемент содержит шаблон страницы, как определено в элементе <fo:simple-page-master>.

Создание шаблона страницы: <fo:simple-page-master>

Как можно догадаться из названия, шаблон страницы (page master) применяется для создания страницы. Шаблон страницы задает фактическую схему и конфигурацию страницы. Каждому шаблону страницы должно быть задано уникальное имя, к которому и нужно обращаться при работе с шаблоном.

В настоящей спецификации XSL существует единственный вид шаблона страницы, <fo:simple-page-master>, и здесь я буду применять этот элемент для форматирования страниц. С объектом <fo:simple-page-master> можно использовать следующие свойства XSL-FO (их описание приведено в Приложении Б):

- **общие свойства полей для блоков: margin-top, margin-bottom, margin-left, margin-right, space-before, space-after, start-indent, end-indent;**

- master-name;
- page-height;
- page-width;
- reference-orientation;
- writing-mode.

В таблице стилей XSLT, которую я применил к planets.xml, я задал простому шаблону страницы имя «page» при помощи свойства master-name. После этого шаблон стал именованным, и когда мне нужно создать страницы по этому шаблону, я обращаюсь к нему по этому имени. Я также задаю размеры и поля страницы при помощи свойств страницы и полей следующим образом:

```
<?xml version="1.0"?>
<xsl stylesheet xmlns: xsl="http //wwww3.org/1999/XSL/Transform"
    xmlns:fo»"http://www.w3.org/1999/XSL/Format"
    version»"1.0">

    <xslrtemplate match="PLANETS">
        <fo:root xmlns:fo-"http://www.w3,org/1999/XSL/Format">

            <fo:1ayout-ma ster-set>
                <fo:simple-page-master master-name="page"
                    page-height="400mm" page-width»"300mm"
                    margin-top-"lOrnm" margin-bottom»"10mm"
                    margin-1eft-"20mm" margin-right="20mm">
                    .
                    .
                    .
```

В дополнение к заданию схемы расположения полей страницы, у <fo:simple-page-master> есть дети, определяющие одну или несколько областей страницы, что позволяет точно настроить схему расположения.

i

Создание областей

В версии 1.0 спецификации XSL у шаблонов страниц имелось пять областей (region). Центральная область, соответствующая основной части, телу страницы, называется *областью тела* (body region). Верхняя часть страницы, верхний колонтитул (header), называется *передней областью* (before region); нижняя часть страницы, нижний колонтитул (footer), называется *задней областью* (after region). В языках, которые читаются слева направо, как английский язык, левая часть страницы называется *начальной областью* (start region), а правая часть называется *конечной областью* (end region). В языках, которые читаются справа налево, начальная и конечная области меняются местами. Начальная и конечная области аналогичны боковым полям, расположенным по бокам области тела.

Этим областям соответствуют следующие элементы XSL-FO:

- <fo:region-before>;
- <fo:region-after>;
- <fo:region-body>;
- <fo:region-start>;
- <fo:region-end>.

С ними можно применять следующие свойства:

- общие свойства границ, заполнения и заднего фона: background-attachment, background-color, background-image, background-repeat, background-position-horizontal, background-position-vertical, border-before-color, border-before-style, border-before-width, border-after-color, border-after-style, border-after-width, border-start-color, border-start-style, border-start-width, border-end-color, border-end-style, border-end-width, border-top-color, border-top-style, border-top-width, border-bottom-color, border-bottom-style, border-bottom-width, border-left-color, border-left-style, border-left-width, border-right-color, border-right-style, border-right-width, padding-before, padding-after, padding-start, padding-end, padding-top, padding-bottom, padding-left, padding-right;
- общие свойства полей для блоков: margin-top, margin-bottom, margin-left, margin-right, space-before, space-after, start-indent, end-indent;
- clip;
- column-count;
- column-gap;
- display-align;
- extent;
- overflow;
- region-name;
- reference-orientation;
- writing-mode.

Области страницы можно настраивать по вашему желанию, как в таблице стилей XSLT, в которой я задал поля для области тела. Четыре внешних области (но не область тела) имеют свойство extent, задающее их размер, которым я здесь воспользуюсь:

```
<?xml version-"1.0"?>
<xsl:stylesheet xmlns:xsl-"http //www.w3.org/1999/XSL/Transform"
    xmlns:fo="http:7Vwww.w3.org/1999/XSL/Format"
    version-"1.0">

    <xsl template match-"PLANETS">
        <fo:root xmlns:fo-"http://www.w3.org/1999/XSL/Format">

            <fo:layout-master-set>
                <fo:sinple-page-master master-name-"page"
```

```
            page-height-"400mm"  page-width-"300mm"
            ma rgi n-top-"1Omm" margi n-bottorn-"lOmm"
            ma rgi n-1eft-"20mm" ma rgi n-ri ght-"20mm">

            <fo:region-body
              margin-top-"0mm"  margin-bottom»"10mm"
              margi n-1eft-"Omm" margi n•ri ght-"Omm"/>

            <fo:region-after extent-"10mm"/>
        </fo:simple-page-master>
    </fo:layout-master-set>
```

Это завершает используемый в документе шаблон, простой шаблон страницы «page», почему здесь также завершается и элемент <fo:layout-master-set>.

В дополнение к элементу <fo:layout-master-set>, документ с форматирующими объектами, как правило, также содержит один или несколько элементов <fo: page-sequence», задающих последовательности страниц, создаваемых по определенному в <fo:layout-master-set> шаблону, и я добавил в документ последовательность страниц.

Создание последовательностей страниц: <fo:page-sequence>

Что такое последовательность страниц? Это ряд страниц с одинаковыми характеристиками (например, глава в книге), которые при желании можно форматировать одинаковым способом. Страницы в результирующем документе в действительности создаются тогда, когда процессор XSL-FO обрабатывает элементы <fo:page-sequence>.

Каждый элемент <fo:page-sequence> ссылается либо на элемент <fo:page-sequence-master>, либо на шаблон страницы, и фактическая схема страниц определяется этими шаблонами. Несколько более сложно создание последовательностей, у которых место расположения номера страницы меняется — номеру располагаются на разных сторонах страницы, как при создании страниц книги.

К объекту <fo:page-sequence> применимы следующие свойства:

- country;
- format;
- language;
- letter-value;
- grouping-separator;
- grouping-size;
- id;
- initial-page-number;

- force-page-count;
- master-name.

В последней рекомендации XSL W3C шаблон страницы для последовательности страниц задается атрибутом master-name элемента <fo:page-sequence>. Я назвал свой простой шаблон страницы «page», и здесь я устанавливаю данный атрибут в это имя:

```
<?xml version="1.0"?>
<xsl:sty!esheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fo="http://www w3.org/1999/XSL/Format"
    version="1.0">

    <xsltemplate match»"PLANETS">
        <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

            <fo:layout-master-set>
                <fo:simple-page-master master-name="page"
                    page-height="400mm" page-width-"300mm"
                    ma rg i n-top-"1Omm" ma rg i n-bottom-"1Omm"
                    ma rgi n-1eft="20mm" ma rgi n-ri ght="20mm">

                    <fo:region-body
                      ma rg i n-1 op="Omm" ma rg i n-bottom-"10mm"
                      margin-left="Omm" margin-right="0mm'7>

                    <fo:region-after extent-"1Omm"/>
                </fo:simple-page-master>
            </fo:layout-master-set>

            <fo:page sequence master-name="page">
                .
                .
                .
            </fo:page-sequence»
                .
                .
                .
```

Этот код определяет шаблон страницы, который будет использован для последовательности страниц. Теперь мне нужно задать *содержимое* (content) последовательности, что делается при помощи детей *потока* (flow) элемента <fo:page-sequence>.

Создание объектов потока: <fo:flows>

Объекты потока носят такое название потому, что текст в них «заливается» и выравнивается, заполняя страницу, программой, отвечающей за вывод на экран. Содержимое страницы управляется объектами потока.

Есть два вида объектов потока: <fo:static-content> и <fo:flow>. Объект <fo:static-content> хранит содержимое (например, текст в верхних и нижних колонтитулах), которое будет повторяться на страницах последовательности (как мы увидим

в главе 12). Объект <fo:flow> содержит сам текст, образующий содержимое документа.

К <fo:flow> применимо следующее свойство:

- flow-name.

Я воспользуюсь элементом <fo:flow> для обработки текстового содержимого planets.xml. Чтобы гарантировать, что текст будет преобразован в поток, я применю элемент <xsl: a pply-tem plates»:

```
<?xml version-"1.0"?»
<xsl:stylesheet xmlns:xsl-"http://www.w3.org/1999/XSl/Transfomf
    xmlns:fO="http://www w3.org/1999/XSL/Format"
    version«"1.0">

    <xsltemplate match-"PLANETS"»
        <fo:root xmlns:fo-"http://www.w3.org/1999/XSL/Format"»

            <fo:layout-master-set>
              .
              .
              .
            </fo:1ayout-master-set>

            <fo:page-sequence master-name="page^B>

                <fo:flow flow-name-"xsl-region-body"»
                    <xsl:apply-templates/>
                </fo:flow>
            </fo:page-sequence»
    .
    .
    .
```

Здесь мы заканчиваем рассмотрение элемента <fo:page-sequence>; я указал шаблон для этой последовательности, который предоставит процессору XSL-FO способ форматирования содержимого в страницах форматированного документа. После описания схемы страниц, которые я хочу создать, теперь, наконец, пора перейти к их содержимому. Первый элемент, отображающий содержимое, — <fo:block>.

Создание содержимого уровня блока: <fo:block>

Блоки играют важную роль в XSL-FO: они применяются для создания прямоугольных областей вывода, отделенных от других областей вывода в документе. Форматирующий объект <fo:block> применяется для форматирования таких элементов, как абзацы, заголовки, подписи к рисункам и таблицам и т. д. Вот пример из начала главы, где я создаю элемент блока и задаю различные свойства и текст в блоке:

```
<fo:block font-family»"Times" line-height="48pt" font-size-"36pt">
    Welcome to XSL formatting.
</fo:block»
```

С объектом <fo:block> можно применять следующие свойства:

- общие свойства доступа: source-document, role;
- общие звуковые свойства: azimuth, cue-after, cue-before, elevation, pause-after, pause-before, pitch, pitch-range, play-during, richness, speak, speak-header, speak-numeral, speak-punctuation, speech-rate, stress, voice-family, volume;
- общие свойства границ, заполнения и заднего фона: background-attachment, background-color, background-image, background-repeat, background-position-horizontal, background-position-vertical, border-before-color, border-before-style, border-before-width, border-after-color, border-after-style, bOrder-after-width, border-start-color, border-start-style, border-start-width, border-end-color, border- end-style, border-end-width, border-top-color, border-top-style, border-top-width, border-bottom-color, border-bottom-style, border-bottom-width, border-left-color, border-left-style, border-left-width, border-right-color, border-right-style, border-right-width, padding-before, padding-after, padding-start, padding-end, padding-top, padding-bottom, padding-left, padding-right;
- общие свойства шрифта: font-family, font-size, font-stretch, font-size-adjust, font-style, font-variant, font-weight;
- общие свойства переноса: country, language, script, hyphenate, hyphenation-character, hyphenation-push-character-count, hyphenation-remain-character-count;
- общие свойства полей для блоков: margin-top, margin-bottom, margin-left, margin-right, space-before, space-after, start-indent, end-indent;
- break-after;
- break-before;
- color;
- font-height-override-after;
- font-height-override-before;
- hyphenation-keep;
- hyphenation-ladder-count;
- id;
- keep-together;
- keep-with-next;
- keep-with-previous;
- last-line-end-indent;
- linefeed-treatment;
- line-height;
- line-height-shift-adjustment;
- line-stacking-strategy;
- orphans;
- relative-position;

- space-treatment;
- span;
- text-align;
- text-align-last;
- text-indent;
- visibility;
- white-space-collapse;
- widows;
- wrap-option;
- z-index.

Например, я могу добавить в документ заголовок «The Planets Table» (Таблица планет), задав шрифт Times (в данный момент fop поставляется с встроенными шрифтами Times, Helvetica, Courier, Symbol, sans-serif, serif и ZapfDingbats) в свойстве font-family, размер шрифта 36 пунктов в свойстве font-size и полужирный стиль, установив свойство font-weight в «bold». Высоту блока я задам при помощи свойства line-height и покажу заголовок голубым при помощи свойства color:

```
<?xml version-"1.0"?>
<xsl: styl esheet xmlnsxsl="http //www, w3.org/1999/XSL/Transform"
    xmlns:fo-"http://www.w3.org/1999/XSL/Format"
    version="1.0">                                    :

    <xsl:tempi ate match-"PLANETS">
        <fo:root  xmlns:fo="http://www.w3.org/1999/XSL/Format">

            <fo:layout-master-set>



             </fo:1ayout-master-set>

             <fo:page-sequence master-name-"page">

                 <fo:flow flow-name-"xsl-region-body">
                     <fo:block font-weight-"bold" font-size-"36pt"
                         line-height="48pt" font-family-Times"
                         color»"blue">
                         The Planets Table
                     </fo:block>
                     <xsl:apply-templates/>
                 </f0:flow>
             </fo:page-sequence>
```

Этот код создаст блок заголовка, который показан в верхней части текста на рис. 11.1. Таким способом я могу создать аналогичные блоки для каждого элемента данных каждой планеты при помощи таблицы стилей XSLT. Заметьте так-

же, что я вывожу имя каждой планеты курсивом, установив свойство **font-style** в **«italic»,** и я подчеркиваю остальной текст при помощи свойства **text-decoration** элементов **<fo:inline>,** которые мы рассмотрим в следующей главе:

```
«'xml version="1.0"?>
<xsl:stylesheet xmlns:xsl-"http://www.w3.org/1999/XSL/Transform"
    xmlns:fo«"http://www.w3.org/1999/XSL/Format"
    version-"1.0">

    <xsl:template match«"PLANETS">
        <fo:root xmlns fo="http //www.w3.org/1999/XSL/Format">

            <fo:layout-master-set*
            .
            .
            .
             </fo:layout-master-set>

             <fo:page-sequence master-name-"page">
            .
            .
            .
             </fo:page-sequence»

        </fo:root>
    </xsl:template>

    <xslrtemplate match-"PLANET/NAME">
        <fo:block font-we1ght-"bold" font-size»"28pt"
            1i ne-hei ght-"48pt" font-fami1y-"Times"
            font•style-"i tali c">
            Planet:
            <xsl:apply•tempiates/>
        </fo:block*
    </xsl:template*

    <xsl template match-"PLANET/MASS">
        <fo:block font-size-"24pt" line-height-"32pt"
            font-family»"Times"*
            <fo:1nline text-decorat1on="underline">
                Mass
            </fo:1nl1ne>:
            <xsl:apply-templates/>
            [Earth - 1]
        </fo:block>
    </xsl:template*
        .
        .
        .
</xsl:stylesheet*
```

Вот и все. Вы создали свое первое преобразование из XML в XSL-FO, преобразовав planets.xml в planets.fo. Процессор fop создаст из planets.fo файл planets.pdf, и вы можете вернуться к рис. 11.1 и посмотреть на результат.

Это преобразование форматировало данные в planets.xml и отображало их в блоках, одно за другим. С другой стороны, в предыдущих главах мы видели

данные планет в виде таблицы. Можно ли это сделать в XSL-FO? Никаких проблем.

Создание таблиц

Таблица — одна из самых полезных структур, которые можно форматировать при помощи XSL-FO. Таблица в XSL-FO во многом похожа на таблицу в HTML: это прямоугольная сетка, состоящая из строк и столбцов ячеек. Для создания таблиц можно применять девять форматирующих элементов:

- <fo:table-and-caption>;
- <fo:table>;
- <fo:table-column>;
- <fo:table-caption>;
- <fo:table-header>;
- <fo:table-footer>;
- <fo:table-body>;
- <fo:table-row>;
- <fo:table-cell>.

Создание таблиц в XSL-FO аналогично их созданию в HTML Вы создаете элемент <fo:table>, содержащий всю таблицу целиком, затем форматируете каждый столбец при помощи элемента <fo:table-column>. После чего вы создаете элемент <fo:table-body>, чтобы задать тело таблицы. Элемент <fo:table-body> содержит все элементы <fo:table-row>, каждый из которых создает строку таблицы. Каждый элемент <fo:table-row> содержит элементы <fo:table-cell>, в которых содержатся данные ячеек таблицы.

Следующий пример (листинг 11.4) демонстрирует работу с этими элементами. Приведенная таблица стилей XSLT преобразует planets.xml в документ XSL-FO, форматирующий данные планет в таблицу XSL-FO.

Листинг 11.4. tables.xsl

```
<?xml version-"1.0"?>
<xsl:stylesheet xmlns:xsl-"http://www.w3.org/1999/XSL/Transfarm"
    xmlns:fo-"http://www.w3.org/1999/XSL/Format"
    version-"1.0">

    <xsl template match-"PLANETS">
        <fo:root xmlns:fo-"http://www.w3.org/1999/XSL/Format">
            <fo:layout-master-set>
                <fo:s1mple-page-master master-name-"page"
                    page-he1ght«"400mm" page-width«"300mm"
                    margin-top-"10mm" margin-bottom-"10mm"
                    margin-1eft-"20mm" margin-r1ght-"20mm">

                    <fo:region-body
                      margin-top-"0mm" margin-bottom-"10mm"
                      margin-1eft-"Omm" margin-right-"0run"/>
```

```
        <fo:reglon-after extent-"10mm"/>
    </fo:simple-page-master>
</fo:layout-master-set>

<fo:page-sequence master-name-"page">

    <fo:flow flow-name="xsl-region-body">
        <fo table>

            <fo:table-column column-width-"30mm"/>
            <fo:table-column colurnn-width-"30mm"/>
            <fo:table-column column-w1dth-"30mm"/>
            <fo:table-column column-width-"30ntn"/>
            <fo:table-column column-width«"30mm"/>
            <fo:table-column column-width-"30mm"/>

            <fo:table-body>
                <fo:tatle-row>
                    <fo:table-eel 1 border-width-"0.5mm">
                        <fo:block font-size«"18pt"
                            font-weight-"bold">
                            Name
                        </f0:blOCk>
                    </fo table-cell>

                    <fo:table-eel1 border-wi dth-"0.5mm">
                        <fo:block font-size-T8pt"
                            font-wen ght-"bold">
                            Mass
                        </fo:block>
                    </fo:table-cell>

                    <fo:table-eel 1 border-width' ="0,5mm">
                        <fo:block font-size-"18pt"
                            font-weight-"bold">
                            Day
                        </fo:block>
                    </fo table-cell>

                    <fo: table-cel 1 border-width-''0. 5mm">
                        <fo:block font-size-"18pt"
                            font-weight-"bold">
                            Radius
                        </fo:block>
                    </fo:table-cell>

                    <fo:table-eel 1 border-width-"0.5mm">
                        <fo:block font-size-"18pt"
                            font-wei ght="bold">
                            Density
                        </fo:block»
                    </fo:table-cel1>

                    <fo table-cell border-width-"0.5mm">
                        <fo:block font-size-"18pt"
                            font-weight-bold'>
                            Distance
```

продолжение &

Листинг 11.4 (продолжение)

```
                                        </forblcck*
                                    </fortable-cell*
                                </fo:table-row>

                                <xsl:apply-templates/>

                            </fortable-body*
                        </fo:table>
                    </fo:flow>
                </fo:page-sequence>
            </fo:root>
        </xsl:tcmplate>

        <xslrtemplate match-"PLANET">
            <fo:table-row>
                <xsl:apply-templates/>
            </fo:table-row>
        </xsl:template*

        <xslrtemplate match-"NAME">
              <for table-cH1 border-width-"0.5mm"*
                  <fo:block font-size-"18pt">
                      <xslrvalue-of select-','/>
                  </fo:block*
              </fo:table-cell*
        </xslrtemplate*

        <xslrtemplate match="MASS">
              <fo:table-cel1 border-wi dth«"0.5mm">
                  <fo block font-size«"18pt"*
                      <xsl:value-of select-'.'/>
                  </fo:block*
              </fo:table-cell*
        </xslrtemplate*

        <xslrtemplate match-"DAY">
              <fo:table-cell border-width-"0.5mm"*
                  <fo:block font-size-"18pt">
                      <xsT:value-of select-'.'/*
                  </fo:block*
              </fo:table-cell*
        </xslrtemplate*

        <xs1rtemplate match-"RADIUS">
              <fo:table-cell border-width-"0.5mm"*
                  <fo:block font-s1ze-"l8pt">
                      <xslrvalue-of select-'.'/>
                  </fo:block*
              </fortable-cell*
        </xslrtemplate*

        <xslrtemplate match •"DENSITY"*
              <fo r table-cel1 border-wi dth-"0.5mm">
                  <fo:block font-size-"18pt">
                      <xslrvalue-of select-'.'/>
                  </forblock*
```

```
        </fo-table-cel1>
    </xsl:tempiate>

    <xsl:template match-"DISIANCE">
        <fo:table-cell border-w1dth-"0 5mm">
            <fo:block font-size-"18pt">
                <xsl:value-of select-'.'/>
            </fo:block>
        </fo:table-cel1>
    </xsl:tempiate>

</xsl stylesheet>
```

Вот результат после преобразования в документ XSL-FO (листинг 11.5).

Листинг 11.5. tables.fo

```
<?xml version="l 0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

    <fo 1ayout-master-set>
        <fo simple-page-master margin-right="20mm" margin-left-"20mm"
            margin-bottom-"10mm" margin-top="10mm" page-width-"300mm"
            page-he i ght-"400mm" ma ster-name-"page">

            <fo:region-body margin-r1ght="0mm" margin-left-"0mm"
                margi n-bottom-"10mm" ma rgi n-top-"Omm"/>

            <fo:regi on-after extent-"10mm"/>

        </fo:simple-page-master>
    </fo:layout-master-set>

    <fo:page-sequence master-name="page">

        <fo:flow flow-name="xsl-region-body">

            <fo:table>
                <fo:table-column column-width="30mm"/>
                <fo:table-column column-width-"30mm"/>
                <fo table-column column-width-"30mm"/>
                <fo:tab1e-column column-wi dth-"30mm"/>
                <fo:table-column column-width="30mm"/>
                <fo:table-column column-width-"30mm"/>

                <fo:table-body>
                    <fo:table-row>
                        <fo:table-cell border-width-"0.5mm">
                            <fo block font-weight-"bold" font-size-"18pt">
                                Name
                            </fo:block>
                        </fo:table-cell>
                        <fo:table-cel1 border-width«"0.5mm">
                            <fo:block font-weight="bold" font-size-"18pt">
                                Mass
                            </fo:block>
                        </fo:table-cell>
                        <fo:table-cel1 border-width-"0.5mm">
```

продолжение &

Листинг 11.5 (продолжение)

```
                <fo:block font-weight-"bold" font-size="18pt">
                    Day
                </fo:block>
            </fo:table-cel1>
            <fo table-cell border-width-"0 5mm">
                <fo:block font-weight="bold" font-size-"18pt">
                    Radius
                </fQ:block>
            </fo:table-cell>
                <fo:table eel1 border-wi dth-"0.5mm">
                    <fo:block font-weight-"bold"
                        font-size-"18pt">
                        Density
                    </fo:block>
            </fo:table-cell>
            <fo:table-cell border-width-"0.5mm">
                <fo:block font-weight-"bold" font-size-"18pt">
                    Distance
                </fo;block>
            </fo:table-cell>
        </fo:table-row>

        <fo:table-row>
            <fo table-cell border-width-"0 5mm">
                <fo:block font-size-"18pt">Mercury</fo:block>
            </fo:table-cell>
            <fo:table-cell border-width-"0.5mm">
                <fo:block font-size-"18pt">.0553</fo:block>
            </fo:table-cell>
            <fo table-cell border-width="0 5mm">
                <fo:block font-si ze-"18pt ">58,65</fo:block>
            </fo:table-cell>
            <fo:table-cel1 border-width="0.5mm">
                <fo:block font-si ze-"18pt">1516</fo:block>
            </fo:table-cell>
            <fo.table-cel1 border-width-"0.5mm">
                <fo:block font size-"18pt">.983</fo:block>
            </fo:table-cel1>
            <fo:table-cell border-width-"0.5nm">
                <fo:block font-si ze-"18pt">43.4</fo:block>
            </fo:table-cell>
        </fo:table-row>

        <fo:table-row>
            <fo:table-cel1 border-width-"0.5mm">
                <fo:block font-size-"18pt">Venus</fo:block>
            </fo:table-cell>
            <fo:table-cel1 border-width-"0.5mm">
                <fo:block font-size-"18pt">.815</fo:b1ock>
            </fo table-cel1>
            <fo:table-cell border-width-"0,5mm">
                <fo:block font-size="18pt">116.75</fo:block>
            </fo:table-cel1>
            <fo:table-cel1 border-wi dth-"0.5mm">
                <fo:block font-size-"18pt"*3716</fo:b1ock>
            </fo-table-cel1>
            <fo:table-cell border-width-"G 5mm">
```

```
                    <fo:block font-size-"18pt">.943</fo:block>
                </fo:table-cell>
                <fo:table-cel1 border-wi dth-"0.5mm">
                    <fo;block font-size-"I8pt">66,8</fo:block>
                </fo:table-cel1>
            </fo table-row>

            <fo:table-row>
                <fo:table-cel1 border-wi dth-"0.5mm">
                    <fo:block font-s i ze-"18pt">Ea rth</fo:block>
                </fo:table-cel1>
                <fo:table-cel1 border-width-"0.5mm">
                    <fo: bl ock font - si ze-" 18pt ">1'</fo: bl ock>
                </fo:table-cel1>
                <fo:table-cel1 border-width-"0.5mm">
                    <fo:block font-s i ze-"18pt">1</fo:block>
                </fo:table-cell>
                <fo:table-cell border-width-"0.5mm">
                    <fo.block font-s i ze-"18pt">2107</fo:block>
                </fo:table-cel1>
                <fo:table-cell border-width-"0.5mn">
                    <fo:block font-size-"18pt">l</fo:block>
                </fo:table-cell>
                <fo:table-cell border-width -"0.5mm">
                    <fo:block font-si ze="18pt">128.4</fo:block>
                </fo:table-cell>
            </fO:table-row>
        </fo table-body>
    </fo:table>
</fo:flow>
</fo:page-sequence>
</fo:root>
```

После обработки этого документа, tables.fo, процессором fop будет создан файл tables.pdf, который показан на рис. 11.2. Вот как выглядит таблица XSL-FO, хотя существует также весьма много других параметров — например, можно задать цвет заднего фона ячеек при помощи свойства background-color. По умолчанию в таких таблицах отсутствуют границы, но я добавил границу толщиной в 0,5 мм при помощи свойства border-width. Заметьте также, что для того, чтобы установить размер шрифта для текста в каждой ячейке, я использую блок внутри каждой ячейки таблицы:

```
<fo:table-cel1 border-wi dth-"0.5mm">
    <fo:block font-si ze-"18pt">Earth</fo:block>
</fo:table-cell>
```

Давайте рассмотрим различные элементы, нужные для создания таблицы, и начнем мы с большого элемента, <fo:table>.

Создание таблиц: <fo:table>

Как можно догадаться из названия, элемент <fo:table> применяется для создания новой таблицы. Сама таблица состоит из необязательного заголовка, необязательного подстрочного примечания и одного или более тел таблицы (table body). Таблица из ячеек, выровненная по строкам и столбцам, выводится в теле таблицы.

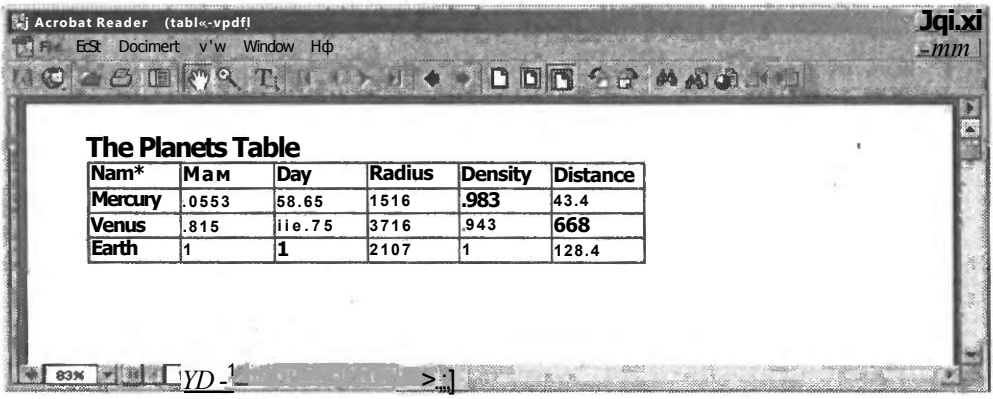


Рис. 11.2. Таблица, отформатированная при помощи XSL-FO, в Adobe Acrobat Reader

С элементом <fo:table> можно применять следующие свойства:

- общие свойства доступа: source-document, role;
- общие звуковые свойства: azimuth, cue-after, cue-before, elevation, pause-after, pause-before, pitch, pitch-range, play-during, richness, speak, speak-header, speak-numeral, speak-punctuation, speech-rate, stress, voice-family, volume;
- общие свойства границ (border), заполнения (padding) и заднего фона (background): background-attachment, background-color, background-image, background-repeat, background-position-horizontal, background-position-vertical, border-before-color, border-before-style, border-before-width, border-after-color, border-after-style, border-after-width, border-start-color, border-start-style, border-start-width, border-end-color, border-end-style, border-end-width, border-top-color, border-top-style, border-top-width, border-bottom-color, border-bottom-style, border-bottom-width, border-left-color, border-left-style, border-left-width, border-right-color, border-right-style, border-right-width, padding-before, padding-after, padding- start, padding-end, padding-top, padding-bottom, padding-left, padding- right;
- общие свойства полей: margin-top, margin-bottom, margin-left, margin-right, space-before, space-after, start-indent, end-indent;
- block-progression-dimension;
- border-collapse;
- border-separation;
- break-after;
- break-before;
- id
- inline-progression-dimension;
- height;
- keep-together;

- keep-with-next;
- keep-with-previous;
- relative-position;
- table-layout;
- table-omit-footer-at-break;
- table-omit-header-at-break;
- width;
- writing-mode;

Как видно из листинга 11.5, таблица tables.fo начинается с <fo:table>:

```
<fo:table>


</fo table>
```

Следующий шаг в создании таблицы — задать столбцы таблицы при помощи <fo:table-column>.

Создание столбцов таблицы: <fo:table-column>

Для каждого столбца в таблице нужно включить один элемент <fo:table-column>. Этот элемент позволяет задать характеристики ячеек в одном и том же столбце таблицы. Одно из самых важных свойств здесь — свойство column-width, определяющее ширину каждого столбца.

С элементом <fo:table-column> можно применять следующие свойства:

- общие свойства границ, заполнения и заднего фона: background-attachment, background-color, background-image, background-repeat, background-position-horizontal, background-position-vertical, border-before-color, border-before-style, border-before-width, border-after-color, border-after-style, border-after-width, border-start-color, border-start-style, border-start-width, border-end-color, border-end- style, border-end-width, border-top-color, border-top-style, border-top-width, border-bottom-color, border-bottom-style, border-bottom-width, border-left-color, border-left-style, border-left-width, border-right-color, border-right-style, border-right-width, padding-before, padding-after, padding-start, padding-end, padding-top, paddiftg-bottom, padding-left, padding-right;
- column-number;
- column-width;
- number-columns-repeated;
- number-columns-spanned;
- visibility.

В tables.fo я установил для каждого столбца одинаковую ширину, 30 мм:

```
<fo:table>
    <fo:table-column column-wi dth-"30mm"/>
    <fo:table-column column-width-"30mm"/>
    <fo:table-column column-wi dth-"30mm"/>
    <fo:table-column column-width-"30mm"/>
    <fo:table-column column-width-"30mm"/>
    <fo:table-column column-wi dth»"30mm"/>
        .
        .
        .
</fo:table>
```

После задания каждого столбца создается тело таблицы.

Создание тел таблицы: <fo:table-body>

Тело таблицы (table body) содержит фактическое содержимое таблицы; тела создаются при помощи элемента <fo:table-body>. Этот элемент содержит элементы <fo:table-row>, которые, в свою очередь содержат элементы <fo:table-cell>, содержащие данные таблицы.

С элементом <fo:table-body> можно применять следующие свойства:

- общие свойства границ, заполнения и заднего фона: background-attachment, background-color, background-image, background-repeat, background-position-horizontal, background-position-vertical, border-before-color, border-before-style, border-before-width, border-after-color, border-after-style, border-after-width, border-start-color, border-start-style, border-start-width, border-end-color, border-end-style, border-end-width, border-top-color, border-top-style, border-top-width, border-bottom-color, border-bottom-style, border-bottom-width, border-left-color, border-left-style, border-left-width, border-right-color, border-right-style, border-right-width, padding-before, padding-after, padding-start, padding-end, padding-top, padding-bottom, padding-left, padding-right;
- id;
- relative-position.

Как можно видеть в листинге 11.5, для создания тела таблицы в tables.fo я использовал <fo:table-body>:

```
<fo:table>
    <fo:table-column column-width-"30mm"/>
    <fo table-column column-width-"30mm"/>
    <fo:table-column col umn-width-" 30m"/>
    <fo:table-column column-width-"30mm"/>
    <fo:table-column column-width-"30mm"/>
    <fo:table-column column-width-"30mm"/>

    <fo:table-body>
        .
        .
        .
```

```
    </fo:table-body>
</fo table>
```

После создания тела таблицы нужно создать строки таблицы при помощи элемента <fo:table-row> — по одному элементу для каждой строки.

Создание строк таблицы: <fo:table-row>

Аналогично таблицам HTML, для создания строк таблицы применяется элемент <fo:table-row>, и каждая строка может содержать ячейки таблицы. Как и браузеры HTML, процессор XSL-FO определяет размеры таблицы по количеству строк.

С элементом <fo:table-row> можно применять следующие свойства:

- общие свойства доступа: source-document, role;
- общие звуковые свойства: azimuth, cue-after, cue-before, elevation, pause-after, pause-before, pitch, pitch-range, play-during, richness, speak, speak-header, speak-numeral, speak-punctuation, speech-rate, stress, voice-family, volume;
- общие свойства границ, заполнения и заднего фона: background-attachment, background-color, background-image, background-repeat, background-position-horizontal, background-position-vertical, border-before-color, border-before-style, border-before-width, border-after-color, border-after-style, border-after-width, border-start-color, border-start-style, border-start-width, border-end-color, border-end-style, border-end-width, border-top-color, border-top-style, border-top-width, border-bottom-color, border-bottom-style, border-bottom-width, border-left-color, border-left-style, border-left-width, border-right-color, border-right-style, border-right-width, padding-before, padding-after, padding-start, padding-end, padding-top, padding-bottom, padding-left, padding-right;
- block-progression-dimension;
- break-after;
- break-before;
- id;
- height;
- keep-together;
- keep-with-next;
- keep-with-previous;
- relative-position.

Я добавил строки в таблицу в tables.fo при помощи элементов <fo:table-row> следующим образом:

```
<fo:table>
    <fo:table-column column-width-"30mm"/>
    <fo:table-column column-width-H30mm"/>
    <fo:table-column column-width®"30mm"/>
    <fo:table-column column-wi dth-H30mm"/>
    <fo:table-column column-width-"30mm"/>
```

```
        <fo:table-column column-width-"30mm"/>

        <fo:table-body>
            <fo:table-row>
                .
                .
                .
            </fo:table-row>

            <fo-.table-row>
                .
                .
                .
            </fo:table-row>
            .
            .
        </fo:table-body>
    </fo:table>
```

Создав нужные строки, мы теперь готовы к включению в ячейки таблицы самих данных.

Создание ячеек таблицы: <fo:table-cell>

Снова, почти как при создании таблицы в HTML, вы помещаете данные в отдельные ячейки таблицы при помощи элемента <fo:table-cell>. Заметьте: чтобы задать шрифт и другие характеристики этого содержимого, внутри каждого элемента <fo:table-cell> можно заключить элемент <fo:block>. При желании вы можете связать ячейку таблицы с определенным столбцом при помощи свойства column-number или даже растянуть ячейку на несколько строк и столбцов, как в таблицах HTML.

С элементом <fo:table-cell> можно применять следующие свойства:

- **общие свойства доступа: source-document, role;**
- **общие звуковые свойства: azimuth, cue-after, cue-before, elevation, pause-after, pause-before, pitch, pitch-range, play-during, richness, speak, speak-header, speak-numeral, speak-punctuation, speech-rate, stress, voice-family, volume;**
- **общие свойства границ, заполнения и заднего фона: background-attachment, background-color, background-image, background-repeat, background-position-horizontal, background-position-vertical, border-before-color,border-before-style, border-before-width, border-after-color, border-after-style, border-after-width, border-start-color, border-start-style, border-start-width, border-end-color, border-end-style, border-end-width, border-top-color, border-top-style, border-top-width, border-bottom-color, border-bottom-style, border-bottom-width, border-left-color, border-left-style, border-left-width, border-right-color, border-right-style, border-right-width, padding-before, padding-after, padding-start, padding-end, padding-top, padding-bottom, padding-left, padding-right;**
- **block-progression-dimension;**

- column-number;
- display-align;
- relative-align;
- empty-cells;
- ends-row;
- height;
- id;
- number-columns-spanned;
- number-rows-spanned;
- relative-position;
- starts-row;
- width.

В tables.fo я поместил в ячейки таблицы и подписи к каждому столбцу, и данные таблицы следующим образом:

```
<fo:table>
    <fo:table-column column-width-"30mm"
    <fo:table-column column-wi dth-"30mm"/>
    <fo:table-column column-w1dth-"30mrn'7>
    <fo: tabl e-col umn column-widtn-’’30mm"/>
    <fo:table-column column-width-"30mm"/>
    <fo:table-column column-width-"30mm"/>

    <fo:table-body>
        <fo:table-row>
            <fo:table-eel1 border-width-"0.5mm">
                <fo:block font-weight*="bold" font-size-"18pt">
                    Name
                </fo:block>
            </fo:table-eel1>
            <fo:table-cell border-wi dth-"0.5mm">
                <fo:block font-weight"'"bold" font-s1ze-"l8pt">
                </fo:block>
                    Mass
            </fo:table-cell>
            <fo:table-cell border-width-"0.5mm">
                <fo:block font-weight-"bold" font-size-"18pt">
                    Day
                </fo:block>
            </fo .-table-cell >
            <fo:table-cell border-width-"0.5mm">
                <fo:block font-weight›"bold" font-s1ze-"l8pt">
                    Radius
                </fo:block>
            </fo:table-cell>
            portable-eel 1 border - wi dth-" 0. 5mm ">
                <fo:block font-we1ght-"bold" font-size="18pt">
                    Density
                </fo:block>
```

```
            </fo:table-cell>
            <fo:table-eel1 border-wi dth-"0.5mm">
                <fo:block font-we1ght-"bold" font-size-"18pt"
                    Distance
                </fo:block>
            </fo:table-cell>
        </fo:table-row>

        <fo table-row>
            <fo:table-cell border-w1dth-"0.5mm">
                <fo:block font-s1ze-"18pt">Mercury</fo:block>
            </fo:table-cell>
            <fo:table-cell border-w1dth-"0.5irm">
                <fo:block font-si ze-"18pt">.0553</fo:block>
            </fo:table-cell>
            <fi:o:table-cel1 border-width-"0.5mm">
                <fo:block font- si ze-"18pt">58.65</fo:block>
            </fo:table-cell>
            <fo:table-eel1 border-w1dth-"0.5mm">
                <fo:block font-si ze-"18pt">1516</fo:block>
            </fo:table-cell>
            <fo:table-eel1 border-w1dth-M0.5mm">
                <fo:block font-s1ze-"18pt">.983</fo:block>
            </fo:table-cell>
            <fo:table-eel1 border-width-"0.5mm">
                <fo:block font-s1ze-"18pt">43.4</fo:block>
            </fo:table-cell>
        </fo-table-row>
        .
        .
        .
    </fo:table-body>
</fo:table>
```

Теперь все готово — результат показан на рис. 11.2, где вы можете увидеть полностью отформатированную таблицу, созданную процессором fop. Теперь вы знаете, как создавать таблицы при помощи форматирующих объектов XSL.

Безусловно, можно еще много сказать о XSL-FO. В следующей главе мы рассмотрим работу со списками, размещение, колонки, последовательности страниц и другие элементы.

12

Документы XSL-FO

В этой главе мы продолжим работу с XSL-FO. Здесь мы рассмотрим работу со списками, рисунками и колонками, размещение текста, создание многостраничных последовательностей и другое. По прочтении этой главы вы усвоите основы работы с XSL-FO — по крайней мере, с той частью, которая реализована в современном программном обеспечении. Я начну главу со списков XSL-FO.

Создание списков XSL-FO

Как мы видели в главе 11, таблицы XSL-FO работают во многом так же, как в HTML, и это в той же мере относится к спискам. Как и в HTML, список XSL выводит вертикальный список элементов. Для создания списков служат четыре элемента XSL-FO:

- **<fo:list-block>;**
- **<fo:list-item>;**
- **<fo:list-item-label>;**
- **<fo:list-item-body>.**

Весь список заключается в элемент <fo:list-block>, и каждый элемент списка — в элемент <fo:list-item>. Для создания подписи к элементу списка служит элемент <fo:list-item-label>, а для вставки данных в каждый элемент списка — элемент <fo:list-item-body>.

Пример lists.xsl (листинг 12.1) преобразует planets.xml в список формата XSL-FO, в котором каждый элемент списка отображает название планеты.

Листинг 12.1. lists.xsl

```
<?xml version-'1.0"?>
<xs1:stylesheet  xmlns:xs1-"http://www.w3.org/1999/XSL/Transform"
```

продолжение &

Листинг 12.1 (продолжение)

```
xml ns: f o~" http;./ /www. w3. org/1999/ XSL /Format"
version-"1.0">

<xsl template match-"PLANETS"
    <fo:root xmlns:fo-"http://www.w3.org/1999/XSL/Format">

        <fo:1ayout-maste r-set>
            <fo:simple-page-master master-name-"page"
                page-height-"400mm" page-width="300mm"
                margin-top-"10mm" margin-bottom-"10mm"
                margin-1eft-"20mm" margin-right="20mm">

                <fo region-body
                    ma rgi n-top-"Omm" ma rg i n-bottom-"1Omm"
                    margin-left-"0mm" margin-right="0mm'7>

                <fo:region-after extent-"10mm"/>
            </fo:simple-page-master>
        </fo:layout-master-set>

        <fo:page-sequence master-name="page">

            <fo:f1ow f1ow-name-"xsl-region-body">

                <fo:block font-size-"24pt"»The Planets Table</fo:block>
                <fo:list-block
                    provisional-di stance-between-starts-"15mm"
                    provisional-label-separation«"5mm"»

                    <xsl:apply-templates/»

                </fo:list-block>
            </fO:flow>
        </fo:page-sequence»

    </fO:TOOt>
</xsl:template»

<xsl:tempi ate match-"PLANET">
    <fo:1ist-item line-height-"20mm">
        <fo:list-item-label>
            <fo:block font-fami1y="sans-seri f"
                font-size-"36pt">
                <xsl:number/»
                <xsl:text>. </xsl:text>
            </fo:block»
        </fo list-item-label»

        <xsl:apply-templates/»
    </fo:list-item>
</xsl:tempiate»

<xsl:template match-"NAME">
    <fo:list-item-body»
        <fo:block font-family-"sans-serif"
            font-size-"36pt"»
            <xsl:value-of select-'.'/»
```

```
            </fo:block>
        </fo:1i st-i tern-body»
    </xsl:tempiate>

    <xsl:template match-"MASS">
    </xsl:tempiate>

    <xsl:template match-"RADIUS">
    </xsl:template»

    <xsl:template match="DENSITY">
    </xsl:tempiate>

    <xsl:template match-"DAY">
    </xsl:tempiate>

    <xsl:tempiate match-"DISTANCE">
    </xsl:template»

</xsl:stylesheet»
```

Вот результирующий документ XSL-FO, Lists.fo (листинг 12.2).

Листинг 12.2. lists.fo

```
<?xml version-"1.0" encoding-"UTF-8"?>
<fo:root xmlns:fo-"http://www.w3.org/1999/XSL/Format">

    <fo:layout-master-set»
        <fo:simple-page-master margin-right-"20mm" margin-left="20mm"
            ma rg i n-bottorn-"1Omm" margin-top="10mm" page-width="300mm"
            page-height-"400mm" master-name="page">

            <fo:region-body margin-right="Omm" margin-left="Omm"
                margin-bottom-"10mm" margin-top="0mm'7>

            <fo:region-after extent-"1Omm"/>

        </fo:simple-page-master>
    </fo:layout-master-set>

    <fo:page-sequence ma ster-name»"page">
        <fo:flow flow-name-"xsl-region-body"»

            <fo:block font-size="24pt"»The Planets Table</fo:block»

            <fo:list-block provisional-label-separation="5mm"
                provisional-distance-between-starts="15mm">

                <fo:1i st-i tem 1i ne-he i ght-"20mm"»
                    <fo:list-item-label»
                        <fo:block font-size-"36pt"
                            font-family="sans-serif"»1. </fo:block»
                    </fo:list-item-label»

                    <f0:1ist-i tem-body»       >
                        <fo:block font-slze-"36pt"
                            font-family-"sans-senf">Mercury</f6:block»
```

продолжение ^

Листинг 12.2 (продолжение)

```
                </fo:1i st-item-body>
            </fo list-item>

            <fo:list-item line-height="20mm">
                <fo:list-item-label>
                    <fo:block font-size-"36pt"
                        font-family-"sans-serif">2  </fo:block>
                </fo:list-item-label>

                <fo:list-item-body>
                    <fo:block font-size-"36pt"
                        font-fami ly»"sans-serif">Venus</fôblock>
                </fo:list-item-body>
            </fo:list-item>

            <fo: 1 i st - i tem1 i ne -hei ght-" 20mm">
                <fo:list-item-label>
                    <fo:block font-size="36pt"
                        font-fami ly^'sans-serif"^. </fo:block>
                </fo:list-item-label>

                <fo list-item-body>
                    <fo:block font-size«"36pt"
                        font-fami 1y-"sans-seri f">Ea rth</fo:ыock>
                </fo:1i st-i tem-body>
            </fo:list-item>

        </fo:list-block>
    </fo:flow>
</fo:page-sequence»
</fo:root>
```

Результирующий файл PDF со списком в Adobe Acrobat Reader показан на рис. 12.1.

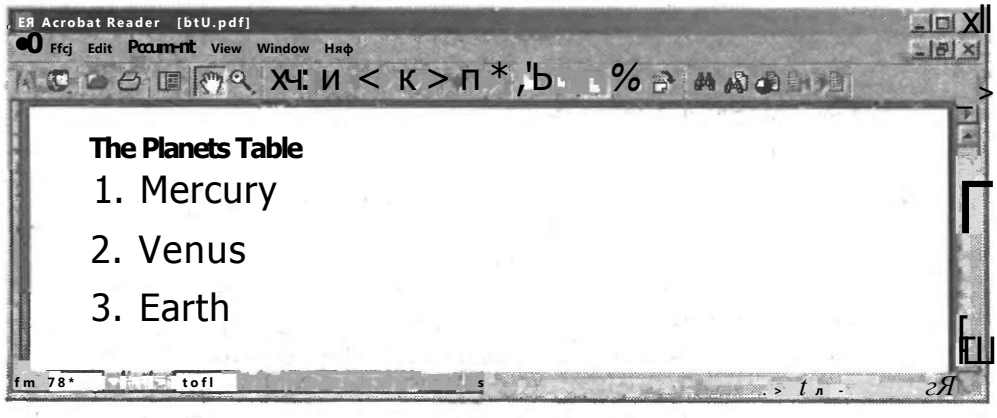


Рис. 12.1. Список, отформатированный при помощи XSL-FO, в Adobe Acrobat Reader

Давайте теперь подробнее рассмотрим, как работает lists.fo, и изучим форматирующие объекты для списка.

Создание списка: <fo:list-block>

Дтя начала воспользуйтесь элементом <fo:list-block>, чтобы создать список XSL-FO; этот объект содержит элементы <fo:list-item>, содержащие данные списка.

С элементом <fo:list-block> можно применять следующие свойства:

- общие свойства доступа: source-document, role;
- общие звуковые свойства: azimuth, cue-after, cue-before, elevation, pause-after, pause-before, pitch, pitch-range, play-during, richness, speak, speak-header, speak-numeral, speak-punctuation, speech-rate, stress, voice-family, volume;
- общие свойства границ, заполнения и заднего фона: background-attachment, background-color, background-image, background-repeat, background-position-horizontal, background-position-vertical, border-before-color, border-before-style, border-before-width, border-after-color, border-after-style, border-after-width, border-start-color, border-start-style, border-start-width, border-end-color, border-end-style, border-end-width, border-top-color, border-top-style, border-top-width, border-bottom-color, border-bottom-style, border-bottom-width, border-left-color, border-left-style, border-left-width, border-right-color, border-right-style, border-right-width, padding-before, padding-after, padding-start, padding-end, padding-top, padding-bottom, padding-left, padding-right;
- общие свойства полей: margin-top, margin-bottom, margin-left, margin-right, space-before, space-after, start-indent, end-indent;
- break-after;
- break-before;
- id;
- keep-together;
- keep-with-next;
- keep-with-previous;
- provisional-distance-between-starts;
- provisional-label-separation;
- relative-position.

Ниже показано, как начинается список в листинге 12.2. В данном случае я задаю значения для свойства provisional-distance-between-starts, определяющего предпочитаемое расстояние от начального отступа метки элемента списка и начального отступа тела элемента списка, и для свойства provisional-label-separation, определяющего предпочитаемое расстояние между концом метки элемента списка (такой, как маркер или номер) и началом тела элемента списка:

```
<fo:1ist-block  provisional-label-separation-"5mm"
                provisional-distance-between-starts-"15mm''>

.
.
.
</fo:list-block>
```

Создание элементов списка: <fo:list-item>

Затем при помощи элемента <fo:list-item> нужно поместить в список метку и тело элемента списка. В каждом элементе списка должен присутствовать один из этих объектов.

С элементом <fo:list-item> можно применять следующие свойства:

- **общие свойства доступа: source-document role;**
- **общие звуковые свойства: azimuth, cue-after, cue-before, elevation, pause-after, pause-before, pitch, pitch-range, play-during, richness, speak, speak-header, speak-numeral, speak-punctuation, speech-rate, stress, voice-family, volume;**
- **общие свойства границ, заполнения и заднего фона: background-attachment, background-color, background-image, background-repeat, background-position-horizontal, background-position-vertical, border-before-color, border-before-style, border-before-width, border-after-color, border-after-style, border-after-width, border-start-color, border-start-style, border-start-width, border-end-color, border-end-style, border-end-width, border-top-color, border-top-style, border-top-width, border-bottom-color, border-bottom-style, border-bottom-width, border-left-color, border-left-style, border-left-width, border-right-color, border-right-style, border-right-width, padding-before, padding-after, padding-start, padding-end, padding-top, padding-bottom, padding-left, padding-right;**
- **общие свойства границ, заполнения и заднего фона: margin-top, margin-bottom, margin-left, margin-right, space-before, space-after, start-indent, end-indent;**
- **break-after;**
- **break-before;**
- **id;**
- **keep-together;**
- **keep-with-next;**
- **keep-with-previous;**
- **relative-align;**
- **relative-position.**

В lists.fo встречается три элемента <fo:list-item>:

```
<fo:11st-block provisional-label-separation="5mm"
    provisional-distance-between-starts="15mrn">

    <fo:11st-item line-height="20mm">
    •
    •

    </fo:list-1tem>
    <fo:11st-1tem 11ne-height="20mm">
    •
    •
    •
```

```
        </fo:l1st-item>

        <fo:list-item 1i ne-hei ght-"20nm">
          .
          .
          .
        </fo:l1st-item>

      </fo:list-block>
```

Теперь можно создать метку и добавить данные в каждый элемент списка.

Создание меток элемента списка: <fo:list-item-label>

Метка для элемента списка создается элементом <fo:(>t-item-label>, при помощи которого можно перенумеровать или пометить дело элемента списка.

К элементу <fo:list-item-label> можно применять следующие свойства:

- общие свойства доступа: source-document, role;
- id;
- keep-together.

В lists.fo я использую элементы <fo:block> для отображения номера планеты в качестве метки элемента списка (lists.xsl находит этот номер при помощи <xsl:number> и вставляет его в lists.fo):

```
<fo list-block provisional-label-separation="5mm"
    provisional-distance^between-starts-'lSmm'^

    <fo:1i st-i tern 1i ne-hei ght="20mm">
      <fo:l1st-item-label>
        <fo:block font-size-"36pt"
        font-fami1y*"sans-serif">
          1.
        </fo:block>
      </fo:list-1tem-label>


    </fo:list-item>

    <fo:list-item line-height-"20mm">
      <fo:1i st-i tem-label>
        <fo:block font-size-"36pt"
          font-fami1y"sans-seri f">
            2.
          </fo:block>
      </fo:1i st-i tem-label>


    </fo:list-item>

    <fo:list-item line height-'20rn">
      <f0:1ist-item-label>
        <fo:block font-s1ze-"36pt"
```

```
                    font-fami1y-"sans-serif">
                        3.
                    </fo:block>
                </fo:list-item-label»
                .
                .
                .
            </fo:list-item»
        </fo<list-block»
```

Все, что осталось, — добавить в этот список сами данные.

Создание тел элементов списка: <fo:list-item-body>

Для включения тела элемента списка служит элемент <fo:list-item-body>. Заметьте, что для форматирования тела элемента списка требуемым вам образом вы можете включить в элемент <fo:list-item-body> объект <fo:block>.

С элементом <fo:list-item-body> можно применять следующие свойства:

- **общие свойства доступа: source-document, role;**
- **id;**
- **keep-together.**

В lists.fo элементы <fo:list-item-body> содержат названия планет:

```
<fo:list-block provisional -label-separation-'5rnfn"
    provi s i ona1-di stanee-between-starts-"15mm">

    <fo: >1 st- i tem 1 i ne-height-*'20mm">
        <fo:list-item-label>
            <fo:block font-s1ze-"36pt"
               font-family«"sans-serif">
                  1.:
            </fO:blOCk>
        «/fo:list-1tem-label>

        <fo:11st-i tem-body>
            <fo:block font-size-"36pt"
                font-fami1y-"sans-serif">
                Mercury
            </fo:block»
        </fo:11st-Item-body»
    </fo list-item»

    <fo:1ist-item 1ine-height-'^Omm"»
        <fo:list-item-label»
            <fo:block font-size-"36pt"
                font-family-"sans-ser1f"»
                2.:
            </fo:block»
        </fo:list-1tem-label>

        <fo:l1st-1tem-body>
            <fp:block font-s1ze-"36pt"
                font•fami1y-"sans-serif"»
```

```
                    Venus
                </fo:block>
            </fo:1ist-item-body>
        </folist-item>

        <fo:list-item line-height-"20mm">
            <fo:list-item-label>
                <fo:block font-size^"36pt"
                    font-family=-"sans-serif">
                        3.:
                </fO:bIOCk>
            </fo:list-item-label>

            <fo:11st-1tem-body>
                <fo:block font-size-"36pt"
                    font-family"sans-ser1f">
                    Earth
                </fo:block>
            </fo:1ist-item-body>
        </fo:list-item>

    </fo:list-block>
```

На этом завершается список, который был показан на рис. 12.1. Работа со списками приемлема в XSL-FO, но я предпочитаю выводить данные при помощи таблиц или простых блоков. Тем не менее, полезно знать, что списки существуют, и ими можно пользоваться.

Размещение текста при помощи контейнеров блоков: <fo:block-container>

Процессоры XSL-FO в одном отношении похожи на браузеры HTML: они вставляют блоки в «поток» (flow) страницы, что означает, что эти блоки могут перемещаться по документу, как в HTML-браузере. С другой стороны, иногда бывает важно поместить элементы в странице на определенные места. И в XSL-FO у вас есть возможность разместить элементы либо в абсолютных координатах страницы, либо в координатах относительно других элементов.

Для демонстрации я воспользуюсь элементом <fo:block-container>, который, как можно предположить, выступает в качестве контейнера блоков. Удобно то, что такие контейнеры блоков поддерживают свойства абсолютного позиционирования, которые сами блоки не поддерживают.

С элементом <fo:block-container> можно применять следующие свойства:

- общие свойства абсолютной позиции: absolute-position, top, right, bottom, left;
- общие свойства границ, заполнения и заднего фона: background-attachment, background-color, background-image, background-repeat, background-position-horizontal, background-position-vertical, border-before-color, border-before-style, border-before-width, border-after-color, border-after-style, border-after-width, border-start-color, border-start-style, border-start-width, border-end-color, border-end-style, border-end-width, border-top-color, border-top-style, border-top-width.

border-bottom-color, border-bottom-style, border-bottom-width, border-left-color, border-left-style, border-left-width, border-right-color, border-right-style, border-right-width, padding-before, padding-after, padding-start, padding-end, padding-top, padding-bottom, padding-left, padding-right;

- общие свойства полей для блоков: margin-top, margin-bottom, margin-left, margin-right, space-before, space-after, start-indent, end-indent;
- block-progression-dimension;
- break-after;
- break-before;
- clip;
- display-align;
- height;
- id;
- inline-progression-dimension;
- keep-together;
- keep-with-next;
- keep-with-previous;
- overflow;
- reference-orientation;
- span;
- width;
- writing-mode.

В следующем примере (листинг 12.3) я использую контейнеры блоков как оболочки для блоков, которые я хочу поместить в определенные места страницы

Листинг 12.3. blockcontainer.fo

```
<?xml version-"1.0" encoding-"UTF-8"?>
<fo:root xml ns: f o-" http: / /www. w3. org/1999/ XSL/Format">

    <fo:1ayout-master-set>
        <fo:simple-page-master margin-right="20mm" margin-left="20mm"
            margin-bottom-"10mm" margin-top-"10mm" page-width-"300mm"
            page-height-"400iimf master-name-"page">

            <fo:region-body margin-right-"Omra" margin-left-"Omm"
                margin-bottom-"10mm" margin-top-"0mm"/>

            <fo:region-after extent-"10mm"/>

        </fo:s i mple-page-master>
    </fo:1ayout-master-set>

    <fo page-sequence master-name-"page">
        <fo:flow flow-name-"xsl-region-body">
            <fo:block-container height-"4cm" width-"12cm" top-"0cm"
                left-"Ocm"
```

```
    pos1tion""absolute">
    <fo:block text-align-"start" line-height-"18pt"
    font-family-"sans-serif" font-weight-"bold" font-size-"14pt">
        Starpowder
    </fo:block»
    <fo:block text-align-"start" line-height-"18pt"
        font-famlly-sans-serif"
        font-size«"14pt"
        The Starpowder Building:
    </fo:block»
    <fo:block text-align-"start" line-height-"18pt"
        font-fami 1y="sans-seri f"
        font-size-"14pt"
        1 Starpowder Avenue
    </fo:block>
    <fo:block text-al1gn-"start" line-height-"18pt"
        font-fami 1y-"sans-serif"
        font-size-"14pt"
        New York. NY. 10011
    </fo:block>
</fo:block-contai ner>

<fo:block-container height-"lcm" width-"6cm" top="0cm"
    left-"14cm"
    position-"absolute"
    <fo:block text-align-"start" line-height-"22pt"
        font-family-"sans-serif"
        font-size-"23pt">
        Invoice
    </fo:block>
</fo:block-container»

<fo:block-container border-color-"black" border-style-"solid"
    border-width-"lpt" height-"0.7cm" width-"3.5cm" top-"1.2cm"
    left-"12.0cm" padding-"2pt" position-"absolute">
    <fo:block text-align"start" line-height-"15pt"
        font-fam1ly-sans-serlf"
        font-size-"12pt"
        Date
    </fo:block>
</fo:block-contai ner>

<fo:block-container border-color-"black" border-style-"solid"
    border-width-"lpt" height-"0.7cm" width-"3.5cm" top-"1.9cm"
    left-"12.0cm" padding-"2pt" posit1on="absolute">
    <fo:block text-align-"start" line-height="15pt"
        font-fami1y-"sans-serif" font-size-"12pt">
        January 1. 2002
    </fo:block>
</fo:block-container»

<fo:block-container border-color-"black" border-style-"solid"
    border-width-"lpt" height-"0.7cm" width-"3.5cm" top-"1.2cm"
    left-"15.5cm" padding-"2pt" position-"absolute"»
    <fo:block text-align«"start" line-height-"15pt"
        font-family-"sans-serif" font-size-"12pt">
        Terms
```

продолжение &

Листинг 12.3 (продолжение)

```
            </fo:block>
        </fo:block-conta i ner>

        <fo:block-container border-color-"black" border-style-"solid"
           border-width="lpt" height-"0.7cm" width-"3.5cm" top="1.9cm"
           1eft-"15.5cm" padding-"2pt" position="absolute">
           <fo:bldck text-align-"start" line-height-"15pt"
               font-family-"sans-serif" font-size="12pt"
               Immediate
           </fo:block>
        </fo:block-conta i ner>

        <fo:block-container border-color-"black" border-style-"solid"
            border-width-"lpt" hei ght-"1.0cm" width-"9cm" top-"3cm"
            left-"Ocm" padding-"2pt" p6sition-"absolute"
            <fo:block text-align-"center" line-height="22pt"
                font-family-"sans-serif" font-size="18pt">
                Description of Service
            </fo:block>
        </fo:block-contai ner>

        <fo block-container border-color-"black" border-style="solid"
            border-width-"lpt" he1ght="4cm" width-"9cm" top-"4.0cm"
            left-"0cm"
            padding-"2pt" position-"absolute">
            <fo:block text-align-"start" line-height-"15pt"
                font-family-"sans-serif" font-size="12pt"
            </fo:block>
        </fo:block-conta i ner>

        <fo:block-container border-color-"black" border-style-"solid"
            border-width-"lpt" height-"1.0cm" width-"9cm" top-"3cm"
            left-"10cm"
            padding-"2pt" position-"absolute">
            <fo:block text-align-"center" line-height-"22pt"
                font-fami 1y-"sans-seri f" font-s i ze-"18pt">
                Address for Payment:
                </fo:block>
        </fo:block-contai ner>

        <fo:block-container border-color-"black" border-style-"solid"
            border-width-"lpt" height-"4cm" width-"9cm" top-"4.0cm"
            left-"10cm"
            padding-"2p,t" position-"absolute"
            <fo:block text-align-"start" line-height-"15pt"
                font-fami 1y="sans-seri f" font-s1ze-"12pt"
            </fo:block>
        </fo:block-contai ner>

    </fo:flow>
</fo:page-sequence>
</fo:root>
```

Документ PDF, который процессор fop создаст из blockcontainers.fo, показан на рис. 12.2. Как можно видеть на рисунке, я поместил различные блоки в разные места документа. Некоторые прямоугольники должны быть размещены сразу же поверх других, поэтому в данном случае размещение играет важную роль.

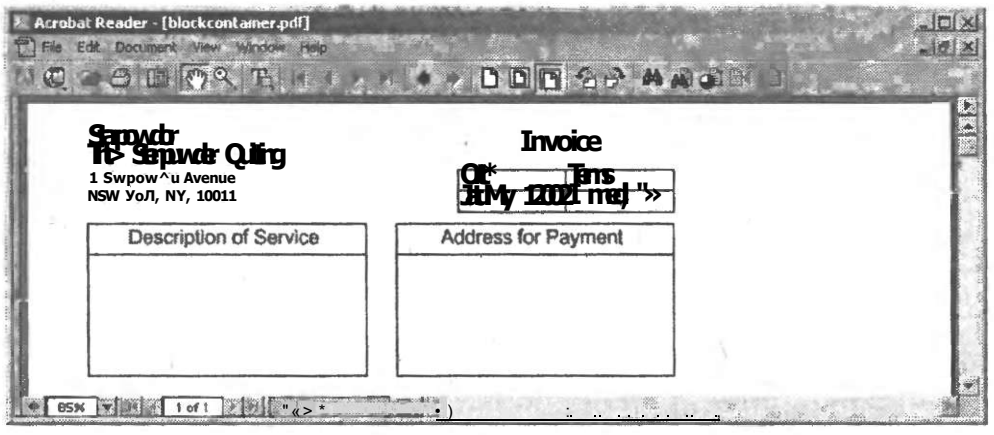


Рис. 12.2. Применение контейнеров блоков

На самом деле работать с контейнерами блоков просто. Взгляните, например, на прямоугольник «Description of Service» (Описание услуги) на рис. 12.2. Я создал его при помощи двух контейнеров блоков: один из них содержит заголовок «Description of Service», а второй — пустой прямоугольник сразу под первым. Для создания прямоугольника заголовка я применил элемент <fo:block-container>, указав размеры прямоугольника в свойствах height (высота) и width (ширина) и задав границу в свойстве border-width. Положение прямоугольника на странице я задал, установив свойство position в absolute и установив свойства left (левый) и top (верхний) для размещения верхнего левого угла прямоугольника по отношению к верхнему левому углу страницы:

```
<fo:block-container border-color-"black" border-style-"solid"
    border-width-"lpt" height"10cm" width-"9cm" top-"3cm"
    left-"Ocm" padding="2pt" position-"absolute">

    .
    .

</fo:block-container>
```

Теперь я могу включить в контейнер блок, содержащий заголовок «Description of Service»:

```
<fo:block-container border-color-"black" border-style-"solid"
    border-width-"lpt" height-"1.0cm" width-"9cm" top«"3cm"
    left-"0cm" padding="2pt" position-"absolute"»
    <fo:block text-align»"center" line-height="22pt"
        font-family-"sans-serif"
        font-size="18pt">
        Description of Service
    </fo:block»
</fo:block-container»
```

После этого я размещаю пустой прямоугольник сразу под прямоугольником заголовка:

```
</fo:block-container»
<fo:block-container border-color-"black" border-style-"solid"
```

```
            border-width«"lpt" height-"1.0cm" width-"9cm" top-"3cm"
            left»"0cm" padding-"2pt" position-"absolute">
            <fo:block text-align-"center" line-height-"22pt"
                font-family«"sans-serif"
                font-size-"18pt">
                Description of Service
            </fo:block>
        </fo:block-contai ner>

        <fo:block-container border-color»"black" border-style-"solid"
            border-width-"lpt" height-"4cm" width-"9cm" top-"4.0cm"
            left-"0cm"
            padding-"2pt" posit1on-"absolute">
            <fo:block text-align®"start" line-height»"15pt"
            font - f ami 1 y" sans - seri f"
            font-size-"12pt">
            </fo:block>
    </fo:block-contai ner>
```

Результат показан на рис. 12.2. Я считаю эту возможность одной из самых важных в XSL-FO: вы можете помещать элементы в точности туда, куда вам нужно. Допустимо полагаться на поток объектов, создаваемый процессором XSL-FO, но иногда требуется расположить объекты в определенном месте, и теперь вы знаете, как это сделать.

Встроенные форматирующие объекты

Помимо объектов блока, в XSL-FO можно также создавать *встроенные* **(inline) объекты. Встроенный объект представляет собой часть большей форматирующей области, такой как блок; например, встроенный объект может представлять одно или два слова в блоке. Встроенные объекты обычно применяются для форматирования части текста после того, как этот текст был размещен на странице в соответствии с обычным потоком. Можно увеличить первый символ в абзаце, выделить всю первую строку голубым цветом, вставить номера страниц, добавить рисунки и т. д.**

Встроенные объекты можно применять для форматирования частей текста блока; например, в главе 11 мы видели, как свойство text-decoration объекта <fo:inline> применялось для подчеркивания текста:

```
    <xslrtemplate match-"PLANET/MASS">
        <fo:block font-size-"24pt" line-height-"32pt"
            font -fami 1y-"Ti mes">
            <fo:inline text-decoration-"underline">
                Mass;
            </fo:inline>:
            <xsl:apply-templates/>
            [Earth - 1]
        </fo:block>
</xsl:tempiate>
```

В следующем списке перечислены встроенные форматирующие элементы:

- <fo:bidi-override>;
- <fo:character>;
- <fo:initial-property-set>;
- <fo:external-graphic>;
- <fo:instream-foreign-object>;
- <fo:inline>;
- <fo:inline-container>;
- <fo:leader>;
- <fo:page-number>;
- <fo:page-number-citation>.

Я опишу несколько самых распространенных встроенных элементов, начав с самого <fo:inline>.

Создание встроенных областей: <fo:inline>

Как вы уже видели в главе 11, при помощи элемента <fo:inline> вы можете форматировать части текста, задавая для них задний фон, подчеркивая текст или заключая текст в границы. Элемент позволяет форматировать встроенную область из нескольких слов в блоке текста как если бы они сами представляли собой блок.

С элементом <fo:inline> можно применять следующие свойства:

- общие свойства доступа: source-document, role;
- общие звуковые свойства: azimuth, cue-after, cue-before, elevation, pause-after, pause-before, pitch, pitch-range, play-during, richness, speak, speak-header, speak-numeral, speak-punctuation, speech-rate, stress, voice-family, volume;
- общие свойства границ, заполнения и заднего фона: background-attachment, background-color, background-image, background-repeat, background-position-horizontal, background-position-vertical, border-before-color, border-before-style, border-before-width, border-after-color, border-after-style, border-after-width, border-start-color, border-start-style, border-start-width, border-end-color, border-end-style, border-end-width, border-top-color, border-top-style, border-top-width, border-bottom-color, border-bottom-style, border-bottom-width, border-left-color, border-left-style, border-left-width, border-right-color, border-right-style, border-right-width, padding-before, padding-after, padding-start, padding-end, padding-top, padding-bottom, padding-left, padding-right;
- общие свойства шрифтов: font-family, font-size, font-stretch, font-size-adjust, font-style, font-variant, font-weight;
- общие свойства встроенных полей: space-end, space-start;
- alignment-adjust;
- baseline-identifier;

- baseline-shift;
- color;
- dominant-baseline;
- id;
- keep-together;
- keep-with-next;
- keep-with-previous;
- line-height;
- line-height-shift-adjustment;
- relative-position;
- text-decoration;
- visibility;
- z-index.

Например, мы уже видели, как подчеркнуть отдельное слово внутри остального текста при помощи свойства text-decoration:

```
<xsl:tempi ate match-"PLANET/MASS">
        <fo:block font-size="24pt" line-height="32pt"
            font-fami 1y-'"'Times">
            <fo:1nline text-decoration="underline">
                Mass
            </fo:1nline>:
            <xsl:apply-templates/>
            CEarth - 1]
        </fo block>
    </xsl:tempiate>
```

Мы еще встретим элемент <fo:inline>, когда будем рассматривать сноски позже в этой главе.

Обработка отдельных символов: <fo:character>

Как можно догадаться по имени, объект <fo:character> позволяет работать с отдельными символами в документе. Одна из областей применения <fo:character> — случай, когда нужно заменить определенные символы на другие. В следующем примере я выбираю элемент с именем <PASSWORD> и заменяю его символы на символ «*»:

```
<xsltemplate match-"PASSWORD">
    <fo:character character»"*"»
        <xsl:value-of select-"."/>
    </fo:character»                                          \ix
</xsl:tempiate>
```

При помощи элемента <fo:character> можно форматировать отдельные символы, как в следующем случае, где символы в слове «Hello» выделяются различными цветами:

```
<fo:character character="H" font-size-"24pt" co1or-"red"/>
<fo:character character-"E" font-size-"24pt" color-"yenow"/>
<fo:character character-"L" font-size-"24pt" color«"green"/>
<focharacter character-"L" font-size-"24pt" co1or-"blue'7>
<fo:character character-"0" font-size-"24pt" color-"orange"/>
```

С элементом <fo:character> можно применять следующие свойства:

- **общие звуковые свойства: azimuth, cue-after, cue-before, elevation, pause-after, pause-before, pitch, pitch-range, play-during, richness, speak, speak-header, speak-numeral, speak-punctuation, speech-rate, stress, voice-family, volume;**
- **общие свойства границ, заполнения и заднего фона: background-attachment, background-color, background-image, background-repeat, background-position-horizontal, background-position-vertical, border-before-color, border-before-style, border-before-width, border-after-color, border-after-style, border-after-width, border-start-color, border-start-style, border-start-width, border-end-color, border-end-style, border-end-width, border-top-color, border-top-style, border-top-width, border-bottom-color, border-bottom-style, border-bottom-width, border-left-color, border-left-style, border-left-width, border-right-color, border-right-style, border-right-width, padding-before, padding-after, padding-start, padding-end, padding-top, padding-bottom, padding-left, padding-right;**
- **общие свойства шрифтов: font-family, font-size, font-stretch, font-size-adjust, font-style, font-variant, font-weight;**
- **общие свойства переносов: country, language, script, hyphenate, hyphenation-character, hyphenation-push-character-count, hyphenation-remain-character-count;**
- **общие свойства встроенных полей: space-end, space-start;**
- **alignment-adjust;**
- **treat-as-word-space;**
- **baseline-identifier;**
- **baseline-shift;**
- **character;**
- **color;**
- **dominant-baseline;**
- **font-height-override-after;**
- **font-height-override-before;**
- **glyph-orientation-horizontal;**
- **glyph-orientation-vertical;**
- **id;**
- **keep-with-next;**
- **keep-with-previous;**
- **letter-spacing;**
- **line-height;**

- line-height-shift-adjustment;
- relative-position;
- score-spaces;
- suppress-at-line-break;
- text-decoration;
- text-shadow;
- text-transform;
- word-spacing.

Создание номеров страниц: <fo:page-number>

Встроенный форматирующий объект <fo:page-number> создает встроенную область, отображающую номер текущей страницы. Например:

```
<fo:block>
    You are now reading page <fo:page-number/>.
</fo:block>
```

С элементом <fo:page-number> можно применять следующие свойства:

- общие свойства доступа: source-document, role;
- общие звуковые свойства: azimuth, cue-after, cue-before, elevation, pause-after, pause-before, pitch, pitch-range, play-during, richness, speak, speak-header, speak-numeral, speak-punctuation, speech-rate, stress, voice-family, volume;
- общие свойства границ, заполнения и заднего фона: background-attachment, background-color, background-image, background-repeat, background-position-horizontal, background-position-vertical, border-before-color, border-before-style, border-before-width, border-after-color, border-after-style, border-after-width, border-start-color, border-start-style, border-start-width, border-end-color, border-end-style, border-end-width, border-top-color, border-top-style, border-top-width, border-bottom-color, border-bottom-style, border-bottom-width, border-left-color, border-left-style, border-left-width, border-right-color, border-right-style, border-right-width, padding-before, padding-after, padding-start, padding-end, padding-top, padding-bottom, padding-left, padding-right;
- общие свойства шрифтов: font-family, font-size, font-stretch, font-size-adjust, font-style, font-variant, font-weight;
- общие свойства встроенных полей: space-end, space-start;
- alignment-adjust;
- baseline-identifier;
- baseline-shift;
- dominant-baseline;
- id;
- keep-with-next;

- keep-with-previous;
- letter-spacing;
- line-height;
- line-height-shift-adjustment;
- relative-position;
- score-spaces;
- text-decoration;
- text-shadow;
- text-transform;
- word-spacing.

Пример применения номеров страниц будет показан при обсуждении последовательностей страниц в конце этой главы.

Вставка графики: <fo:external-graphic>

Популярный элемент при форматировании XSL-FO — <fo:external-graphic>, при помощи которого можно вставить в документ рисунок.

С элементом <fo:external-graphic> можно применять следующие свойства:

- общие свойства доступа: source-document, role;
- общие звуковые свойства: azimuth, cue-after, cue-before, elevation, pause-after, pause-before, pitch, pitch-range, play-during, richness, speak, speak-header, speak-numeral, speak-punctuation, speech-rate, stress, voice-family, volume;
- общие свойства границ, заполнения и заднего фона: background-attachment, background-color, background-image, background-repeat, background-position-horizontal, background-position-vertical, border-before-color, border-before-style, border-before-width, border-after-color, border-after-style, border-after-width, border-start-color, border-start-style, border-start-width, border-end-color, border-end-style, border-end-width, border-top-color, border-top-style, border-top-width, border-bottom-color, border-bottom-style, border-bottom-width, border-left-color, border-left-style, border-left-width, border-right-color, border-right-style, border-right-width, padding-before, padding-after, padding-start, padding-end, padding-top, padding-bottom, padding-left, padding-right;
- общие свойства встроенных полей: space-end, space-start;
- alignment-adjust;
- baseline-identifier;
- baseline-shift;
- block-progression-dimension;
- content-height;
- content-type;
- content-width;

- dominant-baseline;
- height;
- id;
- inline-progression-dimension;
- keep-with-next;
- keep-with-previous;
- line-height;
- line-height-shift-adjustment;
- relative-position;
- overflow;
- scaling;
- scaling-method;
- sre;
- width.

Как в HTML, можно устанавливать размер рисунка в документе — в XSL-FO для этого служат свойства content-height, content-width и scaling; если их не задать, рисунок будет отображен в своем первоначальном размере. Следующий пример, graphics.fo (листинг 12.4), отображает рисунок, xslfo.jpg, и подпись к нему:

Листинг 12.4. graphics.fo

```
<?xml version-"1.0" encoding-"UTF-8"?>
<fo:root xmlns:fo«"http://www.w3.org/1999/XSL/Format"»

    <fo:1ayout-master-set»
        <fo:simple-page-master margin-right="20mm"
            marg1n-1eft-"20mm" margi n-bottom-"10mm"
            margin-top-"10mm" page-width-"3G0mm"
            page-height-"400mm" master-name-"page">

            <fo: region-body margin-right-^nm"
                margin «left-"Omm" margin-bottom-" 10mm"
                margin-top-"0mm'7>

         <fo:region-after extent-"10mm"/>

        </fo:simple-page-master>
    </fo:1ayout-master-set>

    <fo:page-sequence master-name-"page">
        <fo:f1ow f1ow-name-"xs1 -regi on-body">

            <fo:block space-after-"12pt" font-weight-"bold"
                font-size-"36pt" text-align-"center">
                Using Graphics
            </fo:block»

            <fo:block text-align-"center">
                <fo:external-graphic src-"file:xslfo jpg"/>
            </fo:block»
```

```
        <fo:block space-before«"10pt" text-align="center"
            font-size-"24pt">
            An image embedded in a document.
        </fo:block>

    </foflow>
  </fo:page-sequence»
</fo:root>
```

Документ PDF, созданный из graphics.fo, показан на рис. 12.3.

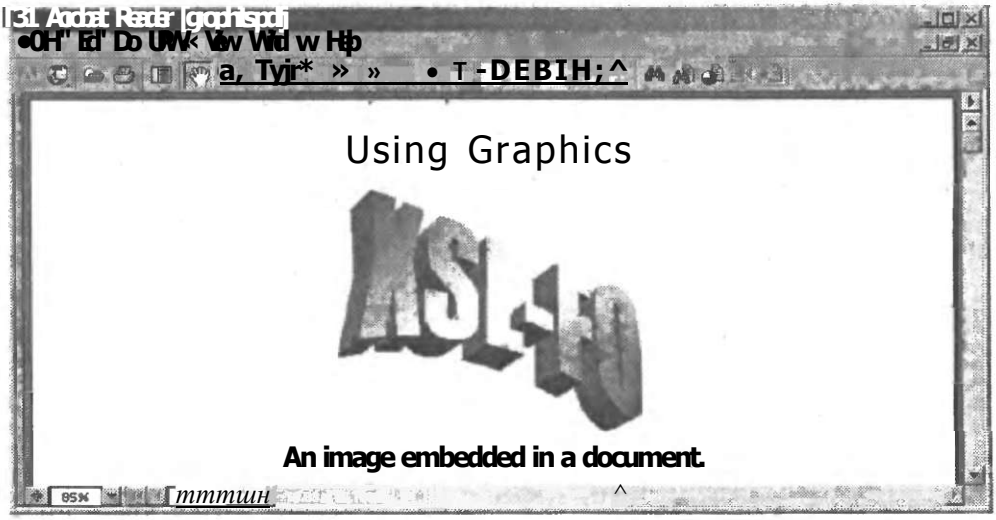


Рис. 12.3. Отображение рисунка при помощи форматирующих объектов

Вставить рисунок в результирующий документ легко, если вставку поддерживает ваше программное обеспечение. Процессор fop теперь поддерживает рисунки (еще несколько версий назад он этого не делал), поэтому вы можете вставлять рисунки при помощи элемента <fo:external-graphic> следующим образом:

```
<fo block text-align«"center">
    <fo:external-graphic src="fi1e:xslfo.jpg"/>
</fo:block>
```

Форматирование первых строк: <fo:initial-property-set>

При помощи элемента <fo:initial-property-set> вы можете задать свойства для первой строки в блоке и отформатировать ее. В следующем примере я форматирую первую строку блока капителью (small caps) при помощи свойства font-variant (которое, кстати, fop не поддерживает):

```
<fo:block>
    <fo:initial-property-set  font-variant-''small-caps'7>
        This text will be displayed in the result document.
        The first line will be displayed using small caps.
</fo:block>
```

С элементом <fo:initial-property-set> можно применять следующие свойства:

- общие свойства доступа: source-document, role;
- общие звуковые свойства: azimuth, cue-after, cue-before, elevation, pause-after, pause-before, pitch, pitch-range, play-during, richness, speak, speak-header, speak-numeral, speak-punctuation, speech-rate, stress, voice-family, volume;
- общие свойства границ, заполнения и заднего фона: background-attachment, background-color, background-image, background-repeat, background-position-horizontal, background-position-vertical, border-before-color, border-before-style, border-before-width, border-after-color, border-after-style, border-after-width, border-start-color, border-start-style, border-start-width, border-end-color, border-end-style, border-end-width, border-top-color, border-top-style, border-top-width, border-bottom-color, border-bottom-style, border-bottom-width, border-left-color, border-left-style, border-left-width, border-right-color, border-right-style, border-right-width, padding-before, padding-after, padding-start, padding-end, padding-top, padding-bottom, padding-left, padding-right;
- общие свойства шрифтов: font-family, font-size, font-stretch, font-size-adjust, font-style, font-variant, font-weight;
- color;
- id;
- letter-spacing;
- line-height;
- tine-height-shift-adjustment;
- relative-position;
- score-spaces;
- text-decoration;
- text-shadow;
- text-transform;
- word-spacing.

На этом мы заканчиваем обзор встроенных форматирующих объектов; в следующих разделах мы рассмотрим подстрочный форматирующий объект — сноски.

Создание сносок: <fo:footnote> и <fo:footnote-body>

Сноски называют «подстрочными» (out-of-line) форматирующими объектами, потому что они добавляют текст в нижнюю часть страницы. Сноска создается при помощи элемента <fo:footnote>, а текст тела сноски — при помощи элемента <fo:footnote-body>. В текущей спецификации XSL-FO эти элементы не поддерживают никаких свойств.

В следующем примере, footnotes.fo (листинг 12.5), я использую две сноски в теле документа и добавляю в них текст.

Листинг 12.5. footnotes.fo

```
<?xml version-"1.0" encoding«"UTF-8"?>
<f|o:root xmlns:fo-"http://www.w3.org/1999/XSL/Format">

    <fo 1ayout-master-set>
        <fo:simple-page-master margin-right="20mm" margin-left-'^Omm"
            margin-bottom-"10mm" margin-top-"10mm" page-width-"300mm"
            page-height-'1400mm" master-name-"page">

            <fo:region-body margin-right-"0mm" margin-left-"0mm"
                margiл-bottom-"10mm" margin-top-"0mm"/>

            <fo:regi on-after extent-"10mm"/>

        </fo simple-page-master>
    </fo:layout-master-set>

    <fo:page-sequence master-name-"page">
        <fo:flow flow-name="xsl-region-body">

            <fo:block space-after-"12pt" font-weight="bold"
                font-size-"36pt" text-align-"center">
                Using Footnotes
            </fo:block>

            <fo:block font-size-"24pt">
                This:
                <fo:footnote>
                    <fo:inline>footnote
                        <fo:inline font-size-"16pt"
                        vertical-align-"super">
                            1
                        </fo:inline>
                    </fo:inline>
                    <fo footnote-body>
                        <fo:block>
                            1. Here's the first footnote's text.
                        </fo:block>
                    </fo:footnote-body>
                </fo:footnote>
                  refers to text at the bottom of this page.
            </fo:block>

            <fo:block font-size-"24pt">
                This second;
                <fo:footnote>
                    <fo:inline>footnote
                        <fo:inline font-size-"16pt"
                        vert1ca1 -a1i gn-"super">
л.                          2
R                       </fo:inline>
                    </fO:inline>
V                   <fo. footnote - body>
-i,                     <fo:block>
```

продолжение &

Листинг 12.5 (продолжение)

```
                    2. And here's the second footnote's text.
                  </fo:block>
                </fo:footnote-body>
              </fo:footnote>
              also refers to text at the bottom of this page.
            </fo:block>

        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

Текст, дополненный верхними индексами для сносок 1 и 2, показан на рис. 12.4. Текст двух сносок внизу той же самой страницы показан на рис. 12.5.

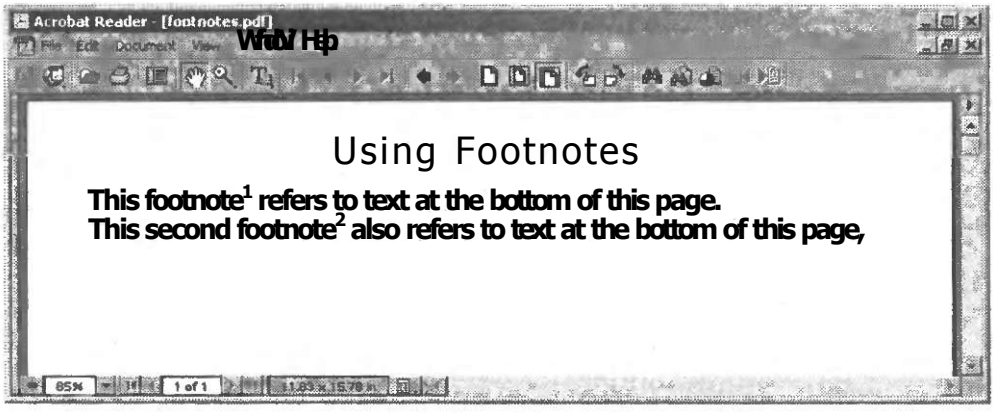


Рис. 12.4. Текст со сносками

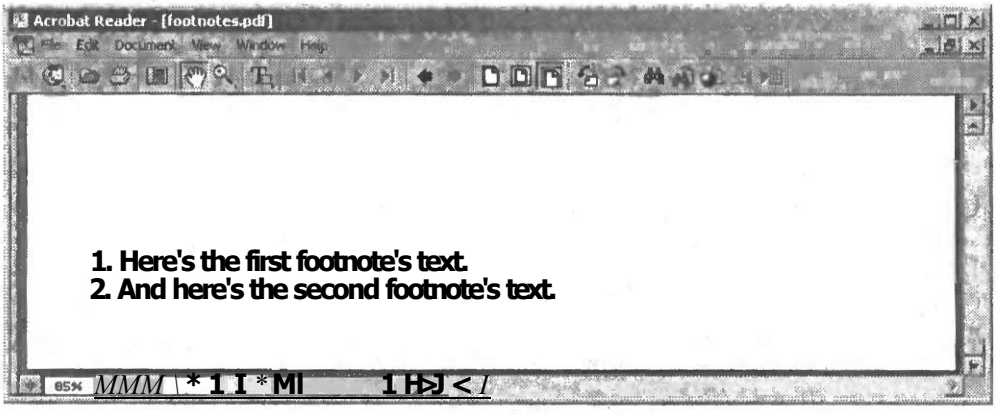


Рис. 12.5. Текст сносок внизу страницы

Для создания текста сносок применяется элемент <fo:footnote>. При работе с процессором fop вы должны самостоятельно добавлять фактический номер сноски в верхнем индексе — такой, как 1 или 2. Вот как это выглядит в footnotes.fo,

где для создания верхнего индекса я установил свойство vertical-align встроенного элемента в «super»:

```
<fo:block font-size-"24pt">
                This:
                <fo:footnote»
                    <fo:i nli ne»footnote
                        <fo:i nli ne font-si ze""16pt" verti cal-align-"super"»l</fo:i nli ne>
                    </fo:inline*


            , </fo:block>
```

Чтобы создать текст сноски, появляющийся внизу страницы, нужно применить элемент <fo:footnote-body> внутри элемента <fo:footnote>. Сам процессор fop не нумерует сноски, поэтому за добавление номеров сносок отвечаете вы:

```
<fo bl ock font-s i ze-"24pt"»
    This
    <fo:footnote»
        <fo:i nli ne>footnote
            <fo:i nline font-s i ze-"16pt"
            vertical-ali gn-"super">
                1
            </fo:inline»
        </fo:inline»
        <fo:footnote-body»
            <fo:block»
                1. Here's the first footnote's text.
            </fo:block»
        </fo:footnote-body»
    </fo footnote»
      refers to text at the bottom of this page.
</fo bl ock>
```

Этот код на XSL-FO создает и первый текст со сносками на рис. 12.4, и сам текст первых сносок, показанный на рис. 12.5.

Создание ссылок: <fo:basic-link>

Еще одним мощным средством XSL-FO является возможность использовать ссылки от документа к документу. В данный момент эта возможность поддерживается в процессоре fop при помощи элемента XSL-FO <fo:basic-link>, который действует во многом похоже на обычную гиперссылку.

С элементом <fo:basic-link> можно применять следующие свойства:

- общие свойства доступа: source-document, role;
- общие звуковые свойства: azimuth, cue-after, cue-before, elevation, pause-after, pause-before, pitch, pitch-range, play-during, richness, speak, speak-header, speak-numeral, speak-punctuation^speech-rate, stress, voice-family, volume;

- общие свойства границ, заполнения и заднего фона: background-attachment, background-color, background-image, background-repeat, background-position-horizontal, background-position-vertical, border-before-color, border-before-style, border-before-width, border-after-color, border-after-style, border-after-width, border-start-color, border-start-style, border-start-width, border-end-color, border-end-style, border-end-width, border-top-color, border-top-style, border-top-width, border-bottom-color, border-bottom-style, border-bottom-width, border-left-color, border-left-style, border-left-width, border-right-color, border-right-style, border-right- width, padding-before, padding-after, padding-start, padding-end, padding-top, padding-bottom, padding-left, padding-right;
- общие встроенные свойства встроенных полей: space-end, space-start;
- общие свойства относительного расположения: top, right, bottom, left, relative-position;
- alignment-adjust;
- alignment-baseline;
- baseline-shift;
- destination-placement-offset;
- dominant-baseline;
- external-destination;
- id;
- indicate-destination;
- internal-destination;
- keep-together;
- keep-with-next;
- keep-with-previous;
- line-height;
- line-height-shift-adjustment;
- show-destination;
- target-processing-context;
- target-presentation-context;
- target-stylesheet.

Следующий пример, links.fo (листинг 12.6), содержит ссылку на документ PDF, созданный ранее в этой главе, graphics.pdf.

Листинг 12.6. links.fo

```
<?xml version-"1.0" encoding-"UTF-8"?>
<fo:root xmlns:fo-"http://www.w3.org/1999/XSL/Format">

    <fo:layout-master-set>
        <fo:simple-page-master margin-right-"20mm" margin-feft-"20mm"
            margin-bottom="lGmm" margin-top^'lOmm" page-witfth-"300mm"
```

```
            page-height-"400mm" master-name-"page">

            <fo:region-body margin-right-"Omm" margin-1 eft-"Omm"
                margin-bottom-"10mm" margin-top="0mm"/>

            <fo:region-after extent-"10mm"/>

        </fo simple-page-master»
    </fo:layout-master-set»

    <fo:page-sequence master-name-"page">
        <fo:flow flow-name="xsl-region-body">

            <fo:block space-after-"12pt" font-weight-"bold"
                font-size="36pt"
                text-a1i gn-"center">
                Using Links
            </fo:block»

            <fo:block font-size-"24pt"»
                If you'd like to see some images, click
                <fo basic-link external-destination-"graphics.pdf"
                    <fo:i nli ne text-decorati on-"under1i ne"»here</fo:i nli ne»
                </fo:basic-link».
            </fo:block»

        </fo:flow>
    </fo:page-sequence»
</fo:root>
```

Документ PDF, созданный на основе links.fo, показан на рис. 12.6. Заметьте, что я явно подчеркнул ссылку, чтобы она была больше похожа на гиперссылку. Когда в Adobe Acrobat вы помещаете указатель мыши над ссылкой, он меняет свою форму и становится в точности такой формы, которую Internet Explorer использует для гиперссылок, как показано на рис. 12.6. При щелчке на ссылке Acrobat переходит на graphics.pdf (который для работы примера нужно поместить в тот же каталог, что и links.pdf), отображая этот новый документ.

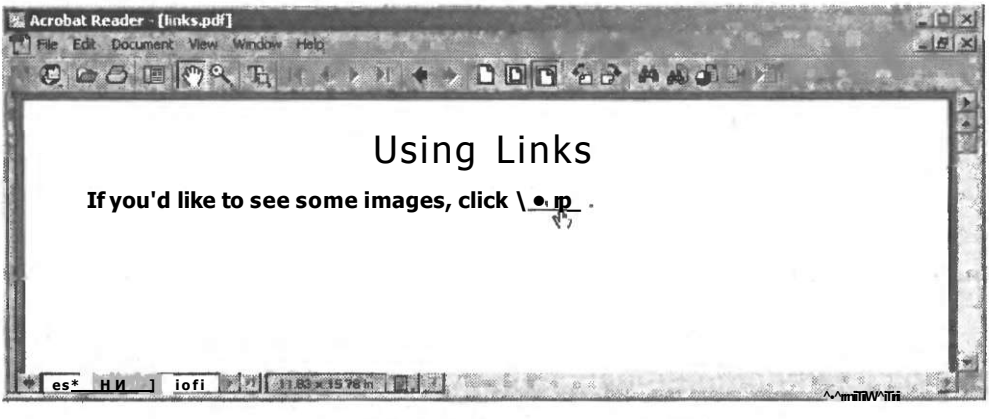


Рис. 12.6. **Поддержка ссылок в XSL-FO**

В данном случае создать ссылку очень просто — достаточно установить свойство external-destination элемента <fo:basic-link> в «graphics.pdf»:

```
<fo:block font-size-"24pt"
              If you'd like to see some images, click
              <fo:basic-link external-destination»"graphics.pdf">
              .
              .
              .
              </fo:basic-link>.
          </f0:bl0Ck>
```

Теперь осталось только добавить подчеркнутый текст, чтобы пользователь мог щелкнуть по нему и перейти к новому документу. Конечно, не обязательно выделять текст подчеркиванием; но иначе ничто не будет указывать на то, что текст является ссылкой, — за исключением указателя мыши, который изменит форму при перемещении над текстом. Подчеркнутый текст можно добавить следующим образом:

```
<fo:block font-size="24pt">
              If you'd like to see some images, click
              <fo:basic-1i nk external-destinati on-"graphics.pdf">
                  <fo:i nli ne text-decorati on-"underli ne">here</fo:i nl1ne>
              </fo:basic-link>.
          </fo:block>
```

Пока в XSL-FO реализованы только обычные ссылки (basic link), как показано в примере. Но если вы знакомы со спецификациями XML для XPointer и XLink, вам следует ожидать в будущем значительного развития в этой области.

Создание колонок

При помощи XSL-FO можно разбить текст на колонки. Создать колонки несложно, нужно только воспользоваться свойствами column-count и column-gap элемента <fo:region-body> и предоставить все остальное процессору XSL-FO.

Давайте рассмотрим пример, columns.fo. Чтобы разбить текст на две колонки с расстоянием в четверть дюйма между ними, я установил свойство column-count в «2», а свойство column-gap в «0.25in»:

```
<?xml version-"1.0" encoding="utf-8"?>
<fo:root xmlns fo»"http://www.w3.org/1999/XSL/Format">

    <fo:layout-master-set>
        <fo:simple-page-master margin-right="20mm" margin-left="20mm"
            margin-bottom-"10mm" margin-top-"10irni" page-width-"300mm"
            page-height-"400mm" master-name-"page">

            <fo:region-body margin-top»"3cm" margin-bottom-"2cm"
                CDlumn-C0unt-"2" column-gap»"0.25in"/>

            <fo:region-before extent-"3cm"/>
            <fo:region-after extent-"2cm"/>
```

```
        </fo simple-page-master»
    </fo:layout-master-set>
```

Теперь осталось только вывести в этих колонках какой-либо текст; заметьте, что я также создаю заголовок при помощи элемента <fo:static-content>, о котором будет еще сказано в следующем разделе (листинг 12.7).

Листинг 12.7. columns.fo

```
<?xml version-"1.0" encoding-'^tf-e"?»
<fo:root xmlns;fo-"http://www.w3.org/1999/XSL/Format">

    <fo layout-master-set>
        <fo:simple-page-master margin-right-"20mm" margin-left="20mm"
            margin-bottom-"10mm" margin-top-"10mm" page-width="300mm"
            page-height-"400mm" master-name="page">

            <f0:region-body margin-top-"3cm" margin-bottom="2cm"
                column-count-"2" column-gap-"0.25ln'7>

            <fo region-before extent-"3cm"/>
            <fo:region-after extent-"2cm'7>

        </fo:simple-page-master»
    </fo:1ayout-master-set>

    <fo:page-sequence ma ster-name-"page">

        <fo:static-content flow-name-"xsl-region-before">
            <fo:block text-align-"center" font-size="36pt"
                font-family-"sans-serif"
                line height-"48pt">
                Creating Columns
            </fo:block»
        </fo:static-content»

        <fo:f 1 ow flow-name-"xsl-region-body"»
        <fo:block font-size»"24pt" font-family-"sans-serif" space-a fter="15pt"»
                Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
                Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
                Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
                Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
                Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
                Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
                Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
                Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
                Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
                Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
                Sample Text, Sample Text. Sample Text. Sample Text. Sample Text.
                Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
```

продолжение

Листинг 12.7 (продолжение)

```
                Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
                Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
                Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
                Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
                Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
                Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
                Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
                Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
                Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
                Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
                Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
            </fo:block>
        </f0:f10W>
    </fo:page-sequence»
</fo:root>
```

Результат обработки этого документа, columns.pdf, показан на рис. 12.7.

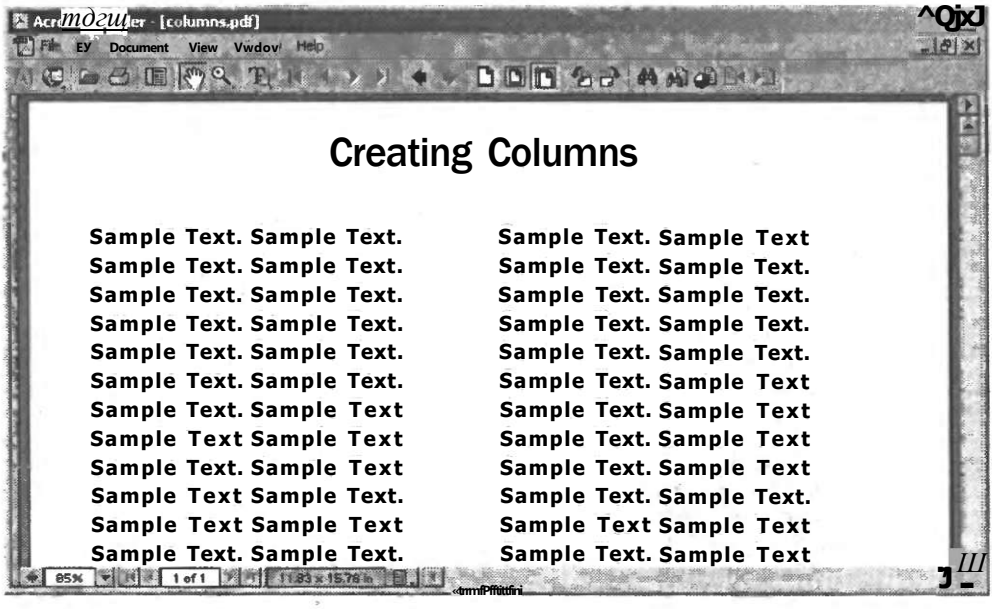


РИС. 12.7. Создание колонок при помощи XSL-FO

Последовательности страниц и нумерация страниц

Пока я использовал один и тот же **шаблон** страницы для всех страниц в рассмотренных нами документах **XSL-FO.** Если содержимое документа занимает **более одной** страницы, процессор **XSL-FO** использует тот же **шаблон** страницы для всех последующих страниц.

Но вам может быть нужно использовать различные шаблоны страницы в разных местах документа. Например, первую страницу нужно отформатировать отлично от следующих. При помощи XSL-FO это можно сделать.

Каждый элемент <fo:page-sequence>, который я применял во всех примерах XSL-FO, ссылается либо на шаблон страницы, либо на элемент <fo:page-sequence-master>. При помощи элемента <fo:page-sequence-master> вы можете задать различные шаблоны страницы для последовательности страниц.

Пример pages.fo продемонстрирует вышесказанное. Я создам для первой страницы один простой шаблон страницы, «first» (первая), в котором текстовая часть смещена вниз страницы, для чего я устанавливаю свойство margin-top элемента <fo:region-body> в «50шт»:

```
<?xml version-"1.0" encoding-"utf-8"?>
<fo:root xmlns:fo-"http://www.w3.org/1999/XSL/Format"»

  <fo:layout-master-set>
       <fo:simple-page-master margin-right-"20mm" margin-left="20mm"
           margin-bottom-"10mm" margin-top-"lOmm" page-width-"300mm"
           page-height-"400mm" master-name-"first">

           <fo:region-body margin-right-"Omm" margin-left-"0mm"
               margin-bottom-"10mm" margi n-top="50mm"/>

           <fo:region-after extent-"10mm"/>
           <fo:regi on-before extent-"10mm"/>

      . </fo:simple-page-master>
```

Для всех остальных страниц я создам новый шаблон страницы, «rest» (остальные), в котором текст будет начинаться сверху страницы, для чего я устанавливаю свойство margin-top элемента <fo:region-body> в «20т*п»:

```
<?xml version-"!.0" encoding="utf-8"?>
<fo:root xmlns:fo-"http://www.w3.org/1999/XSL/Format">

  <fo:layout-master-set>
       <fo:simple-page-master margin-right="20mm" margin-Teft-"20mm"
           margin-bottom="10mm" margin-top-"10mm" page-widthf"30Qmm"
           page-height="400mrti" master-name-"first">

       </fo:simple-page-master>

       <fo:simple-page-master margin-r1ght-"25mm" margin-1e№""25тт"
           ma rgi n-bottom-"15nm" margin-top-"15mm" page-width="300mm"
           page-height-"400mm" master- name-".rest">

           <fo:region-body ma rgi n *right-"0mm" marg1n-left-"0mm";
               margin-bottom-"10mm" margin-top»"20mm"/>
```

```
        <fo:region-after extent-"10mm"/>
        <fo:region-before extent-"10mm"/>

    </fo:simple-page-master»
```

Чтобы создать шаблон последовательности страниц, использующий простые шаблоны страниц «first» и «rest», я применяю элемент <fo:page-sequence-master>:

```
<?xml version-"! O" encoding-"utf-8"?>
<fo:root  xmlns:fo-"http://www.w3.org/1999/XSL/Format">

  <fo layout-master-set>
      <f0:simple-page-master margin-right-"20mm" margin-left-"20mm"
          margin-bottom-"10mm" margin-top-"lQmm" page-width-"300mm"
          page-height-"400mm" master-name-"first">
          .
          .
          .
      </fo:simple-page-master>

      <fo:simple-page-master margin-rigtit-"25mm" marg1n-left-"25mm"
          margin-bottom="15mm" margin-top-"15mm" page-width-"300mm"
          page-height-"400mm" master-name-"rest">
          .
          .
          .
      </fo:simple-page-master>

      <fo:page-sequence-master master-name-"sequence" >
          .
          .
          .
      </fo:page-sequence-master»

  </fo:1ayout-master-set>
```

С элементом <fo:page-sequence-master> вы можете применять следующее свойство:

- master-name.

Здесь я просто назвал новую последовательность страниц «sequence» (последовательность). Я создаю шаблон последовательности страниц повторяющегося типа, и для задания имен шаблонов страниц в новой последовательности служит элемент <fo:repeatable-page-master-alternatives>:

```
    <fo:page-sequence-master master-name-"sequence">
        <fo:repeatable-page-master-alternati ves>
            .
            .
            .
        </fo:repeatable-page-master-alternati ves>
    </fo:page-sequence-master»
```

Очередное свойство элемента <fo:repeatable-page-master-alternatives> определяет, сколько раз полагается повторять последовательность:

- maximum-repeats.

Наконец, надо задать шаблоны страниц, которые будут использованы в шаблоне последовательности, при помощи элемента <fo:conditional-page-master- reference». Этот элемент ссылается на шаблон страницы при помощи свойства master-name, и этот шаблон страницы будет использован при выполнении определенного условия. Чтобы выбрать первую страницу, нужно установить свойство page-position в «first», а для выбора остальных страниц — в «rest»:

```
<fo page-sequence-master master-name»'sequence" >
    <fo:repeatable-page-master-a1ternat1ves>
<fo:conditional-page-master-reference master-name-"first"
                page-posit1or»-"first" />
            <fo:conditional-page-master-reference master-name-"rest"
                page-posl t1on-"resf7>
        </fo:repeatable-page-master-a1ternati ves>
    </fo:page-sequence-master>
```

С элементом <fo:conditional-page-master-reference> можно применять следующие свойства:

- master-name;
- page-position;
- odd-or-even;
- blank-or-not-blank.

Теперь при создании последовательности страниц с помощью элемента <fo: page-sequence> я указываю, что процессор XSL-FO должен использовать шаблон последовательности, «sequence», который я только что создал, установив атрибут master-name элемента <fo:page-sequence>:

```
<fo:page-sequence master-name'sequence'>
```

номером текущей страницы, так что установить нумерацию страниц не составит труда. Чтобы вывести номер в верхней части каждой страницы, я создаю верхний колонтитул страницы при помощи элемента <fo:static-content>.

Существует два вида потоковых объектов: <fo:static-content> и <fo:flow>. Мы уже видели, как при помощи <fo:flow> добавляются страницы в поток документа. Элемент <fo:static-content> используется для добавления в документ верхних и нижних колонтитулов С <fo:static-content> можно применять следующее свойство:

- flow-name.

Для создания колонтитула нужно только поместить элемент <fo:static-content> перед элементом <fo:flow> в последовательности страниц:

```
<fo:page-sequence master-name-"sequence">

    <fo:static-content flow-name-"xsl-region-before">
```

```
            <fo:block text-align-"end"
                font-size-"24pt"
                font•fami1y-"sans-seri f"
                1ine-height-"36pt">
                Sample Document p. <fo:page-number/»
            </f0:block»
        </fo:static-content»

        <fo:flow flow-name="xsl-region-body"»
        .
        .
        .
```

УСТАНОВКА НАЧАЛЬНОГО НОМЕРА СТРАНИЦЫ

Для установки начального номера страницы в последовательности служит свойство initial-page-number элемента <fo:page-sequence>, что позволяет вам, например, отдельно форматировать главы, начиная каждую главу с правильного номера страницы.

Наконец, нам осталось только включить в pages.fo (листинг 12.8) образец форматируемого текста, так чтобы документ содержал более одной страницы.

Листинг 12.8. pages.fo

```
<?xml version-"1.0" encoding-"utf-8"?>
<fo:root xmlns:fo-"http://www.w3.org/1999/XSL/Format">

  <fo:layout-master-set»
      <fo simple-page-master margin-right-"20mm" margin-left="20mm"
          margin-bottom-IOmm" margin-top-"10mm" page-width-"300mm"
          page-height-"400mm" master-name-"first">

          <fo region-body margin-right="Omm" margin-left="0mm"
              margin-bottom-"10mm" margin-top-"50mm"/>

          <fo:reg ion-after extent-"10mm"/>
          <fo:region-before extent-"10mm"/>

      </fo:simple-page-master»

      <fo:simple-page-master margin-right-"25mm" margin-left-"25mm"
          margin-bottom-"15mm" margin-top-"15mm" page-width-"300mm"
          page-height-"400mm" master-name-"rest"»

          <fo:region-body margin-right-"0mm" margin-left-"0mm"
              margin-bottom-"10mm" margin-top-"20mm"/>

          <fo:region-after extent-"10mm"/>
          <fo:regi on-before extent-"10mm"/»

      </fo simple-page-master»

      <fo:page-sequence-master master-name-"sequence" »
          <fo:repeatable-page-master-alternati ves>
              <fo conditional-page master-reference master-name-"first":
                  page-position-"first" /»
```

```
            <fo:conditional-page-master-reference master-name*"rest";
                page-position-"rest" />
        </f o: repeatabl e-page-master-al ter'nati ves>
    </fo:page-sequence-master>

</fo:layout-master-set>

<fo:page- sequence ma ster-name-"sequence">

    <fo:static-content flow-name-"xs1-region-before">
        <fo:block text-align="end"
            font-size-"24pt"
            font - f ami ly=" sans - s en f"
            line-height-"36pt">
            Sample Document p. <fo:page-number/?
        </fo block>
    </fo static-content>

    <fo:flow flow-name-"xsl-region-body">

        <fo:block font-size="36pt" font-family="Times"
            text-ali gn-"center"
            space-after-"24pt">
            Sample Document
        </fo:block>
        <fo:block font-size="24pt" font-family-"sans-serif" color="gray">
         Sample Text. Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
```

Листинг 12.8 (продолжение)

```
            Sample Text. Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
            Sample Text. Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
            Sample Text. Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
            Sample Text. Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
            Sample Text. Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
            Sample Text. Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
            Sample Text. Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
            Sample Text. Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
            Sample Text. Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
            Sample Text. Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
            Sample Text. Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
            Sample Text. Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
            Sample Text. Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
            Sample Text. Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
            Sample Text. Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
            Sample Text. Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.
        </fo:block»
      </fo:flow>
    </fo:page-sequence»
</f0:root»
```

Теперь все готово. Первая страница, созданная pages.fo, показана на рис. 12.8; как можно заметить, текст смещен вниз.

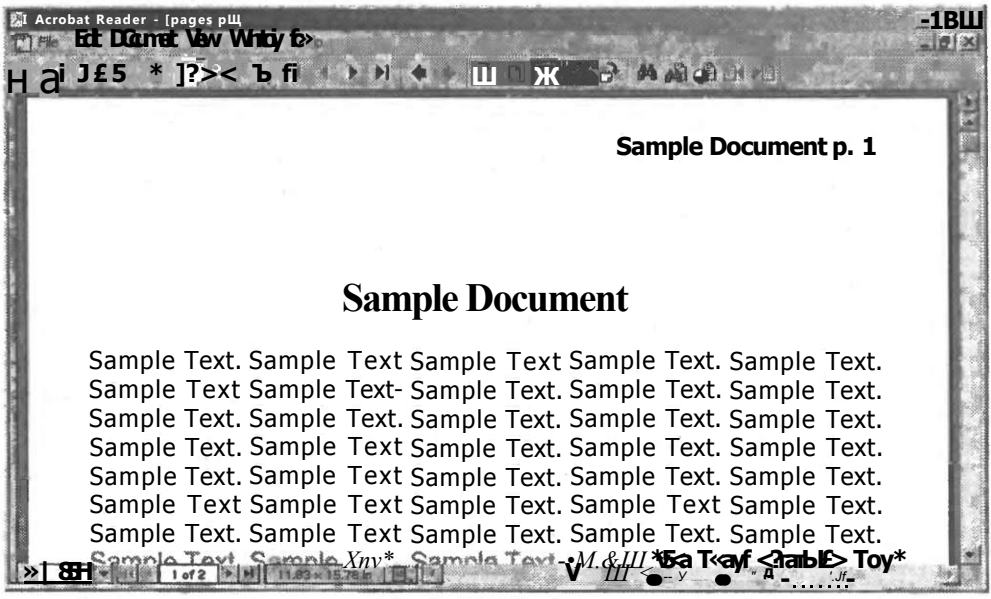


Рис. 12.8. Первая страница, отформатированная при помощи XSL-FO

Текст на второй странице начинается сверху, как видно на рис. 12.9.

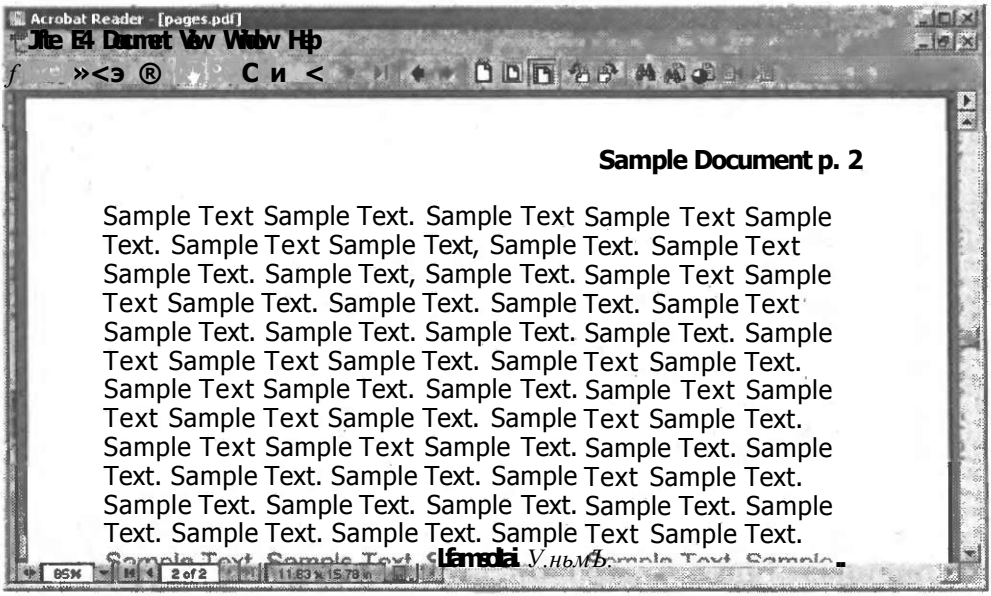


Рис 12.9. Вторая страница, отформатированная при помощи XSL-FO

Как можно заметить из листинга 12.8, я включил в pages.fo большой число строк, не содержащих ничего, кроме слов «Sample Text.», для того чтобы гарантированно получить многостраничный документ. Но для перехода на новую страницу не обязательно располагать большим количеством текста: можно тауже установить свойство break-after элемента <fo:block> в «page», и процессор XSL-FO перейдет на новую страницу после текущего блока:

```
<fo:flow flow-name^"xsl-region-body">

  <fo:block font-size-"36pt" font-family-"Times" text-allgn="center"
  space-after-"24pt">
  Sample Document
  </fo:block>
  <fo:block font-size-"24pt" font-family-"sans-serif" color-"gray"
      break-after-"page">
      Sample Text. Sample Text. Sample Text. Sample Text. Sample Text. ;
      Sample Text.
      Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.:
      Sample Text.
  </fo:block>

  <fo:block font-size-"24pt" font-family-"sans-serif" color="gray">
      Sample Text. Sample Text. Sample Text. Sample Text. Sample Text.:
      Sample Text.
      Sample Text  Sample Text. Sample Text. Sample Text. Sample Text.:
      Sample Text.
  </f0:bl0Ck>

</fo:flow>
```

Вот и все. Теперь процессор XSL-FO перейдет на следующую страницу после блока. Для создания разрывов можно использовать следующие свойства и настройки:

- break-after. Указывает, что последняя область, созданная при обработке данного форматирующего объекта, будет последней помещенной в определенный контекст. Устанавливается в auto | column | page | even-page | odd-page | inherit;
- break-before. Указывает, что первая созданная область должна быть первой, помещенной в определенный контекст. Устанавливается в auto | column | page | even-page | odd-page | inherit.

Рассмотрим еще один пример применения последовательностей страниц. Если вы посмотрите на книги, напечатанные на таких западных языках, как английский, немецкий или французский, вы заметите, что страницы с четными номерами, как правило, расположены слева, а страницы с нечетными номерами — справа. Четные и нечетные страницы можно отформатировать по-разному: например, можно задать дополнительное пространство поля у переплета, что даст дополнительное пространство левого поля для нечетных страниц и дополнительное пространство правого поля для четных.

Чтобы реализовать такое форматирование, можно применить атрибут odd-or-even элемента <fo:conditional-page-master-reference>, устанавливая его в значение «even» или «odd», выбирая соответственно четные или нечетные страницы:

```
<?xml version-"1.0" encoding="utf-8"?>
<fo:root xmlns:fo-"http://www.w3.org/1999/XSL/Format">

    <fo:layout-master-set>
        <fo:simple-page-master margin-right="50imi" margin-1eft-"20mm"
            margin-bottom»"10mm" margin-top-"10mm" page-width="300mm"
            page-height="400mm" master-name="leftpage">

            <fo:region-body margin-right-"Omm" margin-left-"0mm"
                margi n-bottom»"10mm" margi n-top-"50mm"/>

            <fo:region-after extent-"10mm"/>
            <fo:region-before extent-"10mm"/>

        </fo:si mple-page-master>

        <fo:simple-page-master margin-right-"20mm" margin-left-"50mm"
            margin-bottom-"lOircn" margin-top="10mm" page-width-"300mm"
            page-height-"400mm" master-name="rightpage">

            <fo:region-body margin-right-"0irai" margin-left-"0mm"
                margi n- bottom-"10mm" margi n-top-"20mm"/>

            <fo:region-after extent-"10mm"/>
            <fo:regi on-before extent-"10mm"/>

        </fo:simple-page-master>

        <fo:page-sequence-master master-name-"alternati ngpages">
```

```
        <fo:repeatable-page-master-alternatives>
            <fo:conditional-page-master-reference master-name="rightpage"
                odd-or-every="odd" />
            <fo:conditional-page-master-reference master-name="leftpage"
                odd-or-even="even" />
        </fo:repeatable-page-master-alternatives>
    </fo:page-sequence-master>

</fo:layout-master-set>

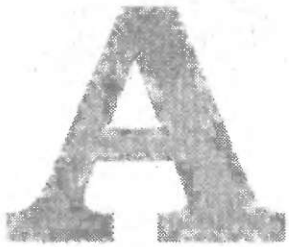
<fo:page-sequence master-name="alternatingpages">
```

Теперь у страниц, расположенных справа, левое поле у переплета книги будет расширено, а у страниц, расположенных слева, также будет расширено правое поле.

Как видите, форматирующие объекты — довольно объемная тема, и, как я уже говорил, для большой части материала в этой книге не хватает места. Дополнительные подробности можно узнать на web-узле W3C, www.w3.org/TR/xsl/. Существует не так уж много пакетов программ для работы с форматирующими объектами, хотя в будущем ситуация должна измениться.

На этом мы заканчиваем рассмотрение XSL-FO — а вместе с ним и книгу. Вы познакомились со всеми видами XSLT-преобразований: из XML в XML, в HTML, в XHTML, в RTF, в простой текст, в JavaScript, в реляционные базы данных, и теперь — в XSL-FO. Вы увидели все возможные элементы, атрибуты и функции XSLT и рассмотрели много работающих примеров. Теперь осталось только использовать всю эту мощь в работе. Удачи вам в применении XSLT!

DTD XSLT



В данном приложении приводится фрагмент определений DTD для таблиц стилей XSLT 1.0, раздел C рекомендации W3C XSLT 1.0 по адресу www.w3.org/TR/xslt. Здесь для справки перечислены элементы XSLT 1.0 и их атрибуты.

Фрагмент DTD для таблиц стилей XSLT

ПРИМЕЧАНИЕ

Этот фрагмент DTD ненормативен, поскольку объявления DTD XML 1.0 не поддерживают пространства имен XML и, таким образом, не могут правильно описывать допустимую структуру таблицы стилей XSLT.

При помощи следующей сущности можно конструировать объявления для таблиц стилей XSLT, создающих экземпляры конкретного результирующего DTD. Перед тем как ссылаться на сущность, DTD таблицы стилей должно определить сущность параметра result-elements, перечисляющую допустимые типы результирующих элементов. Например:

```
<!ENTITY % result-elements "
   | fo:inline-sequence
   | fo.block
">
                t
```

Такие результирующие элементы должны быть объявлены с атрибутами xsl:use-attribute-sets и xsl:extension-element-prefixes. Следующая сущность объявляет для этих целей параметр result-element-atts. XSLT допускает для результирующих элементов то же содержимое, что и для элементов XSLT, объявленных в следующей сущности с моделью содержимого %template;. DTD может использовать более строгую модель содержимого, чем %template;, для того чтобы отразить ограничения результирующего DTD.

DTD может определить сущность параметра non-xsl-top-level, чтобы сделать допустимыми дополнительные элементы высокого уровня из пространств имен, отличных от пространства имен XSLT.

ПРИМЕЧАНИЕ И ЛИЦЕНЗИОННОЕ СОГЛАШЕНИЕ ДЛЯ ДОКУМЕНТОВ W3C

Copyright © 1994-2000 World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). Все права защищены, http ://www.w3 .org/Consortium/Legal/

Открытые документы на web-узле W3C защищены правами в соответствии с данной лицензией. На программное обеспечение или объявления типов документа (DTD, Document Type Definitions), использующие спецификации W3C, распространяется Примечание о программном обеспечении (Software Notice). Используя и/или копируя этот документ, или документ W3C, ссылающийся на данное утверждение, вы (сторона, обладающая лицензией) соглашаетесь, что вы прочитали, поняли и будете выполнять следующие положения и условия:

разрешается использовать, копировать и распространять содержимое этого документа, или документа W3C, ссылающегося на данное утверждение, любыми средствами для любых целей бесплатно и без каких-либо отчислений, при условии, что во ВСЕ используемые копии документа или его частей вы включите следующее:
URL или ссылку на исходный документ W3C;

существующее примечание об авторских правах первоначального автора, или, если оно отсутствует, примечание вида "Copyright © [$date-of-document] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique eten Automatique, Keio University). All Rights Reserved, http://www.w3.org/ Consortium/Legal/" (Желательна гиперссылка, но разрешается и текстовое представление.);
если существует, СТАТУС (STATUS) документа W3C;

если позволяет место, должен быть включен полный текст данного ПРИММЕЧАНИЯ (NOTICE). Мы требуем, чтобы ссылка на авторство присутствовала в любом программном обеспечении, документах или других продуктах, созданных вами на основе реализации содержимого этого документа или любой его части;

согласно данной лицензии, вы не получаете права модифицировать документы W3C или создавать производные от них документы. Тем не менее, при выполнении дополнительных требований (документированных в Copyright FAQ), W3C в некоторых случаях предоставляет право создавать модификации или производные документы.

ЭТОТ ДОКУМЕНТ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», И ВЛАДЕЛЬЦЫ АВТОРСКИХ ПРАВ НЕ ПРЕДОСТАВЛЯЮТ НИКАКИХ УТВЕРЖДЕНИЙ ИЛИ ГАРАНТИЙ —ЯВНЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ — В ТОМ ЧИСЛЕ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ — ПРИГОДНОСТИ ДЛЯ КОММЕРЧЕСКОГО ИСПОЛЬЗОВАНИЯ, ПРИГОДНОСТИ ДЛЯ КОНКРЕТНЫХ ЦЕЛЕЙ, ОТСУТСТВИЯ НАРУШЕНИЯ АВТОРСКИХ ПРАВ ИЛИ ПРАВ СОБСТВЕННОСТИ, А ТАКЖЕ ЧТО ИСПОЛЬЗОВАНИЕ ДАННОГО СОДЕРЖИМОГО НЕ НАРУШИТ КАКИХ-ЛИБО ПАТЕНТОВ, АВТОРСКИХ ПРАВ, ТОРГОВЫХ МАРОК ТРЕТЬИХ ФИРМ ИЛИ ДРУГИХ ПРАВ.

ВЛАДЕЛЬЦЫ АВТОРСКИХ ПРАВ НЕ ПОДЛЕЖАТ ОТВЕТСТВЕННОСТИ ЗА ЛЮБОЙ ПРЯМОЙ, КОСВЕННЫЙ, СПЕЦИАЛЬНЫЙ ИЛИ ПОСЛЕДУЮЩИЙ УЩЕРБ, ВОЗНИКШИЙ ОТ ЛЮБОГО ИСПОЛЬЗОВАНИЯ ДОКУМЕНТА ИЛИ ВЫПОЛНЕНИЯ ИЛИ РЕАЛИЗАЦИИ ЕГО СОДЕРЖИМОГО.

Имя и торговые марки владельцев авторских прав НЕ могут быть использованы в рекламе или публичных ссылках на этот документ или его содержимое без особого письменного разрешения владельцев. Собственность на авторские права на этот документ при любых условиях останется у владельцев авторских прав.

Использование префикса xsl: в этом DTD не означает, что таблицы стилей XSLT обязаны его применять. Любые элементы, объявленные в этом DTD, могут иметь атрибуты, имена которых начинаются с xmlns: или равны xmlns, в дополнение к атрибутам, объявленным в этом DTD.

```
<!ENTITY % char-instructions "
   | xsl:apply-templates
   | xsl:call-template
   | xsl:apply-imports
   | xsl:for-each
   | xsl:value-of
   | xsl:copy-of
   | xsl:number
   | xsl:choose
   | xsl:if
   | xsl:text
   | xsl:copy
   | xsl:variable
   | xsl:message
   | xsl:fallback
">

<!ENTITY % instructions "
  Xchar-iristructions:
   j xsl:processing-instruction
   j xsl:comment
   j xsl:element
   j xsl-.attribute
">

<!ENTITY % char-tempi ate "
 (fPCOATA
  Ichar-instructions:)*
">

<!ENTITY 3! template "
 (#PCDATA
  ^instructions:
  ^result-elements:)*
">

<!-- Используется для типа значения атрибута, явля'още ося ссылкой URI.-->
<! ENTITY % URI "GDAfA'!>

<!-- Используется для типа значения атрибута, являющегося образцом.-->
<!ENTITY % pattern "CDATA">

<!-- Используется для типа значёния атрибута, яьляющегося
     шаблоном значений атрибута.г->.
<!ENTITY % avt "CDATA">

<!-- Используется для типа значения атрибута, являющегося QName: префикс
     раскрывается процессором XSLT. -->
<!ENTITY % qname "NMTOKEN">

<!-- Как qname. но список QName. разделенных символам разделителями  -->
<!ENTITY % qnames "NMTOKENS">
```

```
<!-- Используется для типа значения атрибута, являющегося выражением.--»
<!ENTITY * expr "CDATA">

<!-- Используется для типа значения атрибута, состоящего
     из одного символа.--»
<!ENTITY % char "CDATA"»

<!-- Используется для типа значения атрибута, являющегося приоритетом. -->
<!ENTITY % priority "NMTOKEN"»

<!ENTITY * space-att "xml:space (default|preserve) #IMPLIED">

<!-- Это может быть перекрыто, для того чтобы модифицировать множество элементов
допустимых на верхнем уровне. -->

<!ENTITY % non-xsl-top-level "">

<!ENTITY % top-level "
 (xsl:import*,
  (xsl:include
  | xsl:strip-space
  j xsl:preserve-space
  j xsl-.output
  j xsl:key
  I xsl:decimal-format
  j xsl attribute-set
  I xsl:variable
  I xsl:param
  j xsl:template
  j xsl:namespace-alias
  Knon-xsl-top-level:)*)
">

<!ENTITY % top-level-atts '
  extension-element-prefixes CDATA #IMPLIED
  exclude-result-prefixes CDATA #IMPLIED
  id ID #IMPLIED
  version NMTOKEN #REQUIRED
  xmlns:xsl CDATA #FIXED "http://wvM.w3.org/1999/XSL/Transform"
  Xspace att:
•>

<!-- Эта сущность определена для использования в объявлении ATTLIST
для результирующих элементов. -->

<!ENTITY % result-element-atts [1]
  xsl:extension-element-prefixes CDATA #IMPLIED
  xsl:exclude-result-prefixes CDATA #1MPLIED
  xsl:use-attribute-sets Sqnames: #IMPLIED
  xsl:version NMTOKEN #IMPLIED
•>

<!ELEMENT xsl:stylesheet Stop-level:»
<!ATTLIST xsl:stylesheet *top-level-atts:>

<fELEMENT xsltransform Stop-level:>
<!ATTLI5T xsltransform Stop-level-atts:»
```

```
<!ELEMENT xsl:import EMPTY>
<!ATTLIST xsl:import href XURI: #REQUIRED>

<!ELEMENT xsl:include EMPTY>
<!ATTLIST xsl:include href Ш 1 ; #REQUIRED>

<!ELEMENT xsl:strip-space EMPTY>
<!ATTLIST xsl:strip-space elements CDATA #REQUIRED>

<!ELEMENT xsl:preserve-space EMPTY>
<!ATTLIST xsl:preserve-space elements CDATA #REQUIRED>

<!ELEMENT xsl:output EMPTY>
<!ATTLIST xsl:output
  method Sqname: #IMPLIED
  version NMTOKEN #IMPLIED
  encoding CDATA #IMPLIED
  omit-xml-declaration (yes|no) #IMPLIED
  standalone (yes|no) #IMPLIED
  doctype-public CDATA #IMPLIED
  doctype-system CDATA #IMPLIED
  cdata-section-elements Xqnames: #IMPLIED
  indent (yes|no) #IMPLIED
  media-type CDATA #IMPLIED
>

<!ELEMENT xsl:key EMPTY>
<!ATTLIST xsl:key
  name Sqname: #REQUIRED
  match ^pattern: #REQUIRED
  use Sexpn fREQUIRED
>

<!ELEMENT xsl:decimal-format EMPTY>
<!ATTLIST xsl:decimal-format
  name Xqname; #IMPLIED
  decimal-separator !Schar: "."                                    «Ч.
  grouping-separator fcchar: "."
  infinity CDATA "Infinity"
  minus-sign Bchar: "-"
  NaN CDATA "NaN"
  percent Xchar: "%"
  per-mille Kchar; "&#x2030:"
  zero-digit fcchar; "0"
  digit fcchar: "#"
  pattern-separator fcchar: ":"
>

<!ELEMENT xsl:namespace-alias EMPTY>
<!ATTLIST xsl:namespace-alias
  stylesheet-prefix CDATA #REQUIRED
  result-prefix CDATA #REQUIRED
>

<!ELEMENT xsl:tempi ate
 (#PCDATA
  ^instructions:
```

```
  %result-elements;
  | xsl:param)*
>

<!ATTLIST xsl:tempiate
  match «pattern: #IMPLIED
  name «qname: #IMPLIED
  priority «priority: #IMPLIED
  mode «qname: #IMPLIED
  «space-att:
>

<!ELEMENT xsl:value-of EMPTY>
<!ATTLIST xsl:value-of
  select «expr; #REQUIRED
  disable-output-escaping (yes|no) "no"
>

<!ELEMENT xsl:Copy-of EMPTY>
<!ATTLIST xsl:Copy-of select «expr: #REQUIRED>

<!ELEMENT xsl:number EMPTY>
<!AmiST xsl:number
   level (single|mult1ple|any) "single"
   count «pattern: #IHPL1ED
   from ^pattern: #IMPLIED
   value Ke>;pr: #IMPLIED
   format «avt; "Г
   lang «avt: #IMPLIED
   letter-value «avt: #IMPLIED
   grouping-separator «avt: #IMPLIED
   grouping-size *avt: IIMPLIED
>

<!D.EMEfJT xsl: apply-templ ates (xsl-.sort | xsl :with-param)*>
<!ATTLJST xsl:apply-templates
  select lexpr: "nodeO"
  mode «qname: #IMPLIED
>

<!ELEMENT xsl:apply-imports EMPTY>

<!-- xsl:sort не может встречаться после любых других элементов или
любого символа, не являющегося разделителем -->

<!ELEMENT xsl:for-each
 (IPCDATA
  «instructions:
  «result-elements:
  | xsl:sort)*
>

<!ATTLIST xsl:for-each
  select «expr: #REQUIRED
  «space-att;
>

<!ELEMENT xsl:s0rt EMPTY>
```

```
<!ATTLIST xsl:SOrt
  select Sexpr; "."
  lang Savt; #IMPLIED
  data-type fcavt: "text"
  order Savt; "ascending"
  case-order Xavt: #IMPLIED
>

<!ELEMENT xsl:if Itemplate:»
<!ATTLIST xsl:if
  test Sexpr: #REQUIRED
  Sspace-att:
>

<!ELEMENT xsl:choose (xsl:when+. xsl:otherwise?)»
<!ATTLIST xsl:choose Sspace-att:>

<!ELEMENT xsl:when ttemplate:>
<!ATTLIST xsl:when
  test Sexpr; #REQUIRED
  «space-att:
>

<!ELEMENT xslotherwise «template:»
<!ATTLIST xslotherwise «space-att:>

<!ELEMENT xsl:attribute-set (xsl:attribute)*»
<!ATTLIST xsl:attribute-set
  name «qname: #REQUIRED
  use-attribute-sets «qnames: #IMPLIED
>

<!ELEMENT xsl:call-template (xsl:with-param)*>
<!ATTLIST xsl:call-template
  name Sqname: #REQUIRED
>

<!ELEMENT xsl:with-param «template:»
<!ATTLIST xsl:with-param
  name «qname: #REQUIRED
  select «expr; #IMPLIED
>

<!ELEMENT xslvariable «template:»
<!ATTLIST xsl:variable
  name Sqname: #REQUIRED
  select «expr; #IMPLIED
>

<!ELEMENT xsl:param «template:»
<!ATTLIST xsl:param
  name «qname: #REQUIRED
  select «expr: #IMPLIED
>

<!ELEMENT xsl:text (#PCDATA)>
<!ATTLIST xsl:text
  disable-output-escaping (yes|no) "no"
>
```

```
<!£LEf1EUT xsl:processing-instruction «char-template:»
<!ATTLIST xslprocessing-instruction
  mme «avt: «REQUIRED
  «space-att:
>

<!ELEMENT xsl:element «template:»
<!ATTLIST xsl:element
  name «avt: #REQUIRED
  namespace «avt: #IMPLIED
  use-attr1bute-sets «qnames: #IMPLIED
  *space-att:
>

<!ELEMENT xslattribute «char-tempiate:»
<!ATTLIST xslattribute
  name «avt: #REQUIRED
  namespace «avt: #IMPLIED
  «space-att:
>

<!ELEMENT xsl:comment «char-tempiate:»
<!ATTLIST xsl-.comment «space-att:»

<!ELEMENT xsl:copy «template:»
<!ATTLIST xsl:copy
  «space-att:
  use-attribute-sets «qnames: #IMPLIED
>

<!ELEMENT xsl:message «template:»
<!ATTLIST xsl:message
  «space-att:
  terminate (yes|no) "no"
>

<!ELEMENT xsl:fallback «template:»
<!ATTLIST xsl:fall back «space-att:»
```

Форматирующие свойства XSL-FO



В этом приложений перечислены все форматирующие свойства XSL-FO в спецификации XSL 1.0. Подробнее о свойствах XSL-FO можно узнать по адресу www.w3.org/TR/xsl/slice7.html. Многие свойства XSL-FO унаследованы из спецификации для каскадных таблиц стилей версии 2, которую вы можете найти по адресу www.w3.org/TR/REC-CSS2.

В следующем списке приведен синтаксис, используемый в этом приложении, соответствующий спецификации XSL-FO:

- о обозначает единицы значений (см. таблицу B.1), или свойства (например, <color>), которые уже были определены;
- | означает вариант выбора, использовать можно только один из них;
- || означает вариант выбора, можно использовать один или несколько вариантов в любом порядке;
- [] означает групповые инструкции, вычисляемые во многом так же, как и математические выражения;
- * означает, что предыдущий терм может встречаться ноль или более раз;
- + означает, что предыдущий терм может встречаться один или более раз;
- ? означает, что предыдущий терм необязателен;
- {} окружают пары чисел, задающие минимальное и максимальное число раз встречи терма (как в {1, 4});

Во многих свойствах XSL-FO задан определенный формат для присваиваемых им значений, как, например <color> или <angle>. Эти форматы и их описание перечислены в следующем списке.

- <absolute-size>. Абсолютные размеры шрифта; могут быть xx-small, x-small, small (маленький), medium (средний), large (крупный), x-large, xx-large;

- <angle>. Углы; мо1ут измеряться в deg (градусах), grad (градах) или rad (радианах);
- <border-style>. Граница прямоугольника, может принимать значения поле (нет), dotted (пунктиром), dashed (штрихами), solid (непрерывная), double (двойная), groove (канавкой), ridge (с толстой кромкой), inset (вложенная) или outset (вынесенная);
- <border-width>. Устанавливает ширину границы; может принимать значения thin (тонкая), medium (средняя), thick (толстая) или явного размера;
- <color>. Цвет; может быть задан предопределенным значением цвета или тройным цветовым значением RGB;
- <country-language>. Устанавливается в значение <language>;
- <family-name>. Название семейства шрифтов, такого как Arial, Times New Roman или Courier;
- <frequency>. Значения частоты; единицами измерения могут быть Hz (Гц) или KHz (кГц);
- <generic-family>. Обобщенные названия шрифтов, применяемых в качестве последней попытки, если браузер не может найти определенный шрифт. Например, serif (браузер должен выбрать шрифт serif), sans-serif (браузер должен выбрать семейство sans-serif) и monospace (браузер должен выбрать моноширинный шрифт);
- <generic-voice>. Звук голоса; может быть male (мужской), female (женский) или child (детский);
- <integer>. Стандартные целые значения;
- <keep>. Контекстно-зависимое свойство, принимает значение Auto для того, чтобы не удерживать вместе ограничения, или Always для жесткого удержания ограничений вместе. Дополнительная информация приведена в спецификации XSL;
- <language>. Определитель языка, соответствующий спецификации RFC1766 (которую можно найти по адресу www.w3.org/TR/xsl/sliceD.html#RFC1766);
- <length>. Длина, может начинаться с + или -, за которыми следует число. Число может содержать десятичную точку, за которой следует идентификатор единиц, принимающий значения em (размер шрифта соответствующего шрифта), ex (x-height, высота шрифта), px (пикселов, pixel, определяется устройством просмотра), pt (пунктов, point, 1/72 дюйма), in (дюймов, inch), cm (сантиметров), mm (миллиметров) или pc (пик, pica, 1/6 дюйма);
- <length-bp-ip-direction>. Задает расстояние, разделяющее смежные границы ячейки в направлении размещения строк. Дополнительная информация приведена на ww7.w3.org/TR/xsl/ slice7.html; л
- <length-conditional>. Составное значение, задающее ширину и возможный тип передней границы Дополнительная информация приведена на www.w3.org/ TK/ xsl/slice7. htm l#pr- section; -

- <length-range>. Определяет диапазон длины, как расписано в спецификации XSL;
- <margin-width>. Может принимать значения <length>, <percentage> или auto. Работа значения auto зависит от контекста, подробности приведены на www.w3.org/TR/REC-CSS2/visudet.html#Computing_widths.and_margins;
- <number>. Число, может включать знак и десятичную точку;
- <padding-width>. Устанавливается в значение <length>;
- «percentage». Число, может содержать знак, за которым следует знак процента (%);
- <relative-size>. Размер шрифта относительно родительского элемента, может быть больше или меньше;
- <shape>. Пока может задавать только прямоугольник (shape — фигура), как, например: rect(<top> <right> <bottom> <left>);
- <space>. Задает минимальное, оптимальное и максимальное значения для пробела. Подробности см. на www.w3.org/TR/xsl/slice4.html#spacecond;
- <specific-voice>. Задает определенный голос. Подробности см. на www.w3.org/TR/REC-CSS2/aural.html#propdef-voice-family;
- <time>. Единицы времени, указанные как число, за которым сразу же следует ms (миллисекунды) или s (секунды);
- <uri-specification>. Uniform Resource Indicator (URI), web-адрес элемента страницы, такого, как рисунок.

Отметьте также, что свойства XSL-FO, как и свойства CSS2, могут принимать значение «inherit», что означает, что значение свойства должно быть унаследовано от его родительского элемента.

В XSL-FO есть следующие свойства:

- absolute-position. Определяет, является ли позиция элемента абсолютной. Устанавливается в auto | absolute | fixed | inherit;
- active-state. Определяет, какие из наборов <fo:multi-property-sets> будут использоваться для форматирования дочерних потоковых (flow) объектов. Устанавливается в link | visited | active j hover | focus;
- alignment-adjust. При помощи этого свойства можно установить позицию базовой линии (baseline), заданной свойством alignment-baseline.устанавливается в auto | baseline J before-edge | text-before-edge | middle | central j after-edge | text-after-edge | ideographic | alphabetic j hanging | mathematical | <percentage> | <length> J inherit;
- alignment-baseline. Определяет выравнивание объекта по отношению к его родителю. Устанавливается в auto | baseline | before-edge } text-befOre-edge | middle | central | after-edge | text-after-edge | ideographic ! alphabetic | hanging j mathematical | inherit; a.
- auto-restore. Определяет, будет ли восстанавливаться исходный <fo:multi-case> при скрытии <fo:multi-switch>. Устанавливается; в true J false;

- azimuth. Устанавливает азимут угла. Устанавливается в <angle> | [[left-side | far-left | left | center-left | center | center-right | right | far-right | right-side] | | behind] | leftwards | rightwards | inherit;
- background. Свойство для быстрой установки сразу всех отдельных свойств заднего фона (background-color, background-image, background-repeat, background-attachment и background-position). Устанавливается в <background-color> || <background-image> || <background-repeat> || <background-attachment> || <background-position>]] | inherit;
- background-attachment. Определяет, будет ли прокручиваться задний фон. Устанавливается в scroll | fixed | inherit;
- background-color. Задает цвет заднего фона элемента. Устанавливается в <color> | transparent | inherit;
- background-image. Задает рисунок заднего фона элемента. Устанавливается в <uri-specification> | none | inherit;
- background-position. Определяет начальное положение рисунка заднего фона. Устанавливается в <percentage> | <length>]{1,2}} | [[top | center | bottom] 11 [left | center | right]]] | inherit;
- background-position-horizontal. Определяет горизонтальное положение заднего фона, если он задан. Устанавливается в <percentage> | <length> | left | center | right | inherit;
- background-position-vertical. Определяет вертикальное положение заднего фона, если он задан. Устанавливается в <percentage> | <length> | top | center | bottom | inherit;
- background-repeat. Определяет, может ли рисунок заднего фона располагаться мозаикой, и если да, то каким образом. Устанавливается в repeat | repeat-x | repeat-y | no-repeat | inherit;
- baseline-shift Переустанавливает положение базовой линии относительно базовой линии родителя. Устанавливается в baseline | sub | super | <percentage> | <length> | inherit;
- blank-or-not-blank. Формирует часть правил выбора, чтобы определить, можно ли выбрать в данный момент указанный шаблон страницы в последовательности страниц. Устанавливается в blank | not-blank j any | inherit;
- block-progression-dimension. То же, что и свойство CSS2 с тем же именем. Устанавливается в auto | <length> | <percentage> j <4length-range> | inherit;
- border. Свойство для быстрой установки одинаковых ширины, цвета и стиля для всех четырех границ (верхней, нижней, левой и правой) прямоугольника. Устанавливается в [<border-width> | | <border-style> | j <color>] [inherit;

{ • border-after-color. Задает цвет задней границы области. Устанавливается в <color> | inherit;

- • border-after-precedence. Определяет порядок следования спецификации границы для задней границы. Устанавливается в force | <integer> | inherit;

- border-after-style. Задает стиль задней границы. Устанавливается в <border-style> | inherit;
- border-after-width. Задает ширину задней границы. Устанавливается в <border-width» | <length-conditional> | inherit;
- border-before-color. Задает цвет передней границы. Устанавливается в <color> | inherit;
- -border-before-precedence. Определяет порядок следования спецификации границы для передней границы. Устанавливается в force | <integer> J inherit;
- border-before-style. Задает стиль передней границы. Устанавливается в <border-style> | inherit;
- border-before-width. Задает ширину передней границы. Устанавливается в <border-width> J <length-conditional> | inherit;
- border-bottom. Свойство для быстрой установки ширины, стиля и цвета нижней границы области блока или встроенной области. Устанавливается в [<border-top-width> || <border-style> || <color>] J inherit;
- border-bottom-color. Задает цвет нижней границы. Устанавливается в <color> | inherit;
- border-bottom-style. Задает стиль нижней границы. Устанавливается в <border-style> | inherit;
- border-bottom-width. Задает ширину нижней границы. Устанавливается в <border-width> | inherit;
- border-collapse. Задает модель границы таблицы. Устанавливается в collapse | separate | inherit;
- border-color. Задает цвет сразу всех четырех границ. Устанавливается в <color> {1,4} | transparent | inherit;
- border-end-color. Задает цвет конечной границы. Устанавливается в <color> | inherit;
- border-end-precedence. Определяет порядок следования спецификации границы для конечной границы. Устанавливается в force j <integer> | inherit;
- border-end-style. Задает стиль конечной границы. Устанавливается в <border-style> | inherit;
- border-end-width. Задает ширину конечной границы. Устанавливается в <border-width> j <length-conditionab> | inherit
- border-left. Свойство для быстрой установки одновременно ширины, стиля и цвета левой границы. Устанавливается в [<border-top-width> || <border-style> || <color>] | inherit; „
- border-left-color. Задает цвет левой границы. Устанавливается в <color> | inherit;
- border-left-style. Задает стиль левой границы. Устанавливается в <border-style> | inherit;

- border-left-width. Задает ширину левой границы. Устанавливается в «border-width> | inherit;
- border-right. Свойство для быстрой установки одновременно ширины, стиля и цвета правой границы. Устанавливается в [<bordef-top-width> || -«border-style» || <color>] | inherit;
- border-right-color. Задает цвет правой границы. Устанавливается в <color> | inherit;
- border-right-style. Задает стиль правой границы. Устанавливается в «border-style» | inherit;
- border-right-width. Задает ширину правой границы. Устанавливается в «border-width» | inherit;
- border-separation. Определяет разделение границей между смежными ячейками. Устанавливается в «length-bp-ip-direction> | inherit;
- border-spacing. Задает расстояние между смежными границами ячейки. Устанавливается в «length» «length»? | inherit;
- border-start-color. Задает цвет начальной границы. Устанавливается в «color»;
- border-start-precedence. Определяет порядок следования спецификации границы для начальной границы. Устанавливается в force | «integer» | inherit;
- border-start-style. Задает стиль начальной границы. Устанавливается в «border-Style» | inherit;
- border-start-width. Задает ширину начальной границы. Устанавливается в «border-width» | «length-conditional» | inherit;
- border-style. Задает стиль четырех границ. Устанавливается в «border-style» {1,4} | inherit;
- border-top. Свойство для быстрой установки одновременно ширины, стиля и цвета верхней границы области блока или встроенной области. Устанавливается в [«border-top-width» || «border-style<@062» || «color»] | inherit;
- border-top-color. Задает цвет верхней границы. Устанавливается в «color» | inherit;
- border-top-style. Задает стиль линии границы прямоугольника (непрерывная, двойная, штриховая и т. д.);
- border-top-width. Задает ширину верхней границы. Устанавливается в «border-width» | inherit;
- border-width. Свойство для быстрой установки одновременно свойств border-top-width, border-right-width, border-bottom-width и border-left-width. Устанавливается в «border-width»{1,4} | inherit;
- bottom. Определяет, насколько нижний край содержимого прямоугольника (box) смещен вверх от нижней границы содержащего прямоугольник блока. Устанавливается в «length» | «percentage» | auto | inherit;

- break-after. Указывает, что последняя сгенерированная область будет последней помещенной в заданный контекст. Устанавливается в auto | column j page | even-page | odd-page | inherit;
- break-before. Указывает, что первая сгенерированная область будет первой помещенной в заданный контекст. Устанавливается в auto | column | page | even-page | odd-page | inherit;
- caption-side. Определяет положение подписи. Устанавливается в before | after | start | end | top | bottom j left | right | inherit;
- case-name. Задает имя для элемента <fo:multi-case>. Устанавливается в <string>;
- case-title. Задает описательный заголовок для элемента <fo:multi-case>. Устанавливается в <string>;
- character. Задает вставляемый символ Unicode. Устанавливается В «character»;
- clear. Указывает, какие стороны прямоугольников элементов могут быть не выровнены по предыдущему плавающему (floating) прямоугольнику. Устанавливается в start | end J left | right | both | none | inherit; ,
- clip. Усекает элементы, у которых свойство «overflow» имеет значение, отличное от «У18&1е».Устанавливается в <shape> j auto j inherit;
- color. Задает цвет переднего фона текста элемента. Устанавливается в <color> | inherit;
- color-profile-name. Задает имя для профиля цветов. Устанавливается в «пате» | inherit
- column-count Задает число столбцов в области. Устанавливается в <number> | inherit;
- column-gap. Задает ширину промежутка между смежными колонками в многоколоночной области. Устанавливается в <length> | «percentage» J inherit;
- column-number. Устанавливает номер столбца для ячеек таблицы. Устанавливается в <number>;
- column-width. Задает ширину колонки. Устанавливается в auto J scale-to-fit | <length> J «percentage» (7.25.9) — ширина колонки и отсутствие наследования;
- content-height. Задает высоту такого объекта, как внешняя графика. Устанавливается в auto | scale-to-fit | «length» | «percentage» \ inherit;
- content-type. Это свойство определяет тип содержимого и может быть использовано клиентом для выбора способа отображения данных. Устанавливается в «string» | auto;
- content-width. Задает ширину объекта — такого, как внешняя графика. Устанавливается в auto j scale-to-fit | «length» J «percentage» | inherit;
- country. Определяет страну для языковых или связанных с локализацией средств форматирования. Устанавливается в none J «country»] inherit;

- cue. То же, что и свойство CSS2 с тем же именем. Устанавливается в «cue-before» | | <cue-after» | inherit;
- cue-after. То же, что и свойство CSS2 с тем же именем. Устанавливается в <uri-specification» | none | inherit;
- cue-before. То же, что и свойство CSS2 с тем же именем. Устанавливается в <uri-specification» | none | inherit;
- destination-placement-offset. Устанавливает расстояние от верха страницы до самой внутренней области строки, содержащей первую область назначения. Устанавливается в «length»;
- direction. Определяет основное направление написания текста для двунаправленного алгоритма Unicode. Устанавливается в ltr | rtl | inherit;
- display-align. Определяет выравнивание дочерних областей заданной области. Устанавливается в auto | before | center | after | inherit;
- dominant-baseline. Применяется для задания масштабированной по базовой линий таблицы. Устанавливается в auto | use-script J no-change | reset-size | ideographic | alphabetic | hanging | mathematical | inherit;
- elevation. То же, что и свойство CSS2 с тем же именем. Устанавливается в «angle» | below | level | above | higher] lower | inherit;
- empty-cells. Управляет отображением границ вокруг ячеек, у которых нет видимого содержимого. Устанавливается в show | hide | inherit;
- endIndent. Устанавливает расстояние от конечного края блока до конечного края содержащей его заданной области. Устанавливается в «length» | inherit;
- ends-row. Определяет, завершает ли данная ячейка строку таблицы. Устанавливается в true J false;
- extent. Задает ширину начальной (region-start) или конечной (region-end) областей; или высоту передней (region-before) или задней (region-after) областей. Устанавливается в <length» | «percentage» J inherit;
- external-destination. Задает ресурс назначения для <fo:basic|ink>. Устанавливается в «uri-specification»;
- float. Определяет, будет ли прямоугольник смещен влево, вправо, или вообще не будет смещен. Устанавливается в before | start | end | left | right | none | inherit;
- flow-name. Задает имя потока (flow). Устанавливается в <name>;
- font. Свойство для быстрого задания одновременно свойств font-style, font-variant, font-weight, font-size, line-height и font-family. Устанавливается в [[«font-style» || «font-variant» || «font-weight»]? «font-size» [/ «line-height»]? «font-family»] | caption | icon | menu | message-box | small-caption f status-bar | inherit;
- font-family. Задает список имен семейств шрифтов и/или обобщенных

i. имен семейств в порядке предпочтения. Устанавливается в [[«family-name» | «generic-family»],]* «family-name» | «generiq-famUy»] | inherit;,

- font-size. Задает размер шрифта. Устанавливается в <absolute-size> j «relative-size» | <length> j «percentage» j inherit;
- font-size-adjust. Изменяет размер шрифта в заданной пропорции. Устанавливается в «number» | none | inherit;
- font-selection-strategy. Определяет стратегию выбора шрифта. Устанавливается в auto J character-by-character | inherit (7.7.3);
- font-stretch. Выбирает обычный, сжатый или разреженный вид из семейства шрифтов. Устанавливается в normal | wider | narrower | ultra-condensed | extra-condensed J condensed j semi-condensed | semi-expanded J expanded | extra-expanded | ultra-expanded | inherit;
- font-style. Задает стиль шрифта. Устанавливается в normal | italic | oblique | backslant | inherit;
- font-variant. Выбирает шрифты, имеющие в своем наборе как строчные, так и прописные буквы (bicameral fonts). Устанавливается в normal | small-caps | inherit;
- font-weight. Устанавливает насыщенность шрифта. Устанавливается в normal | bold j bolder | lighter 1100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit;
- force-page-count. Накладывает ограничение на число страниц в последовательности. Устанавливается в auto | even | odd | end-on-even j end-on-odd | no-force | inherit;
- format. Формат XSLT. Устанавливается в «string»;
- glyph-orientation-horizontal Задает ориентацию глифов, применяется только к тексту, написанному в горизонтальном режиме. Устанавливается в «angle» | inherit;
- glyph-orientation-vertical. Задает ориентацию глифов, применяется только к тексту, написанному в вертикальном режиме. Устанавливается в «angle» | inherit (7.26.3);
- grouping-separator. Разделитель групп разрядов формата XSLT. Устанавливается в «character»;
- grouping-size. Размер группы формата XSLT. Устанавливается в «number»;
- height Задает высоту содержимого прямоугольников, создающихся элементами уровня блока и замещенными элементами. Устанавливается в «length» | «percentage» | auto | inherit;
- hyphenate. Определяет, разрешаются ли переносы при разрыве строки. Устанавливается в false | true | inherit;;
- hyphenation-character. Определяет символ Unicode, используемый при переносе. Устанавливается в «character» | inherit;
- hyphenation-keep. Определяет, можно ли выполнять перенос в последней строке заданной области. Устанавливается в auto | column j page | inherit;

- hyphenation-ladder-count. Устанавливает предел числа последовательных областей строк с переносом, которые средство форматирования может генерировать в блоке. Устанавливается в no-limit | <number> | inherit;
- hyphenation-push-character-count Положительное целое, определяющее минимальное число символов в перенесенном слове после символа переноса. Устанавливается в <number> | inherit;
- hyphenation-remain-character-count. Положительное целое, определяющее минимальное число символов в перенесенном слове до символа переноса. Устанавливается в <number> | inherit;
- id. Идентификатор, уникальный среди всех объектов результирующего дерева в одном пространстве имен. Устанавливается в <number> | inherit;
- indicate-destination. Определяет, будет или не будет указано назначение (destination). Устанавливается в true | false;
- initial-page-number. Устанавливает начальный номер страницы. Устанавливается в auto | auto-odd | auto-even | <number> [inherit;
- inline-progression-dimension. То же, что и свойство CSS2 с тем же именем. Устанавливается в auto | <length> | «percentage* J <length-range> | inherit;
- internal-destination. Задает потоковый объект назначения элемента <fo:basic-link>. Устанавливается в empty string j «idref»;
- keep-together. Определяет, какие элементы нужно держать вместе. Устанавливается в <keep> | inherit;
- keep-with-next. Устанавливает условия «держать вместе со следующим» для форматирующих объектов. Устанавливается в «keep» | inherit;
- keep-with-previous. Устанавливает условия «держать вместе с предыдущим» для форматирующих объектов. Устанавливается в <keep> | inherit;
- language. Определяет язык для средства форматирования (formatter). Устанавливается в none | «language» J inherit;
- last-line-end-indent Определяет выравнивание для конечного края последней области строки в блоке. Устанавливается в «length» | «percentage» | inherit;
- leader-alignment. Определяет выравнивание указывающей линии. Устанавливается в none | reference-area | page | inherit;
- leader-length. Задает длину указывающей линии. Устанавливается в «length-range» | inherit;
- leader-pattern. Устанавливает образец для указывающей линии. Устанавливается в space | rule | dots j use-content | inherit;
- leader-pattern-width. Устанавливает ширину образца для указывающей линии. Устанавливается в use-font-metrics | «length» | inherit;
- left Определяет, насколько левый край содержимого прямоугольника (box) смещён вправо от левой границы содержащего прямоугольник блока. Устанавливается в «length» j «percentage» | auto | inherit,

- letter-spacing. Определяет разбивку пробелами между символами текста. Устанавливается в normal | «length» | <space> | inherit;
- letter-value. Устанавливает номер XSLT для атрибутов преобразования строк. Устанавливается в auto | alphabetic j traditional;
- linefeed-treatment. Определяет способ обработки символов перевода строки. Устанавливается в ignore | preserve | treat-as-space | treat-as-zero-width-space | inherit;
- line-height. Задает минимальную высоту каждого созданного встроенного прямоугольника. Устанавливается в normal | <length> | <number> | «percentage» | <space>} inherit;
- line-height-shift-adjustment. Определяет, будет ли высота строки подстраиваться для содержимого, сдвинутого относительно базовой линии. Устанавливается в consider-shifts j disregard-shifts j inherit;
- line-stacking-strategy. Выбирает стратегию размещения смежных строк. Устанавливается в line-height | font-height J max-height | inherit;
- margin. Свойство для быстрого задания одновременно свойств margin-top, margin-right, margin-bottom и margin-left. Устанавливается в <margin-width> {1,4} | inherit;
- margin-bottom. Задает нижнее поле прямоугольника. Устанавливается в <margin-width> | inherit;
- margin-left. Задает левое поле прямоугольника. Устанавливается в «margin-width» j inherit;
- margin-right Задает правое поле прямоугольника. Устанавливается в «margin-width» | inherit;
- margin-top. Задает верхнее поле прямоугольника. Устанавливается в «margin-width» | inherit;
- marker-class-name. Включает элемент «fo:marker» в группу других элементов с тем же именем, каждый из которых можно извлечь при помощи элемента <fo:retrieve-marker», имеющего то же значение для свойства «retrieve-class-name». Устанавливается в «пате»;
- master-name. Задает или выбирает шаблон (master). Устанавливается в «пате»;
- max-height. Задает максимальную высоту прямоугольника. Устанавливается в «length» | «percentage» | none j inherit;
- maximum-repeats. Задает максимальное количество страниц в подпоследовательности страниц, которые могут быть сгенерированы элементом «fo: page-sequence», использующим элементы «fO:repeatable-page-master- reference» или «fo:repeatable-page-master-alternatives». Устанавливается в «^number» | no-limit | inherit;
- max-width. Задает максимальную ширину прямоугольника. Устанавливается в «length» | «percentage» j none | inherit;

t media-usage. Определяет, каким образом выбранное средство отображения должно представить страницы, указанные таблицей стилей. Устанавливается в auto | paginate | bounded-in-one-dimension | unbounded;

- min-height. Задает минимальную высоту прямоугольника. Устанавливается в <length> | «percentage* | inherit;
- min-width. Задает минимальную ширину прямоугольника. Устанавливается в <length> | «percentage* | inherit;
- number-columns-repeated. Задает повторение спецификации столбца таблицы. Устанавливается в «number*;
- number-columns-spanned. Задает число столбцов, занимаемых ячейкой таблицы. Устанавливается в «number*;
- number-rows-spanned. Задает число строк, занимаемых ячейкой таблицы. Устанавливается в «number*;
- odd-or-even. Используется в правиле выбора, определяя, будет ли применяться указанный шаблон страницы в данном месте последовательности страниц. Устанавливается в odd | even J any | inherit;
- orphans. То же, что и свойство CSS2 с тем же именем. Устанавливается в «integer* | inherit;
- overflow. Определяет, будет ли содержимое элемента блока урезаться при выходе за границы. Устанавливается в visible | hidden | scroll | error-if-overflow | auto | inherit;
- padding. Свойство для быстрой установки сразу всех свойств padding-top, padding-bottom, padding-left и padding-right. Устанавливается в «padding-width*{l,4} | inherit;
- padding-after. Задает ширину заполнения на заднем крае. Устанавливается в «padding-width* | «length-conditional* | inherit;
- padding-before. Задает ширину заполнения на переднем крае. Устанавливается в «padding-width* | inherit;
- padding-bottom. Задает ширину заполнения на нижнем крае. Устанавливается в «padding-width* | inherit;
- padding-end. Задает ширину заполнения на конечном крае. Устанавливается в «padding-width* | «length-conditional* | inherit;
- padding-left. Задает ширину заполнения на левом крае. Устанавливается в «padding-width* | inherit;
- padding-right, Задает ширину заполнения на правом крае. Устанавливается в «padding-vyidth* [inherit;
- padding-start. ^Задет ширину заполнения на начальном крае. Устанавливается в «padding-width* | «length-conditional* | inherit;

.j« padding-top. Задает ширину заполнения на верхнем крае. Устанавливается в «padding-width* [inherit; ив:

- page-break-after. То же, что и свойство CSS2 с тем же именем. Устанавливается в auto | always | avoid | left | right | inherit;
- page-break-before. То же, что и свойство CSS2 с тем же именем. Устанавливается в auto | always | avoid | left | right J inherit;
- page-break-inside. То же, что и свойство CSS2 с тем же именем. Устанавливается в auto | always J avoid | left | right | inherit;
- page-height. Задает высоту страницы. Устанавливается в auto | indefinite | <length> | inherit;
- page-position. Применяется как часть правила выбора, определяя, можно ли использовать указанный шаблон страницы в данном месте последовательности страниц. Устанавливается в first | last | rest | any | inherit;
- page-width. Задает ширину страницы. Устанавливается в auto | indefinite | <length> | inherit;
- pause. То же, что и свойство CSS2 с тем же именем. Устанавливается в <time> | «percentage»] {1,2} j inherit;
- pause-after. То же, что и свойство CSS2 с тем же именем. Устанавливается в <time> | «percentage» | inherit;
- pause-before. То же, что и свойство CSS2 с тем же именем. Устанавливается в «time» | «percentage» | inherit;
- pitch. Устанавливает высоту тона звука. Устанавливается в «frequency» | x-low | low | medium | high | x-high | inherit;
- pitch-range. Задает диапазон высоты тона. Устанавливается в «number» | inherit;
- play-during. То же, что и свойство CSS2 с тем же именем. Устанавливается в «uri-specification» mix? repeat? | auto | none | inherit;
- position. Задает используемую схему размещения. Устанавливается в static | relative | absolute | fixed J inherit;
- precedence. Определяет, какая область (передняя, задняя, начальная или конечная) будет считаться первой. Устанавливается в true | false | inherit;
- provisional-distance-between-starts. Задает предварительное расстояние между начальным отступом метки элемента списка и начальным отступом тела элемента списка. Устанавливается в «length» | inherit;
- provisional-label-separation. Задает расстояние между меткой элемента списка и началом тела элемента списка. Устанавливается в «length» | inherit;
- reference-orientation. Указывает направление для вершины текущего прямоугольника содержимого. Устанавливается в 0 (90 1180 | 270 j -90 J -180 | -270 | inherit;
- ref-id. Ссылка на объект с заданным уникальным идентификатором. Устанавливается в «idref» | inherit;

- region-name. Указывает имя области. Устанавливается в xsl-region-body | xsl-region-start | xsl-region-end | xsl-region-before | xsl-region-after | xsl-before-float-separator | xsl-footnote-separator | <name>;
- relative-align. Задает выравнивание между двумя или более областями в направлении продвижения блока (block-progression-direction). Устанавливается в before | baseline | inherit;
- relative-position. То же, что и свойство CSS2 с тем же именем. Устанавливается в static | relative | inherit;
- rendering-intent. Определяет вид отображения цветового профиля, отличного от установленного по умолчанию. Устанавливается в auto [perceptual J relative-colorimetric | saturation | absolute-colorimetric | inherit;
- retrieve-boundary. Устанавливает границу извлечения. Устанавливается в page | page-sequence | document;
- retrieve-dass-name. Определяет, что у свойства marker-dass-name элемента <fo:marker>, чьи дети извлекаются элементом <fo:retrieve-marker>, должно быть такое же значение, что и у этого свойства. Устанавливается в <name>;
- retrieve-position. Устанавливает предпочтение, с которым дети элемента <fo: marker* будут извлекаться элементом <fo:retrieve-marker>. Устанавливается в first-starting-within-page | first-including-carryover | last-starting-within-page | last-ending-within-page;
- richness. Задает яркость тона. Устанавливается в <number> | inherit;
- right. Определяет, насколько край содержимого прямоугольника (box) смещен влево от правой границы содержащего прямоугольник блока. Устанавливается в <length> | «percentage* | auto | inherit;
- role. Задает подсказку для альтернативных средств вывода (таких, как средства чтения вслух) в качестве роли элемента XML, используемого в этом форматирующем объекте. Устанавливается в «string* | «uri-specification* | none | inherit;
- rule-style. Задает образец правила. Устанавливается в none | dotted | dashed | solid j double | groove | ridge | inherit;
- rule-thickness. Определяет общую толщину правила. Устанавливается в «length*;
- scaling. Определяет, нужно ли масштабирование для сохранения пропорции. Устанавливается в uniform, non-uniform J inherit;
- scaling-method. Задает предпочтение при форматировании растровой графики. Устанавливается в auto | integer-pixels | resample-any-method | inherit;
- score-spaces. Определяет, следует ли применять свойство text-decoration к пробелам. Устанавливается в true J false | inherit;
- script. Указывает сценарий, который будет применен средством форматирования при операциях, связанных с локализацией. Устанавливается в none | auto j «script* J inherit;

- show-destination. Определяет, будет ли отображен ресурс назначения. Устанавливается в replace | new;
- size. Задает размер и ориентацию прямоугольника страницы. Устанавливается в <length>{1,2} | auto | landscape | portrait | inherit;
- source-document. Устанавливает указатель на исходный документ XML. Устанавливается в <uri-specification> <uri-specification>]* | none j inherit;
- space-after. Задает значение для спецификатора пространства для пространства после областей, созданных этим объектом. Устанавливается в <space> | inherit;
- space-before. Задает минимальное, оптимальное и максимально^ значения для пространства перед любыми областями, созданными этим объектом, а также условия и порядок следования для этого пространства. Устанавливается в <space> | inherit;
- space-end. Задает значение спецификатора пространства для пространства после областей, созданных этим форматирующим объектом. Устанавливается в <space> | inherit;
- space-start. Задает значение для спецификатора пространства для пространства перед областями, созданными этим форматирующим объектом. Устанавливается в <space> | inherit;
- space-treatment. Определяет способ обработки пробелов и других символов-разделителей, за исключением символов перевода строки. Устанавливается в ignore | preserve | ignore-if-before-linefeed | ignore-if-after-linefeed | ignore-if-surrounding-linefeed | inherit;
- span. Определяет, нужно ли поместить объект уровня блока в текущий столбец или растянуть его на все столбцы в области. Устанавливается в none | all | inherit;
- speak. То же, что и свойство CSS2 с тем же именем. Устанавливается в normal | none | spell-out j inherit;
- speak-header. Определяет заголовок для звуковой операции. Устанавливается в once | always | inherit;
- speak-numeral. То же, что и свойство CSS2 с тем же именем. Устанавливается в digits j continuous | inherit;
- speak-punctuation. То же, что и свойство CSS2 с тем же именем. Устанавливается в code | none | inherit;
- speech-rate. Задает скорость речи. Устанавливается в <number> { x-slow | medium | fast | x-fast | faster | slower | inherit;
- sre. Задает ссылку URI, определяющую внешний ресурс. Устанавливается в <uri-specification> | inherit;
- start-indent. Определяет расстояние от начального края прямоугольника содержимого области, содержащей блок, до начального края прямоугольника содержимого области этого блока. Устанавливаете^ гв <length> | inherit;

- starting-state. Управляет начальным отображением элемента <fo:multi-case>. Устанавливается в show | hide;
- starts-row. Определяет, начинает ли эта ячейка строку. Устанавливается в true | false;
- stress. Определяет звуковое ударение. Устанавливается в <number> | inherit;
- suppress-at-line-break. Подавляет символы в разрывах строки. Устанавливается в auto j suppress | retain J inherit;
- switch-to. Определяет, на какие элементы <fo:multi-case> должен переключиться этот элемент <fo:multi-toggle>. Устанавливается в xsl-precedeing | xsl-following | xsl-any J <name> <name>]*;
- table-layout. Определяет алгоритм для размещения ячеек, строк и столбцов таблицы. Устанавливается в auto | fixed | inherit;
- table-omit-footer-at-break. Определяет, должна ли таблица, последняя область которой расположена не в конце созданной таблицей области, завершаться содержимым форматирующего объекта <fo:table-footer>. Устанавливается в true | false;
- table-omit-header-at-break. Определяет, должна ли таблица, первая область которой расположена не в начале созданной таблицей области, начинаться с содержимого форматирующего объекта <fo:table-header>. Устанавливается в true | false;
- target-presentation-context. Определяет представление приемника, такое как XML, XHTML, SVG. Устанавливается в use-target-processing-context | <uri-specification>;
- target-processing-context. Определяет корень виртуального документа, который процессор должен обработать в случае, если внешнее назначение является ресурсом обрабатываемого типа (например, XML, SVG). Устанавливается в document-root | <uri-specification>;
- target-stylesheet. Определяет таблицу стилей, которая должна быть применена для обработки ресурса. Устанавливается в use-normal-stylesheet | <uri-specification>;
- text-align. Определяет способ выравнивания встроенного содержимого блока. Устанавливается в start j center | end | justify | inside | outside | left | right | <string> | inherit;
- text-align-last. Задает выравнивание для последней области строки в блоке. Устанавливается в relative | start | center | end | justify | inside | outside | left | right | inherit;
- text-altitude. Задает «высоту» подъема над базовой линией. Устанавливается в use-font-metrics j <length> | inherit;
- text-decoration. Задает параметры оформления, добавляемого тексту элемента. Устанавливается в none J [[underline | no-underline] (| [overline | no-overline] | | [line-through | no-line-through] | | [blink | no-blink]] | inherit;

- text-depth. Задает глубину спуска ниже базовой линии. Устанавливается в use-font-metrics | <length> | inherit;
- text-indent. Определяет выравнивание первой строки текста в блоке. Устанавливается в <Length> | <percentage> | inherit;
- text-shadow. Задает разделенный запятыми список теневых эффектов, которые нужно применить к тексту элемента. Устанавливается в. none | <color> || <length> <length> <length>? ,]* <color> || <length> <length> <length>?] j inherit;
- text-transform. Определяет выделение текста элемента заглавными буквами. Устанавливается в capitalize | uppercase | lowercase | none | inherit;
- top. Определяет, насколько край содержимого смещен ниже верхнего края содержащего его блока. Устанавливается в <length> | «percentage» | auto | inherit;
- treat-as-word-space. Определяет, нужно ли трактовать символ как разделитель между словами или как обычную букву. Устанавливается в auto | true | false | inherit;
- unicode-bidi. То же, что и свойство CSS2 с тем же именем. Устанавливается в normal | embed | bidi-override | inherit;
- vertical-align. Определяет вертикальное расположение. Устанавливается в baseline | middle | sub | super | text-top | text-bottom | «percentage» | <length> | top | bottom | inherit;
- visibility. Определяет, будут ли отображены создаваемые элементом прямоугольники. Устанавливается в visible | hidden | collapse | inherit;
- voice-family. Устанавливает семейство голоса. Устанавливается в [[«specific-voice> | <generic-voice>],]* [<specific-voice> | <generic-voice>] | inherit;
- volume. Задает громкость звука. Устанавливается в <number> | «percentage» | silent | x-soft | soft | medium | loud | x-loud | inherit;
- white-space. Определяет способ обработки символов-разделителей (white-space) внутри элемента. Устанавливается в normal | pre | nowap | inherit;
- white-space-collapse. Задает способ обработки последовательных символов-разделителей. Устанавливается в false | true | inherit;
- widows. Устанавливает минимальное количество строк абзаца, которые должны быть оставлены внизу страницы. Устанавливается в <integer> | inherit;
- width. Задает ширину содержимого прямоугольников, созданных элементами уровня блока и замененными элементами. Устанавливается в <length> | «percentage» | auto | inherit;
- word-spacing. Определяет промежуток между словами. Устанавливается в normal | <length> | <space> | inherit;

- wrap-option. Определяет наличие оболочки строк форматирующего объекта. Устанавливается в no-wrap | wrap | inherit;
- writing-mode. Определяет режим написания. Устанавливается в lr-tb | rl-tb | tb-rl j tr | rl | tb | inherit;
- xrpl:lang. Определяет язык и страну для средства форматирования. Устанавливается в <country-language> | inherit;
- z-index. Определяет уровень размещения элементов. Устанавливается в auto| <integer> | inherit.

Алфавитный указатель

#@%

<!DOCTYPE>, элемент, 241
<fo:basic-link>, элемент, 493
<fo:block>, элемент, 452, 473
<fo:block-container>, элемент, 477
<fo:character>, элемент, 484
<fo:conditional-page-master-reference>, 501
<fo:external-graphic>, элемент, 487
<fo:flow>, элемент, 452
<fo:footnote>, элемент, 490
<fo:footnote-body>, элемент, 490
<fo:initial-property-set>, элемент, 489
<fo:in!ine>, элемент, 483
<fo:layout-master-set>, элемент, 447
<fo:list-item>, элемент, 474
<fo:list-item-body>, элемент, 476
<fo:list-item-label>, элемент, 475
<fo:page-number>, элемент, 486
<fo:page-sequence>, элемент, 450
<fo:page-sequence-master>, элемент, 499
<fo:repeatable-page-master-alternatives>, элемент, 500
<fo:root>, элемент, 446
<fo:simple-page-master>, элемент, 447
<fo:static-content>, элемент, 451, 501
<fo:table>, элемент, 461
<fo:table-body>, элемент, 464
<fo:table-cell>, элемент, 466
<fo:table-column>, элемент, 463
<fo:table-row>, элемент, 465
<xsl:apply-imports>, элемент, 105
<xsl:apply-templates>, элемент, 113
<xsl:attribute>, элемент, 247
<xsl:attribute>, элемент, 254
<xsl:attribute-set>, 254
<xsl:call-template>, элемент, 363
<xsl:**choose>, элемент, 191**
<xsl:comment>, элемент, 249
<xsl:copy>, элемент, 96, 141
<xsl:copy-of>, элемент, 144
<xsl:decimal-format>, элемент, 349
<xsl:**document>, элемент, 251**
<xsl:element>, элемент, 245
<xsl:fallback>, элемент, 226
<xsl:for-each>, элемент, 99, 198, 258, 360
<xsl:if>, элемент, 187
<xsl:import>, элемент, 104
<xsl:include>, элемент, 103
<xsl:key>, элемент, 174, 318, 375
<xsl:message>, элемент, 145
<xsl:namespace>, элемент, 254
<xsl:namespace-alias>, элемент, 385
<xsl:number>, элемент, 206
<xsl:**output>, элемент, 94, 97, 229**
<xsl:param>, элемент, 366
<xsl:**preserve-space>, элемент, 131**
<xsl:script>, элемент[1], 218
<xsl:sort>, элемент, 201, 394
<xsl:strip-space>, элемент, 130
<xsl:template>, элемент, 80, 111, 264

<xsl:text>, элемент, 121
<xsl:value-of>, элемент, 87, 114, 117
<xsl:variable>, элемент, 355
<xsl:when>, элемент, 193
<xsl:with-param>, элемент, 367

A

ActiveXObject, класс, 398
ancestor, ось XPath, 287
ancestor-or-self, ось XPath, 288
ASP, Active Server Pages, 418

B

Boolean test, 162
boolean, функция XPath, 345

C

CDATA
    создание раздела, 262
ceiling, функция XPath, 342
comment()
    условие узла, 157
concat, функция XPath, 330
contains, функция XPath, 331
CSS, 34

current, функция XSLT, 307

D

descendant, ось XPath, 289
descendant-or-self, ось XPath, 290
document, функция XSLT, 308
DSSSL, 23
DTD, 29

E

element-available(), функция XSLT, 311
element-available, функция XSLT, 226
external object, 223

F

false, функция XPath, 346
floor, функция XPath, 342
following, ось XPath, 291
following-sibling, ось XPath, 292
fop, процессор XSL-FO, 445
format-number(), строка XSLT, 311
format-number, функция XSLT, 349
function-available, функция XSLT» 222, 314

G

generate-id, функция XSLT, 258, 315
getError, функция, 391

I

ID, 173
id, функция XPatb, 325
import precedence, 104
inline object, 482
Internet Explorer, 32, 42, 45. 220, 389
    таблицы стилей, 107

3

Java, 37
    виртуальная машина Microsoft, 40
    компилятор, 405
    серверные страницы, JSP, 421
    сервлет, 422
JavaScript, 45, 389
    генерация из XML, 232
JSP, Java Server Pages. 421

K

key, функция XSLT, 318

L

lang, функция XPath, 346
last, функция XPath, 326
local-name, функция, 327
location step, 284

M

master, 447
match pattern, 115, 147
mode, 264
MSXML3, 45, 220

N

name, функция XPath, 327
namespace, ось XPath, 293
namespace-uri, функция XPath, 327
node
    условие узла, 158
node set, 164, 275
node test, 157
normalize-space, функция XPath, 333
not, функция XPathf 347

O

Or, операция, 176
Oracle, 41
  процессор XSLT, взаимодействие с Java, 408

P

parent, ось XPath, 294
pattern, 83
PDF, 25
position test, 161
position, функция XPath, 328
preceding, ось XPath, 295
preceding-sibling, ось XPath, 297

R

region, 448
round, функция XPath, 344
RTF, 97, 240

S

Saxon, процессор XSLT, 40
  взаимодействие с Java, 405
self, ось XPath, 298
servlet, 422
SGML, 23
SQL, 24
starts-with, функция XPath, 334
string, функция XPath, 336
string-length, функция XPath, 338
stylesheet, 33
substring, функция XPath, 340
substring-after, функция XPath, 341
substring-before, функция XPath, 341
sum, функция XPath, 345
system-property, функция XSLT, 320

T

tag, 28
translate, функция XPath, 341
true, функция XPath, 349

U

Unicode, 55
unparsed-entity-uri(), функция XSLT, 320
URI, 34, 92

W

W3C, 23
  спецификации, 25
web-сервер, 47
whitespace, 129

X

Xalan, процессор XSLT, 41
  взаимодействие с Java, 403
XHTML, 27, 51, 241
XML, 22
  кодировка, 235
  объявление, 256
  редактор, 56
  схема, 29
  участки, 399
XML Base, 92
XML links, 92
XPath, 68, 69, 99, 111, 272
  логические значения, 282
  набор узлов, 275
  определение W3C, 272
  ось, 285
  предикат, 286
  представление чисел, 169
  сокращенный синтаксис, 300
  типы данных, 275
  условие узла, 286
  числа, 278
XPath 2.0, 302
  планируемые функции, 353
  предикаты, 163
XSL, 22
XSL-FO, 23, 24, 59, 425
  блоки, 452
  колонки, 496
  процессор, 59
  ресурсы, 59
  ссылки, 493
  таблицы, 456
  форматирующие объекты, 427
  форматирующие свойства, 430
  шаблоны, 447
XSLT, 22
  версии, 26
  встроенные функции, 306
  инструкции, 81
  код Java, 216
  преобразование, 35
  применение с ASP, 418
  применение с JSP, 421
  применение с сервлетами Java, 422
  процессор, 35, 36

XSLT (*продолжение*)
расширения, 214
редактор, 55, 57
утилиты, 58
элементы, 74
XSLT 2.0
планируемые функции, 353
XT, процессор XSLT, 39

А

активные серверные страницы, ASP, 418

Б

блок, 452
браузер, 32

В

внешний объект, 223
встроенный объект, 482
выравнивание, 132, 235

Д

дерево, 68
документ XML, 27
допустимый, 29
хорошо сформированный, 29

И

именованный шаблон, 363
значение по умолчанию, 374
параметры, 366
рекурсивный вызов, 372
инструкции обработки, 75
инструкция обработки
<?xsl:stylesheet?>, 75
информационное множество, 73

К

ключ, 174
кодировка
установка, 263
колонки, 496
корневой элемент, 69

М

метод вывода, 93, 230
HTML, 95, 231
XML, 97, 234
текст, 97, 239

Н

набор узлов, 164, 275
функции, 275
нумерация, 206
на произвольных уровнях, 210
одноуровневая, 207

О

область, 448
область видимости, 356
образец, 83
образец выбора, 115, 147
абсолютный, 153
в терминах XPath, 152
выбор по ID, 173
выбор по ключу, 174
определение W3C, 153
относительный, 153
правила сокращения, 156
примеры, 178
образец шага, 153
ось, 154
предикат, 161
условие узла, 157
объявление XML, 256
объявление типа документа, DTD, 29
ось, 154
XPath, 285
ошибки разбора, 390

П

параметр
задание значений в командной строке, 375
параметры, 366
переменная, 355
область видимости, 356
потомок, 70
предикат, 148, 161, 286
логические значения, 166
набор узлов, 164
предикаты
сокращенный синтаксис, 173
числа, 169
предок, 70
преобразование, 35
XML-XML, 49
из XML в XHTML, 51
создание нескольких результирующих документов, 381

приоритет, 139
пространство имен, 78
процессор
  XSL-FO, 426
процессор XSLT, 35, 36, 80
  API для связи с Java, 402
  Oracle, 41
  Saxon, 40
  Xalan, 41
  XT, 39
  и разборщик XML, 73
  на Java, Л8
путь расположения
  абсолютный, 284
  относительный, 284
  примеры 299

Р

рабочий проект, 25
разборщик XML, 73
расширения XSLT, 214
РБНФ, 273
редактор
  XML, 56
  XSLT, 55, 57
режим, 264
рекомендации, 25
рекурсия, 372

С

серверные страницы Java, JSP, 421
сервлет, 422
символ-разделитель, 71, 129
сортировка, 201
  по нескольким критериям, 205
спецификации, 25
ссылки, 493
схема XML, 29

Т

таблица стилей, 33
  встроенная, 100
  упрощенная, 99
таблица, XSL-FO, 456
тег, 28
требования, 25

У

узел, тип, 69
узел-разделитель, 72
условие
  логическое, 162
  позиции, 161
  узла, 157, 286
участок XML, 399

Ф

форматирующие объекты, 427
  встроенные, 482
  потока, 451
форматирующие свойства, 430
функции
  XSLT, 306
  для работы с набором узлов, 275
  планируемые в XPath 2.0, 353
  планируемые в XSLT 2.0, 353
  расширения, 215

Ш

шаблон, 111
  выбор, 116
  значения атрибута, 127
  именованный, 363
  приоритет, 139
шаблон XSL-FO, 447
  страницы, 447, 499
шаг расположения, 284

Э

элемент
  буквального результата, 82
  создание на этапе выполнения, 245
элемент XSL
  <xsl:stylesheet>, 77
  <xsl:for-each>, 143
элементы XSL
  высокого уровня, 79
элементы XSLT, 74
элементы расширения, 223

Щ

Стивен Холзнер

БИБЛИОТЕКА ПРОГРАММИСТА

XSLT

Добро пожаловать в XSLT, книгу, посвященную языку преобразований расширенной таблицы стилей (Extensible Stylesheet Language Transformations). Книга писалась так, чтобы быть настолько исчерпывающей (и доступной), насколько это возможно для единственной книги по XSLT. XSLT представляет собой сродство преобразования документов XML в другие документы XML или в документы совершенно другого типа; сейчас эта тематика очень популярна. Автор постарался написать наиболее полную книгу по XSLT и считает, что это ему удалось.

Книга рассчитана на всех, кто хочет изучать XSLT и способы его использования на практике. Единственное требование — иметь представление о том, как создавать документы XML, и HTML.
Нет необходимости быть экспертом в XML, однако общее представление об этом языке будет нелишним, Больше никакой специальной подготовки не требуется.

В книге полностью описаны официальные спецификации XSLT. установленные консорциумом World Wide Web Consortium.

- создание выходных данных в формате XML, HTML, RTF и простого текста
- рекурсивная обработка шаблонов
- поиск совпадений элементов, дочерних и производных элементов, атрибутов и т. д.
- реализация конструкций условного выбора
- обработка множественного выделения
- использование процессоров Oracle, Saxon, XML4Java, XT в коде Java при помощи вызовов API
- использование JavaScript для XSLT в Internet Explorer
- форматирующие объекты XSL-FO
- преобразования XSLT на стороне сервера при помощи JSP, ASP и сервлетов Java

Уровень пользователя:
опытный/эксперт
Серия:
библиотека программиста

ISBN 5-94723-271-5

Посетите наш web-магазин www.piter.comx

New Riders
ПИТЕР
WWW PITER COM