bhv

Ильдар Хабибуллин

САМОУЧИТЕЛЬ

XML

Конструкции и реализации языка XML

Описание документов средствами XSD

XQuery как инструмент для извлечения данных

Оформление стилей с помощью XSL-FO

Теория и практика создания XML-документов

Ильдар Хабибуллин

САМОУЧИТЕЛЬ XML

Санкт-Петербург
«БХВ-Петербург»
2003

УДК 681.3.068+800.92XML
ББК 32.973.26-018.1
X12

Хабибуллин И. Ш.

X12 Самоучитель XML. — СПб.: БХВ-Петербург, 2003. — 336 с.: ил.

ISBN 5-94157-339-1

Книга предназначена для самостоятельного изучения новой компьютерной технологии — XML (eXtensible Markup Language, расширяемый язык разметок), быстро проникающей буквально во все области обмена информацией. Представлены новейшие аспекты технологии XML — язык описания документов XSD, язык создания запросов XQuery, форматирование документов на языке XSL-FO. Начав с основ технологии XML, автор выводит читателя на уровень самостоятельного создания программ-обработчиков документов XML. Изложение основано на авторском курсе лекций, его отличает краткость и простота. Большое количество примеров и упражнений позволяет глубоко освоить материал.

Для программистов

УДК 681.3.068+800.92XML
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Редактор	*Наталья Сержантова*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 11.09.03.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 27,1.
Тираж 4000 экз. Заказ №
"БХВ-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар № 77.99.02.953.Д.001537.03.02
от 13.03.2002 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в Академической типографии "Наука" РАН
199034, Санкт-Петербург, 9 линия, 12.

ISBN 5-94157-339-1

Содержание

Предисловие

Еще несколько десятков лет назад придуман удобный способ хранения представленной в виде таблиц информации — реляционные базы данных. Всевозможные списки, реестры, счета, квитанции сохраняются в таблицах баз данных и легко извлекаются оттуда в удобной для анализа форме. Гораздо хуже обстоит дело с информацией, которую неудобно представить в виде таблицы. Статьи, книги, отчеты, приказы, инструкции приходится хранить в виде файлов или "складывать" в одну ячейку таблицы реляционной базы данных. Это значительно затрудняет решение очень важной и актуальной задачи — поиска и извлечения информации.

Есть много подходов к решению этой задачи. Из документов выбираются ключевые слова, строятся полные индексы и словари, применяются методы нечеткого поиска. Еще один подход, приведший к появлению XML (eXtensible Markup Language, расширяемый язык разметок), заключается во внесении структуры в документ. Раз уж не удается разбить документы на строки и столбцы, внеся в них структуру таблицы, то, может быть, удастся создать другую структуру? С незапамятных времен документы разбивают на главы, разделы, параграфы, абзацы. Это значительно облегчает поиск информации по оглавлению документа. Если развить и уточнить правила разбиения документов на отдельные элементы, то можно будет добиться быстрого поиска и выделения нужных сведений.

Эта идея разметки и разбиения документов оказалась чрезвычайно плодотворной. В сущности, весь Интернет в той его части, которая называется World Wide Web, пользуется ею. Технология XML бурно развивает ее уже более пяти лет, охватывая все новые и новые области хранения и обработки данных. Разбиение документа на структурные элементы облегчает не только поиск, но и анализ данных, получение из них новых выводов, того, что называется "data mining". Поэтому все больше и больше информации хранится в виде документов XML. Многие системы управления базами данных, такие как Oracle, предоставляют средства удобного хранения документов XML.

Разрабатываются базы данных, специально предназначенные для хранения информации в виде XML. Самые популярные браузеры, такие как Internet Explorer, Mozilla, Opera, "учатся" показывать документы XML. Многие текстовые процессоры, даже Microsoft Office, начинают переходить на формат XML.

Технология XML сейчас находится на подъеме. Она еще сравнительно невелика, ее описание уместилось в одну книгу, которую вы сейчас держите в руках. Будущее предвещает ее дальнейшее развитие, значительное усложнение и укрупнение. Поэтому если уж изучать XML, то начинать надо прямо сейчас!

Введение

Как известно, компьютер "понимает" только язык электромагнитных сигналов, имеющих два уровня. Один уровень обозначается нулем, другой — единицей. Поэтому самый простой способ записи текста в компьютере заключается в том, чтобы закодировать каждый символ текста какой-нибудь последовательностью нулей и единиц. Имея в распоряжении один электромагнитный импульс, можно закодировать два символа, обозначив один из них единицей, а другой нулем. С помощью двух импульсов можно закодировать уже четыре символа. Например, можно обозначить букву А комбинацией 00, букву Б — 01, букву В — 10, а букву Г — двумя единицами 11. Три импульса дают возможность закодировать восемь символов и т. д.

Обычные английские тексты используют около сотни символов: заглавные и строчные буквы латиницы, цифры, знаки препинания, некоторые математические символы. Для их кодирования применяют комбинации из семи символов, набор которых давно стандартизирован. Этот стандарт называется ASCII (American Standard Code for Information Interchange). Число семь неудобно для компьютера, в машине лучше оперировать степенями двойки. Поэтому в коде ASCII обычно применяют комбинации не из семи, а из восьми нулей и единиц — *байты*, начиная каждую комбинацию с нуля. В стандарте ASCII с использованием восьми разрядов первая буква английского алфавита A кодируется комбинацией 01000001, вторая буква B — комбинацией 01000010, а, например, знак процента % — комбинацией 00100101.

Многие языки, в том числе и русский, требуют записи дополнительных символов — букв кириллицы, символов с диакритическими знаками. Такие символы просто добавляются к набору ASCII. Для их записи используют комбинации из восьми нулей и единиц, начинающиеся с единицы. Разработаны международные стандарты для кодирования дополнительных символов. Символы европейских алфавитов с диакритическими знаками образуют кодировку ISO8859-1, часто называемую Latin 1. Буквы греческого алфавита попали в кодировку ISO8859-2, обычно определяемую как Latin 2.

Для кодирования кириллицы разработана международная кодировка ISO8859-5, но она редко применяется в России. Гораздо чаще используются другие кодировки. На русифицированных компьютерах, управляемых операционными системами MS Windows, применяется кодировка CP1251, разработанная корпорацией Microsoft. На машинах, работающих под управлением операционных систем семейства UNIX, используется кодировка KOI8-R. Со времен MS DOS осталась кодировка CP866. Машины Apple Macintosh применяют кодировку MacCyrillic.

Это многообразие кириллических кодировок приводит к дополнительной нагрузке на пользователя. Для того чтобы правильно прочитать кириллический текст, он должен знать, в какой кодировке текст был записан, или перебрать несколько кодировок в своем текстовом редакторе.

Итак, для записи простого, как говорят, "плоского" (flat) текста, применяются байтовые кодировки: один символ — один байт. Это удобно для передачи текста. На любом компьютере есть средства для чтения плоского текста. Если знать его кодировку, то текст будет правильно прочитан. Мировое компьютерное сообщество решило освободиться и от этого условия и перейти на двухбайтовую шестнадцатиразрядную кодировку, позволяющую закодировать 65 536 символов. Такая кодировка называется *Unicode*. В ней перед каждым кодом Latin 1 стоит нулевой байт, перед каждым кодом кириллицы — байт 00000100 и т. д. Для текстов, записанных в Unicode, нет необходимости указывать кодировку, но сам текст занимает ровно в два раза больше места в файле.

Компромиссное решение предоставляет кодировка *UTF-8* (Unicode Transfer Format). В ней символы, входящие в ASCII, кодируются одним байтом, национальные символы большинства языков — двумя байтами, зато менее распространенные символы кодируются тремя байтами.

Выделение фрагментов текста

Часто возникает необходимость выделения каких-то фрагментов текста курсивом, подчеркиванием, жирным шрифтом или каким-то другим способом. Иногда надо увеличить размер шрифта, изменить его начертание, вставить в текст буквицу, формулу, какое-то изображение. У байтовых кодировок и у Unicode нет таких возможностей, ведь для курсива или подчеркнутых символов нужно задавать дополнительные коды. Для выделения фрагментов текста надо вводить новые средства. Развитие текстовых редакторов пошло двумя путями.

Первый путь — применение графических текстовых редакторов, таких как MS Word. Они используют не кодировку символа, а описание его графического изображения. Полученное описание записывается в файл специального формата, прочитать который может только такой же текстовый редактор.

Второй путь — внести в текст пометки, указывающие на особенности того или иного фрагмента текста. Этот способ давно применяется в полиграфии. При подготовке документа к печати его рукописный вариант испещряется корректурными знаками и цветными пометками, указывающими наборщику, какой шрифт выбрать при наборе текста. Именно по этому пути пошла Web-технология, успешно воплотив его в гипертекстовом языке разметок HTML (Hypertext Markup Language). В этом языке пометки, называемые *тегами* (tags), записываются в угловых скобках, чтобы отличить их от символов самого текста. Некоторые теги языка HTML показаны в листинге В1.

Листинг В1. Текст с тегами языка HTML

```
<html>
<body>
<h2 align=center>Заголовок</h2>
Обычный текст, <i> курсив</i>, снова обычный текст.<p> Теперь
<u>подчеркнутый текст</u>. <p>Текст <font size=+3> увеличенного
размера. </font>
</body>
</html>
```

В листинге В1 тег `<h2>` (header) показывает, что дальше идет заголовок второго уровня. Параметр `align=center` тега `<h2>` предписывает размещение заголовка посередине окна браузера. Тег `<i>` (italic) означает, что дальше пойдет курсив, а тег `<u>` (underline) — что дальше надо выводить текст с подчеркиванием. Тег `<p>` (paragraph) начинает новый абзац. Тег `<font>` изменяет шрифт текста, в частности параметр `size=+3` увеличивает размер шрифта на три единицы.

Как видите, многим тегам языка HTML соответствуют закрывающие теги, отмеченные наклонной чертой. Они отменяют действие открывающих тегов, возвращая текст к первоначальному виду. Сам текст записывается между тегами `<body>` и `</body>`, а весь документ — между тегами `<html>` и `</html>`.

Текст с тегами языка HTML "понимают" все браузеры. На рис. В1 показано, как выглядит текст листинга В1 в окне браузера Internet Explorer.

Язык разметок HTML прост и нагляден, но у него есть два существенных ограничения. Во-первых, набор тегов HTML строго фиксирован, его нельзя расширить или изменить. Все браузеры должны таким образом интерпретировать одинаковые теги, чтобы пользователь, написавший текст с разметками HTML, был уверен, что этот текст будет одинаково выглядеть во всех браузерах. Во-вторых, теги языка HTML показывают только визуальную разметку, внешний вид документа, но ничего не говорят о его структуре.

Например, заголовки следует помечать одинаковым тегом, например тегом <h2>, даже если все они одного уровня.

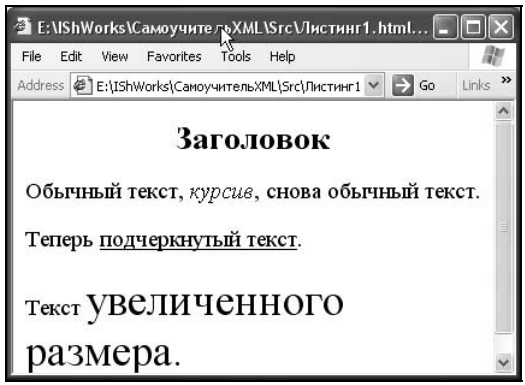


Рис. В1. Текст HTML в окне браузера

Простота языка HTML перевесила все его недостатки. Она привела к взрывному росту числа Web-сайтов и авторов многочисленных Web-страничек. Обычные пользователи компьютеров ощутили себя творцами, получили возможность заявить о себе, высказать свои мысли и чувства, найти в Интернете своих единомышленников.

Ограниченные возможности языка HTML быстро перестали удовлетворять поднаторевших разработчиков, почувствовавших себя профессионалами. Введение таблиц стилей CSS (Cascading Style Sheet) и включений на стороне сервера SSI (Server Side Includes) лишь ненадолго уменьшило недовольство разработчиков. Профессионалу всегда не хватает средств разработки, он постоянно испытывает потребность добавить к ним какое-то свое средство, позволяющее воплотить все его фантазии.

Стандарт SGML

Такая возможность есть. Еще в 1986 году стал стандартом язык создания языков разметки SGML (Standard Generalized Markup Language), с помощью которого и был создан язык HTML. Основная особенность языка SGML заключается в том, что он позволяет создать новый язык разметок, определив набор тегов создаваемого языка. Каждый конкретный набор тегов, разработанный по правилам SGML, снабжается описанием DTD (Document Type Definition) — *определением типа документа*, разъясняющим связь тегов между собой и правила их применения. Специальная программа — драйвер принтера или SGML-браузер — руководствуется этим описанием для печати или отображения документа на экране дисплея.

В это же время выявилась еще одна, самая важная область применения языков разметки — поиск и выборка информации. В настоящее время подавляющее большинство информации хранится в реляционных базах данных. Они удобны для хранения и поиска информации, представляемой в виде таблиц: анкет, ведомостей, списков и т. п., — но неудобны для хранения различных документов, планов, отчетов, статей, книг, которые не могут быть представлены в виде таблиц. Между тем, человечество накопило несметное количество таких документов. Большинство из них уже переведено в электронную форму, как правило, в байтовых кодировках. Очень часто ставится задача выбрать какие-то определенные сведения из таких документов: адреса, фамилии, даты, решения, резюме.

Тегами языка разметки можно задать структурную, а не визуальную разметку документа, разбить документ на главы, параграфы и абзацы или на какие-то другие элементы, выделить важные для поиска участки документа. Например, можно определить тег `<address>` и помечать им все адреса, встречающиеся в тексте. Легко написать программу, анализирующую размеченный такими тегами документ и извлекающую из него нужную информацию. В нашем примере анализирующая программа просто извлечет весь текст, заключенный между тегами `<address>` и `</address>`.

Язык SGML оказался слишком сложным, требующим тщательного и объемистого описания элементов создаваемого с его помощью языка. Только одна его спецификация содержит более пятисот страниц. Он применяется лишь в крупных проектах, например, для создания единой системы документооборота крупной фирмы. Скажем, man-страницы операционной системы Solaris Operational Environment написаны на специально сделанной реализации языка SGML.

Язык разметок XML

Золотой серединой между языками SGML и HTML стал язык XML (eXtensible Markup Language) — расширяемый язык разметок. Это подмножество языка SGML, избавленное от излишней сложности, но позволяющее разработчику Web-страниц создавать свои собственные теги. Язык XML достаточно широк, чтобы можно было создать все нужные теги, и достаточно прост, чтобы можно было быстро их описать.

Разработка XML началась в 1996 году. Ею занимается общественная организация W3C (World Wide Web Consortium), основной сайт которой находится по адресу **http://www.w3c.org/**. Сведения об XML собраны на странице **http://www.w3c.org/XML/**. В 1998 году консорциум выпустил спецификацию XML версии 1.0. Она постоянно совершенствуется, последний вариант спецификации всегда находится по адресу **http://www.w3c.org/TR/rec-xml**.

Появление новой версии Unicode 3.0 вызвало необходимость перевода на нее XML. Консорциум W3C начал разработку второй версии XML 1.1. Во

время написания этих строк она находилась в начальной стадии, появилась только рекомендация. Ее текущее состояние можно посмотреть по адресу **http://www.w3c.org/TR/xml11**. Уже создана версия Unicode 4.0, поэтому, видимо, выпуск XML 1.1 опять будет отложен.

Правила написания документа XML не сложны. Мы подробно рассмотрим их в *главе 1*. Но с каждой разновидностью документов XML надо связать ее описание DTD (Document Type Definition) или какое-то другое описание структуры и способа разметки документов. Такое описание часто бывает сложнее самого документа. Оно выполняется на специальном языке описаний, значит, надо изучить и этот язык. Мы посвятим тщательному разбору таких языков описаний *главы 2* и *3*.

Одного описания документа недостаточно для успешной работы с ним. Нужны еще средства поиска информации в документе, перекрестные ссылки одних частей документа на другие. Этому посвящены следующие главы книги.

Но и этого мало для работы с документами. Необходимо уметь быстро извлекать из документа нужную информацию, уметь преобразовывать документ и представлять его в различных видах. Предпоследние главы книги подробно объясняют способы решения этих задач, предложенные технологией XML.

Наконец, в *главах 10* и *11* раскрывается "кухня" технологии XML, показываются методы работы с документами XML, объясняется, как создаются программы-обработчики этих документов.

Структура книги

Книга состоит из одиннадцати глав, разбитых на две части.

В *части I* рассматриваются конструкции языка XML и его реализаций, непосредственно связанных с технологией XML.

Глава 1 описывает конструкцию самого документа XML. Из нее вы узнаете, из чего состоит документ XML, как правильно записать элементы XML и научитесь корректно составлять хорошо оформленные документы XML.

К каждому хорошо оформленному документу XML должно быть приложено его описание. Его можно сделать по-разному. Из *главы 2* вы узнаете, как делать описание документа на языке DTD (Document Type Definition). Этот язык, пришедший в XML из стандарта SGML, оказался не совсем подходящим для описания документов XML. Он недостаточно подробен, у него не хватает средств для полного описания всех конструкций XML и сейчас он постепенно заменяется другими языками. Тем не менее его надо знать, потому что уже тысячи документов снабжены такими описаниями.

В *главе 3* описан наиболее мощный язык описания документов XML — язык XSD (XML Schema Definition Language). Этот язык сам является реализацией XML. Описание документа, сделанное на нем, само будет документом XML. Эта особенность языка XSD позволяет автоматически обрабатывать описания как обычный документ XML. Прочитав *главу 3*, вы научитесь точно и правильно описывать любой документ на языке XSD.

Успех языка HTML во многом определен возможностью легко создавать гиперссылки на другие документы или на иные точки того же самого документа. Такая возможность есть и в технологии XML. Ее предоставляет язык XLink (XML Linking Language). Возможности языка XLink гораздо шире, чем возможности тега `<a>` языка HTML. Вы изучите их, прочитав *главу 4*.

Иногда надо сделать ссылку не на определенный элемент документа, а на что-нибудь вроде "третьего абзаца пятого параграфа четвертого раздела договора". Это легко сделать на языке XPointer, также входящем в технологию XML. Изучению этого языка и способам его практического применения посвящена *глава 5*.

Глава 6 также познакомит вас со ссылками, но не на одно конкретное место документа, а, например, на некоторое множество элементов или на определенное слово, где бы оно ни находилось. Вы не удивитесь, узнав, что такие ссылки делаются на специальном языке адресации XPath. Это очень мощный язык, используемый во многих других языках, входящих в технологию XML. Область его применения все время расширяется и уже сейчас его просто необходимо знать для понимания особенностей большого числа конструкций XML.

Всем известно, какую большую роль играет язык SQL в работе с базами данных. Поскольку сейчас XML все чаще применяется для хранения документов в базах данных, нужны какие-то средства для быстрого и точного извлечения данных самого разного вида из документов XML. Такие средства предоставляет новый язык XQuery, только что прошедший стадию предварительной разработки. Его подробному изложению посвящена *глава 7*.

На этом первая часть книги, посвященная описанию базовых средств технологии XML, заканчивается. Прочитав ее, проделав упражнения и ответив на вопросы, вы можете быть уверены, что понимаете суть технологии XML и способны написать хорошо оформленные документы XML, содержащие все средства, облегчающие работу с ними.

Часть II книги описывает средства обработки готовых документов XML.

В *главе 8* подробно рассмотрены конструкции языка преобразования документов XML, называемого XSLT (eXtensible Stylesheet Language for Transformations). Этот язык позволяет на базе одного документа XML автоматически построить другой, может быть, имеющий совершенно иную структу-

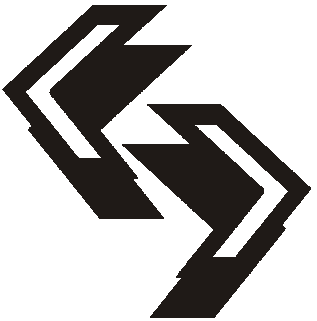
ру, или выделить какие-то части документа и составить на их основе новый документ или даже несколько документов. Если вам надо собирать и размножать сведения, содержащиеся в документах XML, то язык XSLT поможет вам в этом.

Каждый документ XML, в конечном счете, должен быть распечатан или выведен на экран дисплея, сотового телефона, пейджера или даже на какое-то голосовое устройство. Для этого он должен быть предварительно отформатирован в расчете на конкретное представление в устройстве вывода. Такое представление можно сделать на языке XSL-FO (eXtensible Stylesheet Language Formatting Objects), описанию которого посвящена *глава 9*.

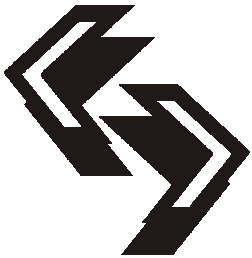
Две последние главы книги — *10* и *11* — предназначены для любителей программирования, которые хотят понять алгоритмы обработки документов XML. В них изложены методы разбора, анализа и обработки документов XML. В *главе 10* рассматривается обработка документов, основанная на событиях, а в *главе 11* — обработка, основанная на построении дерева документа в оперативной памяти. Освоив материал этих глав, вы сможете написать свои собственные программы-обработчики документов XML.

Технология XML развивается очень быстро и бурно. В настоящее время она еще сравнительно проста и обозрима в пределах одной небольшой книги. Через несколько лет XML обрастет сложными и громоздкими конструкциями, овладеть которыми будет нелегко. Чем раньше вы приступите к изучению конструкций XML, тем легче вам будет следить за развитием данной технологии и быть в курсе всех событий, связанных с ней. Давайте начнем!

ЧАСТЬ I



Конструкции языка XML и его реализаций



ГЛАВА 1

Структура документа XML

Вы, наверное, уже поняли из введения, что язык XML не содержит готового набора тегов, подобно HTML, а описывает правила создания своих собственных тегов. Эти теги будут размечать ваши документы так, как это нужно. После создания тегов вы получаете одну из *реализаций* языка XML и используете ее для разметки документов. Поэтому точнее говорить не "документ XML", а "документ, размеченный тегами, созданными по правилам XML". Мы не будем употреблять такую длинную фразу, но полезно иметь ее в виду. В тех случаях, когда надо будет подчеркнуть, что речь идет о конкретном наборе тегов, мы будем говорить о "реализации XML".

Уже создано много общепринятых реализаций XML. Наиболее известны такие языки:

- XHTML — язык гипертекстовой разметки HTML, приведенный в соответствие с правилами XML;
- MathML — язык записи математических формул;
- CML — язык записи химических формул;
- VoxML — язык записи звуков;
- WML — язык, применяемый в беспроводной технологии.

Есть еще множество других языков, их число растет с каждым днем.

Сам язык XML задает только общие правила, по которым создаются теги и оформляется документ XML. Эти правила изложены в спецификации XML, расположенной по адресу **http:/www/w3.org/TR/REC-xml**. Документ XML, написанный с учетом всех правил спецификации, называется *хорошо оформленным* (well-formed) документом. Хорошо оформленный документ будет гарантированно читаться и анализироваться любым XML-редактором, хотя результат анализа может оказаться совершенно неожиданным. Для того что-

бы XML-редактор правильно проанализировал прочитанный им документ и "понял" смысл тегов, необходимо, чтобы документ был не только хорошо оформленным, но и *верным* (valid).

Правила оформления документа XML, изложенные в спецификации XML, не очень обременительны. В этой главе мы их подробно изучим. По спецификации XML документ начинается с необязательного пролога.

Пролог документа XML

Пролог (prolog), начинающий документ XML, состоит из двух частей.

В первой части пролога, занимающей одну строку, записывается *объявление XML* (XML Declaration). Оно заключено между символами `<?...?>`, содержит пометку `xml` и номер версии спецификации XML. Все это вместе выглядит так:

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

Кроме этого, объявление XML может содержать указание на кодировку символов, в которой написан документ. По умолчанию считается, что документ записан в кодировке UTF-8. Написанное выше объявление XML эквивалентно такому:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Большинство текстовых редакторов в России работает в кодировке CP1251, поэтому объявление XML часто выглядит так:

```
<?xml version="1.0" encoding="windows-1251"?>
```

В объявление можно поместить еще параметр `standalone` со значениями `"yes"` или `"no"`. Значение `"no"` показывает, что документ использует определения элементов, сделанные в другом, внешнем, документе. По умолчанию принимается значение `"yes"`. Итак, полное объявление XML может выглядеть следующим образом:

```
<?xml version="1.0" encoding="windows-1251" standalone="yes"?>
```

Вторая часть пролога — *объявление типа документа,* DTD (Document Type Declaration) — может занимать одну или несколько строк. В этой части объявляются теги, использованные в документе, или приводится ссылка на файл, в котором записаны такие объявления. Объявление типа документа начинается с символов `<!DOCTYPE`, а заканчивается угловой скобкой — знаком "больше" `>`. Его содержимое зависит от способа объявления тегов. Способам объявления типа документа посвящены следующие две главы, а сейчас перейдем к примерам.

Пример: разметка адресной книжки

Создавая описание документа на языке XML, надо, прежде всего, продумать структуру документа. Пусть мы решили, наконец, упорядочить свою записную книжку с адресами и телефонами. В ней записаны фамилии, имена и отчества родственников, сослуживцев и знакомых, их дни рождения, адреса, состоящие из почтового индекса, названия города, улицы, номера дома и квартиры, а также телефоны, если они есть: рабочие и домашние. Мы определим теги для выделения каждого из этих элементов, продумаем связь между тегами и получим структуру, показанную в листинге 1.1.

Листинг 1.1. Пример XML-документа

```
<?xml version="1.0" encoding="Windows-1251"?>
<!DOCTYPE notebook SYSTEM "ntb.dtd">

<notebook>

  <person>

    <name>
      <first-name>Иван</first-name>
      <second-name>Петрович</second-name>
      <surname>Сидоров</surname>
    </name>

    <birthday>25.03.1977</birthday>

    <address>
      <street>Садовая, 23-15</street>
      <city>Урюпинск</city>
      <zip>123456</zip>
    </address>

    <phone-list>
      <work-phone>2654321</work-phone>
      <work-phone>2654023</work-phone>
      <home-phone>3456781</home-phone>
    </phone-list>

  </person>

  <person>

    <name>
      <first-name>Мария</first-name>
```

```
      <second-name>Петровна</second-name>
      <surname>Сидорова</surname>
    </name>

    <birthday>17.05.1969</birthday>

    <address>
      <street>Ягодная, 17</street>
      <city>Жмеринка</city>
      <zip>234561</zip>
    </address>

    <phone-list>
      <home-phone>2334455</home-phone>
    </phone-list>

  </person>

</notebook>
```

Разумеется, можно назвать теги по-другому, создать другой набор тегов и расположить их иначе. Например, выделить номер дома из адреса в отдельный элемент. Дата рождения может быть разбита на день, месяц и год рождения. Все это определяется теми целями, ради которых документ разбивается тегами на отдельные элементы. Если мы захотим в начале месяца выбрать имена всех своих знакомых, родившихся в этом месяце, то лучше выделить его в отдельный элемент. Это облегчит работу программе поиска.

Упражнения

1. Продумайте еще раз структуру записной книжки. Стоит ли разбивать адрес на более мелкие составляющие? Может быть, следует объединить имя, отчество и фамилию в один элемент? Нужно ли вводить в книжку дополнительные сведения? Как это сделать?
2. У вас большая библиотека. Продумайте структуру ее библиографического описания и создайте теги для оформления этой структуры.
3. Как бы вы записали структуру этой книги?

Корневой элемент

У хорошо оформленного документа все, что находится после пролога, обязательно должно содержаться в *корневом элементе* (root element). В листинге 1.1 это элемент, начинающийся с тега `<notebook>` и заканчивающийся тегом `</notebook>`. Такое требование возникло потому, что один документ

XML можно вложить в другой. При этом корневой элемент вложенного документа станет просто одним из элементов документа, в который он вложен. Такое вложение не нарушит структуру документа.

Имя корневого элемента считается именем всего документа и указывается во второй части пролога — объявлении типа документа (Document Type Declaration) — после слова DOCTYPE. Объявление типа документа записано во второй строке листинга 1.1. В ней после слова DOCTYPE и имени документа, в квадратных скобках должно идти *определение типа документа* — DTD (Document Type Definition):

```
<!DOCTYPE notebook [ Сюда заносится описание DTD ]>
```

Очень часто определение DTD составляется сразу для нескольких документов XML. В таком случае его удобно записать отдельно от документа. Если определение DTD отделено от документа, то во второй части пролога вместо квадратных скобок записывается одно из слов SYSTEM или PUBLIC. За словом SYSTEM идет адрес в форме URI файла с определением DTD, а за словом PUBLIC, кроме того, можно записать дополнительную информацию. Определение DTD дает возможность убедиться в *верности* документа.

Внимание!

Не путайте понятие "объявление типа документа" — DTD (Document Type Declaration) — вторую часть пролога, с понятием "определение типа документа" — DTD (Document Type Definition), которое вставляется во вторую часть пролога <!DOCTYPE>. К сожалению, оба понятия обозначаются аббревиатурой DTD.

Верный документ

Для описания адресной книжки в листинге 1.1 нам понадобились открывающие теги <notebook>, <person>, <name>, <address>, <street>, <city>, <zip>, <phone-list>, <work-phone>, <home-phone> и соответствующие им закрывающие теги, помеченные наклонной чертой. Теперь необходимо дать им объявление. В объявлении указываются только самые общие признаки логической взаимосвязи элементов и их тип. Документ, выдерживающий такую взаимосвязь, будет *верным* документом. Программа, анализирующая документ, сможет проверить правильность его разметки, сверив ее с объявлениями тегов.

Вот как может выглядеть объявление тегов листинга 1.1:

- элемент <notebook> может содержать в себе только нуль или больше элементов <person> и больше ничего;

- элемент <person> содержит ровно один элемент <name>, нуль или несколько элементов <address>, а также нуль или один элемент <phone-list>;
- элемент <name> имеет не более чем по одному элементу <first>, <second> и <surname>, содержимое которых — строки символов;
- элемент <address> содержит по одному элементу <street>, <city> и <zip>;
- элементы <street> и <city> содержат по одной текстовой строке;
- элемент <zip> содержит одно целое число;
- необязательный элемент <phone-list> содержит нуль или более элементов <work-phone> и <home-phone>;
- элементы <work-phone> и <home-phone> содержат по одной строке, состоящей только из цифр.

Это словесное описание, называемое *схемой* документа XML, формализуется несколькими способами. Наиболее распространены два способа: можно сделать определение DTD (Document Type Definition), пришедшее в XML из SGML, или описать схему на языке XSD (XML Schema Definition Language). Этим описаниям посвящены следующие две главы, а пока ограничимся словесным описанием.

Прочитав формализованное описание и узнав из него схему документа, программа-анализатор может проверить соответствие каждого документа его схеме и сделать вывод, верен этот документ или нет.

Элементы документа XML

Документ XML состоит из *элементов* (elements). Элемент начинается *открывающим тегом* (start-tag) в угловых скобках, затем идет необязательное *содержимое* (content) элемента, после него записывается *закрывающий тег* (end-tag) в угловых скобках. Например:

```
<surname>Сидорова</surname>
```

Закрывающий тег содержит наклонную черту, после которой повторяется имя открывающего тега.

Язык XML, в отличие от языка HTML, требует обязательно записывать закрывающие теги. Правда, из этого правила есть одно исключение. Если в содержимом элемента нет ни одного символа, даже пробела, то закрывающий тег можно не записывать. В этом случае открывающий тег должен заканчиваться символами />, например, в языке XHTML перевод строки записывается тегом

```
<br />
```

Эта запись равносильна записи

```
<br></br>
```

Элемент без всякого содержимого называется *пустым* (empty) элементом.

В записи `<br />` перед наклонной чертой оставлен пробел. Это не обязательно, но многие браузеры пропускают пустой элемент без такого пробела, искажая вид документа.

Сразу надо сказать, что язык XML, в отличие от HTML, различает регистры букв. Теги `<surname>`, `<Surname>` `<SURNAME>` — совершенно различные теги, не имеющие друг к другу никакого отношения.

Внимание!

Все реализации языка XML различают регистры букв!

Из листинга 1.1 видно, что содержимым элемента может быть другой элемент или несколько элементов, т. е. элементы документа XML могут быть вложены друг в друга. Надо следить за тем, чтобы элементы не пересекались, а полностью вкладывались друг в друга. Как уже говорилось выше, все элементы, составляющие хорошо оформленный документ, вложены в корневой элемент этого документа. Тем самым, хорошо оформленный документ наделяется структурой дерева, ветви которого состоят из вложенных элементов. На рис. 1.1 показана структура адресной книжки, описанной в листинге 1.1.

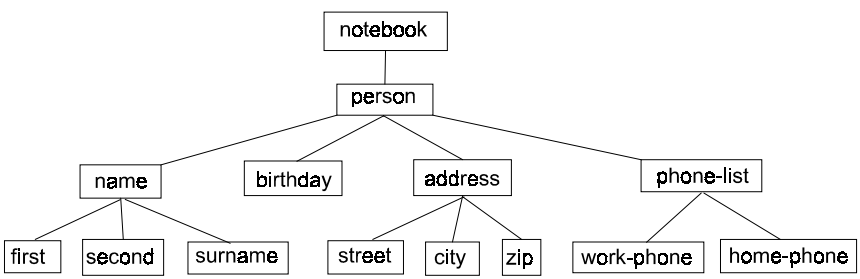


Рис. 1.1. Дерево элементов документа XML

Что делать, если в содержимом элемента есть знаки "больше" или "меньше"? Ведь их появление будет понято как начало или конец вложенного тега. В таком случае знак "меньше" заменяют строкой `<`, а знак "больше" — строкой `>`. "Увидев" символ амперсанда, программа-анализатор XML "поймет", что дальше идет условное обозначение знака "меньше" или "больше", завершающееся точкой с запятой. Но что делать, если в содержимое элемента надо вставить символ амперсанда? Правило таково: в содержимом элемента амперсанд заменяется строкой `&`.

Например, следующий фрагмент программного кода

```
if (x < MAX_VALUE && x > MIN_VALUE) proc1();
```

можно записать в документе XML так:

```
<progcode>
   if (x &lt; MAX_VALUE &amp;&amp; x &gt; MIN_VALUE) proc1();
</progcode>
```

Такие наборы символов необычного вида представляют собой примеры ссылок на сущности. Речь о них пойдет в следующем разделе, а пока надо заметить, что вместо символов "меньше", "больше" и амперсанда можно использовать их числовые коды. Они тоже записываются обозначениями специального вида — `&#60;`, `&#62;` и `&#38;` соответственно, — чтобы анализатор не принял их за обыкновенные числа. Коды можно записать в шестнадцатеричной форме, добавив букву "икс" — `&#x3C;`, `&#x3E;`, `&#x26;`. Обратите внимание на то, что коды завершаются точкой с запятой. С использованием приведенных выше обозначений предыдущий фрагмент выглядит так:

```
<progcode>
   if (x &#60; MAX_VALUE &#38;&#38; x &#62; MIN_VALUE) proc1();
</progcode>
```

Ссылки на сущности

Ссылки на сущности (entity references), начинающиеся с амперсанда и заканчивающиеся точкой с запятой, — это указание программе-анализатору подставить вместо них заранее определенную строку символов — *сущность*. Например, я могу записать ссылку на сущность `&author;`. Программа-анализатор подставит вместо нее фамилию автора. Почему? Как анализатор узнает, что именно надо подставить вместо ссылки? Ответ прост: сущности задаются в определении типа документа (Document Type Definition), которое мы рассмотрим в следующей главе.

Если текст содержит много знаков "больше", "меньше" и амперсандов, например, требуется весь вложенный элемент понимать как простую строку символов, то удобнее организовать так называемую секцию CDATA.

Секция CDATA

Секция CDATA (character data) начинается со строки `<![CDATA[`. Дальше записывается все то, что следует считать не разметкой, а простой строкой символов. Секция завершается двумя закрывающими квадратными скобками и знаком "больше". Например:

```
<![CDATA[<surname>Сидорова</surname>]]>
```