XML новые перспективы WWW

Бумфрей Ф., Диренцо О.,
Дакетт Й. и др.

Расширенный язык разметки

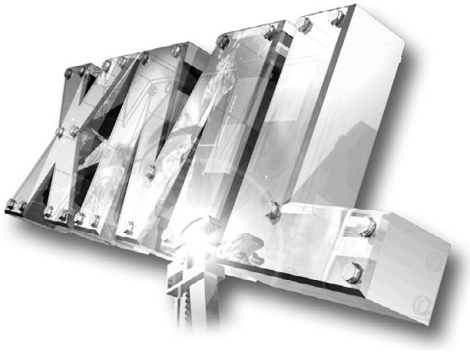
XML

для программистов

ДМК Издательство

Фрэнк Бумфрей, Оливия Диренцо, Йон Дакетт, Джо Грэф,
Дэйв Холэндер, Пол Хоул, Тревор Дженкинс, Питер Джоунс,
Эдриан Кингсли-Хьюз, Кэти Кингсли-Хьюз, Крэг Маккуин и Стивен Мор

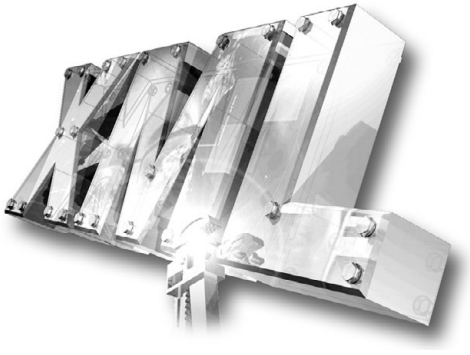
XML
Новые перспективы WWW



XML Applications

Frank Boumphrey,
Olivia Direnzo,
Jon Duckett,
Joe Graf, Paul Houle,
Dave Hollander,
Trevor Jenkins,
Peter Jones,
Adrian Kingsley-Hughes,
Kathy Kingsley-Hughes,
Craig McQueen
and Stephen Mohr

wrox
PROGRAMMERS TO PROGRAMMER™



Серия «Для программистов»

XML
Новые перспективы WWW

Фрэнк Бумфрей,
Оливия Диренцо,
Йон Дакетт,
Джо Грэф,
Дэйв Холэндер,
Пол Хоул,
Тревор Дженкинс,
Питер Джоунс,
Эдриан Кингсли-Хьюз,
Кэти Кингсли-Хьюз,
Крэг Маккуин
и Стивен Мор

ДМК
издательство

Москва

ББК 32.973.26-018.1
Б97

Бумфрей Ф., Диренцо О., Дакетт Й. и др.
Б97 XML. Новые перспективы WWW. Пер. с англ. – М.: ДМК. – 688 с.: ил. (Серия «Для программистов»).

ISBN 5-93700-007-2

В книге в сжатой форме излагаются основы XML – расширяемого языка разметки, а также приводятся примеры его практического использования. На сегодняшний день этот язык считается самым перспективным средством создания Web-документов. Широки его возможности и в качестве средства работы с базами данных и мощного механизма преобразования формата сообщения. Главные достоинства XML – гибкость, свобода в создании самых разнообразных тэгов, способность объединять информацию из различных источников в единый непротиворечивый документ. С языком XML тесно связаны самые новейшие разработки в Web-технологиях, такие как XML-схемы и пространства имен.

ББК 32.973.26-018.1

ISBN 1-861001-9-08 (англ.)
ISBN 5-93700-007-2 (рус.)

Содержание

Глава 1. Складываем мозаику XML

В настоящее время язык XML только-только «выходит в люди», поэтому изучать его – подлинное удовольствие. И это несмотря на то, что нынешний момент не самое лучшее время для такого знакомства. Еще ничего не устоялось, все – терминология, методики, коды, средства поддержки – находится в движении. Не стихает дискуссия по поводу предлагаемых стандартов, постоянно поступают все новые и новые предложения, тем не менее мы надеемся, что сумеем довести вас до цели, пусть даже она порой напоминает движущуюся мишень. В этой главе мы покажем, как собрать воедино различные части XML-мозаики и создать работающее XML-приложение.

Также мы познакомим наших читателей с некоторыми новыми идеями, чтобы каждому из вас стало ясно, за какими направлениями стоит проследить особо. В этой главе мы обсудим:

- определения типа документа (DTD);
- различия между синтаксически правильными (well-formed) и состоятельными (valid) XML-документами;
- таблицы стилей для форматирования XML-документов;
- анализаторы;
- создание ссылок в XML;
- просмотр XML-файла в браузере;
- новые предложения, касающиеся пространств имен (Namespaces);
- новые предложения для XML-схем, включая XML-данные и описание содержания документа (DCD);
- причины, по которым язык XML должен вам понравится.

Биты и части

Как было отмечено во введении, для практического использования языка XML и его просмотра Web-браузером необходим набор частей, которые в совокупности и составляют XML-мозаику. В данной главе приведен обзор этих элементов, с тем чтобы при их детальном рассмотрении в последующих главах вы представляли, как они соотносятся друг с другом. Желая продемонстрировать, что представляют собой отдельные фрагменты XML-мозаики, вернемся к примеру с библиотекой компакт-дисков, который мы уже приводили. Напомним, что полный XML-файл, описывающий ее фонд, должен был бы состоять из множества размеченных аналогично приводимому примеру записей о компакт-дисках, заключенных в корневые тэги `<cdlib>`.

```
<cdlib>
   <cd>
```

```
    <artist>Arnold Schwarzenegger</artist>
    <title>I'll Be Bach</title>
    <format>album</format>
    <description>Arnie plays Bach's Brandenburg Concertos 1-3 on
    the Hammond Organ</description>
    </cd>

    <cd>
        ...
    </cd>

</cdlib>
```

Определение типа документа

Чтобы разметить XML-документ удобными и понятными тэгами, несущими информацию о своем содержании, необходимо установить правила, понятные для языка разметки, а именно:

- описать, что является разметкой;
- точно определить, что означает тот или иной элемент разметки.

Таким образом, на практике мы должны не только подробно описать каждый из элементов, но и порядок их следования, а также указать, какие именно атрибуты они могут принимать. Спецификация языка XML как раз и определяет данные правила, при этом используется *определение типа документа* (Document Type Definition, DTD). Когда DTD посылается вместе с XML-файлом, пользовательский агент вправе ожидать, что документ соответствует приложенному DTD.

Следует, однако, заметить, что поскольку XML продолжает модифицироваться, необходимо иметь представление и о других способах описания посылаемой информации. Все они объединены под названием *XML-схем* (XML schemas). В первую очередь мы рассмотрим определения типа документа, поскольку они являются частью первоначальной и основной спецификации XML 1.0. В конце этой главы мы также остановимся и на других способах представления данных, которые в настоящее время еще находятся на стадии разработки.

Определение типа документа может содержаться внутри *описания типа документа* (Document Type Declaration) или являться внешним файлом

Если DTD является внешним файлом, то оно подсоединяется к документу следующим способом:

```
<!DOCTYPE cdlib SYSTEM "cdlib.dtd">
```

Язык HTML тоже имеет определение типа документа, но оно присутствует в неявной форме, поскольку содержится внутри наиболее распространенных Web-браузеров и поэтому никем не может быть прочитано. С ним можно познакомиться, посетив сайт консорциума W3C: http://www.w3.org/TR/REC-html140/loose.dtd. В соответствии со стандартом HTML, предполагается, что при создании HTML-документа в код должна быть включена следующая строка:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 //EN">
```

Эта запись сообщает агенту, где находится определение типа документа

для HTML-файла. На практике этой строкой часто пренебрегают, поскольку в большинстве случаев в ней нет особой необходимости. Однако в том случае, когда используются тэги какого-либо конкретного браузера, в какой-то мере не соответствующие формальной спецификации, результаты подобного пренебрежения могут оказаться непредсказуемыми.

Работа над созданием вашего личного языка разметки, в котором используется DTD, не представляет больших трудностей. Рассмотрим, например, определение типа документа, описанное внутри XML-файла, относящегося к библиотеке компакт-дисков. Как видите, это достаточно простое DTD. Деталями мы займемся в следующей главе, однако сразу следует сказать, что этого определения достаточно, чтобы пример с библиотекой компакт-дисков работал.

```
<!DOCTYPE cdlib [
<!ELEMENT cdlib (cd+)

<!ELEMENT cd (artist+, title+, format?, description?)>
<!ELEMENT artist (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT format (#PCDATA)>
<!ELEMENT description(#PCDATA)>

]>
```

Короче говоря, строку `<!DOCTYPE cdlib [` используют для того, чтобы отличить документ с описанием библиотеки компакт-дисков от других типов документов. В этой строке указывается имя, совпадающее с именем корневого элемента документа. Строка `<!ELEMENT>` используется для объявления элементов в формате:

```
<!ELEMENT name (contents)>
```

Здесь `name` (имя) соответствует имени элемента, а `contents` (содержание) описывает, какой тип данных может быть использован и какие элементы разрешается вкладывать в данный элемент. Элемент `<cd>` должен включать в себя элементы `<artist>` и `<title>` (на это указывает символ `+`, который означает «один или более»), в то время как элементы `<format>` и `<description>` не обязательны (на это указывает символ `?`).

Затем определяются элементы, включенные в элемент `<cd>`. Они практически могут представлять собой любой обычный текст без разметки (на исключениях мы остановимся в следующей главе).

Уточнение *Очень легко перепутать определение типа документа (DTD) и описание типа документа (тоже DTD). Чтобы не совершить ошибку, помните, что описание типа документа либо ссылается на определение типа документа, находящееся вне документа, либо содержит его в виде объявлений разметки, как в примере, который мы только что продемонстрировали.*

Скорее всего, с развитием XML число применяемых DTD, которые можно загрузить из сети и использовать для описания файла, будет постоянно расти. Следует ожидать, что скоро появятся стандарты, в соответствии с которыми необходимо

будет размечать определенные типы файлов. Этот процесс уже идет – обратите внимание на такие примеры как *формат определения канала* (Channel Definition Format, CDF) и *химический язык разметки* (Chemical Markup Language, CML), которые мы тоже рассмотрим в этой главе. Но даже если вы не собираетесь заниматься описанием собственных определений типа документа, понимание того, как они работают, очень важно для создания XML-приложений.

Правильные и состоятельные документы

Спецификация XML определяет два типа документов: *синтаксически правильные*, или просто *правильные* (well-formed), и *семантически корректные*, или *состоятельные* (valid). Чтобы быть правильным, документ должен соответствовать следующим трем правилам:

- содержать по крайней мере один элемент;
- содержать корневой элемент, который представляет собой уникальный открывающий и закрывающий тэг, заключающий в себя весь документ;
- все другие элементы, находящиеся внутри документа, должны быть вложены друг в друга без перекрытия.

Итак, если мы вновь обратимся к нашему примеру с библиотекой записей, то вот как должен выглядеть правильный XML-документ:

```
<cdlib>
   <cd>
      <artist>Arnold Schwarzenegger</artist>
      <title>I'll Be Bach</title>
      <format>album</format>
      <description>Arnie plays Bach's Brandenburg Concertos
      1-3 on the Hammond Organ</description>
   </cd>
</cdlib>
```

Здесь налицо один обязательный элемент и несколько других. В документе содержится корневой элемент<cdlib>, который можно сравнить с открывающим <HTML> и закрывающим </HTML> тэгами HTML-документов. Его подэлементы, или дочерние элементы, вложены без перекрытия. Таким образом, документ соответствует критерию правильности.

С другой стороны, состоятельные документы обязаны быть не только правильными, но и иметь такое определение типа документа, которое соответствует тем или иным критериям. Это означает, что в таком документе могут быть использованы только те элементы, которые были объявлены в заданном порядке и имеют разрешенные типы содержания, определенные в DTD.

Концепция состоятельного документа заимствована из языка SGML, однако в SGML документы *обязаны* быть состоятельными, поэтому не существует концепции просто правильных SGML-документов. Конкретные вопросы, связанные с правильными и состоятельными документами, будут рассмотрены в второй главе. Короче говоря, язык XML был создан как раз для пользователей Web, и если правильный документ может быть принят, то есть воспроизведен браузером или

другим агентом, нет необходимости посылать определение типа документа – это просто лишний трафик. Это кажется странным, но тем не менее при некоторых обстоятельствах можно без соответствующего DTD пустить в дело XML-документ, несмотря на то, что агент не знает точно, что означают созданные вами тэги.

Хотя язык XML гораздо проще SGML, фактически он гораздо строже, именно поэтому для XML-файла не обязательно определение типа документа. Так происходит потому, что строгость языка XML позволяет XML-процессору по одному только виду документа типа well-formed сделать заключение о том, какие правила к нему применить. Процессор при этом выстраивает дерево всех вложенных элементов и устанавливает соотношения между всеми его частями. Язык SGML не требует закрывающих тэгов, поэтому невозможно обработать SGML-документ таким же образом, не имея его DTD. Итак, не обязательно тратить пропускную способность канала на пересылку определения типа документа, поскольку правильный XML-документ всегда будет использован или воспроизведен с сохранением функциональных возможностей.

Это не означает, что можно ограничиться беглым просмотром включенного в главу 2 раздела, посвященного определению типа документа. В любом случае желательно, чтобы созданные в XML тэги соответствовали некоторому нормированному DTD. Это нужно для того, чтобы документы, создаваемые и используемые разными людьми, были совместимы друг с другом, то есть подчинялись правилам, установленным в определении типа документа. Соответствие одним и тем же правилам помогает поддерживать структуру сети и вызывает ощущение того, что в рамках одного проекта трудятся сразу много авторов-единомышленников. Определения типа документа также позволяют использовать программу, называемую верифицирующим анализатором (анализатор с проверкой на состоятельность), которая позволяет авторам убедиться в том, что они не нарушают правила DTD. Это будет просто здорово, если подобная программа поможет любому пользователю свести усилия по описанию какого-либо материала до минимума.

Познакомившись с определением типа документа, следует перейти к тому, как оно подсоединяется к XML-документу, и как документ, подсоединенный к таблице стилей, затем форматируется для браузера (рис 1.1). Теперь самое время остановиться на таблицах стилей.

Совет *Не забывайте, что во время подготовки книги к изданию основные браузеры практически не поддерживали язык XML. Несмотря на то, что скорей всего браузер Netscape Communicator 5, а также браузер Microsoft IE5 в какой-то степени начнут все-таки поддерживать XML, существуют способы, с помощью которых можно воспроизводить XML и в нестандартных браузерах. К ним мы вернемся позже, после обсуждения таблиц стилей.*

Таблицы стилей

В отличие от тех методик, идей и предложений, которые обсуждались здесь до сих пор, таблицы стилей ни в коем случае нельзя считать открытием последних

дней. Они стары в той же степени, что и печатное дело. Таблицы стилей – это, по существу, набор правил, устанавливающих, как должен выглядеть документ. Даже во времена ручного набора издатели книг пользовались комплектом письменных инструкций, которые при необходимости подсказывали наборщику, как должна выглядеть страница. Без них рабочий просто не знал, как разметить рукопись. Поскольку наши XML-документы при показе документов в Web-браузере не содержат детальной информации, в каком виде должно выглядеть содержание (при этом обязательно должно быть сохранено разграничение между оформлением и содержанием), нам тоже приходится использовать таблицы стилей, чтобы документы выглядели красно или были удобны для работы.

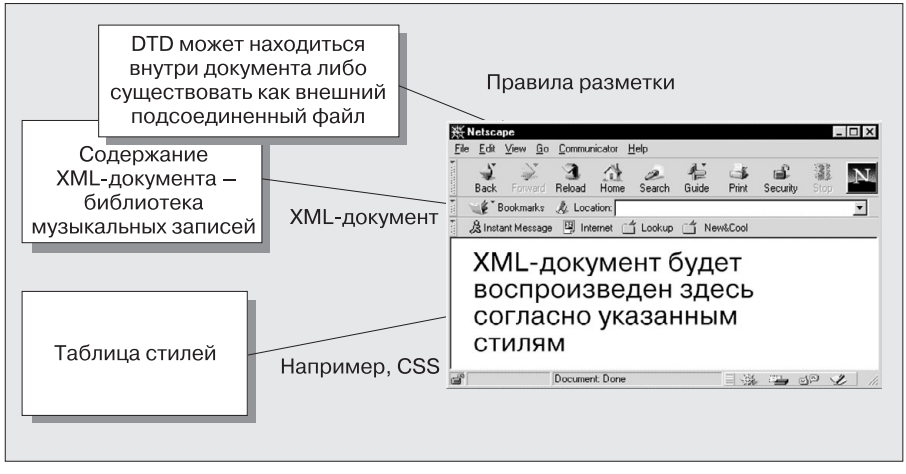


Рис. 1.1. Связь DTD-определений с XML-документом

Преимущества таблиц стилей

Итак, таблицы стилей, являясь необходимым дополнением к DTD и определяя внешний вид документа, имеют ряд преимуществ более общего характера:

- придают документам ясность;
- позволяют уменьшить время загрузки, трафик сети и нагрузку на сервер;
- в случае необходимости могут представить один и тот же источник информации различными способами;
- позволяют изменить вид нескольких файлов изменением только одного файла, который содержит правила о том, как должны выглядеть документы.

Как уже говорилось во введении, с того момента, когда созданием Web-страниц занялись пользователи, не имеющие отношения к науке, чрезмерная озабоченность внешним видом документов привела к созданию огромного числа новых и неудобных стилистических тэгов и атрибутов. В результате размеры файлов стремительно выросли – это явление называют *перегрузкой страниц* (page bloat), – что заметно усложнило чтение кодов и, как следствие, поддержку Web-траниц. Рассмотрим два примера.

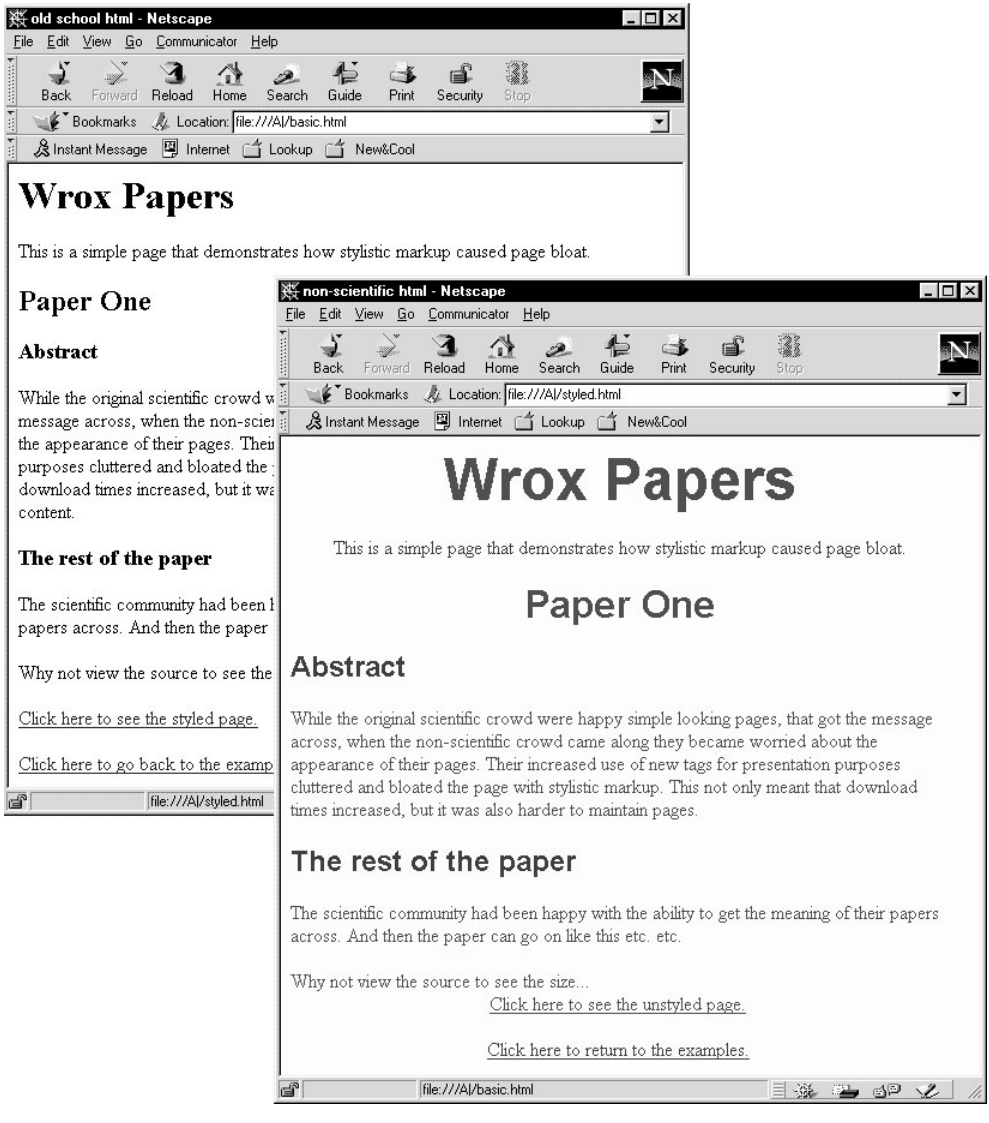


Рис. 1.2. Примеры отображения HTML-документ со стилистической разметкой и без нее

Использование стилистической разметки на второй странице увеличивает размер файла и ухудшает разборчивость кода. Эти страницы можно просмотреть на нашем Web-сайте:

http://webdev.wrox.co.uk/books/1525

Далее показано, как отличаются размеры файлов кода.

```
<HTML>
<HEAD>
<TITLE>old school html</TITLE>
</HEAD>

<BODY>



<H1>Wrox Papers</H1>

This is a simple page that
demonstrates how stylistic
markup caused page bloat.



<H2>Paper One</H2>

<H3>Abstract</H3>


While the original scientific
crowd...
...see their content.

<H3>The rest of the paper</H3>



The scientific community had been
...
```

```
<HTML>
<HEAD>
<TITLE>non-scientific html</TITLE>
</HEAD>

<BODY BGCOLOR="#F5FFFA"
ALINK="#F4A460" LINK="#993200"
VLINK="#556B2F" TEXT="#556B2F">

<H1 ALIGN=CENTER><FONT FACE=ARIAL
SIZE=7>Wrox Papers</FONT></H1>

<DIV ALIGN=CENTER><FONT
COLOR="#993200">This is a simple
page that demonstrates how stylistic
markup caused page
bloat.</FONT></DIV>

<H2 ALIGN=CENTER><FONT FACE=ARIAL
SIZE=6>Paper One</FONT></H2>

<H3><FONT FACE=ARIAL SIZE=5
COLOR="#993200">Abstract</FONT></H3>

While the original scientific
crowd...
...see their content.

<H3><FONT FACE=ARIAL SIZE=5
COLOR="#993200">The rest of the
paper</FONT></H3>

The scientific community had been
...
```

Чтобы поддерживать необходимый внешний облик документов, содержащих более одной страницы, на любом Web-сайте или во внутренней сети (intranet) какой-либо компании или организации необходимо вместе с каждой страницей пересылать браузеру одни и те же правила стилей. Поскольку браузеры помещают данные, полученные из Web-страниц, в кэш, эта многократно повторяемая процедура пересылки одних и тех же правил является пустой тратой времени, ложится тяжким бременем на пропускную способность канала связи, замедляет загрузку документа и затрудняет работу сервера. Таблицы стилей как раз и собирают все правила стилей в один отдельный документ, причем это делается таким образом, чтобы обойтись только одним запросом. Каждая из страниц сайта, обратившись к кэш-памяти браузера, затем может ссылаться на одну и ту же таблицу стилей, которая там хранится. Ясно, что подобная методика существенно сберегает сетевую окружающую среду.

К этому можно добавить, что вместо того, чтобы менять правила стилей на каждой странице, внешний вид всего сайта можно преобразовать изменением лишь одной таблицы стилей, поскольку все стилихранятся в одном файле (рис. 1.3).

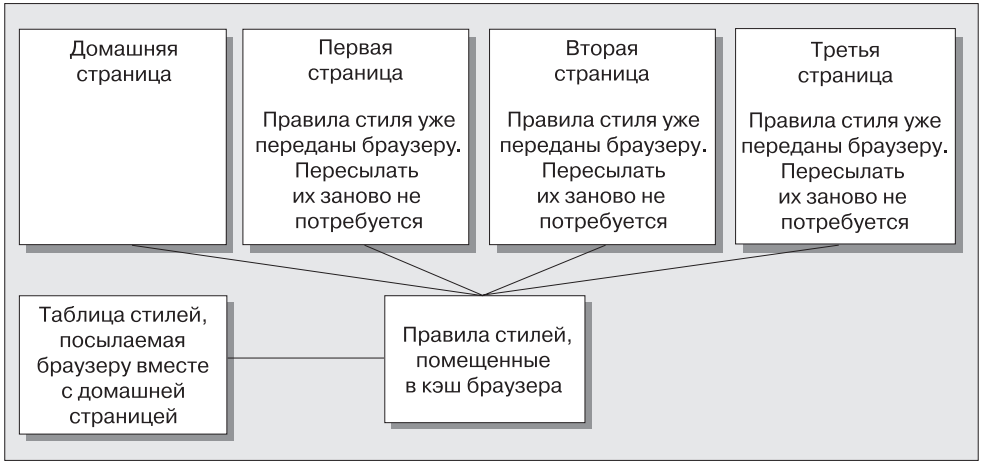


Рис. 1.3. Действие каскадных таблиц стилей

Если вы на нескольких страницах используете одни и те же данные, причем, каждая страница представляет данные различными способами, все что вам надо сделать – это изменить строку, где находятся ссылки на различные таблицы стилей. Примером может служить ситуация, когда в вашем сайте присутствуют различные разделы, в которых одна и та же информация должна быть представлена в особом стиле, соответствующем данной части. Или же, если вам нужно представить информацию для людей с плохим зрением в виде крупного текста, вы можете предложить им ту же самую страницу, но подсоединить к ней стиль с большим размером шрифта.

Каждая из страниц содержит ссылку на таблицу стилей, поэтому агент пользователя знает, откуда ее получить – это похоже на то, как в HTML используются ссылки на рисунок. Существует несколько языков таблиц стилей:

- каскадные таблицы стилей (CSS);
- расширяемый язык таблиц стилей (XSL);
- язык семантики и спецификаций стилей документа (DSSSL);
- XS, также известный как DSSSL-0;

Двумя из них – CSS и XSL – стоит поинтересоваться особо. В свое время DSSSL являлся официальным языком стилей для SGML – это очень мощное и в не меньшей степени сложное средство описания таблицы стилей. По этой причине он и не получил широкого признания среди программистов. Язык XS или DSSSL-0 создавался как упрощенная форма DSSSL, но на практике он все еще оставался слишком сложным, так что в Internet не прижился.

Давайте кратко остановимся на первых двух языках.

Каскадные таблицы стилей

Спецификация каскадных таблиц стилей первого уровня (CSS1) была выпущена консорциумом W3C в конце 1996 года. В некоторой степени этот язык поддерживается как браузером Communicator 4, так и браузером IE4, при этом оба основных производителя браузеров клятвенно обещают полностью поддерживать CSS1 в новых версиях Communicator 5 и IE5.

Совет *Первый уровень спецификации CSS (CSS1) вскоре будет заменен вторым уровнем CSS (CSS2). Рекомендации к выпуску были даны в мае 1998 года. Однако в настоящее время даже первый уровень внедрен еще не полностью, так что придется подождать, пока производители браузеров обеспечат полную поддержку второму уровню CSS.*

Каскадные таблицы стилей уже активно проникают в Web. Такое название появилось вследствие того, что при наличии нескольких таблиц стилей из них обычно формируется каскад, то есть иерархия со свойствами, взятыми из всех таблиц. Любые конфликты разрешаются в соответствии с некоторым набором правил. Каскадные таблицы стилей конструируются достаточно легко, и коль скоро они созданы, не существует предела числу страниц, которые могут их использовать. Чтобы присоединить к документу CSS, достаточно указать в HTML-документе одну лишь строку:

```
<LINK REL="stylesheet" TYPE="text/css" HREF="example.css">
```

В настоящее время консорциум W3C изыскивает способ, с помощью которого XML-документы будут присоединяться к каскадным таблицам стилей. Сообщение по этому поводу находится по адресу:

http://www.w3.org/TR/NOTE-xml-stylesheet

Один из самых распространенных сейчас способов включения CSS выглядит следующим образом:

```
<?xml-stylesheet href="cdlib.css" type="text/css"?>
```

Вот пример каскадной таблицы стилей, который может быть использован совместно с рассмотренной нами библиотекой компакт-дисков:

```
artist {
   display: block;
   font-family: Arial, Helvetica;
   font-weight: bold;
   font-size: 20pt;
   color: #9370db;
   text-align: center;
   }

title {
   display: block;
   font-family: Arial, Helvetica;
   font-size: 20pt;
```

```
    color: #c71585;
    }

format {
    display: block;
    font-family: Arial, Helvetica;
    font-size: 16pt;
    color: #9370db;
    }

description {
    display: block;
    font-family: Arial, Helvetica;
    font-style: italic;
    font-size: 16pt;
    color: #FF1010;
    }
```

При просмотре в браузере наша библиотека компакт-дисков с данной таблицей стилей теперь будет выглядеть так, как показано на рис. 1.4. Изображение было получено с использованием бета-версии браузера IE5, и если вы им пользуетесь, можно попробовать выполнить ту же операцию, используя материал нашего Web-сайта:

http://webdev.wrox.co.uk/books/1525

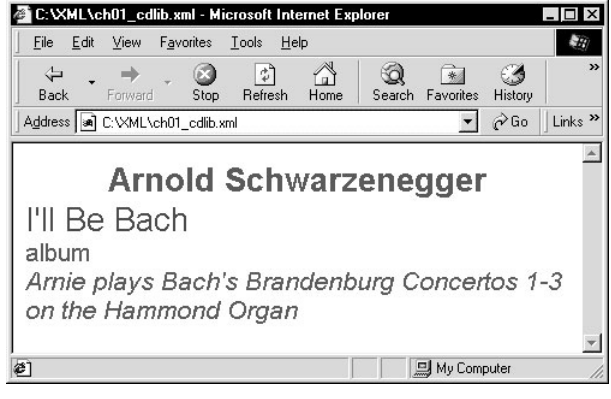


Рис. 1.4. Отображение документа с применением каскадных таблиц стилей

Использование каскадных таблиц стилей дает много преимуществ. Подробнее о таких таблицах стилей можно прочитать в главе 7. Следует иметь в виду, что язык CSS использует фиксированный набор типов разметки, так что здесь вам не удастся создать собственные тэги – CSS не допускает расширения. Для этого существует другое, более мощное средство – язык каскадных таблиц стилей XSL.

Расширяемый язык таблиц стилей XSL

Многие разработчики, пишущие на языке XML, для отображения XML-документов несложной структуры будут вполне удовлетворены функциональными