Лабораторная работа Отображение XML-документов с использованием XSL-таблиц стилей

XSL (Extensible Stylesheet Language) – расширяемый язык таблиц стилей. Подобно таблице каскадных стилей (CSS в HTML), XSL-таблица стилей связывается с XML-документом и сообщает браузеру, как отображать данные XML. Это позволяет открывать XML-документ непосредственно в браузере без посредничества HTML-страницы.

XSL-таблица стилей – более мощный и гибкий инструмент для отображения XML-документов, чем CSS-таблица. Используя XSL-таблицы стилей можно не только задать формат для каждого элемента XML, как при использовании CSS-таблицы, но и обеспечить средства контроля над выводимыми данными. XSL позволяет выбрать те данные XML, которые необходимо отобразить, представить эти данные в любом порядке, свободно модифицировать или добавлять информацию. XSL предоставляет доступ ко всем компонентам XML (элементам, атрибутам, комментариям и инструкциям по обработке). Эта таблица позволяет легко сортировать и фильтровать данные XML, дает возможность включать в таблицу стиля сценарии и предоставляет набор полезных методов, которые можно использовать при обработке информации.

Базовая форма XSL-таблицы стилей избирательно преобразует XML-документ в HTML-страницу, воспринимаемую и отображаемую затем браузером. Получая доступ к богатому арсеналу HTML, к методам преобразования XSL добавляются новые возможности по форматированию и обработке данных.

Однако XSL-таблицы являются более сложными для понимания, чем CSS-таблицы. Работа с ними требует знания языка HTML.

Официальные спецификациям XSL можно посмотреть на Web-страницах, предоставленных консорциумом World Wide Web Consortium (W3C):

– "Extensible Stylesheet Language (XSL) Version 1.0" http://www.w3.org/TR/WD-xsl

– "XSL Transformations (XSLT) Version 1.0" http://www.w3.org/TR/WD-xslt.

Подробнее узнать о поддержке XSL в Internet Explorer можно на соответствующих Web-страницах, предоставленных Microsoft Developer Network (MSDN): "XSL Developer's Guide".

Основы использования XSL-таблиц стилей

Существуют два основных шага для отображения XML-документа при использовании XSL-таблицы стилей:

1. **Создание файла XSL-таблицы стилей**. XSL является приложением XML, т. е. XSL-таблица представляет собой корректно сформированный XML-документ, который отвечает правилам XSL. Подобно любому XML-документу, XSL-таблица стилей содержит простой текст, и можно создать ее с помощью любого текстового редактора.

2. **Связывание XSL-таблицы стилей с XML-документом**. Можно связать XSL-таблицу стилей с XML-документом, включив в документ инструкцию по обработке xml-stylesheet, которая имеет следующую обобщенную форму записи:

<?xml-stylesheet type="text/xsl" href=XSLFilePath?>

Здесь XSLFilePath – заключенный в кавычки URL, указывающий местонахождение файла таблицы стилей. Можно использовать полный URL, например:

<?xml-stylesheet type="text/xsl" href="http:/www.my_site.com/Tab1.xsl"?>

Чаще используют неполный URL, который задает местонахождение относительно месторасположения XML-документа, содержащего инструкцию по обработке xml-stylesheet, например:

<?xml-stylesheet type="text/xsl" href="1.xsl"?>

Относительный URL встречается чаще, поскольку обычно файл таблицы стилей хранится в той же папке, где хранится XML-документ, либо в одной из вложенных в нее папок.

Можно связать XSL-таблицу стилей с использованием полного URL, таблица стилей при этом должна размещаться на том же домене, что и XML-документ, с которым ее связывают. Например, если домен http://docs.soft.com/ содержит XML-документ, то и XSL-таблица стилей должна размещаться на том же домене.

Обычно инструкция по обработке xml-stylesheet добавляется в пролог XML-документа вслед за объявлением XML.

Если XSL-таблица стилей связана с XML-документом, то можно открыть этот документ непосредственно в Internet Explorer или в другом браузере, и браузер отобразит XML-документ с использованием инструкций по преобразованию, содержащихся в таблице стилей. В отличие от таблиц каскадных стилей, если с XML-документом связывается более одной XSL-таблицы стилей, браузер использует первую таблицу и игнорирует все остальные. Если с XML-документом связаны и CSS-таблица и XSL-таблица стилей, браузер использует только XSL-таблицу стилей.

Если XML-документ не связан ни с CSS-таблицей, ни с XSL-таблицей стилей, браузер отобразит документ с помощью встроенной XSL-таблицы, которая используется по умолчанию. Эта таблица стилей отображает исходный XML-текст в виде дерева с возможностью свертывания/развертывания уровней.

Использование одного шаблона XSL

В отличие от CSS, содержащей правила, XSL-таблица стилей включает один или несколько шаблонов, каждый из которых содержит информацию для отображения в определенной ветви элементов в XML-документе.

Создайте xml-документ в любом текстовом редакторе (Блокнот, Notepad++), сохранив его с расширением ***xml***, предварительно выбрав тип файла – **Все файлы** или **xml**, с информацией о книге: название книги, фамилия, имя автора книги, тип обложки, количество страниц и цена, сохраните его, например, **книга.xml**:

Пример 1. книга.xml

```
<?xml version="1.0" encoding="windows-1251" ?>
- <книга>
    <название>Web-программирование на Java и JavaScript</название>
  - <автор>
      <имя>Андрей</имя>
      <фамилия>Гарнаев</фамилия>
    </автор>
    <обложка>мягкая</обложка>
    <страницы>1040</страницы>
    <цена>413 рублей</цена>
  </книга>
```

Рисунок 1 – Отображение xml-документа книга.xml в Internet Explorer

Откройте созданный файл любым браузером и проверьте корректность отображения данных. На рисунке 1 показано отображение содержимого файла книга.xml в Internet Explorer.

Создайте для xml-документа книга.xml стилевой файл в любом текстовом редакторе с отображением всей информации xml-документа и подписями. Для подписей примените курсив (при помощи тега span и его атрибута style: <SPAN STYLE="font-style:italic">), например, как на рисунке 2.

Обратите внимание, что во втором теге <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"> также необходимо указать атрибут version="1.0", без него стилевой файл будет просматриваться без ошибок (рисунок 2), но при подгрузке стилевого файла в xml-документ будет выдаваться ошибка об отсутствии данного атрибута.

Пример 2. книга.xsl

```
<?xml version="1.0" encoding="windows-1251" ?>
- <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  - <xsl:template match="/">
      <H2>Описание книги</H2>
      <SPAN STYLE="font-style:italic">Автор:</SPAN>
      <xsl:value-of select="книга/автор" />
      <BR />
      <SPAN STYLE="font-style:italic">Название:</SPAN>
      <xsl:value-of select="книга/название" />
      <BR />
      <SPAN STYLE="font-style:italic">Цена:</SPAN>
      <xsl:value-of select="книга/цена" />
      <BR />
      <SPAN STYLE="font-style:italic">Тип обложки:</SPAN>
      <xsl:value-of select="книга/обложка" />
      <BR />
      <SPAN STYLE="font-style:italic">Количество страниц:</SPAN>
      <xsl:value-of select="книга/страницы" />
    </xsl:template>
  </xsl:stylesheet>
```

Рисунок 2 – Отображение стилевого файла книга.xsl с отсутствующим атрибутом version во втором теге

В тегах <***xsl:value-of select***="…"/> указан «полный путь» к дочерним элементам, обращение начинается с корневого элемента «книга», через «/» указывается дочерний элемент, например, автор: <xsl:value-of select="книга/автор"/>

Для подключения стилевого файла добавьте в xml-документ вторым тегом **<?xml-stylesheet type="text/xsl" href="книга.xsl"?>**

На рисунке 3 показано отображение XML-документа в Internet Explorer в соответствии с инструкциями из таблицы стилей.

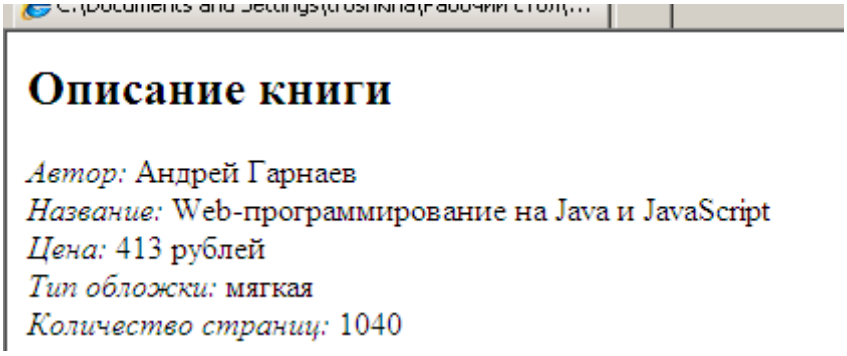

Рисунок 3 – Отображение xml-документа с применением стилевого файла

Каждая XSL-таблица стилей должна иметь элемент Документ, представленный ниже (элемент Документ или корневой элемент – XML-элемент верхнего уровня, который содержит все остальные элементы):

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <!-- один или несколько элементов шаблона …-->
</xsl:stylesheet>
```

Элемент Документ xsl:stylesheet служит не только хранилищем других элементов, но также идентифицирует документ как XSL-таблицу стилей. Этот элемент является одним из XSL-элементов специального назначения, используемых в таблице стилей. Все XSL-элементы принадлежат пространству имен xsl – т. е. имя каждого XSL-элемента начинается с префикса xsl:, обозначающего пространство имен. Это пространство имен определяется в начальном теге элемента xsl:stylesheet, например, следующим образом:

```
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
```

Это определение позволяет использовать пространство имен внутри элементов таблицы стилей.

Элемент Документ xsl:stylesheet XSL-таблицы стилей должен содержать один или несколько шаблонов элементов. Элемент Документ из примера 2 содержит только один шаблон, который имеет следующую форму:

```
<xsl:template match="/">
  <!-- дочерние элементы … -->
</xsl:template>
```

Браузер использует шаблон для отображения определенной ветви элементов в иерархии XML-документа, с которым связана таблица стилей. Атрибут match шаблона указывает на определенную ветвь (атрибут match аналогичен селектору в правиле CSS). Значение атрибута match носит название образца (pattern). Образец в данном примере ("/") представляет корневой элемент всего XML-документа. Этот шаблон, таким образом, содержит инструкции для отображения всего XML-документа.

Каждая XSL-таблица стилей должна содержать один и только один шаблон с атрибутом match, который имеет значение "/". Также можно включить один или несколько дополнительных шаблонов с инструкциями для отображения определенных подчиненных ветвей в структуре XML-документа; каждая из них должна иметь образец, отвечающий определенной ветви.

Корневой образец ("/") не представляет элемент Документ (или корневой элемент) XML-документа. Он представляет весь документ, для которого элемент Документ является дочерним.

Пример полного описания шаблона из рассматриваемой таблицы стилей:

```
<xsl:template match="/">
  <H2>Описание книги</H2>
  <SPAN STYLE="font-style:italic">Автор: </SPAN>
  <xsl:value-of select="книга/автор"/><BR/>
  <SPAN STYLE="font-style:italic">Название: </SPAN>
  <xsl:value-of select=" книга /название"/><BR/>
  <SPAN STYLE="font-style:italic">Цена: </SPAN>
  <xsl:value-of select=" книга /цена"/><BR/>
  <SPAN STYLE="font-style:italic">Тип обложки: </SPAN>
  <xsl:value-of select=" книга /обложка"/><BR/>
  <SPAN STYLE="font-style:italic">Количество страниц: </SPAN>
  <xsl:value-of select=" книга /страницы"/>
</xsl:template>
```

Шаблон содержит два вида XML-элементов:

- **XML-элементы, представляющие HTML-разметку**. Примерами подобного вида XML-элемента из рассматриваемой таблицы стилей являются:

```
<H2>Описание книги</H2>
```

который отображает заголовок второго уровня,

```
<SPAN STYLE="font-style:italic">Автор: </SPAN>
```

который отображает блок текста, набранного курсивом (Автор:), и

```
<BR/>
```

который создает пустую строку.

Все эти XML-элементы являются корректно сформированными и представляют стандартные HTML-элементы. Браузер просто копирует каждый HTML-элемент непосредственно на выход HTML, который воспринимает и отображает их.

Каждый из элементов, представляющих HTML-разметку, должен быть корректно сформированным XML-элементом, а также стандартным HTML-элементом (XSL-таблица стилей является XML-документом). Следовательно, нельзя использовать HTML-конструкции, которые не являются корректно сформированным XML, такие, как элементы, состоящие только из начального тега. Например, чтобы задать элемент перевода строки в HTML, нужно использовать корректно сформированный тег пустого XML-элемента,
.

- **XSL-элементы**. Примерами XSL-элементов из рассматриваемой таблицы стилей являются элементы xsl:value-of, например:

```
<xsl:value-of select="книга/автор"/>
```

Браузер отличает XSL-элемент от элемента, представляющего HTML, поскольку первый имеет в качестве префикса описание пространства имен xsl:. XSL-элементы в шаблоне не копируются на выход HTML. Они лишь содержат инструкции по выбору и модификации данных XML, либо используются для выполнения других задач.

XSL-элемент ***value-of*** добавляет текстовое содержимое определенного XML-элемента, а также любых его дочерних элементов, которые он имеет, в выходной модуль HTML, который воспринимается и отображается браузером. Указывается определенный XML-элемент заданием образца, который присваивается атрибуту ***select*** XSL-элемента ***value-of***. В рассмотренном выше примере элемента ***value-of*** атрибуту ***select*** присвоен образец ***"книга/автор"***, что приводит к выводу текстового содержимого элемента ***автор*** XML-документа. Текстовое содержимое элемента ***автор*** состоит из символьных данных, принадлежащих двум его дочерним элементам, ***фамилия*** и ***имя***.

Обратите внимание, что XML-элемент в образце задается с помощью оператора пути (в данном случае ***книга/автор***), который определяет местонахождение элемента в иерархии XML-документа. (Оператор пути аналогичен пути к файлу, который операционная система использует для указания местонахождения файла или папки).

Главный момент, на который здесь следует обратить внимание, состоит в том, что оператор пути в значении атрибута ***select*** относится к текущему элементу. Каждый контекст внутри XSL-таблицы стилей относится к текущему элементу. Поскольку рассматриваемый пример шаблона относится к корневому элементу всего документа (посредством установки атрибута match="/"), текущим «элементом» для данного шаблона является корневой элемент документа. (В данном случае текущий элемент не обладает соответствующим литералом, а является родителем элемента Документ). Таким образом, внутри этого шаблона оператор пути книга/автор указывает на элемент автор, вложенный в элемент книга, вложенный в корневой элемент документа. (Оператор пути в значении атрибута ***select*** аналогичен неполному пути к файлу, задающему местонахождение файла относительно текущей рабочей папки).

Если атрибут ***select*** для XSL-элемента ***value-of*** отсутствует, элемент будет осуществлять вывод текстового содержимого плюс текстовое содержимое всех дочерних элементов текущего элемента. (В нашем примере текущим является корневой элемент, пропуск атрибута ***select*** приведёт к выводу всех символьных данных XML-документа).

Целью представленного в рассматриваемом примере шаблона элементов является отображение текста названия для каждого из дочерних XML-элементов в документе (***автор***, ***название***, ***цена***, ***обложка*** и ***страницы***) плюс текстового содержимого каждого элемента. Обратите внимание, что порядок элементов ***value-of*** в шаблоне определяет порядок, в котором браузер отображает эти элементы. Из этой простой таблицы стилей видно,

что XSL-таблица стилей является гораздо более гибкой, чем CSS, которая всегда отображает элементы в том порядке, в котором они следуют в документе.

XSL-таблица стилей сообщает браузеру, как отобразить XML-документ путем избирательного преобразования XML-элементов в блок HTML-разметки, который воспринимается и отображается браузером аналогично разметке, содержащейся на HTML-странице. Не нужно включать в XSL-шаблон элементы, представляющие элементы HTML или ***body***, которые являются стандартными составными частями HTML-страницы, поскольку браузер сам эффективно их формирует.

На рисунке 4 показано как браузер генерирует первую часть блока HTML-разметки для документа и таблицы стилей из примеров 1и 2.

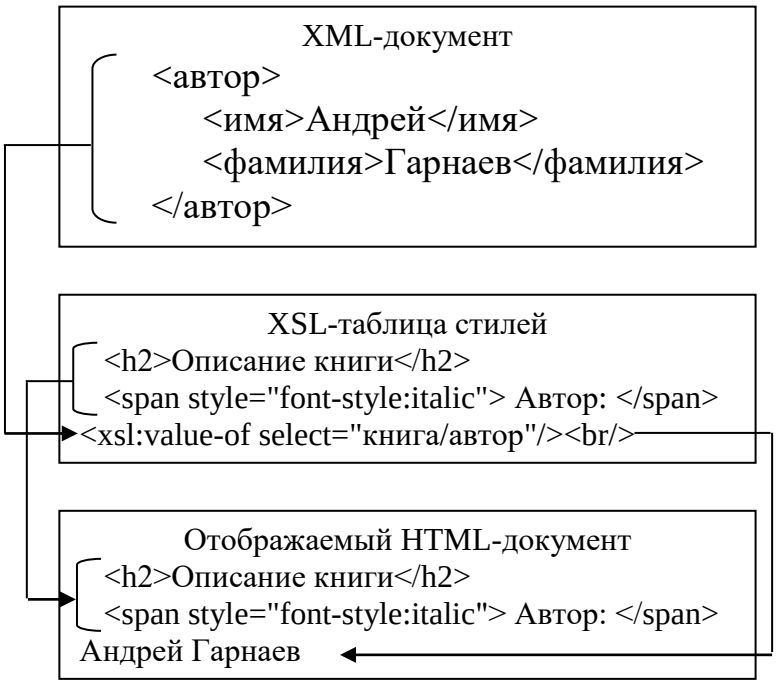

Рисунок 4 – Генерация первой части блока HTML-разметки для документа и таблицы стилей из примеров 1и 2

XML-документ
<автор>
<имя>Андрей</имя>
<фамилия>Гарнаев</фамилия>
</автор>

XSL-таблица стилей

```
<H2>Описание книги</H2>
<SPAN STYLE="font-style:italic">Автор: </SPAN>
<xsl:value-of select="книга/автор"/><BR/>
```

Отображаемый HTML

```
<H2>Описание книги</H2>
<SPAN STYLE="font-style:italic">Автор: </SPAN>
Андрей Гарнаев
```

С официальной спецификацией HTML можно познакомиться на Web-сайте, предоставленном W3C: http://www.w3.org/TR/REC-html40/.

Отображение переменного числа элементов

В примере 1 XML-документ содержал только один элемент ***книга***. В случае, если документ содержит несколько элементов ***книга***, методика, использованная в примере 2, способна отобразить только один из элементов. Например, XML-документ содержит следующий элемент Документ:

Пример 3. книги.xml

```
<?xml version="1.0" encoding="windows-1251" ?>
- <Список_книг>
  - <книга>
      <название>Введение в SQL</название>
    - <автор>
        <имя>Мартин</имя>
        <фамилия>Грубер</фамилия>
      </автор>
      <обложка>твердая</обложка>
      <страницы>378</страницы>
      <цена>618 рубля</цена>
    </книга>
  - <книга>
      <название>Microsoft SQL Server 2012. Основы T-SQL</название>
    - <автор>
        <имя>Ицик</имя>
        <фамилия>Бен-Ган</фамилия>
      </автор>
      <обложка>мягкая</обложка>
      <страницы>400</страницы>
      <цена>743 рубля</цена>
    </книга>
  - <книга>
      <название>SQL. Справочник</название>
    - <автор>
        <имя>Кевин Е.</имя>
        <фамилия>Кляйн</фамилия>
      </автор>
      <обложка>мягкая</обложка>
      <страницы>656</страницы>
      <цена>1259 рублей</цена>
    </книга>
</Список_книг>
```

Предположим, что таблица стилей, используемая для отображения этого документа, содержит следующий шаблон:

Пример 4. книги.xsl

```
<?xml version="1.0" encoding="windows-1251" ?>
- <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/ Transform">
  - <xsl:template match="/">
      <H2>Описание книг</H2>
      <SPAN STYLE="font-style:italic">Автор:</SPAN>
      <xsl:value-of select="Список_книг/книга/автор" />
      <BR />
      <SPAN STYLE="font-style:italic">Название:</SPAN>
      <xsl:value-of select="Список_книг/книга/название" />
      <BR />
      <SPAN STYLE="font-style:italic">Цена:</SPAN>
      <xsl:value-of select="Список_книг/книга/цена" />
      <BR />
      <SPAN STYLE="font-style:italic">Обложка:</SPAN>
      <xsl:value-of select="Список_книг/книга/обложка" />
      <BR />
      <SPAN STYLE="font-style:italic">Количество страниц:</SPAN>
      <xsl:value-of select="Список_книг/книга/страницы" />
    </xsl:template>
  </xsl:stylesheet>
```

Этот шаблон использует методику, описанную в примере 2. Обратите внимание, что образец присваиваемых каждому атрибуту ***select*** начинается с указания элемента Документ, в данном случае Список_книг (например, "Список_книг/книга/автор").

Каждый образец, однако, соответствует трем различным элементам. Например, "Список_книг/книга/автор" соответствует элементу ***автор*** для всех трех элементов ***книга***. В подобной ситуации браузер использует только первый из соответствующих элементов. Таким образом, таблица стилей отобразит содержимое только первого элемента ***книга***, как показано на рисунке 5.

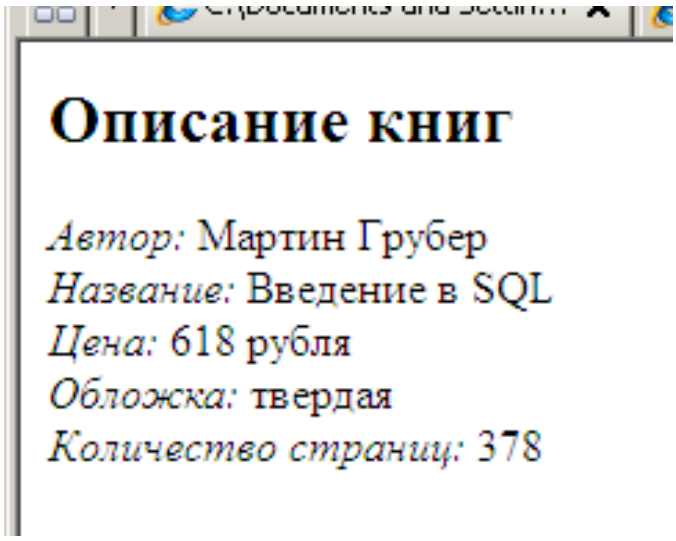

Рисунок 5 – Отображение только первого элемента

Чтобы отобразить все отвечающие образцу элементы, следует использовать XSL-элемент ***for-each***, который вызывает повторный вывод для каждого из содержащихся в XML-файле элементов. XSL-таблица стилей,

представленная в примере 5, демонстрирует данную методику. Эта таблица стилей связана с XML-документом из примера 3:

Пример 5. Модифицированный стилевой файл

```
<?xml version="1.0" encoding="windows-1251" ?>
- <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/ Transform">
  - <xsl:template match="/">
      <H2>Описание книг</H2>
    - <xsl:for-each select="Список_книг/книга">
        <SPAN STYLE="font-style:italic">Название:</SPAN>
        <xsl:value-of select="название" />
        <BR />
        <SPAN STYLE="font-style:italic">Автор:</SPAN>
        <xsl:value-of select="автор" />
        <BR />
        <SPAN STYLE="font-style:italic">Обложка:</SPAN>
        <xsl:value-of select="обложка" />
        <BR />
        <SPAN STYLE="font-style:italic">Количество страниц:</SPAN>
        <xsl:value-of select="страницы" />
        <BR />
        <SPAN STYLE="font-style:italic">Цена:</SPAN>
        <xsl:value-of select="цена" />
        <P />
      </xsl:for-each>
    </xsl:template>
  </xsl:stylesheet>
```

После подключения этого стилевого файла в XML-документ содержимое уже будет отображаться так:

Описание книг

Название: Введение в SQL
Автор: Мартин Грубер
Обложка: твердая
Количество страниц: 378
Цена: 618 рубля

Название: Microsoft SQL Server 2012. Основы T-SQL
Автор: Ицик Бен-Ган
Обложка: мягкая
Количество страниц: 400
Цена: 743 рубля

Название: SQL. Справочник
Автор: Кевин Е. Кляйн
Обложка: мягкая
Количество страниц: 656
Цена: 1259 рублей

Шаблон в таблице стилей из примера 5 содержит следующий элемент for-each:

```
<xsl:for-each select="Список_книг/книга">
      <SPAN STYLE="font-style:italic">Название: </SPAN>
      <xsl:value-of select="название"/><BR />
      <SPAN STYLE="font-style:italic">Автор: </SPAN>
      <xsl:value-of select="автор"/><BR />
      <SPAN STYLE="font-style:italic">Обложка: </SPAN>
      <xsl:value-of select="обложка"/><BR />
      <SPAN STYLE="font-style:italic">Количество страниц: </SPAN>
      <xsl:value-of select="страницы"/><BR />
      <SPAN STYLE="font-style:italic">Цена: </SPAN>
      <xsl:value-of select="цена"/><P />
    </xsl:for-each>
```

Элемент for-each выполняет две основные задачи:

- осуществляет вывод блока элементов, содержащихся внутри элемента for-each, повторяя его для каждого XML-элемента в документе, отвечающего образцу, присвоенному атрибуту select элемента for-each. В данном примере цикл выполняется по одному разу для каждого элемента **книга**, найденного в элементе Документ с именем **Список_книг**. Образец, присваиваемый атрибуту select элемента for-each, работает точно так же, как образец, присваиваемый атрибуту select элемента value-of;
- внутри элемента for-each задает текущий элемент, устанавливаемый атрибутом select элемента for-each (**/Список_книг/книга** в нашем примере указывает на элемент **книга** внутри элемента **Список_книг**, входящего в корневой элемент документа) следующим образом:

```
<xsl:stylesheet xmlns:xsl=http://www.w3.org/1999/XSL/Transform>
 <xsl:template match="/">
   <!-- Здесь текущим является корневой "элемент"документа, "/". -->
 <xsl:for-each select="Список_книг/книга">
 <!-- Здесь текущим является элемент / Список_книг/книга. -->
 </xsl:for-each>
 </xsl:template>
</xsl:stylesheet>
```

Аналогично, внутри элемента for-each каждый дочерний элемент может быть выбран путем задания образца, содержащего только имя элемента, например:

```
<xsl:value-of select="название"/>
```

В результате выводятся данные из всех элементов **книга**, найденных в документе, независимо от того, сколько этих элементов содержит документ.

Использование нескольких шаблонов

Другой способ отображения повторяющихся XML-элементов состоит в создании отдельного шаблона для каждого элемента с последующим вызовом этого шаблона с использованием XSL-элемента ***apply-templates***. Пример использования подобной методики приведен в XSL-таблице стилей, представленной в примере 6.

Пример 6. Отображение элементов при помощи отдельного шаблона

```
<?xml version="1.0" encoding="windows-1251" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <H2>Описание книг</H2>
    <xsl:apply-templates select="Список_книг" />
  </xsl:template>
  <xsl:template match="книга">
    <SPAN STYLE="font-style:italic">Название:</SPAN>
    <xsl:value-of select="название" />
    <BR />
    <SPAN STYLE="font-style:italic">Автор:</SPAN>
    <xsl:value-of select="автор" />
    <BR />
    <SPAN STYLE="font-style:italic">Обложка:</SPAN>
    <xsl:value-of select="обложка" />
    <BR />
    <SPAN STYLE="font-style:italic">Количество страниц:</SPAN>
    <xsl:value-of select="страницы" />
    <BR />
    <SPAN STYLE="font-style:italic">Цена:</SPAN>
    <xsl:value-of select="цена" />
    <P />
  </xsl:template>
</xsl:stylesheet>
```

Эта таблица стилей предназначена для связывания с XML-документом из примера 3; установить эту связь можно путем модификации инструкции xml-stylesheet в документе следующим образом:

<?xml-stylesheet type="text/xsl" href="имя_стилевого_файла.xsl"?>

Рассматриваемая в примере таблица стилей содержит два шаблона. Один шаблон содержит инструкции для отображения всего документа (путем установки match="/", указывающей на корневую часть документа). Все XSL-таблицы стилей требуют наличия такого шаблона. Другой шаблон содержит инструкции для отображения элемента ***книга*** (шаблон с установкой match="книга"). Сначала браузер обрабатывает шаблон, соответствующий корневой части элемента:

```
<xsl:template match="/">
  <H2>Описание книг</H2>
  <xsl:apply-templates select="Список_книг/книга" />
</xsl:template>
```

XSL-элемент apply-templates сообщает браузеру, что для каждого элемента ***книга*** внутри корневого элемента ***Список_книг*** он должен обрабатывать шаблон, отвечающий элементу ***книга*** – т. е. шаблон, для атрибута match которого установлено значение "книга". Таблица стилей включает следующий шаблон, отвечающий элементу ***книга***:

```
<xsl:template match="книга">
  <SPAN STYLE="font-style:italic">Название: </SPAN>
  <xsl:value-of select="название"/><BR/>
  <SPAN STYLE="font-style:italic">автор: </SPAN>
  <xsl:value-of select="автор"/><BR/>
  <SPAN STYLE="font-style:italic">Обложка: </SPAN>
  <xsl:value-of select=" обложка"/><BR/>
  <SPAN STYLE="font-style:italic">  Количество страниц: </SPAN>
  <xsl:value-of select=" страницы"/><BR/>
  <SPAN STYLE="font-style:italic">Цена: </SPAN>
  <xsl:value-of select="цена"/><P/>
</xsl:template>
```

Поскольку этот шаблон отвечает элементу ***книга***, элемент ***книга*** является текущим элементом в контексте шаблона. В связи с этим доступ к дочерним элементам ***книга*** осуществляется посредством образца, содержащего только имя элемента, как в примере:

```
<xsl:value-of select="название"/>
```

Если не указывать атрибут ***select*** для элемента ***apply-templates***, браузер обрабатывает соответствующий шаблон (если он имеется) для каждого дочернего элемента текущего элемента. В рассматриваемом примере элемента ***apply-templates*** единственным дочерним элементом для текущего элемента (корневая часть документа) является элемент ***Список_книг***, который не имеет соответствующего шаблона. Таким образом, если опустить атрибут ***select***, никакие данные не будут выведены.

Браузер обрабатывает шаблон ***книга*** один раз для каждого элемента ***книга***, отображая всю информацию о книгах, имеющуюся в документе:

Описание книг

Название: Введение в SQL
Автор: Мартин Грубер
Обложка: твердая
Количество страниц: 378
Цена: 618 рубля

Название: Microsoft SQL Server 2012. Основы T-SQL
Автор: Ицик Бен-Ган
Обложка: мягкая
Количество страниц: 400
Цена: 743 рубля

Название: SQL. Справочник
Автор: Кевин Е. Кляйн
Обложка: мягкая
Количество страниц: 656
Цена: 1259 рублей

Фильтрация данных XML

Значение, которое присваивается атрибутам ***match*** или ***select***, представляет собой образец, соответствующий одному или нескольким элементам в XML-документе (атрибут ***match*** используется для элемента ***template***, а атрибут ***select*** – для элементов ***value-of***, ***for-each*** и ***apply-templates***). Образцы, с которыми мы имели дело до сих пор, содержали только оператор пути, который задавал имя элемента и, возможно, одного или нескольких вложенных элементов. Можно ограничить количество элементов, отвечающих шаблону, введя фильтр – выражение, заключенное в квадратные скобки ([]) и следующее непосредственно за оператором пути. Например, образец, присвоенный следующему атрибуту ***match***, указывает, что соответствующий элемент должен носить имя ***книга***, и кроме того (это определяется фильтром), должен иметь дочерний элемент ***обложка***, который содержит текст "***мягкая***":

<xsl:template match="книга[обложка='мягкая']">

Если в фильтр включено только имя элемента, то соответствующий элемент должен иметь дочерний элемент с указанным именем. Например, следующий образец отвечает любому элементу ***название***, имеющему

дочерний элемент с именем ***стол***, независимо от содержимого элемента ***стол***:

match="название[стол]"

Следующий образец отвечает любому элементу ***товар***, имеющему дочерний элемент ***цвет***, содержащий текст "***синий***":

match="товар[цвет=синий]"

А следующий образец, наоборот, отвечает любому элементу ***товар***, имеющему дочерний элемент ***цвет***, который не содержит текст "***синий***":

select="товар[цвет!='синий']"

Если элемент имеет более одного дочернего элемента с именем, указанным в условии фильтрации, оператор сравнения применяется только к первому дочернему элементу. Например, если элемент ***товар*** имеет два дочерних элемента ***цвет***, образец "товар[цвет=синий]" будет отвечать элементу, только если первый элемент ***цвет*** содержит слово "синий".

Сортировка данных XML

Для упорядочивания элементов можно использовать атрибут ***order-by***, в котором браузер обрабатывает элементы, осуществляя сортировку данных XML. Атрибуту ***order-by*** можно назначать один или несколько образцов, разделяя их точкой с запятой. Браузер будет сортировать элементы с использованием образцов в том порядке, в котором они перечислены. Для указания направления сортировки (по возрастанию или по убыванию) следует предварить образец префиксом + или – .

Например, атрибут ***order-by***, установленный для следующего элемента ***for-each***, предписывает браузеру сортировать элементы ***книга*** по фамилиям авторов в порядке возрастания, а также осуществлять сортировку для одинаковых фамилий по именам, также по возрастанию:

<xsl:for-each select="Список_книг/книга" order-by="+автор/фамилия; +автор/имя">

В другом примере следующая установка ***order-by*** осуществляет сортировку элементов ***книга*** по названиям книг по убыванию:

<xsl:apply-templates select="Список_книг/книга" order-by="-название">

Оператор пути, который присваивается атрибуту ***order-by***, действует относительно образца, назначенного атрибуту ***select***. Так, в данном примере

установка order-by="-название" указывает на элемент ***название*** внутри элемента ***книга***, вложенного в элемент ***Список_книг***.

Пример таблицы стилей, осуществляющей фильтрацию и сортировку

Для применения фильтрации и сортировки нужно использовать пространство имен xmlns:xsl="http://www.w3.org/TR/WD-xsl"

Пример 7. XSL-таблиц стилей с фильтрацией и сортировкой элементов *книга*

```
<?xml version="1.0" encoding="windows-1251" ?>
- <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  - <xsl:template match="/">
      <H2>Описание книг</H2>
    - <xsl:for-each select="Список_книг/книга[обложка='мягкая']" order-by="+автор/фамилия; +автор/имя">
        <SPAN STYLE="font-style:italic">Название:</SPAN>
        <xsl:value-of select="название" />
        <BR />
        <SPAN STYLE="font-style:italic">Автор:</SPAN>
        <xsl:value-of select="автор" />
        <BR />
        <SPAN STYLE="font-style:italic">Обложка:</SPAN>
        <xsl:value-of select="обложка" />
        <BR />
        <SPAN STYLE="font-style:italic">Количество страниц:</SPAN>
        <xsl:value-of select="страницы" />
        <BR />
        <SPAN STYLE="font-style:italic">Цена:</SPAN>
        <xsl:value-of select="цена" />
        <P />
      </xsl:for-each>
    </xsl:template>
  </xsl:stylesheet>
```

Обе таблицы стилей разработаны для связывания с XML-документом из примера 3 (книги.xml). В них использован следующий фильтр, предписывающий браузеру отображать только книги, имеющие мягкую обложку:

[обложка='мягкая']

В обоих примерах используется следующая установка ***order-by***, задающая сортировку элементов ***книга*** по возрастанию по фамилиям авторов, а затем по именам авторов:

order-by="+автор/фамилия; +автор/имя"

Пример 8. XSL-таблиц стилей (с шаблоном <xsl:apply-templates select="Список_книг/книга" order-by="+автор/фамилия; +автор/имя"/>) с фильтрацией и сортировкой элементов *книга*

```
<?xml version="1.0" encoding="windows-1251" ?>
- <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  - <xsl:template match="/">
      <H2>Описание книг</H2>
      <xsl:apply-templates select="Список_книг/книга" order-by="+автор/фвмилия; +автор/имя" />
    </xsl:template>
  - <xsl:template match="книга[обложка='мягкая']">
      <SPAN STYLE="font-style:italic">Название:</SPAN>
      <xsl:value-of select="название" />
      <BR />
      <SPAN STYLE="font-style:italic">Автор:</SPAN>
      <xsl:value-of select="автор" />
      <BR />
      <SPAN STYLE="font-style:italic">Обложка:</SPAN>
      <xsl:value-of select="обложка" />
      <BR />
      <SPAN STYLE="font-style:italic">Количество страниц:</SPAN>
      <xsl:value-of select="страницы" />
      <BR />
      <SPAN STYLE="font-style:italic">Цена:</SPAN>
      <xsl:value-of select="цена" />
      <P />
    </xsl:template>
  </xsl:stylesheet>
```

На рисунке 6 показано как выглядит первая часть выводимой информации, которая является одинаковой для обеих таблиц стилей:

Рисунок 6 – Вывод содержимого xml-документа после применения таблиц стилей с фильтрацией и сортировкой

Таблица стилей из примера 7 использует элемент ***for-each*** для отображения множества элементов ***книга***. В приведенной ниже таблице стилей для элемента ***for-each*** установлены и фильтр, и атрибут ***order-by***:

```
<xsl:for-each
  select="Список_книг/книга[обложка='мягкая']"
  order-by="+автор/фамилия; +автор/имя">
  <!-- отображение текущего элемента книга -->
</xsl:for-each>
```

Таблица стилей из примера 8 использует для отображения множества элементов ***книга*** элемент ***apply-templates*** вместе с отдельными шаблонами, отвечающими элементам ***книга***. В этой таблице стилей фильтр добавлен к шаблону, соответствующему элементам ***книга***:

```
<xsl:template match="книга[обложка='мягкая']">
```

Добавление фильтра к элементу ***apply-templates*** будет иметь тот же эффект.

Атрибут ***order-by*** может быть добавлен к элементу ***apply-templates*** следующим образом:

```
<xsl:apply-templates select="Список_книг/книга"
  order-by="+автор/фамилия; +автор/имя"/>
```

Атрибут ***order-by*** следует добавить к элементу ***apply-templates***, поскольку элемент ***template*** не распознает этот атрибут (можно использовать атрибут ***order-by*** только для элемента, который указывает браузеру осуществить просмотр среди множества элементов – а именно, ***for-each*** и ***apply-templates***).

Доступ к атрибутам XML

XSL трактует атрибут, принадлежащий элементу в XML-документе, как дочерний элемент. Однако для ссылки на атрибут в образце XSL необходимо предварить имя атрибута символом @, что указывает на то, что имя относится к атрибуту, а не к элементу.

Например, фильтр в следующем начальном теге выделяет все элементы ***книга*** с атрибутом ***продажа***, имеющем значение "да", т.е. он выбирает только книги, которые имеются в продаже:

```
<xsl:for-each select="Список_книг/книга[@продажа='да']">
```

Можно использовать XSL-элемент ***value-of*** для извлечения значений атрибута точно так же, как это делается для извлечения текстового содержимого элемента. Например, следующий элемент ***value-of*** получает значение атрибута ***ГодИздания***, принадлежащего элементу ***автор***:

```
<xsl:value-of select="автор/@ГодИздания"/>
```

Таблица стилей, представленная в примере 10, демонстрирует технику доступа к атрибутам, принадлежащим элементам в XML-документе. Эта таблица стилей связана с XML-документом из примера 9 и отображает все имеющиеся в продаже книги из списка.

Пример 9. Пример XML-документа с атрибутами

```
<?xml version="1.0" encoding="windows-1251" ?>
- <Список_книг>
  - <книга продажа="да">
      <название>Введение в SQL</название>
    - <автор ГодИздания="2015">
        <имя>Мартин</имя>
        <фамилия>Грубер</фамилия>
      </автор>
      <обложка>твердая</обложка>
      <страницы>378</страницы>
      <цена>618 рубля</цена>
    </книга>
  - <книга продажа="нет">
      <название>Microsoft SQL Server 2012. Основы T-SQL</название>
    - <автор ГодИздания="2013">
        <имя>Ицик</имя>
        <фамилия>Бен-Ган</фамилия>
      </автор>
      <обложка>мягкая</обложка>
      <страницы>400</страницы>
      <цена>743 рубля</цена>
    </книга>
  - <книга продажа="да">
      <название>SQL. Справочник</название>
    - <автор ГодИздания="2013">
        <имя>Кевин Е.</имя>
        <фамилия>Кляйн</фамилия>
      </автор>
      <обложка>мягкая</обложка>
      <страницы>656</страницы>
      <цена>1259 рублей</цена>
    </книга>
</Список_книг>
```

Пример 10. Пример XSL-документа для обработки XML-документа с атрибутами

```
<?xml version="1.0" encoding="windows-1251" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <H2>Книги в продаже</H2>
    <TABLE BORDER="1" CELLPADDING="5">
      <THEAD>
        <TH>Название</TH>
        <TH>Автор</TH>
        <TH>Обложка</TH>
        <TH>Количество страниц</TH>
        <TH>Цена</TH>
      </THEAD>
      <xsl:for-each select="Список_книг/книга[@продажа='да']">
        <TR ALIGN="CENTER">
          <TD>
            <xsl:value-of select="название" />
          </TD>
          <TD>
            <xsl:value-of select="автор" />
            <BR />
            (Год издания
            <xsl:value-of select="автор/@ГодИздания" />
            )
          </TD>
          <TD>
            <xsl:value-of select="обложка" />
          </TD>
          <TD>
            <xsl:value-of select="страницы" />
          </TD>
          <TD>
            <xsl:value-of select="цена" />
          </TD>
        </TR>
      </xsl:for-each>
    </TABLE>
  </xsl:template>
</xsl:stylesheet>
```

Каждый элемент ***книга*** в XML-документе содержит атрибут ***продажа***, имеющий значение "да" или "нет", указывающий наличие или отсутствие книги на складе. Каждый элемент автор имеет атрибут ***ГодИздания***, содержащий год издания книги данного автора.

Вместо отображения значения атрибута ***продажа*** таблица стилей использует атрибут в условии фильтрации с целью избежать отображения элементов ***книга*** для книг, которых нет в наличии:

```
<xsl:for-each select="Список_книг/книга[@продажа='да']">
  <!-- отображение каждого элемента книга -->
</xsl:for-each>
```

Таблица стилей отображает каждый элемент ***книга*** в виде HTML-таблицы, а не через список элементов SPAN, как в предыдущих примерах. Она отображает значение атрибута ***ГодИздания*** после значения элемента ***автор***, используя XSL-элемент ***value-of***. Следующие элементы создают ячейку таблицы для отображения этих значений:

```
<TD>
  <xsl:value-of select="автор"/> <BR/>
  (год издания <xsl:value-of select="автор/@ГодИздания"/>)
</TD>
```

Книги в продаже

Название	Автор	Обложка	Количество страниц	Цена
Введение в SQL	Мартин Грубер (Год издания 2015)	твердая	378	618 рубля
SQL. Справочник	Кевин Е. Кляйн (Год издания 2013)	мягкая	656	1259 рублей