Школа

XSL

Данная серия документов подготовлена на основе материалов сайта Школы Консорциума W3C. Этот сайт является экспериментальным сервером, на котором содержание документов хранится в формате XML. Пользователям сайта эти документы доступны в виде HTML (преобразование на строне клиента с помощью таблицы стилей XSLT) и в виде PDF (преобразование тех же документов в XSL-FO, а затем в формат PDF).

Добро пожаловать в школу XSL

Школа XSL

В школе XSL вы узнаете что такое XSL. Вы также узнаете как применять XSL для того, чтобы трансформировать XML-документы в другие форматы, например, в HTML. Изучайте XSL![1] (open link)

Элементы XSL

В школе W3Schools вы найдете полное описание XSL-элементов, используемых в IE5[2] (open link), а также описание XSL-элементов по стандарту W3C[3] (open link).

Книги по XSL

Ищите хорошую книгу по XSL? Почитайте наш обзор! [4] (open link)

Содержание

1: http://xml.nsu.ru/xsl/xsl_intro.xml
2: http://xml.nsu.ru/xsl/xsl_references_ie5.xml
3: http://xml.nsu.ru/xsl/xsl_references_w3c.xml
4: http://www.w3schools.com/xsl/xsl_books.asp
5: http://xml.nsu.ru/xsl/xsl_intro.xml
6: http://xml.nsu.ru/xsl/xsl_languages.xml
7: http://xml.nsu.ru/xsl/xsl_browsers.xml
8: http://xml.nsu.ru/xsl/xsl_transformations.xml

Как в XSL используются шаблоны для того, чтобы задать преобразование XML в другой выходной формат.

Элемент for each[10] (open link)

Элемент <xsl:for-each> применяется для отбора в выходной поток всех элементов.

XSL на клиенте[11] (open link)

Как применяется XML-парсер для преобразования XML-документа в HTML-документ на машине-клиенте.

XSL на сервере[12] (open link)

Как применяется XML-парсер для преобразования XML-документа в HTML-документ на машине-сервере.

Сортировка в XSL[13] (open link)

Как заставить XML-парсер провести сортировку вашего XML-документа до того, как он преобразует его в HTML.

Фильтрование в XSL[14] (open link)

Как заставить XML-парсер провести фильтрование вашего XML-документа до того, как он преобразует его в HTML.

Условие IF в XSL[15] (open link)

Как заставить XML-парсер преобразовывать ваш XML-документ в HTML на основе каких-либо условий.

Условный отбор в XSL[16] (open link)

Как заставить XML-парсер производить условный отбор преобразований.

Описание элементов XSL

Описание элементов XSLT в IE5/5.5[17] (open link)

Internet Explorer 5 поддерживает 20 элементов XSLT.

Описание элементов XSLT по рекомендации W3C[18] (open link)

В этом разделе приводится список элементов XSLT по рекомендации W3C.

Ресурсы по XSL

Книги по XML[19] (open link)

Ищете хорошую книгу по XML? Почитайте наше обозрение!

9: http://xml.nsu.ru/xsl/xsl_templates.xml
10: http://xml.nsu.ru/xsl/xsl_for_each.xml
11: http://xml.nsu.ru/xsl/xsl_on_client.xml
12: http://xml.nsu.ru/xsl/xsl_on_server.xml
13: http://xml.nsu.ru/xsl/xsl_sorting.xml
14: http://xml.nsu.ru/xsl/xsl_filtering.xml
15: http://xml.nsu.ru/xsl/xsl_if.xml
16: http://xml.nsu.ru/xsl/xsl_choose.xml
17: http://xml.nsu.ru/xsl/xsl_references_ie5.xml
18: http://xml.nsu.ru/xsl/xsl_references_w3c.xml
19: http://www.w3schools.com/xsl/xsl_books.asp

Введение в XSL

XSL, язык стилей XML, гораздо изощреннее, чем CSS

CSS - язык стилей HTML

Поскольку HTML использует изначально заданные тэги, значение этих тэгов хорошо известно и понятно: элемент <p> задает параграф, а элемент <h1> задает заголовок - браузер знает, как их показывать.

Добавлять элементам HTML стилевые характеристики показа с помощью CSS несложно. Сообщить браузеру о том каким цветом и каким шрифтом показывать тот или иной элемент - это просто делается и легко понятно браузеру.

XSL - язык стилей XML

Поскольку XML не использует изначально заданные тэги (мы можем использовать какие угодно тэги), значение их не понятно изначально: <table> может означать HTML-таблицу, а может означать часть мебели. По самой природе XML, браузер не знает как показывать XML-документ.

Чтобы показать XML-документ, необходимо иметь механизм описания того, как должен выглядеть документ. Одним из таких механизмов является CSS, но XSL (the eXtensible Stylesheet Language - расширяемый язык стилей) - гораздо более предпочтительней как язык стилей XML. Кроме того, он гораздо многообразней, чем CSS который применяется в HTML.

XSL - больше, чем просто таблица стилей

XSL состоит из трех частей:

- метода преобразования XML-документов
- метода задания частей и путей к элементам XML
- метода форматирования XML-документов

Если вам это непонятно, смотрите на XSL как на язык, который может преобразовывать XML в HTML, который может фильтровать и сортировать XML-данные, который может адресно обращаться к различным частям XML-документа, который может форматировать XML-данные в зависимости от их значения (например, показывать отрицательные числа красным цветом) и который может подготавливать XML-данные к выводу на различные устройства, например, экран, бумагу или звуковое воспроизведение.

XSL - это стандарт для WWW

XSL - стандарт, рекомендованый World Wide Web Consortium.

Первые две части этого языка стали официальной рекомендацией W3C в ноябре 1999 года. Полная рекомендация по XSL, включая XSL-форматирование, стала кандидатом в официальные рекомендации в ноябре 2000 года.

Вы можете больше узнать о деятельности W3C на нашей школе W3C School.[20] (open link)

20: http://www.w3schools.com/w3c/w3c_xsl.asp

Языки XSL

Вообще-то XSL состоит из трех языков. Самый важный - XSLT

XSL состоит из трех языков

На самом деле XSL остоит из трех языков:

- XSLT - язык преобразований XML
- XPath - язык определения частей и путей к элементам XML
- XSL Formatting Objects - язык определения показа XML

XSLT - язык преобразования XML в другие типы документов или в другие XML-документы.

XPath - язык обращений к частям XML-документа. XPath создан для использования языком XSLT.

Форматирование - процесс превращения результата XSL-преобразования в форму, удобную для читателя или слушателя.

XSLT и XPath были предложены в виде двух отдельных официальных рекомендаций W3C 16 ноября 1999 года. Для языка XSL Formatting Objects нет отдельной рекомендации, но его описание можно найти в рекомендациях по языку XSL 1.0.

XSLT - XSL-преобразования

XSLT является наиболее важной частью стандарта XSL. Именно эта часть XSL применяется для того, чтобы преобразовывать XML документ в другой XML-документ или в другие типы документов.

XSLT можно применять для преобразования XML-документа в формат, знакомый браузерам. Одним из таких форматов является HTML. Обычно XSLT достигает этого, преобразуя каждый XML-элемент в HTML-элемент.

Кроме того, XSLT может добавлять совершенно новые элементы в выходной файл или удалять элементы. Этот язык может изменить порядок элементов, произвести проверку и на ее основе решить какие элементы показывать и еще многое другое.

Наиболее общее описание процесса преобразования звучит так: XSL использует XSLT для преобразования XML-дерева-оригинала в XML-дерево-результат (или XML-документ-оригинал в XML-документ-результат).

Как это происходит?

В процессе преобразования XSLT использует XPath для определения тех частей в документе-оригинале, которые соответствуют одному или более заранее заданным шаблонам. Когда такое соответствие обнаруживается, XSLT преобразует соответствующую часть в документе-оригинале в документ-результат. Те части документа-оригинала, которые не соответствуют шаблону будут (как гласит общее правило) попадать в документ-результат немодифицированными.

Этот учебник будт фокусироваться на XSLT и XPath

Большая часть глав этого учебника будет фокусироваться на XSLT и XPath. Мы будем использовать XSLT для задания XML-преобразований, и язык XPath - для определения путей к элементам, которые должны быть подвергнуты преобразованию.

Хотя XSL состоит из трех различных частей, имеющих разные названия, в этом учебнике бы будем использовать единое название XSL.

Браузеры XSL

В настоящее время очень небольшое количество браузеров поддерживает XSL
В этом учебнике для демонстрации XSL мы используем Internet Explorer

Встроенный в Internet Explorer XML-парсер

Для того, чтобы обработать XML-документ используя XSL, вам понадобится XML-парсер с XSL-машиной. В настоящее время Internet Explorer 5.0 является единственным широко распространенным браузером, который имеет встроенный XML-парсер с XSL-машиной.

В настоящее время примеры кода в этом учебнике будут работать только в Internet Explorer 5.0 или его более поздних версиях.

Internet Explorer 5 не очень хороший

XSL, реализованный в Internet Explorer 5 не совместим с официальной рекомендацией W3C по языку XSL.

На момент, когда Internet Explorer 5.0 появился на рынке (март 1999-го года) стандарт XSL оставался в виде рабочей версии W3C.

Поскольку окончательная W3C-рекомендация по XSL отличается от рабочего варианта, поддержка XSL, реализованная в IE 5 не полностью совместима с официальным стандартом XSL.

Это относится не только к Internet Explorer 5.0, но и к 5.5.

Internet Explorer 6 лучше

Internet Explorer 6 полностью поддерживает официальную рекомендацию W3C по языку XSL.

XML-парсер msxml3, встроенный в Internet Explorer 6.0 и Windows XP, разработан на основе рекомендаций W3C XSLT 1.0 и XPath 1.0.

Примеры этой школы будут работать только в Internet Explorer 5 и выше. Из них многие будут работать только в Internet Explorer 6.0.

Если вы всерьез настроены изучить XSL, вам следует сделать апгрейд до Internet Explorer 6.0.

Правильное объявление таблицы стилей

Вот как выглядит правильное объявление таблицы стилей XSL в соответствии с рекомендациями W3C по XSL:

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

Если вы используете Internet Explorer 6.0, вам следует применять именно этот код.

Неправильное объявление таблицы стилей

Вот как выглядит правильное объявление таблицы стилей XSL в соответствии с устаревшей рабочей версией W3C по XSL:

```
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/TR/WD-xsl">
```

Если вы используете Internet Explorer 5.0, вам следует применять этот (не правильный) код.

Парсер MSXML

MSXML 2.0 - так называется XML-парсер, встроенный в IE 5.0.

MSXML 2.5 - так называется парсер, который идет с Windows 2000 и IE 5.5.

MSXML 3.0 - это предпоследний релиз XML-парсера, он идет вместе с Internet Explorer 6.0 и Windows XP.

Как утверждает Microsoft, парсер MSXML 3.0 соответствует на 100% официальной рекомендации W3C по языку XSL:

Парсер MSXML 3.0 представляет значительное продвижение вперед по сравнению с MSXML 2.5: безопасный для сервера HTTP-доступ, полная реализация XSLT и XPath, внесены изменения в SAX (Simple API for XML), достигнуто лучшее согласие со стандартами W3C, устранено несколько багов.

Здесь вы найдете дополнительную информацию по этому вопросу:
http://msdn.microsoft.com/xml/general/xmlparser.asp[1]

Вы можете узнать больше о последних релизах браузера IE а нашем разделе Browser Section:[21] (open link)

Преобразования в XSL

Пример для изучения: как преобразовать XML в HTML, используя XSL.
В деталях этот пример мы разберем в следующих разделах.

Начинаем с нашего XML-документа

Для начала возьмем XML-документ, который мы хотим преобразовать в HTML:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
```

1: http://msdn.microsoft.com/xml/general/xmlparser.asp
21: http://www.w3schools.com//browsers/

```
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
.
.
.
```

Если вы используете Internet Explorer 5.0, вы можете посмотреть исходный XML-документ:[22]
(open xml)

Создайте таблицу стилей: XSL-документ

Затем создаем таблицу стилей XSL с шаблоном преобразования:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <table border="2" bgcolor="yellow">
      <tr>
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Если вы используете Internet Explorer 5.0, вы можете посмотреть данный XSL-документ:[23]
(open xsl)

Свяжите XML-документ с таблицей стилей

Теперь следует добавить ссылку на таблицу стилей XSL в вашем XML-документе:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
```

22: http://xml.nsu.ru/xsl/cdcatalog.xml
23: http://xml.nsu.ru/xsl/cdcatalog.xsl

```
    <price>10.90</price>
    <year>1985</year>
  </cd>
.
.
.
</catalog>
```

Если ваш браузер поддерживает XSL, например, Internet Explorer 5.0, то он преобразует ваш XML в HTML-документ.

Взгляните на результат (для Internet Explorer 5.0):[24] (open xml)

Взгляните на результат (для Internet Explorer 6.0):[25] (open xml)

Объяснение примера

Подробный разбор этого примера и многие другие примеры будут приведены в следующих разделах.

Шаблоны XSL

Для описания того, в каком виде выводить XML-файл, в XSL используются шаблоны.

В CSS используются правила

Если вы проходили обучение в школе CSS, то вы знаете, что в CSS для задания вида выводимых HTML-элементов используются одно или несколько правил. Для связывания определенного правила с HTML-элементом используются селекторы.

В ниже приведенном CSS-правиле селектор "p" показывает, что элемент <p> должен отображаться шрифтом Arial:

```
p { font-family: arial }
```

В XSL используются шаблоны

Для определения вида, в котором будут выводиться XML-элементы, в XSL используется один или несколько шаблонов. Для связывания шаблона с XML-элементом используется специальный атрибут соответствия (атрибут соответствия может также применяться для задания шаблона отображения всего XML-документа).

Взгляните на пример XSL-таблицы стилей, в которой задан шаблон для вывода XML-каталога компакт дисков (пример из предыдущего раздела):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
 <html>
```

24: http://xml.nsu.ru/xsl/cdcatalog_with_xsl_old.xml
25: http://xml.nsu.ru/xsl/cdcatalog_with_xsl.xml

```
  <body>
    <table border="1">
      <tr>
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <tr>
        <td>.</td>
        <td>.</td>
      </tr>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Поскольку таблица стилей сама по себе является XML-документом, она начинается с декларации XML: <?xml version='1.0' encoding='ISO-8859-1'?>

Тэг xsl:stylesheet, который находится на второй строке документа задает начало таблицы стилей.

Тэг xsl:template в третьей строке задает начало шаблона. Атрибут шаблона match="/" связывает (приводит в соответствие) шаблон и корень (/) оригинального XML-документа.

Остальная часть документа содержит сам шаблон, кроме двух последних строк, которые задают конец шаблона и конец таблицы стилей.

Если вы используете Internet Explorer 5.0, вы можете открыть исходный XML-документ: [26] (open xml)

Так выглядит наша таблица стилей XSL: [27] (open xsl)

А так выглядит результат преобразования: [28] (open xml)

Элемент <xsl:value-of>

Мы только что получили несколько странный результат, поскольку данные из нашего XML-документа не дошли до нас.

Для того, чтобы ввести определенный XML-элемент в выходной поток XSL-преобразования, применяется элемент <xsl:value-of>:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
 <html>
 <body>
   <table border="1">
     <tr>
       <th>Title</th>
       <th>Artist</th>
     </tr>
     <tr>
```

26: http://xml.nsu.ru/xsl/cdcatalog.xml
27: http://xml.nsu.ru/xsl/cdcatalog_ex1.xsl
28: http://xml.nsu.ru/xsl/cdcatalog_with_ex1.xml

```
      <td><xsl:value-of select="catalog/cd/title"/></td>
      <td><xsl:value-of select="catalog/cd/artist"/></td>
     </tr>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Обратите внимание: синтаксис значения атрибута select называется XSL-паттерном (XSL Pattern). Он устроен так же, как навигация в файловой системе, где прямой слэш (/) указывает на под-директории.

Если вы используете Internet Explorer 5.0, вы можете открыть исходный XML-документ: [29] (open xml)

Так выглядит наша обновленная таблица стилей XSL: [30] (open xsl)

Взгляните на результат (для Internet Explorer 5.0):[31] (open xml)

Взгляните на результат (для Internet Explorer 6.0):[32] (open xml)

В следующем разделе вы узнаете, как использовать элемент <xsl:for-each> для отбора в выходной поток трансформации каждого XML-элемента.

Элемент <xsl:for-each>

Мы снова получили немного странный результат, поскольку только одна строка информации была скопирована из XML-файла на выход.

Для внесения в выходной поток XSL-преобразования каждого XML-элемента можно применить XSL-элемент <xsl:for-each>:

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
  <html>
  <body>
    <table border="1">
      <tr>
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="CATALOG/CD">
      <tr>
        <td><xsl:value-of select="TITLE"/></td>
        <td><xsl:value-of select="ARTIST"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
```

29: http://xml.nsu.ru/xsl/cdcatalog.xml
30: http://xml.nsu.ru/xsl/cdcatalog_ex2.xsl
31: http://xml.nsu.ru/xsl/cdcatalog_with_ex2_old.xml
32: http://xml.nsu.ru/xsl/cdcatalog_with_ex2.xml

```
</xsl:stylesheet>
```

Элемент xsl:for-each определяет местоположение элементов в XML-документе и повторяет часть шаблона для каждого.

Если вы используете Internet Explorer 5.0, вы можете открыть исходный XML-документ: [33] (open xml)

Так выглядит наша финальная таблица стилей XSL: [34] (open xsl)

А так выглядит окончательный результат преобразования: [35] (open xml)

Элемент for each

Элемент <xsl:for-each> применяется для отбора в выходной поток всех элементов

Элемент <xsl:for-each>

В предыдущем разделе мы снова получили немного странный результат, поскольку только одна строка информации была скопирована из XML-файла на выход.

Для внесения в выходной поток XSL-преобразования каждого XML-элемента можно применить XSL-элемент <xsl:for-each>:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <table border="1">
      <tr>
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Элемент xsl:for-each определяет местоположение элементов в XML-документе и повторяет часть шаблона для каждого из них.

Если вы используете Internet Explorer 5.0, вы можете открыть исходный XML-документ: [36]

33: http://xml.nsu.ru/xsl/cd_catalog.xml
34: http://xml.nsu.ru/xsl/cd_catalog_ex3.xsl
35: http://xml.nsu.ru/xsl/cd_catalog_with_ex3.xml

(open xml)

Так выглядит наша финальная таблица стилей XSL: [37] (open xsl)

Взгляните на результат (для Internet Explorer 5.0):[38] (open xml)

Взгляните на результат (для Internet Explorer 6.0):[39] (open xml)

XSL на клиенте

Если ваш браузер поддерживает XML, для преобразования XML-документа в HTML можно применять XSL

Применение JavaScript

В предыдущей главе мы познакомились с тем, как с помощью XSL можно преобразовывать документ из XML в HTML. Мы просто добавляли к XML-документу таблицу стилей XSL и браузер производил преобразование.

Несмотря на то, что этот метод замечательно работает, не всегда желетельно включать в XML-файл ссылку на внешнюю таблицу стилей, кроме того такой метод не будет работать в браузерах, которые не поддерживают XSL.

Более универсальное решение - предоставить JavaScript проводить преобразование из XML в HTML.

Применяя JavaScript, мы получаем следующие возможности:

- JavaScript может проверить тип браузера
- В зависимости от желаний пользователя и типа браузера можно использовать различные таблицы стилей

В этом и заключена прелесть XSL. Одно из главных преимуществ XSL - это возможность перобразования данных из одного формата в другой, поддерживать различные браузеры и желания пользователей.

XSL-преобразования на машине-клиенте скорее всего будет основной работой браузеров в будущем - уже сейчас можно наблюдать рост рынка специализированных браузеров (со шрифтами Брайля, голосовой веб, карманные компьютеры, мобильные телефоны...)

XML-файл и XSL-файл

Снова взгляните на XML-документ, который вы видели в предыдущей главе:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
```

36: http://xml.nsu.ru/xsl/cdcatalog.xml
37: http://xml.nsu.ru/xsl/cdcatalog_ex3.xsl
38: http://xml.nsu.ru/xsl/cdcatalog_with_ex3_old.xml
39: http://xml.nsu.ru/xsl/cdcatalog_with_ex3.xml

```
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.
.
.
</catalog>
```

Вы можете открыть этот XML-документ в Internet Explorer: [40] (open xml)

А вот связанная с ним таблица стилей XSL:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <table border="2" bgcolor="yellow">
      <tr>
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Вы можете открыть этот XSL-документ в Internet Explorer: [41] (open xsl)

Синтаксис этого XSL-документа был объяснен в предыдущей главе, так что мы его не будем здесь расшифровывать. Но обязательно обратите внимание, что XML не ссылается на XSL-файл, а он, в свою очередь, не ссылается на XML-файл.

ЭТО ВАЖНО: Выше приведенный пример показывает, что XML-файл может быть преобразован с использованием многих разных XSL-файлов.

Преобразование XML в HTML в вашем браузере

Вот простой пример кода, который преобразует XML-файл в HTML на машине-клиенте:

```
<html>
<body>
<script type="text/javascript">
// Load XML
var xml = new ActiveXObject("Microsoft.XMLDOM")
xml.async = false
```

40: http://xml.nsu.ru/xsl/cdcatalog.xml
41: http://xml.nsu.ru/xsl/cd_catalog.xsl

```
xml.load("cdcatalog.xml")

// Load the XSL
var xsl = new ActiveXObject("Microsoft.XMLDOM")
xsl.async = false
xsl.load("cdcatalog.xsl")

// Transform
document.write(xml.transformNode(xsl))
</script>

</body>
</html>
```

(В этом примере используется JavaScript. Если вы еще не знакомы с языком JavaScript, посетите нашу школу JavaScript:[42] (open link))

Первый блок кода производит включение XML-парсера Microsoft (XMLDOM) и загружает XML-документ в память. Второй блок кода производит еще одно включение парсера и загружает в память XSL-документ. Последняя строка кода преобразует XML-документ используя XSL-документ и выводит результат в HTML-документ.

Просто и красиво.

Используя Internet Explorer 6.0, вы можете попробовать работу скрипта сами:[43] (open editor)

Или для Internet Explorer 5.0:[44] (open editor)

XSL на сервере

Поскольку не все браузеры поддерживают XML и XSL, возможное решение: преобразование XML в HTML на сервере

Решение, не зависящее от браузера

В предыдущей главе мы познакомились с тем, как с помощью XSL можно преобразовывать документ из XML в HTML браузером: JavaScript использует XML-парсер и проводит преобразование.

Этот метод не будет работать в браузерах, которые не поддерживают XML-парсер.

Чтобы сделать XML-данные доступными для всех видов браузеров, нам следует преобразовывать XML-документ на сервере и отправлять браузеру уже чистый HTML.

Это еще одно преимущество XSL: одной из целей создания XSL было сделать возможным преобразование на сервере данных из одного формата в другой, отправляя данные в пригодном для чтения виде для всех типов браузеров.

XSL-преобразования на машине-сервере скорее всего будет основным занятием интернетовских информационных серверов - уже сейчас можно наблюдать рост рынка специализированных браузеров (со шрифтами Брайля, голосовой веб, карманные компьютеры, мобильные

42: http://xml.nsu.ru/js/js_home.xml
43: http://www.w3schools.com/xsl/tryit.asp?filename=cdcatalog
44: http://www.w3schools.com/xsl/tryit.asp?filename=cdcatalog_old

телефоны...)

XML-файл и XSL-файл

Снова взгляните на XML-документ, который вы видели в предыдущей главе:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.
.
.
</catalog>
```

Вы можете открыть этот XML-документ в Internet Explorer: [45] (open xml)

А вот связанная с ним таблица стилей XSL:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <table border="2" bgcolor="yellow">
      <tr>
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Вы можете открыть этот XSL-документ в Internet Explorer: [46] (open xsl)

Синтаксис этот XSL-документа был объяснен в предыдущей главе, так что мы его не будем здесь расшифровывать. Но обязательно обратите внимание, что XML не ссылается на XSL-файл, а он, в свою очередь, не ссылается на XML-файл.

ЭТО ВАЖНО: Выше приведенный пример показывает, что XML-файл на сервере может быть

45: http://xml.nsu.ru/xsl/cdcatalog.xml
46: http://xml.nsu.ru/xsl/cdcatalog.xsl

преобразован с использованием многих разных XSL-файлов.

Преобразование XML в HTML на сервере

Вот простой пример кода, который преобразует XML-файл HTML на сервере:

```
<%
'Load the XML
set xml = Server.CreateObject("Microsoft.XMLDOM")
xml.async = false
xml.load(Server.MapPath("cdcatalog.xml"))

'Load the XSL
set xsl = Server.CreateObject("Microsoft.XMLDOM")
xsl.async = false
xsl.load(Server.MapPath("cdcatalog.xsl"))

'Transform the file
Response.Write(xml.transformNode(xsl))
%>
```

(Это пример ASP-файла, а код написан на VBScript. Если вы не знакомы с ASP или VBScript, посетите нашу школу ASP:[47] (open link))

Первый блок кода производит включение XML-парсера Microsoft (XMLDOM) и загружает XML-документ в память. Второй блок кода производит еще одно включение парсера и загружает в память XSL-документ. Последняя строка кода преобразует XML-документ используя XSL-документ и отправляет результат браузеру.

Просто и красиво.

Посмотрите, как это работает:[48] (open link)

Сортировка в XSL

XSL можно применять для сортировки элементов в XML-документе

Куда помещать сортировочную информацию

Взгляните на XML-документ, который вы уже видели во многих разделах:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.
```

47: http://www.w3schools.com/asp/default.asp
48: http://www.w3schools.com/xsl/cdcatalog.asp

```
.
.
</catalog>
```

Вы можете открыть этот XML-документ в Internet Explorer: [49] (open xml)

Чтобы вывести этот файл как простой файл HTML и заодно его отсортировать, просто добавьте элемент sort внутрь элемента for-each в вашем XSL-файле, например, так:

```
<xsl:sort select="artist"/>
```

Теперь посмотрите, что должно у нас получиться:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <table border="2" bgcolor="yellow">
      <tr>
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <xsl:sort select="artist"/>
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Вы можете открыть этот XSL-документ в Internet Explorer: [50] (open xsl)

Преобразование документа в браузере

Вот простой пример кода, который необходим для преобразования XML-файл HTML в браузере:

```
<html>
<body>
<script type="text/javascript">
// Load XML
var xml = new ActiveXObject("Microsoft.XMLDOM")
xml.async = false
xml.load("cdcatalog.xml")

// Load the XSL
var xsl = new ActiveXObject("Microsoft.XMLDOM")
xsl.async = false
```

49: http://xml.nsu.ru/xsl/cdcatalog.xml
50: http://xml.nsu.ru/xsl/cdcatalog_sort.xsl

```
xsl.load("cdcatalog_sort.xsl")

// Transform
document.write(xml.transformNode(xsl))
</script>

</body>
</html>
```

Используя Internet Explorer 5, вы можете попробовать работу скрипта сами. [51] (open editor)

Фильтрование в XSL

XSL можно применять для фильтрования элементов в XML-документе

Куда помещать информацию о фильтровании

Взгляните на XML-документ, который вы уже видели много раз:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.
.
.
</catalog>
```

Вы можете открыть этот XML-документ в Internet Explorer: [52] (open xml)

Чтобы профильтровать этот XML-файл, нужно просто добавить фильтр к атрибуту select в элементе for-each XSL-файла, примерно так:

```
<xsl:for-each select="catalog/cd[artist='Bob Dylan']">
```

Допустимыми операциями фильтрования являются:

- = (равно)
- != (не равно)
- < (меньше)
- > (больше)

Теперь посмотрите, что должно у нас получиться:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
```

51: http://www.w3schools.com/xsl/tryxsl.asp?filename=cdcatalog_sort
52: http://xml.nsu.ru/xsl/cdcatalog.xml

```
<html>
<body>
 <table border="2" bgcolor="yellow">
  <tr>
   <th>Title</th>
   <th>Artist</th>
  </tr>
  <xsl:for-each select="catalog/cd[artist='Bob Dylan']">
  <tr>
   <td><xsl:value-of select="title"/></td>
   <td><xsl:value-of select="artist"/></td>
  </tr>
  </xsl:for-each>
 </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Вы можете открыть этот XSL-документ в Internet Explorer: [53] (open xsl)

Преобразование документа в браузере

Вот простой пример кода, который необходим для преобразования XML-файл в HTML в браузере. Вставьте приведенный ниже код в .htm или .html-файл:

```
<html>
<body>
<script type="text/javascript">
// Load XML
var xml = new ActiveXObject("Microsoft.XMLDOM")
xml.async = false
xml.load("cdcatalog.xml")

// Load the XSL
var xsl = new ActiveXObject("Microsoft.XMLDOM")
xsl.async = false
xsl.load("cdcatalog_filter.xsl")

// Transform
document.write(xml.transformNode(xsl))
</script>

</body>
</html>
```

Используя Internet Explorer 6.0, вы можете попробовать работу скрипта сами. [54] (open editor)

Условие IF в XSL

XSL может применять проверку IF для отфильтровывания информации из XML-документа

53: http://xml.nsu.ru/xsl/cdcatalog_filter.xsl
54: http://www.w3schools.com/xsl/tryxsl.asp?filename=cdcatalog_filter

Куда помещать условие IF

Взгляните на XML-документ, который вы уже видели много раз:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.
.
.
</catalog>
```

Вы можете открыть этот XML-документ в Internet Explorer: [55] (open xml)

Чтобы наложить условие IF на содержание файла, просто добавьте элемент xsl:if в свой XSL-файл, примерно так:

```
<xsl:if test="artist='Bob Dylan'">
... some output ...
</xsl:if>
```

Теперь посмотрите, какой XSL-файл должен у нас получиться:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <table border="2" bgcolor="yellow">
      <tr>
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <xsl:if test="artist='Bob Dylan'">
        <tr>
          <td><xsl:value-of select="title"/></td>
          <td><xsl:value-of select="artist"/></td>
        </tr>
      </xsl:if>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Вы можете открыть этот XSL-документ в Internet Explorer: [56] (open xsl)

55: http://xml.nsu.ru/xsl/cdcatalog.xml

Преобразование документа в браузере

Вот простой пример кода, который необходим для преобразования XML-файл HTML в браузере:

```
<html>
<body>
<script type="text/javascript">
// Load XML
var xml = new ActiveXObject("Microsoft.XMLDOM")
xml.async = false
xml.load("cdcatalog.xml")

// Load the XSL
var xsl = new ActiveXObject("Microsoft.XMLDOM")
xsl.async = false
xsl.load("cdcatalog_if.xsl")

// Transform
document.write(xml.transformNode(xsl))
</script>

</body>
</html>
```

Используя Internet Explorer 6.0, вы можете попробовать работу скрипта сами: [57] (open editor)

Условный отбор в XSL

XSL может применять для фильтрования XML-документа условный отбор

Куда помещать условия отбора

Взгляните на XML-документ, который вы видели почти в каждом разделе:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.
.
.
</catalog>
```

Вы можете открыть этот XML-документ в Internet Explorer: [58] (open xml)

56: http://xml.nsu.ru/xsl/cdcatalog_if.xsl
57: http://www.w3schools.com/xsl/tryxsl.asp?filename=cdcatalog_if

Чтобы наложить условия отбора на содержание файла, просто добавьте элементы xsl:choose, xsl:when и xsl:otherwise в свой XSL-файл, примерно так:

```
<xsl:choose>
<xsl:when test="artist='Bob Dylan'">
 ... некоторый код ...
</xsl:when>
<xsl:otherwise>
... некоторый код ....
</xsl:otherwise>
</xsl:choose>
```

Вот получившаяся слегка усложненная таблица стилей XSL:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <table border="2" bgcolor="yellow">
      <tr>
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
              <tr>
        <td><xsl:value-of select="title"/></td>
              <xsl:choose>
          <xsl:when test="artist='Bob Dylan'">
            <td bgcolor="#ff0000">
             <xsl:value-of select="artist"/>
            </td>
          </xsl:when>
          <xsl:otherwise>
            <td><xsl:value-of select="artist"/></td>
          </xsl:otherwise>
        </xsl:choose>
        </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Вы можете открыть этот XSL-документ в Internet Explorer: [59] (open xsl)

Преобразование документа в браузере

Вот простой пример кода, который необходим для преобразования XML-файл в HTML в браузере:

```
<html>
```

58: http://xml.nsu.ru/xsl/cdcatalog.xml
59: http://xml.nsu.ru/xsl/cdcatalog_choose.xsl

```
<body>
<script type="text/javascript">
// Load XML
var xml = new ActiveXObject("Microsoft.XMLDOM")
xml.async = false
xml.load("cdcatalog.xml")

// Load the XSL
var xsl = new ActiveXObject("Microsoft.XMLDOM")
xsl.async = false
xsl.load("cdcatalog_choose.xsl")

// Transform
document.write(xml.transformNode(xsl))
</script>

</body>
</html>
```

Используя Internet Explorer 6.0, вы можете попробовать работу скрипта сами:[60] (open editor)

Описание элементов XSLT в IE5

Internet Explorer 5 поддерживает 20 элементов XSLT
XSLT-элементы в IE5 не полностью совместимы со стандартом W3C

Internet Explorer 5 не очень хороший

XSL, реализованный в Internet Explorer 5 не совместим с официальной рекомендацией W3C по языку XSL.

На момент, когда Internet Explorer 5.0 появился на рынке (март 1999-го года) стандарт XSL оставался в виде рабочей версии W3C.

Поскольку окончательная W3C-рекомендация по XSL отличается от рабочего варианта, поддержка XSL, реализованная в IE 5 не полностью совместима с официальным стандартом XSL.

Это относится не только к Internet Explorer 5.0, но и к 5.5.

Internet Explorer 6 лучше

Internet Explorer 6 полностью поддерживает официальную рекомендацию W3C по языку XSL.

XML-парсер msxml3, встроенный в Internet Explorer 6.0 и Windows XP, разработан на основе рекомендаций W3C XSLT 1.0 и XPath 1.0.

Примеры этой школы будут работать только в Internet Explorer 5 и выше. Из них многие будут работать только в Internet Explorer 6.0.

Если вы всерьез настроены изучить XSL, вам следует сделать апгрейд до Internet Explorer 6.0.

xsl:apply-templates

60: http://www.w3schools.com/xsl/tryxsl.asp?filename=cdcatalog_choose

Применяет шаблон к текущему элементу

Атрибуты

Атрибут	Значение	Описание
order-by	+|- pattern	
select	pattern	

xsl:attribute

Добавляет новый атрибут к текущему выходному элементу

Атрибуты

Атрибут	Значение	Описание
name	attribute-name	

xsl:cdata

Добавляет новый раздел CDATA в выходной поток

xsl:choose

Предоставляет механизм отбора, основанный на условиях

xsl:comment

Добавляет узел-комментарий в выходной поток

xsl:copy

Копирует текущий узел в выходной поток

xsl:define-template-set

Задает новый набор шаблонов

xsl:element

Добавляет новый элементный узел в выходной поток

Атрибуты

Атрибут	Значение	Описание

name	name

xsl:entity-ref

Добавляет новый узел-ссылку на сущность в выходной поток

Атрибуты

Атрибут	Значение	Описание
name	name	

xsl:eval

Предоставляет механизм получения выходного содержания в результате работы скрипта

Атрибуты

Атрибут	Значение	Описание
language	language	

xsl:for-each

Предоставляет механизм для создания цикла в выходном потоке

Атрибуты

Атрибут	Значение	Описание
select	pattern	
order-by	-|+ pattern	

xsl:if

Предоставляет механизм условного ветвления, основанный на проверке условий

Атрибуты

Атрибут	Значение	Описание
match	pattern	

xsl:node-name

Добавляет имя текущего узла в выходной поток

xsl:otherwise

Часть отборного механизма (см. xsl:choose)

xsl:pi

Добавляет в выходной поток процессуальные инструкции

Атрибуты

Атрибут	Значение	Описание
name	name	

xsl:script

Задает скриптовую вставку внутри шаблона

Атрибуты

Атрибут	Значение	Описание
language	language	

xsl:stylesheet

Задает корневой элемент таблицы стилей

Атрибуты

Атрибут	Значение	Описание
xmlns:xml	namespace	
language	language	
indent-result	yes|no	

xsl:template

Задает шаблон

Атрибуты

Атрибут	Значение	Описание
match	pattern	
language	language	

xsl:value-of

Задает узел, вставляемый в выходной поток

Атрибуты

Атрибут	Значение	Описание
select	pattern	

xsl:when

Часть отборного механизма (см. xsl:choose)

Атрибуты

Атрибут	Значение	Описание
test	expression	

Описание элементов XSLT по рекомендации W3C

В этом разделе перечисляются элементы XSLT по официальной рекомендации W3C

xsl:apply-imports

Применяет правило-шаблон из импортированной таблицы стилей. Элемент содержит 0 или несколько дочерних элементов <xsl:with-param>

Синтаксис

```
<xsl:apply-imports>
  <xsl:with-param>
</xsl:apply-imports>
```

Примеры

Допустим, у нас имеется таблица стилей, которая называется standard.xsl. Эта таблица стилей содержит общее правило-шаблон для отображения узла message:

```
<xsl:template match="message">
  <xsl:value-of select="to"><br />
  <xsl:value-of select="from"><br />
  <xsl:value-of select="text">
</xsl:template>
```

Вторая таблица стилей, которая называется special.xsl, импортирует таблицу стилей standard.xsl и вставляет элементы h2 вокруг элементов to, from и text:

```
<xsl:import href="standard.xsl"/>
<xsl:template match="message">
  <h2>
    <xsl:apply-imports/>
  </h2>
```

```
</xsl:template>
```

xsl:apply-templates

Применяет шаблон к текущему элементу - задается набор узлов для обработки текущим элементом.

Синтаксис

```
<xsl:apply-templates select="expression" mode="name">
  <xsl:with-param>
  <xsl:sort>
</xsl:apply-templates>
```

Атрибуты

Атрибут	Значение	Описание
select	expression	Не обязателен. Множество обрабатываемых узлов. Если пропущен, обрабатываются все дочерние узлы данного узла
mode	name	Не обязателен. Режим обработки

Примеры

Все узлы типа title в XML-документе будут на выходе отображаться как элементы h1:

```
<xsl:template match="title">
  <h1><xsl:apply-templates/></h1>
</xsl:template>
```

Все узлы в дереве узлов элемента message будут на выходе отображаться как элементы h1:

```
<xsl:template match="message">
  <h1><xsl:apply-templates select="title"/></h1>
</xsl:template>
```

Все узлы в дереве узлов элемента message, у которых атрибут mode установлен на значение "big" будут на выходе отображаться как элементы h1:

```
<xsl:template match="message">
  <h1><xsl:apply-templates select="//*" mode="big"/></h1>
</xsl:template>
```

Все узлы типа title в дереве узлов элемента message, у которых атрибут mode установлен на значение "big" будут на выходе отображаться как элементы h1:

```
<xsl:template match="message">
  <h1><xsl:apply-templates select="title" mode="big"/></h1>
</xsl:template>
```

Все узлы в дереве узлов элемента message будут отображаться на выходе рассортированные по порожденному элементу title:

```
<xsl:template match="message">
  <xsl:apply-templates/>
  <xsl:sort select="title"/>
</xsl:template>
```

xsl:attribute

Добавляет атрибут к ближайшему содержащему элементу. Элемент можно использовать внутри тела шаблона

(template-body) или внутри элемента <xsl:attribute-set>

Элемент <xsl:attribute> замещает существующие атрибуты с тем же именем. Элемент <xsl:attribute> должен вставляться до того, как добавлен какой-либо порожденный элемент

Синтаксис

```
<xsl:attribute name="attributename" namespace="uri">
template-body
</xsl:attribute>
```

Атрибуты

Атрибут	Значение	Описание
name	attributename	Обязателен. Имя добавляемого атрибута
namespace	uri	Не обязателен. Задает uri пространства имен добавляемого атрибута

Примеры

Добавление атрибута ширина (width) к элементу таблица (table):

```
<table>
  <xsl:attribute name="width">100%</xsl:attribute>
</table>
```

Создание набора атрибутов, которые можно применить к любому выходному элементу:

```
<xsl:attribute-set name="font">
  <xsl:attribute name="font-name">Arial</xsl:attribute>
  <xsl:attribute name="font-size">14px</xsl:attribute>
  <xsl:attribute name="font-weight">bold</xsl:attribute>
</xsl:attribute-set>
```

xsl:attribute-set

Задает обладающий собственным именем набор атрибутов. Набор атрибутов можно применить к любому элементу

Синтаксис

```
<xsl:attribute-set
name="name" use-attribute-sets="name-list">
  <xsl:attribute>
</xsl:attribute-set>
```

Атрибуты

Атрибут	Значение	Описание
name	attributename	Обязателен. Имя набора атрибутов
use-attribute-sets	name-list	Не обязателен. Список (разделение пробелами) других наборов атрибутов, которые используются в данном наборе

Примеры

Создание набора атрибутов, который можно применить к любому выходному элементу:

```
<xsl:attribute-set name="font">
  <xsl:attribute name="font-name">Arial</xsl:attribute>
  <xsl:attribute name="font-size">14px</xsl:attribute>
  <xsl:attribute name="font-weight">bold</xsl:attribute>
</xsl:attribute-set>
```

Как применить набор атрибутов к элементу:

```
<p xsl:use-attribute-set="font">
This is a paragraph.
</p>
```

xsl:call-template

Позволяет вызывать шаблон, обладающий собственным именем. Элемент должен содержать 0 или несколько дочерних элементов <xsl:with-param>. Элемент <xsl:call-template> всегда используется внутри тела шаблона.

Синтаксис

```
<xsl:call-template name="templatename">
  <xsl:with-param>..
</xsl:call-template>
```

Атрибуты

Атрибут	Значение	Описание
name	templatename	Обязателен. Имя вызываемого шаблона

Примеры

```
<xsl:template match="ol">
  <xsl:call-template name="orderedlist"/>
</xsl:template>
```

xsl:choose

Элемент <xsl:choose> применяется для реализации выбора между несколькими альтернативами на основе проверки условия. Он содержит один и более элементов <xsl:when>, за которыми может следовать необязательный элемент <xsl:otherwise>. Элемент <xsl:choose> применяется всегда внутри тела шаблона.

Синтаксис

```
<xsl:choose>
  <xsl:when>..
  <xsl:otherwise>..
</xsl:choose>
```

Примеры

Объявляется переменная color (цвет), ее значение приравнивается атрибуту color текущего элемента. Если у текущего элемента нет атрибута color, тогда значение переменной будет приравнено red (красный):

```
<xsl:variable name="color">
```

```
  <xsl:choose>
    <xsl:when test="@color">
      <xsl:value-of select="@color"/>
    </xsl:when>
    <xsl:otherwise>red</xsl:otherwise>
  </xsl:choose>
</xsl:variable>
```

xsl:comment

Создает XML-комментарий. Всегда используется внутри тела шаблона.

Синтаксис

```
<xsl:comment>
  template-body
</xsl:comment>
```

Примеры

```
<xsl:comment>This is a comment!</xsl:comment>
```

xsl:copy

Копирует текущий узел в выходной поток. В отличие от элемента <xsl:value-of>, она предварительно не конвертирует его в строку. Дочерние узлы и атрибуты не копируются, а пространство текущего узла копируется.

Синтаксис

```
<xsl:copy use-attribute-sets="name-list">
 template-body
</xsl:copy>
```

Атрибуты

Атрибут	Значение	Описание
use-attribute-sets	name-list	Не обязателен. Список (с разделителями-пробелами) других наборов атрибутов для использования в этом узле. Этот атрибут применяется только при копировании элементных узлов.

Примеры

В этом примере делается копирование узла p в XML-документе в текущий выходной поток:

```
<xsl:template match="p">
  <xsl:copy>
    <xsl:apply-templates/>
  </xsl:copy>
</xsl:template>
```

xsl:copy-of

Копирует текущий узел в выходной поток (с дочерними узлами и атрибутами) без предварительного конвертирования их в строку, как это делает элемент <xsl:value-of>. Данный элемент применяется тогда, когда одни и те же данные нужны в нескольких местах выходного потока и когда нужно скопировать без изменений под-дерево из входного документа в выходной.

Синтаксис

```
<xsl:copy-of select="expression"/>
```

Атрибуты

Атрибут	Значение	Описание
select	expression	Обязателен. Элемент, который копируется в выходной поток.

Примеры

В этом примере делается копирование узла p (со всеми дочерними узлами и атрибутами) в текущий выходной поток:

```
<xsl:template match="p">
  <xsl:copy-of select="node()|@*"/>
  <xsl:apply-templates/>
</xsl:template>
```

В этом примере делается копирование узла p (со всеми дочерними узлами b) в текущий выходной поток:

```
<xsl:template match="p">
  <xsl:copy-of select="b"/>
  <xsl:apply-templates/>
</xsl:template>
```

xsl:decimal-format

Задает знак/строку предназначенную для использования при конвертации чисел в строки с форматно-числовой функцией. Не везде для разделения десятичной части от целой части чисел и для группирования цифр используются одни и те же символы. Этот элемент позволяет изменить эти специальные символы на другие символы. Это элемент верхнего уровня. Форматно-числовая функция сожет ссылаться на элемент <xsl:decimal-format> по имени.

Синтаксис

```
<xsl:decimal-format
name="name"
decimal-separator="char"
grouping-separator="char"
infinity="string"
minus-sign="char"
NaN="string"
percent="char"
per-mille="char"
zero-digit="char"
digit="char"
pattern-separator="char"/>
```

Атрибуты

Атрибут	Значение	Описание

name	name	Не обязателен. Имя десятичного формата
decimal-separator	char	Не обязателен. Символ, разделяющий целую и десятичную часть числа. По умолчанию "."
grouping-separator	char	Не обязателен. Символ для группировки цифр в числе. По умолчанию ","
infinity	string	Не обязателен. Строка, представляющая бесконечность. По умолчанию "infinity"
minus-sign	char	Не обязателен. Символ, представляющий отрицательные числа. По умолчанию "-"
NaN	string	Не обязателен. Строка, представляющая сообщение "Не число". По умолчанию "NaN"
percent	char	Не обязателен. Символ-знак процентов. По умолчанию "%"
per-mille	char	Не обязателен. Символ-знак промиллей. По умолчанию "‰"
zero-digit	char	Не обязателен. Символ, отмечающий место, где требуется впереди идущий знак нуля. По умолчанию "0".
digit	char	Не обязателен. Символ, отмечающий место, где требуется цифра. По умолчанию "#".
pattern-separator	char	Не обязателен. Символ, разделяющий форматные строки. По умолчанию ";"

Примеры

Третий аргумент форматно-числовой функции ссылается на формат, заданный именным атрибутом в элементе xsl:decimal-format:

```
<xsl:decimal-format name="eurofrmt"
decimal-separator="," grouping-separator="."/>

<xsl:template match="/">
<xsl:value-of
select="format-number(26825.8, '#.###,00', 'eurofrmt')"/>
</xsl:template>
```

Получаем на выходе:

```
26.825,80
```

xsl:element

Добавляет новый элементный узел в выходной поток. Этот элемент применяется, когда нужно создать элементы, у которых имя задается "на лету"

Синтаксис

```
<xsl:element name="name" namespace="uri"
use-attribute-sets="namelist">
  template-body
</xsl:element>
```

Атрибуты

Атрибут	Значение	Описание

name	name	Обязателен. Имя создаваемого элемента
namespace	uri	Не обязателен. Uri пространства имен создаваемого элемента
use-attribute-sets	namelist	Не обязателен. Список (разделители - пробелы) наборов атрибутов, в которых содержатся атрибуты, добавляемые к создаваемому элементу

Примеры

В этом примере выбираются атрибуты textfrmt каждого элемента в дереве узлов message и для каждого создается элемент, чье имя одинаково со значением этого атрибута:

XML-файл:

```
<?xml version="1.0"?>
<message>
  <to textfrmt="i">Tove</to>
  <from textfrmt="b">Jani</from>
  <text textfrmt="u">Hello</text>
</message>
```

XSL-файл:

```
<xsl:template match="/">
  <xsl:for-each select="message">
    <xsl:element name="{@textfrmt}">
      <xsl:value-of select="."/><br />
    </xsl:element>
  </xsl:for-each>
</xsl:template>
```

В результате имя Tove будет выведено курсивом, имя Jani - полужирным шрифтом, а Hello будет подчеркуто.

xsl:fallback

Позволяет определить альтернативу в случае невыполнимых инструкций. Если в таблице стилей имеется XSL-элемент, который не может быть выполнен XSL-процессором, этот элемент показывает, что с ним делать

Синтаксис

```
<xsl:fallback>
  template-body
</xsl:fallback>
```

Примеры

В этом примере создан цикл через каждый элемент узла message с придуманного элемента xsl:loop. Если XSL-процессор не поддерживает этот элемент (а он его не поддерживает), он будет вместо него использовать элемент xsl:for-each:

```
<xsl:template match="message">
  <xsl:loop select="message">
    <xsl:fallback>
      <xsl:for-each select="message">
        <xsl:value-of select="."/>
      <xsl:/for-each>
    </xsl:fallback>
    <xsl:value-of select="."/>
  </xsl:loop>
```

```
</xsl:template>
```

xsl:for-each

Позволяет создать цикл из нескольких узлов в выходном потоке

Синтаксис

```
<xsl:for-each select="expression">
  <xsl:sort>
  template-body
</xsl:for-each>
```

Атрибуты

Атрибут	Значение	Описание
select	expression	Обязателен. Обрабатываемый набор узлов

Примеры

В этом примере элемент xsl:for-each используется для создания цикла через каждый элемент дерева узлов CD. При этом для вписывания в выходной поток элементов title и artist используется элемент xsl:value-of:

```
<xsl:template match="/">
    <table>
      <xsl:for-each select="CATALOG/CD">
      <tr>
        <td><xsl:value-of select="TITLE"/></td>
        <td><xsl:value-of select="ARTIST"/></td>
      </tr>
      </xsl:for-each>
    </table>
</xsl:template>
```

Этот пример аналогичен предыдущему, но выходные записи отсортированы по имени исполнителя (ARTIST):

```
<xsl:template match="/">
  <table>
    <xsl:for-each select="CATALOG/CD">
      <xsl:sort select="ARTIST" order="ascending"/>
      <tr>
        <td><xsl:value-of select="TITLE"/></td>
        <td><xsl:value-of select="ARTIST"/></td>
      </tr>
    </xsl:for-each>
  </table>
</xsl:template>
```

xsl:if

Позволяет писать условные выражения. Этот элемент несет шаблон, который будет обрабатываться при выполнении определенного условия

Синтаксис

```
<xsl:if test="expression">
```

```
  template-body
</xsl:if>
```

Атрибуты

Атрибут	Значение	Описание
test	expression	Обязателен. Проверяемое условие

Примеры

Элемент xsl:if проверяет, имеется ли какой-нибудь элемент artist, чье значение равно "Bob Dylan". Если это условие выполняется, значение узла title и значение узла artist вписывается в выходной поток:

```
<xsl:template match="/">
  <xsl:for-each select="CATALOG/CD">
    <xsl:if test="ARTIST='Bob Dylan'">
      <xsl:value-of select="TITLE"/><br />
      <xsl:value-of select="ARTIST"/>
    </xsl:if>
  </xsl:for-each>
</xsl:template>
```

Результат на выходе:

```
Empire Burlesque
Bob Dylan
```

xsl:import

Импортирует в одну таблицу стилей другую таблицу стилей

Элемент <xsl:import> - элемент верхнего уровня и должен появляться как первый дочерний узел элемента <xsl:stylesheet>

Синтаксис

```
<xsl:import href="uri"/>
```

Атрибуты

Атрибут	Значение	Описание
href	uri	Обязателен. Uri-адрес импортируемой таблицы стилей

Примеры

Допустим, у вас есть таблица стилей, которая называется standard.xsl. Эта таблица стилей содержит общий шаблон для отображения узла message:

```
<xsl:template match="message">
  <xsl:value-of select="to"><br />
  <xsl:value-of select="from"><br />
  <xsl:value-of select="text">
</xsl:template>
```

Вторая таблица стилей, которая называется special.xsl, импортирует таблицу стилей standard.xsl и расставляет элементы h2 вокруг элементов to, from и text:

```
<xsl:import href="standard.xsl"/>
<xsl:template match="message">
  <h2>
    <xsl:apply-imports/>
```

```
  </h2>
</xsl:template>
```

xsl:include

Делает включение одной таблицы стилей в другую

Элемент <xsl:include> - элемент верхнего уровня и должен появляться как первый дочерний узел элемента <xsl:stylesheet>

Синтаксис

```
<xsl:include href="uri"/>
```

Атрибуты

Атрибут	Значение	Описание
href	uri	Обязателен. Uri-адрес включаемой таблицы стилей

Примеры

Допустим, у вас есть таблица стилей, которая называется attr.xsl. Эта таблица стилей содержит следующий набор атрибутов:

```
<xsl:attribute-set name="font">
  <xsl:attribute name="font-name">Arial</xsl:attribute>
  <xsl:attribute name="font-size">14px</xsl:attribute>
  <xsl:attribute name="font-weight">bold</xsl:attribute>
</xsl:attribute-set>
```

Вторая таблица стилей, которая называется note.xsl, использует набор атрибутов ("font") из подключаемого файла attr.xsl:

```
<xsl:include href="attr.xsl"/>
<p xsl:use-attribute-set="font">
This is a paragraph.
</p>
```

xsl:key

Позволяет задать ключ. Этот элемент используется с функцией key()

Элемент <xsl:key> - элемент верхнего уровня и должен появляться как первый дочерний узел элемента <xsl:stylesheet>

Синтаксис

```
<xsl:key name="name" match="pattern" use="expression"/>
```

Атрибуты

Атрибут	Значение	Описание
name	name	Обязателен. Имя ключа
match	pattern	Обязателен. Применимые узлы для этого ключа
use	expression	Обязателен. Значение ключа для каждого применимого узла

Примеры

Допустим, у вас есть XML-файл, который называется persons.xml:

```
<persons>
  <person name="Tarzan" id="050676"/>
  <person name="Donald" id="070754"/>
  <person name="Dolly" id="231256"/>
</persons>
```

Вот так вы можете здать ключ в XSL-файле:

```
<xsl:key name="personreg" match="person" use="@id"/>
```

Чтобы найти элемент person, чье id имеет значение "050676", нужно написать:

```
<xsl:apply-templates select="key('personreg','050676')"/>
```

xsl:message

Пишет сообщение в выходном потоке. Этот элемент используется для сообщения об ошибках

Синтаксис

```
<xsl:message terminate="yes|no">
  template-body
</xsl:message>
```

Атрибуты

Атрибут	Значение	Описание
terminate	yes/no	Не обязателен. "Yes" прерывает обработку после того, как сообщение выведено в выходном потоке. "No" продолжает обработку после выведения сообщения. По умолчанию "no"

Примеры

Объявляется переменная color, ее значение устанавливается равным значению атрибута color текущего элемента. Если текущий элемент не имеет атрибута color, на выходе будет выведено сообщение:

```
<xsl:variable name="color">
  <xsl:choose>
    <xsl:when test="@color">
      <xsl:value-of select="@color"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:message>
        <xsl:text>
        The element has no color attribute!
        </xsl:text>
      </xsl:message>
    </xsl:otherwise>
  </xsl:choose>
</xsl:variable>
```

xsl:namespace-alias

Позволяет отобразить пространство имен в таблице стилей в пространство имен в выходном потоке

Элемент <xsl:namespace-alias> - элемент верхнего уровня и должен появляться как первый дочерний узел элемента <xsl:stylesheet>

Синтаксис

```
<xsl:namespace-alias
stylesheet-prefix="ncname"
result-prefix="ncname"/>
```

Атрибуты

Атрибут	Значение	Описание
stylesheet-prefix	ncname #default	Обязателен. Префикс пространства имен, используемого в таблице стилей
result-prefix	ncname #default	Обязателен. Префикс пространства имен, используемого в выходном потоке

xsl:number

Пишет форматированное число в выходном потоке

Синтаксис

```
<xsl:number
level="single|multiple|any"
count="pattern"
from="pattern"
value="expression"
format="{formatstring}"
lang="{languagecode}"
letter-value="{alphabetic|traditional}"
grouping-separator="{character}"
grouping-size="{number}"/>
```

Атрибуты

Атрибут	Значение	Описание
level	single / multiple / any	Не обязателен. Уровень, на котором ведется счет
count	pattern	Не обязателен. Элементы, которые должны быть подсчитаны
from	pattern	Не обязателен. Место начала счета
value	expression	Не обязателен. Число, преобразуемое в текст
format	{formatstring}	Не обязателен. Выходной формат числа
lang	{languagecode}	Не обязателен. Используемый для конверсии язык
letter-value	{alphabetic} / {traditional}	Не обязателен. Буквенное значение для нумерации схем
grouping-separator	{character}	Не обязателен. Символ для разделения групп цифр
grouping-size	{number}	Не обязателен. Число цифр в каждой группе. Показывает где должен быть вставлен grouping-separator (разделитель группирования)

Примеры

```
<xsl:number value='250000' grouping-size='3'
grouping-separator='.'/>
```

Результат:

```
250.000
```

```
<xsl:number value='250000' grouping-size='2'
grouping-separator=','/>
```

Результат:

```
25,00,00
```

```
<xsl:number value='12' grouping-size='1'
grouping-separator='#' format="I"/>
```

Результат:

```
X#I#I
```

xsl:otherwise

Определяет, что происходит, когда не выполняется ни один элемент <xsl:when> внутри элемента <xsl:choose>

Синтаксис

```
<xsl:otherwise>
  template-body
</xsl:otherwise>
```

Примеры

Объявляется переменная color, ее значение устанавливается равным значению атрибута color текущего элемента. Если текущий элемент не имеет атрибута color, значение переменной будет приравнено red:

```
<xsl:variable name="color">
  <xsl:choose>
    <xsl:when test="@color">
      <xsl:value-of select="@color"/>
    </xsl:when>
    <xsl:otherwise>red</xsl:otherwise>
  </xsl:choose>
</xsl:variable>
```

xsl:output

Позволяет контролировать формат выходного потока данной таблицы стилей

Элемент <xsl:output> - элемент верхнего уровня и должен появляться как первый дочерний узел элемента <xsl:stylesheet>

Синтаксис

```
<xsl:output
method="xml|html|text|name"
version="version"
encoding="text"
omit-xml-declaration="yes|no"
standalone="yes|no"
doctype-public="text"
doctype-system="text"
```

```
cdata-section-elements="namelist"
indent="yes|no"
media-type="mimetype"/>
```

Атрибуты

Атрибут	Значение	Описание
method	xml / html / text / name	Не обязателен. Выходной формат
version	version	Не обязателен. XML-версия выходного формата
encoding	text	Не обязателен. Кодировка
omit-xml-declaration	yes / no	Не обязателен. "Yes" указывает, что XML-декларация (?xml... ?) в выходном потоке может быть опущена. "No" указывает, что декларация должна быть включена в выходной поток
standalone	yes / no	Не обязателен. "Yes" указывает, что результат должен быть автономным документом. "No" указывает, что результат не должен быть автономным документом.
doctype-public	text	Не обязателен. Идентификатор public, который будет использован в декларации !doctype в выходном документе
doctype-system	text	Не обязателен. Идентификатор system, который будет использован в декларации !doctype в выходном документе
cdata-section-elements	namelist	Не обязателен. Список (разделители - пробелы) элементов, чье содержание в выходном потоке будет в разделах типа CDATA
indent	yes / no	Не обязателен. "Yes" указывает, что выходной поток должен быть с отступами, отражающими иерархическую структуру (для удобочитаемости). "No" указывает, что выходной поток должен быть без отступов.
media-type	mimetype	Не обязателен. Тип носителя выходного потока

Примеры

Выходной документ в этом примере будет XML версии 1.0. Кодировка ISO-8859-1. Для удобства результат будет с отступами:

```
<xsl:output method="xml" version="1.0"
encoding="iso-8859-1" indent="yes"/>
```

Выходной документ в этом примере будет HTML версии 4.0. Кодировка ISO-8859-1. Для удобства результат будет с отступами:

```
<xsl:output method="html" version="4.0"
encoding="iso-8859-1" indent="yes"/>
```

xsl:param

Позволяет задавать параметры

Элемент <xsl:param> может декларироваться как элемент верхнего уровня (дочерний элемент элемента <xsl:stylesheet>) при объявлении глобального параметра. Также он может декларироваться как первый дочерний элемент элемента <xsl:template> при объявлении локального параметра.

Синтаксис

```
<xsl:param name="name" select="expression">
  template-body
</xsl:param>
```

Атрибуты

Атрибут	Значение	Описание
name	name	Обязателен. Имя параметра
select	expression	Не обязателен. Значение параметра

xsl:preserve-space

Элементы <xsl:preserve-space> и <xsl:strip-space> определяют как будут обрабатываться в XML-документе пустые узлы. Пустой узел - это текстовый узел, состоящий только из пустых символов, таких, как пробел, табуляция, возврат каретки и перевод строки. Элемент <xsl:preserve-space> сохраняет пустые узлы. Элемент <xsl:strip-space> удаляет пустые узлы.

Элементы <xsl:preserve-space> и <xsl:strip-space> - элементы верхнего уровня. Сохранение пустых узлов выполняется по умолчанию, поэтому применение элемента <xsl:preserve-space> нужно лишь в тех случаях, когда применялся элемент <xsl:strip-space>

Синтаксис

```
<xsl:preserve-space elements="list"/>
```

Атрибуты

Атрибут	Значение	Описание
elements	list	Обязателен. Список (разделители - пробелы) элементов, в которых должны удаляться или сохраняться пустые элементы. Обратите внимание: список также может содержать "*" и "prefix:*", так что все элементы или все элементы из опредленного именного пространства могут быть соеденены

Примеры

Пробел между </first> и <last> в этом XML-документе - это пустой текстовый узел:

```
<p>
My name is <first>Hege</first> <last>Refsnes</last>
</p>
```

Чтобы удалить пробел, в XSL-документ следует включить элемент:

```
<xsl:strip-space elements="p"/>
```

Эта строка удаляет пустые узлы внутри элемента p, результат будет такой:

```
My name is HegeRefsnes
```

Так удаляются пустые узлы внутри всех элементов:

```
<xsl:strip-space elements="*"/>
```

Так удаляются пустые узлы внутри элемента message:

```
<xsl:strip-space elements="message"/>
```

Так удаляются пустые узлы из всех элементов, кроме узла message:

```
<xsl:strip-space elements="*"/>
```

```
<xsl:preserve-space elements="message"/>
```

xsl:processing-instruction

Вписывает процессуальные инструкции в выходной поток

Синтаксис

```
<xsl:processing-instruction name="name">
  template-body
</xsl:processing-instruction>
```

Атрибуты

Атрибут	Значение	Описание
name	name	Обязателен. Имя процессуальной инструкции

xsl:sort

Позволяет задавать сортировку набора узлов

Элемент <xsl:sort> всегда является дочерним элементом элемента <xsl:for-each> или элемента <xsl:apply-templates>

Синтаксис

```
<xsl:sort
select="expression"
order="ascending|descending"
case-order="upper-first|lower-first"
lang="language-code"
data-type="text|number|qname"/>
```

Атрибуты

Атрибут	Значение	Описание
select	expression	Не обязателен. Сортировочное выражение
order	ascending / descending	Не обязателен. Порядок сортировки. По умолчанию "ascending" - в порядке возрастания.
case-order	upper-first / lower-first	Не обязателен. Большие буквы идут впереди маленьких или наоборот?
lang	language-code	Не обязателен. Язык, применяемый для сортировки.
data-type	text / number / qname	Не обязателен. Тип сортируемых данных. Это текст, числа или определенный пользователем тип данных. По умолчанию "text"

Примеры

В этом примере применяется элемент xsl:for-each для создания петли через каждый элемент дерева узлов CD. Затем применяется элемент xsl:value-of для выведения title и artists в выходной поток. Выходной поток сортируется по содержанию элемента artist:

```
<xsl:template match="/">
  <table>
```

```
    <xsl:for-each select="CATALOG/CD">
      <xsl:sort select="ARTIST" order="ascending"/>
      <tr>
        <td><xsl:value-of select="TITLE"/></td>
        <td><xsl:value-of select="ARTIST"/></td>
      </tr>
    </xsl:for-each>
  </table>
</xsl:template>
```

xsl:strip-space

Элементы <xsl:preserve-space> и <xsl:strip-space> определяют как будут обрабатываться в XML-документе пустые узлы. Пустой узел - это текстовый узел, состоящий только из пустых символов, таких, как пробел, табуляция, возврат каретки и перевод строки. Элемент <xsl:preserve-space> сохраняет пустые узлы. Элемент <xsl:strip-space> удаляет пустые узлы.

Элементы <xsl:preserve-space> и <xsl:strip-space> - элементы верхнего уровня. Сохранение пустых узлов выполняется по умолчанию, поэтому применение элемента <xsl:preserve-space> нужно лишь в тех случаях, когда применялся элемент <xsl:strip-space>

Синтаксис

```
<xsl:strip-space elements="list"/>
```

Атрибуты

Атрибут	Значение	Описание
elements	list	Обязателен. Список (разделители - пробелы) элементов, в которых должны удаляться или сохраняться пустые элементы. Обратите внимание: список также может содержать "*" и "prefix:*", так что все элементы или все элементы из опредленного пространства имен могут быть соеденены

Примеры

Пробел между </first> и <last> в этом XML-документе - это пустой текстовый узел:

```
<p>
My name is <first>Hege</first> <last>Refsnes</last>
</p>
```

Чтобы удалить пробел, в XSL-документ следует включить элемент:

```
<xsl:strip-space elements="p"/>
```

Эта строка удаляет пустые узлы внутри элемента p, результат будет такой:

```
My name is HegeRefsnes
```

Так удаляются пустые узлы внутри всех элементов:

```
<xsl:strip-space elements="*"/>
```

Так удаляются пустые узлы внутри элемента message:

```
<xsl:strip-space elements="message"/>
```

Так удаляются пустые узлы из всех элементов, кроме узла message:

```
<xsl:strip-space elements="*"/>
<xsl:preserve-space elements="message"/>
```

xsl:stylesheet

Задает корневой элемент таблицы стилей

Элементы <xsl:stylesheet> и <xsl:transform> являются полными аналогами

Синтаксис

```
<xsl:stylesheet
id="name"
version="version"
extension-element-prefixes="list"
exclude-result-prefixes="list">
  top-level-elements
</xsl:stylesheet>
```

Атрибуты

Атрибут	Значение	Описание
version	version	Обязателен. XSLT-версия таблицы стилей
extension-element-prefixes	list	Не обязателен. Список (разделители - пробелы) префиксов пространств имен, используемых для расширения элементов
exclude-result-prefixes	list	Не обязателен. Список (разделители - пробелы) префиксов пространств имен, которые не должны помещаться в выходное дерево
id	name	Не обязателен. Уникальный идентификатор таблицы стилей

Примеры

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

xsl:template

Задает шаблон для выходного потока

Это элемент верхнего уровня

Синтаксис

```
<xsl:template
name="name"
match="pattern"
mode="mode"
priority="number">
  <xsl:param>
  template-body
</xsl:template>
```

Атрибуты

Атрибут	Значение	Описание
name	name	Не обязателен. Имя шаблона. Если этот атрибут опускается должен быть атрибут match
match	pattern	Не обязателен. Совпадающая конфигурация для данного шаблона. Если этот атрибут опускается должен быть атрибут name
mode	mode	Не обязателен. Режим данного шаблона
priority	number	Не обязателен. Число, показывающее приоритет данного шаблона

Примеры

Это пример простой таблицы стилей, которая преобразует элементы и <paragraph> в выходной поток:

```
<xsl:template match="strong">
  <b><xsl:apply-templates/></b>
</xsl:template>

<xsl:template match="paragraph">
  <p><xsl:apply-templates/></p>
</xsl:template>
```

xsl:text

Вписывает текст в выходной поток

Синтаксис

```
<xsl:text
disable-output-escaping="yes|no">
  text
</xsl:text
```

Атрибуты

Атрибут	Значение	Описание
disable-output-escaping	yes / no	Не обязателен. "Yes" указывает, что специальные символы (например, знак меньше) должны выводиться так, как они есть. "No" показывает, что специальные символы (например,знак меньше) должны выводиться как "%lt". По умолчанию "no"

Примеры

Объявляется переменная color, ее значение устанавливается равным значению атрибута color текущего элемента. Если текущий элемент не имеет атрибута color, на выходе будет выведено сообщение:

```
<xsl:variable name="color">
  <xsl:choose>
    <xsl:when test="@color">
      <xsl:value-of select="@color"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:message>
```

```
        <xsl:text>
        The element has no color attribute!
        </xsl:text>
      </xsl:message>
    </xsl:otherwise>
  </xsl:choose>
</xsl:variable>
```

В этом примере создается параграф (paragraph) вокруг элемента car, у котрого есть дочерние элементы year и color. На выходе параграф будет содержать значение дочернего элемента year, за ним пробел, а затем значение дочернего элемента color:

```
<xsl:template match="car">
  <p>
    <xsl:value-of select="year"/>
    <xsl:text> </xsl:text>
    <xsl:value-of select="color"/>
  </p>
</xsl:template>
```

В этом примере создается параграф (paragraph) вокруг элемента car, у котрого есть атрибуты year и color. На выходе параграф будет содержать значение атрибута year, за ним пробел, а затем значение атрибута color:

```
<xsl:template match="car">
  <p>
    <xsl:value-of select="@year"/>
    <xsl:text> </xsl:text>
    <xsl:value-of select="@color"/>
  </p>
</xsl:template>
```

В этом примере создается параграф (paragraph) вокруг элемента car, у котрого есть атрибуты year и color. На выходе параграф будет содержать значение атрибута year, за ним точка, а затем значение атрибута color:

```
<xsl:template match="car">
  <p>
    <xsl:value-of select="@year"/>
    <xsl:text>.</xsl:text>
    <xsl:value-of select="@color"/>
  </p>
</xsl:template>
```

xsl:transform

Задает корневой элемент таблицы стилей

Элементы <xsl:stylesheet> и <xsl:transform> являются полными аналогами

Синтаксис

```
<xsl:transform
id="name"
version="version"
extension-element-prefixes="list"
exclude-result-prefixes="list">
  top-level-elements
</xsl:transform>
```

Атрибуты

Атрибут	Значение	Описание
version	version	Обязателен. XSLT-версия таблицы стилей
extension-element-prefixes	list	Не обязателен. Список (разделители - пробелы) префиксов пространств имен, используемых для расширения элементов
exclude-result-prefixes	list	Не обязателен. Список (разделители - пробелы) префиксов пространств имен, которые не должны помещаться в выходное дерево
id	name	Не обязателен. Уникальный идентификатор таблицы стилей

Примеры

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

xsl:value-of

Создает текстовый узел и вставляет некоторое значение в дерево-результат

Синтаксис

```
<xsl:value-of
select="expression"
disable-output-escaping="yes|no"/>
```

Атрибуты

Атрибут	Значение	Описание
select	expression	Обязателен. Значение, передаваемое на выход
disable-output-escaping	yes/no	Не обязателен. "Yes" показывает, что специальные символы (например, "<") должные передаваться на выход так, как есть. "No" показывает, что специальные символы (например, "<")должны передаваться на выход как "%lt". По умолчанию "no".

Примеры

В этом примере создается параграф вокруг элемента car, который содержит дочерний элемент color. На выходе параграф будет содержать значение дочернего элемента color:

```
<xsl:template match="car">
  <p>
    <xsl:value-of select="color"/>
  </p>
</xsl:template>
```

В этом примере строковое значение текущего узла вписывается в ывходной поток:

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

В этом примере в выходной поток записывается число 10:

```
<xsl:value-of select="5+5"/>
```

В этом примере в выходной поток вписывается значение первого дочернего элемента <book>:

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

В этом примере в выходной поток вписывается значение переменной book:

```
<xsl:value-of select="{$book}"/>
```

xsl:variable

Позволяет объявить переменную

Синтаксис

```
<xsl:variable name="name" select="expression">
  template-body
</xsl:variable>
```

Атрибуты

Атрибут	Значение	Описание
name	name	Обязателен. Имя переменной
select	expression	Не обязателен. Значение переменной

Примеры

В этом примере переменной "car" присваивается значение "Audi":

```
<xsl:variable name="car" select="'Audi'"/>
```

В этом примере элемент <xsl:variable> пуст и не имеет атрибута select. Это означает, что значением переменной является пустая строка:

```
<xsl:variable name="j"/>
```

xsl:when

Задает условие и выполняет действие, если оно выполняется. Этот элемент всегда является дочерним элементом элемента <xsl:choose>

Синтаксис

```
<xsl:when test="expression">
  template-body
</xsl:when>
```

Атрибуты

Атрибут	Значение	Описание
test	expression	Обязателен. Проверяемое условие

Примеры

В этом примере объявляется переменная color и ее значение устанавливается равным атрибуту color текущего элемента. Если у текущего элемента нет атрибута color, значение переменной будет приравнено red:

```
<xsl:variable name="color">
  <xsl:choose>
    <xsl:when test="@color">
      <xsl:value-of select="@color"/>
    </xsl:when>
    <xsl:otherwise>red</xsl:otherwise>
  </xsl:choose>
```

```
</xsl:variable>
```

xsl:with-param

Этот элемент задает значение параметров при вызове шаблона. Элемент <xsl:with-param> должен являться первым дочерним элементом элементов <xsl:apply-templates>, <xsl:call-template>, или <xsl:apply-import>

Синтаксис

```
<xsl:with-param name="name" select="expression">
  template-body
</xsl:with-param>
```

Атрибуты

Атрибут	Значение	Описание
name	name	Обязателен. Имя параметра
select	expression	Не обязателен. Значение параметра

Примеры

В этом примере при обработке узла warning вызывается шаблон, имеющий имя message. Мы хотим передать параметр шаблону элемента message, это достигается использованием элемента <xsl:with-param>. Имя параметра - "user" и значение этого параметра установлено равным значению переменной "user":

```
<xsl:template name="warning">
  <xsl:call-template name="message"/>
    <xsl:with-param name="user" select="$user"/>
  </xsl:call-template>
</xsl:template>
```