(c)2002-2022 И.Б.Государев



# Введение в XSLT, XPath

## Сущность XML-стилей

Обобщенному языку разметки соответствует обобщенный язык стилей.
XML Stylesheet Language = XSL.
XSL состоит из трех языков: XSLT, XPath и XSL-FO

- XSLT (XSL Transformations) – язык для описания преобразований XML-элементов;
- XPath используется для адресации элементов в документе и представляет собой важнейший инструмент;
- XSL-FO описывает форматирование данных XML для вывода в разные виды сред

Можно рассмотреть XSLT как язык программирования. В нем присутствуют:

- типы данных (Булев, числовой, строковый, массив узлов);
- набор операций (применение шаблона, сортировка и т.д.);
- средства управления ходом выполнения (условный оператор и оператор выбора, итерации и т.д.)

В то же время XSLT отличается от императивного/процедурно-ориентированного языка программирования, в котором задан жесткий порядок выполнения команд. Скорее он может быть отнесен к декларативно-логическим языкам, в которых программа основана на наборе правил.

Цель XSLT – извлечение данных из XML-документа и представление их в новой форме, т.е. преобразование. Правила определяют, что именно сопоставляется исходной разметке при преобразовании, и их можно сравнить с запросами к базе данных, учитывая, что в предыдущей работе схемы XML сравнивались со схемами данных.

Таблица стилей XSL состоит из одного или нескольких шаблонов, т.е. структурной единицей таблицы стилей является шаблон.
Шаблон (template) – это набор правил преобразования исходной XML-разметки. Содержащиеся в шаблоне правила должны выполняться, когда происходит совпадение с указанным элементом (узлом). Разбор начинается с шаблона, соответствующего корневому элементу.

Рассмотрим документ с таблицей из предыдущей работы.

```
<?xml version="1.0" encoding="utf-8" ?>
<таблица xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="1table.xsd" >
      <студент имя = "Иванов">
            <отметка дисциплина="Мультимедиа">3</отметка>
            <отметка дисциплина="Веб-дизайн">3</отметка>
            <отметка дисциплина="Графика">5</отметка>
            <отметка дисциплина="Логика">3</отметка>
      </студент>
      <студент имя = "Петров">
            <отметка дисциплина="Мультимедиа">3</отметка>
            <отметка дисциплина="Веб-дизайн">4</отметка>
            <отметка дисциплина="Графика">5</отметка>
            <отметка дисциплина="Логика">5</отметка>
      </студент>
      <студент имя = "Сидоров">
            <отметка дисциплина="Мультимедиа">3</отметка>
            <отметка дисциплина="Веб-дизайн">4</отметка>
            <отметка дисциплина="Графика">5</отметка>
            <отметка дисциплина="Логика">4</отметка>
      </студент>
</таблица>
```

Таблица содержит элементы «студент», а каждый элемент «студент» содержит четыре элемента «отметка». Требуется вывести содержимое на экран в виде таблицы

| | Мультимедиа | Веб-дизайн | Графика | Логика |
|---|---|---|---|---|
| Иванов | | | | |
| Петров | | | | |
| Сидоров | | | | |

Для просмотра исходной структуры, очевидно, работает итеративный алгоритм: для каждого элемента «студент» создать строку таблицы, в которой для каждого элемента «отметка» создать ячейку.

Для этой цели мы можем воспользоваться двумя подходами:

- создать шаблон для корневого элемента (основной) и шаблон для элемента «студент», после чего включить ссылку на него в основной шаблон (apply-templates)
- создать шаблон для корневого документа и реализовать перебор его дочерних элементов (for-each)

В следующем разделе рассматриваются оба подхода.

(c)2002-2022 И.Б.Государев

## Создание стилей

Для начала решим задачу в простом варианте. Будем выводить не всю таблицу, а извлечем только отметки по веб-дизайну.

Строка <?xml-stylesheet type="text/xsl" href="1.xsl"?> в файле XML присоединяет таблицу.

Рассмотрим подробнее текст таблицы (1.xsl).

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">


 <xsl:template match="/">
   <html><head><title></title></head>
   <body style="background-color:gold">
       <xsl:for-each select="таблица/студент">

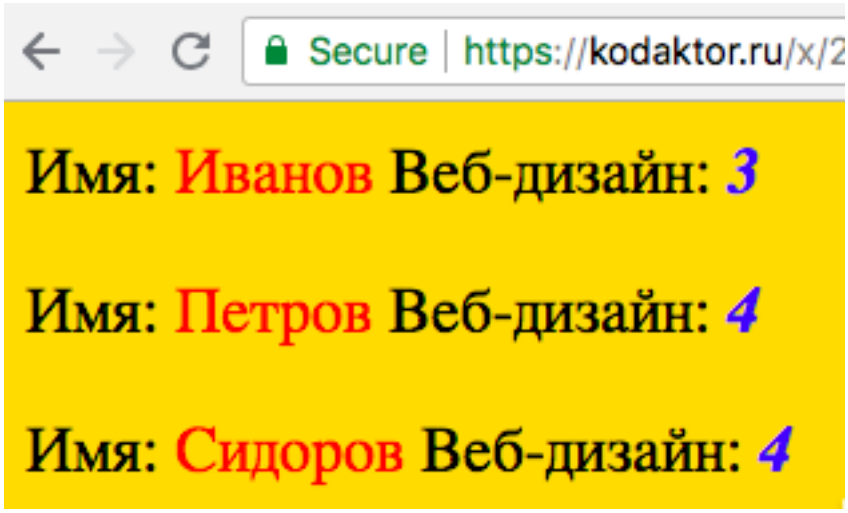
                  <p>Имя: <span style="color:red"><xsl:value-of select="@имя"/></span>
  Веб-дизайн: <strong><em style="color:blue"><xsl:value-of select="отметка[2]"/></em></strong></p>

      </xsl:for-each>
  </body></html>
 </xsl:template>

</xsl:stylesheet>
```

На первом уровне вложения (в корневой элемент *xsl:stylesheet*) находится только один элемент-шаблон *xsl:template* с атрибутом match. Этот атрибут содержит выражение XPath (см. соответствующий раздел) и служит для привязки шаблона к исходной разметке. В данном конкретном случае выражение «/» означает привязку к корневому элементу исходного документа, т.е., очевидно, «таблица».

В шаблон вложена HTML-разметка, которая должна быть подставлена в результирующий документ.



Элемент *xsl:for-each* реализует итеративный перебор всех элементов, на которые указывает атрибут *select*, также содержащий выражение XPath. В данном случае «таблица/студент» означает, что берется элемент «студент», находящийся в элементе «таблица».

Для извлечения конкретных значений конкретных элементов/атрибутов используется элемент *xsl:value-of* – он тоже имеет атрибут *select*, содержащий выражение XPath. В данном случае «@имя» означает, что берется значение атрибута «имя» текущего элемента (т.е. одного из элементов «студент»), а «отметка[2]» означает, что берется второй (из четырех) элементов «отметка». Именно он содержит отметку по веб-дизайну.

Вот какой код образуется в результате работы xsl:for-each

```
<p>Имя: <span style="color: red">Иванов</span> Веб-дизайн: <strong><em style="color: blue">2</em></strong></p>
<p>Имя: <span style="color: red">Петров</span> Веб-дизайн: <strong><em style="color: blue">3</em></strong></p>
<p>Имя: <span style="color: red">Сидоров</span> Веб-дизайн: <strong><em style="color: blue">5</em></strong></p>
```

Теперь приведем пример реализации второго подхода.

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
          <html><head><title></title></head>
                  <body style="background-color:gold">
                          <xsl:apply-templates/>
                  </body></html>
  </xsl:template>
  <xsl:template match="студент">
          <p>Имя: <span style="color:red"><xsl:value-of select="@имя"/></span>
  Веб-дизайн: <strong><em style="color:blue"><xsl:value-of select="отметка[2]"/></em></strong></p>
  </xsl:template>
</xsl:stylesheet>
```

Все то же самое, только отдельно реализован шаблон для элемента «студент», команда на применение которого включена в шаблон для корневого элемента.

Трудно не заметить сходство с устройством схем. Как и в случае с объявлениями глобальных типов, описание шаблонов за пределами шаблона корневого элемента обеспечивает большую структурированность, лучшую читаемость, чем описание всех преобразований в одном шаблоне, напоминающем схему-«матрешку».

Любая система управления базами данных должна обеспечить такие операции как сортировка и фильтрация. Сортировка – это упорядочение по определенному параметру, а фильтрация – отбор по определенному критерию (логическому условию). Рассмотрим это подробнее.

**Сортировка**

Задается с помощью элемента sort; в нашем случае он должен быть добавлен в элемент for-each:

```
<xsl:sort select="@имя"  order="descending"/>
```

Указывая order=”descending” мы устанавливаем нисходящую сортировку, или сортировку по убыванию.

В частности, как показано на рисунке ниже, фамилии студентов будут выведены в порядке, обратном алфавитному.

**Логические условия**

Элемент *if* позволяет выполнить те или иные действия, если указанное условие истинно. Условие указывается в его атрибуте *test*. Пусть, например, нам нужно вывести слово «Исправить» напротив отметки ниже четырех.

Для этого нам нужно добавить в элемент for-each следующий фрагмент:

```
<xsl:if test="отметка[2] &lt;4">
  Исправить!
</xsl:if>
```

Мы не можем использовать знак < так как он является началом тега. Поэтому используется именованная сущность <

Для реализации ветвления типа if-then-else нужно использовать элемент *choose*.

Он содержит дочерние элементы

- *when* (с атрибутом *test*), содержащий то, что должно быть подставлено в результирующий документ при выполнении условия…
- *otherwise*, содержащий то, что подставляется в противном случае.

Пример:

```
<xsl:choose>
    <xsl:when test=" отметка[2] &lt; 4">
      Исправить!
    </xsl:when>
    <xsl:otherwise>
      Не исправлять!
    </xsl:otherwise>
</xsl:choose>
```

Предположим, мы хотели бы, чтобы были выведены только фамилии студентов, у которых отметка по веб-дизайну больше тройки.

Тогда наша таблица должна выглядеть следующим образом:

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html><head><title></title></head>
<body style="background-color:gold">
<xsl:for-each select="таблица/студент">
<p>
        <xsl:if test=" отметка[2] &gt; 3">
           Имя: <span style="color:red"><xsl:value-of select="@имя"/></span>
  Веб-дизайн: <strong><em style="color:blue"><xsl:value-of select="отметка[2]"/></em></strong>
      </xsl:if>
</p>
</xsl:for-each>
</body></html>
</xsl:template>
</xsl:stylesheet>
```

**Объявление переменных**

Для объявления переменных используется элемент *variable* с атрибутом *name*.
Пример:

```
<xsl:variable name="ball">3</xsl:variable>
```

Для того, чтобы сослаться на эту переменную, нужно указать ее так: $ball.

Пример:

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:variable name="ball">3</xsl:variable>
<xsl:template match="/">
<html><head><title></title></head>
<body style="background-color:gold">
<xsl:for-each select="таблица/студент">
<p>
        <xsl:if test=" отметка[2] &gt; $ball">
           Имя: <span style="color:red"><xsl:value-of select="@имя"/></span>
  Веб-дизайн: <strong><em style="color:blue"><xsl:value-of select="отметка[2]"/></em></strong>
```

```
        </xsl:if>
    </p>
    </xsl:for-each>
    </body></html>
    </xsl:template>
    </xsl:stylesheet>
```

**Другие возможности**

XSLT предоставляет богатый инструментарий для генерации разметки. Следует отметить, например, возможность генерации элементов и атрибутов с именами, которые определяются в момент исполнения (вычисляемыми именами) – для этого служат элементы таблицы стилей *element* и *attribute*.

Можно включать содержимое внешних файлов, для этого используются элементы *import* и *include*.

Можно даже создавать ассоциативные массивы (элемент *key*).

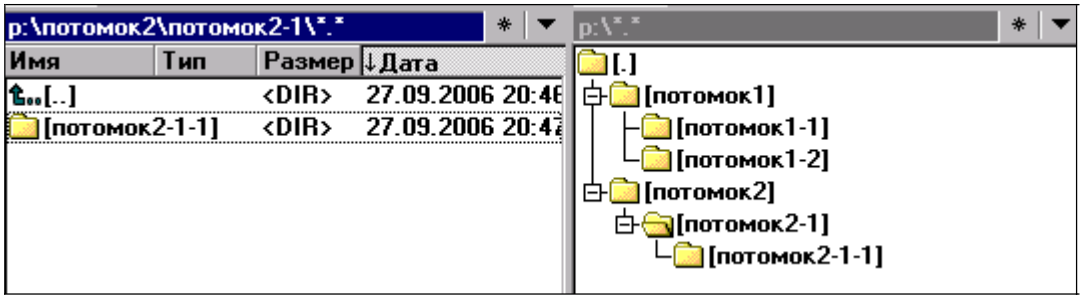
Наконец, можно вызывать шаблоны по имени и встраивать в таблицы сценарии.

Для эффективного использования XSLT нужно иметь представления о диапазоне его возможностей, с тем, чтобы при возникновении соответствующей задачи применить нужный инструмент.

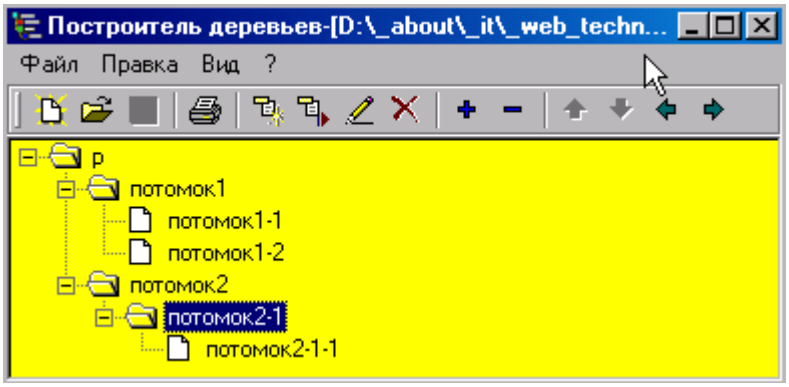
Дополнительные возможности при создании шаблонов предоставляет язык XPath.

## XPath

XPath представляет собой язык, с помощью которого можно описывать пути доступа к частям XML-документа. Синтаксис этого языка не теговый, он напоминает пути к файлам (отсюда название) или URI, что абсолютно оправдано: ведь и файлы с папками, и элементы образуют дерево (ордерево).




```
- <p xsi:noNamespaceSchemaLocation="tree.xsd">
  <потомок уровень="1">
      <потомок уровень="1-1"/>
      <потомок уровень="1-2"/>
    </потомок>
  <потомок уровень="2">
    <потомок уровень="2-1">
        <потомок уровень="2-1-1"/>
      </потомок>
    </потомок>
  </p>
```




На диске путь /потомок1/потомок1-1 определяет файл или папку потомок1-1, находящуюся на диске (в корневой папке) и далее в папке потомок1.

В XML-документе путь /p/потомок1/потомок1-1 определяет элемент потомок1-1, находящийся в корневом элементе и далее в элементе потомок1.

В первом рассмотренном нами примере имелся шаблон, соответствующий корневому элементу, а элемент for-each выбирал всех студентов. Для каждого студента выводилась вторая по счету отметка (по веб-дизайну). Рассмотрим, как можно добиться аналогичного эффекта не с помощью отсчитывания, а задавая путь:

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">


 <xsl:template match="/">
   <html><head><title></title></head>
   <body style="background-color:gold">
      <xsl:for-each select="/таблица/студент/отметка[@дисциплина='Веб-дизайн']">
            <p>Имя: <span style="color:red"><xsl:value-of select="../@имя"/></span>
  Веб-дизайн: <strong><em style="color:blue"><xsl:value-of select="."/></em></strong></p>

      </xsl:for-each>
   </body></html>
 </xsl:template>

</xsl:stylesheet>
```

1
2
3

Выражение (1) описывает все элементы «отметка», находящиеся внутри элементов «студент», находящихся внутри элемента «таблица»; при этом у этих элементов «отметка» значение атрибута «дисциплина» должно быть «Веб-дизайн».

Выражение (2) внутри элемента *for-each* ссылается на родительский по отношению к текущему элементу элемент. Так как *for-each* выбирает отметку, то родительским будет студент. В итоге мы получим значение его атрибута «имя».

Выражение (3) ссылается на текущий элемент. Оно дает значение отметки по веб-дизайну.

Более подробно синтаксис XPath и некоторые важные понятия (контекст, ось и др.) рассматриваются в работе, посвященной DOM.

(c)2002-2022 И.Б.Государев

## Дополнительные опциональные упражнения

**0**. Используя элементы when…otherwise создайте таблицу, в которой фамилия студента с отметкой по веб-дизайну ниже четырех выводится красным цветом, иначе – на красном фоне.

Имя: Иванов Веб-дизайн: *3*

Имя: Петров Веб-дизайн: *4*

Имя: Сидоров Веб-дизайн: *4*

**1.** Создайте таблицу для вывода информации об отметках студентов в виде:

| Поля (дисциплины): / Записи (студенты): | Мультимедиа | Веб-дизайн | Графика | Логика |
|---|---|---|---|---|
| Иванов | 3 | 4 | 5 | 3 |
| Петров | 4 | 5 | 3 | 4 |
| Сидоров | 5 | 3 | 4 | 5 |

**2**. Создайте таблицу, которая выводит из документа «Курсы», созданного в предыдущей работе, названия всех курсов в алфавитном порядке в виде упорядоченного списка

Другие задания могут находиться на странице https://kodaktor.ru/g/xsl_intro и в курсе в Moodle.