Лабораторная работа № 2

Форматирование XML-документов при помощи языка XSL.

Цель: Научиться форматировать XML-документы при помощи языка XSL.

Порядок выполнения:

1. Ознакомиться с основами языка XSL.
2. Выполнить задание согласно варианта.

Язык преобразований XSL преобразует дерево документа XML в новую древовидную структуру. Эта новая древовидная структура представляет другой документ XML, документ HTML или документ в любом другом формате. XSL – это описательный, управляемый событиями, основанный на правилах язык программирования.

Разберем первый пример:

```
<?xml version="1.0" encoding="WINDOWS-1251"?>
<knowledgeDatabase>
<tutorial>
<title>"XSL"</title>
<author>Иванов Иван Иванович </author>
</tutorial>
</knowledgeDatabase>
```

Перейдем к шаблону преобразования – к XSL-файлу. Задача XSL-файла – преобразовать дерево XML-файла в другое дерево, которое, например, будет соответствовать формату HTML и может быть изображено на экране браузера с учетом форматирования, выбора шрифтов и т.п.

Для того, чтобы браузер выполнил необходимое преобразование, нужно в XML-файле указать ссылку на XSL-файл

```
<?xml version="1.0" encoding="WINDOWS-1251"?>
<?xml-stylesheet type='text/xsl' href='1.xsl'?>
```

Рассмотрим теперь текст XSL-файла:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<p><strong><xsl:value-of select="//title"/></strong></p>
<p><xsl:value-of select="//author"/></p>
</xsl:template>
</xsl:stylesheet>
```

Первая строка файла содержит тег элемента *xsl:stylesheet*. Атрибуты элемента - номер версии и ссылка на пространство имен. Эти атрибуты элемента *xsl:stylesheet* являются обязательными. В нашем случае пространство имен - это все имена элементов и их атрибутов, которые могут использоваться в XSL-файле. Для XSL-файлов ссылка на пространство имен является стандартной.

Заметим, что XSL-файл является одной из разновидностей XML-файлов. Он не содержит пользовательских данных, но формат его тот же самый. Файл содержит элемент верхнего уровня *xsl:stylesheet*, а далее идет дерево правил преобразования.

В первом примере мы посмотрели, как с помощью элемента *xsl:value-of* можно вывести в HTML-формате содержание элемента (текст, заключенный между тегами). Теперь мы посмотрим, как при помощи того же самого элемента можно вывести значение атрибута элемента.

Рассмотрим следующий XML-файл 1.xml

```
<?xml version="1.0" encoding="WINDOWS-1251"?>
<?xml-stylesheet type='text/xsl' href='1.xsl'?>
<tutorial>
<dog caption="Собака: " name="Шарик">
<dogInfo weight="18 кг" color="рыжий с черными подпалинами"/>
</dog>
</tutorial>
```

В этом файле информация хранится не в содержании элементов, а в виде значений атрибутов. Файл 1.xsl имеет вид

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<P>
      <B><xsl:value-of select="//dog/@caption"/></B>
      <xsl:value-of select="//dog/@name"/>
      <xsl:value-of select="//dogInfo/@weight"/>
<xsl:value-of select="//dogInfo/@color"/>.
</P>
</xsl:template>
</xsl:stylesheet>
```

Обратите внимание на синтаксис ссылки на атрибут элемента – *//dog/@name*. Имя элемента и имя атрибута разделены парой символов *"/@"*. В остальном синтаксис тот же самый, что и для ссылки на содержание элемента.

Результат имеет следующий вид:

Собака: Шарик. 18 кг, рыжий с черными подпалинами.

До тех пор, пока мы работаем с несколькими реквизитами одного и того же объекта, разницы между XML и HTML практически нет. Однако стоит нам перейти к информации, содержащей несколько строк, как выгоды XML становятся очевид-

ны. Но прежде чем перейти к выгодам, научимся выводить на экран простую таблицу.

Рассмотрим следующий XML-файл – 1.xml. Текст его приведен ниже.

```
<?xml version="1.0" encoding="WINDOWS-1251"?>
<?xml-stylesheet type='text/xsl' href='1.xsl'?>
<tutorial>
<enimals>
 <dogs>
 <dogsCaption>Собаки</dogsCaption>
 <dogsCaptionName>Кличка</dogsCaptionName>
 <dogsCaptionWeight caption="кг">Вес</dogsCaptionWeight>
 <dogsCaptionColor>Цвет</dogsCaptionColor>
 <dog>
  <dogName>Шарик</dogName>
  <dogWeight caption="кг">18</dogWeight>
  <dogColor>рыжий с черными подпалинами</dogColor>
 </dog>
 <dog>
  <dogName>Тузик</dogName>
  <dogWeight caption="кг">10</dogWeight>
  <dogColor>белый с черными пятнами</dogColor>
 </dog>
 <dog>
  <dogName>Бобик</dogName>
  <dogWeight caption="кг">2</dogWeight>
  <dogColor>бело-серый</dogColor>
 </dog>
 <dog>
  <dogName>Трезор</dogName>
  <dogWeight caption="кг">25</dogWeight>
  <dogColor>черный</dogColor>
 </dog>
 </dogs>
</enimals>
</tutorial>
```

Простая таблица

Первый шаг – это добавление шаблона преобразования. Модифицируем наш файл, добавив в него ссылку на шаблон.
В этот файл добавлен шаблон преобразования 1.xsl.

Рассмотрим этот шаблон подробнее:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<table border="1">
<tr bgcolor="#CCCCCC">
<td align="center"><strong><xsl:value-of se-
lect="//dogsCaptionName"/></strong></td>
<td align="center"><strong><xsl:value-of se-
lect="//dogsCaptionWeight"/></strong></td>
<td align="center"><strong><xsl:value-of se-
lect="//dogsCaptionColor"/></strong></td>
</tr>
<xsl:for-each select="tutorial/enimals/dogs/dog">
<tr bgcolor="#F5F5F5">
<td><xsl:value-of select="dogName"/></td>
<td align="right"><xsl:value-of select="dogWeight"/> <xsl:value-of se-
lect="dogWeight/@caption"/></td>
<td><xsl:value-of select="dogColor"/></td>
</tr>
</xsl:for-each>
</table>
</xsl:template>
</xsl:stylesheet>
```

Первые две строки шаблона являются уже привычными. Следующие шесть строк – это строка, содержащая заголовки столбцов таблицы. Конструкция для извлечения текста заголовков таблицы вам уже знакома. А вот девятая строка является новой:

```
<xsl:for-each select="tutorial/enimals/dogs/dog">
```

Этот элемент шаблона позволяет выбрать и просмотреть все группы информации, полный путь к которым задается списком тегов "tutorial/enimals/dogs/dog". Обратите внимание – путь задается полностью, ни один из тегов опустить нельзя. Далее в ячейки таблицы помещается информация о наших собаках. В отличие от первых примеров путь к соответствующей информации тоже задается полностью.

Результат выполнения:

Кличка	Вес	Цвет
Шарик	18 кг	рыжий с черными подпалинами
Тузик	10 кг	белый с черными пятнами
Бобик	2 кг	бело-серый
Трезор	25 кг	черный

Сортировка

В предыдущих примерах порядок строк в таблице полностью соответствовал группам тегов в XML-файле. Этот порядок можно изменять. Добавим в тег
<xsl:for-each select="tutorial/enimals/dogs/dog">
атрибут order-by
<xsl:for-each select="tutorial/enimals/dogs/dog" order-by="dogName">
Таблица примет вид:

Кличка	Вес	Цвет
Бобик	2 кг	бело-серый
Трезор	25 кг	черный
Тузик	10 кг	белый с черными пятнами
Шарик	18 кг	рыжий с черными подпалинами

Более интересные результаты мы получим, если попытаемся отсортировать таблицу по столбцу "Вес". Вначале попробуем сделать по аналогии с предыдущим примером – атрибут order-by="dogName" заменим на order-by="dogWeight".

Результат приведен ниже

Кличка	Вес	Цвет
Тузик	10 кг	белый с черными пятнами
Шарик	18 кг	рыжий с черными подпалинами
Бобик	2 кг	бело-серый
Трезор	25 кг	черный

Таблица действительно отсортирована по столбцу "вес", но это не числовая, а строковая сортировка. Для того, чтобы браузер воспринял значения как числа, ему необходимо об этом сказать, – вместо *order-by="dogWeight"* необходимо написать *order-by="number(dogWeight)"*.

Теперь получили правильный результат:

Кличка	Вес	Цвет
Бобик	2 кг	бело-серый
Тузик	10 кг	белый с черными пятнами
Шарик	18 кг	рыжий с черными подпалинами
Трезор	25 кг	черный

Приведем теперь пример сортировки по нескольким столбцам. Различные элементы в атрибуте *order-by* должны разделяться символом ";" - *order-by="number(dogWeight); dogName"*

Таблица приведена ниже:

Кличка	Вес	Цвет
Бобик	2 кг	бело-серый
Тузик	10 кг	белый с черными пятнами
Шарик	18 кг	рыжий с черными подпалинами
Трезор	25 кг	черный

Задание:

Создать XSL-файл и применить его к XML-документу, полученному в предыдущей лабораторной работе.

Требования к представлению XML-документа:

- данные выводятся в виде таблицы;
- использовать строковую сортировку;
- использовать сортировку по числам.

xsl:apply-imports	Применяет шаблон из импортированной таблицы стилей
xsl:apply-temlates	Применяет шаблон к текущему элементу
xsl:attribute	Добавляет атрибут к ближайшему элементу
xsl:attribute-set	Определяет именованный набор атрибутов
xsl:call-template	Предоставляет способ вызова шаблона
xsl:choose	Предоставляет способ выбора среди нескольких вариантов в зависимости от выполнения тех или иных условий
xsl:comment	Создает комментарий XML
xsl:copy	Копирует текущий узел без его дочерних узлов и атрибутов в древовидную структуру результатов
xsl:copy-of	Копирует текущий узел со всеми его дочерними узлами и атрибутами в древовидную структуру результатов
xsl:decimal-format	Определяет символ (или строку), который используется при преобразовании чисел к строкам
xsl:element	Добавляет новый узел элемента в древовидную структуру результатов
xsl:fallback	Предоставляет способ определения альтернативы инструкциям, которые не реализованы
xsl:for-each	Предоставляет способ создания цикла в потоке результирующих данных
xsl:if	Предоставляет способ создания условной инструкции
xsl:import	Импортирует таблицу стилей
xsl:include	Добавляет таблицу стилей
xsl:key	Предоставляет способ определения ключевого элемента
xsl:message	Помещает сообщение в результирующий документ
xsl:namespace-alias	Предоставляет способ связывания пространства имен с другим пространством имен
xsl:number	Помещает число определенного формата в результирующий документ
xsl:otherwise	Указывает, что именно должно выполняться в том случае, если ни одно из условий, указанных в элементе xsl: when, вложенном в элемент xsl: choose, не будет удовлетворено
xsl:output	Предоставляет способ контроля над преобразованием результирующего документа
xsl:param	Предоставляет способ определения параметров
xsl:preserve-space	Предоставляет способ определения обработки пробелов, отмечая их как значимые
xsl processing-instruction	Заносит инструкцию на обработку в результирующий документ
xsl:sort	Предоставляет способ определения способа сортировки
xsl:strip-space	Предоставляет способ определения обработки пробелов, отмечая их как незначимые
xsl: stylesheet	Определяет корневой элемент таблицы стилей
xsl: template	Определяет шаблон для результирующего документа

xsl: text	Заносит текстовые данные в результирующий документ
xsl: transform	Определяет корневой элемент таблицы стилей
xsl: value-of	Создает текстовый узел и помещает значение в результирующую древовидную структуру
xsl: variable	Предоставляет способ определения переменной
xsl: when	Определяет условие, которое нужно проверить, а также действие, которое выполняется в случае положительного результата. Этот элемент всегда является дочерним по отношению к элементу xsl: choose

Пример преобразования XML-документа в HTML-документ.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
  <xsl:apply-templates />
</xsl:template>
<xsl:template match="факультет">
  <html>
  <head>
    <title><xsl:value-of select="title" /></title>
  </head>
  <body>
    <h2><xsl:value-of select="title" /></h2>
    <xsl:apply-templates />
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Перед тем, как начать описывать шаблоны элементов, которые встречаются в XML-документе, необходимо заставить интерпретатор начать разбор с корневого элемента делать это надо с помощью конструкции в строках 2-4 этого листинга.

В строке 3 применена XSLT-инструкция *<xsl:apply-templates />*. Она дает интерпретатору команду продолжить разбор всех вложенных элементов. Другими словами, в то место, в котором встречена данная инструкция, будут вставлены все шаблоны дочерних элементов по отношению к тому, шаблон которого интерпретируется в данный момент (в данном случае — всех элементов структуры, описанной в XMLфайле. Эта же инструкция применяется в шаблоне для элемента «факультет». В это место будут выведены шаблоны всех дочерних разделов элемента «факультет» Если опустить инструкцию в строке 12, то интерпретатор проведет разбор только элемента «факультет», не вдаваясь в дальнейшие подробности

Еще одна новая инструкция, это — *<xsl:valueof select*="...", которая заменяется значением содержимого соответствующего элемента. В данном случае (строка 8), она будет заменена на содержимое элемента title, являющегося дочерним по отношению к элементу факультет (но не к элементу title, дочернему по отношению к кафедра). Кроме того, эта инструкция может «подставлять» значение атрибутов самого элемента. Для этого в атрибуте select необходимо указать имя атрибута, предварив его символом @. Например, конструкция *<xsl:value-of select="@SRC"> внутри шаблона <xsl:templae match = "IMG">* будет заменена на значение атрибута SRC элемента IMG. Для того чтобы получить содержимое самого интерпретируемого в данный момент элемента необходимо применять значение «.»

Попробуем собрать все это вместе. XSLT-скрипт, который мы хотим применить для преобразования нашего XML-файла в HTML-представление (назовем его BRU.xsl), необходимо указать в XML-файле с помощью элемента

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

А сам скрипт дополним шаблонами для других элементов:

```
<?xml version="1.0" encoding="Windows-1251"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">

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

<xsl:template match="факультет">
<html>
<head>
<title><xsl:value-of select="title" /></title>
</head>
<body>
<h2><xsl:value-of select="title" /></h2>
Домашняя страница:
<a>
<xsl:attribute name="href">
<xsl:value-of select="homepage" />
</xsl:attribute>
<xsl:value-of select="homepage" />
</a>
<hr/>
<h3>Кафедры:</h3>
<xsl:apply-templates />
</body>
</html>
</xsl:template>

<xsl:template match="кафедра">
<b style="color: navy"><xsl:value-of select="title" /></b>
<a><xsl:attribute name="href">
<xsl:value-of select="homepage" />
</xsl:attribute>
<xsl:value-of select="homepage" /></a>
<xsl:apply-templates />
</xsl:template>

<xsl:template match="person">
  <br /><xsl:value-of select="firstname" />
      <xsl:value-of select="secondname" />
<a><xsl:attribute name="href">
mailto:<xsl:value-of select="email" />
</xsl:attribute>
<xsl:value-of select="email" /></a>
</xsl:template>
</xsl:stylesheet>
```