1 Введение

Практикум «XML – технологии» предназначена для обучения студентов основам использования XML–технологий.

Лабораторный практикум «XML – технологии» предназначен для формирования у студентов компетенций, связанных с описанием структур данных на основе XML, преобразованием XML-данных, проверкой правильности структуры документов XML.

Целью лабораторного практикума является содействие в формировании следующих компетенций:

- способен выбирать общесистемное программное обеспечение и прикладные пакеты программ (ПК-4);
- способен разрабатывать и отлаживать компоненты аппаратно-программных комплексов с помощью современных автоматизированных средств проектирования (ПК-7);
- умеет разрабатывать интерфейсы «человек - ЭВМ» (ПК-12);

В результате выполнения первой части лабораторного практикума студент должен уметь:

- описывать структуры данных с использованием технологии XML;
- разрабатывать XPath-запросы;
- разрабатывать XSLT-преобразования с фиксированной и адаптируемой структурой.

Введение в язык XML

XML является упрощенной версией языка SGML (Standard Generalized Markup Language, стандартный обобщенный язык разметки). SGML был утвержден ISO в качестве стандарта в 1986 году. SGML предназначен для создания других языков разметки, он определяет допустимый набор тэгов, их атрибуты и внутреннюю структуру документа. Контроль над правильностью использования тэгов осуществляется при помощи специального набора правил, называемых DTD- описаниями, которые используются при разборе документа.

Из-за своей сложности SGML использовался, в основном, для описания синтаксиса других языков разметки (наиболее известным из которых является HTML), и немногие приложения работали с SGML- документами напрямую.

XML является подмножеством SGML. То есть XML не содержит фиксированного набора тэгов и предназначен для создания языков разметки, подобных HTML. В XML используются только те возможности SGML, которые реально необходимы в Web.

XML обеспечивает ряд функциональных возможностей, которые отсутствуют в HTML:

- Позволяет разработчикам определять собственные тэги и атрибуты так, как это позволяет делать SGML.
- Предоставляет возможность проверки действительности структуры документов во время их обработки с помощью DTD или схем данных.

Существует два основных варианта использования XML:

1. Моделирование предметных областей и создание языков разметки на основе XML.

На основе XML создаются такие новые языки разметки, как:

- XHTML – расширяемый вариант HTML
- MathML (Mathematical Markup Language) - Формат описания математических формул.
- CML (Chemical Markup Language) - Формат описания химических формул.

- WML (Wireless Markup Language) - Вариант HTML для сотовых телефонов, используемый в WAP-технологии.
- SVG (Scalable Vector Graphics) - язык описания двухмерных векторных изображений.

2. Использование XML в качестве обменного формата в гетерогенных компьютерных системах (технология веб-сервисов).

Описание структур данных с помощью XML

Множество:

```
<?xml version="1.0" encoding="Windows-1251"?>
<!-- Множество -->
<Множество>
     <Элемент_множества id="1">
          <Параметры/>
     </Элемент_множества>
     <Элемент_множества id="2">
          <Параметры/>
     </Элемент_множества>
     <!-- . . . -->
     <Элемент_множества id="N">
          <Параметры/>
     </Элемент_множества>
</Множество>
```

Массив:

```
<?xml version="1.0" encoding="Windows-1251"?>
<!-- Массив -->
<Массив>
     <Элемент_массива номер="1">
          <Значение/>
     </Элемент_массива>
     <Элемент_массива номер="2">
          <Значение/>
     </Элемент_массива>
```

```
     <!-- . . . -->
     <Элемент_массива номер="n">
          <Значение/>
     </Элемент_массива>
</Массив>
```

Описание массива и множества не отличаются друг от друга. Элементы массива могут быть упорядочены по следованию друг за другом или по значению атрибута «номер». В множестве упорядоченность не учитывается.

Дерево:

```
<?xml version="1.0" encoding="Windows-1251"?>
<!-- Дерево -->
<Дерево>
     <Ветвь_11>
          <Лист/>
          <Ветвь_21>
               <Лист/>
               <Лист/>
               <Лист/>
          </Ветвь_21>
          <Ветвь_22>
               <Лист/>
               <Лист/>
          </Ветвь_22>
     </Ветвь_11>
     <Ветвь_12>
          <Лист/>
     </Ветвь_12>
</Дерево>
```

Древовидные структуры являются «естественными» для модели данных XML.

Граф (связи задаются в вершинах):

```
<?xml version="1.0" encoding="Windows-1251"?>
```

```
<Граф>

     <Вершина id="1">
          <Данные_о_вершине/>
          <Выходы>
               <Выход вершина="2"/>
               <Выход вершина="3"/>
          </Выходы>
     </Вершина>

     <Вершина id="2">
          <Данные_о_вершине/>
          <Выходы>
               <Выход вершина="1">
                    <Данные_о_связи/>
               </Выход>
          </Выходы>
     </Вершина>

     <Вершина id="3">
          <Данные_о_вершине/>
     </Вершина>

</Граф>
```

Граф (вершины и связи задаются в отдельных секциях документа):

```
<?xml version="1.0" encoding="Windows-1251"?>
<Граф>
     <Вершины>
          <Вершина id="1">
               <Данные_о_вершине/>
          </Вершина>
          <Вершина id="2"/>
          <Вершина id="3"/>
     </Вершины>
     <Связи>
```

```
        <Связь вершина_1="1" вершина_2="2">
            <Данные_о_связи/>
        </Связь>
        <Связь вершина_1="1" вершина_2="3"/>
        <Связь вершина_1="2" вершина_2="1"/>
    </Связи>
</Граф>
```

Такое описание удобно использовать как для ориентированного, так и для неориентированного графа.

Таким образом, с помощью модели данных XML достаточно просто описывать различные структуры данных.

Введение в XSLT

Для преобразования XML–документов в другие форматы была разработана технология XSLT.

XSLT расшифровывается как Extensible Stylesheet Language Transformations, расширяемый язык стилевых преобразований.

Как правило, с помощью XSLT выполняют три варианта преобразований:

1. Из XML в HTML. Этот вариант часто применяется в Web-приложениях.
2. Из XML в другой словарь XML (в другой набор тэгов). Этот вариант называют преобразованием словарей XML.
3. Из XML в текстовый формат, например в CSV (comma separated values – формат, в котором разделителями являются запятые).

Единственная задача, которую нельзя решить напрямую с помощью технологии XSLT – это преобразование из XML в двоичный формат. Для решения этой задачи можно использовать технологию расширений XSLT, когда из XSLT-преобразования вызывается исполняемый код.

XSLT-преобразование осуществляется специализированной программой или библиотекой, которая называется XSLT-процессор.

Принцип обработки XML-документов заключается в следующем: при разборе XSLT-документа XSLT-процессор обрабатывает инструкции этого языка и каждому элементу, найденному в XML- дереве, ставит в соответствие набор

тэгов HTML, XML или текст, которые определяют форматирование этого элемента.

Инструкции XSLT определяют точное положение элемента XML в документе с помощью XPath-выражений, поэтому существует возможность применять различные стили оформления к элементам с одинаковыми названиями, в зависимости от их положения в документе.

XSLT-преобразования с фиксированной структурой

Рассмотрим простой пример преобразования XSLT, который преобразует XML-документ, посвященный языкам разметки, в HTML-документ.

Файл XML:

```
<?xml version="1.0" encoding="Windows-1251"?>
<?xml-stylesheet type="text/xsl" href="Example_1.xsl"?>
<!-- Языки разметки -->
<languages>
     <language id="1">
          <name>HTML</name>
          <year>01.01.1990</year>
          <howold>19</howold>
     </language>
     <language id="2">
          <name>XML</name>
          <year>01.01.1998</year>
          <howold>11</howold>
     </language>
     <language id="3">
          <name>SGML</name>
          <year>01.01.1986</year>
          <howold>23</howold>
     </language>
</languages>
```

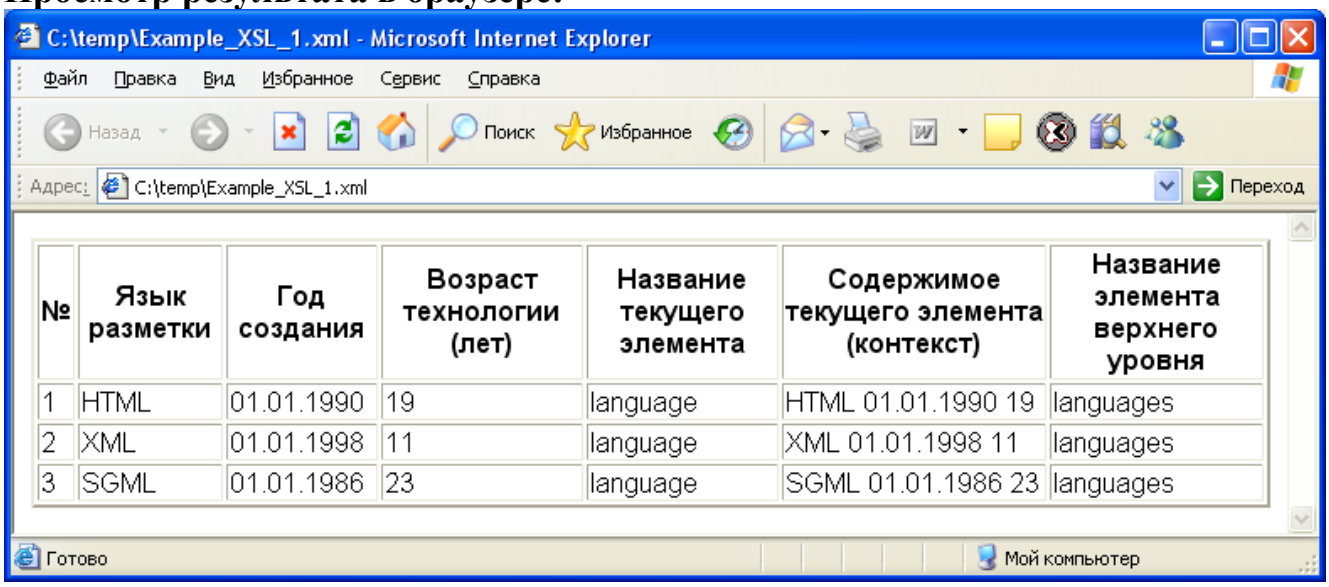
Файл XSLT:

```
<?xml version="1.0" encoding="Windows-1251"?>
<!--XSLT - документ является XML - документом. -->
```

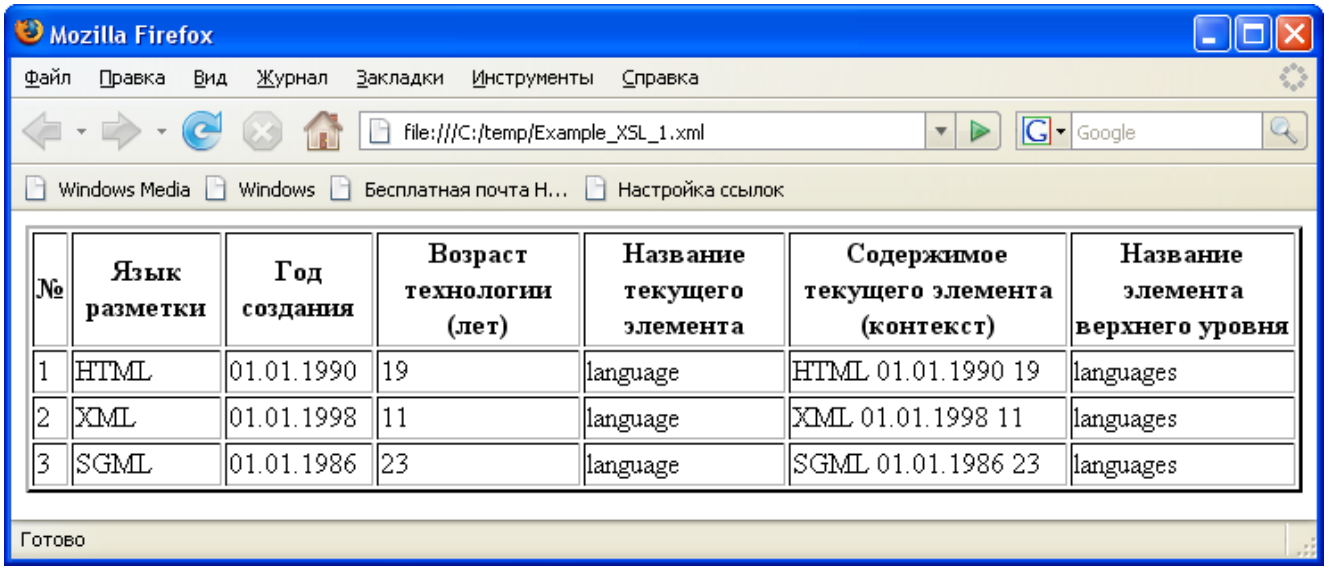
```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<!-- Описание XSLT - документа -->
<xsl:template match="/">
<!-- Правило обработки корневого элемента XML - документа -->
     <HTML>
     <BODY>
     <TABLE BORDER="2">
     <TR>
          <TH>№</TH>
          <TH>Язык разметки</TH>
          <TH>Год создания</TH>
          <TH>Возраст технологии (лет)</TH>
          <TH>Название текущего элемента</TH>
          <TH>Содержимое текущего элемента (контекст)</TH>
          <TH>Название элемента верхнего уровня</TH>
     </TR>
     <xsl:for-each select="languages/language">
<!-- Перебор в цикле всех элементов language, вложенных в элемент
languages. -->
     <TR>
          <TD><xsl:value-of select="@id"/></TD>
<!-- Выбор значения атрибута id элемента language -->
          <TD><xsl:value-of select="name"/></TD>
<!-- Выбор значения элемента name, вложенного в элемент language -->
          <TD><xsl:value-of select="year"/></TD>
<!-- Выбор значения элемента year, вложенного в элемент language -->
          <TD><xsl:value-of select="howold"/></TD>
<!-- Выбор значения элемента howold, вложенного в элемент Language
-->
          <TD><xsl:value-of select="name(.)"/></TD>
<!-- Название текущего элемента -->
          <TD><xsl:value-of select="."/></TD>
<!-- Содержимое текущего элемента (контекст) -->
          <TD><xsl:value-of select="name(parent::*)"/></TD>
<!-- Название элемента верхнего уровня (также можно использовать
```

```
select="name(..)") -->
      </TR>
      </xsl:for-each>
      </TABLE>
      </BODY>
      </HTML>
</xsl:template>
</xsl:stylesheet>
```

Просмотр результата в браузере:



№	Язык разметки	Год создания	Возраст технологии (лет)	Название текущего элемента	Содержимое текущего элемента (контекст)	Название элемента верхнего уровня
1	HTML	01.01.1990	19	language	HTML 01.01.1990 19	languages
2	XML	01.01.1998	11	language	XML 01.01.1998 11	languages
3	SGML	01.01.1986	23	language	SGML 01.01.1986 23	languages



№	Язык разметки	Год создания	Возраст технологии (лет)	Название текущего элемента	Содержимое текущего элемента (контекст)	Название элемента верхнего уровня
1	HTML	01.01.1990	19	language	HTML 01.01.1990 19	languages
2	XML	01.01.1998	11	language	XML 01.01.1998 11	languages
3	SGML	01.01.1986	23	language	SGML 01.01.1986 23	languages

В файле XML наиболее интересна вторая строка (инструкция обработки xml-stylesheet):

```
<?xml-stylesheet type="text/xsl" href="Example_1.xsl"?>
```

Значение атрибута type="text/xsl" указывает на то, что для отображения

документа должно быть использовано XSLT-преобразование. В атрибуте href задается URI для доступа к файлу XSLT. Расширение файла XSLT обычно « xsl ».

Если открыть документ с помощью анализатора XML, который поддерживает инструкцию xml-stylesheet и может вызывать XSLT-процессор, то XSLT-преобразование будет применено к документу. Такие анализаторы, например, встроены в браузеры Mozilla и Internet Explorer.

Обратим внимание, что преобразование XSLT является документом XML. Этот документ содержит программу на языке XSLT. Команды XSLT задаются в виде XML-элементов. Выделить эти команды просто, так как перед ними стоит префикс « xsl: » (как выяснится позже, замечание по поводу префикса потребует уточнения).

Рассмотрим более подробно текст преобразования XSLT.

```
<?xml version="1.0" encoding="Windows-1251"?>
```

Преобразование XSLT является XML-документом, поэтому документ начинается с инструкции обработки <?xml . . . ?>

```
<!--XSLT - документ является XML - документом. -->
```

В преобразовании XSLT используются XML-комментарии <!-- . . . -->

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

Элемент stylesheet является корневым элементом документа и соответствует полной «программе» на языке XSLT. Атрибут version с указанием версии обязателен. Атрибут xmlns:xsl задает префикс пространства имен.

Вместо элемента xsl:stylesheet может быть использован элемент xsl:transform. Эти элементы являются синонимами.

```
<!-- Описание XSLT - документа -->
<xsl:template match="/">
```

Элемент template (шаблон) соответствует «процедуре». Элементов template в преобразовании может быть несколько, в этом примере он один. Атрибут match определяет, какому элементу преобразуемого документа соответствует шаблон. Значение match="/" показывает, что шаблон соответствует корневому элементу преобразуемого документа. Шаблон со значением match="/" вызывается первым. В общем случае значение атрибута match – это XPath-выражение.

```
<!-- Правило обработки корневого элемента XML - документа -->
     <HTML>
     <BODY>
     <TABLE BORDER="2">
     <TR>
          <TH>№</TH>
          <TH>Язык разметки</TH>
          <TH>Год создания</TH>
          <TH>Возраст технологии (лет)</TH>
          <TH>Название текущего элемента</TH>
          <TH>Содержимое текущего элемента (контекст)</TH>
          <TH>Название элемента верхнего уровня</TH>
     </TR>
```

Тэги языка HTML (без префикса «xsl:») просто выводятся в результирующий документ. Это можно рассматривать как оператор вывода. Тэг TR формирует строку таблицы в HTML. Элементы TH формируют заголовки в первой строке таблицы.

```
     <xsl:for-each select="languages/language">
<!-- Перебор в цикле всех элементов language, вложенных в элемент
languages. -->
```

Элемент for-each – единственная разновидность циклов в XSLT. Никаких других видов циклов в XSLT нет. В атрибуте select указывается XPath выражение, которое возвращает перебираемые в цикле элементы.

Содержимое элемента for-each (тело цикла) выполняется для каждого узла, который возвращается в атрибуте select.

Поэтому тело цикла будет выполняться три раза, для каждого элемента language.

```
     <TR>
          <TD><xsl:value-of select="@id"/></TD>
<!-- Выбор значения атрибута id элемента language -->
```

Обратите внимание, что, попадая внутрь тела цикла, мы «проваливаемся» в контекст, который задается текущим значением атрибута select элемента for-each. Все XPath-выражения внутри тела цикла должны быть контекстными.

Здесь мы формируем строку таблицы с помощью элемента TR. Элемент TD задает ячейку таблицы.

Элемент value-of возвращает значение, задаваемое в виде XPath-выражения в атрибуте select. Обратите внимание, что это XPath-выражение контекстное, оно возвращает атрибут id для текущего элемента language, который соответствует текущей итерации цикла.

Атрибут select используется и в элементе for-each, и в элементе value-of. И в том, и в другом случае это XPath-выражение. Однако смысл у них различный.

В элементе for-each предполагается, что это выражение возвращает несколько значений. Для каждого значения выполняется итерация цикла.

В элементе value-of предполагается, что выражение возвращает единственное значение. Если по ошибке написать неоднозначное выражение, то, как правило, XSLT-процессор возвращает первое значение.

Далее с помощью элемента value-of формируются другие ячейки таблицы.

Значение select="." возвращает XML-значение для элемента language. Но браузер не отображает XML-тэги, а отображает только текстовое содержимое вложенных элементов.

Все XPath-выражения являются контекстными. Выражение <xsl:value-of select="name(.)"/> возвращает название текущего элемента (language), а выражение <xsl:value-of select="name(parent::*)"/> возвращает название элемента верхнего уровня (languages). Вместо select="name(parent::*)" также можно использовать select="name(..)".

```
          <TD><xsl:value-of select="name"/></TD>
<!-- Выбор значения элемента name, вложенного в элемент language -->
          <TD><xsl:value-of select="year"/></TD>
<!-- Выбор значения элемента year, вложенного в элемент language -->
          <TD><xsl:value-of select="howold"/></TD>
<!-- Выбор значения элемента howold, вложенного в элемент language -
->
          <TD><xsl:value-of select="name(.)"/></TD>
<!-- Название текущего элемента -->
          <TD><xsl:value-of select="."/></TD>
```

```
<!-- Содержимое текущего элемента (контекст) -->
          <TD><xsl:value-of select="name(parent::*)"/></TD>
<!-- Название элемента верхнего уровня (также можно использовать
select="name(..)") -->
     </TR>
```

Далее следуют необходимые закрывающие тэги.

```
     </xsl:for-each>
     </TABLE>
     </BODY>
     </HTML>
</xsl:template>
</xsl:stylesheet>
```

XSLT-преобразования с адаптируемой структурой

Рассмотренный шаблон XSLT является шаблоном с фиксированной структурой выходного HTML–документа. Если во входном XML-документе поменять местами элементы <name> и <year>, то это не приведет к перестановке колонок в таблице.

Рассмотрим XSLT-преобразование, которое не содержит жестко заданную структуру выходного документа, а адаптируется к структуре входного документа.

Адаптивность XSLT-преобразования достигается за счет того, что используются несколько элементов template, каждый из которых соответствует элементу входного документа.

Порядок вызова элементов template не задается. Они вызываются в такой последовательности, в которой встречаются соответствующие элементы во входном документе.

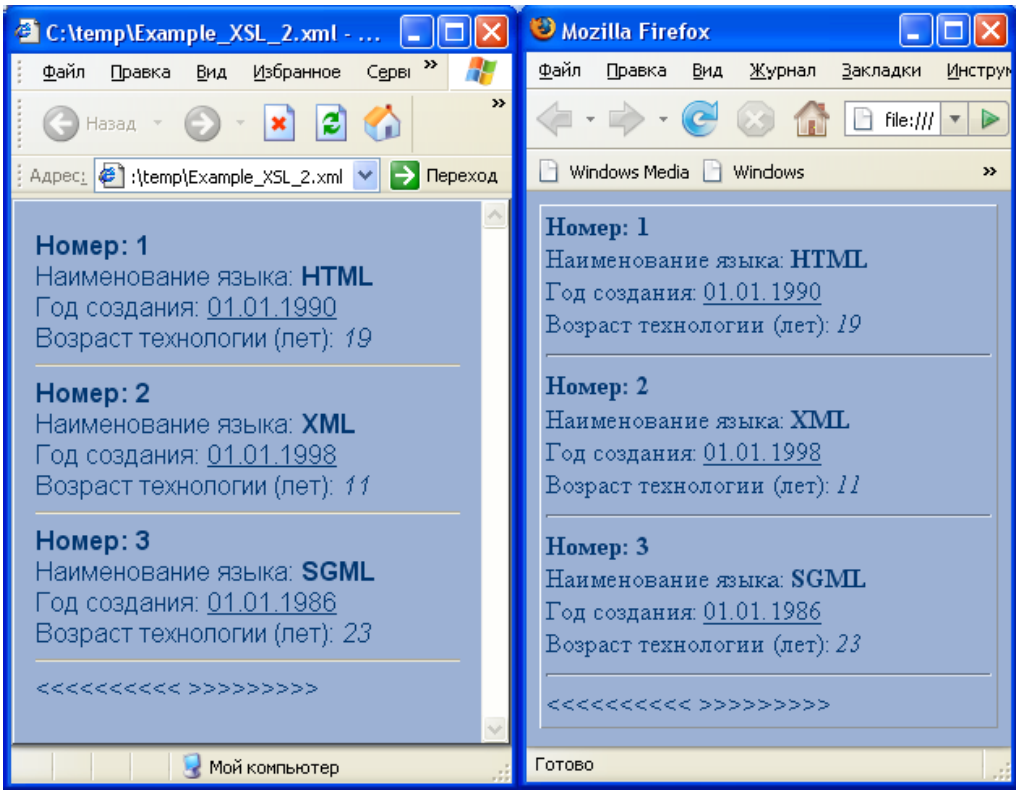
Файл XSLT:

```
<?xml version="1.0" encoding="Windows-1251"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  -->
<!-- Шаблон обработки корневого элемента XML - документа -->
<xsl:template match="/">
```

```
     <HTML>
     <HEAD>
     <LINK href="met.css" rel="stylesheet" type="text/css"/>
     </HEAD>
     <BODY>
     <xsl:apply-templates/>
     </BODY>
     </HTML>
</xsl:template>
<!-- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++  -->

<!-- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++  -->
<!-- Шаблон обработки элемента languages -->
<xsl:template match="languages">
<!-- Вызов шаблона для элемента language (шаблон вызывается
необходимое количество раз) -->
     <xsl:apply-templates/>
</xsl:template>
<!-- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++  -->

<!-- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++  -->
<!-- Шаблон обработки элемента language -->
<xsl:template match="language">

<!-- Получение значения атрибута id (префикс @ означает атрибут) -->
     <B>Номер: <xsl:value-of select="@id"/></B><BR/>

<!-- Вызов шаблонов для элементов name, year и howold -->
     <xsl:apply-templates/>
     <HR/>
</xsl:template>
<!-- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++  -->

<!-- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++  -->
<!-- Шаблон обработки элемента name -->
<xsl:template match="name">
```

```
     Наименование языка: <B><xsl:value-of select="."/></B><BR/>
<!-- select="." - получение значения текущего элемента -->
</xsl:template>
<!-- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++   -->

<!-- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++   -->
<!-- Шаблон обработки элемента year -->
<xsl:template match="year">
     Год создания: <U><xsl:value-of select="."/></U><BR/>
</xsl:template>
<!-- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++   -->

<!-- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++   -->
<!-- Шаблон обработки элемента howold -->
<xsl:template match="howold">
     Возраст технологии (лет): <I><xsl:value-of
select="."/></I><BR/>
</xsl:template>
<!-- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++   -->
</xsl:stylesheet>
```

Просмотр результата в браузере:



XML файл в этом примере такой же, как и предыдущем случае. Для применения к нему XSLT-преобразования можно использовать инструкцию

```
<?xml-stylesheet type="text/xsl" href="  название XSLT-
преобразования.xsl  "?>
```

В этом случае результат отображается в браузере. Или можно использовать отладчик XSLT, встроенный в XMLPad.

Рассмотрим более подробно текст преобразования XSLT.

```
<?xml version="1.0" encoding="Windows-1251"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++  -->
<!-- Шаблон обработки корневого элемента XML - документа -->
<xsl:template match="/">
```

Как и в предыдущем примере, здесь присутствует шаблон для корневого элемента. Но в отличие от предыдущего примера здесь присутствуют и другие шаблоны.

```
<HTML>
<HEAD>
<LINK href="met.css" rel="stylesheet" type="text/css"/>
```

В секции HEAD документа HTML встречается тэг LINK, который связывает HTML-документ с внешней таблицей стилей. После обработки XML-документа с помощью XSLT к полученному HTML-документу будет применена таблица стилей CSS.

```
</HEAD>
<BODY>
<xsl:apply-templates/>
```

Главной инструкцией в этом примере является apply-templates. Эта инструкция выполняет следующие действия:

1. В текущем контексте (в текущем тэге) входного документа находит все непосредственно вложенные элементы.

2. Для каждого такого элемента пытается найти соответствующий шаблон (template) и выполнить его. Соответствующим является шаблон вида <xsl:template match=" название элемента ">.

В нашем примере в корневой элемент непосредственно вложен элемент languages, поэтому далее будет вызван <xsl:template match="languages">.

Обратите внимание, что реальный корневой элемент документа «languages» считается вложенным в корневой элемент «/». Поэтому элементы language не вложены непосредственно в «/».

```
     </BODY>
     </HTML>
</xsl:template>
<!-- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++  -->

<!-- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++  -->
<!-- Шаблон обработки элемента languages -->
<xsl:template match="languages">
<!-- Вызов шаблона для элемента language (шаблон вызывается
необходимое количество раз) -->
     <xsl:apply-templates/>
```

В нашем примере в элемент languages непосредственно вложен элемент language, поэтому далее будут вызваны шаблоны <xsl:template match="language"> для каждого элемента language.

```
</xsl:template>
<!-- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++  -->

<!-- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++  -->
<!-- Шаблон обработки элемента language -->
<xsl:template match="language">

<!-- Получение значения атрибута id (префикс @ означает атрибут) -->
     <B>Номер: <xsl:value-of select="@id"/></B><BR/>

<!-- Вызов шаблонов для элементов name, year и howold -->
     <xsl:apply-templates/>
```

В нашем примере в элемент language непосредственно вложены элементы name, year и howold. Инструкция apply-templates будет вызывать шаблоны для этих элементов.

```
     <HR/>
</xsl:template>
<!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  -->

<!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  -->
<!-- Шаблон обработки элемента name -->
<xsl:template match="name">
     Наименование языка: <B><xsl:value-of select="."/></B><BR/>
```

Так как в этом шаблоне мы находимся в контексте элемента name (то есть уже «провалились» внутрь элемента name), то для получения значения текущего элемента надо использовать XPath-выражение «.» или «self::*».

Если вместо выражения <xsl:value-of select="."/> использовать <xsl:value-of select="name"/>, то оно вернет пустое значение, так как будет искать несуществующий элемент name внутри текущего элемента name.

Шаблоны для элементов year и howold построены аналогично шаблону для элемента name.

```
</xsl:template>
<!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  -->

<!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  -->
<!-- Шаблон обработки элемента year -->
<xsl:template match="year">
     Год создания: <U><xsl:value-of select="."/></U><BR/>
</xsl:template>
<!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  -->

<!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  -->
<!-- Шаблон обработки элемента howold -->
<xsl:template match="howold">
     Возраст технологии (лет): <I><xsl:value-of
select="."/></I><BR/>
```

```
</xsl:template>
<!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++   -->

</xsl:stylesheet>
```

Отметим, что текстовое содержимое элемента CDATA_Example скопировано в выходной поток, что в нашем случае является нежелательным эффектом. Поэтому, в преобразовании лучше создавать шаблоны для всех возможных элементов. Например, если добавить следующий шаблон,

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

то ненужные символы угловых скобок не будут отображаться. Этот шаблон фактически указывает, что для элемента CDATA_Example не нужно выполнять никаких действий.

Вместо этого можно использовать для элемента apply-templates атрибут select. Этот атрибут содержит XPath-выражение, которое указывает, для каких элементов нужно искать и выполнять шаблоны.

Если в шаблоне обработки корневого элемента команду

```
<xsl:apply-templates/>
```

заменить на

```
<xsl:apply-templates select="//language"/>
```

то элемент CDATA_Example не будет обрабатываться и ненужные символы угловых скобок не будут отображаться. Так как элемент language не вложен непосредственно в «/», то используется XPath-выражение «//language», а не «language».

Можно или создавать шаблоны для всех элементов входного документа или указывать область видимости в атрибуте select.

В атрибуте select может быть указано произвольное XPath-выражение, которое не обязательно возвращает непосредственно вложенные элементы.

Таким образом, использование нескольких элементов template и конструкции apply-templates позволяет разрабатывать преобразования, адаптирующиеся к структуре входного документа.

2 Схема и описание лабораторной установки

В качестве лабораторной установки используется компьютер со следующим программным обеспечением:

- операционная система Windows 7 и выше
- XML-редактор XMLPad (свободно-распространяемое ПО).

3 Содержание отчета по лабораторным работам

Отчеты разрабатываются отдельно по каждой лабораторной работе. Отчет по каждой лабораторной работе должен включать:

- титульный лист;
- тексты XML-документов, XPath-запросов, XSLT-преобразований;
- результаты работы XPath-запросов, XSLT-преобразований.

4 Задачи и порядок выполнения работ

Задание 1

Описание структур данных с использованием XML.

Разработайте пример описания выбранной Вами предметной области в виде документа XML. Документ должен содержать около 30-50 XML-элементов.

Разработанный документ XML должен содержать элементы описания структур данных в виде множества (или массива), дерева, графа.

Задание 2

Разработка XPath-запросов.

Для документа, разработанного в предыдущей лабораторной работе, разработайте запросы XPath, содержащие обращение к элементам и атрибутам, фильтры и сравнения.

Задание 3

Разработка XSLT-преобразования с фиксированной структурой.

Разработайте XSLT-преобразование, генерирующее выходной HTML–документ с фиксированной структурой.

Преобразование должно содержать один элемент template, соответствующий корневому элементу XML-документа.

Задание 4

Разработка XSLT-преобразования с адаптируемой структурой.

Разработайте XSLT-преобразование, генерирующее выходной HTML–документ, который адаптирован к структуре входного документа.

Преобразование должно содержать несколько элементов template, которые вызываются с использованием конструкции apply-templates.

5 Контрольные вопросы

1. Какие основные виды конструкций XML Вы знаете?
2. Приведите примеры структур данных, которые можно описать с использованием XML.
3. Как с использованием XML описать множество?
4. Как с использованием XML описать дерево?
5. Как с использованием XML описать граф?

6. Для чего предназначен язык XPath?
7. Какие основные выражения используются в XPath?
8. Чем отличается обращение к элементам от обращения к атрибутам в XPath?

9. Каким образом в XPath можно использовать фильтры и операторы сравнения?
10. Что такое контекстный XPath-запрос и чем он отличается от неконтекстного? Почему контекстные XPath-запросы так широко используются?
11. Для чего предназначена технология XSLT?
12. Какие три варианта преобразований обычно выполняют с помощью XSLT?
13. Что такое XSLT-процессор?
14. Для чего в XSLT используется конструкция stylesheet?
15. Для чего в XSLT используется конструкция value-of?
16. Для чего в XSLT используется конструкция for-each?
17. Каким образом совместно используются XPath и XSLT?
18. Какие элементы технологии XSLT позволяют создавать преобразования, адаптирующиеся к структуре входного документа?
19. Для чего в XSLT используется конструкция template?
20. Приведите пример преобразования, в котором используется несколько конструкций template.
21. Для чего в XSLT используется конструкция apply-templates?
22. Каким образом можно использовать конструкцию apply-templates для выбранных элементов входного XML-документа?

6 Литература

1. Расширяемый язык разметки (XML) 1.0 (вторая редакция), 2000. [электронный ресурс] – Режим доступа: http://www.rol.ru/news/it/helpdesk/xml01.htm – Загл. с экрана.

2. Язык XML Path (XPath) версия 1.0, 1999. [электронный ресурс] – Режим доступа: http://www.rol.ru/news/it/helpdesk/xpath01.htm – Загл. с экрана.
3. Язык преобразований XSL (XSLT) версия 1.0, 1999. [электронный ресурс] – Режим доступа: http://www.rol.ru/news/it/helpdesk/xslt01.htm – Загл. с экрана.