4.2. Язык преобразований XSLT

XSLT (eXtensible Stylesheet Language Transformations) - это декларативное описание преобразования (трансформации) любого XML-документа. Спецификация XSLT входит в состав XSL и является рекомендацией W3C.

Существует три основных способа преобразования XML-документов с помощью XSLT в другие форматы, например, в HTML:

- XML-документ и связанная с ним таблица стилей отправляются клиенту (веб-браузеру), который преобразует документ как указано в таблице стилей, и после этого представляет результат пользователю.
- Сервер применяет таблицу стилей XSLT к XML-документу и преобразует его в другой формат (обычно, в HTML). После этого результат отправляется клиенту (веб-браузеру).
- Какая-то программа преобразует оригинальный XML-документ в другой формат (обычно, в HTML), затем результат помещается на сервер. И сервер, и клиент имеет дело с уже преобразованным документом (рис.4.2.).

К разным документам можно применять разные таблицы стилей – и каждый раз получать различный результат.

Для обращения к элементам XML-документа XSLT использует XPath. Для объявления таблицы XSLT используется корневой элемент <xsl:stylesheet> (возможен вариант <xsl:transform>, варианты равнозначны):

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

Xml-файл, содержащий таблицу XSLT, состоит из набора шаблонов. Шаблон – это набор инструкций для преобразования входного XML-документа. Для создания шаблонов используется элемент <xsl:template>:

```
<xsl:template match = “XPath-выражение”>
        Тело шаблона
</xsl:template>
```

Структура элемента <xsl:template>:

```
<xsl:template
        match = pattern
```

```
        name = qname
        priority = number
        priority = qname>
            Тело шаблона
</xsl:template>
```

Правило шаблона задается элементом <xsl:template>. Атрибут match соответствует пути XPath, который идентифицирует исходные узел или узлы, к которым это правило применяется. Если элемент <xsl:template> не имеет атрибута name, атрибут match обязателен. Атрибуты priority и mode определяют последовательность и обязательность выполнения шаблонов. Содержимое элемента <xsl:template> является шаблоном, который обрабатывается если данное правило шаблона задействовано.

Для получения значений элемента XML и вывода его, например, на экран, используется элемент <xsl:value-of>:

```
<xsl:template match="employee">
        <xsl:value-of select="name" />
  </xsl:template>
```

Элемент <xsl:apply-templates> дает указание преобразователю сравнивать каждый дочерний элемент исходного элемента (который соответствует данному шаблону) с другими шаблонами в таблице стилей и, если соответствие обнаружено, выводить шаблон для соответствующего узла.

В отсутствие атрибута select инструкция <xsl:apply-templates> обрабатывает все дочерние узлы текущего узла, включая узлы текста:

```
<xsl:template match="chapter">
        <fo:block>
            <xsl:apply-templates/>
        </fo:block>
</xsl:template>
```

Чтобы обрабатывать не все дочерние узлы, а лишь узлы, отобранные по некому выражению, может использоваться атрибут select. Значением атрибута select является XPath-выражение. После обработки этого выражения должен получиться набор узлов.

Шаблоны можно вызывать по имени. Именованный шаблон задается элементом <xsl:template> с атрибутом name. Элемент <xsl:call-template> вызывает шаблон по имени name, идентифицирующий шаблон, который должен быть вызван.

Синтаксис именованного шаблона:

```
<xsl:template name = "MyTemplate">
        Тело шаблона
</xsl:template>
```

Синтаксис вызова именованного шаблона:

```
<xsl:call-template name = "MyTemplate">
        Тело шаблона
</xsl:template>
```

Создавать новые узлы в XML-документе можно с помощью конструкции <xsl:element>:

```
<xsl:element
        name = “elementName”
        namespace = “URI”
        use-attribute-sets = “nameList”>
            Содержание элемента
</xsl:element>
```

Элемент <xsl:element> позволяет создавать элемент с вычисляемым названием. Единственным обязательным атрибутом данной конструкции является атрибут name.

Новые атрибуты создаются с помощью конструкции <xsl:attribute>:

```
<xsl:attribute
        name = “attrName”
        namespace = “URI”>
            Содержание атрибута
</xsl:attribute>
```

Чтобы к конечным элементам добавить атрибуты, можно использовать элемент <xsl:attribute> независимо от того, созданы ли первые фиксированными конечными элементами или такими инструкциями, как <xsl:element>.

Шаблон также может содержать текстовые узлы. Каждый текстовый узел в шаблоне, оставшийся после удаления пробельных символов, создаст в конечном дереве текстовый узел с тем же самым строковым значением. Смежные текстовые узлы в конечном дереве автоматически объединяются. Синтаксис создания текстового узла:

```
<xsl:text
        disable-output-escaping = "yes" | "no">
        <!--Содержание текстового узла: #PCDATA -->
</xsl:text>
```

Для создания в конечном дереве узла комментариев используется элемент <xsl:comment>. Содержимое элемента <xsl:comment> является шаблоном для строкового значения данного узла комментария:

```
<xsl:comment>
      <!—Текст комментария -->
</xsl:comment>
```

Для создания узла инструкции обработки (процессинговых инструкций) используется элемент <xsl:processing-instruction>. Содержимое элемента <xsl:processing-instruction> является шаблоном для строкового значения узла инструкции обработки. Элемент <xsl:processing-instruction> имеет обязательный атрибут name, который определяет название данного узла инструкции обработки:

```
<xsl:processing-instruction
        name = “name”>
            <!-- Содержание инструкции -->
</xsl:processing-instruction>
```

Ниже приведен пример XSLT-преобразования XML-документа. Исходный XML-документ:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="emp1.xsl"?>

<employees xmlns:au="http://www.demotest.com">
 <employee>
        <name>Adam Stein</name>
          <salary>23500</salary>
          <jobtitle>Programmer</jobtitle>
          <region>Redmond</region>
 </employee>
 <employee>
         <name>Susan Tjarnberg</name>
         <salary>51000</salary>
         <jobtitle>Tester</jobtitle>
          <region>Minneapolis</region>
```

```
</employee>
<employee>
        <name>Catherine Turner</name>
        <salary>45000</salary>
       <jobtitle>System Architect</jobtitle>
        <region>Dallas</region>
</employee>
<employee>
        <name>Wendy Vasse</name>
        <salary>72000</salary>
        <jobtitle>Project Manager</jobtitle>
        <region>Washington D.C.</region>
</employee>
<employee>
         <name>Paula Thurman</name>
        <au:salary>34500</au:salary>
         <jobtitle>Programmer</jobtitle>
        <region>Sydney</region>
        </employee>
<employee>
       <name>Richard Marshall</name>
        <au:salary>30000</au:salary>
       <jobtitle>Programmer</jobtitle>
        <region>Canberra</region>
</employee>
</employees>
```

Таблица преобразований XSLT:

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

<xsl:template match="/">
  <HTML>
       <BODY>
            <H1>Salaries Report</H1>
            <xsl:apply-templates/>
  </BODY>
  </HTML>

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

```
        <xsl:apply-templates select="employee[salary]"/>
        <xsl:apply-templates select="employee[au:salary]"/>
 </xsl:template>
<xsl:template match="employee[salary]">
          <p><xsl:value-of select="name"/>, <xsl:value-of select="jobtitle"/>,
       <xsl:value-of select="region"/>, <xsl:value-of select="salary"/> USD
(US dollars)</p>
 </xsl:template>
<xsl:template match="employee[au:salary]">
  <p><xsl:value-of select="name"/>,
<xsl:value-of select="jobtitle"/>, <xsl:value-of select="region"/>,
<xsl:value-of select="au:salary"/> AUD (Australian dollars)</p>
 </xsl:template>
</xsl:stylesheet>
```

Код, получившийся в результате преобразования:

```
<HTML xmlns:au="http://www.demotest.com">
        <BODY>
                <H1>Salaries Report</H1>
                <p>Adam Stein, Programmer, Redmond,
                        23500 USD (US dollars)</p>
                <p>Susan Tjarnberg, Tester, Minneapolis,
                        51000 USD (US dollars)</p>
                <p>Catherine Turner, System Architect, Dallas,
                        45000 USD (US dollars)</p>
                <p>Wendy Vasse, Project Manager, Washington D.C.,
                        72000 USD (US dollars)</p>
                <p>Paula Thurman, Programmer, Sydney,
                        34500 AUD (Australian dollars)</p>
                <p>Richard Marshall, Programmer, Canberra,
                        30000 AUD (Australian dollars)</p>
        </BODY>
</HTML>
```

Отображение результата в браузере (Рис.1):