Лабораторная работа № 13
РАСШИРЕНИЕ XSLT

Цель работы: изучить расширение XSLT, получить навыки создания таблиц стилей с элементами сортировки и элементом с условиями.

Теоретические сведения для выполнения работы

Понятие XSL

XSL (eXtensible Stylesheet Language) —язык таблиц стилей для XML. XSL служит языком трансформирования документов XML и состоит из XML-словаря семантики форматирования. Имея класс произвольно структурированных XML-документов и файлов данных, дизайнеры используют таблицы стилей XSL для указания на то, как это структурированное содержимое должно быть представлено; как содержимое-источник должно быть стилизовано, расположено и разбито на странице в Web-браузере.

Процессор таблицы стилей XSL принимает документ или данные на языке XML и таблицу XSL и производит представление содержимого XML-источника так, как это задумано дизайнером данной таблицы стилей. Существуют два аспекта этого процесса представления: первый — конструирование результирующего дерева из дерева XML-источника, второй — интерпретация результирующего дерева для производства форматированного вывода, пригодного для показа на экране дисплея. Первый аспект называется трансформация дерева, а второй называется форматирование. Процесс форматирования выполняется форматировщиком. Этот форматировщик может быть утилитой вывода браузера.

Форматирование становится доступным при включении семантики форматирования в результирующее дерево. Семантика форматирования выражена в терминах каталога классов объектов форматирования. Узлы результирующего дерева это объекты форматирования. Классы объектов форматирования обозначают типографические абстракции, например такие как страница, параграф, таблица. Точный контроль за представлением этих абстракций осуществляется с помощью набора свойств форматирования, таких как управление отступами, расстояниями между словами и

буквами, висячими строками и переносами. XSL классы объектов форматирования и свойства форматирования предоставляют словарь для выражения целей представления.

Имеются 4 класса свойств XSL, которые можно идентифицировать как:

1. Свойства, скопированные из CSS (с неизмененной семантикой CSS2).
2. Свойства CSS с расширенными значениями.
3. Свойства CSS, разбитые на части и/или расширенные.
4. "Чистые" свойства XSL.

Процессоры XSL обязаны использовать механизмы пространства имен XML Names для распознавания элементов и атрибутов пространства имен **http://www.w3.org/1999/XSL/Format**.

Элементы из пространства имен XSL распознаются только в таблицах стилей, но не в документе-источнике. Разработчики обязаны не расширять пространство имен XSL за счет дополнительных элементов и атрибутов. Вместо этого любое расширение обязано находиться в отдельном пространстве имен.

Спецификация XSL использует префикс fo: для ссылки на элементы пространства имен XSL. В то же время таблицы стилей XSL могут использовать любые префиксы при наличии объявления пространства имен, связывающего префикс с URL пространства имен XSL. Спецификация XSL размещена по адресу **https://www.w3.org/TR/2001/REC-xsl-20011015/**, в которой представлены свойства и примеры использования префиксов, используемые в XSL.

Следует отметить, что XSL использует XSLT и XPath для конструирования дерева отбора патернов, предоставляя таким образом возможности управления тем, как представлены части содержимого-источника и какие свойства ассоциированы с этими частями содержимого, где используются смешанные пространства имен. XSLT определяет набор примитивов для описания преобразования документа, а XPath определяет синтаксис описания различных мест в документах XML.

Расширение 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:

1. XML-документ и связанная с ним таблица стилей отправляются клиенту (веб-браузеру), который преобразует документ как указано в таблице стилей и затем предоставляет результат преобразования пользователю.

2. Сервер применяет таблицу стилей XSLT к XML-документу и преобразует его в другой формат (обычно, в HTML). После этого результат отправляется клиенту (веб-браузеру).

3. Какая-то программа преобразует оригинальный XML-документ в другой формат (обычно, в HTML), затем результат помещается на сервер. Таким образом, сервер и клиент имеет дело с преобразованным документом

При помощи XSLT можно добавлять/удалять элементы и атрибуты в конечный файл. Также, можно реорганизовывать и сортировать элементы, выполнять тесты, определять, какие элементы скрыть или отобразить, и т. п. Таблица стилей XSL содержит один или больше наборов правил преобразования, которые называются шаблонами преобразования. Шаблон преобразования содержит правила, которые применяются, когда найден узел (элемент, атрибут, текст, комментарий), соответствующий условию поиска. Пример использования XSLT представлен в таблице 13.1

Таблица 13.1

Расширение XSLT

XML-документ	**XSLT**
<?xml version="1.0" encoding="UTF-8"?> <?xml-stylesheet type="text/xsl" href="catalog.xsl"?> <catalog> <cd> <title>Empire Burlesque</title> <artist>Bob Dylan</artist> <country>USA</country> <company>Columbia</company> <price>10.90</price> <year>1985</year> </cd>	<?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>My first template rule</title> </head> <body> My CD Collection <table border="1"> <tr bgcolor="#9acd32"> <th>Title</th>

Продолжение таблицы 13.1

XML-документ	XSLT
<cd> <title>Hide your heart</title> <artist>Bonnie Tyler</artist> <country>UK</country> <company>CBS Records</company> <price>9.90</price> <year>1988</year> </cd></catalog>	<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>

Приведенный пример трактуется следующим образом:

1. **<xsl:stylesheet>** определяет, что данный документ является таблицей стилей XSLT с атрибутами номера версии и пространства имен XSLT.

2. **<xsl:output>**выбирает HTML как выходной формат,

3. **<xsl:template>** указывает, как должны преобразовываться части документа XML. Значение «/» атрибут match используется, чтобы определить шаблон для всего XML документа целиком.

4. **<xsl:value-of>** используется для извлечения значения отобранного XML элемента и добавления его в выходной поток преобразовываемого документа.

5. **<xsl:for-each>** может использоваться для выбора каждого XML элемента заданного узлового набора

6. **<xsl:sort>** используется для сортировки выходных данных и располагается внутри элемента **<xsl:for-each>**

Также могут быть использованы следующие элементы:

1. **<xsl:apply-templates>** применяет некий шаблон к текущему элементу или к дочернему узлу текущего элемента. Если в элемент **<xsl:apply-templates>** добавить атрибут ***select***, то он будет относиться только к дочернему элементу, который соответствует значению этого атрибута и может использоваться для определения порядка, в котором будут обрабатываться дочерние узлы.

2. **<xsl:choose>** используется вместе с элементами **<xsl:when>** и **<xsl:otherwise>**, чтобы определить проверку на выполнение условия. Пример использования элемента выбора с

условием представлен на рис. 13.1. Согласно примеру, если условие выполняется, то ячейка будет выделена цветом *#ff00ff*.

```
<xsl:for-each select="catalog/cd">
  <tr>
    <td><xsl:value-of select="title"/></td>
   <xsl:choose>
     <xsl:when test="price &gt; 10">
       <td bgcolor="#ff00ff">
       <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>
```

Рис. 13.1 Пример использования элементов условия

К атрибутам элемента <xsl:sort> относятся:

– ***select*** — обязательный атрибут, значением которого является выражение, называемое также ключевым выражением. Это выражение вычисляется для каждого узла обрабатываемого множества, преобразуется в строку и затем используется как значение ключа при сортировке. По умолчанию значением этого атрибута является ".", что означает, что в качестве значения ключа для каждого узла используется его строковое значение;

– ***order*** – необязательный атрибут, определяет порядок, в котором узлы должны сортироваться по своим ключам. Этот атрибут может принимать только два значения – "*ascending*", указывающее на восходящий порядок сортировки, и "*descending*", указывающее на нисходящий порядок. Значением по умолчанию является "ascending", то есть восходящий порядок;

– ***lang*** – необязательный атрибут, определяет язык ключей сортировки. В разных языках символы алфавита могут иметь различный порядок, что, соответственно, должно учитываться при сортировке. Атрибут ***lang*** в XSLT может иметь те же самые значения, что и атрибут xml:lang (например: "en", "en-us", "ru" и т. д.).

Если значение этого атрибута не определено, процессор может либо определять язык исходя из параметров системы, либо сортировать строки исходя из порядка кодов символов Unicode;

– ***data-type*** – необязательный атрибут, определяет тип данных, которые несут строковые значения ключей.

Все атрибуты элемента ***xsl:sort*** должны обладать фиксированными значениями.

Задания к лабораторной работе № 13

Задание 1 Оформите задание 1 лабораторной работы № 11 через подключение XSLT с сортировкой по возрастанию.

Задание 2 Создайте новый XML-документ, в котором должна быть информация об аттестации студентов, преобразовав с помощью XSLT в таблицу с условием, при котором ячейки с оценками ниже 4 должны быть выделены красным фоном, а оценки выше 8 зеленым фоном.

Контрольные вопросы

1. Что такое XSL? Чем является XSLT?
2. Каково основное назначение технологии XSLT?
3. Для чего предназначен <xsl:template>?
4. Что означает значение match="/"?
5. Как подключить XSLT к xml?
6. Для чего предназначено <xsl:stylesheet>?
7. Что означает <xsl:apply-templates>?
8. Для чего и какие атрибуты имеет <xsl:sort>?
9. С помощью какого элемента можно осуществить сортировку с условиями?
10. Для чего используется элемент <xsl:otherwise>?
11. В чем заключается предназначение <xsl:when>?
12. Что относится к XSL?
13. Как строятся шаблоны преобразований в XSLT?
14. Каков алгоритм преобразования XML-документа с помощью языка XSLT?
15. Для чего предназначено <xsl:value-of>?
16. Для чего используются <xsl:for-each>? Какие элементы XSL могут быть внутри него?