**Лабораторная работа № 7**

**Основы XML, XSL**

Цель работы­: изучить методику создания XML-документа, язык XSL, который позволяет преобразовать документы XML,  используя язык преобразований XSLT.

XML (Extensible Markup Language) - это язык разметки, описывающий целый класс объектов данных, называемых XML- документами. Этот язык используется в качестве средства для описания грамматики других языков и контроля за правильностью составления документов. Т.е. сам по себе XML не содержит никаких тэгов, предназначенных для разметки, он просто определяет порядок их создания. Кроме того, XML-документы могут выступать в качестве уникального способа хранения данных, который включает в себя одновременно средства для разбора информации и представления ее на стороне клиента. XML позволяет также осуществлять контроль за корректностью данных, хранящихся в документах, производить проверки иерархических соотношений внутри документа и устанавливать единый стандарт на структуру документов, содержимым которых могут быть самые различные данные. Это означает, что его можно использовать при построении сложных информационных систем, в которых очень важным является вопрос обмена информацией между различными приложениями, работающими в одной системе. Создавая структуру механизма обмена информации в самом начале работы над проектом, менеджер может избавить себя в будущем от многих проблем, связанных с несовместимостью используемых различными компонентами системы форматов данных.

В общем случае XML- документы должны удовлетворять следующим требованиям:   
**1)** В заголовке документа помещается объявление XML, в котором указывается язык разметки документа, номер его версии и дополнительная информация.  
**2)** Каждый открывающий тэг, определяющий некоторую область данных в документе обязательно должен иметь своего закрывающего "напарника".

**3)** В XML учитывается регистр символов.  
**4)** Все значения атрибутов, используемых в определении тэгов, должны быть заключены в кавычки.

**5)**Вложенность тэгов в XML строго контролируется, поэтому необходимо следить за порядком следования открывающих и закрывающих тэгов.  
**6)** Вся информация, располагающаяся между начальным и конечными тэгами, рассматривается в XML как данные и поэтому учитываются все символы форматирования (т.е. пробелы, переводы строк, табуляции).

**Пример:** создать **XML-документ,**в котором содержится информация о книгах: автор, название и цена книги.

<?xml version="1.0" encoding="windows-1251"?>            *// декларацияXML*  
<?xml-stylesheet type="text/xsl" href="example"?>   // связывает наш документ   с таблицей стилей XSL

<library>  
<book>  
<author>ЛЕВ ТОЛСТОЙ</author>

<name>"АННА КАРЕНИНА"</name>

<price currency="р.">6000</price>

</book>

<book>

<author>МИХАИЛ ШОЛОХОВ</author>

<name>"ТИХИЙ ДОН"</name>

<price currency="р.">10000</price>

</book>  
</library>

Любой XML- документ должен всегда начинаться с инструкции **<?xml?>**, внутри которой также можно задавать номер версии языка, номер кодовой страницы и другие параметры, необходимые программе-анализатору в процессе разбора документа.

       Обратите внимание на вторую строку документа, где инструкция **xml-stylesheet** связывает наш документ с таблицей стилей XSL. Синтаксис этой инструкции одинаков как для привычных таблиц CSS, так и для XSL, отличаясь только атрибутом type, который приобретает соответственно значения text/css и text/xsl. Атрибут href знаком нам по языку HTML и содержит ссылку на файл с таблицей стилей.

       Набором всех элементов, содержащихся в документе, задается его структура и определяются все иерархические соотношения. Для корневого элемента нашего документа используется **library.**Тэги book, author, name, priceявляются вложенными тегами.  Тэг **book** означает то, что речь пойдёт о книгах; тэг **author**– об авторе книги, тэг **name**– о названии книги, тэг **price**– о цене книги.

Таким образом, если, например, мы считаем, что для обозначения элемента *rose* в документе необходимо использовать тэг *<flower>*;, то XML позволяет свободно использовать определяемый нами тэг и мы можем включать в документ фрагменты, подобные следующему:

<flower>rose</flower>

Набор тэгов может быть легко расширен.

**Элементы данных**

Элемент - это структурная единица XML- документа. Заключая слово rose в в тэги <flower> </flower> , мы определяем непустой элемент, называемый <flower>, содержимым которого является *rose*. В общем случае в качестве содержимого элементов могут выступать как просто какой-то текст, так и другие, вложенные, элементы документа, секции CDATA, инструкции по обработке, комментарии, - т.е. практически любые части XML- документа.

Любой непустой элемент должен состоять из начального, конечного тэгов и данных, между ними заключенных.

Например, следующие фрагменты будут являться элементами:

<flower>rose</flower>

<city>Novosibirsk</city> ,а эти - нет:

<rose>

<flower>

Производя в последствии поиск в этом документе, программа клиента будет опираться на информацию, заложенную в его структуру - используя элементы документа. Т.е. если, например, требуется найти нужный университет в нужном городе, используя приведенный фрагмент документа, то необходимо будет просмотреть содержимое конкретного элемента <university>, находящегося внутри конкретного элемента <city>. Поиск при этом, естественно, будет гораздо более эффективен, чем нахождение нужной последовательности по всему документу.

В XML документе, как правило, определяется хотя бы один элемент, называемый корневым и с него программы-анализаторы начинают просмотр документа. В приведенном примере этим элементом является <country>

В некоторых случаях тэги могут изменять и уточнять семантику тех или иных фрагментов документа, по разному определяя одну и ту же информацию и тем самым предоставляя приложению-анализатору этого документа сведения о контексте использования описываемых данных. Например, прочитав фрагмент<city>Holliwood</city> мы можем догадаться, что речь в этой части документа идет о городе, а вот во фрагменте <restaurant>Holliwood</restaurant> - о ресторане.

В случае, если элемент не имеет содержимого, т.е. нет данных, которые он должен определять, он называется пустым. Примером пустых элементов в HTML могут служить такие тэги HTML, как <br>, <hr>, <img>;. Необходимо только помнить, что начальный и конечные тэги пустого элемента как бы объединяется в один, и надо обязательно ставить косую черту перед закрывающей угловой скобкой (например, <empty/>;)

**Конструкции языка**

Содержимое XML- документа представляет собой набор элементов, секций CDATA, директив анализатора, комментариев, спецсимволов, текстовых данных. Рассмотрим каждый из них подробней.

**Комментарии**

Комментариями является любая область данных, заключенная между последовательностями символов <!-- и --> Комментарии пропускаются анализатором и поэтому при разборе структуры документа в качестве значащей информации не рассматриваются.

**Атрибуты**

Если при определении элементов необходимо задать какие-либо параметры, уточняющие его характеристики, то имеется возможность использовать атрибуты эдлемента. Атрибут - это пара "название" = "значение", которую надо задавать при определении элемента в начальном тэге. Пример:

<color RGB="true">#ff08ff</color>

<color RGB="false">white</color>

или

<author id=0>Ivan Petrov</author>

Примером использования атрибутов в HTML является описание элемента <font>:

<font color=”white” name=”Arial”>Black</font>

**Cпециальные символы**

Для того, чтобы включить в документ символ, используемый для определения каких-либо конструкций языка (например, символ угловой скобки) и не вызвать при этом ошибок в процессе разбора такого документа, нужно использовать его специальный символьный либо числовой идентификатор. Например, &lt; , &gt; &quot; или &#036;(десятичная форма записи), &#x1a (шестнадцатеричная) и т.д. Строковые обозначения спецсиволов могут определяться в XML документе при помощи компонентов (entity), о чем мы еще поговорим немного позже.

**Директивы анализатора**

Инструкции, предназначенные для анализаторов языка, описываются в XML документе при помощи специальных тэгов - *<?*и *?>*;. Программа клиента использует эти инструкции для управления процессом разбора документа. Наиболее часто инструкции используются при определении типа документа (например, <? Xml version=”1.0”?>) или создании пространства имен.

**CDATA**

Чтобы задать область документа, которую при разборе анализатор будет рассматривать как простой текст, игнорируя любые инструкции и специальные символы, но, в отличии от комментариев, иметь возможность использовать их в приложении, необходимо использовать тэги <![CDATA] и ]]>. Внутри этого блока можно помещать любую информацию, которая может понадобится программе- клиенту для выполнения каких-либо действий (в область CDATA, можно помещать, например, инструкции JavaScript). Естественно, надо следить за тем, чтобы в области, ограниченной этими тэгами не было последовательности символов**.**

Как видно, сам процесс создания XML документа очень прост и требует от нас лишь базовых знаний HTML и понимания тех задач, которые мы хотим выполнить, используя XML в качестве языка разметки. Таким образом, у разработчиков появляется уникальная возможность определять собственные команды, позволяющие им наиболее эффективно определять данные, содержащиеся в документе. Автор документа создает его структуру, строит необходимые связи между элементами, используя те команды, которые удовлетворяют его требованиям и добивается такого типа разметки, которое необходимо ему для выполнения операций просмотра, поиска, анализа документа.

Документы XML используют для разделения содержания сайта и его дизайна, так как они хорошо подходят для хранения информации. Ведь главная задача XML - это структурирование информации, а не ее отображение.

**Язык XSL** включает в себя несколько компонентов, но в первую очередь, интересует такая его составная часть, как язык преобразований XSLT (XSL Transformation Language), который позволяет преобразовать документы с одного диалекта XML на другой.

Для того чтобы ознакомиться, как работает эта технология, нам понадобится XML-файл (который описан выше) и файл с набором инструкций для преобразований XSLT, который часто называют таблицей стилей XSL.

**Пример XSL-документа:**

<?xml version="1.0" encoding="windows-1251"?>    *//* декларация XML  
<xsl:stylesheetversion="1.0"xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

       <xsl:template match="library">     *//*создание шаблона для   корневого элемента документа  
          <html>  
          <head>  <title>  Книги  </title> </head>  
         <body>  
          <table border="1" cellspacing="0" cellpadding="3"> <xsl:apply-templates/>

          </table>  
          </body>  
          </html>  
       </xsl:template>

     <xsl:templatematch="book">                    *//* создающий для каждого элемента строку таблицы

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

         </xsl:template>

        <xsl:template match="author | name | price">   *//*для указанных элементов применяется  один и тот же шаблон

         <td> <xsl:apply-templates/>

         <xsl:value-ofselect="@currency" />           *//* извлечение значения  атрибута currency и помещение его в итоговый документ

</td>

</xsl:template>

        <xsl:template match="text()"> <xsl:value-of select="." />  
        </xsl:template>

</xsl:stylesheet>

Язык XSLT сам является подмножеством XML, и поэтому правила оформления таблицы стилей должны подчиняться всем правилам оформления XML-документов. Первой строкой нашего файла как раз и будет декларация XML, но в данном случае нам необходимо указать схему с описанием диалекта языка XSLT, по правилам которого мы составили наш документ. Это указание необходимо для программ, которые будут обрабатывать XML-документ с таблицей стилей, будь то браузеры или XSLT-процессоры, для подтверждения, что мы используем общепринятую схему для данного диалекта.

Строка **<xsl:stylesheetversion="1.0" xmlns:xsl="http://www.w3.org/ 1999/ XSL/Transform">**  указывает, что мы будем использовать схему для создания таблиц стилей xsl:stylesheet, которую можно обнаружить по адресу, указанному в атрибуте xmlns:xsl. Здесь также появляется префикс xsl, который задает название пространства имен, используемых в этой схеме. Он встретится во всех тегах нашей таблицы стилей XSL, указывая на их принадлежность к языку XSLT.

Первым делом, надо создать шаблон для корневого элемента нашего документа **library**, с помощью тега xsl:template с атрибутом match, задающим элемент, для которого шаблон будет применён. В нашем случае задаем в соответствие корневому элементу стандартные блоки HTML-документа: html, head, title, body, а также создаем таблицу для более удобного представления информации, хранящейся в XML-документе. В блоке body мы применили инструкцию, задача которой - передать содержание элемента libraryна обработку остальных шаблонов в нашей таблице стилей. Так как это корневой элемент, содержащий все элементы, то и для них будут применены прописанные далее в таблице шаблоны. По такому же принципу построен шаблон для тега **book**, создающий для каждого элемента строку таблицы.

В следующем шаблоне используем конструкции языка XPath. Его задача - обнаруживать определенные узлы XML по описанию заданному пользователем и передавать их другим инструментам, обрабатывающим документ XML. Благодаря ему мы можем воспользоваться **оператором** "**|**" в качестве логического "ИЛИ" для указания элементов, к которым мы хотим применить один и тот же шаблон. Таким образом, мы создаем один шаблон для элементов author, name и price.

Команда **xsl:value-of** извлекает значение узла XML и помещает его в итоговый документ. Последний шаблон нашего документа обрабатывает текстовое содержание тегов XML, что также указывается с помощью команды XPath text(). Этот шаблон прописывать было не обязательно, так как предыдущий шаблон неявно передает на обработку содержание узлов author, name и price, даже без наших указаний. Тем не менее явная обработка текстовых элементов документа делает наш шаблон более понятным. В команде xsl:value-of в атрибуте select тоже используется команда XPath ".", которая ссылается на текущий элемент. Таким образом, последний шаблон выполняется при нахождении текстовой строки, извлечет ее значение и поместит в результирующий документ. Команда xsl:value-of используется также в шаблоне для элементов author, name, price - для извлечения значения атрибута currency тега price. В языке XPath символ "@" используется для обозначения атрибутов элементов, а выражение **<xsl:value-ofselect= "@currency"/>**дает указание извлечь значение атрибута currency и поместить его в итоговый документ. Так как атрибут currency есть только в теге price, то во время обработки тегов author и name не будет возвращено никаких значений, что позволяет использовать эту инструкцию в одном шаблоне для нескольких тегов.

Результат обработки XML-документа с помощью нашей таблицы стилей XSL можно посмотреть браузерах последнего поколения: браузер показывает источник нашего документа как xml-документ, а в окне браузера он отображается как web-страница - со всей информацией, которую хотели вывести:

**Команды XSL**

# Элемент <xsl:for-each>

Применяет шаблон несколько раз - по очереди к каждому узлу набора.

<xsl:for-each

select = Expression

</xsl:for-each>

# *Пример :*

Файл XSLT в данном примере задает структуру документа вывода. Вывод представляет собой элемент верхнего уровня HTML, содержащий элементы <BODY> и<TABLE>. Таблица содержит по нескольку записей для каждого клиента. Файл XSLT также использует шаблоны для создания элементов <TD> для исходных элементов имени, адреса и телефона.

### **XML-файл (customers.xml)**

<?xml version="1.0"?>

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

<customers>

<customer>

<name>John Smith</name>

<address>123 Oak St.</address>

<state>WA</state>

<phone>(206) 123-4567</phone>

</customer>

<customer>

<name>Zack Zwyker</name>

<address>368 Elm St.</address>

<state>WA</state>

<phone>(206) 423-4537</phone>

</customer>

<customer>

<name>Albert Aikens</name>

<address>368 Elm St.</address>

<state>WA</state>

<phone>(206) 423-4537</phone>

</customer>

<customer>

<name>Albert Gandy</name>

<address>6984 4th St.</address>

<state>WA</state>

<phone>(206) 433-4547</phone>

</customer>

<customer>

<name>Peter Furst</name>

<address>456 Pine Av.</address>

<state>CA</state>

<phone>(209) 765-4321</phone>

</customer>

<customer>

<name>Dan Russell</name>

<address>9876 Main St.</address>

<state>PA</state>

<phone>(323) 321-7654</phone>

</customer>

</customers>

### **XSLT-файл (foreach.xsl)**

<?xml version="1.0"?>

<xsl:stylesheet version="1.0"

xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >

<xsl:template match="/">

<HTML>

<BODY>

<TABLE>

<xsl:for-each select="customers/customer">

<xsl:sort select="state" order="descending"/>

<xsl:sort select="name"/>

<TR>

<TD><xsl:value-of select="name" /></TD>

<TD><xsl:value-of select="address" /></TD>

<TD><xsl:value-of select="phone" /></TD>

</TR>

</xsl:for-each>

</TABLE>

</BODY>

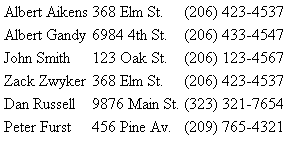
</HTML>

</xsl:template>

</xsl:stylesheet>

### **Вывод**

Далее приведен форматированный вывод.



# Элемент <xsl:if>

Позволяет создавать простые условные фрагменты шаблонов.

<xsl:if

test = boolean-expression

</xsl:if>

# *Пример :*

В этом примере имена в группе имен форматируются в виде списка, разделенного запятой.

## [XML-файл (names.xml)](javascript:void(0))

<?xml version='1.0'?>

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

<namelist>

<name>Albert</name>

<name>Terrance</name>

<name>Will</name>

<name>Sylvia</name>

<name>Timothy</name>

<name>Gordon</name>

<name>James</name>

<name>Robert</name>

<name>Dan</name>

<name>Sasha</name>

</namelist>

## [XSLT-файл (ifcomma.xsl)](javascript:void(0))

<?xml version='1.0'?>

<xsl:stylesheet version="1.0"

xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >

<xsl:template match="namelist/name">

<xsl:apply-templates/>

<xsl:if test="position()!=last()">, </xsl:if>

</xsl:template>

</xsl:stylesheet>

## [Вывод](javascript:void(0))

Далее приведен форматированный вывод.

Albert, Terrance, Will, Sylvia, Timothy, Gordon, James, Robert, Dan, Sasha

# Элемент <xsl:value-of>

Вставляет значение выбранного узла в виде текста.

<xsl:value-of

select = Expression // обязательный атрибут

disable-output-escaping = "yes" | "no" //значения по умолчанию

</xsl:value-of>

# *Пример :*

Приведенный далее файл XSLT создает элемент <p> из элемента <person> с дочерними элементами <given-name> и <family-name>. Элемент <p> будет содержать строковое значение первого дочернего элемента <given-name> текущего узла, за которым будет следовать пробел и строковое значение первого дочернего элемента <family-name> текущего узла.

### **XML-файл (family.xml)**

<?xml version="1.0"?>

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

<family>

<person>

<given-name age="10">Fred</given-name>

<family-name>Smith</family-name>

</person>

<person>

<given-name age="13">Jill</given-name>

<family-name>Jones</family-name>

</person>

</family>

### **XSLT-файл (family.xsl)**

<?xml version='1.0'?>

<xsl:stylesheet version="1.0"

xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="person">

<p>

<xsl:value-of select="given-name"/>

<xsl:text> </xsl:text>

<xsl:value-of select="family-name"/>

</p>

</xsl:template>

</xsl:stylesheet>

### **Вывод**

Далее приведен форматированный вывод.

Fred Smith

Jill Jones

# Элемент <xsl:template>

Определяет повторно используемый шаблон для формирования желаемого вывода для узлов определенного типа в определенном контексте.

<xsl:template

name= Qname

match = Pattern // шаблон, к которому применяется правило

priority = number // номер приоритета для шаблона

mode = QName // значение режима

</xsl:template>

# *Пример :*

У этого правила шаблона есть шаблон, который обнаруживает элементы <stock> и создает в качестве вывода элемент <DIV> с атрибутом STYLE="font-weight:bold":

### **XML-файл (portfolio.xml)**

<?xml version="1.0"?>

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

<portfolio xmlns:dt="urn:schemas-microsoft-com:datatypes" xml:space="preserve">

<stock exchange="nyse">

<name>zacx corp</name>

<symbol>ZCXM</symbol>

<price dt:dt="number">28.875</price>

</stock>

<stock exchange="nasdaq">

<name>zaffymat inc</name>

<symbol>ZFFX</symbol>

<price dt:dt="number">92.250</price>

</stock>

<stock exchange="nasdaq">

<name>zysmergy inc</name>

<symbol>ZYSZ</symbol>

<price dt:dt="number">20.313</price>

</stock>

</portfolio>

### **XSLT-файл (templ.xslt)**

<?xml version='1.0'?>

<xsl:stylesheet version="1.0"

xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="stock">

<DIV STYLE="font-weight:bold">

Symbol: <xsl:value-of select="symbol" />,

Price: <xsl:value-of select="price" />

</DIV>

</xsl:template>

</xsl:stylesheet>

### **Вывод**

Далее приведен форматированный вывод.

Symbol: ZCXM, Price: 28.875

Symbol: ZFFX, Price: 92.250

Symbol: ZYSZ, Price: 20.313

# Элемент <xsl:text>

Создает текстовый узел на основе таблицы стилей. Узлы, содержащие только пробелы, в выводе сохраняются.

<xsl:text

disable-output-escaping = "yes" | "no"> // значения по умолчанию

</xsl:text>

# *Пример :*

### **XML-файл (text.xml)**

<?xml version="1.0"?>

<topic>

<text>First line.</text>

<text>Second line.</text>

<text></text>

</topic>

### **XSLT-файл (text.xsl)**

Данная таблица стилей использует пустые элементы <xsl:text/>, чтобы удалить все символы пробелов (пробелы, переводы строк и символы табулятора), которые были бы созданы правилами шаблонов в противном случае. Результат в виде форматированного вывода:

<?xml version='1.0'?>

<xsl:stylesheet version="1.0"

xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="text"/>

<xsl:template match="/">

<xsl:apply-templates/>

</xsl:template>

<xsl:template match="text">

<xsl:text/>"<xsl:value-of select="."/>"<xsl:text/>

</xsl:template>

</xsl:stylesheet>

# Элемент <xsl:copy>

Копирует текущий узел из источника в вывод.

<xsl:copy

use-attribute-sets = QNames// Разделенный пробелами список наборов атрибутов, заданный как список объектов

</xsl:copy>

# *Пример :*

Следующий код выполняет преобразование идентичности для всего документа. Преобразование идентичности копирует каждый узел источника в вывод, создавая логически эквивалентное дерево. Посимвольная эквивалентность не обеспечивается: сущности при этом подвергаются расширению, а пробелы, не помеченные как существенные, могут быть удалены.

### **XML-файл (booksshort.xml)**

<?xml version="1.0"?>

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

<catalog>

<book id="bk101">

<author>Gambardella, Matthew</author>

<title>XML Developer's Guide</title>

<genre>Computer</genre>

<price>44.95</price>

<publish\_date>2000-10-01</publish\_date>

<description>An in-depth look at creating applications with

XML.</description>

</book>

<book id="bk102">

<author>Ralls, Kim</author>

<title>Midnight Rain</title>

<genre>Fantasy</genre>

<price>5.95</price>

<publish\_date>2000-12-16</publish\_date>

<description>A former architect battles corporate zombies,

an evil sorceress, and her own childhood to become queen of the

world.</description>

</book>

<book id="bk103">

<author>Corets, Eva</author>

<title>Maeve Ascendant</title>

<genre>Fantasy</genre>

<price>5.95</price>

<publish\_date>2000-11-17</publish\_date>

<description>After the collapse of a nanotechnology society

in England, the young survivors lay the foundation for a new

society.</description>

</book>

</catalog>

### **XSLT-файл (identityxfm.xsl)**

<?xml version="1.0"?>

<xsl:stylesheet version="1.0"

xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >

<xsl:template match="/ | @\* | node()">

<xsl:copy>

<xsl:apply-templates select="@\* | node()"/>

</xsl:copy>

</xsl:template>

</xsl:stylesheet>

### **Вывод**

Далее приводится фрагмент форматированного вывода, усеченный справа:

Gambardella, MatthewComputer44.952000-10-01An in-depth look and her own childhood to become queen of the world.Corets, EvaFa

# Элемент <xsl:apply-templates>

Предписывает XSLT-процессору найти и применить соответствующий шаблон в зависимости от типа и контекста каждого выбранного узла.

<xsl:apply-templates

select = Expression // обязательный, исп. для узлов, выбранных с помощью данного выражения, вместо обработки всех узлов

mode = QName // Атрибут mode позволяет обрабатывать элемент согласно его имени

</xsl:apply-templates>

# *Пример :*

Таблица стилей в данном примере преобразует данные о заказчиках из формата XML в элемент HTML <TABLE>. В выводной таблице каждая строка соответствует заказчику, а столбцы содержат его имя, адрес и телефон. Элемент <xsl:sort> сортирует заказчиков по штатам, а внутри каждого штата - по именам.

## [XML-файл (customers.xml)](javascript:void(0))

<?xml version="1.0"?>

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

<customers>

<customer>

<name>John Smith</name>

<address>123 Oak St.</address>

<state>WA</state>

<phone>(206) 123-4567</phone>

</customer>

<customer>

<name>Zack Zwyker</name>

<address>368 Elm St.</address>

<state>WA</state>

<phone>(206) 423-4537</phone>

</customer>

<customer>

<name>Albert Aikens</name>

<address>368 Elm St.</address>

<state>WA</state>

<phone>(206) 423-4537</phone>

</customer>

<customer>

<name>Albert Gandy</name>

<address>6984 4th St.</address>

<state>WA</state>

<phone>(206) 433-4547</phone>

</customer>

<customer>

<name>Peter Furst</name>

<address>456 Pine Av.</address>

<state>CA</state>

<phone>(209) 765-4321</phone>

</customer>

<customer>

<name>Dan Russell</name>

<address>9876 Main St.</address>

<state>PA</state>

<phone>(323) 321-7654</phone>

</customer>

</customers>

## [XSLT-файл (applyt.xsl)](javascript:void(0))

<?xml version="1.0"?>

<xsl:stylesheet version="1.0"

xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >

<xsl:template match="/">

<HTML>

<BODY>

<TABLE border="1" cellspacing="0" cellpadding="2">

<xsl:apply-templates select="customers/customer">

<xsl:sort select="state"/>

<xsl:sort select="name"/>

</xsl:apply-templates>

</TABLE>

</BODY>

</HTML>

</xsl:template>

<xsl:template match="customer">

<TR>

<xsl:apply-templates select="name" />

<xsl:apply-templates select="address" />

<xsl:apply-templates select="state" />

<xsl:apply-templates select="phone" />

<xsl:apply-templates select="phone" mode="accountNumber"/>

</TR>

</xsl:template>

<xsl:template match="name">

<TD STYLE="font-size:14pt font-family:serif">

<xsl:apply-templates />

</TD>

</xsl:template>

<xsl:template match="address">

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

</xsl:template>

<xsl:template match="state">

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

</xsl:template>

<xsl:template match="phone">

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

</xsl:template>

<xsl:template match="phone" mode="accountNumber">

<TD STYLE="font-style:italic">

1-<xsl:value-of select="."/>-001

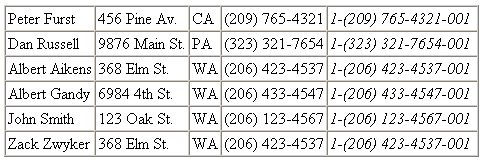
</TD>

</xsl:template>

</xsl:stylesheet>

## [Вывод](javascript:void(0))

Далее приведен форматированный вывод.



# Элемент <xsl:copy>

Копирует текущий узел из источника в вывод.

<xsl:copy

use-attribute-sets = QNames // Указание этого атрибута объявляет каждый атрибут в каждом перечисленном наборе атрибутов.

</xsl:copy>

# *Пример :*

Следующий код выполняет преобразование идентичности для всего документа. Преобразование идентичности копирует каждый узел источника в вывод, создавая логически эквивалентное дерево. Посимвольная эквивалентность не обеспечивается: сущности при этом подвергаются расширению, а пробелы, не помеченные как существенные, могут быть удалены.

### **XML-файл (booksshort.xml)**

<?xml version="1.0"?>

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

<catalog>

<book id="bk101">

<author>Gambardella, Matthew</author>

<title>XML Developer's Guide</title>

<genre>Computer</genre>

<price>44.95</price>

<publish\_date>2000-10-01</publish\_date>

<description>An in-depth look at creating applications with

XML.</description>

</book>

<book id="bk102">

<author>Ralls, Kim</author>

<title>Midnight Rain</title>

<genre>Fantasy</genre>

<price>5.95</price>

<publish\_date>2000-12-16</publish\_date>

<description>A former architect battles corporate zombies,

an evil sorceress, and her own childhood to become queen of the

world.</description>

</book>

<book id="bk103">

<author>Corets, Eva</author>

<title>Maeve Ascendant</title>

<genre>Fantasy</genre>

<price>5.95</price>

<publish\_date>2000-11-17</publish\_date>

<description>After the collapse of a nanotechnology society

in England, the young survivors lay the foundation for a new

society.</description>

</book>

</catalog>

### **XSLT-файл (identityxfm.xsl)**

<?xml version="1.0"?>

<xsl:stylesheet version="1.0"

xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >

<xsl:template match="/ | @\* | node()">

<xsl:copy>

<xsl:apply-templates select="@\* | node()"/>

</xsl:copy>

</xsl:template>

</xsl:stylesheet>

### **Вывод**

Далее приводится фрагмент форматированного вывода, усеченный справа:

Gambardella, MatthewComputer44.952000-10-01An in-depth look and her own childhood to become queen of the world.Corets, EvaFa

# XLink - ссылки в XML

XLink определяет методы создания ссылок внутри XML документов.

### Как используется XLink

* XLink используется для создания гиперссылок внутри XML документов
* Любой элемент XML документа может вести себя как ссылка
* XLink поддерживает простые ссылки (как HTML) и расширенные ссылки (для связывания нескольких ресурсов)
* При помощи XLink ссылки могут определяться за пределами связанных файлов
* XLink является стандартом W3C

### **Синтаксис XLink**

В HTML гиперссылки определяются элементом [<a>](http://msiter.ru/references/html-reference/a). В XML же это работать не будет. В XML документах разрешено использовать элементы с любыми именами, таким образом браузеры не могут знать заранее, какие элементы в XML документах используются в качестве ссылок.

Ниже приводится простой пример использования XLink для создания ссылок в XML документе:

<?xml version="1.0" encoding="UTF-8"?>

<homepages xmlns:xlink="http://www.w3.org/1999/xlink">

<homepage xlink:type="simple" xlink:href="http://msiter.ru">Наш учебный сайт</homepage>

<homepage xlink:type="simple" xlink:href="http://www.w3.org">Консорциум W3C</homepage>

</homepages>

Чтобы иметь возможность использовать XLink, мы должны декларировать пространство имен XLink. Пространством имен XLink является:

"http://www.w3.org/1999/xlink".

* Атрибуты ***xlink:type*** и **xlink:href** в элементе **<homepage>** берутся из пространства имен XLink.
* Атрибут ***xlink:type="simple"*** создает простую "HTML-подобную" ссылку (означает "нажмите сюда, чтобы перейти туда").
* Атрибут ***xlink:href*** определяет URL ссылки.

### **Поддержка XLink браузерами**

Браузеры не имеют поддержки XLink в XML документах. Тем не менее, все основные браузеры поддерживают простые XLink в SVG.

### **Пример XLink**

Следующий XML документ содержит функционал XLink:

<?xml version="1.0" encoding="UTF-8"?>

<bookstore xmlns:xlink="http://www.w3.org/1999/xlink">

<book title="Harry Potter">

<description

xlink:type="simple"

xlink:href="/images/HPotter.gif"

xlink:show="new">

As his fifth year at Hogwarts School of Witchcraft and

Wizardry approaches, 15-year-old Harry Potter is.......

</description>

</book>

<book title="XQuery Kick Start">

<description

xlink:type="simple"

xlink:href="/images/XQuery.gif"

xlink:show="new">

XQuery Kick Start delivers a concise introduction

to the XQuery standard.......

</description>

</book>

</bookstore>

Объяснение:

* В начале документа декларируется пространство имен XLink (***xmlns:xlink="http://www.w3.org/1999/xlink"***)
* Атрибут ***xlink:type="simple"*** создает простую "HTML-подобную" ссылку
* Атрибут ***xlink:href*** определяет URL ссылки (в данном случае - изображение)
* Атрибут ***xlink:show="new"*** определяет, что ссылка должна открываться в новом окне

В примере выше были продемонстрированы простые ссылки XLink. Однако XLink становится интереснее, когда необходимо создать доступ к удаленный ресурсам, а не к отдельным страницам.

Если установить атрибуту ***xlink:show*** значение ***"embed"***, то ресурс, на который создается ссылка, будет встроен в код страницы. Если создается такая ссылка на другой XML документ, то появляется возможность создавать иерархию XML документов.

Также, вы можете определять то, когда ресурс должен отображаться. За это отвечает атрибут ***xlink:actuate***.

### **Атрибуты XLink**

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Значение** | **Описание** |
| xlink:actuate | onLoad onRequest other none | Определяет, когда подключенный ресурс должен быть прочитан и показан:   * onLoad – ресурс должен быть загружен и показан сразу после загрузки документа * onRequest – ресурс загружается и показывается только при нажатии на ссылку |
| xlink:href | URL | Определяет URL ссылки |
| xlink:show | embed new replace other none | Определяет, где открывать ссылку. По молчанию действует "replace" |
| xlink:type | simple extended locator arc resource title none | Определяет тип ссылки |

**Пример выполнения лабораторного задания**

Составить 2 XML-документа.  В одном описать, магазин телефонов с его составляющими: а) название фирмы, чья продукция в магазине, б) вид телефонов, в) модель; во втором – каталог цен. Данные свести в таблицы. Документы соединяются при помощи ссылки.

Мобильные телефоны: марка, модель, цвет / полифония, камера, каталог цен.

**XML-документ**<?xml version="1.0" encoding="utf-8"?>

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

<shop>

<phone>

<brand>SAMSUNG</brand>

<model>s3</model>

<color>blue</color>

<camera>8Mpx</camera>

<price currency="р.">6000</price>

</phone>

<phone>

<brand>NOKIA</brand>

<model>Lumia</model>

<color>Red</color>

<camera>1Mpx</camera>

<price currency="р.">100</price>

</phone>

<phone>

<brand>iPhone</brand>

<model>5</model>

<color>Silver</color>

<camera>4Mpx</camera>

<price currency="р.">90000000000000</price>

</phone>

</shop>

**XSL-документ**

<?xml version="1.0" encoding="utf-8"?>

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

<xsl:template match="shop">

<html>

<head> <title> Каталог телефонов </title> </head>

<body style= "background: lavender;font-family:scada">

<center> Каталог телефонов</center>

<center> <table border="1" cellspacing="0" cellpadding="3">

<tr><th>Марка</th>

<th>Модель</th>

<th>Цвет</th>

<th>Камера</th>

<th>Цена</th>

</tr>

<xsl:apply-templates/>

</table>

</center>

</body>

</html>

</xsl:template>

<xsl:template match="phone">

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

</xsl:template>

<xsl:template match="brand | model | color | camera | price">

<td>

<xsl:apply-templates/>

<xsl:value-of select="@currency" />

</td>

</xsl:template>

<xsl:template match="text()"> <xsl:value-of select="." />

</xsl:template>

</xsl:stylesheet>

**Варианты заданий.**

Составить 2 XML-документа.  В одном описать, магазин одежды с её составляющими: а) название фирмы, чья одежда продаётся в магазине, б) вид одежды, в) модель; во втором – каталог цен. Данные свести в таблицы. Документы соединяются при помощи ссылки.

1. Магазин одежды: город, улица, фирма  / вид, модель, каталог цен.
2. Автомобили: марка, год выпуска, модель / цвет, салон, каталог цен.
3. Игрушки: страна, производитель, магазин, / материал, наименование, каталог цен.
4. Библиотека: автор книги, название, количество страниц / издательство, тираж, год издания книги.
5. Рестораны: улица, название, кухня / зал, меню, каталог цен.
6. Мобильные телефоны: марка, модель, цвет / полифония, камера, каталог цен.
7. Бытовая техника: вид, фирма, модель / крупногабаритные, мелкогабаритные, каталог цен.
8. Строительные материалы: вид, страна, фирма / внутренние работы, наружные работы, каталог цен.
9. Канцелярские товары: вид, фирма, страна / назначение, материал, каталог цен.
10. Продукты питания: магазин, наименование, производитель / группы продтоваров, состав, каталог цен.
11. Мебель: вид, страна, фирма / ткань, цвет, каталог цен.
12. Цветы: название, место произрастания, размеры, / сорт, цвет, каталог цен.
13. Музыкальные инструменты: магазин, фирма, наименование / духовые, струнные, клавишные.
14. Спортивный инвентарь: вид спорта, страна, изготовитель / материал, наименование, каталог цен.
15. Ювелирный магазин: город, название магазина, наименование / изделие, вес, каталог цен.
16. Туристические фирмы: название, направление, вид транспорта / отель, питание, каталог цен.
17. Салон красоты: название, страна, адрес / виды услуг, продолжительность сеанса, каталог цен.
18. Животный мир: континент, страна, название / вес, кожный покров, тип питания.
19. Студенты: вуз, факультет, группа / год рождения, ФИО, средний балл.
20. Киноактёры: страна, фильм, кассовый сбор / роль, премия (Оскар), гонорар.

**Контрольные вопросы.**

**1.**Охарактеризуйте элементы XML- документа, секции CDATA, инструкции по обработке, комментарии, атрибуты, директивы анализатора?

**2.**Какие главные отличия XMLот HTML?

**3.**Как создать шаблон для корневого элемента документа?

**4.**Что нужно сделать, чтобы преобразовать XML–документ по XSL-таблице?

**5.**Как происходит процесс преобразования на стороне клиента?

**6.**Как происходит процесс преобразования на стороне сервера?

**7.**Способ использования SAX(SimpleAPIforXML)?

**8**. Возможности XSLT?

**Литература**

* XML для профессионалов. Дидье Мартин, и др. Серия "Программист - программисту". Изд. "Лори", 2001.
* XML. Питц-Моултис Н., Кирк Ч. СПб.:BHV - Санк-Петербург, 2000.
* Технология XSLT. Алексей Валиков. СПб.:БХВ - Санк-Петербург, 2002.
* Microsoft® XML Core Services (MSXML) 4.0 SDK
* <https://msdn.microsoft.com/ru-ru/library/ms256165.aspx>
* http://vfpdev.narod.ru/docs/xmlint\_r.html