# Лабораторная работа №4

**Форматирование и преобразование XML документа с помощью XSL. XSLT преобразование XML документа.**

Цель работы

Знакомство с методами форматирования и преобразования XML документов на основе XSLT преобразований.

Теоретический материал

## Как используется XML

XML используется во многих аспектах веб-разработки, но основная его задача — облегчение хранения и передачи данных.

XML отделяет данные от HTML

Если вам в HTML документе необходимо отображать динамические данные, то это будет занимать слишком много времени, если всякий раз, когда эти данные изменились, редактировать сам HTML документ.

С XML данные можно хранить в отдельных файлах XML. При этом вы сосредотачиваетесь на использовании HTML/CSS для отображения и шаблонизации и можете быть уверены, что поступающие новые данные не потребуют каких-либо изменений в коде HTML документа.

XML упрощает распределение данных

В реальном мире компьютерные системы и базы данных используют данные в несовместимых форматах.

XML данные хранятся в простом текстовом формате. Это обеспечивает программную и аппаратную независимость.

XML упрощает передачу данных

Одной из самых время затратных проблем разработчиков всегда была и остается до сих пор проблема обмена данными между несовместимыми между собой системами.

Передача данных в виде XML значительно снижает сложность этой проблемы, так как данные в этом формате могут быть прочитаны разными несовместимыми приложениями.

XML упрощает модификацию платформы

Переход на новые системы (аппаратные или программные платформы) всегда занимает много времени. Множество данных необходимо конвертировать в новые форматы. При этом часто несовместимые данные теряются.

XML данные хранятся в текстовом формате. Это значительно облегчает расширение или модернизацию операционных систем, переход на новые приложения или браузеры без опасности потерять данные.

XML делает ваши данные более доступными

Доступ к вашим данным могут получать не только HTML документы, но и любые другие приложения.

Благодаря XML ваши данные становятся доступными для всех видов "машин чтения" (голосовых машин, новостных каналов и т.д.), что позволяет значительно проще обращаться к ним людям с ограничениями по зрению и другими физическими проблемами.

## Как используется XSL

**XSL** (e**X**tensible **S**tylesheet **L**anguage) — семейство рекомендаций [консорциума W3C](https://ru.wikipedia.org/wiki/W3C), описывающее языки преобразования и визуализации [XML](https://ru.wikipedia.org/wiki/XML)-документов. Состоит из трех частей:

* XSL Transformations ([XSLT](https://ru.wikipedia.org/wiki/XSLT)) — язык преобразований XML-документов.
* XSL Formatting Objects ([XSL-FO](https://ru.wikipedia.org/wiki/XSL-FO)) — [язык разметки](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8) типографских макетов и иных предпечатных материалов.
* [XPath](https://ru.wikipedia.org/wiki/XPath) — язык путей и выражений, используемый в XSLT для доступа к отдельным частям XML-документа.

***XSLT*** можно определить следующим образом:

* XSLT обозначает *XSL Transformations*.
* XSLT является самой важной частью XSL преобразования
* XSLT использует XPath для перемещения по структуре XML документа.
* XSLT является W3C рекомендацией.

## XSL-трансформация

С помощью XSL можно трансформировать XML-документ в любой вид, будь то HTML, WML, RTF, PDF, SQL, или даже SWF (flash-запускаемый модуль), а также XSL. XSL несёт в себе информацию о том, как будет оформлен документ, где и как должны располагаться данные.

XSLT используется для преобразования XML документа в другой XMLдокумент либо в документ другого распознаваемого браузерами типа, например HTML или XHTML. Обычно XSLT делает это, преобразуя каждый XML элемент в соответствующий ему (X)HTML элемент.

С помощью XSLT можно добавлять или удалять элементы и атрибуты в результирующем документе. Также возможна перегруппировка и сортировка элементов, фильтрация элементов при отображении и многое другое.

Подавляющее большинство браузеров имеют поддержку XML и XSLT**.**

## Импорт XML документов в БД MS Access

Для того чтобы импортировать несколько XML документов в одну таблицу базы данных MS Access необходимо выполнить следующие действия:

1) Файл – Внешние данные – Импорт

2) В строке Тип файлов указать XML (\*.xml, \*.xsd)

3) Выбрать необходимую XSD схему

4) В появившемся окне «Импорт документа» нажать OK.

Для импорта XML документов необходимо повторить вышеописанные действия, только в окне «Импорт документа» нажать Параметры и выбрать параметры импорта – Добавить данные в таблицы.

**Примечание: при создании xsl-файла в Oxygen XML Editor выбирайте тип документа XSLT Stylesheet .**

## Объявление XSL

Корневым элементом, указывающим на то, что документ является XSL таблицей стилей является следующий:

*<xsl:stylesheet>*

или полностью равноценный ему

*<xsl:transform>*

В соответствии с W3C XSLT Рекомендацией корректный способ объявления таблицы стилей XSL выглядит следующим образом:

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

или

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

Для того чтобы XSLT элементы, атрибуты и характеристики были доступны в начале документа необходимо объявить пространство имен XSLT:

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

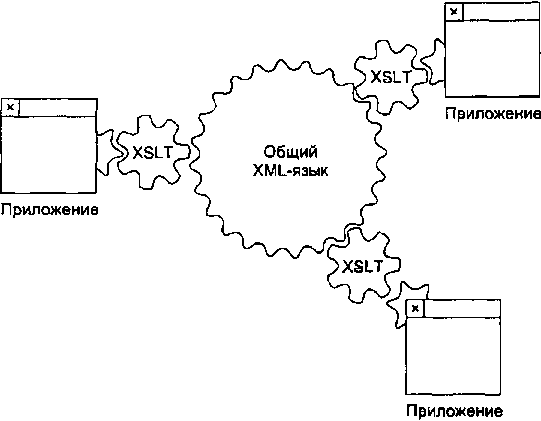
указывающее на официальное пространство имен W3C XSLT. При этом также следует указать атрибут *version="1.0"*.

**Реализация преобразования с помощью сценария**

XSLT преобразование из XML в XHTML выполняемое самими браузерами на основе таблицы стилей XSL является не всегда желательным, поскольку может поддерживаться не всеми браузерами.

Использование в качестве альтернативы JavaScript позволяет:

* Выполнять проверку типа браузера
* Использовать подходящие таблицы стилей в зависимости от типа браузера и потребностей пользователей.



## Порядок выполнения лабораторной работы

**1.** Рассмотрим простой пример XML-файла (1.xml).

<?xml version="1.0" encoding="UTF-8"?>  
<?xml-stylesheet type='text/xsl' href='1.xsl'?>  
<tutorial>  
 <title>"Заметки об XSL"</title>  
 <chapter> Введение в CSP </chapter>  
</tutorial>

И создадим XSL-файл 1.xsl. Текст файла приведен ниже.

<?xml version="1.0" encoding="UTF-8"?>  
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"  
 version="1.0">  
 <xsl:template match="/">  
 <p><strong><xsl:value-of select="//title"/></strong></p>  
 <p><xsl:value-of select="//chapter"/></p>  
 </xsl:template>  
</xsl:stylesheet>

Рассмотрим теперь текст XSL-файла

Легко заметить, что порядок вывода строк у нас определяется только содержанием шаблона преобразования - XSL-файла. При необходимости шаблон можно легко поменять, абсолютно не меняя наш основной XML-файл.

Перейдем теперь 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.

**2.** Рассмотрим следующий XML-файл - 2.xml. Текст его приведен ниже.

<?xml version="1.0" encoding="UTF-8"?>  
<?xml-stylesheet type='text/xsl' href='2.xsl'?>  
<tutorial>  
 <enimals>  
 <dogs>  
 <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>

Выведем на экран соответствующую таблицу с помощью xsl.

В этот файл добавлен шаблон преобразования 2.xsl.

Рассмотрим этот шаблон подробнее. Вот его текст.

<?xml version="1.0" encoding="UTF-8"?>  
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"  
 version="1.0">  
 <xsl:template match="/">  
 <table border="1">  
 <tr bgcolor="#CCCCCC">  
 <td align="center"><strong>Кличка</strong></td>  
 <td align="center"><strong>Вес</strong></td>  
 <td align="center"><strong>Цвет</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 select="dogWeight/@caption"/></td>  
 <td><xsl:value-of select="dogColor"/></td>  
 </tr>  
 </xsl:for-each>  
 </table>  
 </xsl:template>  
</xsl:stylesheet>

Первая строка касается кодировки. Следующие две строки шаблона являются уже привычными. Следующие шесть строк - это строка, содержащая заголовки столбцов таблицы. А вот десятая строка тоже является новой:

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

Конструкция for-each применяется для повторяющихся элементов.

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.

## ПРИМЕРЫ:

1. **Реализация XSLT преобразования XML документа в XHTML средствами браузера.**
2. Создайте файл *ttable.xml* следующего содержания:

|  |
| --- |
| <?xml version="1.0"?>  <?xml-stylesheet type="text/xsl" href="ttable.xsl"?>  <timetable>  <lesson>  <timeFrom>12.10</timeFrom>  <timeTo>13.45</timeTo>  <subject>Deutsch</subject>  <teacher>Borisova</teacher>  </lesson>  <lesson>  <timeFrom>13.55</timeFrom>  <timeTo>15.30</timeTo>  <subject>Administration</subject>  <teacher>Petrov</teacher>  </lesson>  <lesson>  <timeFrom>15.40</timeFrom>  <timeTo>17.15</timeTo>  <subject>Administration</subject>  <teacher>Ivanov</teacher>  </lesson>  <lesson>  <timeFrom>17.25</timeFrom>  <timeTo>19.00</timeTo>  <subject>Web-technology</subject>  <teacher>Loktev</teacher>  </lesson>  </timetable> |

1. Подготовьте для него соответствующий файл таблицы стилей (*ttable.xsl*):

|  |
| --- |
| <?xml version="1.0" encoding="ISO-8859-1"?>  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  <xsl:template match="/">  <html>  <body>  <h2>My academical timetable</h2>  <table border="1">  <tr bgcolor="#a5abfa">  <th align="left">Time From</th>  <th align="left">Time To</th>  <th align="left">Subject</th>  </tr>  <xsl:for-each select="timetable/lesson">  <tr>  <td><xsl:value-of select="timeFrom"/></td>  <td><xsl:value-of select="timeTo"/></td>  <td><xsl:value-of select="subject"/></td>  </tr>  </xsl:for-each>  </table>  </body>  </html>  </xsl:template>  </xsl:stylesheet> |

Первая строка в файле *ttable.xml* вида:

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

представляет собой ссылку на подходящую таблицу стилей.

Атрибут *match* применяется для связывания XML элемента с шаблоном. Значение атрибута match является выражение *XPath*. В данном случает *match="/"* указывает на весь документ.

Содержимое элемента <*xsl:template*> описывает фрагмент HTML кода в выходном документе. Элемент <*xsl:value-of*> может быть использован для извлечения значения XML элемента и добавления его в выходной поток при преобразовании. Значение атрибута *select* является выражением *XPath*, которое напоминает нотацию, используемую в файловой системе; знак (**/**) указывает на обращение к подкаталогу.

Элемент <*xsl:for-each*> позволяет выбирать каждый элемент XML указанного множества узлов.

1. При наличии подходящего браузера можно будет увидеть следующий результат после загрузке файла *ttable.xml*:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| My academical timetable  |  |  |  | | --- | --- | --- | | **Time From** | **Subject** | **Teacher** | | 09.30 | Deutsch | Borisova | | 11.10 | SAP Administration | Petrov | | 12.40 | SAP Administration | Ivanov | | 14.00 | Wen-technology | Loktev | |

**Сортировка**

Для фильтрации вывода элементов можно добавить атрибут элемента <*xsl:for-each*>, задающий критерий отбора элементов и сортировки. Например:

<xsl:for-each select="timetable/lesson[параметр='значение']">

*Например: [subject='SAP Administration'].*

*<xsl:sort select="timeFrom" order="ascending(descending)"/> (ставится после* <xsl:for-each>*)*

<!--аscending = по возрастанию, descending = по убыванию--!>

Проверьте работу данного фильтра.

Допустимо использование следующих операторов для описания фильтра:

* **=** (равно)
* **!=** (не равно)
* **&lt;** (меньше чем)
* **&gt;** (больше чем)

1. **Реализация преобразования с помощью *JavaScript*.**
2. Подготовьте *html* файл, содержащий код:

|  |
| --- |
| <html>  <head>  <script>  function loadXMLDoc(fname)  {  var xmlDoc;  // Код для IE  if (window.ActiveXObject)  {  xmlDoc=new ActiveXObject("Microsoft.XMLDOM");  }  // Код для Mozilla, Firefox, Opera и др.  else if (document.implementation  && document.implementation.createDocument)  {  xmlDoc=document.implementation.createDocument("","",null);  }  else  {  alert('Your browser cannot handle this script');  }  xmlDoc.async=false;  xmlDoc.load(fname);  return(xmlDoc);  }  function displayResult()  {  xml=loadXMLDoc("ttable.xml");  xsl=loadXMLDoc("ttable.xsl");  // Код для IE  if (window.ActiveXObject)  {  ex=xml.transformNode(xsl);  document.getElementById("example").innerHTML=ex;  }  // Код для Mozilla, Firefox, Opera и др.  else if (document.implementation  && document.implementation.createDocument)  {  xsltProcessor=new XSLTProcessor();  xsltProcessor.importStylesheet(xsl);  resultDocument = xsltProcessor.transformToFragment(xml,document);  document.getElementById("example").appendChild(resultDocument);  }  }  </script>  </head>  <body id="example" onLoad="*displayResult()*">  </body>  </html> |

Функция *loadXMLDoc*() загружает XML и XSL файлы в зависимости от типа браузера.

Функция *displayResult*() используется для отображения XML файла в стиле, задаваемом XSL файлом. Она выполняет следующие действия:

* загружает XML и XSL.
* определяет тип браузера.
* если браузер поддерживает *ActiveX* объекты:
* с помощью метода *transformNode*() таблица стилей XSL применяется к XML документу.
* формируется тело текущего документа.
* если браузер клиента не поддерживает *ActiveX* объекты:
* создается новый объект *XSLTProcessor* и в него импортируется XSL файл.
* с помощью метода *transformToFragment*() таблица стилей XSL применяется к XML документу.
* формируется тело текущего документа.

1. Загрузите этот документ в веб-браузере.