УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНОЙ РАБОТЫ ПО ТЕМЕ "XML-ТЕХНОЛОГИЯ"

Содержание

Краткое введение

XML (eXtensible Markup Language) предназначен не для форматирования и отбражения данных, а для описания данных. В отличие от HTML, в XML, в общем случае, теги не определены и при создании XML -файлов автор создает свои собственные теги. Язык XML был разработан в целях преодоления ограничений языка HTML, в частности для усовершенствования возможностей создания динамического наполнения и управления им.

Тело документа XML состоит из элементов разметки (markup) и непосредственно содержимого документа - данных (content). XML - тэги предназначены для определения элементов документа, их атрибутов и других конструкций языка.

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

Правила создания XML- документа.

В общем случае XML- документы должны удовлетворять следующим требованиям:

- в заголовке документа помещается объявление XML, в котором указывается язык разметки документа, номер его версии и дополнительная информация;
- каждый открывающий тэг, определяющий некоторую область данных в документе обязательно должен иметь своего закрывающего "напарника", т.е., в отличие от HTML, нельзя опускать закрывающие тэги;
- в XML учитывается регистр символов;
- все значения атрибутов, используемых в определении тэгов, должны быть заключены в кавычки;
- вложенность тэгов в XML строго контролируется, поэтому необходимо следить за порядком следования открывающих и закрывающих тэгов;
- 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 не игнорируются, как в HTML).

Для того, чтобы использовать данные, определяемые элементами XML, например, отображать их на экране пользователя, необходимо использовать программу-анализатор (XML-парсер браузера), стилевые таблицы или сценарий написанный на скрипт-языках. В данной работе для преобразования XML документов используются разные методы, реализованные Java Script, XSL(XSLT/XPath), SAX-PHP.

Цель: понимание технологии XML.

Общее задание: освоить навыки создания и преобразования XML документов.

Рекомендации: перед выполнением работы необходимо ознакомиться с основами Java Script, XML, PHP, используя лекции, техническую литературу, интернет-ресурсы. Работу следует выполнять последовательно, следуя изложенным ниже примерам. Все примеры проверены на программной платформе (Apache-PHP4). Результаты представить в виде HTML-страницы с гиперссылками, при активации которых, вызываются различные примеры.

1. Преобразование XML-документа с помощью сценариев Java Script.

Задание: модифицировать приведенный пример, внося различные изменения в коды программ.

В отличии от HTML, XML никак не определяет способ отображения и использования описываемых с его помощью элементов документа, т.е. программе-анализатору предоставляется возможность самой выбирать нужное оформление. Этого требует принцип независимости определения внутренней структуры документа от способов представления этой информации.
Для того, чтобы использовать данные, определяемые элементами XML, например, отображать их на экране пользователя, необходимо написать программу-анализатор, которая бы выполняла эти действия. Если заданные в документе конструкции языка являются синтаксически правильными, то программа-анализатор сможет правильно извлечь определяемые ими элементы документа и передать их прикладной программе, выполняющей необходимые действия по отображению. При этом в некоторых анализаторах способ представления структуры документа основывается на спецификации DOM. Если на Вашем компьютере установлен броузер Internet Explorer, можно использовать встроенный в этот броузер XML- анализатор msxml в сценариях, написанных на Java Script.

Перед тем, как использовать свойства и методы анализатора, его необходимо создать. Делается это при помощи стандартного метода, предназначенного для создания ActiveX- объектов:
var mydoc = new ActiveXObject("msxml");
В результате выполнения этой функции переменной mydoc будет присвоен объект, имеющий тип msxml, свойства и методы которого используются в дальнейшем для получения доступа к структуре XML- документа.
Объектная модель XML- анализатора Microsoft может быть представлена в виде следующего набора внутренних объектов: XML Document, XML Element и Element Collection. Объект XML Document содержит свойства и методы, необходимые нам для работы с XML- документом в целом. XML Element отвечает за работу с каждым из элементов XML- документа. Element Collection представляет из себя набор элементов, доступ к которым доступен при помощи имени или порядкового номера. В следующем примере приводится использование для отображения XML документа сценариев на языке JavaScript.

XML-файл (ok.xml)

```
<?xml version="1.0"?>
<journal>
<title>isb2002 is very good group</title>
<contacts>
<address>isb2002</address>
<tel>8-3272-029016</tel>
```

<email>isb2002@tk.kz</email>
<url>www.tk.kz</url>
</contacts>
<authors-list>
<author ID="1">
<firstname>Saule</firstname>
<lastname>Lastname</lastname>
<email>Saule@tk.kz</email>
</author>
<author ID="2">
<firstname>Gauhar</firstname>
<lastname>Lastname</lastname>
<email>Gauhar@tk.kz</email>
</author>
<author ID="3">
<firstname>Kairat</firstname>
<lastname>Lastname</lastname>
<email>Kairat@tk.kz</email>
</author>
</authors-list>
</journal>

HTML-файл с JS

<html><body bgcolor="beige">
<head>
<script language="javascript">
<!--
var xmldoc = new ActiveXObject("msxml");
var xmlsrc = "ok.xml";
function viewTitle(elem){ // **Отображение заголовка документа, определяемого элементом <title>**

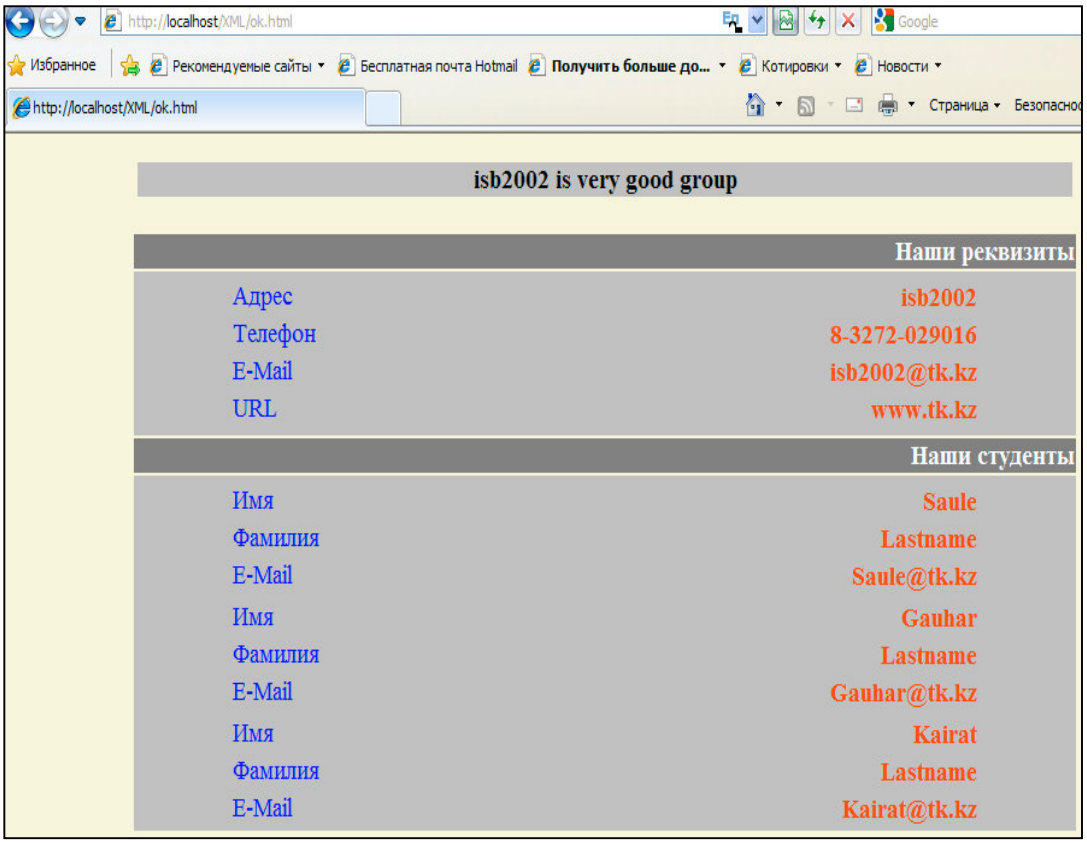
```
this.document.writeln('<center><table width="100%" border=0><tr><td
width="100%" align="center" bgcolor="silver"><b><font
color="black">'+elem.text+'</font></b></td></tr></table></center><br>');
}
function viewContactsList(elem){ // Отображение содержимого дочерних
элементов <author-list>
this.document.writeln('<tr><td align="right" colspan="2"
bgcolor="gray"><b><font color="white">Наши
реквизиты</font></b></td></tr>');
this.document.writeln('<tr><td bgcolor="silver" colspan="2"><center><table
width="80%" border=0>');
if(elem.type==0){
if(elem.children!=null){
this.document.writeln('<tr><td colspan=2 width="100%"> </td></tr>');
var cur_item=elem.children.item("address");
if(cur_item!=null){
this.document.writeln('<tr><td><font color="blue">Адрес</font></td><td
align="right" ><b><font color="red">'+cur_item.text+'</font></b></td></tr>');
}
var cur_item=elem.children.item("tel",0);
if(cur_item!=null){
this.document.writeln('<tr><td><font color="blue">Телефон</font></td><td
align="right" ><b><font color="red">'+cur_item.text+'</font></b></td></tr>');
}
var cur_item=elem.children.item("email");
if(cur_item!=null){
this.document.writeln('<tr><td><font color="blue">E-Mail</font></td><td
align="right"><b><font color="red">'+cur_item.text+'</font></b></td></tr>');
}
var cur_item=elem.children.item("url");
if(cur_item!=null){
this.document.writeln('<tr><td><font color="blue">URL</font></td><td
align="right"><b><font color="red">'+cur_item.text+'</font></b></td></tr>');
}
```

```
}
}
this.document.writeln('<tr><td colspan=2 width="100%"> </td></tr>');
this.document.writeln('</table></center></td></tr>');
}
function viewAuthorsList(elem){ // Отображение содержимого дочерних элементов <author-list>
this.document.writeln('<tr><td align="right" colspan="2"
bgcolor="gray"><b><font color="white">Наши
студенты</font></b></td></tr>');
this.document.writeln('<tr><td bgcolor="silver" colspan="2"><center><table
width="80%" border=0>');
if(elem.type==0){
if(elem.children!=null){
for(i=0;i<elem.children.length;i++){
var cur_author = elem.children.item("author",i);
this.document.writeln('<tr><td colspan=2 width="100%"> </td></tr>');
if(cur_author.children!=null){
var cur_item=cur_author.children.item("firstname");
if(cur_item!=null){
this.document.writeln('<tr><td><font color="blue">Имя</font></td><td
align="right" ><b><font color="red">'+cur_item.text+'</font></b></td></tr>');
}
var cur_item=cur_author.children.item("lastname");
if(cur_item!=null){
this.document.writeln('<tr><td><font color="blue">Фамилия</font></td><td
align="right" ><b><font color="red">'+cur_item.text+'</font></b></td></tr>');
}
var cur_item=cur_author.children.item("email");
if(cur_item!=null){
this.document.writeln('<tr><td><font color="blue">E-Mail</font></td><td
align="right"><b><font color="red">'+cur_item.text+'</font></b></td></tr>');
}
}
```

```
}
}
}
this.document.writeln('</table></center></td></tr>');
}
function viewError(){
this.document.writeln('<center><hr>Error was detected');
}
function parse(root){
if(root==null) return;
var i=0;
var elem;
if(root.children!=null){ // Если вложенные элементы не были
определены, то свойство children будет установленно в null
this.document.writeln('<center><table width="80%" border=0><tr><td>');
// Перебор дочерних элементов
for(i=0;i<root.children.length;i++){
elem=root.children.item(i);
if(root.children.item(i).tagName=="TITLE"){
viewTitle(elem); // Разбор подэлементов <title>
}
if(elem.tagName=="CONTACTS"){
viewContactsList(elem); // Разбор подэлементов <contacts>
}
if(elem.tagName=="AUTHORS-LIST"){
viewAuthorsList(elem); // Разбор подэлементов <authors-list>
}
}
this.document.writeln('</td></tr></table>');
}
}
function viewDocument(){
xmldoc.URL = xmlsrc; // Загрузка XML документа
this.document.writeln('<body bgcolor="white">');
```

```
parse(xmldoc.root); // Начало разбора документа
this.document.writeln('</body>');
}
// Генерирование страницы
viewDocument();
//-->
</script>
</head>></body></html>
```

Результат html-файла:



2. Преобразование XML-документа с помощью стилевых таблиц XSL.

Задание: создать исходный XML - *документ отличный от приведенного примера, модифицировать приведенные примеры отображения и разбора* XML - *документа, внося различные изменения в коды программ.*

В предыдущем разделе для вывода элементов XML- документа на экран броузера мы применяли Java Script-сценарии. Однако, в ряде случаев, для этих целей предпочтительней использование специально предназначенного для этого средства - стилевых таблиц XSL(Extensible Stylesheet Language). Принцип

обработки XML- документов стилевыми таблицами заключается в следующем: при разборе XSL-документа программа-анализатор обрабатывает инструкции этого языка и каждому элементу, найденному в XML- дереве ставит в соответствие набор тэгов, определяющих форматирование этого элемента. Другими словами, мы задаем шаблон форматирования для XML- элементов. Инструкции XSL определяют точное месторасположение элемента XML в дереве, поэтому существует возможность применять различные стили оформления к одинаковым элементам, в зависимости от контекста их использования. В XSL входят XSLT(Extensible Stylesheet Language Transformations), Xpath. XSLT-язык преобразования XML-документа в другие документы. Такое преобразование, в общем случае, позволяет выделить нужную часть информации и представить ее в удобном для чтения виде. Xpath - язык определения частей и путей к элементам XML. Рассмотрим примеры использования XSL. Важные части кодов выделены и даны необходимые краткие комментарии.

XML-XSL(XSLT)

Исходный XML-файл

```
<?xml version="1.0" encoding="windows-1251"?>
<?xml-stylesheet type="text/xsl" href="имя файла *.xsl"?>
<IIT>
<kafedra id="1">
<title>TK</title>
<characters>
<character>
<spec>Информационные системы</spec>
<group>ИСб</group>
</character>
<character>
<spec>ВТ и ПО</spec>
<group>КСУ</group>
</character>
</characters>
<plot>
Техническая кибернетика
</plot>
</kafedra>
<kafedra id="2">
```

<title>ПОС</title>
<characters>
<character>
<spec>Информатика</spec>
<group>ИНб</group>
</character>
<character>
<spec>ВТ и ПО</spec>
<group>ПОС</group>
</character>
</characters>
<plot>
Программное обеспечение систем
</plot>
</kafedra>
</IIT>

XSL(XSLT) файлы

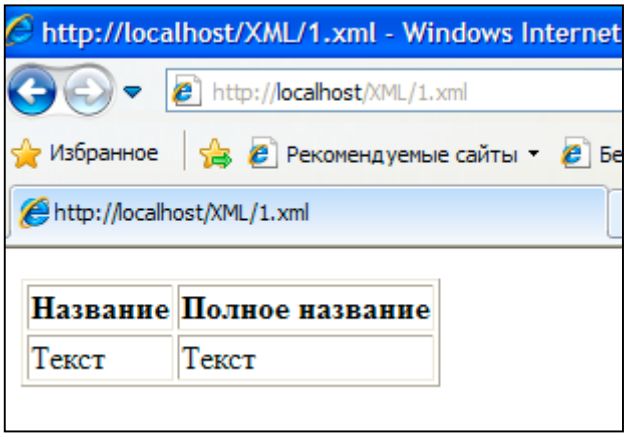
1.xsl (вывод шаблона)

```
<?xml version="1.0" encoding="windows-1251"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html> <body>
<table border="1">
<tr>
<th>Название</th> <th>Полное название</th>
</tr><tr>
<td> Текст </td>
<td> Текст </td>
</tr>
</table></body> </html>
</xsl:template>
```

</xsl:stylesheet>

Выделенные тэги соответственно обозначают : 1)декларация XML, 2) начало таблицы стилей, задается пространство имен,3) тэг связывает шаблон и корень оригинального XML-документа.

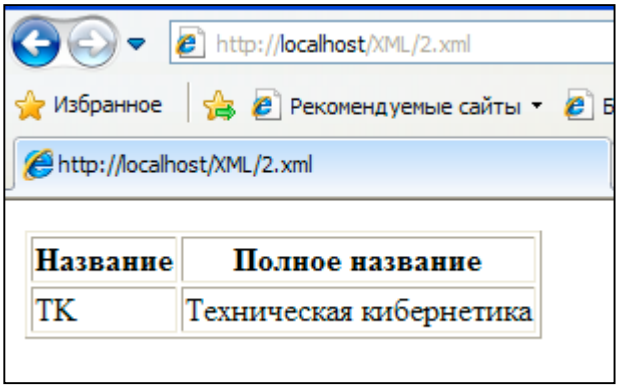
Результат 1.xsl:



2.xsl (Вывод содержимого элемента)

```
<?xml version="1.0" encoding="windows-1251"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html> <body>
<table border="1">
<tr>
<th>Название</th> <th>Полное название</th>
</tr>
<tr>
<td><xsl:value-of select="IIT/kafedra/title"/></td>
<td><xsl:value-of select="IIT/kafedra/plot"/></td>
</tr>
</table>
</body> </html>
</xsl:template>
</xsl:stylesheet>
```

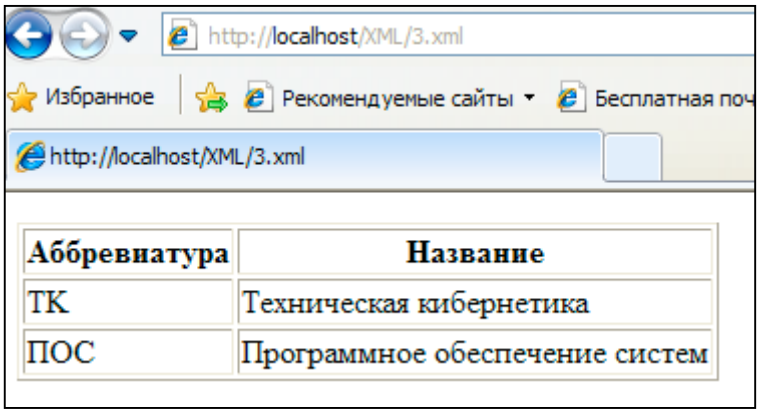
Результат 2.xsl :



3.xsl (цикл)

```
<?xml version="1.0" encoding="windows-1251"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<table border="l">
<tr>
<th>Аббревиатура</th> <th>Название</th>
</tr>
<xsl:for-each select="IIT/kafedra">
<tr>
<td><xsl:value-of select="title"/></td>
<td><xsl:value-of select="plot"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

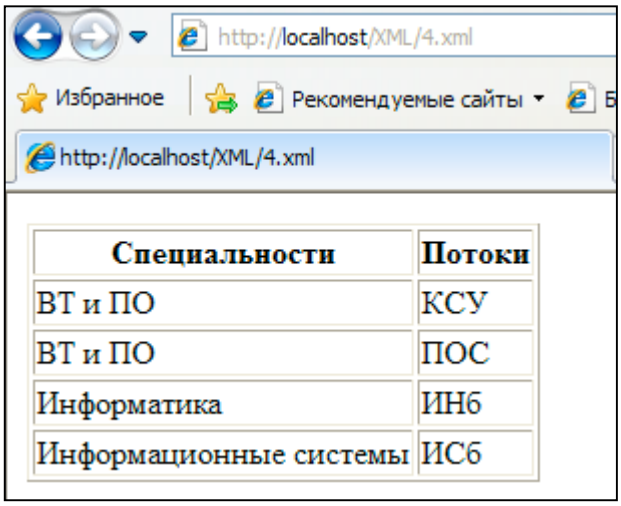
Результат 3.xsl :



4.xsl (сортировка)

```
<?xml version="1.0" encoding="windows-1251"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<table border="1">
<tr>
<th>Специальности</th> <th>Потоки</th>
</tr>
<xsl:for-each select="IIT/kafedra/characters/character">
<xsl:sort select="spec"/>
<tr>
<td><xsl:value-of select="spec"/></td>
<td><xsl:value-of select="group"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

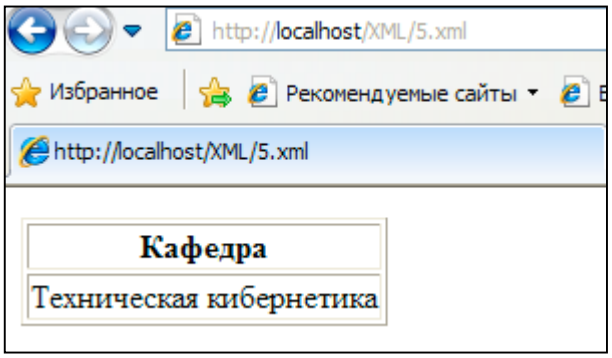
Результат 4.xsl :



5.xsl (фильтрация)

```
<?xml version="1.0" encoding="windows-1251"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<table border="1">
<tr>
<th>Кафедра</th>
</tr>
<xsl:for-each select="IIT/kafedra[title='TK']">
<tr>
<td> <xsl:value-of select="plot"/> </td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Результат 5.xsl :



XML-XSL(XPath)

Исходный XML-файл

```
<?xml version="1.0" encoding="windows-1251"?>
<?xml-stylesheet type="text/xsl" href=" имя файла *.xsl "?>
<note >
<date>
<day>01</day>
<month>01</month>
<year>2010</year>
</date>
<to id="25">Асхат</to>
<to id="26">Сауле</to>
<from>Владимир</from>
<heading>Напоминание</heading>
<body>У меня день рождения!</body>
</note>
```

XSL(XPath) файлы

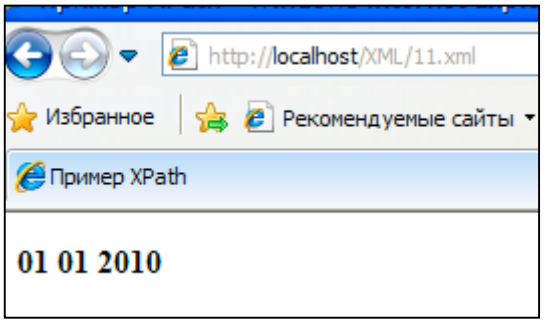
11.xsl (Выделение ветви)

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

```
<xsl:template match="/">
<html>
<head>
<title>Пример XPath</title>
</head>
<body>
<xsl:apply-templates select="note"/>
</body>
</html>
</xsl:template>
<xsl:template match="note">
<b><xsl:value-of select="date[year=2010]"/></b>
</xsl:template>
</xsl:stylesheet>
```

Здесь выводим все элементы **date,** имеющий дочерний элемент **year** со значением 2010.

Результат 11.xsl :



22.xsl (Выделение атрибутов)

```
<?xml version="1.0" encoding="windows-1251"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
```

<xsl:apply-templates/>

</body>

</html>

</xsl:template>

<xsl:template match="//to[@id='25']">

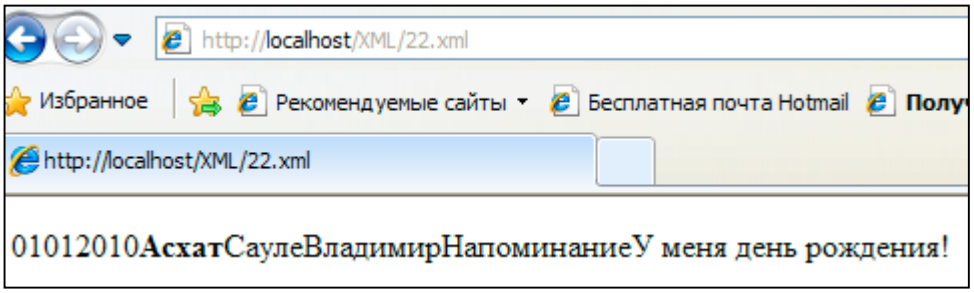
<xsl:value-of select="." />

</xsl:template>

</xsl:stylesheet>

Здесь находим находим все элементы, а элемент имеющий атрибут **id** со значением **'25'** будет выделен**.**

Результат 22.xsl :



33.xsl (использование осей (набор узлов относительно текущего узла))

<?xml version="1.0" encoding="windows-1251"?>

<xsl:stylesheet version="1.0"

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

<xsl:template match="/">

<html>

<body>

<title>XPath</title>

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

</body>

</html>

</xsl:template>

<xsl:template match="note">

<h3>Элемент-сосед</h3>

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

</xsl:template>

<xsl:template match="heading">

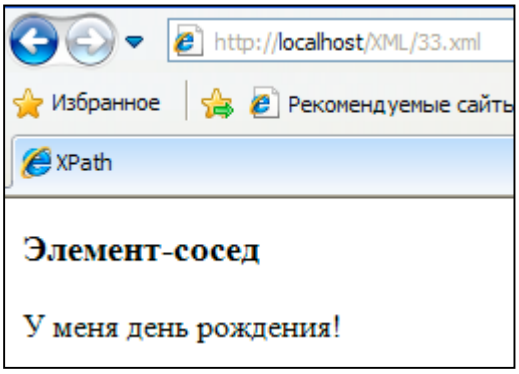
 <xsl:value-of select="following-sibling::body"/>

</xsl:template>

</xsl:stylesheet>

Здесь находим следующий элемент того же поколения , что и элемент **body.**

Результат 33.xsl :



44.xsl (сокращенная запись осей)

<?xml version="1.0"?>

<xsl:stylesheet version="1.0"

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

<xsl:template match="/note">

<html>

<body>

<table>

<th>Element</th>

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

</table>

</body>

</html>

</xsl:template>

<xsl:template match="date">

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

</xsl:template>

<xsl:template match="year">

<tr>

<td>**<xsl:value-of select="../../from"/>** </td>

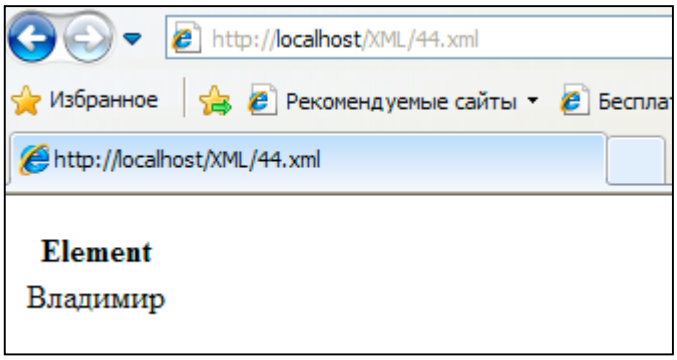
</tr>

</xsl:template>

</xsl:stylesheet>

Здесь сначала определяем элемент **year,** затем выходим на два уровня выше и выбираем элемент **from.**

Результат 44.xsl :



3. Реализация проекта с применением технологии XML
(SAX - парсер новостной ленты RSS на основе PHP)

Задание: ознакомится с особенностями XML-формата, предназначенного для реализации RSS каналов, модифицировать приведенный пример, внося различные изменения в RSS ленту.

RSS (Really Simple Syndication)-это семейство XML-форматов, предназначенных для описания лент новостей, анонсов статей и т.д. Интернет ресурс в формате RSS называется RSS -лентой или RSS -каналом. SAX (Simple Application Programming Intrface for XML) - метод обработки XML-документов, основанный на анализе дерева документа и обработке событий.

Новостная лента rss.xml

<?xml version="1.0" encoding="windows-1251"?>

<rss version="2.0">

<channel>

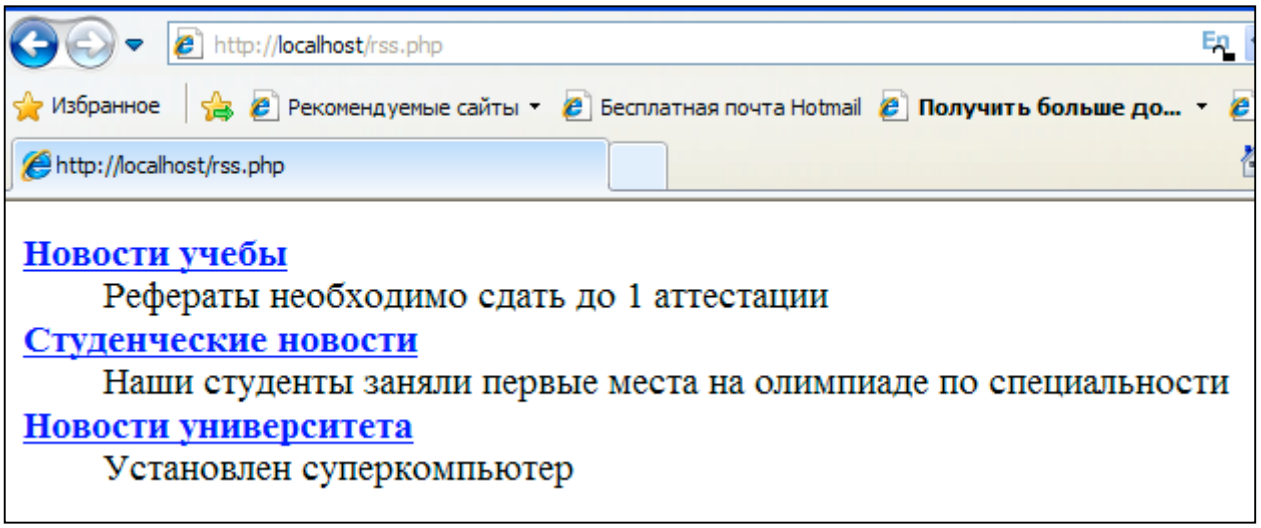
<title>новости</title>
<link>http://www.kaftk.narod.ru</link>
<description>Интерактивные учебные ресурсы</description>
<language>ru</language>
<lastBuildDate>Mon, 28 Aug 2010 18:14:06 +0400</lastBuildDate>
<webMaster>kaftk@mail.ru</webMaster>
<image>
<title>e-learning</title>
<url>http://www.kaftk.narod.ru/logo.gif</url>
<link>http://www.kaftk.narod.ru</link>
<width>100</width>
<height>100</height>
<description>Интерактивные учебные ресурсы</description>
</image>
<item>
<title>Новости учебы</title>
<link>http://www.kaftk.narod.ru/nu.htm</link>
<description>Рефераты необходимо сдать до 1 аттестации</description>
<pubDate>Mon, 28 Nov 2009 16:00:54 +0400</pubDate>
</item>
<item>
<title>Студенческие новости</title>
<link>http://www.kaftk.narod.ru/sn.htm</link>
<description>Наши студенты заняли первые места на олимпиаде по специальности</description>
<pubDate>Thu, 24 May 2009 15:57:32 +0400</pubDate>
</item>
<item>
<title>Новости университета</title>
<link>http://www.kaftk.narod.ru/un.htm</link>
<description>Установлен суперкомпьютер</description>
<pubDate>Thu, 24 May 2009 10:34:53 +0400</pubDate>
</item>
</channel></rss>

SAX парсер rss.php

```
<?php
$insideitem = false;
$tag = "";
$title = "";
$description = "";
$link = "";
function startElement($parser, $name, $attrs)
{
global $insideitem, $tag, $title, $description, $link;
if ($insideitem) {
$tag = $name;
} elseif ($name == "ITEM") {
$insideitem = true;
}
}
function endElement($parser, $name)
{
global $insideitem, $tag, $title, $description, $link;
if ($name == "ITEM") {
$description = htmlspecialchars(trim($description));
$title = htmlspecialchars(trim($title));
printf("<dt><b><a href='%s'>%s</a></b></dt>", trim($link), $title);
printf("<dd>%s</dd>",$description);
$title = "";
$description = "";
$link = "";
$insideitem = false;
}
}
function characterData($parser, $data)
{
global $insideitem, $tag, $title, $description, $link;
```

```
if ($insideitem) {
switch ($tag) {
case "TITLE":
$title .= $data;
break;
case "DESCRIPTION":
$description .= $data;
break;
case "LINK":
$link .= $data;
break;
}
}
}
// Создание парсера
$xml_parser = xml_parser_create();
// Задание функций-обработчиков
xml_set_element_handler($xml_parser, "startElement", "endElement");
xml_set_character_data_handler($xml_parser, "characterData");
// Открытие файла RSS-ленты
$fp = fopen("rss.xml","r") or die("Error reading RSS data.");
// Чтение четырех килобайтов данных из файла
while ($data = fread($fp, 4096))
// Вызов парсера для обработки данных из файла
xml_parse($xml_parser, $data, feof($fp))
or die(sprintf("XML error: %s at line %d",
xml_error_string(xml_get_error_code($xml_parser)),
xml_get_current_line_number($xml_parser)));
// Закрытие файла с данными после того, как он прочитан
полностью.
fclose($fp);
// Освобождение ресурсов парсера.
xml_parser_free($xml_parser);
?>
```

Результат rss.php:



Литература:

1. Бенкен Е.С. PHP,MySQL,XML: программирование для Интернета.-СПб.: БХВ-Петербург,2007.
2. Маршал Б. XML в действии. - М.: Издательство «Триумф», 2002. – 368 с.
3. Рэй Эрик. Изучаем XML. СПб. Символ-Плюс. 2001. 408 с.
4. Ганеев Р.М. Проектирование интерактивных интернет-приложений.-М.: Горячая линия-Телеком,2001.
5. Жумагалиев Б.И. Лабораторный практикум по интернет-технологиям. Алматы,2003