Белгородский Государственный Технологический Университет им. В. Г. Шухова

Кафедра программного обеспечения вычислительной техники  
и автоматизированных систем

## Лабораторная работа №5 по теме: «Использование XML на страницах в веб-приложениях»

**Выполнил:**  
студент группы ПВ-41  
Адаменко И. И.

**Проверил:**  
старший преподаватель  
Гарибов А. И.

Белгород  
2015

**Цель работы:** научиться работать с XML, XSLT и XSD в контексте веб-приложений на PHP и ASP.NET

# Теоретическая часть

## XML

XML — SGML-производный язык разметки документов, позволяющий структурировать информацию разного типа, используя для этого производный набор инструкций. XML-документы выполняют роль универсального формата для обмена информацией.

XML-документ представляет собой обычный текстовый файл, в котором при помощи специальных маркеров создаются элементы данных, последовательность и вложенность которых определяет структуру документа и его содержание. Основным достоинством XML-документов является то, что при относительно простом способе создания и обработки, они позволяют создавать структурированную информацию, которую хорошо «понимают» компьютеры.

## XSD

XSD или XML Schema — это язык описания структуры XML-документа. Используется для определения правил, которым должен подчиняться документ. Это облегчает работу с XML-документами, поскольку, если документ валиден (т. е. проходит проверку на соответствие XSD), то программа, читающая его, может создать модель данных этого документа, которая включает:

* словарь (названия элементов и атрибутов);
* модель содержания (отношения между элементами и атрибутами и их структура);
* типы данных.

Таким образом, ООП-языкам (как и в целом всем остальным) гораздо легче иметь дело с таким объектом, чем с текстовым файлом.

## XSLT

XSLT — язык преобразования XML-документов. Фактически, у него множество применений, однако одна из главных — это отделение данных от представления, согласно парадигме MVC. Другой стандартной задачей является преобразование XML-документов из соответствия одной XSD в другую.

Язык является декларативным, а не процедурным. Вместо определения последовательности исполняемых операторов, этот язык определяет правила, которые будут применяться во время преобразования.

# Практическая часть

В результате работы веб-приложения из базы данных выбираются данные о записях в блоге, сохраняются в XML, который затем трансформируется с помощью XSLT в HTML главной страницы блога.

## XML

Пример обрабатываемого XML-файла:

1. **<?xml** version="1.0"**?>**
2. **<posts>**
3. **<post** link="post?id=42" title="Нельзя делать сайты пиксель-в-пиксель" author="igoradamenko" date="22.11.15 20:03"**>**Нельзя ни в коем случае делать сайты пиксель-в-пиксель, потому что мы сами виноваты в том, что ИЕ 6 победил. В том, что он очень долго не обновлялся. Потому что если сайт выглядит одинаков..**</post>**
4. **<post** link="post?id=41" title="Доктайп — это важно" author="igoradamenko"  
   date="22.11.15 20:01"**>**Только что полчаса пытался поймать с виду какие-то неуловимые баги. В вёрстке размер блока 35px, на продакшене — 18px. Где-то свойство display меняется как мне н..**</post>**
5. **<post** link="post?id=40" title="gulp-minify-css" author="igoradamenko" date="22.11.15 19:59"**>**В повседневной работе для минификации стилей я пользуюсь именно этим плагином для галпа. **<br** **/><br** **/>**Выбор в своё время на него пал из-за в..**</post>**
6. **</posts>**

## XSD

Каждый обрабатываемый XML-файл соответствует следующей схеме:

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<xs:schema** xmlns:xs="http://www.w3.org/2001/XMLSchema"**>**
3. **<xs:element** name="posts"**>**
4. **<xs:complexType>**
5. **<xs:sequence>**
6. **<xs:element** name="post" maxOccurs="unbounded" minOccurs="0"**>**
7. **<xs:complexType** mixed="true"**>**
8. **<xs:sequence>**
9. **<xs:element** type="xs:string" name="br" maxOccurs="unbounded"  
    minOccurs="0"**/>**
10. **</xs:sequence>**
11. **<xs:attribute** type="xs:string" name="link" use="required"**/>**
12. **<xs:attribute** type="xs:string" name="title" use="required"**/>**
13. **<xs:attribute** type="xs:string" name="author" use="required"**/>**
14. **<xs:attribute** type="xs:string" name="date" use="required"**/>**
15. **</xs:complexType>**
16. **</xs:element>**
17. **</xs:sequence>**
18. **</xs:complexType>**
19. **</xs:element>**
20. **</xs:schema>**

## XSLT

XML-файлы обрабатываются с помощью следующего XSLT-шаблона:

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<xsl:stylesheet** version="1.0"
3. xmlns:xsl="http://www.w3.org/1999/XSL/Transform"**>**
5. **<xsl:output** method="html" indent="yes"**/>**
7. **<xsl:template** match="/"**>**
8. **<xsl:text** disable-output-escaping='yes'**>**<!DOCTYPE html**></xsl:text>**
9. **<html** lang="ru"**>**
10. **<head>**
11. **<meta** charset="UTF-8" **/>**
12. **<meta** name="viewport" content="width=device-width, initial-scale=1.0" **/>**
13. **<title>**Блог**</title>**
14. **<link** rel="stylesheet" href="awsm.min.css" **/>**
15. **</head>**
16. **<body>**
17. **<header>**
18. **<h1>**Блог**</h1>**
19. **<nav>**
20. **<ul>**
21. **<li><a** href="/bstu/pv/db/new"**>**новый пост**</a></li>**
22. **<li><a** href="/bstu/pv/db/users"**>**пользователи**</a></li>**
23. **<li><a** href="/bstu/pv/db/tags"**>**теги**</a></li>**
24. **</ul>**
25. **</nav>**
26. **</header>**
27. **<main>**
28. **<xsl:choose>**
29. **<xsl:when** test="count(//post) > 0"**>**
30. **<xsl:apply-templates/>**
31. **</xsl:when>**
32. **<xsl:otherwise>**
33. **<p><em>**Увы, но постов пока нет.**</em></p>**
34. **</xsl:otherwise>**
35. **</xsl:choose>**
37. **</main>**
38. **<footer>**
39. **<p>**Игорь Адаменко**</p>**
40. **</footer>**
41. **</body>**
42. **</html>**
43. **</xsl:template>**
45. **<xsl:template** match="post"**>**
46. **<section>**
47. **<h2>**
48. **<a** href="{@link}"**>**
49. **<xsl:value-of** select="@title" **/>**
50. **</a>**
51. **</h2>**
52. **<aside>**
53. **<xsl:value-of** select="@author" **/>**
54. **<br** **/>**
55. **<xsl:value-of** select="@date" **/>**
56. **</aside>**
57. **<p>**
58. **<xsl:value-of** select="." **/>**
59. **</p>**
60. **</section>**
61. **</xsl:template>**
63. **</xsl:stylesheet>**

## HTML

После работы XSLT получается следующий HTML:

1. <!DOCTYPE html**>**
2. **<html** lang="ru"**>**
3. **<head>**
4. **<meta** http-equiv="Content-Type" content="text/html; charset=UTF-8"**>**
5. **<meta** charset="UTF-8"**>**
6. **<meta** name="viewport" content="width=device-width, initial-scale=1.0"**>**
7. **<title>**Блог**</title>**
8. **<link** rel="stylesheet" href="awsm.min.css"**></head>**
9. **<body>**
10. **<header>**
11. **<h1>**Блог**</h1>**
12. **<nav>**
13. **<ul>**
14. **<li>**
15. **<a** href="/bstu/pv/db/new"**>**новый пост**</a>**
16. **</li>**
17. **<li>**
18. **<a** href="/bstu/pv/db/users"**>**пользователи**</a>**
19. **</li>**
20. **<li>**
21. **<a** href="/bstu/pv/db/tags"**>**теги**</a>**
22. **</li>**
23. **</ul>**
24. **</nav>**
25. **</header>**
26. **<main>**
27. **<section>**
28. **<h2>**
29. **<a** href="post?id=42"**>**Нельзя делать сайты пиксель-в-пиксель**</a>**
30. **</h2>**
31. **<aside>**igoradamenko**<br>**22.11.15 20:03**</aside>**
32. **<p>**Нельзя ни в коем случае делать сайты пиксель-в-пиксель, потому что мы сами виноваты в том, что ИЕ 6 победил. В том, что он очень долго не обновлялся. Потому что если сайт выглядит одинаков..**</p>**
33. **</section>**
34. **<section>**
35. **<h2>**
36. **<a** href="post?id=41"**>**Доктайп — это важно**</a>**
37. **</h2>**
38. **<aside>**igoradamenko**<br>**22.11.15 20:01**</aside>**
39. **<p>**Только что полчаса пытался поймать с виду какие-то неуловимые баги. В вёрстке размер блока 35px, на продакшене — 18px. Где-то свойство display меняется как мне н..**</p>**
40. **</section>**
41. **<section>**
42. **<h2>**
43. **<a** href="post?id=40"**>**gulp-minify-css**</a>**
44. **</h2>**
45. **<aside>**igoradamenko**<br>**22.11.15 19:59**</aside>**
46. **<p>**В повседневной работе для минификации стилей я пользуюсь именно этим плагином для галпа.**<br/><br/>**Выбор в своё время на него пал из-за в..**</p>**
47. **</section>**
48. **</main>**
49. **<footer>**
50. **<p>**Игорь Адаменко**</p>**
51. **</footer>**
52. **</body>**
53. **</html>**

## PHP

В среде PHP используется следующий код, для генерации XML, его сохранении на диск и трансформации с помощью XSLT:

1. <?
2. $xml = **new** XMLWriter();
3. $xml->openMemory();
4. $xml->startDocument();
5. $xml->setIndent(true);
6. $xml->setIndentString("\t");
8. $xml->startElement('posts');
10. // $records — массив записей;
11. **foreach** ($records **as** $rec) {
12. $post = getPost($rec, $authors);
14. $xml->startElement('post');
16. $xml->writeAttribute('link', $post['link']);
17. $xml->writeAttribute('title', $post['title']);
18. $xml->writeAttribute('author', $post['author']);
19. $xml->writeAttribute('date', $post['date']);
20. $xml->writeRaw($post['excerpt']);
22. $xml->endElement();
23. }
25. $xml->endElement();
27. $resultXML = $xml->outputMemory();
29. $xml->flush();
31. file\_put\_contents('xml/index.xml', $resultXML);
33. $xslDoc = DOMDocument::loadXML(file\_get\_contents('xslt/index.xsl'));
34. $xmlDoc = DOMDocument::loadXML($resultXML);
36. $proc = **new** XSLTProcessor();
37. $proc->importStyleSheet($xslDoc);
39. echo $proc->transformToXML($xmlDoc);
40. ?>

## ASP.NET

В среде ASP.NET для тех же задач используется следующий код:

1. **using** System;
2. **using** System.Collections.Generic;
3. **using** System.Data;
4. **using** System.IO;
5. **using** System.Xml.Xsl;
6. **using** System.Xml;
8. **namespace** WebApplication
9. {
10. **public** partial **class** \_Default : System.Web.UI.Page
11. {
12. **protected** List<Dictionary<**string**, **string**>> posts;
13. **protected** **string** content;
15. **protected** **void** Page\_Load(**object** sender, EventArgs e)
16. {
17. // ...
19. var doc = **new** XmlDocument();
20. XmlDeclaration xmlDeclaration = doc.CreateXmlDeclaration("1.0",  
     "UTF-8", **null**);
21. XmlElement root = doc.DocumentElement;
22. doc.InsertBefore(xmlDeclaration, root);
24. XmlElement xmlPosts = doc.CreateElement(**string**.Empty, "posts", **string**.Empty);
25. doc.AppendChild(xmlPosts);
27. // posts — список записей;
28. **foreach** (var post **in** posts)
29. {
30. var x = doc.CreateElement(**string**.Empty, "post", **string**.Empty);
32. var t = doc.CreateDocumentFragment();
33. t.InnerXml = post["excerpt"];
34. x.AppendChild(t);
36. x.SetAttribute("link", post["link"]);
37. x.SetAttribute("title", post["title"]);
38. x.SetAttribute("author", post["author"]);
39. x.SetAttribute("date", post["date"]);
41. xmlPosts.AppendChild(x);
42. }
44. doc.Save(Server.MapPath("~/XML/index.xml"));
46. content = GetHTML(Server.MapPath("~/XSLT/index.xsl"), doc);
47. }
49. **private** **string** GetHTML(**string** xsltPath, XmlDocument xml)
50. {
51. var writer = **new** StringWriter();
52. var transform = **new** XslCompiledTransform();
54. transform.Load(xsltPath);
55. transform.Transform(xml, **null**, writer);
57. **return** writer.ToString();
58. }
59. }
60. }

# Скриншоты



Рисунок 1. Сгенерированная страница

# Вывод

В ходе выполнения этой лабораторной работы я познакомился с XML, XSLT и XSD и научился с ними работать в контексте веб-приложений на PHP и ASP.NET.