Министерство образования и науки Республики Казахстан

Костанайский государственный университет им. А. Байтурсынова

Кафедра программного обеспечения

**В. Синицина**

**Web - технологии**

Практикум

(2 часть)

Костанай, 2014

УДК 32.973-01

С 38

Рецензенты:

Уаисова М.М., зав.кафедрой кафедры информационных технологий КИнЭУ им.М.Дулатова

Байманкулов А.Т., доцент кафедры программного обеспечения КГУ имени А. Байтурсынова, д.ф.-м.н.

Салыкова О.С., доцент кафедры программного обеспечения КГУ имени. А. Байтурсынова, к.т.н.

Составитель: Синицина В.Б., ст.преподаватель кафедры программного обеспечения

Синицина В.Б.

С 24 Web-технологии. Практикум. - Костанай: КГУ имени А. Байтурсынова, 2014.- 88 с.

В рамках практикума рассматривается широкий спектр протоколов, стандартов и технологий, имеющих непосредственное отношение к разработке web-приложений. На практических занятиях изучаются различные роли и ответственность клиентов и серверов для различных приложений в WWW, общие принципы работы клиентских и серверных языков и технологий в WWW, основные протоколы, необходимые для создания и работы web-приложений, основные принципы и подходы к web-интеграции приложений, разнородных компонент и систем, основные продукты и технологии Майкрософт, используемые для разработки web-контента и web-приложений.

Предназначено для студентов, изучающих основы web-технологий. Может быть использовано преподавателями, магистрантами для проведения практических занятий.

Утверждено учебно-методическим советом Костанайского государственного университета имени А.Байтурсынова протокол от \_\_\_\_\_\_\_ № \_\_.

© КГУ имени А. Байтурсынова, 2014

© Синицина В.Б.

Содержание

|  |  |
| --- | --- |
| Лабораторная работа 10. «Работа с источниками данных в ASP.NET»………………………………………………………………... | 4 |
|  |  |
| Лабораторная работа 11. «Структура XML документа. XML схемы» | 16 |
|  |  |
| Лабораторная работа 12. «Программная обработка XML документов с помощью XML DOM»………………………………….. | 33 |
|  |  |
| Лабораторная работа 13. Форматирование и преобразование XML документа с помощью CSS и XSL. XSLT преобразование XML документа …………………………………………….............................. | 44 |
|  |  |
| Лабораторная работа 14. «Разработка веб-службы в ASPNET».…...... | 52 |
|  |  |
| Лабораторная работа 15. «Разработка веб-службы в ASP.NET. Создание прокси-сборки для веб-службы»…………………………… | 60 |
|  |  |
| Лабораторная работа 16. «Примеры разработки RSS-источников и RSS-ридеров»……………………………………………………………. | 68 |
|  |  |
| Лабораторная работа 17. «Реализация асинхронного взаимодействия веб-браузера с веб-сервером с помощью технологии AJAX» …………………………………………………...… | 76 |
|  |  |
| Лабораторная работа 18. «Создание мобильных веб-приложений при помощи ASP.NET»...……………………………………………….. | 82 |

**Лабораторная работа 10. «Работа с источниками данных в ASP.NET»**

*Цель работы:* изучение элементов WebForm, предназначенных для отображения на веб-странице данных, получаемых из источников данных.

В ASP.NET используются два элемента управления WebForm для управления отображением данных, получаемых из источника данных:

* DataGrid - Элемент управления, отображающий содержимое объекта ADO.NET DataSet в виде таблицы.
* DataList - Элемент управления для выбора значений, заполняемых из источника данных.

Если необходимо отобразить данные, полученные по запросу пользователя из источника данных, в виде таблицы на веб-странице, то ASP.NET предоставляет в распоряжение веб-программиста удобный элемент управления DataGrid, который был введен в ASP.NET 1.x. но теперь его функции перекрываются GridView. Элемент управления GridView может не только показывать данные, но и сортировать, выбирать, редактировать их. Если этой функциональности недостаточно, ее можно расширить, написав собственные обработчики событий.

Элементы, которые могут быть связаны с источниками данных, многообразны, например, DropDownList, ListBox, CheckBoxList, RadioButtonList, BulletedList.

***Элемент GridView (DataGrid)***

Элемент управления GridView - главный элемент управления для представления информации из баз данных в ASP.NET. Он является "наследником" элемента управления DataGrid, который использовался в предыдущих версиях ASP.NET. Основное назначение этого элемента управления - представление пользователю информации в табличном виде из источника данных с возможностью фильтрации, сортировки и редактирования.

Проще всего настроить этот элемент управления при помощи встроенного мастера. При использовании мастера необходимо:

* Выбрать тип источника данных.
* В свойствах источника данных (элемента управления DataSource) настроить параметры подключения к базе данных).
* Сгенерировать или ввести код команд SELECT, INSERT, UPDATE и DELETE. При этом необязательно заполнять данные для всех команд.
* Настроить дополнительные параметры объекта GridView, например, разрешить разбиение на страницы, фильтрацию, сортировку, выбрать оформление и т.п.

***DataList***

По сравнению с элементом управления GridView элемент управления DataList предоставляет больше возможностей по настройке внешнего вида данных, которые берутся из источника, однако требует существенно больше времени и усилий по настройке. Добавлять новые записи, также как и при использовании элемента управления GridView, средствами DataList невозможно.

Работа с этим элементом управления DataList (на закладке Default.aspx в режиме Design ) выглядит следующим образом:

* После перетаскивания элемента управления DataList в форму необходимо настроить для него источник данных (либо используя визуальный интерфейс - с помощью объекта DataSource, либо программно через свойство DataSource - DataSet);
* Далее нужно перейти в режим редактирования шаблонов для записей. Для этого в контекстном меню выбрать EditItemTemplate. Для Item Templates появятся четыре области:
  + ItemTemplate - шаблон для отображения обычных элементов на форме;
  + AlternatingItemTemplate - необязательный шаблон для отображения каждой второй записи. Обычно для него только чуть меняется фон по сравнению с ItemTemplate - чтобы удобнее было воспринимать длинные страницы;
  + SelectedItemTemplate - шаблон для отображения выбранной в настоящий момент записи;
  + EditItemTemplate - шаблон для отображения редактируемой в настоящий момент записи.

Эти свойства можно также определить и через раздел Properties в контекстном меню элемента управления DataList. Можно ограничиться также простым выбором одного из шаблонов автоформатирования.

* Затем нужно заполнить требуемыми элементами управления (и просто кодом HTML) каждый из шаблонов. Для каждого поля, информацию о котором нужно предоставлять пользователям, необходимо создать свой элемент управления. Для всех шаблонов, кроме EditItemTemplate, для большинства полей будут использоваться элементы управления Label (хотя возможны и другие варианты). Для EditItemTemplate чаще всего используются элементы управления TextBox. Чтобы связать элементы управления с полями в источнике данных, проще всего использовать ссылку EditDataBindings в мастере, который появляется для каждого из добавляемых элементов управления.

***Порядок выполнения лабораторной работы***

1.Использование элемента GridView (DataGrid).

В следующем ниже примере в качестве источника данных используется база данных в формате MS ACCESS, содержащая таблицу lesson с полями:

* id (тип - счетчик)
* Title (тип - текст)
* Teacher (тип - текст)
* Type (тип - текст)

Создайте файл в формате MS ACCESS, содержащий эту таблицу.

Создайте новый проект по шаблону ASP.NET Web Application.

Для страницы Default.aspx выберите режим отображения Split

Из панели элементов управления Toolbox (отображается с помощью меню View > Toolbox ) перетащите в форму элемент GridView (из группы Data).

Для элемента GridView настройте источник данных (Choose Data Source, Configure Data Source) и форматирование (можно выбрать один из шаблонов автоформатирования – Auto Format).

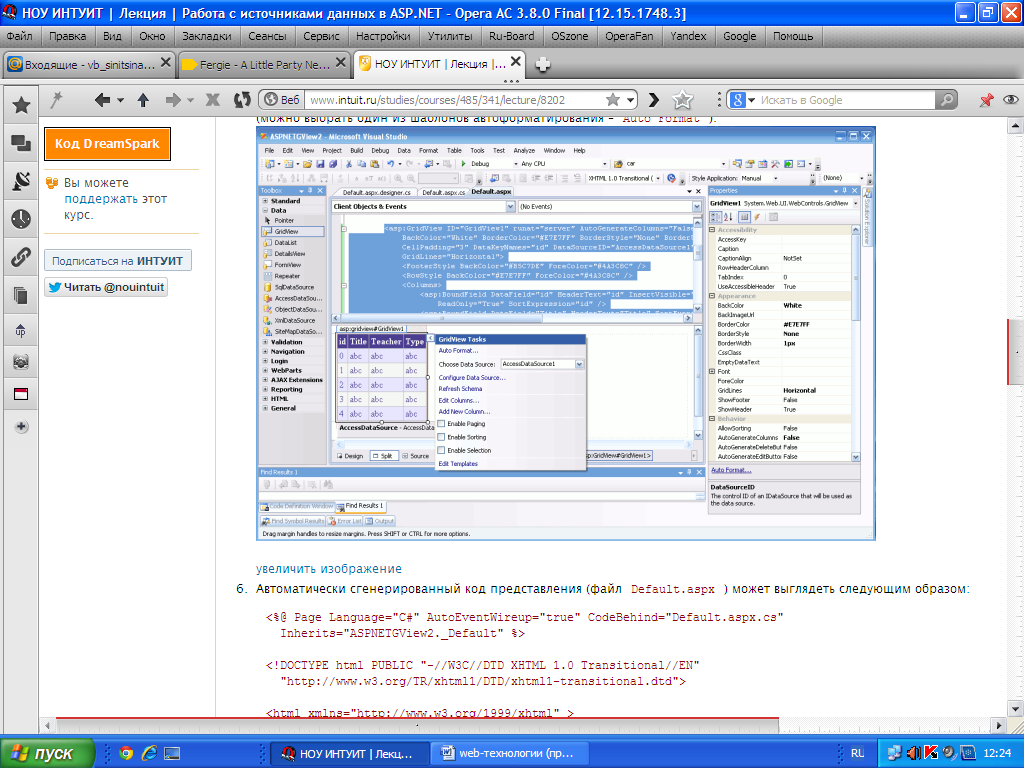


Рисунок 1 – Работа в ASP.NET

Автоматически сгенерированный код представления (файл Default.aspx ) может выглядеть следующим образом:

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"

Inherits="ASPNETGView2.\_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

<title>Untitled Page</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"

BackColor="White" BorderColor="#E7E7FF" BorderStyle="None" BorderWidth="1px"

CellPadding="3" DataKeyNames="id" DataSourceID="AccessDataSource1"

GridLines="Horizontal">

<FooterStyle BackColor="#B5C7DE" ForeColor="#4A3C8C" />

<RowStyle BackColor="#E7E7FF" ForeColor="#4A3C8C" />

<Columns>

<asp:BoundField DataField="id" HeaderText="id" InsertVisible="False"

ReadOnly="True" SortExpression="id" />

<asp:BoundField DataField="Title" HeaderText="Title" SortExpression="Title" />

<asp:BoundField DataField="Teacher" HeaderText="Teacher"

SortExpression="Teacher" />

<asp:BoundField DataField="Type" HeaderText="Type" SortExpression="Type" />

</Columns>

<PagerStyle BackColor="#E7E7FF" ForeColor="#4A3C8C" HorizontalAlign="Right" />

<SelectedRowStyle BackColor="#738A9C" Font-Bold="True" ForeColor="#F7F7F7" />

<HeaderStyle BackColor="#4A3C8C" Font-Bold="True" ForeColor="#F7F7F7" />

<AlternatingRowStyle BackColor="#F7F7F7" />

</asp:GridView>

<asp:AccessDataSource ID="AccessDataSource1" runat="server"

DataFile="D:\CSharp\ASPNETDataGrid\lessons.mdb"

SelectCommand="SELECT \* FROM [lesson]"></asp:AccessDataSource>

</div>

</form>

</body>

</html>

В данном примере использован элемент управления <asp:DataGrid>, имеющий идентификатор "GridView1".

Если источник данных не настроен автоматически (при описании кода представления):

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

<title>Untitled Page</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:DataGrid ID="GridView1" runat="server" BackColor="White"

BorderColor="White" BorderStyle="Ridge" BorderWidth="2px" CellPadding="3"

CellSpacing="1" GridLines="None" >

<FooterStyle BackColor="#C6C3C6" ForeColor="Black" />

<SelectedItemStyle BackColor="#9471DE" Font-Bold="True" ForeColor="White" />

<PagerStyle BackColor="#C6C3C6" ForeColor="Black" HorizontalAlign="Right" />

<ItemStyle BackColor="#DEDFDE" ForeColor="Black" />

<HeaderStyle BackColor="#4A3C8C" Font-Bold="True" ForeColor="#E7E7FF" />

</asp:DataGrid>

</div>

</form>

</body>

</html>

то это можно сделать программно в соответствующем программном коде, например:

using System;

using System.Collections;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.HtmlControls;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Xml.Linq;

using System.Data.OleDb;

namespace ASPNETDataGrid

{

public partial class \_Default : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

// Подключение к источнику данных

OleDbConnection cn = new OleDbConnection();

cn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;"

+ @"data source = D:\From-Nb-D\Work\method\CSharp\ASPNETDataGrid\Lessons.mdb";

cn.Open();

// Формируется строка SQL запроса данных из источника

string str = "SELECT \* from lesson";

// Происходит соединение с базой данных

// с помощью управляемого провайдера OLE DB

OleDbDataAdapter dAdapt = new OleDbDataAdapter(str, cn);

// Получение данных из источника

DataSet myDS = new DataSet("lessons");

dAdapt.Fill(myDS, "lesson");

// Заполнение таблицы данными

GridView1.DataSource = myDS.Tables["lesson"].DefaultView;

GridView1.DataBind();

cn.Close();

}

}

}

}

19.1.

Следует обратить внимание на то, что в программе добавлено пространство имен:

using System.Data.OleDb;

Свойство объекта Page - IsPostBack показывает, была ли страница клиента загружена в первый раз или повторно в ответ на переданные клиентом данные.

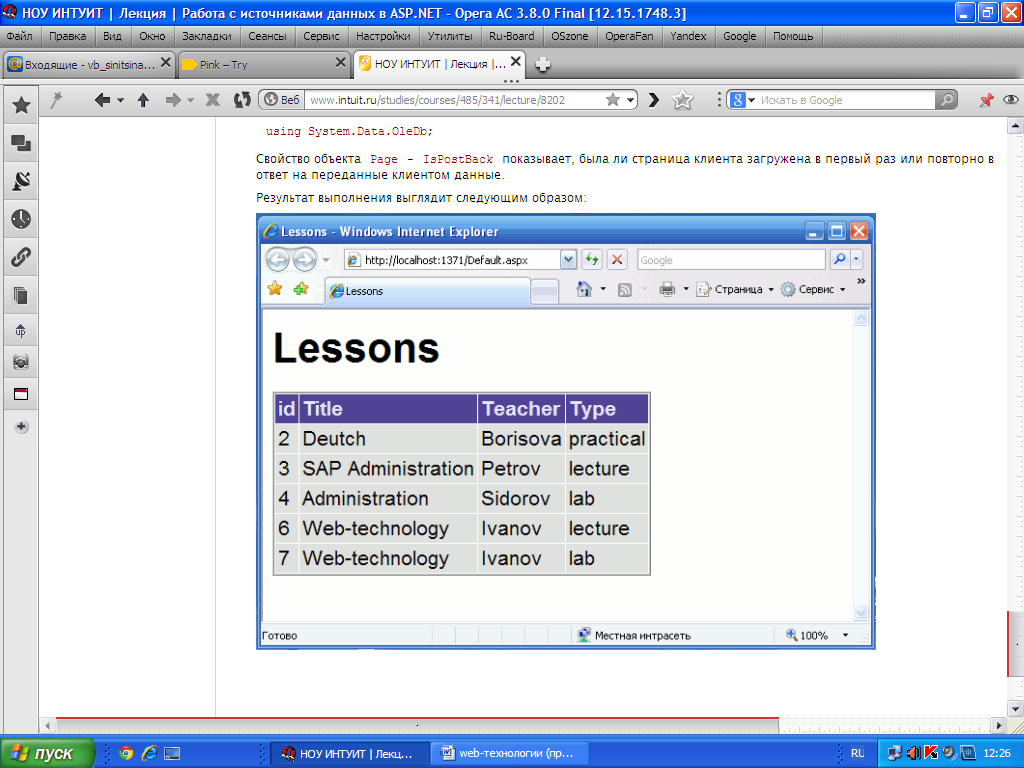


Рисунок 2 – Результат выполнения

2. Использование элемента DataList.

В следующем ниже примере в качестве источника данных используется база данных в формате MS ACCESS, содержащая таблицу lesson (как в предыдущем примере).

Создайте новый проект по шаблону ASP.NET Web Application.

Для страницы Default.aspx выберите режим отображения Split

Из панели элементов управления Toolbox (отображается с помощью меню View > Toolbox ) перетащите в форму элемент DataList (из группы Data ).

Для элемента DataList настройте источник данных (Choose Data Source, Configure Data Source) и форматирование (можно выбрать один из шаблонов автоформатирования – Auto Format ).

Получится следующий код представления:

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"

Inherits="ASPNETGView2.\_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

<title>Untitled Page</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:DataList ID="DataList1" runat="server" BackColor="White"

BorderColor="#999999" BorderStyle="Solid" BorderWidth="1px" CellPadding="3"

DataKeyField="id" DataMember="DefaultView" DataSourceID="AccessDataSource1"

ForeColor="Black" GridLines="Vertical" style="margin-right: 0px" Width="262px" >

<FooterStyle BackColor="#CCCCCC" />

<AlternatingItemStyle BackColor="#CCCCCC" />

<SelectedItemStyle BackColor="#000099" Font-Bold="True" ForeColor="White" />

<HeaderStyle BackColor="Black" Font-Bold="True" ForeColor="White" />

<ItemTemplate>

id:

<asp:Label ID="idLabel" runat="server" Text='<%# Eval("id") %>' />

<br />

Title:

<asp:Label ID="TitleLabel" runat="server" Text='<%# Eval("Title") %>' />

<br />

Teacher:

<asp:Label ID="TeacherLabel" runat="server" Text='<%# Eval("Teacher") %>' />

<br />

Type:

<asp:Label ID="TypeLabel" runat="server" Text='<%# Eval("Type") %>' />

<br />

</ItemTemplate>

</asp:DataList>

<asp:AccessDataSource ID="AccessDataSource1" runat="server"

DataFile="D:\CSharp\ASPNETDataGrid\lessons.mdb"

SelectCommand="SELECT \* FROM [lesson]">

</asp:AccessDataSource>

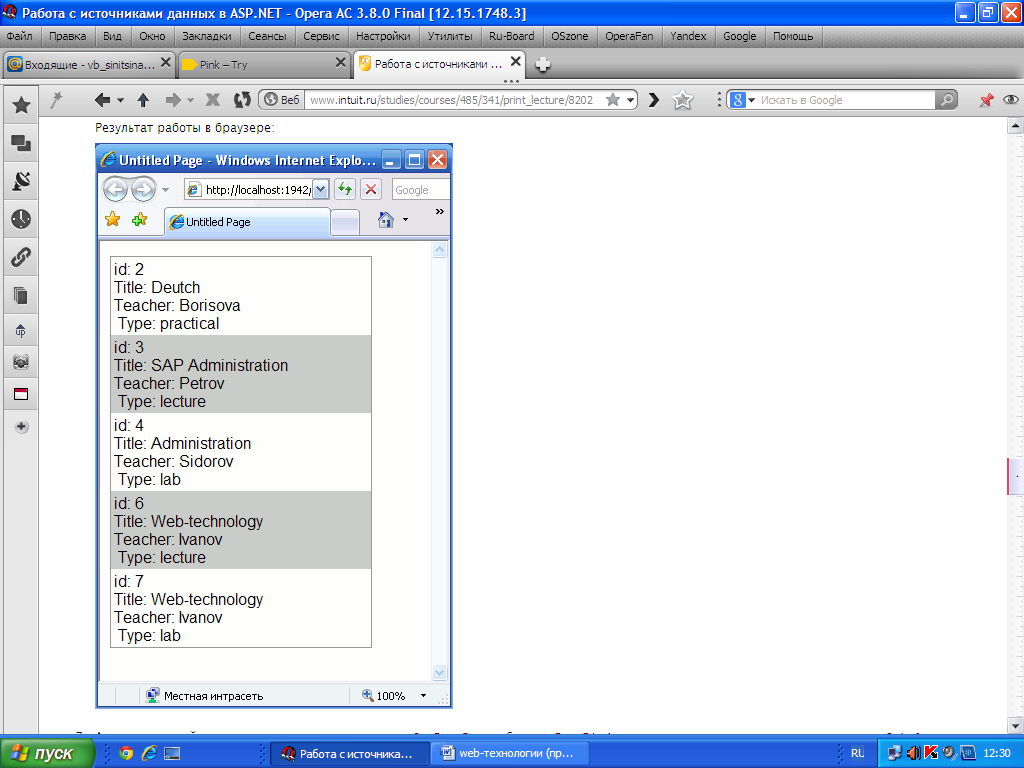
</div>

</form>

</body>

</html>

Результат работы в браузере:



Альтернативный вариант – использование метода GetDataItem объекта DataBinder для заполнения элементов вместо <asp:Label>.

Ниже представлен соответствующий код представления:

<%@ Page Language="C#" AutoEventWireup="true"

CodeBehind="Default.aspx.cs" Inherits="ASPNETDataGrid.\_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head id="Head2" runat="server">

<title>Lessons</title>

</head>

<body>

<form id="form2" runat="server">

<div>

<h1>Lessons</h1>

</div>

<asp:DataList ID="DataList1" runat="server" BackColor="White"

BorderColor="White" BorderStyle="Ridge" BorderWidth="2px" CellPadding="3"

CellSpacing="1">

<FooterStyle BackColor="#C6C3C6" ForeColor="Black" />

<ItemStyle BackColor="#DEDFDE" ForeColor="Black" />

<SelectedItemStyle BackColor="#9471DE" Font-Bold="True" ForeColor="White" />

<HeaderStyle BackColor="#4A3C8C" Font-Bold="True" ForeColor="#E7E7FF" />

<ItemTemplate>

<%# DataBinder.GetDataItem(Container)%>

</ItemTemplate>

</asp:DataList>

</form>

</body>

</html>

Программный код:

using System;

using System.Collections;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.HtmlControls;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Xml.Linq;

namespace ASPNETDataGrid

{

public partial class \_Default : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

ArrayList DWeek = new ArrayList();

DWeek.Add("Sunday");

DWeek.Add("Monday");

DWeek.Add("Tuesday");

DWeek.Add("Wednesday");

DWeek.Add("Thursday");

DWeek.Add("Friday");

DWeek.Add("Saturday");

DataList1.DataSource = DWeek;

DataList1.DataBind();

}

}

}

}

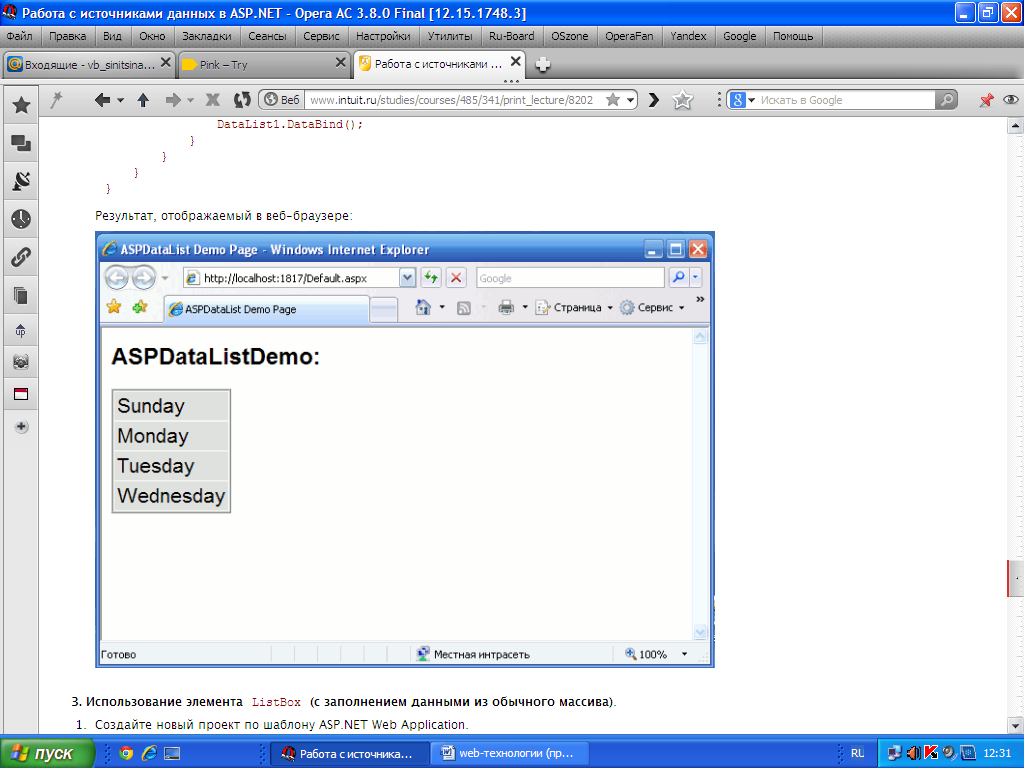


Рисунок 3 – Результат, отображаемый в веб-браузере

3. Использование элемента ListBox (с заполнением данными из обычного массива).

Создайте новый проект по шаблону ASP.NET Web Application.

Для страницы Default.aspx выберите режим отображения Split

Из панели элементов управления Toolbox (отображается с помощью меню View > Toolbox ) перетащите в форму элемент ListBox (из группы Data ).

Получится следующий код представления:

<%@ Page Language="C#" AutoEventWireup="true"

CodeBehind="Default.aspx.cs" Inherits="ASPNETListBox.\_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

<title>Untitled Page</title>

</head>

<body>

<form id="Form1" runat="server">

<asp:ListBox ID="ListBox1" runat="server"></asp:ListBox>

</form>

</body>

</html>

и соответствующий программный код:

using System;

using System.Collections;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.HtmlControls;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Xml.Linq;

namespace ASPNETListBox

{

public partial class \_Default : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

// Массив строк, который нужно вставить

ArrayList DWeek = new ArrayList();

DWeek.Add("Sunday");

DWeek.Add("Monday");

DWeek.Add("Tuesday");

DWeek.Add("Wednesday");

DWeek.Add("Thursday");

DWeek.Add("Friday");

DWeek.Add("Saturday");

// Связывание элемента управления с объектом DWeek

ListBox1.DataSource = DWeek;

ListBox1.DataBind();

}

}

}

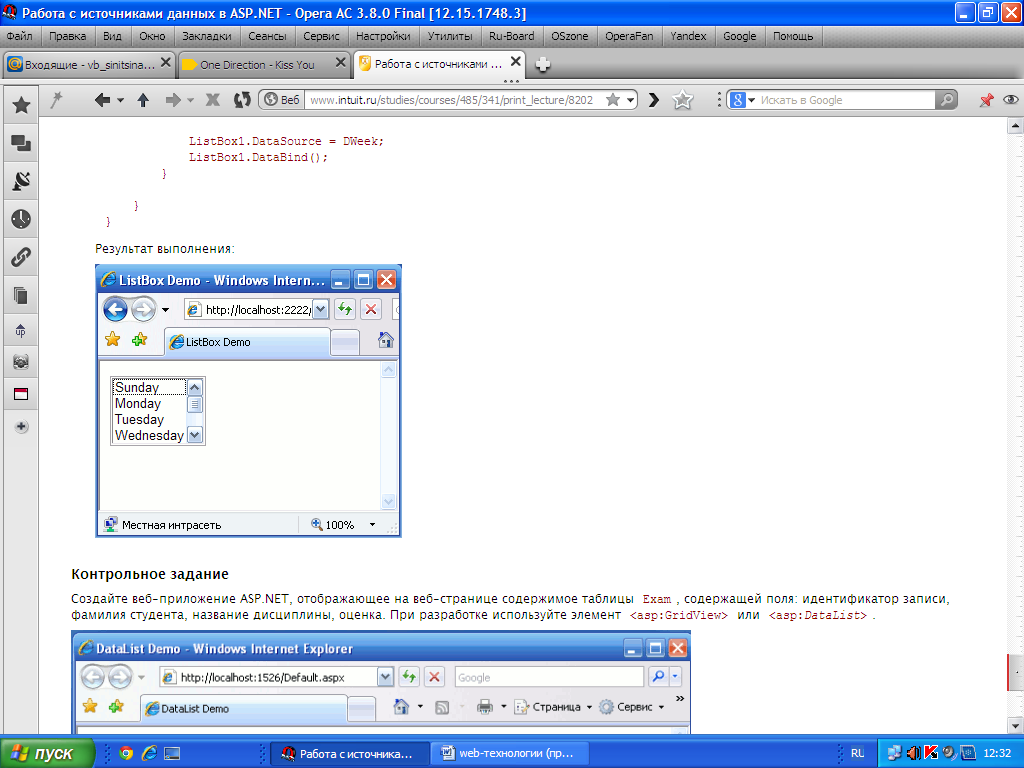


Рисунок 4 - Результат выполнения

**Контрольное задание**

Создайте веб-приложение ASP.NET, отображающее на веб-странице содержимое таблицы Exam, содержащей поля: идентификатор записи, фамилия студента, название дисциплины, оценка. При разработке используйте элемент <asp:GridView> или <asp:DataList>.

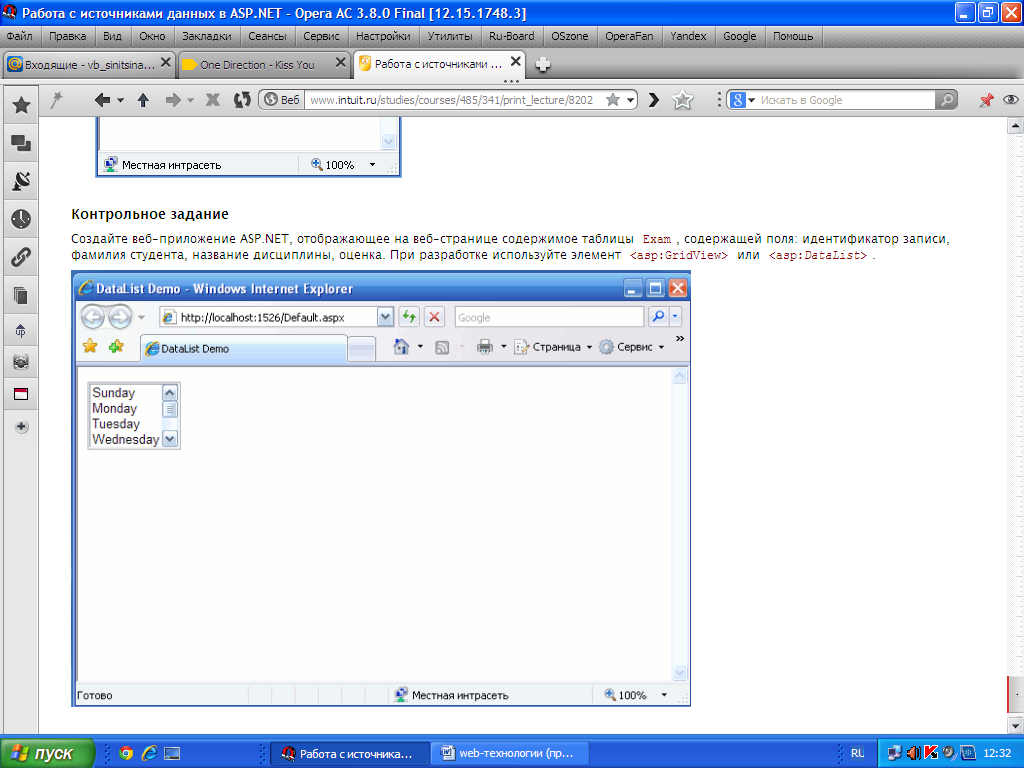


Рисунок 5 – Отображение на веб-странице содержимого таблицы Exam

Внимание! Если Вы увидите ошибку на нашем сайте, выделите её и нажмите Ctrl+Enter.

**Лабораторная работа 11. «Структура XML документа. XML схемы»**

*Цель работы:* знакомство со структурой XML-документа, основами синтаксиса языка и методами контроля содержимого документа с помощью схем

XML документ можно представить в виде древовидной структуры, которая начинается с "корня" и заканчивается "листьями".

Ниже приведен пример XML документа:

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

<mail>

<to>user1@domain.ru</to>

<from>user2@domain.ru </from>

<subject>Встреча</subject>

<body>Позвони мне завтра утром</body>

</mail>

В первой строке помещена XML декларация, описывающая версию XML и используемую кодировку

В следующей строке описан корневой элемент документа.

В последующих четырех строках описаны 4 дочерних элемента относительно корневого: ( to, from, subject и body ).

И в последней строке указывается конец корневого элемента.

XML документ должен содержать корневой элемент, который является родительским для всех остальных элементов. Любой элемент (кроме находящихся на самом нижнем уровне дерева) может иметь вложенные элементы (дочерние элементы).

С помощью терминов родитель, дочерний и потомок описываются отношения между элементами в дереве XML документа. Родители содержат дочерние элементы, а дочерние элементы одного уровня называются потомками (братьями или сестрами).

Элементы могут иметь текстовое содержимое и атрибуты.

Ниже приведена схема, показывающая отношения между элементами и атрибутами этого документа:

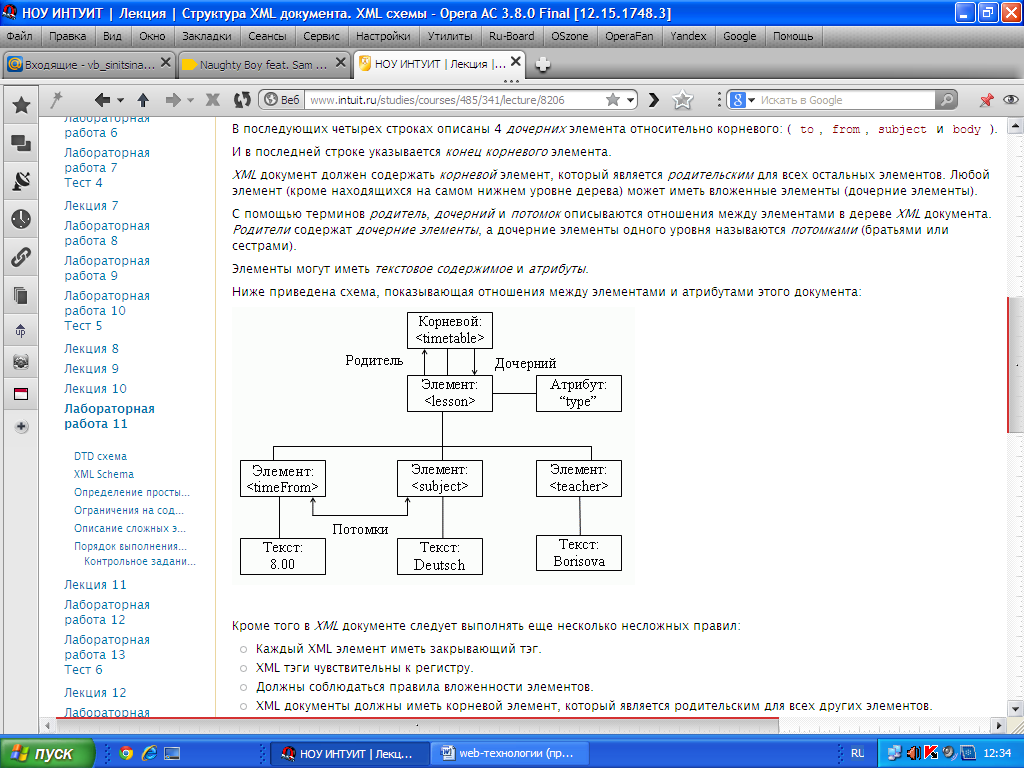


Рисунок 6 – Схема документа

Кроме того в XML документе следует выполнять еще несколько несложных правил:

* Каждый XML элемент иметь закрывающий тэг.
* XML тэги чувствительны к регистру.
* Должны соблюдаться правила вложенности элементов.
* XML документы должны иметь корневой элемент, который является родительским для всех других элементов.
* Значения XML атрибутов должны заключаться в двойные кавычки.

Некоторые из символов (их всего 5) имеют специальное значение в XML. Для их обозначения используются альтернативные сочетания:

|  |  |
| --- | --- |
| замена | символ |
| &lt; | < |
| &gt; | > |
| &amp; | & |
| &apos; | ' |
| &quot; | " |

Комментарии в XML выделяются также как и в HTML:

<!-- Это комментарий -->

В отличие от HTML, который игнорирует повторы пробелов, XML сохраняет полную длину строки из пробелов.

XML документы, оформленные в соответствии с приведенными выше правилами, называются "правильно оформленными". Если при этом они еще и соответствуют правилам DTD (Document Type Definition), то они являются и "действительными".

Действительные XML документы содержат ссылку на DTD файл, например:

<!DOCTYPE note SYSTEM "mail.dtd">

DTD схема предназначена для определения структуры XML документа путем задания списка допустимых элементов, например

<!DOCTYPE mail [

<!ELEMENT note (to,from,subject,body)>

<!ELEMENT to (#PCDATA)>

<!ELEMENT from (#PCDATA)>

<!ELEMENT subject (#PCDATA)>

<!ELEMENT body (#PCDATA)>

]>

Консорциум W3C поддерживает альтернативную к DTD, основанную на XML схему, называемую как "XML Schema":

<xs:element name="mail">

<xs:complexType>

<xs:sequence>

<xs:element name="to" type="xs:string"/>

<xs:element name="from" type="xs:string"/>

<xs:element name="subject" type="xs:string"/>

<xs:element name="body" type="xs:string"/>

</xs:sequence>

</xs:complexType>

</xs:element>

Для проверки действительности XML документа можно использовать специальные валидаторы, например W3C валидатор (http://validator.w3.org/). Для проверки схем также существуют специальные валидаторы, например XML Schema валидатор (http://www.w3.org/2001/03/webdata/xsv).

Согласно спецификации W3C XML программа должна прекратить обработку XML документа, как только будет обнаружена ошибка в этом документе.

*DTD схема*

DTD (Document Type Definition) определяет допустимые строительные блоки XML документа, путем указания списка допустимых элементотв и атрибутов.

DTD может описываться как внутри XML документа, так и с помощью внешней ссылки.

Пример внутреннего описания:

<?xml version="1.0"?>

<!DOCTYPE note [

<!ELEMENT mail (to,from,subject,body)>

<!ELEMENT to (#PCDATA)>

<!ELEMENT from (#PCDATA)>

<!ELEMENT subject (#PCDATA)>

<!ELEMENT body (#PCDATA)>

]>

<note>

<to>user1@domain.ru</to>

<from>user2@domain.ru</from>

<heading>Встреча</heading>

<body> Позвони мне завтра утром </body>

</note>

В данном примере:

!DOCTYPE mail определяет корневой элемент документа mail.

!ELEMENT note определяет элемент note,который содержит четыре элемента:

" to, from, subject, body ".

!ELEMENT to определяет элемент to типа "#PCDATA".

!ELEMENT from определяет элемент from типа "#PCDATA".

!ELEMENT subject определяет элемент subject типа "#PCDATA".

!ELEMENT body определяет элемент body типа "#PCDATA".

Пример внешнего описания:

<?xml version="1.0"?>

<!DOCTYPE note SYSTEM "mail.dtd">

<note>

<to>user1@domain.ru</to>

<from>user2@domain.ru</from>

<heading>Встреча</heading>

<body> Позвони мне завтра утром </body>

</note>

Для чего необходимо использование DTD?

С помощью DTD XML файлы могут содержать описание собственного формата.

Независимые группы людей могут обмениваться данными.

DTD-схема может быть использована для проверки действительности, как документов получаемых извне, так и собственных документов.

С точки зрения DTD все XML документы (HTML документы) строятся из следующих строительных блоков:

Элементы (Elements). Элементы могут содержать текст, другие элементы или быть пустыми.

Атрибуты Attributes). Атрибуты всегда размещаются внутри открывающего тэга элемента. Атрибуты всегда записываются в виде пары имя/значение.

*Entities.*

PCDATA. Означает структурированные символьные данные. Подлежит анализу с помощью парсера на наличие специальных символов (entities) и элементов разметки.

CDATA. Означает текст, содержимое которого не рассматривается анализатором.

При описании структуры элементов, содержащих вложенные элементы можно использовать специальные операторы, задающие множественность их вхождения:

' + ': один и более раз, например, <!ELEMENT note (message+)>

' \* ': ноль и более раз, например, <!ELEMENT note (message\*)>

' ?': ноль и один раз, например, <!ELEMENT note (message?)>

Объявление вида <!ELEMENT note (message|body)> указывает на возможность вхождения в элемент note другого элемента: message ИЛИ body.

Атрибуты элементов описываются в форме:

<!ATTLIST element-name attribute-name attribute-type default-value>

например:

<!ATTLIST payment type CDATA "check">

Соответствует XML коду:

<payment type="check" />.

Таблица 1 - Тип атрибута может принимать одно из значений:

|  |  |
| --- | --- |
| Тип | Описание |
| CDATA | Символьные данные |
| ( en1 | en2 |..) | Значение из списка |

Продолжение таблицы 1

|  |  |
| --- | --- |
| ID | Уникальный id |
| IDREF | id другого элемента |
| IDREFS | Список других id |
| NMTOKEN | Допустимое XML имя |
| NMTOKENS | Список допустимых XML имен |
| ENTITY | Специальные символы |
| ENTITIES | Список специальных символов |
| NOTATION | Имя нотации |
| xml: | Предопределенное XML имя |

Таблица 2 - Значение по умолчанию может быть одним из следующих:

|  |  |
| --- | --- |
| Значение | Интерпретация |
| #REQUIRED | Обязательно |
| #IMPLIED | Не обязательно |
| #FIXED value | Значение фиксировано |

*XML Schema*

Так же как DTD схема XML Schema определяет допустимые строительные блоки XML документа.

XML Schema:

* элементы, входящие в документ
* атрибуты, допустимые в документе
* дочерние элементы
* порядок дочерних элементов
* количество дочерних элементов
* может ли элемент быть пустым или может содержать текст
* типы элементов и атрибутов
* фиксированные и значения по умолчанию элементов и атрибутов

Предполагается, что в перспективе DTD схемы будут замещены XML Schema в большинстве веб-приложений поскольку XML Schema:

* Расширяема для будущих добавлений.
* Более богатые и мощные выразительные возможности.
* Является реализацией XML.
* Поддерживает типы данных.
* Поддерживают пространства имен.

XML Schema стала W3C рекомендацией в 2001 году.

Рассмотрим в качестве примера XML документ:

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

<mail>

<to>user1@domain.ru</to>

<from>user2@domain.ru </from>

<subject>Встреча</subject>

<body>Позвони мне завтра утром</body>

</mail>

Структура данного документа может быть рассмотрена с помощью следующей XML Schema:

<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"

targetNamespace="http://www.myhp.edu"

xmlns=" http://www.myhp.edu"

elementFormDefault="qualified">

<xs:element name="mail">

<xs:complexType>

<xs:sequence>

<xs:element name="to" type="xs:string"/>

<xs:element name="from" type="xs:string"/>

<xs:element name="subject" type="xs:string"/>

<xs:element name="body" type="xs:string"/>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

В данном примере элемент mail имеет тип complexType, поскольку содержит другие элементы. Другие элементы документа имеют простой тип, поскольку не включают других элементов.

Ссылка на схему в XML документе выглядит следующим образом:

<?xml version="1.0"?>

<mail

xmlns="http:// www.myhp.edu "

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.myhp.edu mail.xsd">

<to>user1@domain.ru</to>

<from>user2@domain.ru </from>

<subject>Встреча</subject>

<body>Позвони мне завтра утром</body>

</mail>

Элемент <schema> является корневым элементом любой схемы XML Schema.

Данный элемент может содержать несколько атрибутов, например:

<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"

targetNamespace="http://www.myhp.edu"

xmlns="http://www.myhp.edu"

elementFormDefault="qualified">

...

...

</xs:schema>

Следующий фрагмент:

xmlns:xs="http://www.w3.org/2001/XMLSchema"

указывает на то, что элементы и типы данных, используемые в схеме входят в пространство http://www.w3.org/2001/XMLSchema. Причем, все элементы и типы данных из этого пространства имен должны иметь префикс xs.

Фрагмент:

targetNamespace="http://www.myhp.edu"

указывает на то, что элементы, определяемые в схеме входят в пространство "http://www.myhp.edu".

Во фрагменте:

xmlns="http://www.myhp.edu"

указывается, что пространством имен по умолчанию является "http://www.myhp.edu"

Следующий фрагмент:

elementFormDefault="qualified"

указывает на то, что любые элементы, объявленные в схеме, должны принадлежать пространству имен.

Ссылка на внешнюю схему может выглядеть следующим образом:

<?xml version="1.0"?>

<mail xmlns=" http://www.myhp.edu "

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation=" http://www.myhp.edu mail.xsd">

<to>user1@domain.ru</to>

<from>user2@domain.ru </from>

<subject>Встреча</subject>

<body>Позвони мне завтра утром</body>

</mail>

Здесь строка

xsi:schemaLocation=" http://www.myhp.edu mail.xsd">

указывает на местоположение файла схемы.

***Определение простых элементов***

Синтаксис для определения простого элемента:

<xs:element name="xxx" type="yyy"/>

где xxx – имя элемента и yyy тип данных элемента.

Встроенными типами данных элементов являются следующие:

xs:string

xs:decimal

xs:integer

xs:boolean

xs:date

xs:time

Например, фрагмент XML документа:

<lastname>Refsnes</lastname>

<age>36</age>

<dateborn>1970-03-27</dateborn>

описывается в схеме следующим образом:

<xs:element name="lastname" type="xs:string"/>

<xs:element name="age" type="xs:integer"/>

<xs:element name="dateborn" type="xs:date"/>

Следующие фрагменты:

<xs:element name="color" type="xs:string" default="red"/>

и

<xs:element name="color" type="xs:string" fixed="red"/>

описывают значение элемента по умолчанию и фиксированное значения соответственно.

Все атрибуты описываются простыми типами данных.

Простые элементы не могут иметь атрибутов. Если элемент имеет атрибуты, он рассматривается как имеющий сложный тип. Сам атрибут рассматривается всегда как имеющий простой тип.

Описание атрибута обычно дается следующим образом:

<xs:attribute name="xxx" type="yyy"/>

где xxx – имя атрибута, а yyy - определяет тип данных атрибута.

Встроенными типами данных для атрибутов являются соледующие:

xs:string

xs:decimal

xs:integer

xs:boolean

xs:date

xs:time

Например, XML элемент с атрибутом:

<lastname lang="EN">Smith</lastname>

описывается соответствующей схемой:

<xs:attribute name="lang" type="xs:string"/>

Значение по умолчанию и фиксированное значение атрибута описываются следующим образом:

<xs:attribute name="lang" type="xs:string" default="EN"/>

и

<xs:attribute name="lang" type="xs:string" fixed="EN"/>

соответственно.

Для описания обязательных атрибутов (по умолчанию все атрибуты являются необязательными) используется следующая нотация:

<xs:attribute name="lang" type="xs:string" use="required"/>

Ограничения на содержимое

Для задания допустимых значений величин XML элементов и атрибутов можно использовать ограничения. Ограничения на значения XML элементов называются фасетами.

Таблица 3 - Список возможных ограничений приводится ниже:

|  |  |
| --- | --- |
| Ограничение | Описание |
| enumeration | Определяет список допустимых значений |
| fractionDigits | Указывет максимальное число десятичных позиций. Должно быть неотрицательным |
| length | Указывает точное число символов или элементов в списке. Должно быть неотрицательным |

Продолжение таблицы 3

|  |  |
| --- | --- |
| maxExclusive | Указывает верхнюю границу числовых значений |
| maxInclusive | Указывает верхнюю границу числовых значений |
| maxLength | Указывает максимальное число символов или элементов в списке. Должно быть неотрицательным |
| minExclusive | Указывает нижнюю невключаемую границу числовых значений |
| minInclusive | Указывает нижнюю включаемую границу числовых значений |
| minLength | Указывает минимальное число символов или элементов в списке. Должно быть неотрицательным |
| pattern | Определяет точно последовательность символов (шаблон), которая допустима |
| totalDigits | Определяет точное число цифр. Должно быть неотрицательным |
| whiteSpace | Указывает как обрабатывать неотображаемые символы (пробел, табуляция и др.) |

Примеры ограничения:

<xs:element name="age">

<xs:simpleType>

<xs:restriction base="xs:integer">

<xs:minInclusive value="0"/>

<xs:maxInclusive value="120"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="password">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:pattern value="[a-zA-Z0-9]{8}"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

***Описание сложных элементов***

При описании документов, имеющих сложную иерархическую структуру, можно сначала определить все элементы и атрибуты, а затем ссылаться на них.

Рассмотрим следующий пример:

<employee>

<firstname>John</firstname>

<lastname>Smith</lastname>

</employee>

Сложный элемент в схеме можно определить следующим образом:

<example><xs:element name="firstname" type="xs:string"/>

<xs:element name="lastname" type="xs:string"/>

<xs:element name="employee">

<xs:complexType>

<xs:sequence>

<xs:element ref="firstname"/>

<xs:element ref="lastname"/>

</xs:sequence>

</xs:complexType>

</xs:element>

Другой подход заключается в повторном использовании именованных элементов.

Элемент " employee " element может быть объявлен непосредственно по имени:

<xs:element name="employee">

<xs:complexType>

<xs:sequence>

<xs:element name="firstname" type="xs:string"/>

<xs:element name="lastname" type="xs:string"/>

</xs:sequence>

</xs:complexType>

</xs:element>

Поскольку дочерние элементы " firstname " и " lastname " окружены индикатором <sequence>, то они должны в документе появляться именно в этой последовательности.

Элемент " employee " может иметь атрибут типа, ссылающийся на имя сложного типа:

<xs:element name="employee" type="personinfo"/>

<xs:complexType name="personinfo">

<xs:sequence>

<xs:element name="firstname" type="xs:string"/>

<xs:element name="lastname" type="xs:string"/>

</xs:sequence>

</xs:complexType>

Несколько элементов могут ссылаться на тот же самый сложный тип:

<xs:element name="employee" type="personinfo"/>

<xs:element name="student" type="personinfo"/>

<xs:element name="member" type="personinfo"/>

<xs:complexType name="personinfo">

<xs:sequence>

<xs:element name="firstname" type="xs:string"/>

<xs:element name="lastname" type="xs:string"/>

</xs:sequence>

</xs:complexType>

Также можно строить сложный элемент на базе существующего сложного элемента с добавлением нескольких элементов:

<xs:element name="employee" type="fullpersoninfo"/>

<xs:complexType name="personinfo">

<xs:sequence>

<xs:element name="firstname" type="xs:string"/>

<xs:element name="lastname" type="xs:string"/>

</xs:sequence>

</xs:complexType>

<xs:complexType name="fullpersoninfo">

<xs:complexContent>

<xs:extension base="personinfo">

<xs:sequence>

<xs:element name="address" type="xs:string"/>

<xs:element name="city" type="xs:string"/>

<xs:element name="country" type="xs:string"/>

</xs:sequence>

</xs:extension>

</xs:complexContent>

</xs:complexType>

Пустой сложный элемент не может иметь содержимого, только атрибуты.

С помощью специальных индикаторов можно указывать, как элементы могут использоваться в документах.

Всего используется семь индикаторов:

Индикатор порядка (Order):

* All
* Choice
* Sequence

Индикатор вхождения (Occurrence):

* maxOccurs
* minOccurs

Групповые (Group) индикаторы:

* Group name
* attributeGroup name

Пример:

<xs:element name="person">

<xs:complexType>

<xs:sequence>

<xs:element name="full\_name" type="xs:string"/>

<xs:element name="child\_name" type="xs:string" maxOccurs="10"/>

</xs:sequence>

</xs:complexType>

</xs:element>

***Порядок выполнения лабораторной работы***

1.Описание структуры XML-документа с помощью DTD-схемы.

Создание XML-документа timetable.xml. Содержимое документа приведено ниже.

<?xml version="1.0"?>

<timetable>

<day dayOfWeek="Monday">

<lesson type="practical">

<timeFrom>08.00</timeFrom>

<timeTo>09.30</timeTo>

<subject>Deutsch</subject>

<teacher>Borisova</teacher>

<room>216</room>

</lesson>

<lesson type="lecture">

<timeFrom>09.40</timeFrom>

<timeTo>11.10</timeTo>

<subject>SAP Administration</subject>

<teacher>Egorov</teacher>

<room>384</room>

</lesson>

<lesson type="practical">

<timeFrom>11.20</timeFrom>

<timeTo>12.50</timeTo>

<subject>SAP Administration</subject>

<teacher>Petrov</teacher>

<room>384</room>

</lesson>

</day>

</timetable>

Загрузите этот файл в браузере. Обратите внимание на иерархическую структуру документа.

Добавьте в начало файла внутреннюю DTD-схему:

<?xml version="1.0"?>

<!DOCTYPE timetable[

<!ELEMENT timetable (day\*)>

<!ELEMENT day (lesson+)>

<!ELEMENT lesson (timeFrom, timeTo, subject, teacher, room?)>

<!ELEMENT timeFrom (#PCDATA)>

<!ELEMENT timeTo (#PCDATA)>

<!ELEMENT subject (#PCDATA)>

<!ELEMENT teacher (#PCDATA)>

<!ELEMENT room (#PCDATA)>

<!ATTLIST day

dayOfWeek CDATA #REQUIRED>

<!ATTLIST lesson

type CDATA #REQUIRED>

]>

Сохраните файл и проверьте его на правильность с помощью программы-валидатора, например, по адресу: http://validator.w3.org. Попробуйте удалить какой-либо элемент или атрибут, проверьте документ снова с помощью валидатора.

Вынесите схему во внешний файл с размещением на него ссылки внутри XML-документа. Проверьте, как будет загружаться XML-документ.

2.Описание структуры XML-документа с помощью XML Schema.

Создание XML-документа timetable2.xml. Содержимое документа приведено ниже.

<?xml version="1.0"?>

<timetable xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="timetable.xsd">

<day dayOfWeek="Monday">

<lesson type="practical">

<timeFrom>08.00</timeFrom>

<timeTo>09.30</timeTo>

<subject>Deutsch</subject>

<teacher>Borisova</teacher>

<room>216</room>

</lesson>

<lesson type="lecture">

<timeFrom>09.40</timeFrom>

<timeTo>11.10</timeTo>

<subject>SAP Administration</subject>

<teacher>Egorov</teacher>

<room>384</room>

</lesson>

<lesson type="practical">

<timeFrom>11.20</timeFrom>

<timeTo>12.50</timeTo>

<subject>SAP Administration</subject>

<teacher>Petrov</teacher>

<room>384</room>

</lesson>

</day>

</timetable>

Создание файла timetable.xsd, содержащего XML Schema:

<?xml version="1.0" encoding="ISO-8859-1" ?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name = "timetable">

<xs:complexType>

<xs:sequence>

<xs:element ref = "day" minOccurs = "0" maxOccurs = "unbounded"/>

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name = "day">

<xs:complexType>

<xs:sequence>

<xs:element ref = "lesson" maxOccurs = "unbounded"/>

</xs:sequence>

<xs:attribute name = "dayofWeek" use = "required" type = "xs:string"/>

</xs:complexType>

</xs:element>

<xs:element name = "lesson">

<xs:complexType>

<xs:sequence>

<xs:element ref = "timeFrom"/>

<xs:element ref = "timeTo"/>

<xs:element ref = "subject"/>

<xs:element ref = "teacher"/>

<xs:element ref = "room" minOccurs = "0"/>

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name = "timeFrom" type = "xs:string"/>

<xs:element name = "timeTo" type = "xs:string"/>

<xs:element name = "subject" type = "xs:string"/>

<xs:element name = "teacher" type = "xs:string"/>

<xs:element name = "room" type = "xs:string"/>

</xs:schema>

Проверьте схему на правильность с помощью программы-валидатора, например, по адресу: http://www.w3.org/2001/03/webdata/xsv.

**Контрольное задание**

В XML документе описана экзаменационная ведомость:

<gradeReport id="120851">

<date>10-06-2008</date>

<subject>Computer Science Fundamentals</subject>

<examiner>prof.Litvinov</examiner>

<gradeList>

<gradeRecord id="1">

<student>Ivanov</student>

<grade>4</grade>

</gradeRecord>

<gradeRecord id="2">

<student>Petrov</student>

<grade>3</grade>

</gradeRecord>

<gradeRecord id="3">

<student>Sidorov</student>

<grade>5</grade>

</gradeRecord>

</gradeList>

</gradeReport>

Опишите структуру данного документа с помощью DTD схемы и XML Schema. Проверьте обе схемы с помощью соответствующих программ-валидаторов.

**Лабораторная работа 12. «Программная обработка XML документов с помощью XML DOM»**

*Цель работы:* ознакомление с основными принципами XML DOM и методами программной обработки XML документов путем манипулирования узлами дерева документа.

XML DOM определяет объекты и свойства всех XML элементов и методы (интерфейс) для доступа к ним. Иначе говоря, XML DOM описывает каким образом необходимо получать, изменять, добавлять и удалять XML элементы.

В соответствии с моделью DOM все, что содержится внутри XML документа - является узлом. То есть XML документ представляется в виде дерева узлов, которыми являются элементы, атрибуты и текст.

Поскольку структуры HTML и XML документов очень похожи, а HTML DOM и XML DOM являются частями более общего стандарта DOM, то и многие аспекты HTML DOM легко переносимы в XML DOM. Поэтому основное внимание будет уделено именно специфическим особенностям именно XML DOM. Рекомендуется предварительное ознакомление с "Введение в JavaScript. Программное взаимодействие с HTML документами на основе DOM API" .

***Структурный анализ (парсинг) XML***

Все современные браузеры имеют встроенные XML анализаторы (парсеры) для чтения и обработки XML. Анализатор считывает XML документ, размещает его в памяти и преобразует в XML DOM объект, доступный для языков программирования. Все примеры здесь приведены на JavaScript.

Имеются некоторые отличия между анализаторами в Microsoft и в других браузерах. Первый поддерживает как загрузку XML файлов, так и текстовых строк, содержащих XML код, в то время как в других браузерах используются раздельные анализаторы. При этом все анализаторы имеют функции для перемещения по дереву XML документа, доступа, вставки и удаления узлов в дереве.

Рассмотрим пример загрузки XML объектов (файлов и строк) с помощью XML анализатора Microsoft.

xmlDoc=new ActiveXObject("Microsoft.XMLDOM");

xmlDoc.async="false";

xmlDoc.load("timetable.xml");

В первой строке программы создается пустой объект XML документа Microsoft. Далее для предотвращения работы сценария до полной загрузки документа флаг асинхронности устанавливается в " false ". В третьей строке содержится инструкция загрузить XML файл " timetable.xml ".

В следующем пример происходит загрузка строки с XML кодом для последующего анализа.

xmlDoc=new ActiveXObject("Microsoft.XMLDOM");

xmlDoc.async="false";

xmlDoc.loadXML(txt);

Следует обратить на разницу между методами load() и loadXML() по их назначению.

Замечание. Современные браузеры не допускают междоменные обращения к файлам из соображений безопасности, т.е. сама веб-страница (с программным кодом) и XML файл должны физически находиться на одном сервере. В противном случае браузер выдаст сообщение об ошибке доступа.

Ниже приведены также кроссплатформенные реализации загрузки XML файла и XML строки соответственно.

<html>

<body>

<script type="text/javascript">

try //Internet Explorer

{

xmlDoc=new ActiveXObject("Microsoft.XMLDOM");

}

catch(e)

{

try //Firefox, Mozilla, Opera, etc.

{

xmlDoc=document.implementation.createDocument("","",null);

}

catch(e) {alert(e.message)}

}

try

{

xmlDoc.async=false;

xmlDoc.load("timetable.xml");

document.write("xmlDoc is loaded, ready for use");

}

catch(e) {alert(e.message)}

</script>

</body>

</html>

<html>

<body>

<script type="text/javascript">

text="<timetable>";

text=text+"<lesson>";

text=text+"<timeFrom>08.00</timeFrom>";

text=text+"<subject>Deutsch</subject>";

text=text+"<teacher>Borisova</teacher>";

text=text+"</lesson>";

text=text+"/<timetable>";

try //Internet Explorer

{

xmlDoc=new ActiveXObject("Microsoft.XMLDOM");

xmlDoc.async="false";

xmlDoc.loadXML(text);

}

catch(e)

{

try //Firefox, Mozilla, Opera, etc.

{

parser=new DOMParser();

xmlDoc=parser.parseFromString(text,"text/xml");

}

catch(e) {alert(e.message)}

}

document.write("xmlDoc is loaded, ready for use");

</script>

</body>

</html>

23.1.

***Программный интерфейс XML DOM***

В рамках DOM модели XML можно рассматривать как множество узловых объектов. Доступ к ним осуществляется с помощью JavaScript или других языков программирования. Программный интерфейс DOM включает в себя набор стандартных свойств и методов.

Свойства представляют некоторые сущности (например, <day> ), а методы - действия над ними (например, добавить <lesson> ).

В XML DOM используются практически те же свойства и методы, что и в HTML DOM.

Например, результатом выполнения следующего ниже JavaScript кода будет текстовой содержимое элемента <subject> в файле timetable.xml. .

txt = xmlDoc.getElementsByTagName("subject")[0].childNodes[0].nodeValue;

Результат: "Deutsch ".

В рамках DOM XML возможны 3 способа доступа к узлам:

* С помощью метода getElementsByTagName(name). При этом возвращаются все узлы с указанным именем тэга (в виде индексированного списка). Первый элемент в списке имеет нулевой индекс.
* Путем обхода узлов дерева с использованием циклических конструкций.
* Путем перемещения по дереву с использованием отношений между узлами.

Для определения длины списка узлов используется свойство length.

*Перемещение между узлами дерева*

В XML DOM отношения между узлами определены в виде следующих свойств узлов:

* parentNode
* childNodes
* firstChild
* lastChild
* nextSibling
* previousSibling

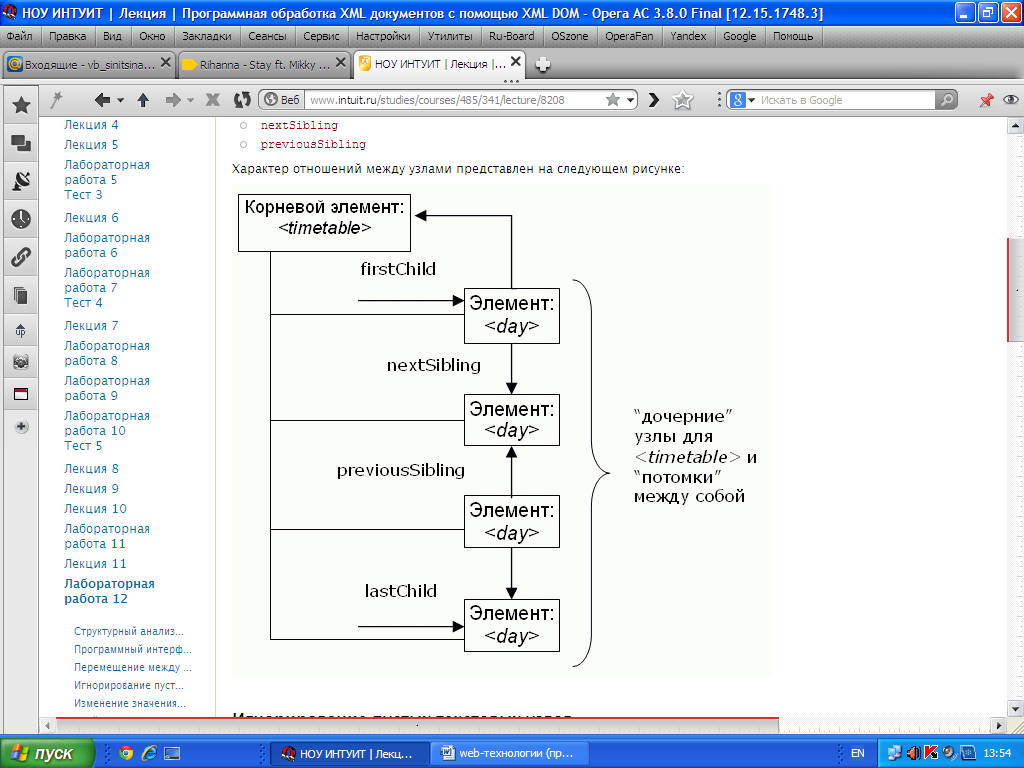


Рисунок 7 - Характер отношений между узлами

*Игнорирование пустых текстовых узлов*

Firefox и некоторые другие браузеры воспринимают неотображаемые символы как текстовые узлы (в отличие от Internet Explorer). Такая ситуация приводит к проблемам при использовании свойств firstChild, lastChild, nextSibling, previousSibling. Для того, чтобы игнорировать такие пустые текстовые узлы можно использовать следующий прием:

function get\_nextSibling(n)

{

y = n.nextSibling;

while (y.nodeType!=1)

{

y = y.nextSibling;

}

return y;

}

Поскольку узлы элементов имеют тип 1, то в том случае, когда узел-потомок не является узлом элемента, будет происходить перемещение к следующему узлу до тех пор, пока не будет найден узел элемента.

*Изменение значения атрибута*

Узлы атрибутов могут принимать текстовые значения. Изменение этого значения реализуется либо через метод setAttribute(), либо через свойство узла атрибута nodeValue

Метод setAttribute() изменяет значение существующего атрибута или создает новый атрибут.

Например:

xmlDoc = loadXMLDoc("timetable.xml");

x = xmlDoc.getElementsByTagName('lesson');

x[0].setAttribute("type","lab");

Свойство nodeValue можно использовать для изменения значения атрибута узла:

xmlDoc = loadXMLDoc("timetable.xml");

x = xmlDoc.getElementsByTagName("lesson")[0];

y = x.getAttributeNode("type");

y.nodeValue = "lab";

Удаление узла из дерева реализуется с помощью метода removeChild():

xmlDoc=loadXMLDoc("timetable.xml ");

y=xmlDoc.getElementsByTagName("lesson")[0];

xmlDoc.documentElement.removeChild(y);

Свойства объекта Node

IE: Internet Explorer, F: Firefox, O: Opera, W3C: (Стандарт)

Таблица 4 – Свойства объекта Node

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Свойство | Описание | IE | F | O | W3C |
| Версия | | |
| baseURI | Возвращает абсолютный URI узла | Нет | 1 | Нет | Да |
| childNodes | Возвращает свойство NodeList (список доч.узлов) | 5 | 1 | 9 | Да |
| firstChild | Возвращает первый дочерний узел | 5 | 1 | 9 | Да |
| lastChild | Возвращает последний дочерний узел | 5 | 1 | 9 | Да |
| localName | Возвращает локальную часть имени узла | Нет | 1 | 9 | Да |
| namespaceURI | Возвращает URI узла в пространстве имен | Нет | 1 | 9 | Да |
| nextSibling | Возвращает следующий дочерний узел | 5 | 1 | 9 | Да |
| nodeName | Возвращает имя узла в зависимости от типа | 5 | 1 | 9 | Да |
| nodeType | Возвращает тип узла | 5 | 1 | 9 | Да |
| nodeValue | Устанавливает или возвращает значение узла в зависимости от типа | 5 | 1 | 9 | Да |
| ownerDocument | Возвращает корневой элемент (объект document ) для узла | 5 | 1 | 9 | Да |
| parentNode | Возвращает родительский узел | 5 | 1 | 9 | Да |
| prefix | Устанавливает или возвращает префикс пространства имен узла | Нет | 1 | 9 | Да |
| previousSibling | Возвращает непосредственно предшествующий узел | 5 | 1 | 9 | Да |
| textContent | Устанавливает или возвращает текстовое содержимое узла | Нет | 1 | Нет | Да |
| xml | Возвращает XML код узла | 5 | Нет | Нет | Нет |

Таблица 5 – Методы объекта Node

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Метод | Описание | IE | F | O | W3C |
| appendChild() | Добавить новый узел в конец списка дочерних узлов | 5 | 1 | 9 | Да |
| cloneNode() | Клонирование узла | 5 | 1 | 9 | Да |
| compareDocumentPosition() | Сравнение позиций двух узлов | Нет | 1 | Нет | Да |
| getFeature(feature,version) | Возвращает объект DOM, реализующий специализированный API |  |  | Нет | Да |
| getUserData(key) | Возвращает объект, ассоциирующийся с ключем текущего узла. Перед этим объект должен быть ассоциирован с текущим узлом путем вызова setUserData с тем же ключем |  |  | Нет | Да |
| hasAttributes() | Возвращает истинное значение, если узел имеет атрибуты | Нет | 1 | 9 | Да |
| hasChildNodes() | Возвращает истинное значение, если узел имеет дочерние узлы | 5 | 1 | 9 | Да |
| insertBefore() | Вставляет новый узел перед существующим узлом | 5 | 1 | 9 | Да |
| isDefaultNamespace(URI) | Определяет, является ли указанный namespaceURI значением по умолчанию |  |  | Нет | Да |
| isEqualNode() | Проверяет равенство двух узлов | Нет | Нет | Нет | Да |
| isSameNode() | Проверяет идентичность двух уздов | Нет | 1 | Нет | Да |
| isSupported(feature,version) | Определяет поддерживается ли указанная характеристика узлом |  |  | 9 | Да |
| removeChild() | Удаляет дочерний узел | 5 | 1 | 9 | Да |
| replaceChild() | Заменяет дочерний узел | 5 | 1 | 9 | Да |
| setUserData(key,data,handler) | Ассоциирует объект с ключем в узле |  |  | Нет | Да |

***Порядок выполнения лабораторной работы***

При выполнении данной лабораторной работы потребуется XML документ timetable.xml, который использовался в "Структура XML документа. XML схемы" :

<?xml version="1.0"?>

<timetable>

<day dayOfWeek="Monday">

<lesson type="practical">

<timeFrom>08.00</timeFrom>

<timeTo>09.30</timeTo>

<subject>Deutsch</subject>

<teacher>Borisova</teacher>

<room>216</room>

</lesson>

<lesson type="lecture">

<timeFrom>09.40</timeFrom>

<timeTo>11.10</timeTo>

<subject>SAP Administration</subject>

<teacher>Egorov</teacher>

<room>384</room>

</lesson>

<lesson type="practical">

<timeFrom>11.20</timeFrom>

<timeTo>12.50</timeTo>

<subject>SAP Administration</subject>

<teacher>Petrov</teacher>

<room>384</room>

</lesson>

</day>

</timetable>

Создание JavaScript сценария загрузки XML документа.

Создайте текстовый файл loadxmldoc.js, содержащий описание функции загрузки XML документа:

function loadXMLDoc(dname)

{

try //Internet Explorer

{

xmlDoc=new ActiveXObject("Microsoft.XMLDOM");

}

catch(e)

{

try //Firefox, Mozilla, Opera, etc.

{

xmlDoc=document.implementation.createDocument("","",null);

}

catch(e) {alert(e.message)}

}

try

{

xmlDoc.async=false;

xmlDoc.load(dname);

return(xmlDoc);

}

catch(e) { alert(e.message) }

return(null);

}

и сохраните его в той же папке, где находится файл timetable.xml.

Код вызова этой функции может выглядеть следующим образом:

<html>

<head>

<script type="text/javascript" src="loadxmldoc.js">

</script>

</head>

<body>

<script type="text/javascript">

xmlDoc=loadXMLDoc("timetable.xml");

document.write("xmlDoc is loaded, ready for use");

</script>

</body>

</html>

Перемещение по дереву узлов.

Подготовьте следующую HTML страницу:

<html>

<head>

<script type="text/javascript" src="loadxmldoc.js">

</script>

</head>

<body>

<script type="text/javascript">

xmlDoc = loadXMLDoc("timetable.xml");

x = xmlDoc.getElementsByTagName("subject");

for (i=0; i<x.length; i++)

{

document.write(x[i].childNodes[0].nodeValue);

document.write("<br />");

}

</script>

</body>

</html>

После загрузки страницы в браузере можно будет увидеть следующий результат:

DeutschSAP AdministrationSAP Administration

Изменение значения элемента.

Следующий пример демонстрирует изменение значения элемента <subject>:

xmlDoc=loadXMLDoc("timetable.xml");

x=xmlDoc.getElementsByTagName("subject")[0].childNodes[0];

x.nodeValue="Java programming";

x = xmlDoc.getElementsByTagName("subject");

for (i=0; i<x.length; i++)

{

document.write(x[i].childNodes[0].nodeValue);

document.write("<br />");

}

Внесите соответствующие изменения в предыдущую страницу и загрузите ее в браузере.

Перемещение по узлам дерева с использованием отношений между ними.

Следующий код показывает, как используя отношения firstChild и nextSibling можно получить для текущего узла список его дочерних узлов:

x = xmlDoc.getElementsByTagName("lesson")[0].childNodes;

y = xmlDoc.getElementsByTagName("lesson")[0].firstChild;

for (i = 0; i < x.length; i++)

{

if (y.nodeType == 1)

{

document.write(y.nodeName + "<br />");

}

y=y.nextSibling;

}

Внесите необходимые изменения в html страницу и загрузите ее в браузере.

**Контрольное задание**

В приведенном ниже XML документе описана экзаменационная ведомость:

<gradeReport id="120851">

<date>10-06-2008</date>

<subject>Computer Science Fundamentals</subject>

<examiner>prof.Litvinov</examiner>

<gradeList>

<gradeRecord id="1">

<student>Ivanov</student>

<grade>4</grade>

</gradeRecord>

<gradeRecord id="2">

<student>Petrov</student>

<grade>3</grade>

</gradeRecord>

<gradeRecord id="3">

<student>Sidorov</student>

<grade>5</grade>

</gradeRecord>

</gradeList>

</gradeReport>

Используя методы DOM XML, сформируйте HTML страницу, содержащую таблицу из трех столбцов: номер, студент, оценка.

Используя методы DOM XML, замените цифровые значения оценок их словесными эквивалентами, например "4" на "good".

**Лабораторная работа 13. Форматирование и преобразование XML документа с помощью CSS и XSL. XSLT преобразование XML документа**

*Цель работы:* знакомство с методами форматирования и преобразования XML документов на основе XSLT преобразований.

XSLT можно определить следующим образом:

* XSLT обозначает XSL Transformations.
* XSLT является самой важной частью XSL преобразования
* XSLT позволяет преобразовывать один XML в другой XML документ.
* XSLT использует XPath для перемещения по структуре XML документа.
* XSLT является W3C рекомендацией.

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.

Фактически XSLT преобразует исходное XML дерево в результирующее XML дерево.

XSLT использует XPath для поиска информации в XML документе, т.е. XPath является инструментов навигации по элементам и атрибутам XML документов.

В процессе преобразования XSLT использует XPath для поиска частей исходного документа, соответствующих одному или более заданных шаблонов. Когда соответствие найдено XSLT преобразует найденную часть исходного документа в соответствующую часть результирующего документа.

XSLT является W3C рекомендацией с ноября 1999 года.

Подавляющее большинство браузеров имеют поддержку XML и XSLT.

Internet Explorer

Начиная с 6 версии, Internet Explorer поддерживает XML, пространства имен, CSS, XSLT и XPath. Версия 5 не совместима с официальной W3C XSL Рекомендацией.

Mozilla Firefox

Начиная с версии 1.0.2, Firefox поддерживает XML и XSLT (CSS).

Mozilla

Mozilla содержит Expat for XML парсер поддерживает отображение XML+CSS. Также имеет поддержка пространства имен. Реализует XSLT преобразования.

Netscape

Начиная с версии 8, Netscape использует в качестве ядра Mozilla, и поэтому имеет такую же поддержку XML/XSLT.

Opera

Начиная с версии 9, Opera поддерживает XML и XSLT (CSS). Версия 8 поддерживает только XML+CSS.

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 JavaScript

В приведенных выше примерах XSLT преобразование из XML в XHTML выполнялось самими браузерами на основе таблицы стилей XSL. Однако такое решение, связанное с включением XSL таблицы стилей в XML документ 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.
* Другим решением для браузеров, не поддерживающих XSLT является преобразование XML в XHTML на веб-сервере.

***Порядок выполнения лабораторной работы***

1.Реализация XSLT преобразования XML документа в XHTML средствами браузера.

Создайте файл ttable.xml следующего содержания:

<?xml version="1.0"?>

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

<timetable>

<lesson>

<timeFrom>09.30</timeFrom>

<timeTo>11.00</timeTo>

<subject>Deutsch</subject>

<teacher>Borisova</teacher>

</lesson>

<lesson>

<timeFrom>11.10</timeFrom>

<timeTo>12.20</timeTo>

<subject>SAP Administration</subject>

<teacher>Petrov</teacher>

</lesson>

<lesson>

<timeFrom>12.40</timeFrom>

<timeTo>14.00</timeTo>

<subject>SAP Administration</subject>

<teacher>Ivanov</teacher>

</lesson>

<lesson>

<timeFrom>14.00</timeFrom>

<timeTo>15.20</timeTo>

<subject>Wen-technology</subject>

<teacher>Loktev</teacher>

</lesson>

</timetable>

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">Subject</th>

<th align="left">Teacher</th>

</tr>

<xsl:for-each select="timetable/lesson">

<tr>

<td><xsl:value-of select="timeFrom"/></td>

<td><xsl:value-of select="subject"/></td>

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

В данном примере для описания шаблона был использован элемент <xsl:template>.

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.

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

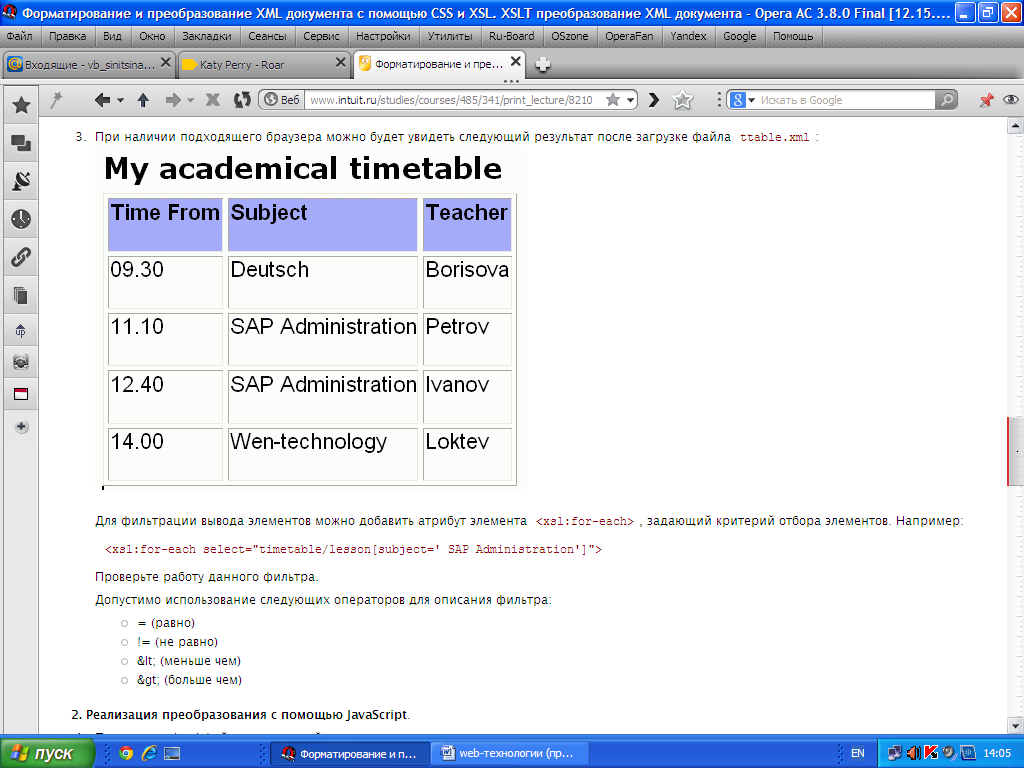


Рисунок 8 – Результат загрузки файла

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

<xsl:for-each select="timetable/lesson[subject=' SAP Administration']">

Проверьте работу данного фильтра.

Допустимо использование следующих операторов для описания фильтра:

= (равно)

!= (не равно)

&lt; (меньше чем)

&gt; (больше чем)

2. Реализация преобразования с помощью JavaScript.

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.

Загрузите этот документ в веб-браузере.

3. Реализация преобразования с помощью JavaScript

Подготовьте файл, содержащий следующий код сценария на языке ASP:

<%

'Load XML

set xml = Server.CreateObject("Microsoft.XMLDOM")

xml.async = false

xml.load(Server.MapPath("ttable.xml"))

'Load XSL

set xsl = Server.CreateObject("Microsoft.XMLDOM")

xsl.async = false

xsl.load(Server.MapPath("ttable.xsl"))

'Transform file

Response.Write(xml.transformNode(xsl))

%>

В начале сценария создается экземпляр парсера Microsoft XML parser (XMLDOM), и XML файл загружается в память. Далее создается еще один экземпляр парсера, и XSL файл загружается в память. В последней строке выполняется собственно преобразование XML файла с использованием XSL файла в XHTML, который отправляется обратно браузеру.

Проверьте работу сценария в веб-браузере.

**Контрольное задание**

В приведенном ниже XML документе описана экзаменационная ведомость:

<gradeReport id="120851">

<date>10-06-2008</date>

<subject>Computer Science Fundamentals</subject>

<examiner>prof.Litvinov</examiner>

<gradeList>

<gradeRecord id="1">

<student>Ivanov</student>

<grade>4</grade>

</gradeRecord>

<gradeRecord id="2">

<student>Petrov</student>

<grade>3</grade>

</gradeRecord>

<gradeRecord id="3">

<student>Sidorov</student>

<grade>5</grade>

</gradeRecord>

</gradeList>

</gradeReport>

Постройте для него XSL файл, необходимый для XSLT преобразования исходного XML документа в HTML страницу, содержащую заголовок ведомости и таблицу оценок со столбцами: номер, студент, оценка.

**Лабораторная работа 14. «Разработка веб-службы в ASPNET»**

*Цель работы:* знакомство с инструментом разработки ASP.NET веб-служб в среде Microsoft Visual Studio.

Рассмотрим в качестве примера создание с помощью ASP.NET веб-службы, которая переводит любое целое десятичное число в один из форматов по выбору: двоичный, восьмеричный, десятичный.

Создайте новый проект (тип: ASP Web Service), например, под именем ASPNETCalcWebService. В результате будет автоматически сгенерирован файл Service1.asmx.cs.

Программная логика веб-службы будет реализована на языке C# в CodeBehind файле Service1.asmx.cs:

using System;

using System.Collections;

using System.ComponentModel;

using System.Data;

using System.Web;

using System.Web.Services;

using System.Web.Services.Protocols;

namespace ASPNETCalcWebService

{

/// <summary>

/// Summary description for Service1

/// </summary>

[WebService(Namespace = "http://tempuri.org/")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

[ToolboxItem(false)]

public class Service1 : System.Web.Services.WebService

{

[WebMethod]

public string HelloWorld()

{

return "Hello World";

}

}

}

Для реализации логики веб-службы в этом файле замените в файле Service1.asmx.cs метод HelloWorld() на 3 новых метода, при помощи которых будут выполняться все преобразования:

using System;

using System.Collections;

using System.ComponentModel;

using System.Data;

using System.Web;

using System.Web.Services;

using System.Web.Services.Protocols;

namespace ASPNETCalcWebService

{

/// <summary>

/// Summary description for Service1

/// </summary>

[WebService(Namespace = "http://tempuri.org/")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

[ToolboxItem(false)]

public class Service1 : System.Web.Services.WebService

{

//Uncomment the following line if using designed components

//InitializeComponent();

}

// Преобразование в двоичную систему счисления

[WebMethod]

public string Binary(int x) {

return Convert.ToString(x, 2);

}

// Преобразование в восьмеричную систему счисления

[WebMethod]

public string Octal(int x)

{

return Convert.ToString(x, 8);

}

// Преобразование в шестнадцатиричную систему счисления

[WebMethod]

public string Hexadecimal(int x)

{

return Convert.ToString(x, 16);

}

}

26.1.

Атрибут WebMethod в этом файле указывает на то, что описываемый метод должен быть доступен по протоколу HTTP для пользователей.

Откомпилируйте и запустите проекта. В результате в браузере должна следующая страница

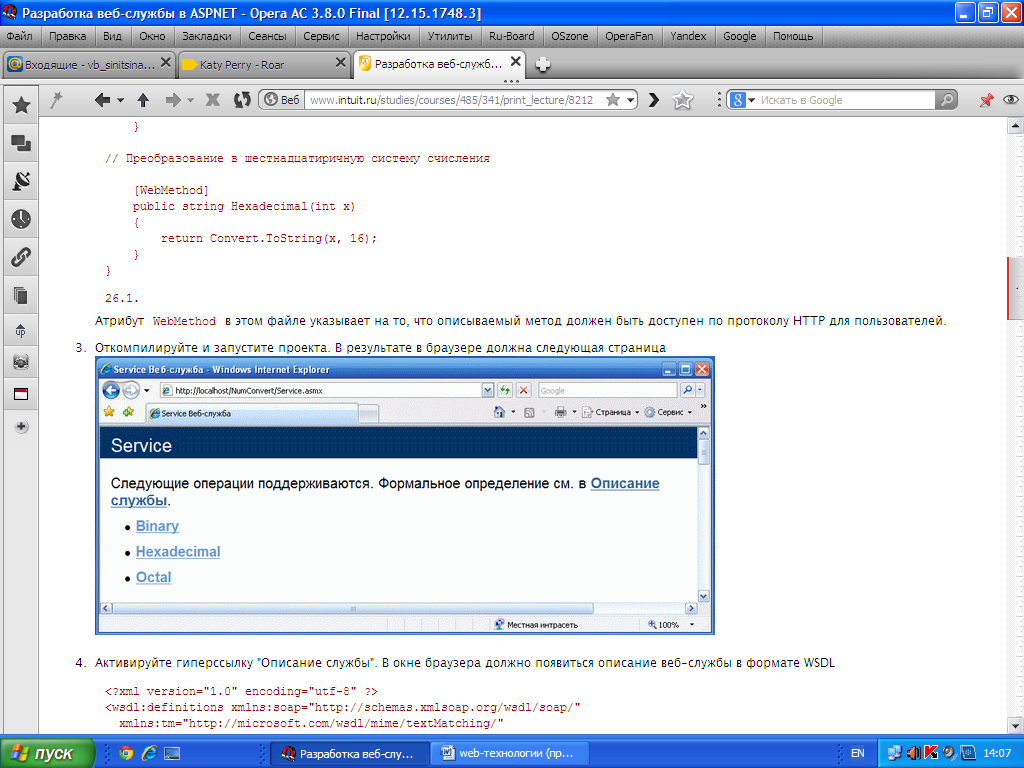


Рисунок 9 – Страница в браузере

Активируйте гиперссылку "Описание службы". В окне браузера должно появиться описание веб-службы в формате WSDL

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

<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"

xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"

xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"

xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"

xmlns:tns="http://tempuri.org/"

xmlns:s="http://www.w3.org/2001/XMLSchema"

xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"

xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"

targetNamespace="http://tempuri.org/"

xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">

<wsdl:types>

<s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">

<s:element name="Binary">

<s:complexType>

<s:sequence>

<s:element minOccurs="1" maxOccurs="1" name="x" type="s:int" />

</s:sequence>

</s:complexType>

</s:element>

<s:element name="BinaryResponse">

<s:complexType>

<s:sequence>

<s:element minOccurs="0" maxOccurs="1" name="BinaryResult" type="s:string" />

</s:sequence>

</s:complexType>

</s:element>

<s:element name="Octal">

<s:complexType>

<s:sequence>

<s:element minOccurs="1" maxOccurs="1" name="x" type="s:int" />

</s:sequence>

</s:complexType>

</s:element>

<s:element name="OctalResponse">

<s:complexType>

<s:sequence>

<s:element minOccurs="0" maxOccurs="1" name="OctalResult" type="s:string" />

</s:sequence>

</s:complexType>

</s:element>

<s:element name="Hexadecimal">

<s:complexType>

<s:sequence>

<s:element minOccurs="1" maxOccurs="1" name="x" type="s:int" />

</s:sequence>

</s:complexType>

</s:element>

<s:element name="HexadecimalResponse">

<s:complexType>

<s:sequence>

<s:element minOccurs="0" maxOccurs="1" name="HexadecimalResult" type="s:string" />

</s:sequence>

</s:complexType>

</s:element>

</s:schema>

</wsdl:types>

<wsdl:message name="BinarySoapIn">

<wsdl:part name="parameters" element="tns:Binary" />

</wsdl:message>

<wsdl:message name="BinarySoapOut">

<wsdl:part name="parameters" element="tns:BinaryResponse" />

</wsdl:message>

<wsdl:message name="OctalSoapIn">

<wsdl:part name="parameters" element="tns:Octal" />

</wsdl:message>

<wsdl:message name="OctalSoapOut">

<wsdl:part name="parameters" element="tns:OctalResponse" />

</wsdl:message>

<wsdl:message name="HexadecimalSoapIn">

<wsdl:part name="parameters" element="tns:Hexadecimal" />

</wsdl:message>

<wsdl:message name="HexadecimalSoapOut">

<wsdl:part name="parameters" element="tns:HexadecimalResponse" />

</wsdl:message>

<wsdl:portType name="ServiceSoap">

<wsdl:operation name="Binary">

<wsdl:input message="tns:BinarySoapIn" />

<wsdl:output message="tns:BinarySoapOut" />

</wsdl:operation>

<wsdl:operation name="Octal">

<wsdl:input message="tns:OctalSoapIn" />

<wsdl:output message="tns:OctalSoapOut" />

</wsdl:operation>

<wsdl:operation name="Hexadecimal">

<wsdl:input message="tns:HexadecimalSoapIn" />

<wsdl:output message="tns:HexadecimalSoapOut" />

</wsdl:operation>

</wsdl:portType>

<wsdl:binding name="ServiceSoap" type="tns:ServiceSoap">

<soap:binding transport="http://schemas.xmlsoap.org/soap/http" />

<wsdl:operation name="Binary">

<soap:operation soapAction="http://tempuri.org/Binary" style="document" />

<wsdl:input>

<soap:body use="literal" />

</wsdl:input>

<wsdl:output>

<soap:body use="literal" />

</wsdl:output>

</wsdl:operation>

<wsdl:operation name="Octal">

<soap:operation soapAction="http://tempuri.org/Octal" style="document" />

<wsdl:input>

<soap:body use="literal" />

</wsdl:input>

<wsdl:output>

<soap:body use="literal" />

</wsdl:output>

</wsdl:operation>

<wsdl:operation name="Hexadecimal">

<soap:operation soapAction="http://tempuri.org/Hexadecimal" style="document" />

<wsdl:input>

<soap:body use="literal" />

</wsdl:input>

<wsdl:output>

<soap:body use="literal" />

</wsdl:output>

</wsdl:operation>

</wsdl:binding>

<wsdl:binding name="ServiceSoap12" type="tns:ServiceSoap">

<soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />

<wsdl:operation name="Binary">

<soap12:operation soapAction="http://tempuri.org/Binary" style="document" />

<wsdl:input>

<soap12:body use="literal" />

</wsdl:input>

<wsdl:output>

<soap12:body use="literal" />

</wsdl:output>

</wsdl:operation>

<wsdl:operation name="Octal">

<soap12:operation soapAction="http://tempuri.org/Octal" style="document" />

<wsdl:input>

<soap12:body use="literal" />

</wsdl:input>

<wsdl:output>

<soap12:body use="literal" />

</wsdl:output>

</wsdl:operation>

<wsdl:operation name="Hexadecimal">

<soap12:operation soapAction="http://tempuri.org/Hexadecimal" style="document" />

<wsdl:input>

<soap12:body use="literal" />

</wsdl:input>

<wsdl:output>

<soap12:body use="literal" />

</wsdl:output>

</wsdl:operation>

</wsdl:binding>

<wsdl:service name="Service">

<wsdl:port name="ServiceSoap" binding="tns:ServiceSoap">

<soap:address location="http://localhost/NumConvert/Service.asmx" />

</wsdl:port>

<wsdl:port name="ServiceSoap12" binding="tns:ServiceSoap12">

<soap12:address location="http://localhost/NumConvert/Service.asmx" />

</wsdl:port>

</wsdl:service>

</wsdl:definitions>

14.2.

Если необходимо сообщить более подробную информацию для пользователей по каждому из доступных в этой веб-службе методов достаточно будет добавить параметр Description в атрибуте WebMethod, например:

[WebMethod (Description = "Перевод целого числа в двоичную систему счисления")]

public string Binary(int x)

{

return Convert.ToString(x, 2);

}

В результате в браузере будет получена следующая страница:

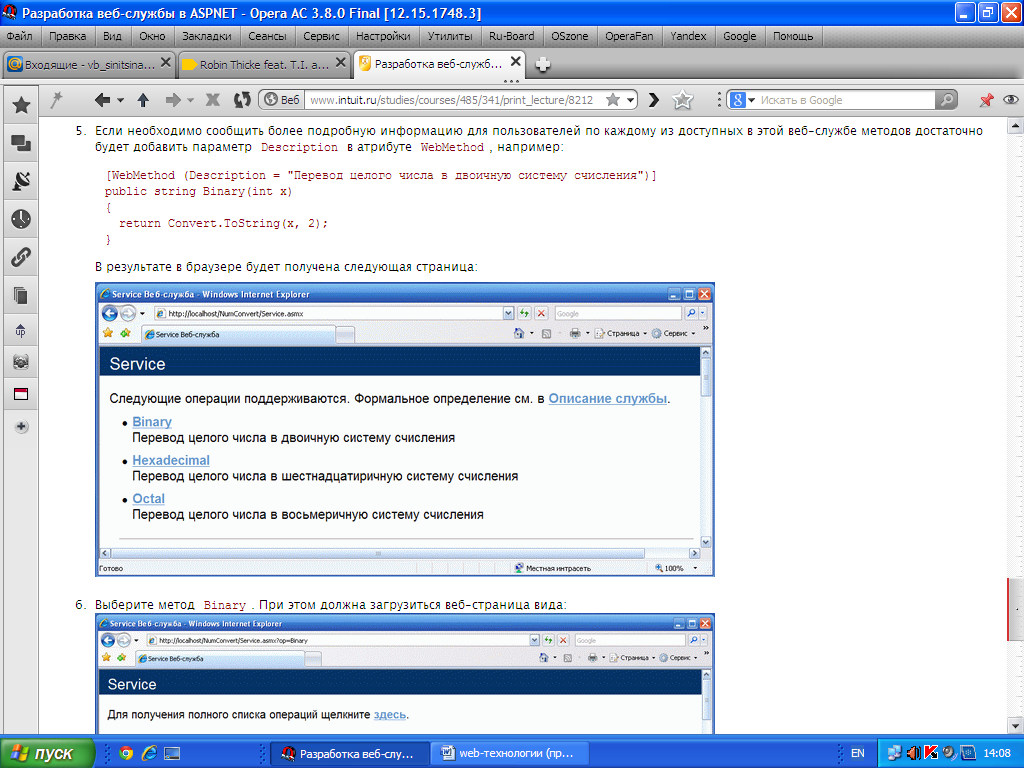


Рисунок 10 - Страница в браузере

Выберите метод Binary. При этом должна загрузиться веб-страница вида:

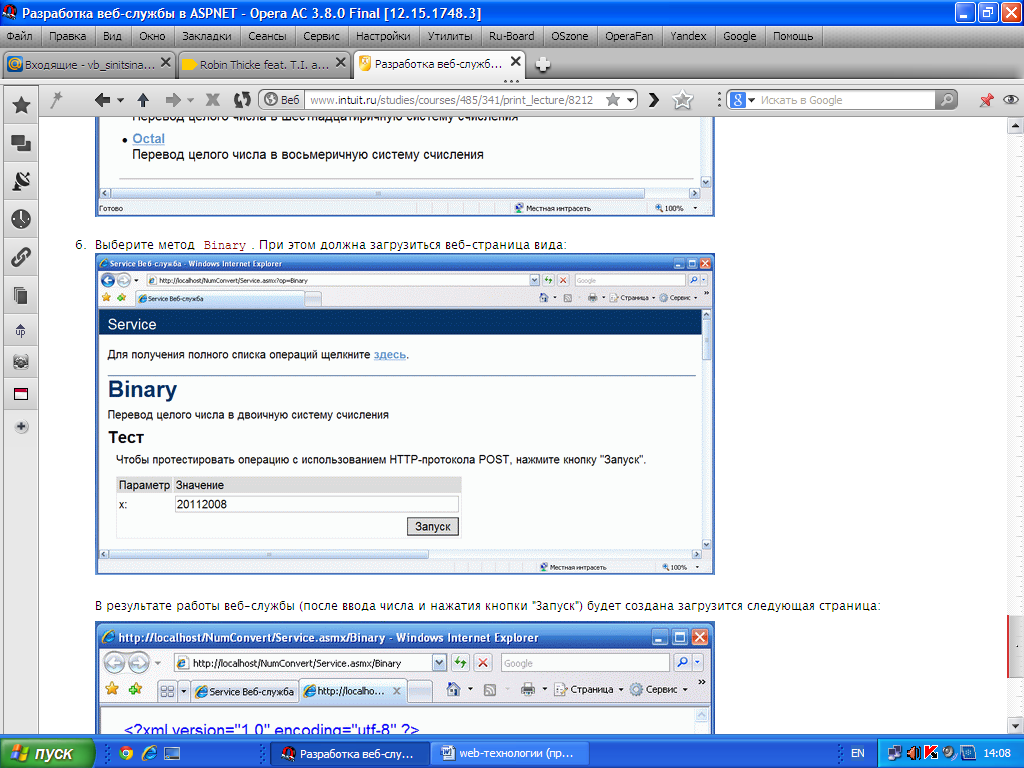


Рисунок 11 - Страница в браузере

В результате работы веб-службы (после ввода числа и нажатия кнопки "Запуск") будет создана загрузится следующая страница:

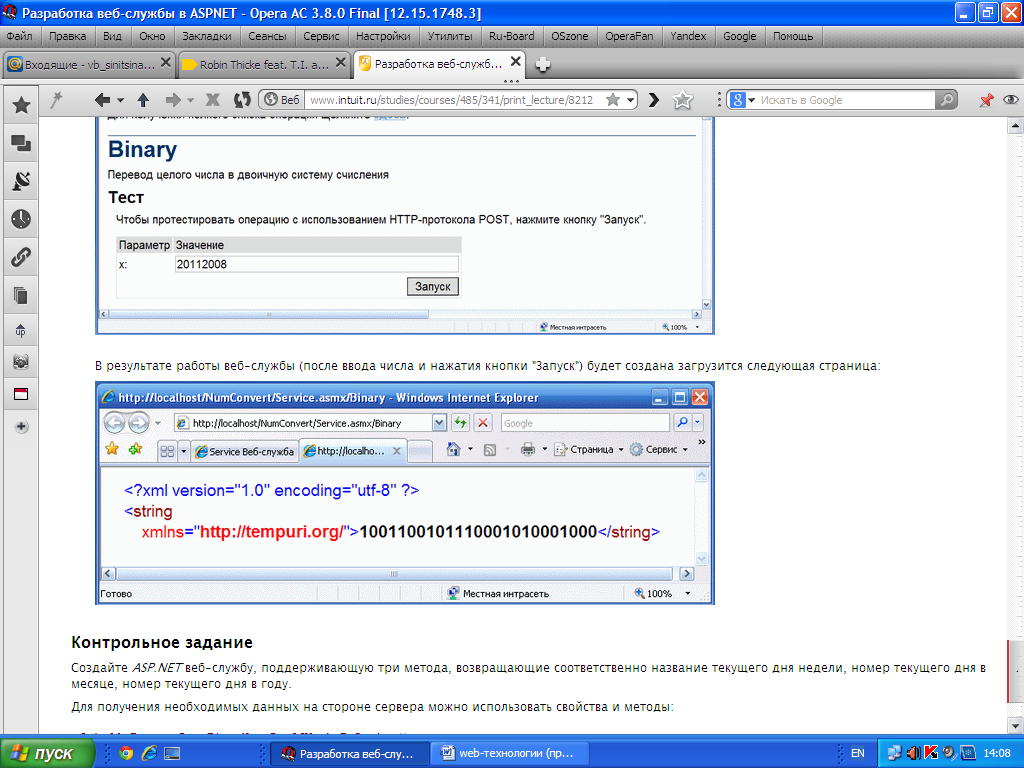


Рисунок 12 - Страница в браузере

**Контрольное задание**

Создайте ASP.NET веб-службу, поддерживающую три метода, возвращающие соответственно название текущего дня недели, номер текущего дня в месяце, номер текущего дня в году.

Для получения необходимых данных на стороне сервера можно использовать свойства и методы:

Label1.Text = DateTime.Now.DayOfWeek.ToString();

Label1.Text = DateTime.Now.Day.ToString();

Label1.Text = DateTime.Now.DayOfYear.ToString();

**Лабораторная работа 15. «Разработка веб-службы в ASP.NET. Создание прокси-сборки для веб-службы»**

*Цель работы:* продолжение знакомства с инструментами разработки ASP.NET веб-служб и их использования с помощью прокси-сборки в среде Microsoft Visual Studio.

В предыдущей лабораторной работе рассматривалось создание веб-службы ASPNETCalcWebService, выполнявшей перевод десятичного целого числа в системы счисления с основаниями 2,8,16. При этом передача данных клиентом (по протоколу SOAP) и получение их производилось в XML формате. Это может быть неудобным для клиента и потребовать разработку дополнительных компонентов для конвертирования данных в привычный для конечного пользователя формат.

Избежать этого можно, если обращаться к веб-службе не напрямую, а с помощью прокси-сборки на C# или другом языке программирования. Такая прокси-сборка создается автоматически в Visual Studio.NET, например, в виде клиента Windows Forms, и берет на себя всю работу по взаимодействию с веб-службой.

***Порядок выполнения лабораторной работы***

1. Создание веб-службы

Cоздайте веб-сайт для проекта типа "ASP.NET Web Service"

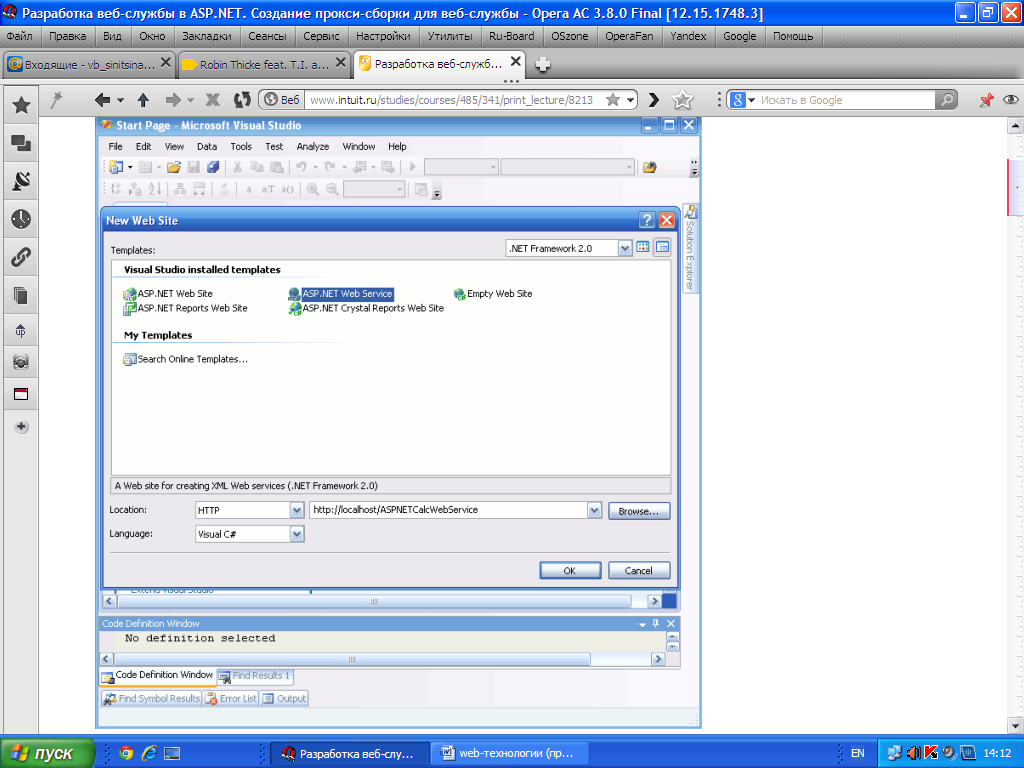


Рисунок 13 – Создание веб-службы

В составе нового проекта будут автоматически созданы файлы Service.asmx (содержащий код представления) и Service.cs (содержащий программный код).

Замените в файле Service.cs метод HelloWorld() на методы, реализующие все преобразования (см. лабораторную работу 14):

// Преобразование в двоичную систему счисления

[WebMethod]

public string Binary(int x) {

return Convert.ToString(x, 2);

}

// Преобразование в восьмеричную систему счисления

[WebMethod]

public string Octal(int x)

{

return Convert.ToString(x, 8);

}

// Преобразование в шестнадцатиричную систему счисления

[WebMethod]

public string Hexadecimal(int x)

{

return Convert.ToString(x, 16);

}

Откомпилируйте проект и выполните пробный запуск веб-службы. URL службы для загрузки в браузере: http://localhost/ASPNETCalcWebService/Service.asmx

2.Создание прокси-сборки.

Создайте проекта с именем, например WebProxy (тип проекта Windows Forms Application).

С помощью дизайнера форм MS Visual Studio подготовьте интерфейс взаимодействия с пользователем в виде формы с двумя текстовыми полями ввода данных, полем отображения результата и кнопкой:

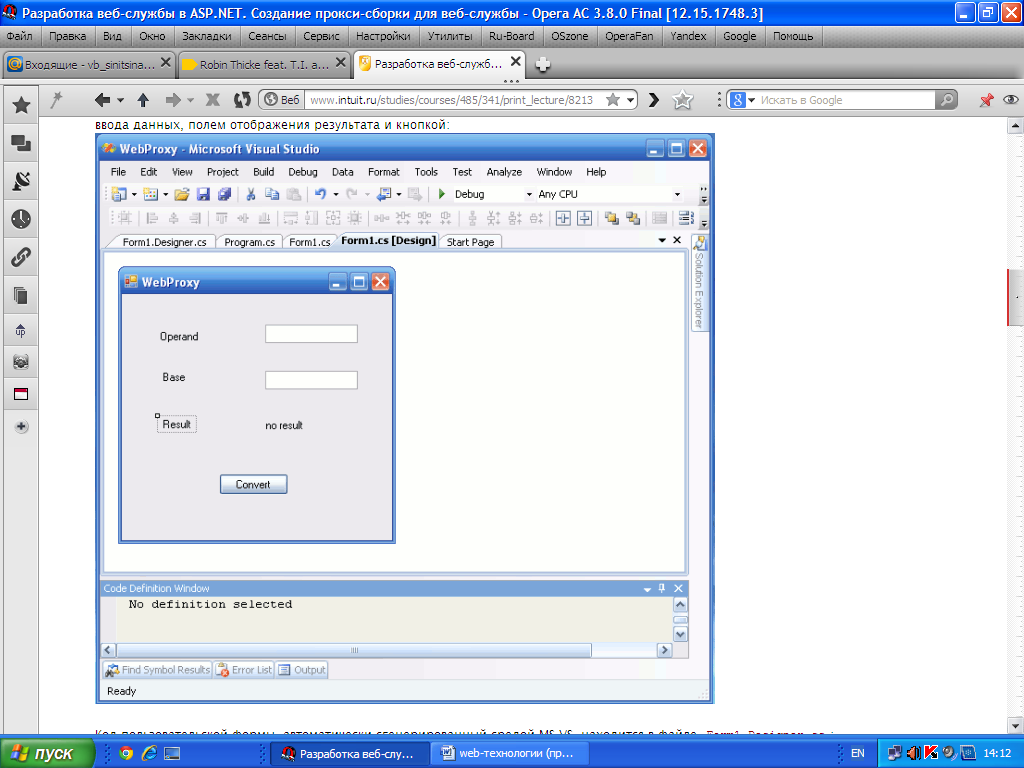


Рисунок 14 – Создание веб-службы

Код пользовательской формы, автоматически сгенерированный средой MS VS, находится в файле Form1.Designer.cs:

namespace WebProxy

{

partial class WebProxy

{

private System.ComponentModel.IContainer components = null;

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

private void InitializeComponent()

{

this.label1 = new System.Windows.Forms.Label();

this.textBox1 = new System.Windows.Forms.TextBox();

this.textBox2 = new System.Windows.Forms.TextBox();

this.button1 = new System.Windows.Forms.Button();

this.label2 = new System.Windows.Forms.Label();

this.label3 = new System.Windows.Forms.Label();

this.label4 = new System.Windows.Forms.Label();

this.SuspendLayout();

//

// label1

//

this.label1.AutoSize = true;

this.label1.Location = new System.Drawing.Point(152, 135);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(47, 13);

this.label1.TabIndex = 0;

this.label1.Text = "no result";

//

// textBox1

//

this.textBox1.Location = new System.Drawing.Point(155, 33);

this.textBox1.Name = "textBox1";

this.textBox1.Size = new System.Drawing.Size(100, 20);

this.textBox1.TabIndex = 1;

//

// textBox2

//

this.textBox2.Location = new System.Drawing.Point(155, 83);

this.textBox2.Name = "textBox2";

this.textBox2.Size = new System.Drawing.Size(100, 20);

this.textBox2.TabIndex = 2;

//

// button1

//

this.button1.Location = new System.Drawing.Point(105, 194);

this.button1.Name = "button1";

this.button1.Size = new System.Drawing.Size(75, 23);

this.button1.TabIndex = 3;

this.button1.Text = "Convert";

this.button1.UseVisualStyleBackColor = true;

this.button1.Click += new System.EventHandler(this.button1\_Click);

//

// label2

//

this.label2.AutoSize = true;

this.label2.Location = new System.Drawing.Point(38, 39);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(48, 13);

this.label2.TabIndex = 4;

this.label2.Text = "Operand";

//

// label3

//

this.label3.AutoSize = true;

this.label3.Location = new System.Drawing.Point(41, 83);

this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(31, 13);

this.label3.TabIndex = 5;

this.label3.Text = "Base";

//

// label4

//

this.label4.AutoSize = true;

this.label4.Location = new System.Drawing.Point(41, 134);

this.label4.Name = "label4";

this.label4.Size = new System.Drawing.Size(37, 13);

this.label4.TabIndex = 6;

this.label4.Text = "Result";

//

// WebProxy

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(292, 266);

this.Controls.Add(this.label4);

this.Controls.Add(this.label3);

this.Controls.Add(this.label2);

this.Controls.Add(this.button1);

this.Controls.Add(this.textBox2);

this.Controls.Add(this.textBox1);

this.Controls.Add(this.label1);

this.Name = "WebProxy";

this.Text = "WebProxy";

this.Load += new System.EventHandler(this.WebProxy\_Load);

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.Label label1;

private System.Windows.Forms.TextBox textBox1;

private System.Windows.Forms.TextBox textBox2;

private System.Windows.Forms.Button button1;

private System.Windows.Forms.Label label2;

private System.Windows.Forms.Label label3;

private System.Windows.Forms.Label label4;

}

}

27.1.

Добавьте веб-ссылку http://localhost/ASPNETCalcWebService/Service.asmx на веб-службу ASPNETCalcWebService в проект:

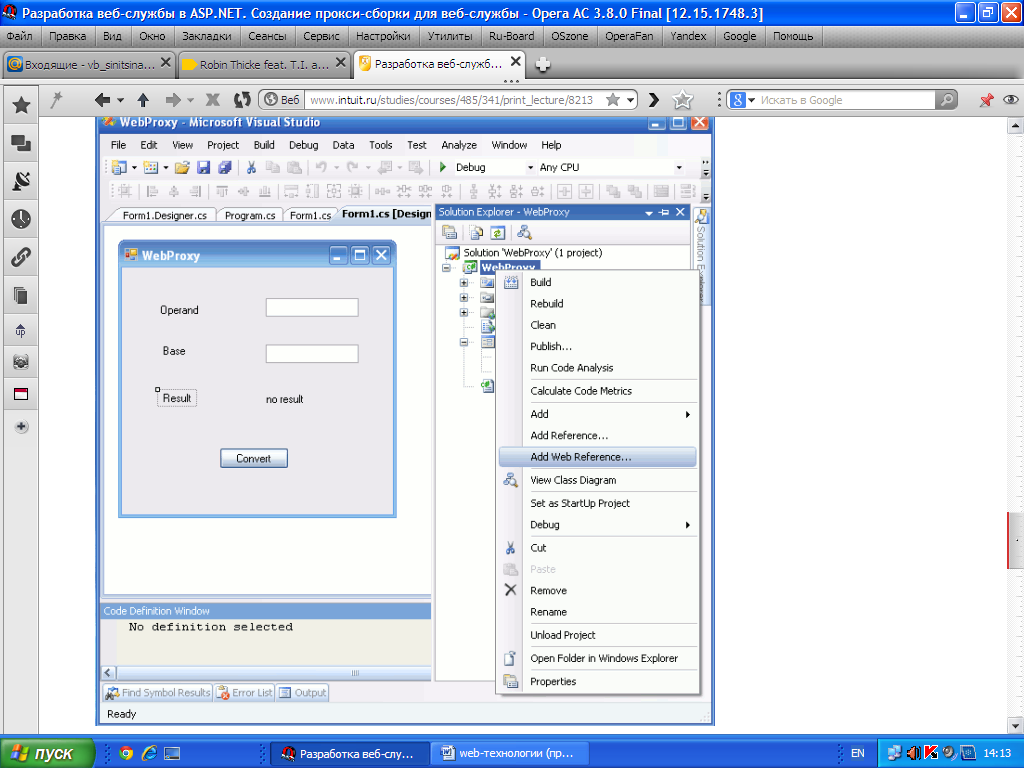


Рисунок 15 – Создание веб-службы

Добавьте обработчик события "нажатие кнопки" в файле Form1.cs:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace WebProxy

{

// добавлено пространство имен localhost

using localhost;

public partial class WebProxy : Form

{

public WebProxy()

{

InitializeComponent();

}

// обработка события нажатия кнопки

private void button1\_Click(object sender, EventArgs e)

{

// создание объекта, связанного с веб-службой

localhost.Service w = new localhost.Service();

string ans;

// преобразование данных из текстовых полей ввода в форме в целые числа

// operand – операнд, NumBase – основание системы счисления

int NumBase = int.Parse(textBox2.Text);

int operand = int.Parse(textBox1.Text);

// выбор подходящей функции веб-службы и взаимодействие с ней

switch (NumBase)

{

case 2: ans = w.Binary(operand); break;

case 8: ans = w.Octal(operand); break;

case 16: ans = w.Hexadecimal(operand); break;

default: ans = "base is undefined"; break;

}

// отображение результата вычисления в форме

label1.Text = ans;

}

}

}

27.2.

Откомпилируйте проект

Проверка работы программы. На скриншоте ниже показана работа программы WebProxy (совместно с веб-службой ASPNETCalcWebService):

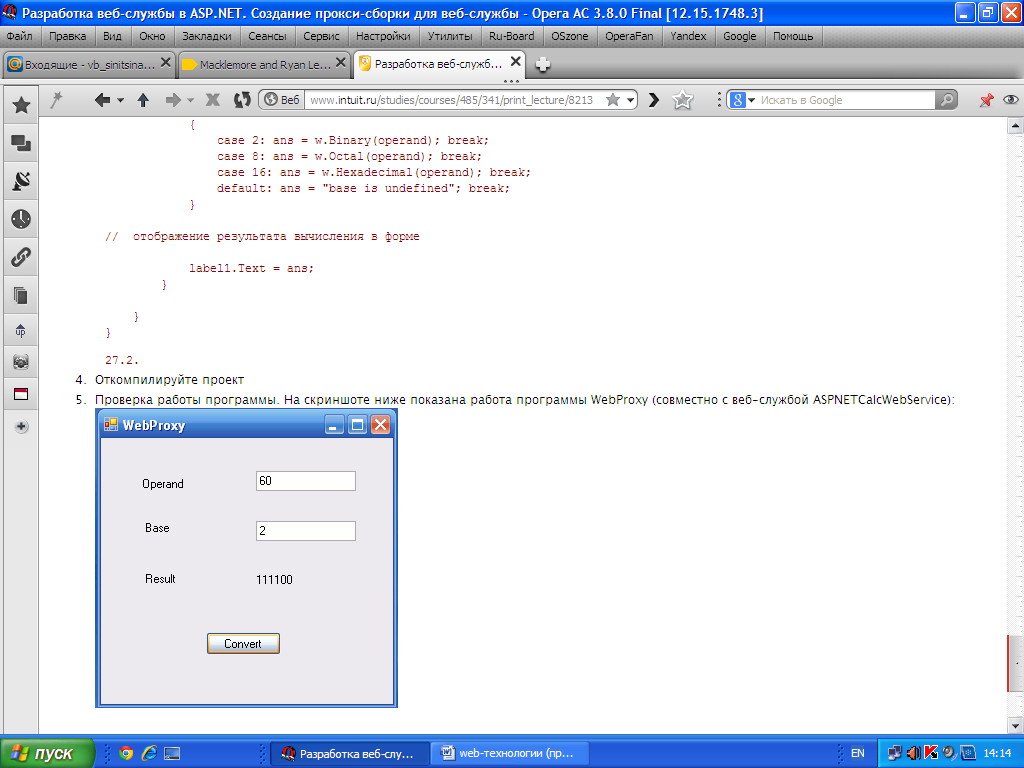


Рисунок 16 – Результат работы программы WebProxy

**Контрольное задание**

Создайте ASP.NET прокси-сборку на языке C# для веб-службы, поддерживающей три метода, возвращающие соответственно название текущего дня недели, номер текущего дня в месяце, номер текущего дня в году.

Для получения необходимых данных на стороне сервера можно использовать свойства и методы:

Label1.Text = DateTime.Now.DayOfWeek.ToString();

Label1.Text = DateTime.Now.Day.ToString();

Label1.Text = DateTime.Now.DayOfYear.ToString();

Выбор метода веб-службы должен быть реализован через форму с элементом типа ListBox или RadioButton.

**Лабораторная работа 16. «Примеры разработки RSS-источников и RSS-ридеров»**

*Цель работы:* Введение в технологию RSS. Изучение структуры RSS документов, их генерации и публикации.

RSS – это метод распространения веб-контента с веб-сайта на других веб-сайтах. RSS позволяет выполнять быстрый просмотр новостей и изменений.

На кого ориентирован формат RSS? В первую очередь RSS предназначен для использования на очень часто обновляемых веб-сайтах, например:

* Новостные сайты (списки новостей).
* Сайты компаний (списки новостей и продуктов).
* Календари (списки предстоящих событий и знаменательных дней).
* Изменения сайтов (список обновленных и новых страниц).

Официального стандарта RSS как такового не существует. Реально используются следующие форматы:

* Порядка 50% RSS-потоков используют RSS 0.91.
* Порядка 25% используют RSS 1.0.
* Оставшиеся 25% RSS-потоков распределены между RSS 0.9 и RSS 2.0.

Как работает RSS?

Создается RSS документ в виде файла с расширением .xml.

Этот файл размещается на веб-сайте.

RSS-поток регистрируется в RSS-аггрегаторах.

Пример RSS документа

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

<rss version="2.0">

<channel>

<title>My Home Page</title>

<link>http://www.MyHP.edu</link>

<description>Free web building tutorials</description>

<item>

<title>RSS Demo</title>

<link>http:// www.MyHP.edu/rss</link>

<description>New RSS demo on my home page</description>

</item>

<item>

<title>XML Demo</title>

<link> www.MyHP.edu/xml</link>

<description>New XML demo on my home page</description>

</item>

</channel>

</rss>

В первой строке размещено объявление версии XML и кодировки документа.

Вторая строка идентифицирует данный документ как RSS документ версии 2.0.

В третьей строке содержится элемент <channel>, описывающий RSS поток. Он в свою очередь содержит дочерние элементы:

* <title> - описывает заголовок RSS канала.
* <link> - описывает гиперссылку на канал.
* <description> - содержит краткую характеристику канала.

Каждый элемент <channel> может содержать один или более <item> элементов. Каждый элемент <item> описывает отдельную статью RSS источника. В свою очередь каждый элемент <item> имеет три обязательных дочерних элемента:

<title>

<link>

<description>

Кроме обязательных элементов <channel> может содержать некоторые дополнительные дочерние элементы.

Элемент <category> описывает категорию потока и используется RSS аггрегаторами для группировки сайтов на основе категорий.

Элемент <copyright> информирует об авторском праве на данный документ.

Элемент <image> предназначен для отображения изображения при показе потока агрегатором. В свою очередь он содержит три обязательных дочерних элемента:

<url> - URL изображения

<title> - альтернативный текст, отображаемый при невозможности показа изображения

<link> - содержит гиперссылку на веб-сайт, содержащий канал

Пример:

<image>

<url>http://www.myhp.edu/images/me.gif</url>

<title>My home page</title>

<link>http://www.myhp.edu</link>

</image>

Элемент <language> информирует о языке документа. Может быть использован аггрегаторами для группировки сайтов по языку.

Таблица 7 - Список дочерних элементов для элемента <channel>.

|  |  |  |
| --- | --- | --- |
| Элемент | Описание | Статус |
| <category> | Указывается одна или несколько категорий потока | Необязательный |
| <cloud> | Регистрирует процессы, получающие немедленное извещение в случае обновления потока | Необязательный |
| <copyright> | Извещает об авторском праве на документ | Необязательный |
| <description> | Краткое описание канала | Обязательный |
| <docs> | Указывает URL документов по форматам, используемым в потоке | Необязательный |
| <generator> | Указывает на программу, генерирующую поток | Необязательный |
| <image> | Позволяет показывать изображение, представляющее поток в аггрегаторе | Необязательный |
| <language> | Указывается язык потока | Необязательный |
| <lastBuildDate> | Определяется дата последнего обновления содержимого потока | Необязательный |
| <link> | Указывается гиперссылка на канал | Обязательный |
| <managingEditor> | Содержится почтовый адрес редактора содержимого потока | Необязательный |
| <pubDate> | Определяет дату последней публикации содержимого потока | Необязательный |
| <rating> | Содержится PICS рейтинг потока | Необязательный |
| <skipDays> | Указывает дни, в которые аггрегаторы не должны обновлять поток. | Необязательный |
| <skipHours> | Указывается количество часов, в течение которых аггрегаторы не должны обновлять поток | Необязательный |

Продолжение таблицы 7

|  |  |  |
| --- | --- | --- |
| <textInput> | Описывается текстовое поле ввода, отображаемой в потоке | Необязательный |
| <title> | Размещается имя канала | Обязательный |
| <ttl> | Указывается количество минут в течение которых поток может находится в кэше до обновления из источника | Необязательный |
| <webMaster> | Помещается адрес электронной почты веб-дизайнера потока | Необязательный |

Таблица 8 - Список дочерних элементов для элемента < item >

|  |  |  |
| --- | --- | --- |
| Элемент | Описание | Статус |
| <author> | Адрес электронной почты автора темы | Необязательный |
| <category> | Описывает одну или более категорий для данной темы | Необязательный |
| <comments> | Позволяет связать тему с комментариями по данной теме | Необязательный |
| <description> | Краткое описание темы | Обязательный |
| <enclosure> | Позволяет включить мультимедиа файл в данную тему | Необязательный |
| <guid> | Содержит уникальный идентификатор темы | Необязательный |
| <link> | Содержит гиперссылку на тему | Обязательный |
| <pubDate> | Содержит последнюю дату публикации темы | Необязательный |
| <source> | Указывает внешний источник для данной темы | Необязательный |
| <title> | Название темы | Обязательный |

*Публикация RSS файла*

Создание RSS потока не ограничивается разработкой RSS документа. Необходимо еще опубликовать этот файл. Для этого потребуется выполнить следующую последовательность действий:

* Выбрать подходящее название для RSS файла. Расширение должно быть .xml.
* Проверить RSS файл на правильность с помощью подходящей программы-валидатора, например, взятой по адресу http://www.feedvalidator.org.
* Разместить RSS файл в подходящем веб-каталоге веб-сервера.
* Скопировать одну из "кнопок":илив веб-каталог.
* Вставить выбранную "кнопку" на исходящую страницу RSS потока в виде гиперссылки на RSS файл, например:

<a href = "www.myhp.edu/rss/myrss.xml">

< img src="http://www.myhp.edu/rss/rss.gif" width="35" height="15">

</a>

* Разместить созданный RSS поток в популярных Каталогах RSS потоков, при этом URL потока должен ссылаться именно на сам XML файл потока, т.е. http://www.myhp.edu/rss/myrss.xml.

Можно также зарегистрировать поток в популярных поисковых системах, например:

Yahoo - http://publisher.yahoo.com/rss\_guide/submit.php

Google - http://www.google.com/ig

MSN - http://w.moreover.com/site/products/ind/pingserver.html

Периодическое обновление потока.

В принципе, можно самостоятельно заниматься формированием данных в соответствующем формате RSS для работы потока. Однако имеются достаточно удобные высокоуровневые средства для автоматизации данной работы. Например:

MyRSSCreator - http://www.myrsscreator.com/

FeedFire - http://www.feedfire.com/site/index.html.

Для чтения RSS потоков используются специальные программы чтения – RSS ридеры. Некоторые из браузеров имеют встроенные RSS ридеры. В частности, веб-браузер MS Internet Explorer 7 может интерпретировать XML файлы RSS потоков и корректно их отображать.

*Добавление RSS-канала с помощью Microsoft Internet Explorer версии 7*

При использовании MS Internet Explorer версии 7 и Office Outlook 2007 можно добавлять и просматривать RSS-каналы с помощью любой из этих программ.

При просмотре веб-страницы, содержащей RSS, в Internet Explorer рядом с кнопкой( Домашняя страница) отображается кнопка.

Нажмите кнопку.

На веб-странице отобразится список доступных RSS-каналов.

Выберите RSS-канал, который необходимо добавить.

Можно также нажать кнопки, или, расположенные на веб-странице.

***Порядок выполнения лабораторной работы***

1. Cоздания RSS документа с помощью PHP сценария и с использованием интерфейса DOM XML.

Подготовьте файл, содержащий сценарий на языке PHP:

<?php

// Создается новый XML документ.

$dfeed = new DOMDocument('1.0', 'utf-8');

// Создание корневого элемента <rss>

$erss = $dfeed->createElement('rss');

$erss->setAttribute('version', '2.0');

// И добавление его в дерево документа

$dfeed->appendChild($erss);

// Создание элемента <channel> и добавление его к <rss>

$echannel = $dfeed->createElement('channel');

$erss->appendChild($echannel);

// Создание и добавление в channel

// узлов <title>, <link>, <description>, <language>, <pubDate>

$echannel->appendChild( $dfeed->createElement('title', 'RSS-channel title') );

$echannel->appendChild( $dfeed ->createElement('link', 'http://www.myhp.edu') );

$echannel->appendChild( $dfeed ->createElement('description', 'my RSS demo') );

$echannel->appendChild( $dfeed ->createElement('language', 'en') );

$echannel->appendChild( $dfeed ->createElement('pubDate', date('r')) );

// Добавление к узлу <channel> 2 узлов <item>

for ( $i = 1; $i <= 2; $i++ )

{

$eitem = $dfeed->createElement('item');

$echannel->appendChild($eitem);

// Создание дочерних элементов для <item>

$eitem->appendChild( $dfeed->createElement('title', 'Item'.$i) );

$eitem->appendChild( $dfeed->createElement('link', 'http://www.myhp.edu/rss/'.$i.'.xml') );

$eitem->appendChild( $dfeed->createElement('description', 'Description for '.$i.' item') );

}

// Сохранение документа в файле demo.rss

$dfeed->save('demo.rss');

?>

30.1.

Созданный файл разместите на веб-сервере, настройте права доступа (право на запись) для веб-сервера к директории, в которой размещается сценарий, либо к специальной директории, в которой будет создан файл demo.rss (это более безопасно).

Выполните сценарий по запросу с веб-браузера.

В результате выполнения сценария получится следующий документ:

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

<rss version="2.0">

<channel>

<title>RSS-channel title</title>

<link>http://www.myhp.edu</link>

<description>my RSS demo</description>

<language>en</language>

<pubDate>Thu, 06 Nov 2008 01:50:53 +0300</pubDate>

<item>

<title>Item1</title>

<link>http://www.myhp.edu/rss/1.xml</link>

<description>Description for 1 item</description>

</item>

<item>

<title>Item2</title>

<link>http://www.myhp.edu/rss/2.xml</link>

<description>Description for 2 item</description>

</item>

</channel>

</rss>

2. Чтение RSS документа с помощью PHP сценария и с использованием интерфейса DOM XML.

Подготовьте файл, содержащий сценарий на языке PHP:

<?php

// Загрузка документа

$dfeed = DOMDocument::load('demo.rss');

// Чтение элемента <channel>

$echannel = $dfeed->firstChild->firstChild;

// Чтение первого дочернего элемента узла <channel> т.е.< title>

$cchild = $echannel->firstChild;

// Вывод содержимого дочерних элементов узла <channel> до элемента <item>:

print "<strong><br>";

while ( $cchild->tagName != 'item' )

{

print $cchild->nodeValue. "<br>";

$cchild = $cchild->nextSibling;

}

print "</strong><br>";

// Получение списка узлов <item>

$litems = $echannel->getElementsByTagName('item');

print "<br>";

foreach ( $litems as $eitem )

{

$title = $eitem->firstChild->nodeValue;

$link = $eitem->firstChild->nextSibling->nodeValue;

$descr = $eitem->lastChild->nodeValue;

print "<a href=".$link.">".$title."</a> :: ".$descr."<br>";

}

print "<br>";

?>

В методе load() в качестве параметра укажите путь к RSS-файлу.

Созданный файл разместите на веб-сервере, настройте права доступа (право на запись) для веб-сервера к директории, в которой размещается сценарий, либо к специальной директории, в которой будет создан файл demo.rss (это более безопасно).

Выполните сценарий по запросу с веб-браузера.

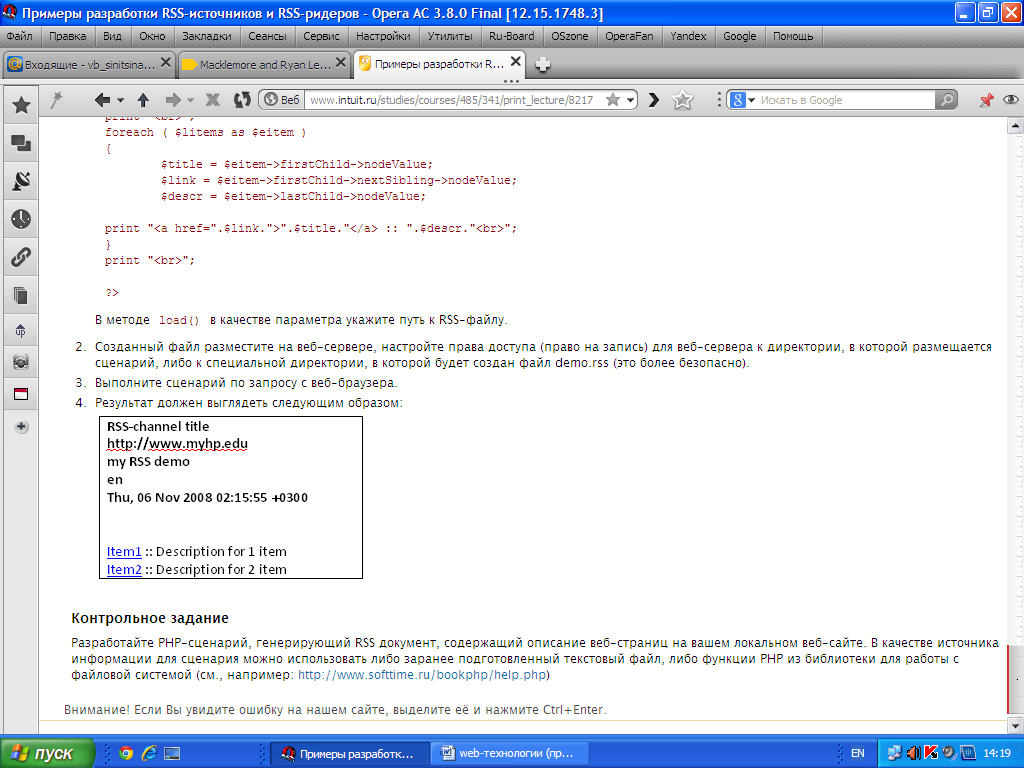


Рисунок 17 - Результат

**Контрольное задание**

Разработайте PHP-сценарий, генерирующий RSS документ, содержащий описание веб-страниц на вашем локальном веб-сайте. В качестве источника информации для сценария можно использовать либо заранее подготовленный текстовый файл, либо функции PHP из библиотеки для работы с файловой системой (см., например: http://www.softtime.ru/bookphp/help.php)

**Лабораторная работа 17. «Реализация асинхронного взаимодействия веб-браузера с веб-сервером с помощью технологии AJAX»**

*Цель работы:* ознакомление с принципами асинхронного взаимодействия между веб-клиентом и веб-сервером в рамках технологии AJAX.

***AJAX = Asynchronous JavaScript and XML***

AJAX – не является новым языком программирования, но технология создания улучшенных, более быстрых и в большей степени интерактивных веб-приложений.

JavaScript сценарий посредством AJAX может непосредственно взаимодействовать с сервером с помощью объекта XMLHttpRequest. Использование данного объекта обмен данными с веб-сервером могут происходить без перезагрузки страницы.

AJAX позволяет веб-страницам запрашивать небольшие объемы информации с сервера нежели целиком всю страницу в результате асинхронной передачи данных (в рамках HTTP протокола) между браузером и сервером.

AJAX не зависит от программного обеспечения веб-сервера и основан на следующих веб-стандартах:

JavaScript

XML

HTML

CSS

Поскольку эти веб-стандарты четко определены и имеют поддержку в наиболее распространенных веб-браузерах, то AJAX приложения являются браузеро- и платформо-независимыми.

Популярность AJAX связана с появлением сервиса Google Suggest в 2005 году. Данный сервис на основе объекта XMLHttpRequest предоставляет в распоряжение пользователя достаточно динамический веб-интерфейс. В процессе ввода символов пользователем в поле поискового запроса JavaScript отправляет их на сервер и получает от него список подсказок.

Объект XMLHttpRequest поддерживается в Internet Explorer (начиная 5 версии и выше), Safari 1.2, Mozilla 1.0 / Firefox, Opera 8+ и Netscape 7.

***Порядок выполнения лабораторной работы***

В данной лабораторной работе рассматривается пример системы, имитирующей работу сервиса Google Suggest на основе AJAX.

1. Реализация клиентской части.

Предполагается, что пользователь может вводить в текстовое поле формы название автомобильной марки, получая при этом динамически список вариантов названий, соответствующих уже введенным символам, без перезагрузки страницы.

Создайте следующую веб-страницу:

<html>

<head>

<script src="chint.js"></script>

</head>

<body>

<form>

First Name:

<input type="text" id="txt1" onkeyup="showHint(this.value)">

</form>

<p>Suggestions: <span id="txtHint"></span></p>

</body>

</html>

Как видно из кода, при наступлении события onkeyup (отжатие клавиши) вызывается обработчик showHint().

В файле chint.js сохраните следующий код обработчика

Код обработчика загружается из файла chint.js:

var xmlHttp;

function showHint(str)

{

if (str.length==0)

{

document.getElementById("txtHint").innerHTML="";

return;

}

xmlHttp=GetXmlHttpObject();

if (xmlHttp==null)

{

alert ("Your browser does not support AJAX!");

return;

}

var url = "ghint.php";

url = url + "?q=" + str;

url = url + "&sid=" + Math.random();

xmlHttp.onreadystatechange = stateChanged;

xmlHttp.open("GET", url, true);

xmlHttp.send(null);

}

function stateChanged()

{

if (xmlHttp.readyState==4)

{

document.getElementById("txtHint").innerHTML = xmlHttp.responseText;

}

}

function GetXmlHttpObject()

{

var xmlHttp=null;

try

{

// Firefox, Opera 8.0+, Safari

xmlHttp = new XMLHttpRequest();

}

catch (e)

{

// Internet Explorer

try

{

xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");

}

catch (e)

{

xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");

}

}

return xmlHttp;

}

33.1.

Из кода видно, что каждый раз, когда вводится символ, вызывается функция-обработчик. Если при этом содержимое текстового поля формы непустое (str.length > 0), функция выполняет следующие действия:

* Формируется url для отправки веб-серверу
* Добавляется значение параметра q, равное содержимому текстового поля, к url
* Добавляется к url случайное число для предотвращения кеширования
* Создается объект XMLHTTP, при этом указывается функция (stateChanged) подлежащая исполнению при наступлении события ввода символа
* Открывается объект XMLHTTP с указанным значением url
* Отправляется HTTP запрос веб-серверу

Если поле ввода пустое, происходит очистка содержимого раздела txtHint на веб-странице.

Ключевым моментом в данной системе является использование объекта XMLHttpRequest.

Данный объект по-разному создается в различных браузерах. Так, Internet Explorer для этого использует ActiveXObject, в то время как остальные браузеры используют встроенный в JavaScript объект XMLHttpRequest.

Для поддержки работы системы в разных браузерах использован оператор " try-catch ".

* Сначала делается попытка создать объект XMLHttpRequest для браузеров Firefox, Opera или Safari:

xmlHttp = new XMLHttpRequest().

* В случае неудачи, делается следующая попытка создания объекта для Internet Explorer 6.0 + :

xmlHttp = new ActiveXObject("Msxml2.XMLHTTP").

* Если это также не удается, то делается попытка создания объекта уже для Internet Explorer 5.5 + :

xmlHttp = new ActiveXObject("Microsoft.XMLHTTP").

В случае, если ни одна из этих попыток не принесла успеха, выдается сообщение об отсутствии поддержки AJAX браузером.

2. Реализация серверной части.

Разместите на веб-сервере в файле с именем ghint.php следующий PHP сценарий::

<?php

header("Cache-Control: no-cache, must-revalidate");

// Прошедшая дата

header("Expires: Mon, 1 Sep 2008 07:30:00 GMT");

// Инициализация массива названий

$a[]="Audi";

$a[]="BMW";

$a[]="Buick";

$a[]="Chevrolet";

$a[]="Citroen";

$a[]="Dodge";

$a[]="Ferrari";

$a[]="Fiat";

$a[]="Ford";

$a[]="Honda";

$a[]="Hyundai";

$a[]="Cherokee";

$a[]="Cherry";

$a[]="Lada";

$a[]="Lamborghini";

$a[]="Lincoln";

$a[]="Mazda";

$a[]="Mercedes";

$a[]="Mitsubishi";

$a[]="Nissan";

$a[]="Opel";

$a[]="Peugeot";

$a[]="Plymoth";

$a[]="Pontiac";

$a[]="Renault";

$a[]="Rover";

$a[]="Saab";

$a[]="Subaru";

$a[]="Suzuki";

$a[]="Toyota";

$a[]="Volkswagen";

$a[]="Volvo";

//получение параметра q из URL

$q = $\_GET["q"];

//поиск соответствий из массива если длина q > 0

if (strlen($q) > 0)

{

$hint = "";

for($i = 0; $i<count($a); $i++)

{

if (strtolower($q) == strtolower(substr($a[$i],0,strlen($q))))

{

if ($hint == "")

{

$hint=$a[$i];

}

else

{

$hint=$hint." , ".$a[$i];

}

}

}

}

// Возврат строки "нет вариантов" если соответствий не найдено

// либо найденное соответствие

if ($hint == "")

{

$response = "no suggestion";

}

else

{

$response = $hint;

}

//вывод результата

echo $response;

?>

33.2.

3. Проверка работоспособности системы.

Проверьте с помощью подходящего веб-браузера работу системы.

На скриншоте показана работа в браузере с веб-страницей ghint.html, использующей AJAX:

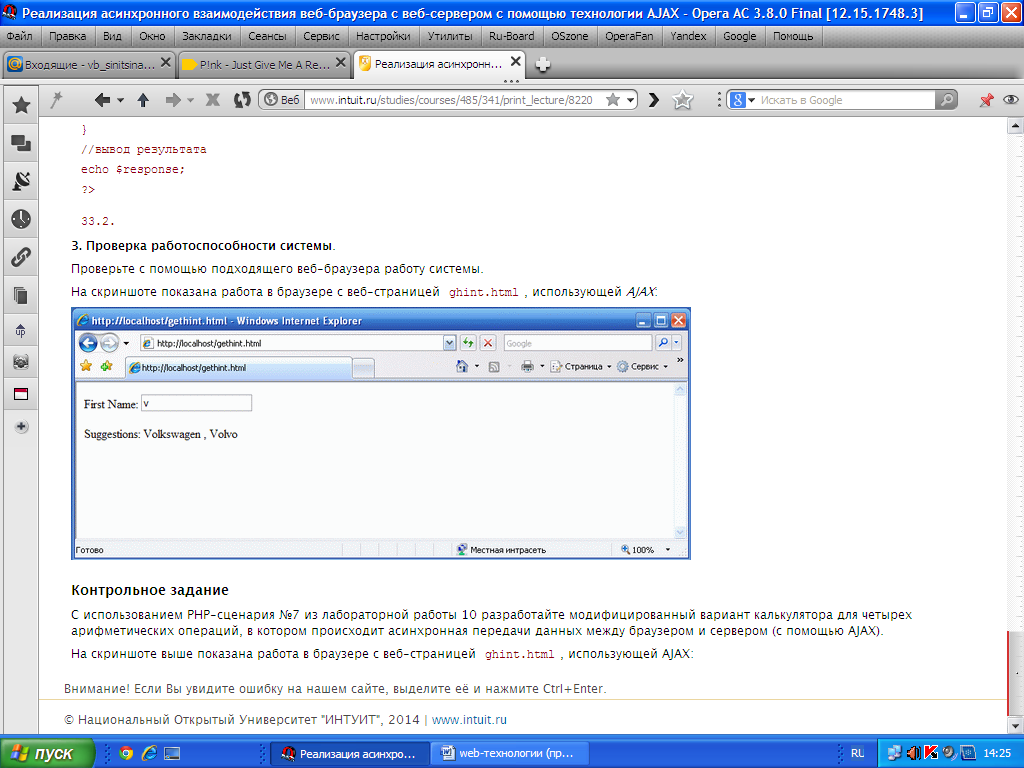


Рисунок 18 – Скриншот работы программы

**Контрольное задание**

С использованием PHP-сценария №7 из лабораторной работы 10 разработайте модифицированный вариант калькулятора для четырех арифметических операций, в котором происходит асинхронная передачи данных между браузером и сервером (с помощью AJAX).

На скриншоте выше показана работа в браузере с веб-страницей ghint.html, использующей AJAX.

**Лабораторная работа 18. «Создание мобильных веб-приложений при помощи ASP.NET»**

*Цель работы:* демонстрация некоторых возможностей разработки мобильных веб-приложений с помощью управляющих элементов ASP.NET Mobile.

В данной лабораторной работе демонстрируется применение управляющих элементов ASP.NET Mobile для создания веб-приложений для мобильных устройств.

.NET Mobile представляет собой расширение Microsoft ASP.NET и Microsoft .NET Framework. По-сути, .NET Mobile представляет собой набор серверных управляющих элементов для форм, ориентированных на использование в беспроводных мобильных устройствах. Эти элементы управления генерируют разный код для различных устройств на языках WML, HTML или Compact HTML (cHTML).

Управляющие элементы ASP .NET Mobile расширяют функции SDP (Smart Device Programmability) и платформу .NET Compact Framework. Они позволяют использовать возможности платформы .NET Compact Framework и среды разработки Visual Studio .NET для создания мобильных веб-приложений благодаря возможности доставки данных на разнообразные мобильные устройства при помощи технологии ASP.NET. Этот подход позволяет создать в среде Visual Studio .NET единое мобильное веб-приложение, автоматически производящее формирование данных для отображения на разнообразных устройствах: мобильных телефонах, пейджерах, смартфонах, Pocket PC и др. При этом интегрированная среда разработки позволяет создавать мобильные веб-приложения, просто перетаскивая управляющие элементы в дизайнер форм.

Система .NET Mobile не устанавливает никаких компонентов на клиентское устройство. Для адаптации форматирования под конкретные браузеры используется серверная логика. Она генерирует данные в форматах WML, HTML, и cHTML. Помимо адаптивной генерации веб-страниц, технология .NET Mobile предоставляет богатый набор средств индивидуализации и расширения, обеспечивая простой способ осуществления поддержки новых устройств. Кроме того, технология .NET Mobile позволяет разработчикам управлять представлением данных для конкретного устройства или класса устройств в одной программной модели и обеспечивает легкую реализацию поддержки новых устройств без переделки существующих веб-приложений.

Общая последовательность работы .NET Mobile выглядит следующим образом:

* Мобильное устройство. С него исходит запрос на загрузку веб-страницы.
* Интернет. Запрос передается через сеть соответствующему веб-серверу.
* Веб-сервер IIS получает запрос от мобильного устройства и передает его обработчику.
* .NET Framework выполняет обработку запроса.
* ASP.NET компилирует запрашиваемый устройством документ.
* .NET Mobile. Реализует элементы веб-страницы с учетом требований конкретного типа мобильного устройства.
* Веб-страница возвращается обратно клиентскому устройству.

Таблица 9 - Наиболее распространенными элементами управления форм для мобильных устройств являются следующие:

|  |  |
| --- | --- |
| Название | Функция |
| Command | Выполнение действия |
| Form | Контейнер для элементов управления |
| Image | Изображение |
| Label | Текст |
| Link | Гиперссылка |
| List | Список |
| MobilePage | Контейнер для элементов управления |
| ObjectList | Список объектов |
| Panel | Контайнер для других элементов управления |
| SelectionList | Список с выбором |
| StyleSheet | Стили для других элементов управления |
| TextBox | Поле для ввода однострочного текста |
| TextView | Поле для ввода многострочного текста |

***Порядок выполнения лабораторной работы***

Для работы с примерами, приводимыми в данной лабораторной работе помимо среды разработки Visual Studio .NET потребуется установка компонентов .NET Mobile и WAP эмулятора.

В качестве WAP эмулятора был использован Series 40 5th Edition SDK (на основе Nokia 6267 ). Для установки эмулятора необходимо скачать с сайта http://www.forum.nokia.com/ (требуется предварительная регистрация) и установить программу Nokia Mobile Internet Toolkit и сам эмулятор Nokia Series 40 5th Edition SDK.

Общая последовательность разработки веб-приложения с использованием .NET Mobile выглядит следующим образом:

* Создание проекта веб-приложения ASP.NET.
* Подключение интерфейса System.Mobile.UI.
* Вставка в веб-страницу управляющих элементов .NET Mobile.

Рассмотрим пример создания простейшей веб-странице, отображающей на экране мобильного устройства сообщение " Hello world!".

Создайте новый проект типа ASP.NET Web Application.

Активируйте с помощью меню окно "Solution Explorer". Из списка файлов проекта удалите файлы Default.aspx.cs и Default.aspx.designer.cs.

Страница Default.aspx должна содержать следующий код:

<%@ Page Inherits= "System.Web.UI.MobileControls.MobilePage " Language= "C# "%>

<%@ Register TagPrefix= "mobile" Namespace= "System.Web.UI.MobileControls "

Assembly= "System.Web.Mobile " %>

<mobile:Form runat="server">

<mobile:Label runat="server">Hello world!</mobile:Label>

</mobile:Form>

Директива в первой строке сообщает обработчику ASP о необходимости обработки данной страницы как предназначенной для мобильных устройств.

Директива во второй строке регистрирует префикс "mobile" для обращения к элементами управления для мобильных устройств.

Тэг <mobile:Form> указывает серверу на необходимость вставки формы.

Тэг <mobile: Label> указывает серверу на необходимость вставки метки с текстом " Hello world!".

Откомпилируйте проект.

Создайте новый виртуальный каталог, связанный с директорией текущего проекта. Добавьте в список документов по умолчанию страницу Default.aspx.

Запустите эмулятор Nokia Series 40 5th Edition SDK и программу Nokia Mobile Internet Toolkit. В программе Nokia Mobile Internet Toolkit укажите полный URL созданной с помощью ASP.NET страницы Default.aspx.

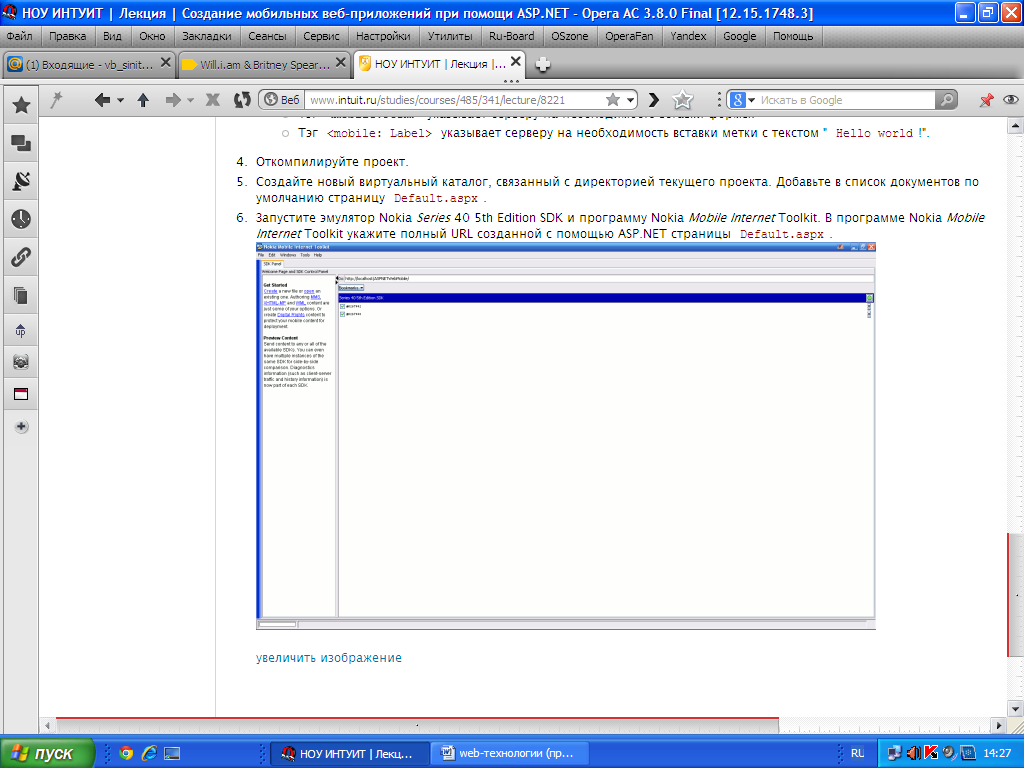


Рисунок 19 - Эмулятор Nokia Series 40 5th Edition SDK

После обработки данной страницы для мобильного устройства, поддерживающего протокол WAP, будет сгенерирована страница, вид которой в эмуляторе приведен ниже:



Рисунок 20 – Страница мобильного устройства

В эмуляторе выберите раздел меню "Tools > Diagnostics". В появившемся окне диагностики можно будет увидеть следующий код страницы, загруженной в эмуляторе:

<?xml version='1.0'?>

<!DOCTYPE wml PUBLIC '-//WAPFORUM//DTD WML 1.1//EN'

'http://www.wapforum.org/DTD/wml\_1.1.xml'>

<wml>

<head>

<meta http-equiv="Cache-Control" content="max-age=0" />

</head>

<card>

<p>Hello world!</p>

</card>

</wml>

Следующий пример демонстрирует использование простого элемента формы для ввода текста (в режиме пароля):

<%@ Page Inherits= "System.Web.UI.MobileControls.MobilePage " Language= "C# "%>

<%@ Register TagPrefix= "mobile " Namespace= "System.Web.UI.MobileControls "

Assembly= "System.Web.Mobile " %>

<script runat="server" language="C#">

string uname;

protected void AgeClick(Object sender, EventArgs e)

{

uname=text1.Text;

ActiveForm=form2;

}

protected void Form2\_Activate(Object sender, EventArgs e)

{

message.Text="Wellcome " + uname + "!";

}

</script>

<Mobile:Form id="form1" runat="server">

<Mobile:Label runat="server">Your name, please</Mobile:Label>

<Mobile:TextBox runat="server" id="text1" password="true"/>

<Mobile:Command runat="server" OnClick="AgeClick" Text="Submit" />

</Mobile:Form>

<Mobile:Form id="form2" runat="server" OnActivate="Form2\_Activate">

<Mobile:Label runat="server" id="message" />

</Mobile:Form>

Если выполнить как в предыдущем примере все шаги с 1 по 8, то после загрузки получившейся страницы в эмуляторе на экране отобразится первая форма:

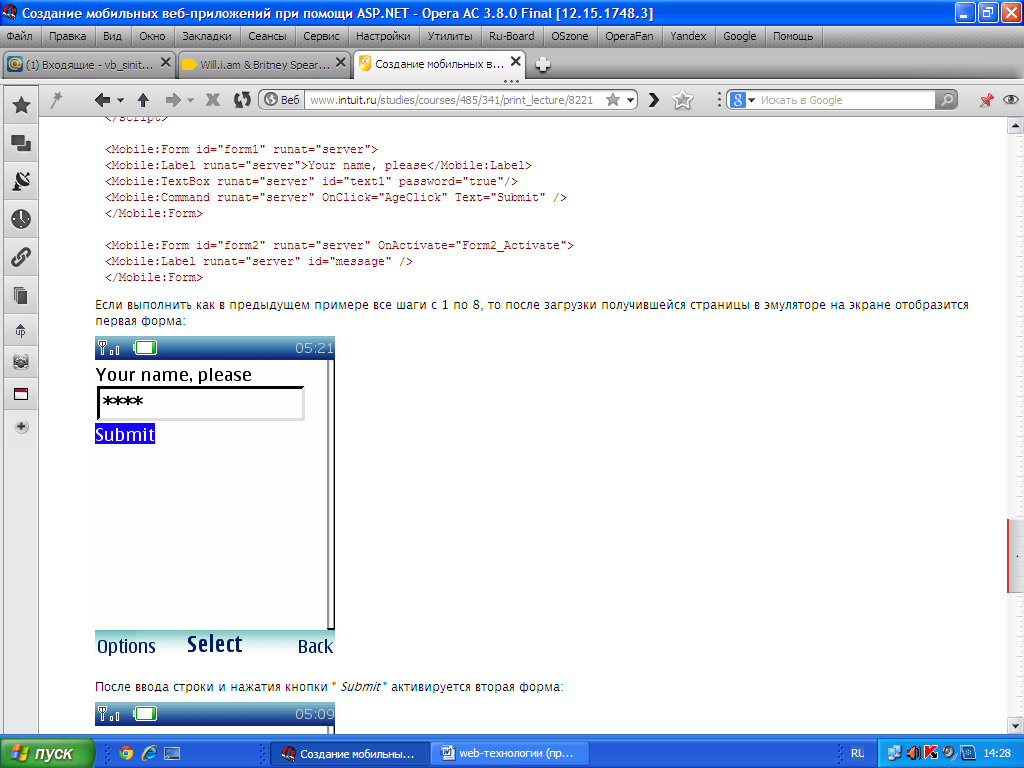


Рисунок 21 – Первая форма

После ввода строки и нажатия кнопки "Submit" активируется вторая форма:

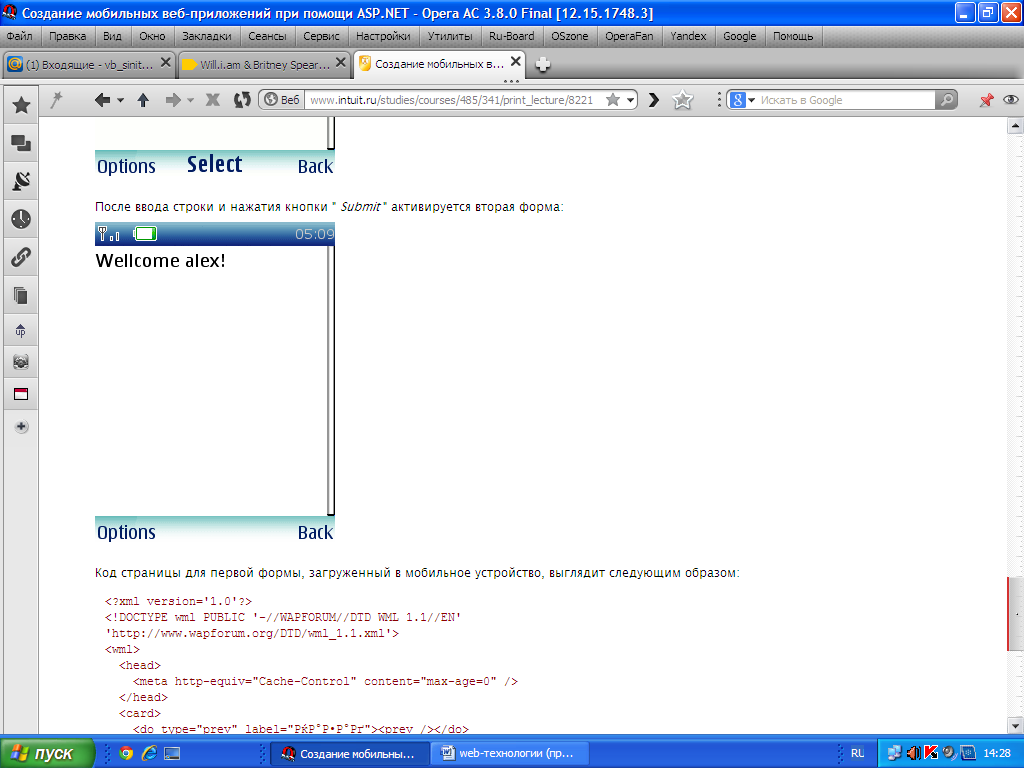


Рисунок 22 – Вторая форма

Код страницы для первой формы, загруженный в мобильное устройство, выглядит следующим образом:

<?xml version='1.0'?>

<!DOCTYPE wml PUBLIC '-//WAPFORUM//DTD WML 1.1//EN'

'http://www.wapforum.org/DTD/wml\_1.1.xml'>

<wml>

<head>

<meta http-equiv="Cache-Control" content="max-age=0" />

</head>

<card>

<do type="prev" label="РќР°Р•Р°Рґ"><prev /></do>

<p>Your name, please<br/>

<input name="mcsvprtws0" type="password" value="" />

<anchor title="Р'РїРµСЂРµРґ">Submit

<go href="Default.aspx?\_\_ufps=058985" method="post">

<postfield name="\_\_EVENTTARGET" value="ctl01" />

<postfield name="text1" value="$(mcsvprtws0)" />

</go>

</anchor>

</p>

</card>

</wml>

**Контрольное задание**

Разработайте с помощью MS Visual Studio .NET серверное веб-приложение, использующее управляющий ASP.NET Mobile элемент: <Mobile:Calendar> Добавьте для этого элемента обработчик события OnSelectionChanged, который должен отображать выбранный пользователем мобильного устройства день календаря с помощью элемента <Mobile:Label>.

Код обработчика события может выглядеть следующим образом:

<script runat="server">

protected void CalChanged(Object sender, EventArgs e)

{

label1.Text = "You selected: " + calendar1.SelectedDate;

}

</script>

Создайте необходимый виртуальный каталог на веб-сервере и проверьте работу приложения с помощью эмулятора мобильного устройства.