ЛАБОРАТОРНАЯ РАБОТА №7 “РАЗРАБОТКА ВЕБ ПРИЛОЖЕНИЯ, ИСПОЛЬЗУЮЩЕГО ТЕХНЛОГИЮ ASP.NET”

1. Теоретическая часть

Обзор ASP.NET Framework

ASP.NET — это технология создания веб-приложений и веб-служб от компании Майкрософт. Она является составной частью платформы Microsoft .NET и развитием более старой технологии Microsoft ASP. Кроме совместимости синтаксиса ASP.NET и ASP, новая платформа предоставляет новую модель программирования и инфраструктуру для более безопасных, масштабируемых и стабильных приложений. Можно легко дополнить существующее приложение ASP, последовательно добавив в него функциональные возможности ASP.NET.

ASP.NET является компилируемой средой на основе технологии .NET, в которой доступно создание приложений на любом совместимом с .NET языке, включая Visual Basic .NET, C# и JScript .NET. Кроме того, среда .NET Framework полностью доступна для любого приложения ASP.NET. Разработчики могут использовать все преимущества этих технологий, включающих в себя управляемую общеязыковую среду выполнения, строгую типизацию, наследование и т.д.

ASP.NET - часть глобальной платформы .NET. Эта платформа - часть новой стратегии Microsoft и соответствует всем современным стандартам разработки как распределенных систем, так и настольных приложений.

.NET Framework предоставляет интерфейс приложениям, сама непосредственно взаимодействуя с операционной системой. Выше лежит интерфейс ASP.NET приложений, на котором в свою очередь базируются вебформы (ASP.NET страницы) и веб-сервисы. Интерфейс .NET Framework позволяет стандартизировать обращение к системным вызовам и предоставляет среду для более быстрой и удобной разработки. В новую платформу встроены такие необходимые возможности, как контроль версий и важная для сетевых решений повышенная безопасность. Среда выполнения кода включает в себя сборщик мусора и набор библиотек, готовых к использованию.

Код для .NET Framework компилируется в общий промежуточный язык (Intermediate Language-IL). В случае ASP.NET код компилируется при первом обращении к странице и сохраняется для последующих вызовов. При выполнении оболочка компилирует промежуточный код в бинарный и выполняет его.

Кэширование готового бинарного кода позволяет улучшить эффективность. Intermediate Language позволяет создавать системы на любом удобном языке. И независимо от того, используется C#, VB.NET, JScript.NET или Perl.NET, в результате получается код, готовый к выполнению.

.NET Framework предоставляет и общий интерфейс обращения к базам данных - ADO.NET. Он тесно интегрирован с XML, что дает дополнительные преимущества при разработке распределенных приложений.

Обзор вебформ

Вебформы - это новая для ASP технология. Вебформы введены для удобства разработки приложений. Они позволяют создавать компоненты интерфейса пользователя для многократного использования, что упрощает работу разработчика. Вебконтролы инкапсулируют html код, что позволяет писать код с более четкой логической структурой. Наконец, контролы упрощают создание средств WYSIWYG разработки. Из всех средств, тестировавшихся нами ранее для ASP, только Dreamweaver UltraDev позволял это в приемлемой мере, и то, по сравнению со средствами RAD разработки, его возможности по быстрому созданию качественного кода не впечатляли. Веб формы – это обычные страницы. Помимо динамического содержания, создаваемого вашим кодом, вы можете включать в них вебконтролы. Итак, простейшая вебформа:

simple_form.aspx

```
<html>
<head>
</head>
<script language="C#" runat="server">
void SubmitBtn_Click(Object sender, EventArgs e) {
Message.Text = "Hello, world!";
}
</script>
<body>
<center>
<form method="post" runat="server">
<asp:button type=submit text="hello" OnClick="SubmitBtn_Click"
runat="server"/><br>
<asp:label id="Message" runat="server"/>
</form>
</center>
</body>
    </html>
```

Заметьте, что у формы стоит свойство runat="server". При запросе страницы, сервер обрабатывает такие контролы и выдает клиенту соответствующий html код. Здесь же был написан простейший обработчик

событий для вебформ. При нажатии на кнопку, когда форма отправляется на сервер, ASP.NET выполняет наш метод SubmitBtn_Click. Это задается в свойстве OnClick кнопки submit. Наш же метод присваивает полю Text созданного нами контрола Message текст.

48 контролов поставляется с ASP.NET, они включают в себя различные компоненты от календаря до валидаторов. Кроме того, вы всегда можете написать свои для повторного использования, а также использовать созданные другими.

Наиболее часто вы, скорее всего, будете применять валидаторы и котролы, связанные с отображением данных. DataGrid позволит вам быстро вывести содержание выборки пользователю, в то время как DataList и Repeater позволят вам сделать это любым способом через шаблоны (поддерживаются шаблоны заголовков, футеров, самого куска данных и разделителя). Покажем также пример использования валидатора.

form_validation.aspx

```
<html>
<head>
</head>
<body>
<form method="post" runat="server">
<asp:textbox id="Text" runat="server"/><br>
<asp:RequiredFieldValidator ControlToValidate="Text"
Display="Dynamic" errormessage="You must enter text in control!"
runat=server/><br>
<asp:button type=submit text="submit" runat="server"/>
</form>
</body>
</html>
```

В данном примере имеется одно текстовое поле и RequiredFieldValidator. Это самый простой из валидаторов, он проверяет, имеются ли в заданном ему поле (в данном случае оно называется Text) какие-нибудь данные. Если нет, и пользователь нажмет submit, то при проверке перед отправкой, скрипт выведет сообщение об ошибке заполнения (вы сами указываете это сообщение) и форма не будет отправлена. Если же у пользователя старый броузер, то проверка будет произведена на сервере. В вашем коде вы можете проверить правильность заполнения всех полей с помощью поля Page.IsValid и вывести суммарное сообщение об ошибках с помощью ValidationSummary.

Упомянем о замечательной возможности - разделение кода и представления, что позволит разделить процессы разработки, упростить локализацию приложения и полностью использовать объектно-

ориентированный подход. В данном случае нашу простейшую вебформу можно было бы переписать так:

form.aspx

```
<%@ Page Inherits="SimpleCode" Src="form.cs" %>
<html>
<head>
</head>
<body>
<form method="post" runat="server">
<asp:label id="text" runat="server"/><br>
<asp:button  type=submit  text="submit"  OnClick="SubmitBtn_Click"
runat="server"/><br>
</form>
</body>
</html>
```

form.cs

```
using System;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
public class SimpleCode : Page {
public Label text;
public void SubmitBtn_Click(Object sender, EventArgs e) {
text.Text = "text";
}
}
```

Структура ASPX-файла

ASPX-файл является, по сути, обычным HTML, в котором можно использовать специальные теги. Дополнитеьные теги делатся на две категории: служебные, задающие параметры страницы и позволяющие внедрять код, и теги контролов, которые представляют собой новые интерфейсные конструкции, настройка которых инкапсулирована в тег.

Первому типу принадлежат серверные директивы, теги <%= %>, <% %>,<%# %>, <script runat="server">...</script>. Сначала рассмотрим директивы. Их синтаксис имеет вид

```
<%@                    Directive                    %>
```

Базовой директивой является @ Page, которая описывает основные параметры страницы, такие как файл с исходным текстом, язык кода, параметры трассировки. Также используются директивы @ Import, @ Assembly, @ OutputCache и другие...

Теги <%= %> используются для выставки вычисляемых значений. Например, конструкция

```
<%= System.DateTime.Now %>
```

после компиляции будет вместо себя подставлять текущее время.

Теги <% %> позволяют вставлять код в страницу - то, что в основном используется в ASP и JSP. Например,

```
<%if (User.Identity.Name == "Admin")
{
%>
<a href="adminpage.htm">Перейти к странице администрирования<a>
<%
}
%>
```

отображает ссылку только для пользователя Admin.

Конструкция <%= %> является аналогом <%= %> для компонент с привязкой к данным. Они подставляют вместо себя значение, зависящее от текущего контекста данных. Приведем пример, который, однако, здесь подробно разобран не будет - привязка к данным будет обсуждена в отдельной главе.

```
<asp:Repeater id="lstData" runat="server" >
<ItemTemplate>
<%# (Container.DataItem as DateTime).ToShortDateString() %>
</ItemTemplate>
</asp:Repeater>
```

Также в страницу можно вставлять дополнительные методы с помощью тега <script runat="server">:

```
<script language="C#" runat="server" >
void SubmitBtn_Click( object sender, EventArgs e )
{
txtMessage.Text = "Hello, world";
}
</script>
```

Параметр language тега script задает язык, на котором он написан.

Теперь рассмотрим теги контролов - они позволяют вставлять в страницу различные элементы управления. Многие из вас сразу вспомнят про стандартные элементы управления вроде кнопок или полей ввода. На самом деле контролы ASP.NET могут представлять не только стандартные элементы, кодируемые одним HTML-тегом, но и довольно сложные DHTML-конструкции. При этом если при написании таких конструкций в обычном HTML вы получаете громоздкий и нерасширяемый код, то при использовании ASP.NET все это для вас инкапсулировано в одном теге. При этом код, который будет

подставляться генерируется на сервере, что позволяет изменять его в зависимости, например, от типа броузера.

Теги серверных контролов допустимы только внутри серверных форм - конструкций вида <form runat="server"> ... </form>. Сами теги контролов пишутся в XML-стиле - с обязательным закрыванием и пространствами имен. Общий вид такого тега:

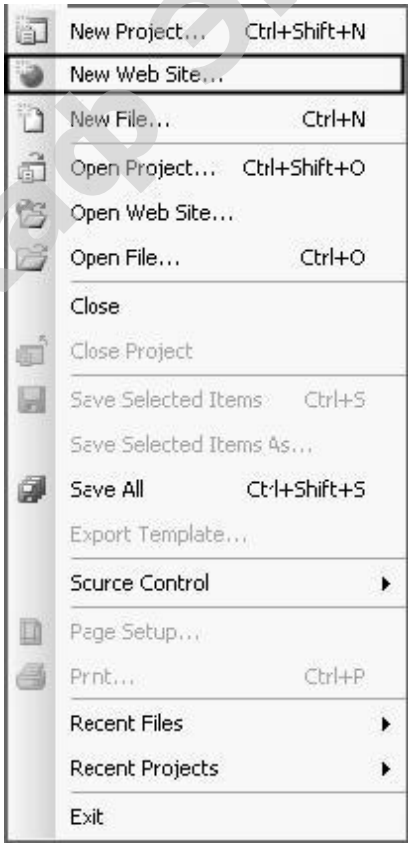
```
<пространство_имен:имя_тега runat="server" > ...
</пространство_имен:имя_тега >
```

Пространства имен используются для того, чтобы случайно не совпали имена тегов. Стандартные элементы управления от Microsoft находятся в пространстве имен asp: <asp:Button />, <asp:TextBox /> и т.п.

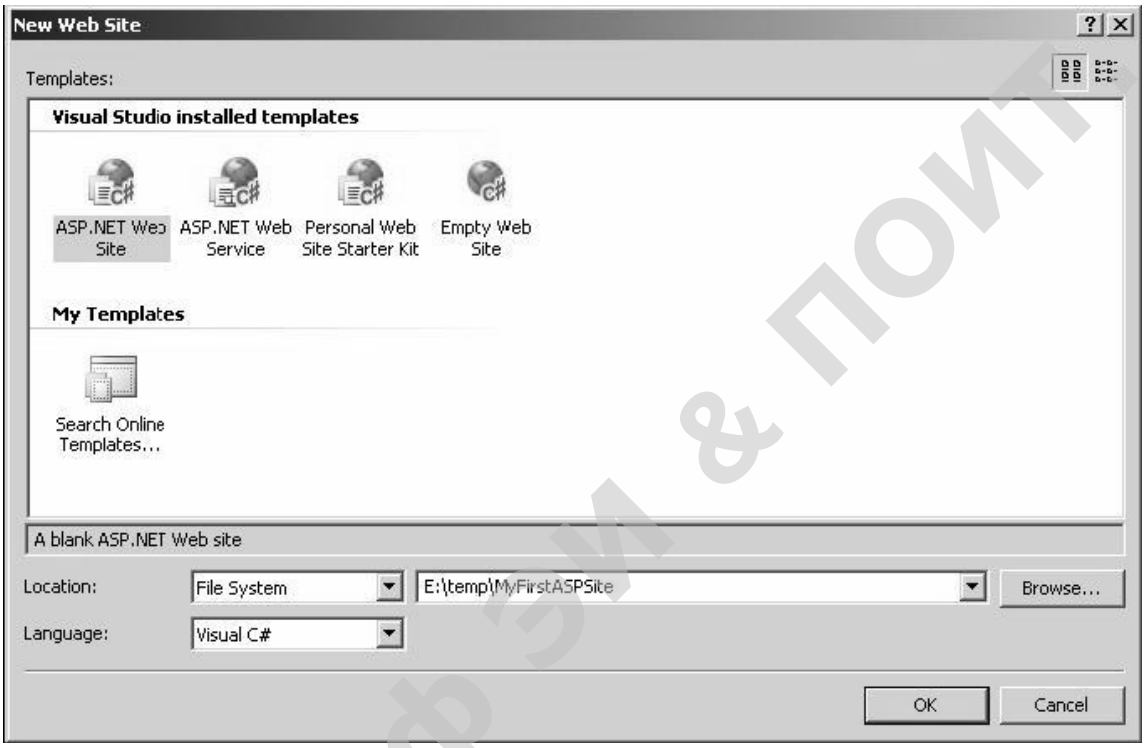
2. Практическая часть

Цель лабораторной работы – приобрести навыки разработки веб приложений ASP.NET и на примере данной лабораторной работы ознакомиться с основными принципами разработки таких приложений для платформы .NET.

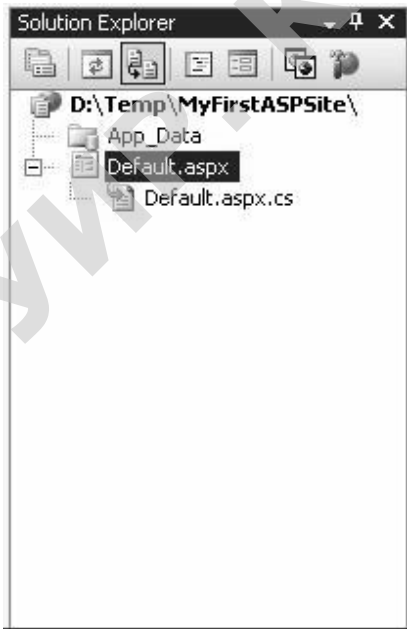
Приступим к созданию нашего первого приложения на платформе ASP.NET. Запустим Visual Studio и выберем в меню File пункт New Web Site:



В появившемся диалоге выберем тип проекта ASP.NET Web Site.В поле Location выбираем File System, что означает что наш сайт будет храниться на жестком диске компьютера(если на вашей машине установлен сервер IIS, то в качестве месторасположения создаваемого сайта можно указать его). В качестве языка программирования в поле Language выбираем C#.



Visual Studio создаст проект по умолчанию. Его структура будет видна в окне Solution Explorer:



Здесь можно увидеть два файла:

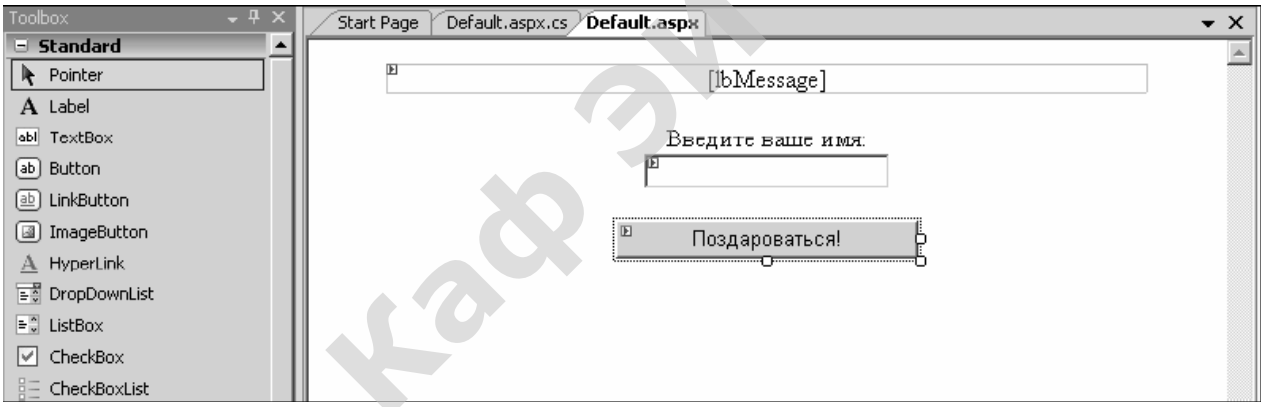
Default.aspx Экранная форма веб страницы

Default.aspx.cs Исходный код формы

По сути, приложение создано. Но оно абсолютно ничего не делает. Добавим нашему приложению простейший функционал. Для этого добавим на нашу форму поле ввода, метку и кнопку. Чтобы это сделать, откроем панель Toolbox.

Сначала добавим поле ввода. Для этого перетащим элемент TextBox с панели на нашу страницу. В панели свойств установим для нашего поля ввода идентификатор txtName.

Таким же способом добавим на страницу метку Label)\ и кнопку Button. Присвоим метке ID=lblMessage, Text - пустую строку, а Button - btnHello и "Поздароваться!" соответственно. После этого наша страница примет следующий вид:



Теперь необходимо добавить некоторую логику. Наша странца будет здороваться с пользователем по имени, после того как он введет имя в поле ввода и нажмет кнопку. Для этого нужно написать обработчик события

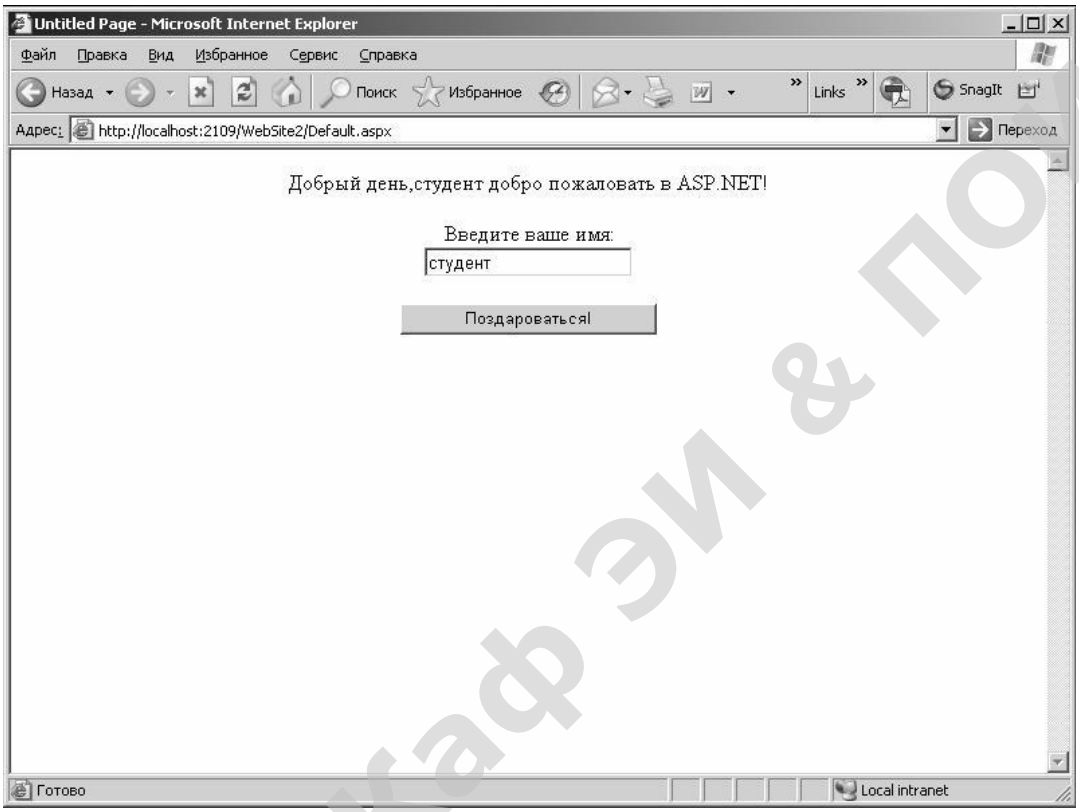
"пользователь нажал на кнопку" и вывести в метке соответствующий текст. Для создания обработчика дважды щелкните мышкой на кнопке. Visual Studio откроет файл с именем WebForm1.aspx.cs. К каждой форме невидимо привязываются файлы с .aspx.cs и .aspx.resx, но их не показывают в проекте, чтобы не нагружать его. В файле .cs содержится код страницы. После того, как вы открыли его двойным щелчком на кнопке, в него добавился соответствующий обработчик. Введите в него следующий код:

```
protected void btnHello_Click(object sender, EventArgs e)
{
```

```
lbMessage.Text = "Добрый день," + txtName.Text + " добро пожаловать
в ASP.NET!";
    }
```

Теперь нужно скомпилировать проект. Это делается комбинацией клавиш Ctrl+Shift+B или команой меню Build -> Build WebSite.

После запуска приложения перед нами откроется окно веб-браузера:



Индивидуальные задания

1. Разработать веб-страницу календарь, которая отображала бы текущие дату и (или) время по выбору пользователя.
2. Разработать веб-страницу с функциями калькулятора.
3. Разработать веб-страницу с функций переворота строки.
4. Разработать веб-страницу, считающую количество символов, вводимых пользователем в два различных поля для ввода текста и выводящую сумму на экран.

5. Разработать веб-страницу, реализующую простейшую доску объявлений с фиксированным числом записей (не менее 7)
6. Разработать веб-страницу, реализующую простейший учет продаж компьютеров. Данные хранятся в форме.
7. Разработать веб-страницу, реализующую конвертер валют.

8. Разработать веб-страницу, реализующую функцию подсчета годовых процентов по акциям.
9. Разработать веб-страницу, реализующую подсчет скидок по товарам.
10. Разработать веб-страницу, реализующую расчета состояния автомобиля в зависимости от пробега.

Вопросы для самопроверки

1. Что такое платформа ASP.NET?
2. Какие функции выполняет платформа ASP.NET?
3. Как происходит компиляция кода в ASP.NET?
4. Что такое веб-форма?
5. Для чего предназначены веб-формы и какие функции они выполняют?
6. Какие основные элементы управления используются на веб-формах?
7. Какую структуру имеет ASPX-файл?
8. Для чего предназначена конструкция <%= %>?
9. На каких языках программирования возможно написание проиложений для ASP.NET платформы?
10. Назовите основные файлы, содержащиеся в создаваемом проекте ASP.NET?