Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра информационных систем и технологий

ПРОГРАММИРОВАНИЕ ИНТЕРНЕТ-ИЗДАНИЙ

**Методические указания к выполнению контрольной работы
для студентов специальности 1-40 01 02-03
«Информационные системы и технологии
(издательско-полиграфический комплекс)»
заочной формы обучения**

Минск 2014

УДК 004.43:004.735.5(072)
ББК 32.973.202.я7
П78

Рассмотрены и рекомендованы к изданию редакционно-издательским советом Белорусского государственного технологического университета

С о с т а в и т е л ь
В. В. Смелов

Р е ц е н з е н т
кандидат технических наук, доцент,
заведующий кафедрой систем обработки информации
и полиграфического оборудования Белорусского
государственного технологического университета
М. С. Шмаков

По тематическому плану изданий учебно-методической литературы университета на 2014 год. Поз. 180.

Предназначены для студентов специальности 1-40 01 02-03 «Информационные системы и технологии (издательско-полиграфический комплекс)» заочной формы обучения

ПРЕДИСЛОВИЕ

Предлагаемое пособие предназначено для студентов специальности «Информационные системы и технологии (издательско-поли графический комплекс)» заочной формы обучения, изучающих дисциплину «Программирование интернет-изданий».

Учебный план студентов заочной формы обучения предусматривает контрольную работу. Выполнение и защита контрольной работы является допуском к сдаче экзамена по дисциплине.

Контрольная включает десять практических работ, выполняемых студентом самостоятельно, каждая из них состоит из нескольких заданий. Задания имеют сквозную нумерацию. При выполнении практических работ следует придерживаться последовательности, предложенной в указаниях, так как формулировка некоторых заданий может опираться на результаты предыдущих. Последнее задание каждой практической работы содержит перечень контрольных вопросов, на которые студент должен будет ответить при защите работы.

Методические указания состоят из десяти разделов, соответствующих практическим работам контрольной. Каждый раздел содержит тезисное описание теоретического материала, необходимого для выполнения заданий. Описание сопровождается ссылками на литературу, рекомендуемую студенту для самостоятельного изучения.

Материал пособия предполагает наличие базовых знаний студента в области программирования и теории операционных систем, а также навыков программирования на языке C# в объеме [1], а для дальнейшего изучения, настоятельно рекомендуется изучить литературу [2]. Кроме того, при выполнении практических работ потребуются знание языков JavaScript [3], HTML [4] и XML [5].

Практически все задания контрольной работы сводятся к разработке web-приложений с помощью технологии ASP.NET. Для знакомства с ASP.NET рекомендуется изучение [6]. В этой книге студент найдет ответы на большинство контрольных вопросов.

Некоторые задания требуют настройки сервера приложений Microsoft Internet Information Server 7. Все необходимые для этого сведения можно найти в источнике [7].

Для выполнения практических работ студенту потребуется компьютер с операционной системой Windows версии 7 или выше. Кроме того, потребуется установка платформы .NET Framework 3.5 и интегрированной среды разработки приложений Visual Studio версии 2010 или выше.

Для самостоятельного ознакомления с другими технологиями разработки web-приложений, рекомендуются источники [8], [9].

Основной задачей данного пособия является обучение студентов навыкам разработки простейших web-приложений с помощью технологии ASP.NET. Для этого предлагается выполнить 10 практических работ.

Первая работа посвящена установке, настройке и проверке работоспособности сервера приложений IIS 7, который по своей сути является контейнером ASP.NET-приложений. За небольшим исключением, все ASP.NET-приложения выполняются под управлением этого сервера.

Во второй работе студент разработает и отладит простейшее web-приложение уровня «Hello World», опубликует его на сервере IIS 7 и проверит его работоспособность.

Третья работа познакомит студента со структурой и моделью событий ASP.NET-приложений шаблона Web Forms.

Для отображения и ввода данных на web-страницах ASP.NET-приложения используются специальные программные компоненты ASP.NET, называемые серверными элементами управления. Практические работы 4–8 предназначены для знакомства с различными типами серверных элементов управления ASP.NET.

В практической работе 9 рассматривается особый тип серверных элементов управления, объединенных под названием AJAX и предназначенных для выполнения асинхронных запросов к серверу.

Последняя, десятая практическая работа посвящена технологии кэширования web-страниц в ASP.NET-приложении. Кэширование применяется для повышения производительности работы web-приложения.

Следует еще раз отметить, что задачей данного пособия является только первоначальное знакомство с технологией ASP.NET. Состав этой технологии значительно шире. Для более продуктивного знакомства с ASP.NET рекомендуется прочитать издание [6]. Более того, программирование в Интернет не ограничивается разработкой web-приложений. Для знакомства с другой современной технологией, часто используемой совместно с ASP.NET и называемой «Windows Communication Foundation», рекомендуется источник [10].

ВВЕДЕНИЕ

Развитие глобальной сети Интернет дало жизнь новому классу компьютерных систем, предоставляющих пользователям доступ к информационным ресурсам, называемых электронными изданиями. Это, в свою очередь, привело к появлению технологий, позволяющих разрабатывать системы такого класса. Наибольшее распространение получили технологии, объединенные под общим названием web-программирование. Под web-программированием подразумевается процесс разработки web-приложений, назначение которых сводится к приему HTTP-запросов и формированию HTTP-ответов.

Технологии разработки web-приложений условно можно разбить на четыре группы: условно называемые PHP-технологии, условно называемые Java-технологии, технология ASP.NET и все остальные

Исторически сложилось так, что первой широкое распространение получили технологии, в основе которых лежит скриптовый язык программирования PHP, разработанный группой энтузиастов в рамках проекта с открытым кодом (open-source software). До недавнего времени количество сайтов (форма организации интерфейса web-приложений), разработанных с помощью этой технологии было подавляющим. С принципами разработки PHP-приложений можно ознакомиться в литературе [9].

Технологии разработки web-приложений, основанные на языке программирования Java (компании Sun Microsystems, а в последствии Oracle), появились в 1999 году после опубликования набора спецификаций под названием «Java Platform, Enterprise Edition» (Java EE). В настоящей момент технологии поддерживаются корпорацией Oracle. Для первоначального знакомства с этими технологиями рекомендуется издание [8].

ASP.NET – технология разработки web-приложений компании Microsoft, является составной частью платформы .NET и развитием предшествующей ей технологии ASP. На текущий момент последней версией является ASP.NET 4.5.

Современная тенденция такова, что общая доля сайтов, разработанных с помощью Java-технологий и ASP.NET, быстро увеличивается и на сегодняшний день составляет более половины.

Практическая работа № 1
УСТАНОВКА И НАСТРОЙКА СЕРВЕРА IIS

1.1. Теоретические сведения

1.1.1. Назначение и применение IIS

IIS (Internet Information Services) – распространяемый вместе с операционными системами семейства Windows NT набор серверов для Интернет-служб компании Microsoft. IIS поддерживает сетевые протоколы HTTP, HTTPS, FTP, POP3, SMTP, NNTP. Последняя версия IIS 7 поставляется вместе операционными системами Windows 7/2008/8/2012.

Для того чтобы выяснить, установлен IIS на компьютере или нет, необходимо просмотреть окно «Компоненты Windows», которое может быть получено через раздел «Программы» пользовательского интерфейса «Панель управления» операционной системы Windows (рис. 1.1).

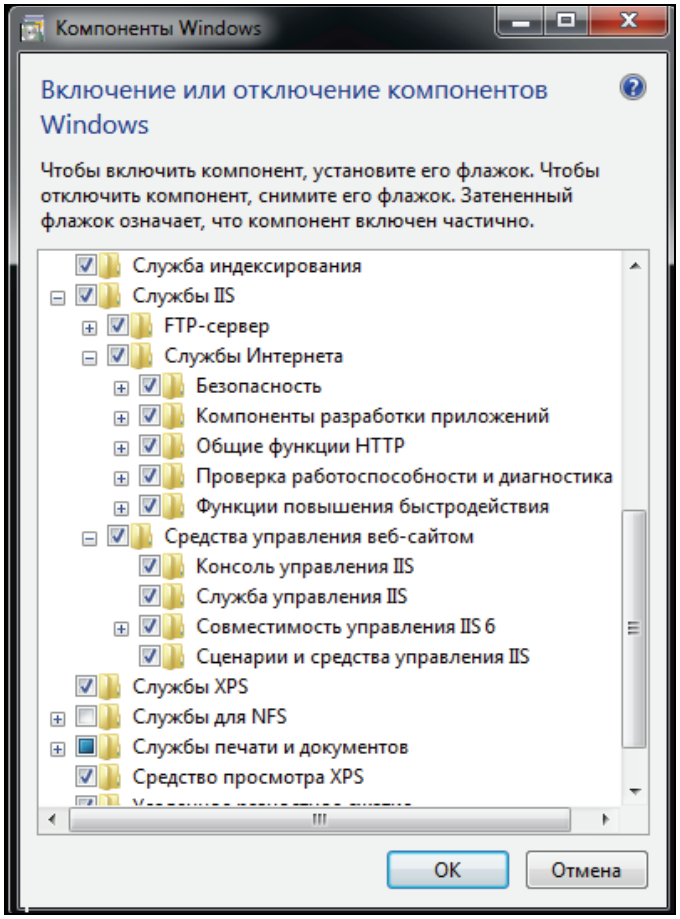

Рис. 1.1. Окно «Компоненты Windows» с информацией об установке IIS

Основным компонентом IIS является web-сервер, позволяющий размещать сайты, доступные в сети Интернет. Работа сайтов обеспечивается web-приложениями, которые должны быть размещены (опубликованы) на сервере IIS.

Создание web-приложений, работающих в рамках IIS, возможно с помощью нескольких технологий: ISAPI (низкоуровневая технология, предоставляющая доступ ко всем возможностям IIS), CGI и FastCGI (клиент-серверные протоколы взаимодействия web-сервера и web-приложения), ASP и ASP.NET (современные технологии создания динамических страниц). Расширив узел «Компоненты разработки приложений» окна «Компоненты Windows» (рис. 1.1) можно получить перечень технологий, поддерживаемых IIS (рис. 1.2).

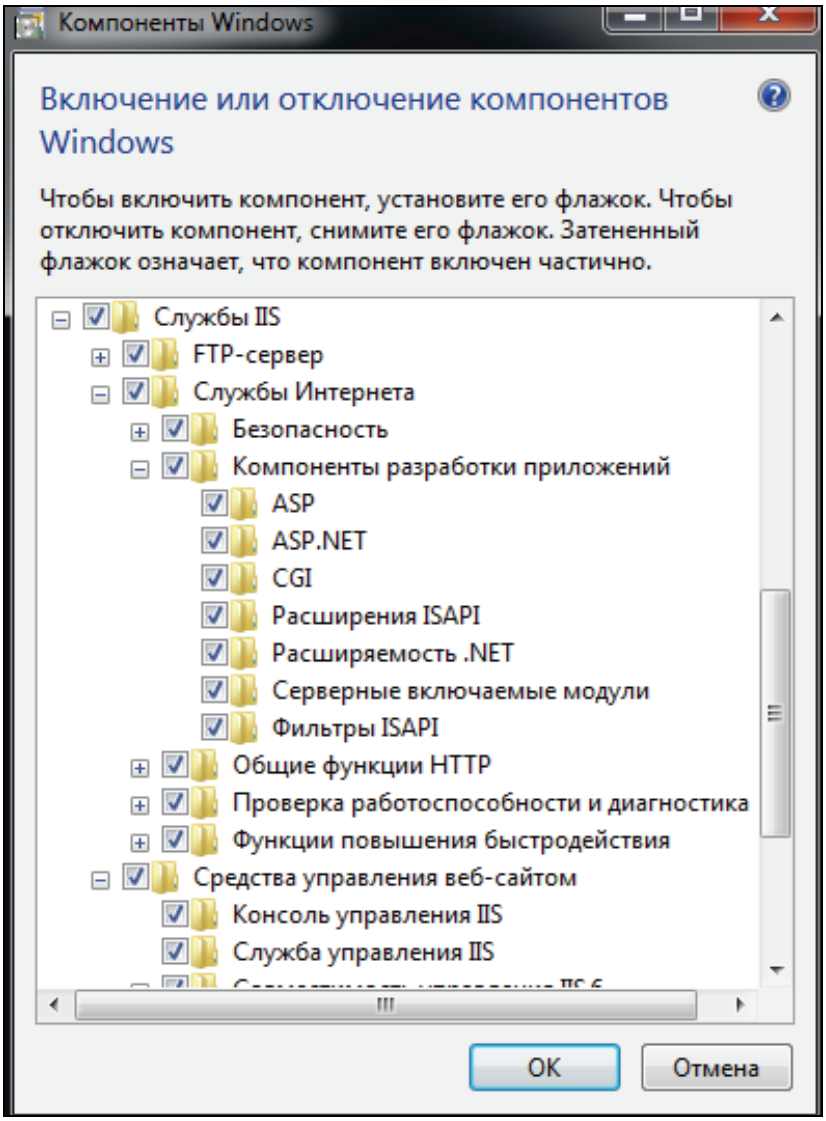

Рис. 1.2. Содержимое узла «Компоненты разработки приложений»

Составной частью IIS является приложение IIS Manager, предназначенное для управления IIS. IIS Manager может быть запущен через командую строку (рис. 1.3). На рис. 1.4 представлено стартовое окно IIS Manager. Приложение позволяет подключаться к локальному или удаленным серверам IIS, добавлять, настраивать и удалять сайты, импортировать или экспортировать приложения и т. п. Более подробно ознакомиться с возможностями приложения IIS Manager можно в источнике [7].

Рис. 1.3. Запуск приложения IIS Manager

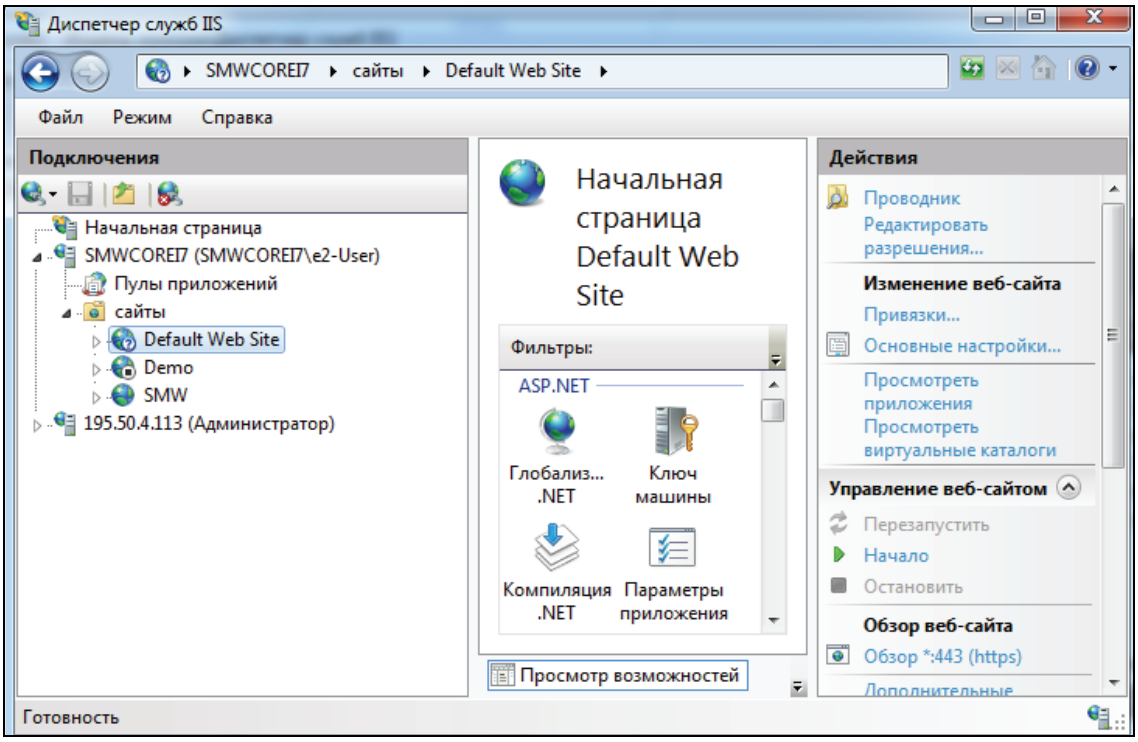

Рис. 1.4. Стартовое окно приложения IIS Manager

1.1.2. Установка и настройка IIS

Установка IIS может быть выполнена как в процессе инсталляции операционной системы, так и после нее. В любом случае, все сводится к выбору пунктов меню в окне «Компоненты Windows» (рис. 1.1

и 1.2). В том случае, если установка и настройка IIS осуществляется уже в рамках установленной операционной системы Windows, доступ к этому окну возможен в следующей последовательности: Панель управления| Программы| Программы и компоненты| Включение и отключение компонентов Windows.

После того как в окне «Компоненты Windows» были выбраны и установлены необходимые программные компоненты сервера IIS, рекомендуется выполнить перезагрузку операционной системы.

После того как Windows с новыми программными компонентами будет загружен, следует проверить работоспособность приложения IIS Manager (см. п. 1.1) и выполнить HTTP-запрос к предустановленному (для проверки работоспособности IIS) сайту. Для этого следует в адресной строке браузера набрать http://localhost. В том случае, если установка IIS прошла успешно, в окне браузера отобразится web-страница, подобная той, что представлена на рис. 1.5.

Рис. 1.5. Проверка работоспособности IIS 7

1.1.3. Технология ASP.NET

Технология ASP.NET (Active Server Pages в среде .NET) является развитием ей предшествующей технологии ASP, часто называемой классической ASP. ASP.NET – технология создания web-приложений и web-сервисов. На данный момент последней версией этой технологии является ASP.NET 4.5.

ASP.NET является частью программной платформы Microsoft .NET и, кроме того, неразрывно связана с двумя другими продуктами Microsoft: web-сервером IIS и интегрированной средой разработки Visual Studio. IIS для ASP.NET-приложения выступает в качестве оболочки (среды выполнения), основным назначением которой является прием и отправка HTTP-сообщений. Visual Studio 2008/2010/2012 представляет собой инструментальное средство, позволяющее разрабатывать, выполнять отладку и публиковать на сервер приложения ASP.NET.

Технология ASP.NET предоставляет возможность разрабатывать два типа приложений: web-приложения и web-сервисы.

Web-приложения – приложения, обеспечивающие работу web-сайтов. В большинстве своем, работа web-приложений сводится к приему http-запросов, обработке (иногда достаточно сложной) и формированию http-ответов. При этом запросы, как правило, поступают от браузеров, которые отображают пользовательский интерфейс приложения с помощью html-страниц. Совокупность страниц, которые могут быть пересланы сервером браузеру для отображения, называют сайтом. Следует отметить, что для разработки сайтов Visual Studio представляет два вида web-приложений: Web Forms и MVC.

Web Forms – это вид web-приложения (ему соответствует свой шаблон проекта в Visual Studio), в основе которого лежит механизм ViewState, позволяющий сохранить состояние элементов управления web-страницы. Web Forms применяется для разработки не очень больших сайтов, не требующих большого числа страниц и глубокой их иерархии.

В основе создания MVC-приложения лежит концепция разделения web-приложения на компоненты, отвечающие за представление данных (model), отображение данных (view) и управление (controller). В состав ASP.NET входит набор библиотек (называемый ASP.NET MVC Framework), обеспечивающих поддержку разработки MVC-приложений. Как правило, приложения этого вида разрабатываются

в том случае, если необходимо разработать большой сайт, требующий высокого уровня масштабирования. На сегодняшний день последней действующей версией является ASP.NET MVC 4.0. Для знакомства с ASP.NET MVC рекомендуется источник [11].

Web-сервис – это особый тип приложения, представляющего собой удаленный (размещаемый на серверных компьютерах) программный компонент, для доступа к методам которого применяется протокол SOAP. Методы web-сервиса могут быть вызваны из .NET-приложения способом, практически не отличающимся от вызова метода обыкновенного (локального) программного объекта. В настоящее время компания Microsoft разработала новую технологию – Windows Communication Foundation (WCF), обобщающую понятие web-сервиса. Для ознакомления с возможностями WCF рекомендуется [10].

В рамках данного пособия рассматриваются web-приложения вида Web Forms.

1.2. Задания

Задание 1. Установка и настройка IIS:

1) добейтесь отображения на мониторе компьютера окна «Компоненты Windows» (рис. 1.1, 1.2);

2) позиционируйте дерево настроек в окне «Компоненты Windows» таким образом, чтобы были видны первоначальные настройки пункта «службы IIS»;

3) получите скриншот окна «Компоненты Windows» с первоначальными настройками служб IIS и включите его в отчет по контрольной работе;

4) отметьте все необходимые компоненты IIS и выполните их установку;

5) перезагрузите операционную систему.

Задание 2. Проверка работоспособности IIS:

1) с помощью командной строки запустите приложение IIS Manager (рис. 1.3) и убедитесь, что на мониторе отобразилось окно, аналогичное представленному на рис.1.4;

2) получите скриншот окна «Диспетчер служб IIS» и включите его в отчет по контрольной работе;

3) запустите браузер, в адресной строке введите http://localhost; убедитесь, что в браузере отобразилось окно, аналогичное представленному на рис. 1.5;

4) получите скриншот окна браузера со стартовой страницей IIS 7 и включите его в отчет по контрольной работе.

Задание 3. Контрольные вопросы:

1) поясните термины: «сервер», «клиент», «браузер», «HTTP», «HTML», «web-приложение»;

2) поясните термины: «IIS», «IIS Manager» «ASP.NET», «Web Forms», «MVC»;

3) поясните термины: «SOAP», «web-сервис», «WCF».

Практическая работа № 2
РАЗРАБОТКАПРОСТЕЙШЕГО ПРИЛОЖЕНИЯ WEB FORMS ASP.NET

2.1. Теоретические сведения

2.1.1. Создание приложения Web Forms ASP.NET

Для создания web-приложения Web Forms ASP.NET необходимо с помощью Visual Studio создать проект соответствующего типа. На рис. 2.1 представлено окно Visual Studio 2010 для создания web-проекта.

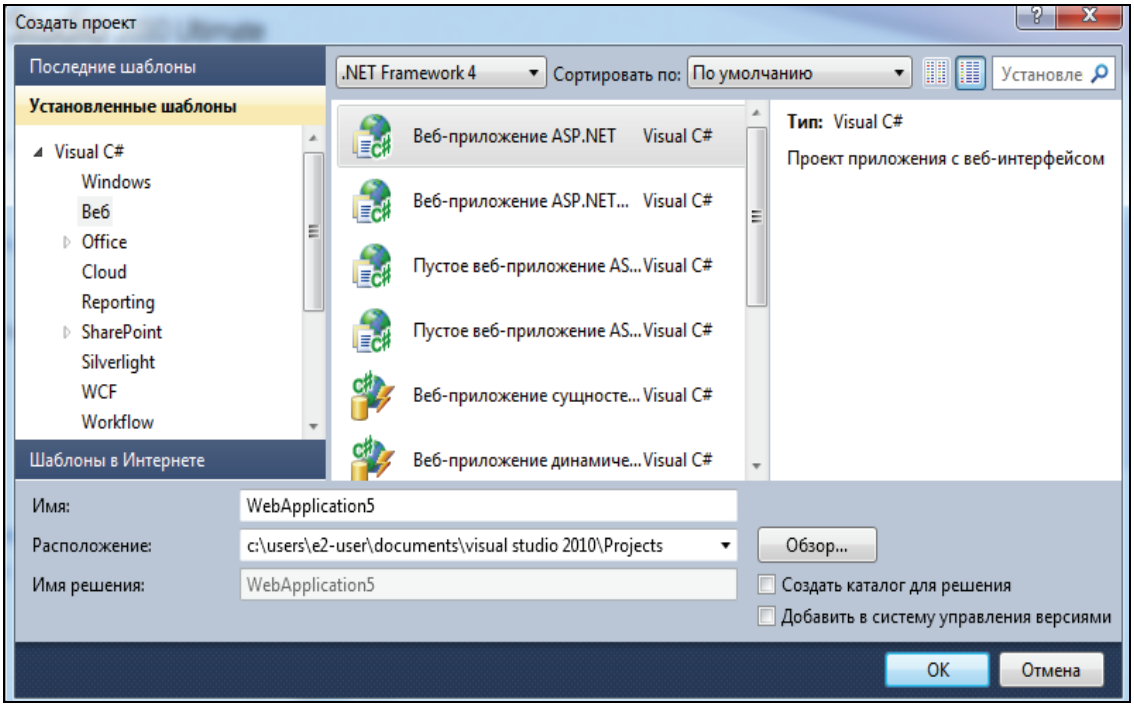

Рис. 2.1. Окно Visual Studio 2010 для создания web-проекта

Окно позволяет выбрать шаблон проекта в категории «Веб». Первый сверху шаблон «Веб-приложение ASP.NET» позволяет создать приложение Web Forms. При нажатии клавиши «ОК» создается шаблонное приложение, которое может быть выполнено в отладочном режиме.

На рис. 2.2 представлена страница, отображаемая шаблонным приложением в окне браузера при запуске в отладочном режиме.

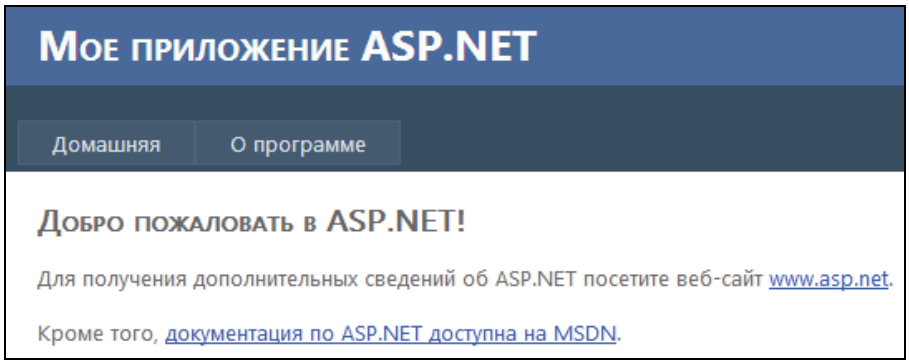

Рис. 2.2. Стартовая страница шаблонного приложения

При разработке и отладке ASP.NET-приложения разработчику требуется многократно запускать это приложение в режиме отладки. Для этого в рамках Visual Studio существует специальный компонент, называемый ASP.NET Development Server. Сервер стартует автоматически при запуске разрабатываемого web-приложения на выполнение с помощью Visual Studio. Находится сервер в области уведомлений (system tray) операционной системы. При двойном нажатии левой клавишей мыши по значку сервера, на мониторе отобразится окно, аналогичное представленному на рис. 2.3.

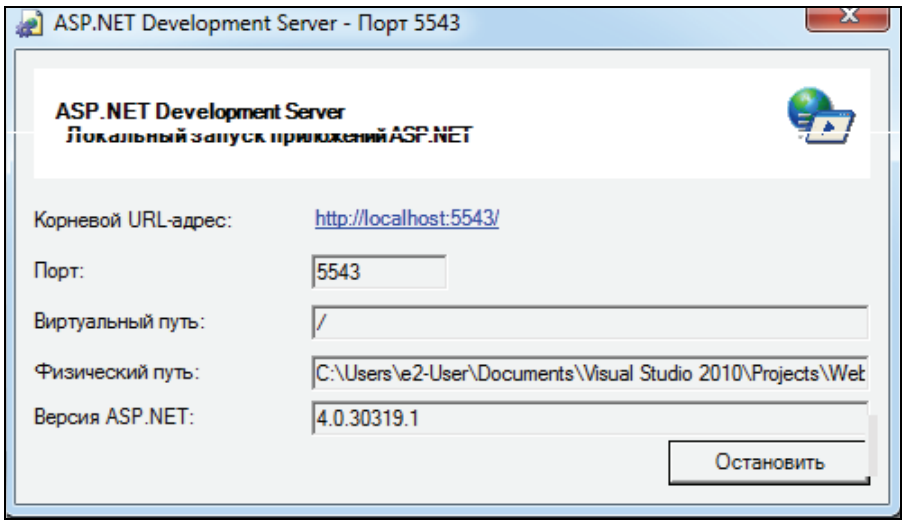

Рис. 2.3. Окно ASP.NET Development Server

2.1.2. Публикация web-приложения на IIS

После того как web-приложение разработано, оно может быть опубликовано на сервере IIS. Процесс публикации заключается в переносе папок и файлов проекта web-приложения Visual Studio в папку сервера IIS.

Публикация может быть выполнена несколькими способами. В простейшем случае ее можно осуществить простым копированием папок и файлов. Для этого следует выполнить следующую последовательность действий.

1. С помощью приложения IIS Manager создать (добавить) новый сайт. На рис. 2.4 представлено окно создания нового сайта. При этом необходимо указать имя сайта, выбрать пул приложений (система общих настроек для web-приложений), физический путь для размещения папок и файлов web-приложения, а также установить привязки (тип протокола, номер TCP-порта и IP-адрес, который будет прослушивать сайт).

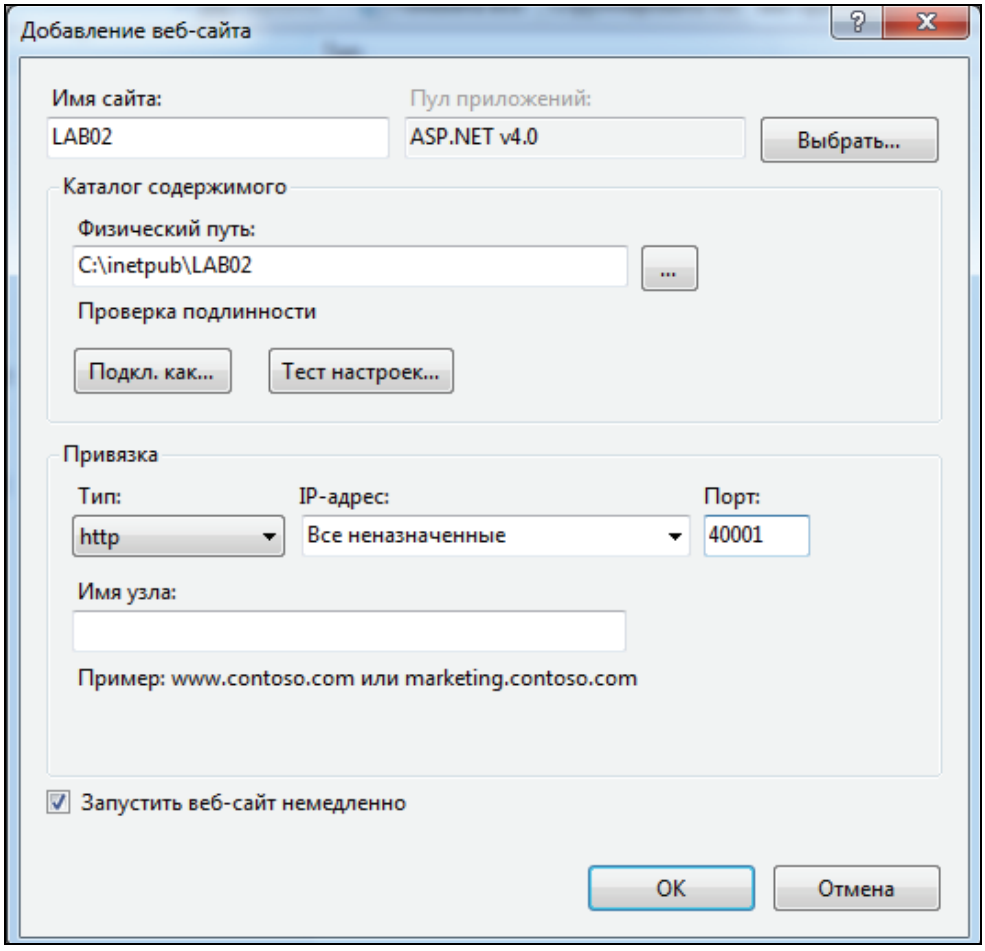

Рис. 2.4. Окно ASP.NET Development Server

2. Скопировать папки и файлы проекта web-приложения из папок Visual Studio в папку созданного на IIS сайта.

3. Проверить работоспособность сайта. Для этого следует на компьютере с установленным сервером IIS запустить браузер и в адресной

строке указать http://localhost:40001, где 40001 – номер порта, указанный при создании сайта (рис. 2.5).

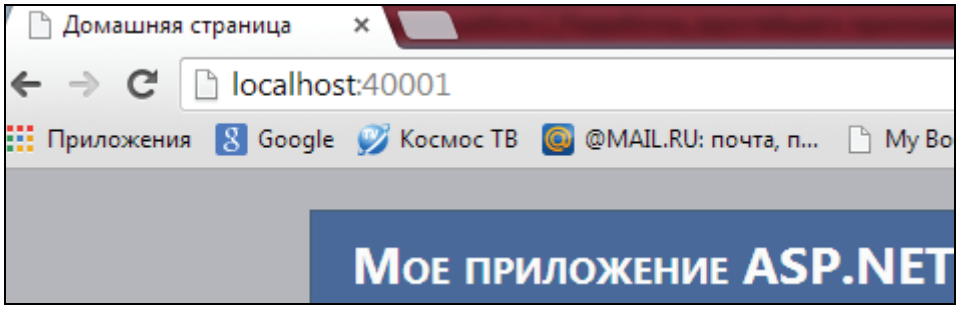

Рис. 2.5. Вызов приложения с помощью браузера

4. Удалить лишние файлы из папки сайта. После удаления лишних файлов (исходных кодов) проекта, папка сайта будет выглядеть примерно так, как представлено на рис. 2.6.

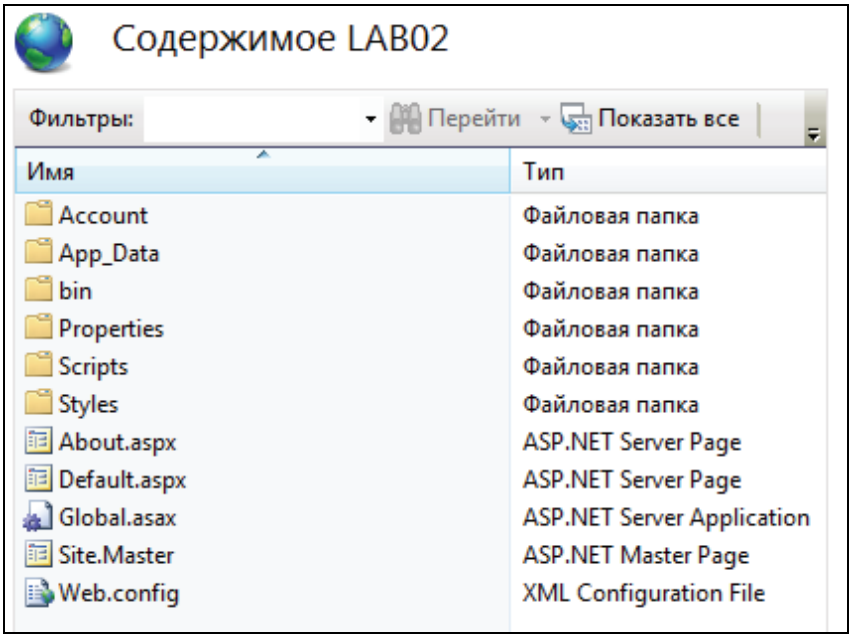

Рис. 2.6. Содержимое физической папки сайта сервера IIS

Другие способы публикации требуют установки дополнительного пакета, который может быть получен с сайта Microsoft Download Center (http://www.microsoft.com/ru-ru/download). После установки этого пакета появляется возможность публиковать web-приложения еще несколькими способами. В любом случае требуется, чтобы на IIS был предварительно создан сайт, в папку которого будут копироваться файлы приложения.

С помощью контекстного меню (рис. 2.7) может быть создан пакет развертывания.

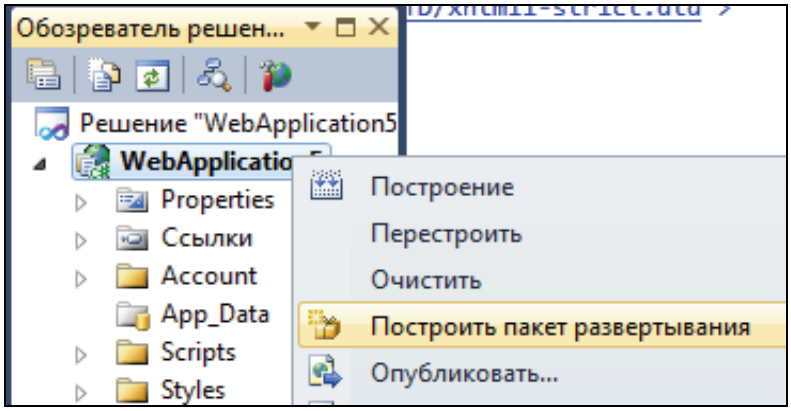

Рис. 2.7. Построение пакета развертывания

Пакет формируется в папке obj\Debug\Package проекта и представляет собой файл с расширением zip, имя которого совпадает с именем проекта. Этот файл используется для импорта web-приложения на сервер IIS. Импорт может быть выполнен с помощью контекстного меню приложения IIS Manager (рис. 2.8).

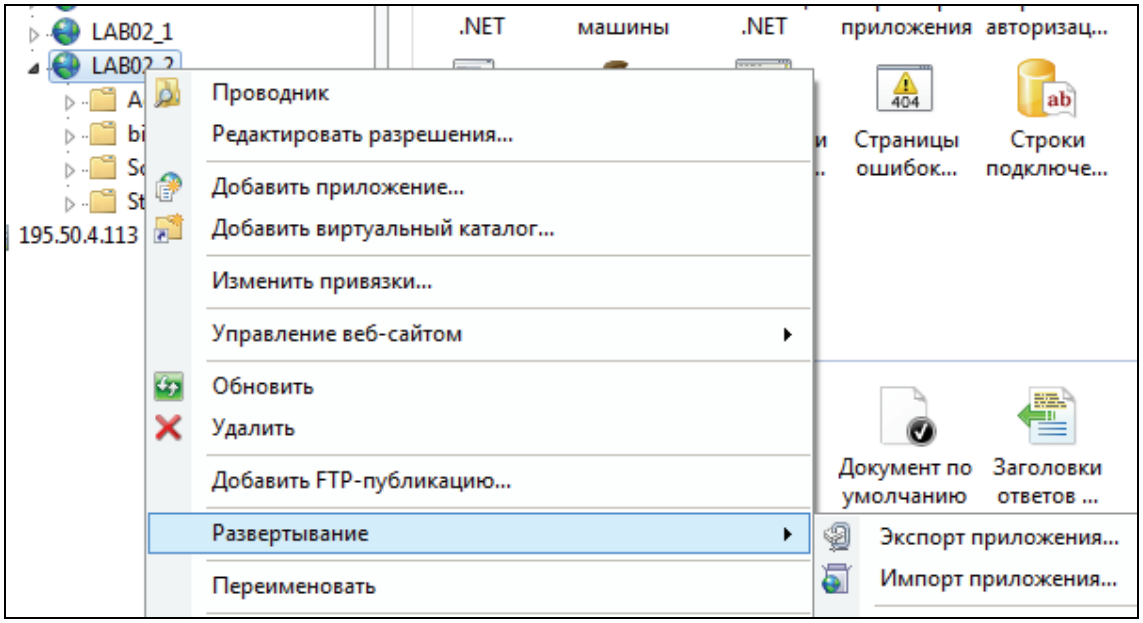

Рис. 2.8. Импорт приложения с помощью приложения IIS Manager

Успешное завершение импорта web-приложения равносильно его публикации. Для более детального знакомства с методами публикации web-приложений на IIS рекомендуются источники [6] и [7].

2.2. Задания

Задание 4. Создание простейшего приложения Web Forms ASP.NET:

1) запустите Visual Studio и создайте шаблонное приложение Web Forms ASP.NET;

2) сделайте скриншот окна «Обозреватель решений» проекта и поместите его в отчет по контрольной работе;

3) откройте в проекте файл Default.aspx и замените текст «Добро пожаловать в ASP.NET!» на текст, содержащий ваше имя, фамилию и отчество;

4) запустите созданное web-приложение на выполнение в отладочном режиме;

5) сделайте скриншот окна браузера, отображающего стартовую страницу приложения, и поместите его в отчет по контрольной работе;

6) откройте окно отладочного сервера ASP.NET Development Server, сделайте скриншот и поместите его в отчет по контрольной работе.

Задание 5. Публикация приложения на IIS:

1) запустите приложение IIS Manager и с его помощью создайте сайт;

2) опубликуйте в этот сайт созданное в задании 4 приложение методом простого копирования файлов;

3) убедитесь в работоспособности сайта;

4) с помощью приложения IIS Manager создайте еще один сайт;

5) опубликуйте в последний созданный сайт созданное в задании 4 приложение методом отличным от простого копирования файлов;

6) сделайте скриншот окна приложения IIS Manager, который бы демонстрировал наличие двух созданных в этом задании сайтов; поместите скриншот в отчет по контрольной работе.

Задание 6. Контрольные вопросы:

1) поясните термины: «шаблон приложения», «ASP.NET Development Server»;

2) поясните термины: «сайт», «публикация приложения»;

3) что такое пакет развертывания приложение, как он может быть сформирован и как применен?

4) перечислите параметры, которые необходимо указать при создании сайта с помощью приложения IIS Manager;

5) перечислите известные вам способы публикации web-приложений.

Практическая работа № 3
ИССЛЕДОВАНИЕ СТРУКТУРЫ ПРИЛОЖЕНИЯ WEB FORMS ASP.NET

3.1. Теоретические сведения

3.1.1. Структура приложения Web Forms ASP.NET

Структура web-приложения в полной мере отображается в окне «Обозреватель решений» Visual Studio (рис. 3.1).

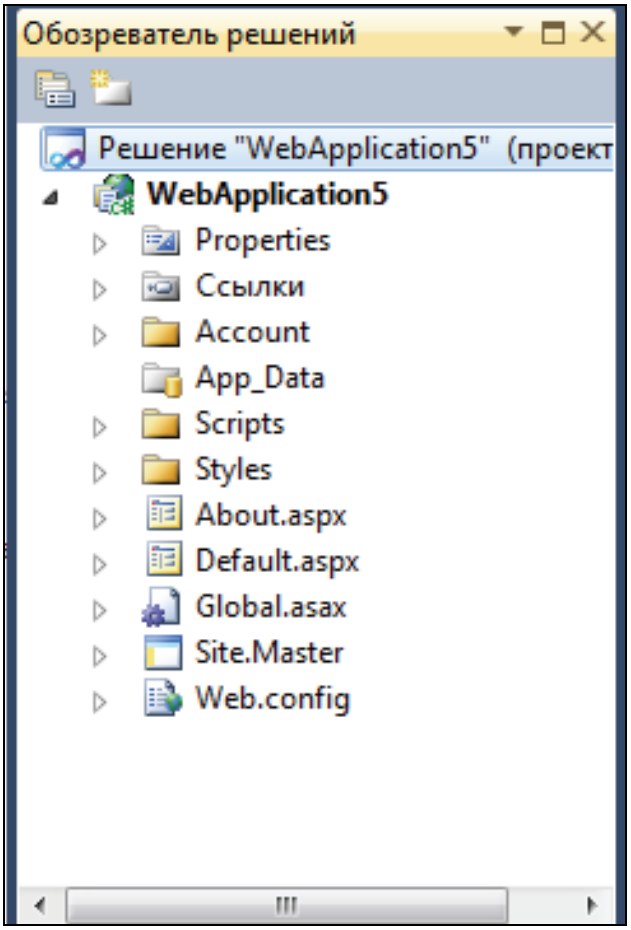

Рис. 3.1. Окно «Обозреватель решений» Visual Studio

Приложение состоит из нескольких компонентов, представляющих собой файлы, расположенные в папках со стандартными именами.

Файлы с расширением aspx – web-страницы ASP.NET. На рис. 3.2 представлена стартовая страница Default.aspx шаблонного приложения.

```
<%@ Page Title="Домашняя страница" Language="C#" MasterPageFile="~/Site.master" AutoEventWireup="true"
    CodeBehind="Default.aspx.cs" Inherits="WebApplication5._Default" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent"></asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
  <h2> Добро пожаловать в ASP.NET! </h2>
  <p>
      Для получения дополнительных сведений об ASP.NET посетите веб-сайт
      <a href="http://www.asp.net" title="Веб-сайт ASP.NET">www.asp.net</a>.
  </p>
  <p>
      Кроме того, <a href="http://go.microsoft.com/fwlink/?LinkID=152368"
                     title="Документация по ASP.NET на MSDN">документация по ASP.NET доступна на MSDN</a>.
  </p>
</asp:Content>
```

Рис. 3.2. Файл Default.aspx

Файлы с расширением aspx содержат разметку web-формы. Для разметки могут использоваться теги языка HTML и специальные aspx-теги, представляющие элементы управления ASP.NET. Все aspx-страницы начинаются с директивы Page, атрибуты которой содержат основные параметры страницы.

Файлы с расширением master называются мастер-страницами. Мастер- страницы представляют собой шаблоны, позволяющие создавать множество web-форм с одинаковой структурой. На рис. 3.2 атрибут MasterPageFile директивы Page указывает имя файла, содержащего мастер-страницу, которая используется при отображении формы.

Файлы с расширением cs – исходные коды приложения. Как правило, эти файлы связаны с aspx-страницами или другими компонентами web-приложения. Некоторые cs-файлы генерируются Visual Studio автоматически. Они, как правило, имеют расширение designer.cs.

Файл Global.asax содержит код обработчиков событий, реагирующих на глобальные события приложения. Разработчик может написать код, который ASP.NET будет выполнять при запуске приложения при его завершении, при возникновении необработанной ошибки и т. п.

Файл Web.config – конфигурационный файл. Файл содержит набор параметров (в формате XML) уровня приложения, отвечающие за настройку всех аспектов выполнения приложения, начиная с безопасности и заканчивая отладкой и управлением состоянием.

В таблице описано назначение основных папок приложения Web Forms ASP.NET.

Назначение папок приложения Web Forms ASP.NET

Наименование папки	Назначение
App_Data	Используется для хранения данных, включая файлы SQL Server Express и файлы XML
Scripts	Используется для хранения JavaScript скриптов (js-файлов), применяемых в web-формах приложения
Styles	Используется для хранения каскадных таблиц стилей (css-файлов)
Account	Часть шаблонного приложения (css-файлы), позволяющая организовать аутентификацию и авторизацию пользователей сайта
Bin	Все скомпилированные сборки .NET приложения

Кроме перечисленных типов файлов и стандартных папок, ASP.NET-приложение может содержать и другие компоненты. С полным перечнем компонентов ASP.NET-приложения можно ознакомиться в источнике [6].

3.1.2. Модель событий приложения Web Forms ASP.NET

Разработка web-приложения – это, по сути, разработка обработчиков событий. Код, разрабатываемый программистом, получает управление от ASP.NET только в результате какого-то события, возникшего в результате действий пользователя, приведших к отправке на сервер http-запроса или события, возникшего на стороне сервера.

Важно понимать, что каждый http-запрос, поступающий со стороны сервера начитает новый цикл жизни web-приложения. Говорят, что приложение не помнит своего состояния. Другими словами, если в программе установить значение некоторой переменной, то это не значит, что в следующем цикле обработки (при последующем запросе) эта переменная будет иметь установленное ранее значение.

В большинстве случаев разработчик программирует обработку событий двух типов: событий Global.asax и событий aspx-страницы.

На рис. 3.3 приведен код класса Global, методы которого являются обработчиками событий Global.asax. В комментариях перечислены события приложения, при которых выполняется код каждого обработчика.

```
namespace WebApplication5
{
    public class Global : System.Web.HttpApplication
    {

        void Application_Start(object sender, EventArgs e)
        { /*Код, выполняемый при запуске приложения*/}

        void Application_End(object sender, EventArgs e)
        { /*Код, выполняемый при завершении работы приложения */}

        void Application_Error(object sender, EventArgs e)
        {/* Код, выполняемый при возникновении необрабатываемой ошибки */}

        void Session_Start(object sender, EventArgs e)
        {/* Код, выполняемый при запуске нового сеанса */ }

    }
}
```

Рис. 3.3. Обработчики событий Global.asax

События aspx-страницы отражают ее жизненный цикл. На рис. 3.4 представлен код класса _Default, содержащий методы-обработчики событий страницы. Комментарии поясняют последовательность вызова обработчиков и события, которые они обрабатывают. Третьими по очереди осуществляется обработка событий элементов управления, которые инкапсулируются aspx-страницей. Количество таких обработчиков зависит от элементов управления, расположенных на странице, и событий, которые они могут генерировать.

```
namespace WebApplication5
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Init(object sender, EventArgs e)
        {  /*  1.вызывается при инициализации страницы */ }

        protected void Page_Load(object sender, EventArgs e)
        {  /* 2. вызывается при при загрузке страницы */  }

        //    3. обработчики событий
        //       элементов управления
        //       расположенных на этой странице

        protected void Page_PreRender(object sender, EventArgs e)
        {  /* 4. вызывается перед формированием HTML-кода */}

        protected void Page_Unload(object sender, EventArgs e)
        { /*  5. вызывается после отправки страницы   */ }
     }
}
```

Рис. 3.4. Обработчики событий Global.asax

Следует отметить, что с точки зрения ASP.NET aspx-страница является таким же элементом управления как, скажем, клавиша. Просто apx-страница – это элемент управления, который «умеет» инкапсулировать другие элементы. Поэтому любой элемент управления имеет ту же модель событий, что и aspx-страница.

Более подробно с моделью событий приложения ASP.NET можно ознакомиться в литературе [6].

3.2. Задания

Задание 7. Исследование структуры приложения и модели событий Web Forms ASP.NET:

1) создайте приложение Web Forms ASP.NET с помощью Visual Studio;

2) измените страницу Default.aspx так, чтобы она была такой же, как на рис. 3.4;

```
<%@ Page Title="Домашняя страница" Language="C#" MasterPageFile="~/Site.master" AutoEventWireup="true"
    CodeBehind="Default.aspx.cs" Inherits="WebApplication5._Default" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent"></asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
  <h2> Практическая работа № 3 </h2>
  <p>
    <br /><br />
    <asp:Button ID="Button1" runat="server" Text="Button1"
                onclick="Button1_Click" />
    <asp:Button ID="Button2" runat="server" Text="Button2" style="margin-left: 34px"
                onclick="Button2_Click" />
    <br /><br />
    <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
  </p>
 </asp:Content>
```

Рис. 3.5. Файл Default.aspx

3) откорректируйте программный код в файле Default.aspx.cs, чтобы он стал таким же, как на рис. 3.6; отличаться может имя пространства имен (указывается в операторе namespace), которое в шаблонном приложении совпадает с именем приложения, задаваемым при создании;

```
namespace WebApplication5
{
        public partial class _Default : System.Web.UI.Page
        {
            private int k = 0;
            protected void Page_Init(object sender, EventArgs e)
            { this.Label1.Text += "Init["+ (++k).ToString()+"]-"; }

            protected void Page_Load(object sender, EventArgs e)
            { this.Label1.Text += "Load[" + (++k).ToString() + "]-"; }

            protected void Button1_Click(object sender, EventArgs e)
            { this.Label1.Text += "Click1[" + (++k).ToString() + "]-"; }

            protected void Button2_Click(object sender, EventArgs e)
            { this.Label1.Text += "Click2[" + (++k).ToString() + "]-";}

            protected void Page_PreRender(object sender, EventArgs e)
            { this.Label1.Text += "PreRender[" + (++k).ToString() + "]-"; }

            protected void Page_Unload(object sender, EventArgs e)
            { this.Label1.Text += "Unload[" + (++k).ToString() + "]-"; }
         }
 }
```

Рис. 3.6. Файл Default.aspx.cs

4) запустите приложение на выполнение в отладочном режиме; убедитесь, что в окне браузера отображается форма такая же, как на рис. 3.7;

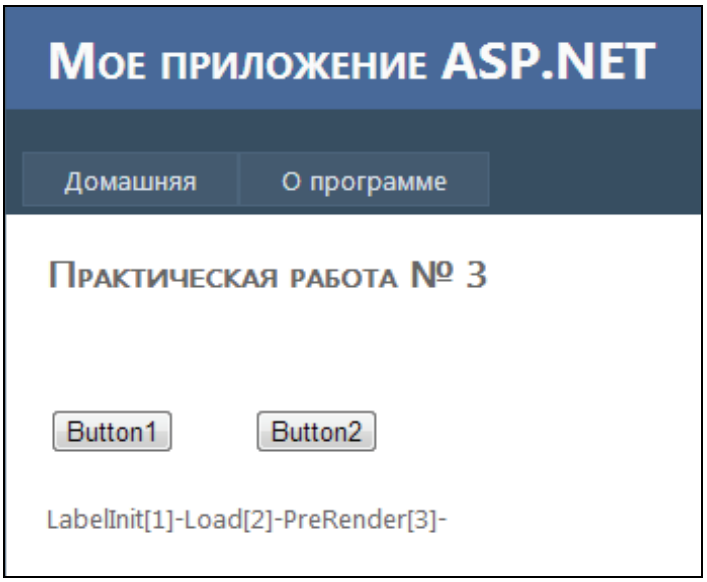

Рис. 3.7. Отображение формы Default.aspx в окне браузера

1) сделайте скриншот окна браузера, отображающего страницу Default.aspx приложения и поместите его в отчет по контрольной работе;

2) с помощью мыши нажмите клавишу Button1; обратите внимание на изменение текста строки, расположенной под кнопками;

3) сделайте скриншот окна браузера, отображающего страницу Default.aspx приложения и поместите его в отчет по контрольной работе;

4) с помощью мыши нажмите клавишу Button2; обратите внимание на изменение текста строки, расположенной под кнопками;

5) сделайте скриншот окна браузера, отображающего страницу Default.aspx приложения и поместите его в отчет по контрольной работе;

6) опубликуйте приложение на IIS и убедитесь в его работоспособности.

Задание 8. Контрольные вопросы:

1) перечислите все типы файлов, которые могут входить в приложение Web Forms ASP.NET, поясните их назначение;

2) перечислите все наименования стандартных папок приложения Web Forms ASP.NET, поясните их назначение;

3) перечислите в порядке их появления все события, которые могут быть обработаны методами класса Global (файл Global.asax.cs);

4) перечислите все события страницы в порядке их появления;

5) объясните принцип формирования строки на форме приложения, разработанного в задании 7.

Практическая работа № 4
РАЗРАБОТКА И ПРИМЕНЕНИЕ HTTP-ОБРАБОТЧИКА ASP.NET

4.1. Теоретические сведения

4.1.1. Разработка HTTP-обработчика ASP.NET

Каждый поступающий в приложение ASP.NET запрос предварительно обрабатывается специально предназначенным для этого компонентом, который называется обработчиком HTTP. Обработчики создаются для каждого типа файлов. В состав ASP.NET входит несколько стандартных обработчиков. Например, для обработки файлов с расширениями aspx, asmx, ashx и т.д.

Сопоставление обработчиков расширениям можно увидеть с помощью приложения IIS Manager. Для этого следует выбрать с помощью курсора мыши сайт, а затем сделать двойной щелчок на иконке с надписью «сопоставление обработчиков» (рис. 4.1).

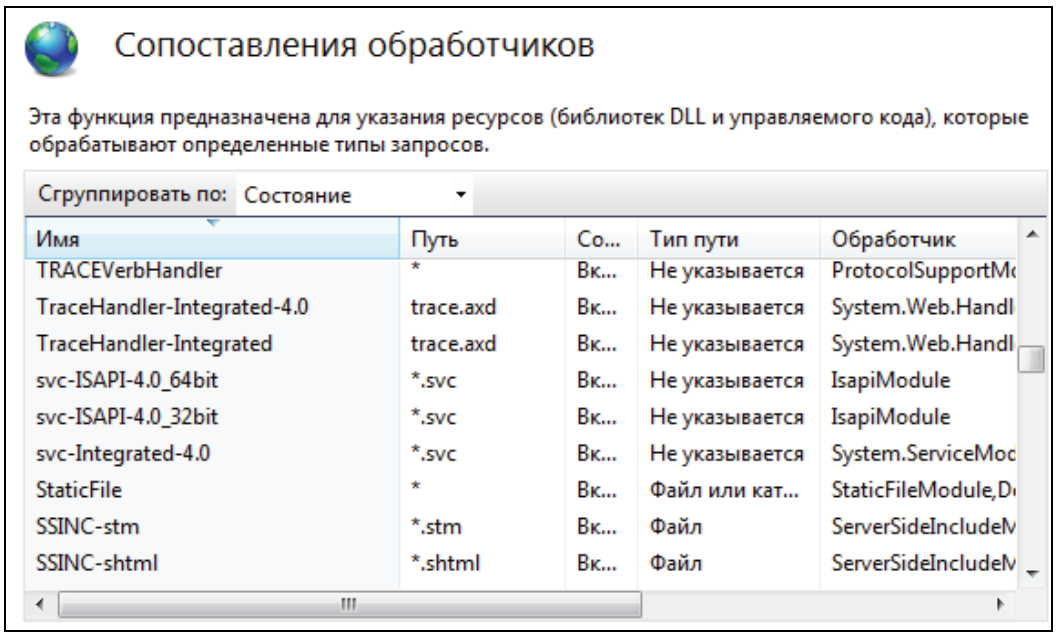

Рис. 4.1. Отображение обработчиков в приложении IIS Manager

При выполнении приложения Web Forms ASP.NET в командной строке запрашивается страница с расширением aspx. ASP.NET обрабатывает этот запрос с помощью стандартного (входящего в состав

ASP.NET) HTTP-обработчика, который и управляет всем жизненным циклом приложения.

Разработка HTTP-обработчиков считается низкоуровневым программированием ASP.NET и применяется для решения специфических задач: например, для скачивания с сервера файла, считывания видео или аудио потоков.

С точки зрения программиста HTTP-обработчик это класс реализующий интерфейс IHttpHandler. На рис. 4.2 представлен пример простейшего HTTP-обработчика.

```
public class FirstHandler : IHttpHandler                // HTTP-обработчик
{
    public bool IsReusable { get { return true; } }     // можно повторно использовать обработчик ?
    public void ProcessRequest(HttpContext context)     // обработка запроса
    {

        HttpResponse response = context.Response;       // объект ОТВЕТ
        HttpRequest request = context.Request;          // объект ЗАПРОС
        string parm = (request.Params["parm"] == null? "null":request.Params["parm"]); // параметр есть ?
        response.Write("Http-обработчик FirstHandler, parm=" + parm);  //  вывод на форму
    }
}
```

Рис. 4.2. Пример HTTP-обработчика

Интерфейс IHttpHandler предполагает реализацию свойства IsReusable и метода ProcessRequest.

Свойство IsReusable принимает два логических значения: истина или ложь. Истинное значение указывает на возможность повторного применения экземпляра класса обработчика в другом потоке. В альтернативном случае для каждого потока будет создаваться собственный экземпляр класса обработчика.

Связь классов-обработчиков с методами HTTP-запроса и расширениями запрашиваемой страницы описывается в разделе handlers конфигурационного файла **Web.config** приложения (рис. 4.3).

```
<system.webServer>
   <modules runAllManagedModulesForAllRequests="true"/>

  <handlers>

    <add verb="GET"  path="*.fotproject"  type="Lib4.FirstHandler" name="GETProject" />

    <add verb="POST" path="*.fotvalidate" type="Lib4.TwoHandler"   name="POSTValidate" />

  </handlers>

</system.webServer>
```

Рис. 4.3. Фрагмент файла Web.config, описывающий HTTP-обработчики

Два тега add описывают два HTTP-обработчика, реализованных как классы FirstHandler и TwoHandler, расположенные в пространстве имен Lib4 (атрибут type). Для каждого обработчика указывается HTTP-метод (verb), расширение, запрашиваемой страницы (patch), а также имя (name), отображаемое приложением IIS Manager в окне «Сопоставление обработчиков» (рис. 4.1).

4.1.2. Разработка HTTP-клиента, взаимодействующего с HTTP-обработчиком ASP.NET

В простейшем случае GET-запрос HTTP-обработчику может быть отправлен с помощью браузера. Для этого достаточно в адресной строке браузера правильно набрать соответствующий обработчику URL. Для формирования POST-запроса, следует разработать web-форму, содержащую тег form со значением POST в атрибуте method и правильным URL в атрибуте action. На рис. 4.4 приведен пример web-формы, формирующая POST-запрос.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head><title> HTTP-обработчик KRAS_12_FOT</title></head>
<body>
  <form  id="f1" action= "http://smwcorei7:40007/FOT.fotvalidate"
         method = "post" enctype ="multipart/form-data" >
      <br /> <input type="file" id="FOTFile" name="FotFile"
                 style="border: 3px solid; width: 400px;"   />
           <input id="Submit1" type="submit"  value="отправить"  />
   </form>
</body>
</html>
```

Рис. 4.4. Web-форма, формирующая POST-запрос

В том случае, если запрос необходимо сформировать в приложении на языке C# следует воспользоваться возможностями предоставляемыми встроенным классом System.Net.WebClient. На рис. 4.5. представлен пример применения класса WebClient для формирования POST-запроса, пересылающего серверу (HTTP-обработчику) файл с клиентского компьютера.

```
static void Main(string[] args)
{
    WebClient client = new WebClient();
    byte[] array = client.UploadFile("http://localhost:40007/FOT.fotproject",  // http-обработчик
                                     @"D:\KRAS_12\FOT.project");
    String s = Encoding.Default.GetString(array, 0, array.Length);            // обработка ответа
}
```

Рис. 4.5. Пример применения встроенного класса WebClient

4.1. Задания

Задание 9. Разработка HTTP-обработчика ASP.NET:

1) разработайте HTTP-обработчик, реагирующий на GET-запрос страниц с расширением bstu; HTTP-обработчик должен принимать значения двух параметров: faculty и course; в результате выполнения в окне браузера должна отобразиться следующая строка:

GET: bstu, faculty = *val_parm1*, course = *val_parm2*,

где *val_parm1* и *val_parm2* – значения первого и второго параметров.

2) разработайте HTTP-обработчик, реагирующий на POST-запрос страниц с расширением bstu; HTTP-обработчик должен принимать значения двух параметров: faculty и course; в результате выполнения в окне браузера должна отобразиться следующая строка:

POST: bstu, faculty = *val_parm1*, course= *val_parm2*,

где *val_parm1* и *val_parm1* – значения первого и второго параметров.

3) опубликуйте HTTP-обработчики на IIS;

4) с помощью приложения IIS Manager отобразите окно «сопоставление обработчиков», найдите разработанные в предыдущих пунктах задания обработчики, сформируйте скриншот и поместите его в отчет по контрольной работе;

5) с помощью браузера выполните GET-запрос к HTTP-обработчику и убедитесь в его работоспособности;

6) разработайте web-форму, формирующую POST-запрос к HTTP-обработчику; опубликуйте эту web-форму IIS; выполните с помощью браузера запрос к форме, сформируйте POST-запрос и убедитесь в работоспособности обработчика; текст разработанной web-формы поместите в отчет по контрольной работе;

7) поместите исходные коды HTTP-обработчиков в отчет по контрольной работе.

Задание 10. Разработка клиента, взаимодействующего с HTTP-обработчиком ASP.NET:

1) разработайте консольное приложение на языке C#, формирующее GET и POST-запросы к HTTP-обработчикам, разработанным в задании 9; приложение должно отображать ответ, оправленный обработчиками в окне консоли;

2) выполните приложение, разработанное в предыдущем пункте задания; убедитесь в работоспособности приложения и HTTP-обработчиков; сформируйте скриншот, отражающий результат вы-

полнения приложения; поместите скриншот и исходный код приложения в отчет по контрольной работе.

Задание 11. Контрольные вопросы:

1) поясните понятие «HTTP-обработчик ASP.NET»;

2) каким образом можно отобразить перечень HTTP-обработчиков, действующих в рамках сайта?

3) поясните структуру HTTP-обработчика (интерфейс, назначение методов);

4) где располагается информация связывающая класс HTTP-обработчика, метод HTTP-запроса и расширение страниц;

5) в каких случаях целесообразно применять HTTP-обработчики?

Практическая работа № 5
ПРИМЕНЕНИЕ СЕРВЕРНЫХ HTML-ЭЛЕМЕНТОВ УПРАВЛЕНИЯ ASP.NET

5.1. Теоретические сведения

5.1.1. Серверные HTML-элементы управления

Для разработки пользовательского интерфейса web-приложения шаблона Web Forms ASP.NET программист использует серверные элементы управления. С их помощью разработчик приложения формирует изображение, отображающееся в окне браузера. Каждому серверному HTML-элементу управления соответствует с одной стороны asp-тег, который может быть размещен на aspx-странице, с другой – класс, производный от класса Control из пространства имен System.Web.UI.

Некоторые серверные элементы управления являются вспомогательными, не генерируют HTML-код и, соответственно, не отображаются браузером. Другие, наоборот, помимо HTML-кода, генерируют JavaScript-код, позволяющий придать динамичность пользовательскому интерфейсу.

С подробной классификацией серверных элементов управления можно ознакомиться в источнике [6], а с полным перечнем просмотрев список в панели элементов (Toolbox) Visual Studio (рис. 5.1).

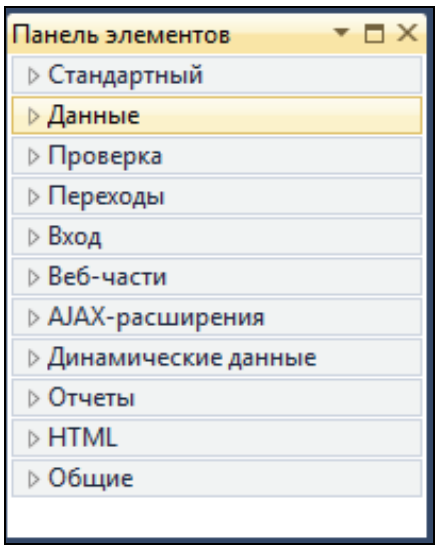

Рис. 5.1. Панель элементов Visual Studio

Серверные HTML-элементы управления являются наиболее простыми в использовании элементами управления ASP.NET. Основной их особенностью является то, что каждому HTML-элементу соответствует один тег HTML. Классы, соответствующие HTML-элементам, являются производными от класса HtmlControl из пространства имен System.Web.UI. Каждый элемент может генерировать одно из двух серверных (обработанных на сервере) событий: ServerClick или ServerChange.

С полным перечнем HTML-элементов можно ознакомиться, открыв вкладку с надписью HTML в панели элементов Visual Studio (рис. 5.2).

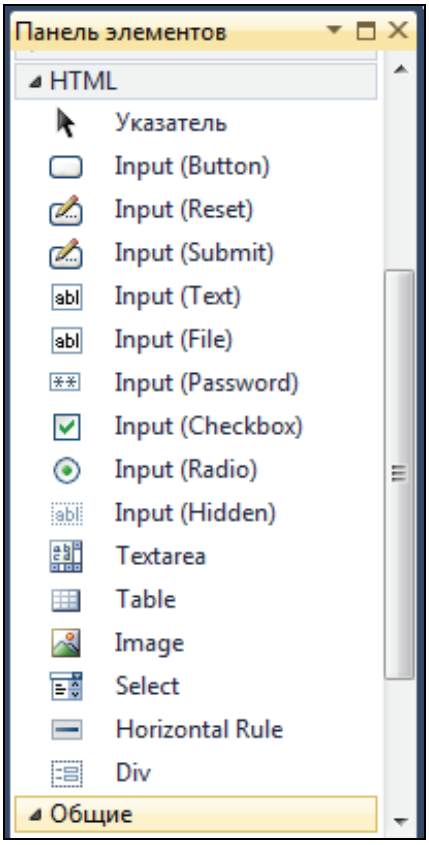

Рис. 5.2. Перечень HTML-элементов в панели элементов Visual Studio

5.1.2. Применение серверных элементов управления HTML

На рис. 5.3 представлен пример aspx-страницы, содержащей два серверных элемента HTML: HtmlInputText (представлен тегом <input type=”text”>) и HtmlInputButton (<input type=”button”>). Элемент HtmlInputText генерирует событие ServerChange, возникающее при изменении содержимого текстового значения соответствующего тега.

Элемент HtmlInputButton генерирует событие ServerClick, возникающее при щелчке мышью по кнопке соответствующего тега.

```
<%@ Page Title="Домашняя страница" Language="C#" MasterPageFile="~/Site.master"
    AutoEventWireup="true" CodeBehind="Default.aspx.cs"
    Inherits="WebApplication7._Default" %>
<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
   <p>

       <input id="Text1" type="text" runat="server" onserverchange="Text1_OnServerChange" />
       <input id="Button1" type="button" value="button" runat="server"
              onserverclick="Button1_OnServerClick"/>

   </p>
</asp:Content>
```

Рис. 5.3. Aspx-страница, содержащая серверные элементы управления HTML

На рис. 5.4 представлен код класса _Default, содержащий два метода-обработчика событий ServerChange и ServerClick.

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    public void Text1_OnServerChange(Object sender, EventArgs e) // serverchange-обработчик
    {
        this.Response.Write("Text1_OnServerChange");
    }
    public void Button1_OnServerClick(Object sender, EventArgs e) // serverсдшсл-обработчик
    {
        this.Response.Write("Button1_OnServerClick");
    }
}
```

Рис. 5.4. Обработчики событий серверных элементов управлений HTML

5.2. Задания

Задание 12. Применение серверных HTTL-элементов управления:

1) исследуйте раздел HTML панели инструментов Visual Studio;

2) разработайте web-приложение, использующее все серверные HTML-элементы управления; код обработчиков событий должен выводить на форму сообщение следующего вида:

html-мег: событие ,

где *html-мег* – наименование тега, *событие* – обработанное событие;

3) текст кода aspx-страницы и C#-кода поместить в отчет по контрольной работе.

Задание 13. Контрольные вопросы:

1) поясните понятие «серверный элемент управления ASP.NET»;

2) назовите общий базовый класс для всех серверных элементов управления ASP.NET;

3) назовите особенности HTML-элементов управления ASP.NET;

4) назовите общий базовый класс для всех серверных HTML-элементов управления ASP.NET;

5) перечислите все серверные HTML-элементы управления, генерирующие событие ServerChange;

6) перечислите все серверные HTML-элементы управления, генерирующие событие ServerClick.

Практическая работа № 6
ПРИМЕНЕНИЕ СЕРВЕРНЫХ БАЗОВЫХ WEB-ЭЛЕМЕНТОВ УПРАВЛЕНИЯ ASP.NET

6.1. Теоретические сведения

6.1.1. Серверные базовые web-элементы управления

Серверными web-элементами управления называются элементы управления, которые являются производными от класса WebControl из пространства имен System.Web.UI.WebControls. Среди этих элементов выделим следующие: Button, CheckBox, FileIpload, HiddenField, HyperLink, Image, ImageButton, ImageMap, Literal, Label, LinkButton, Panel, RadioButton, Table, TableCell, TableRow, TextBox. Эти элементы называют базовыми web-элементами управления ASP.NET. В панели элементов Visual Studio web-элементы расположены во вкладке «Стандартный» (рис. 6.1).

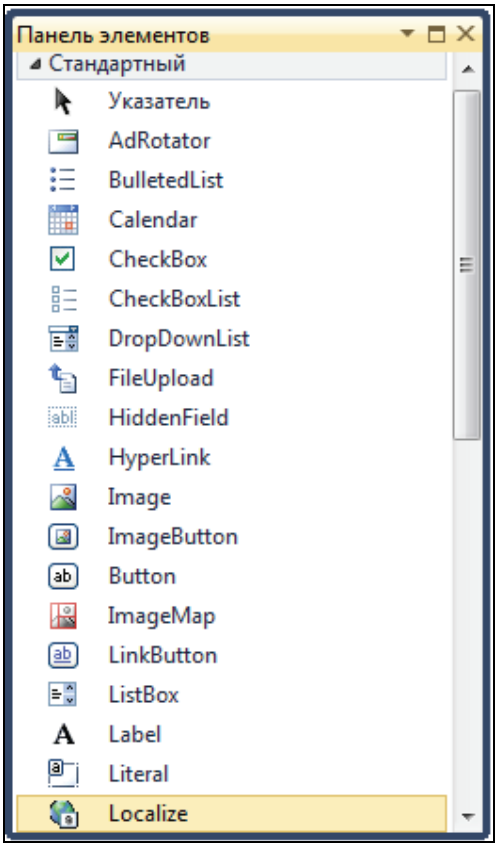

Рис. 6.1. Раздел «Стандартный» панели элементов Visual Studio

На aspx-странице серверные web-элементы управления имеют префикс asp. На рис. 6.2 представлен пример использования двух серверных web-элементов: TextBox и Button. Важной особенностью web-элементов является большое количество свойств и событий, поддерживаемых этими элементами. С помощью Visual Studio можно отобразить окна с перечнем свойств и событий, генерируемых web-элементами (рис. 6.3). Visual Studio позволяет не только просматривать свойства и события, но и задавать значения свойствам и формировать методы-обработчики событий.

```
<%@ Page Title="Домашняя страница" Language="C#" MasterPageFile="~/Site.master"
    AutoEventWireup="true" CodeBehind="Default.aspx.cs"
    Inherits="WebApplication7._Default" %>
<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
   <p>
       <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
       <asp:Button ID="Button1" runat="server" Text="Кнопка" onclick="Button1_Click" />
   </p>
</asp:Content>
```

Рис. 6.2. Aspx-страница, содержащая серверные web-элементы TextBox и Button

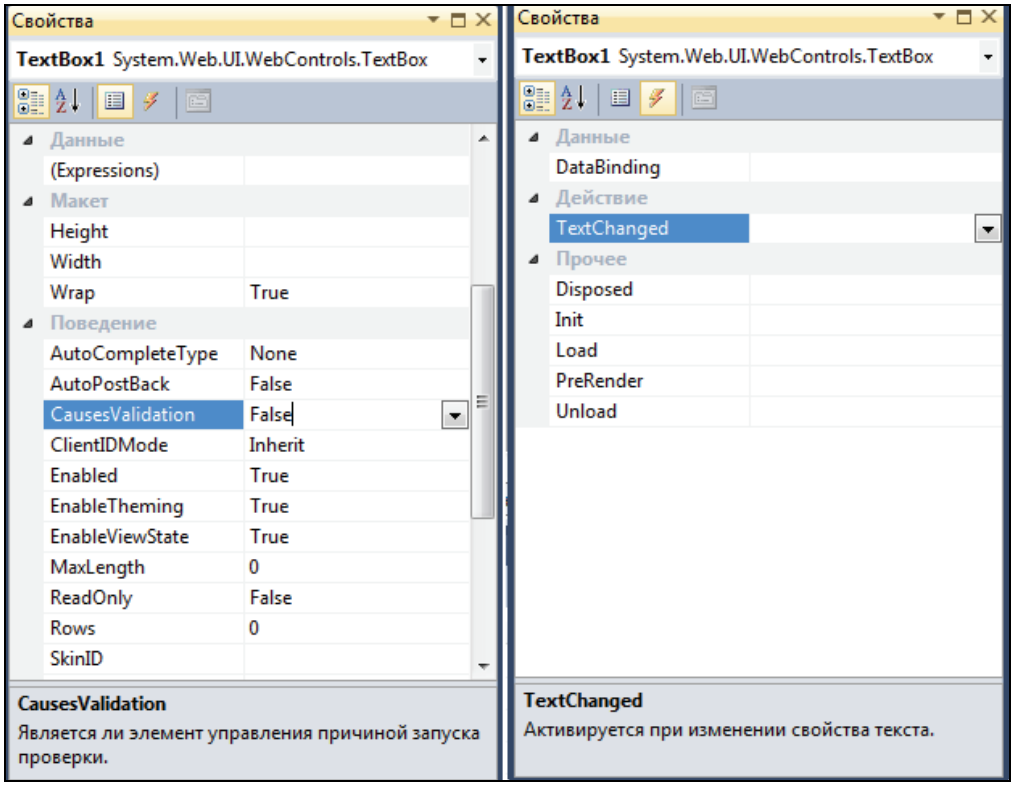

Рис. 6.3. Свойства и события элемента TextBox

Результатом обработки сервером aspx-страницы будет HTML и JavaScript-код, который может интерпретироваться браузером. Обычно отправка на сервер с клиента на сервер осуществляется браузером в случае нажатия пользователем кнопки <input type=”submit”> . Этот тег может быть сгенерирован несколькими web-элементами. Например, элементом Button. Лишь после оправки данных на сервере будут обработаны события, связанные с другими элементами.

Следует обратить внимание на свойство AutoPostBack, которым обладают некоторые серверные web-элементы. Установив значение true для этого свойства, можно осуществить отправку данных на сервер без нажатия кнопки. Такой эффект достигается с помощью JavaScript-кода, который генерируется ASP.NET при трансляции aspx-страницы.

6.1.2. Применение серверных базовых web-элементов управления

Visual Studio предоставляет разработчику мощный инструмент, позволяющий быстро разрабатывать приложения шаблона Web Forms. На рис. 6.4 изображена aspx-страница, отображаемая Visual Studio в режиме конструктора.

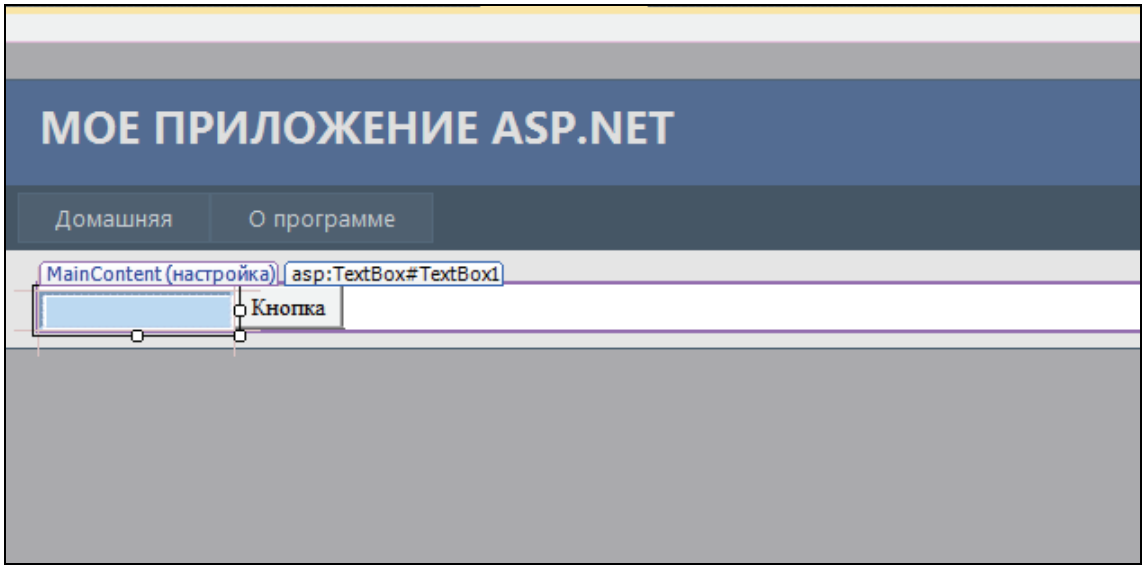

Рис. 6.4. Aspx-страница в режиме конструктора

В этом режиме, разработчик может перетягивать элементы управления с панели управления прямо на форму, динамически менять размеры и месторасположение элементов на форме, с помощью контекстного меню устанавливать свойства элементов и создавать шаблоны обработчиков событий.

Принципы обработка событий для web-элементов практически ничем не отличаются от обработки событий серверных HTML-элементов. На рис. 6.5. приведен пример обработчика события Click для web-элемента Button.

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    protected void Button1_Click(object sender, EventArgs e) // обработчик события
    {
        this.Response.Write("<br /> Button1_Click");
        this.Response.Write("<br /> TextBox1.Text = "+TextBox1.Text);

    }
```

Рис. 6.5. Пример обработчика события для web-элемента Button

6.2. Задания

Задание 14. Применение серверных базовых web-элементов управления:

1) исследуйте раздел «Стандартный» панели инструментов Visual Studio и найдите базовые web-элементы;

2) разработайте web-приложение, демонстрирующее применение базовых web-элементов, перечисленных в таблице; в приложении должны быть реализованы обработчики событий указанных в этой же таблице.

Элементы и события приложения

Элементы	События
Button	Click
TextBox	TextChanged
CheckBox	CheckedChanged
RadioButton	CheckedChanged
HyperLink	

3) разработайте еще одно web-приложение демонстрирующее применение свойства AutoPostBack для web-элемента CheckBox;

4) тексты программ включите в отчет по контрольной работе.

Задание 15. Контрольные вопросы:

1) перечислите отличия между серверными HTML и web-элементами управления ASP.NET;
2) перечислите все базовые web-элементы управления и поясните их назначение и принцип применения;
3) назовите общий базовый класс для всех серверных web-элементов управления ASP.NET;
4) назовите общие свойства для всех серверных web-элементов управления;
5) поясните назначение свойства AutoPostBack web-элементов и механизм его реализации.

Практическая работа № 7
ПРИМЕНЕНИЕ СЕРВЕРНЫХ ПОЛНОФУКЦИОНАЛЬНЫХ ЭЛЕМЕНТОВ УПРАВЛЕНИЯ ASP.NET

7.1. Теоретические сведения

7.1.1. Серверные полнофункциональные элементы управления

Полнофункциональные элементы управления – это серверные web-элементы управления ASP.NET, моделирующие сложные структуры пользовательского интерфейса. В состав ASP.NET входит много полнофункциональных элементов разных категорий. В этой практической работе будут рассматриваться только универсальные элементы: AdRotator, Calendar, View, MultiView, Wizard, Menu и TreeView. Первые пять элементов управления можно обнаружить в разделе «Стандартный» панели элементов Visual Studio, а два последних – в разделе «Переходы».

Элемент AdRotator позволяет создавать рекламный баннер, в котором в соответствии с расписанием, задаваемом в виде XML-файла, меняются изображения.

Calendar – элемент, отображающий на web-странице календарь, позволяющий перемещаться по датам, выбирать одну дату или диапазон дат.

Элементы View и MultiView применяются совместно и позволяют в одной области окна браузера разместить несколько слоев (элемент View) с отображаемыми элементами управления, а также осуществлять переключение (MultiView) между этими слоями, делая один из слоев видимым, а остальные нет.

Элемент Wizard позволяет в пошаговой удобной для клиента форме ввести или выбрать данные.

Элементы TreeView и Menu позволяют предложить пользователю выбор данных в форме древовидного меню или осуществлять переходы между страницами в соответствии с логикой приложения.

7.1.2. Применение полнофункциональных элементов управления

Применение полнофункциональных элементов сводится к трем действиям: 1) разместить элемент на web-форме (перетащить с помощью мыши из панели элементов); 2) настроить вид и свойства элемента; 3) разработать обработчики событий элемента.

На рис. 7.1 изображен элемент Calendar в режиме «Конструктор» Visual Studio. Меню, которое отображается в окне «Автоформат» позволяет выбрать вид, в котором календарь будет отображаться на aspx-странице.

С помощью контекстного меню можно просмотреть и настроить свойства полнофункционального элемента. На рис. 7.2 отображено окно свойств элемента Calendar. С помощью этого же окна можно создать шаблоны методов обработчиков событий этого элемента.

На рис. 7.3 представлен пример обработчика события Selection-Changed, возникающего при выборе пользователем даты или интервала дат.

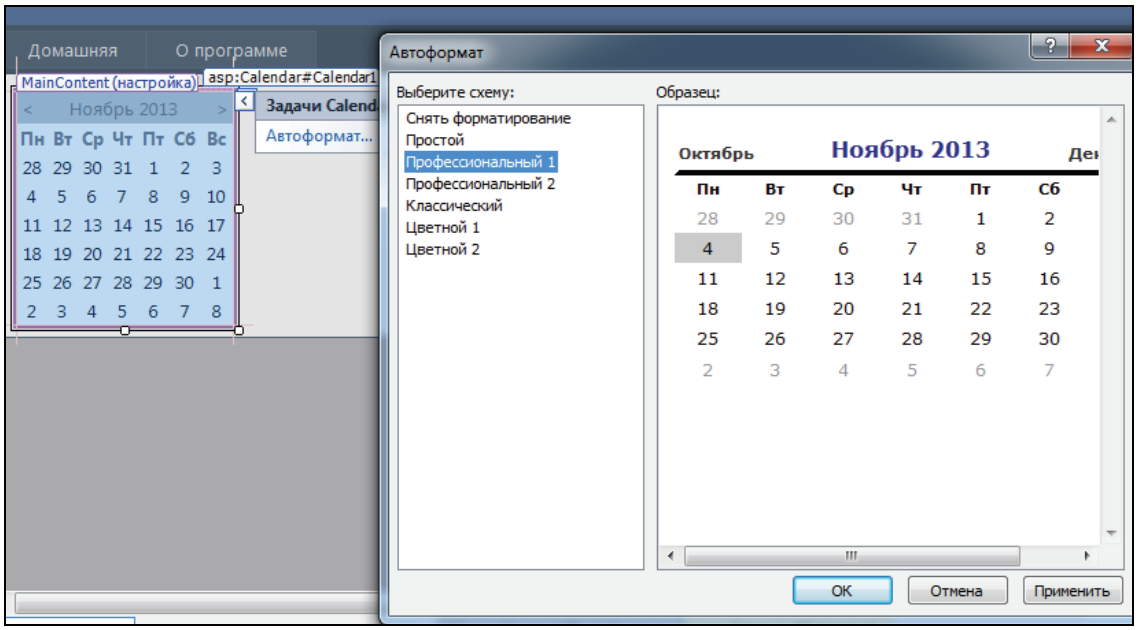

Рис. 7.1. Настройка внешнего вида полнофункционального элемента Calendar

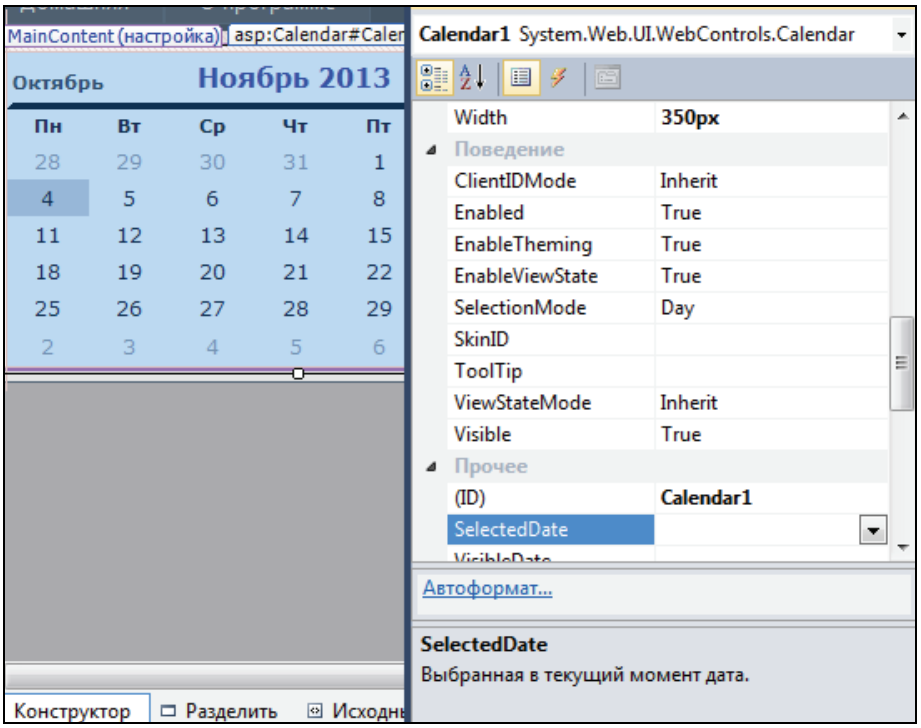

Рис. 7.2. Настройка свойств полнофункционального элемента Calendar

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)...
    protected void Calendar1_SelectionChanged(object sender, EventArgs e)
    {
        foreach (DateTime d in Calendar1.SelectedDates)
        {
            this.Response.Write(d.ToString());

        }
    }
}
```

Рис. 7.3. Пример обработчика события SelectionChanged полнофункционального элемента Calendar

7.2. Задания

Задание 16. Применение серверных базовых web-элементов управления:

1) разработайте приложение, поддерживающее работу шести web-форм, имена и назначение которых приведены в таблице;

Aspx-страницы приложения

Наименование страницы	События
Default	Стартовая страница, содержит пять элементов меню, соответствующих другим страницам и позволяющим на них перейти
View	Страница, демонстрирующая работу элементов View и MultiView. В рамках одного MultiView должно осуществляться переключение между тремя различными View, содержащими различные элементы.
Calendar	Страница, демонстрирующая работу элемента Calendar.
Wizard	Страница, демонстрирующая работу элемента Wizard. Wizard должен иметь не менее пяти шагов с вводом данных с помощью элементов TextBox, CheckBox и группы элементов RadioButton.
AdRotator	Страница, демонстрирующая работу элемента Wizard. Элемент должен отображать три различных графических изображения с частотой соответственно 50, 30 и 20.
TreeView	Страница, демонстрирующая работу элемента TreeView. Элемент должен сдержать не менее трех уровней иерархии в дереве выбора

2) каждый полнофункциональный элемент, разработанного приложения, должен иметь не менее одного обработчика любого (на выбор студента) события;

3) тексты программ включите в отчет по контрольной работе.

Задание 17. Контрольные вопросы:

1) поясните понятие «полнофункциональный элемент ASP.NET»;

2) перечислите все известные вам полнофункциональные элементы управления и поясните их назначение;

3) назовите общий базовый класс для всех полнофункциональных элементов управления ASP.NET.

Практическая работа № 8
ПРИМЕНЕНИЕ СЕРВЕРНЫХ ЭЛЕМЕНТОВ УПРАВЛЕНИЯ ПРОВЕРКОЙ ДОСТОВЕРНОСТИ ASP.NET

8.1. Теоретические сведения

8.1.1. Серверные элементы управления проверкой достоверности

Серверные элементы управления проверкой достоверности (далее просто элементы проверки) предназначены для проверки вводимых пользователем данных. Сами элементы проверки не отображаются браузером, но в окне браузера может быть выведен результат их работы – сообщение об обнаруженной ошибке ввода данных. Как правило, проверке подвергаются данные, вводимые с помощью элементов управления TextBox, но проверка применима и к другим элементам управления, предназначенным для ввода данных: ListBox, DropDownList, RadioButtonList, HtmlInputText, HtmlTextArea и HtmlSelect.

Элементы проверки можно обнаружить в разделе «Проверка» панели элементов Visual Studio (рис. 8.1).

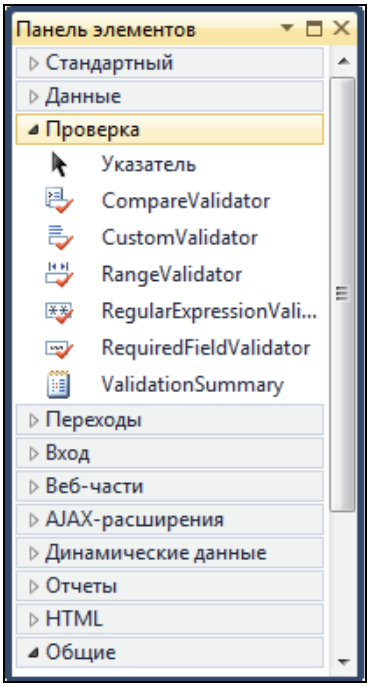

Рис. 8.1. Раздел «Проверка» панели элементов Visual Studio

ASP.NET предлагает к применению шесть элементов проверки: CompareValidator, CustomValidator, RangeValidator, RegularExpressionValidator, RequiredFieldValidator и ValidationSummary. Все классы элементов проверки находятся в пространстве имен System.Web.UI.WebControl и являются производными от класса BaseValidator.

CompareValidator – элемент проверки, позволяющий сравнить вводимое значение с каким-то другим фиксированным значением либо, что встречается чаще, со значением, содержавшемся в другом элементе управления.

CustomValidator – элемент проверки, позволяющий разработчику применить собственную процедуру проверки. Как правило, к этому виду элементов проверки прибегают в том случае, если нет возможности осуществить проверку с помощью другого.

RangeValidator – элемент проверки, предназначенный для контроля диапазона вводимых данных;

RegularExpressionValidator – элемент проверки, позволяющий сравнить вводимое значение с образцом, записанным с помощью регулярного выражения.

RequiredFieldValidator – элемент проверки значения на пустоту.

ValidationSummary – элемент, не выполняющий никакой проверки, а предназначенный для объединение сообщений нескольких элементов проверки в общую группу для их совместного отображения.

Несколько элементов проверки, расположенных на одной форме, могут быть объединены в группу (свойство Validation Group), которая может быть связана с определенным элементом, генерирующим submit.

Следует отметить, что проверка данных осуществляется два раза: первый раз на стороне клиента (для этого генерируется специальный JavaScript-код), второй – после успешной проверки на стороне клиента и нажатия клавиши submit на стороне сервера.

8.1.2. Применение элемента управления проверкой достоверности

На рис. 8.2 представлен пример aspx-страницы, содержащей два элемента типа TextBox (с идентификаторами TextBox1 и TextBox2) и один элемент проверки (CompareValidator1) типа CompareValidator.

ASP-тег CompareValidator описывает элемент проверкиCompareValidator1, который сравнивает значение, введенное элементом с идентификатором TextBox2 (атрибут ControlToValidate) со значением введенным элементом идентификатором TextBox1 (ControlToCompare).

```
<%@ Page Title="Домашняя страница" Language="C#" MasterPageFile="~/Site.master"
    AutoEventWireup="true" CodeBehind="Default.aspx.cs"
    Inherits="WebApplication7._Default" %>
<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">


    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
    <asp:Button ID="Button1" runat="server" Text="Button" />

    <asp:CompareValidator ID="CompareValidator1" runat="server"
        ErrorMessage="Ошибка: Второе значение &lt;=   первого"
        ControlToCompare="TextBox1" ControlToValidate="TextBox2"
        Operator="GreaterThan"></asp:CompareValidator>

</asp:Content>
```

Рис. 8.2. Пример применения элемента проверки типа CompareValidator

В случае, если значение не удовлетворяет условию проверки (Operator), формируется соответствующее сообщение (ErrorMesage). На рис. 8.3. представлена aspx-страница (рис. 8.2) после ее интерпретации браузером.

Рис. 8.3. Отображение сообщения, сформированного элементом проверки

В результате сравнения значений, введенных в TextBox-элементы, элемент проверки (CompareValidator1) сформировал сообщение.

8.2. Задания

Задание 18. Применение серверных элементов управления проверкой достоверности:

1) разработайте приложение, обеспечивающее ввод и контроль данных с помощью элементов проверки, описанных в таблице;

Вводимые данные

Данные	Проверка вводимых данных
Фамилия	Обязательный ввод, только буквы русского языка
Имя	Обязательный ввод, только буквы русского языка
Отчество	Обязательный ввод, только буквы русского языка
Дата рождения	Обязательный ввод, дата, не превышающая текущую дату
Адрес e-mail	В соответствии с правилами записи e-mail-адресов
Пароль	Обязательный ввод, неотображаемый ввод, не меньше семи символов, символы не должны повторяться

2) примените элемент ValidationSummary для формирования протокола ошибок ввода;

3) текст программы включите в отчет по контрольной работе.

Задание 19. Контрольные вопросы:

1) поясните понятие «элемент управления проверкой достоверности ASP.NET»;

2) перечислите все известные вам элементы управления проверкой достоверности и поясните их назначение;

3) назовите общий базовый класс для всех элементов управления проверкой достоверности ASP.NET;

4) для чего применяются группы проверки достоверности?

Практическая работа № 9
ПРИМЕНЕНИЕ СЕРВЕРНЫХ ЭЛЕМЕНТОВ УПРАВЛЕНИЯ AJAX ASP.NET

9.1. Теоретические сведения

9.1.1. Серверные элементы управления AJAX

AJAX (Asynchronous JavaScript and XML) – методология построения интерфейса web-приложения, позволяющая асинхронно выполнять запросы к серверу, получать и обрабатывать ответы. В результате, при получении данных от сервера web-страница не перегружается полностью, а обновляется только ее часть. Такой подход позволяет с одной стороны ускорить выполнение запроса (за счет снижения объема пересылаемых данных), с другой – улучшить внешний вид интерфейса.

Методология AJAX основывается на возможностях объекта браузера XMLHTTPRequest (позволяет формировать асинхронные запросы и обрабатывать ответы), на стандарте XML и формате JSON (форматирование пересылаемых данных), модели DOM (программный интерфейс для доступа к содержимому HTML, XHTML и XML-документов).

В панели элементов Visual Studio элементы AJAX находятся в разделе «AJAX-расширение» (рис. 9.1).

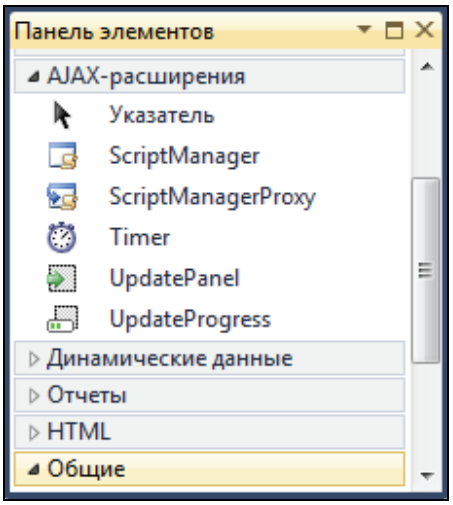

Рис. 9.1. Раздел «AJAX-расширение» панели элементов Visual Studio

Элемент ScriptManager является вспомогательным, но обязателен на aspx-странице при применении других AJAX-элементов. ScriptMa-

nager не генерирует HTML-код и поэтому не имеет визуального представления. Основное его назначение – формирование ссылок на JavaScript-библиотеки AJAX ASP.NET.

Элемент ScriptManagerProxy применяется в тех случаях, когда элемент ScriptManager располагается на мастер-странице.

Элемент UpdatePanel позволяет организовать частичное обновление aspx-страницы на основании обработки ответа асинхронного запроса.

Элемент UpdateProgress работает в сочетании с частичной визуализацией осуществляемой UpdatePanel, и предназначен для отображения сообщения и/или изображения в процессе длительного выполнения асинхронного запроса.

Элемент Timer функционально аналогичен UpdatePanel, но выполнение асинхронного запроса осуществляется автоматически в результате срабатывания таймера.

9.1.2. Применение серверных элементов управления AJAX

На рис. 9.2 представлена aspx-страница, использующая серверный элемент UpdatePanel.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
        Inherits="WebApplication9.WebForm1" %>
<!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></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
         <ContentTemplate>

            <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
            <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
            <asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
            <asp:Button ID="Button1" runat="server" Text="+" onclick="Button1_Click" />

         </ContentTemplate>
        </asp:UpdatePanel>
    </div>
    </form>
</body>
</html>
```

Рис. 9.2. Aspx-старница, использующая серверный элемент UpdatePanel

Тело тега UpdatePanel содержит три элемента типа TextBox и элемент типа Button. В окне браузера страница будет иметь примерно такой вид, как на рис. 9.3.

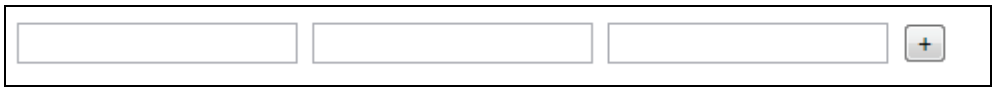

Рис. 9.3. Отображение aspx-страницы (рис. 9.2) в окне браузера

Нажатие кнопки приведет к асинхронной оправке данных на сервер и выполнению кода обработчика (рис. 9.4), а также к обновлению области aspx-страницы, заключенной между начальным и конечным тегами элемента UpdatePanel.

```
public partial class WebForm1 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {}

    protected void Button1_Click(object sender, EventArgs e)
    {

        int x = 0, y = 0;
        if (Int32.TryParse(this.TextBox1.Text, out x) && Int32.TryParse(this.TextBox2.Text, out y))
        {
            this.TextBox3.Text = (x + y).ToString();
        }
    }
}
```

Рис. 9.4. Код обработчика Click-события кнопки (рис. 9.3)

9.2. Задания

Задание 20. Применение серверных элементов управления AJAX:

1) разработайте приложение, обеспечивающее ввод двух числовых значений с помощью элементов TextBox и вывод произведения этих значений с помощью элемента Label; вычисление и отображение результата должно выполняться автоматически (обратите внимание: кнопка на форму не выводится) каждые 5 секунд; используйте AJAX-элемент Timer;

2) текст программы включите в отчет по контрольной работе.

Задание 21. Контрольные вопросы:

1) поясните понятия «методология AJAX», «объект XMLHTTPRequest», «язык XML», «формат JSON», «модель DOM», «асинхронный запрос»;

2) поясните, какой эффект в web-приложении достигается с помощью элементов AJAX;

3) перечислите все известные вам элементы управления AJAX и поясните их назначение.

Практическая работа № 10
КЭШИРОВАНИЕ СТРАНИЦ ASP.NET

10.1. Теоретические сведения

10.1.1. Кэширование aspx-страниц

Важнейшей задачей, стоящей перед разработчиком web-приложения, является обеспечение приемлемого времени ответа на запросы клиента. Одним из слагаемых величины времени ответа является промежуток, затрачиваемый на подготовку данных, отображаемых на web-странице. Этот промежуток может складываться из математических вычислений, выполнения запроса к базе данных и других слагаемых.

С другой стороны, высокий уровень актуальности отображаемой на странице информации требуется не так часто. Другими словами, не всегда для пользователя web-приложения важно: когда на странице вычислены данные: в текущий момент или пятью секундами раньше. В таких случаях однажды сформированная страница может быть сохранена в оперативной памяти, и если другой запрос вызывает ту же страницу, она может быть извлечена и отправлена клиенту. Через некоторое заданное время сохраняемая страница считается устаревшей и при очередном запросе она снова формируется и сохраняется. Таким образом может быть снижено время, необходимое на ответ клиенту и в большинстве случаев можно сэкономить вычислительный ресурс. Процесс временного хранения, извлечения и обновления страниц называется кэшированием вывода, а специальный программный объект ASP.NET, методы которого позволяют выполнять эти операции, называют кэшем вывода.

Кэш вывода представляет собой ассоциативную память. Он позволяет запоминать страницу, связав ее с некоторым ключевым значением, а также извлекать или обновлять страницу по ключу. В зависимости от того, каким образом формируется значение ключа для сохраняемой страницы, различают несколько видов кэширования: кэширование страницы, кэширование по параметрам, кэширование по заголовкам, пользовательское кэширование.

При простом кэшировании страницы, ключом является URL страницы.

Ключ кэширования по параметрам состоит из URL и значений определенных параметров запроса.

Кэширование по заголовкам использует ключ, состоящий из URL и значений определенных заголовков запроса.

В случае пользовательского кэширования ключ формируется функцией пользователя, которая должна быть размещена в файле Global.asax.

10.1.2. Применение кэширования aspx-страниц

На рис. 10.1 представлена aspx-страница, для которой применяется кэширование по параметру, а на рис. 10.2 пример ее отображения в окне браузера. Следует обратить внимание на директиву OutputCache aspx-страницы. С ее помощью задаются параметры кэширования страницы: интервал устаревания кэша (параметр Duration) и имя параметра (VaryByParam), значения которого являются составной частью ключа кэширования.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebCache.aspx.cs"
        Inherits="Cache.WebCache" %>
<%@ OutputCache Duration = "10" VaryByParam="parm1"  %>
<!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> Cache/Substitution </title></head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:Literal ID="Literal1"  Text = "Cache DateTime.Now"  runat="server"/>
        <asp:Label ID="Label1" runat="server" Text=""></asp:Label>
        <br />
        <asp:Button ID="Button1" runat="server" Text="Button" onclick="Button1_Click" />
        <asp:Literal ID="Literal2"  Text = "Substitution DateTime.Now"  runat="server"/>
        <asp:Substitution ID="Substitution1" runat="server"  MethodName="GetCurDateTime" />
    </div>
    </form>
</body>
</html>
```

Рис. 10.1. Кэширование aspx-страницы

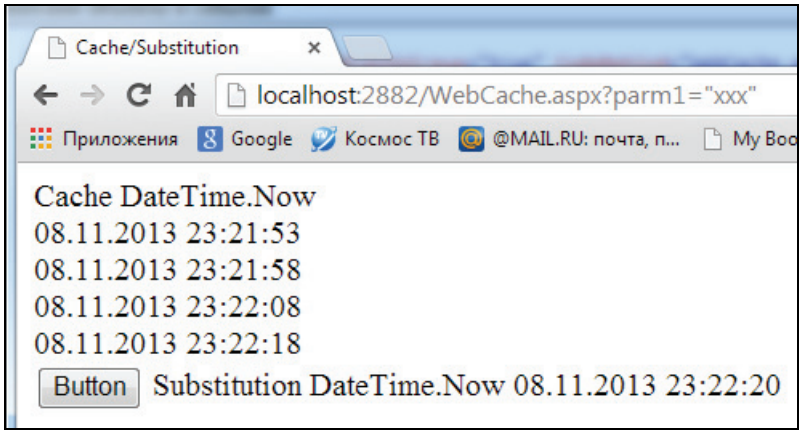

Рис. 10.2. Отображение браузером aspx-страницы, представленной на рис. 10.1

Кроме того, на aspx-странице (рис. 10.1) применен элемент Substitution, позволяющий вычислить значение (имя функции, указанное значением атрибута MethodName), заменяющее элемент Substitution и не участвующее в процессе кэширования. Часто такое использование элемента Substitution называют послекэшевой подстановкой.

Результат, полученный на рис. 10.2 отображает результат, который может быть получен при многократном нажатии кнопки с частотой примерно два раза в секунду. При этом в столбце, озаглавленном «Cache DateTime.Now» строки добавляются по одной каждые 10 секунд (значение Duration), а в строке, подписанной «Substitution DateTime.Now», каждую секунду (элемент Substitution).

На рис. 10.3 представлен код обработчиков событий aspx-страницы (рис. 10.1). Обратите внимание, что код в методе Page_Load выполняется при каждом нажатии кнопки и если бы не применялось кэширование, строки в столбце «Cache DateTime.Now» (рис. 10.2) добавлялись бы каждую секунду. При этом функция GetCurDate будет выполняться при каждом нажатии кнопки (элемент Substitution) и результат ее выполнения отображаться сразу, без задержки.

```
public partial class WebCache : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        this.Label1.Text +=  ("<br />" + DateTime.Now.ToString());
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
    }
    private  static  string GetCurDateTime(HttpContext ctx)
    {
        return DateTime.Now.ToString();
    }
}
```

Рис. 10.3. Код обработчиков событий aspx-страницы, представленной на рис. 10.1

10.2. Задания

Задание 22. Применение кэширования aspx-страниц:

1) разработайте приложение, применяющее кэширование на 5 секунд aspx-страницы по параметру; с помощью вывода в окно браузера, продемонстрируйте, что кэширование действительно выполняется;

2) добавьте в aspx-страницу разработанного приложения элемент Substitution, применение которого демонстрировало бы обновление части aspx-страницы без задержек, связанных с кэшированием;

3) текст кода программы включите в отчет по контрольной работе.

Задание 23. Контрольные вопросы:

1) поясните понятие «кэширование вывода»;

2) поясните принцип реализации кэширования вывода в ASP.NET;

3) поясните, в каких случаях целесообразно применять кэширование вывода

4) перечислите все типы кэширования вывода в ASP.NET поясните принцип их реализации;

5) поясните термин «послекэшевая подстановка».

ЛИТЕРАТУРА

1. C# / Х. Дейтл [и др.]. – СПб.: БХВ-Петербург, 2006. – 1056 с.
2. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Дж. Рихтер. – СПб.: Питер, 2013. – 896 с.
3. Пауэлл, Т. Полный справочник по JavaScript / Т. Пауэлл, Ф. Шнайдер. – М.: Вильямс, 2006. – 960 с.
4. HTML, XHTML и CSS. Библия пользователя / Б. Пфаффенбергер [и др.]. – М.: Вильямс, 2007. – 752 с.
5. Хабибуллин, И. Ш. Самоучитель XML / И. Ш. Хабибуллин. – СПб.: БХВ-Петербург, 2006. – 336 с.
6. Мак-Дональд, М. Microsoft ASP.NET 3.5 с примерами на C# 2008 и Silverlight 2 / М. Мак-Дональд, М. Шпурта. – М.: Вильямс, 2009. – 1408 с.
7. Адамс, К. Администрирование сервера IIS 7 / К. Адамс. – М.: ООО «Бином», 2010. – 364 с.
8. Смелов, В. В. Основы web-программирования на Java / В. В. Смелов. – Минск: БГТУ, 2009. – 140 с.
9. Мазуркевич, А. PHP: настольная книга программиста / А. Мазуркевич, Д. Еловой. – М.: Новое знание, 2006. – 495 с.
10. WCF 4: Windows Communication Foundation и .NET 4 для профессионалов / П. Сибраро [и др.]. – М.: Вильямс, 2011. – 464 с.
11. Чедвик, Д. ASP.NET MVC 4. Разработка реальных веб-приложений с помощью ASP.NET MVC/ Д. Чедвик, П. Хришикен, Т. Снайдер. – М.: Вильямс, 2013. – 432 с.

ОГЛАВЛЕНИЕ

ПРОГРАММИРОВАНИЕ ИНТЕРНЕТ-ИЗДАНИЙ

Составитель
Смелов Владимир Владиславович

Методические указания к выполнению контрольной работы

Редактор *К. В. Великода*
Компьютерная верстка *К. В. Великода*
Корректор *К. В. Великода*

Издатель:
УО «Белорусский государственный технологический университет».
Свидетельство о государственной регистрации издателя, изготовителя
распространителя печатных изданий
№ 1/227 от 20.03.2014.
Ул. Свердлова, 13а, 220006, г. Минск.