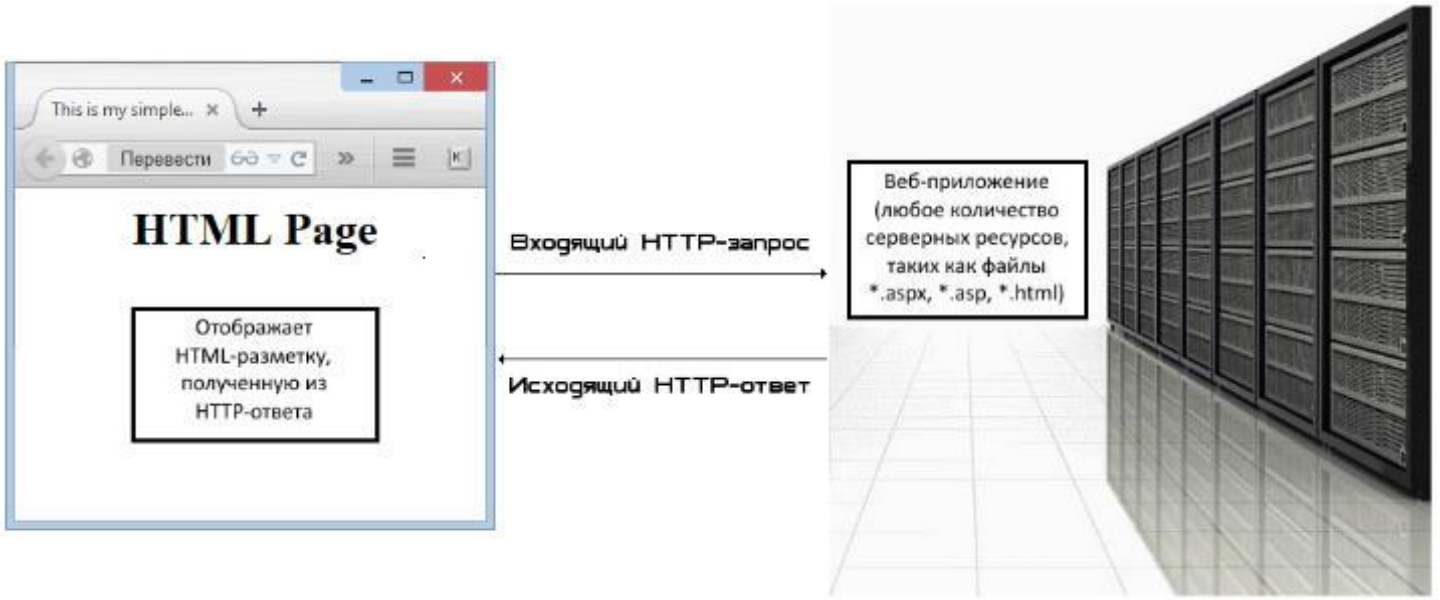
Протокол Http

Профессиональное веб-приложение всегда подразумевает существование как минимум двух машин, объединенных в сеть: на одной развернут веб-сайт, на другой просматриваются данные с помощью веб-браузера.

Сетевой протокол, соединяющий такие компьютеры, называется **HTTP** (Hypertext Transfer Protocol – протокол передачи гипертекста)

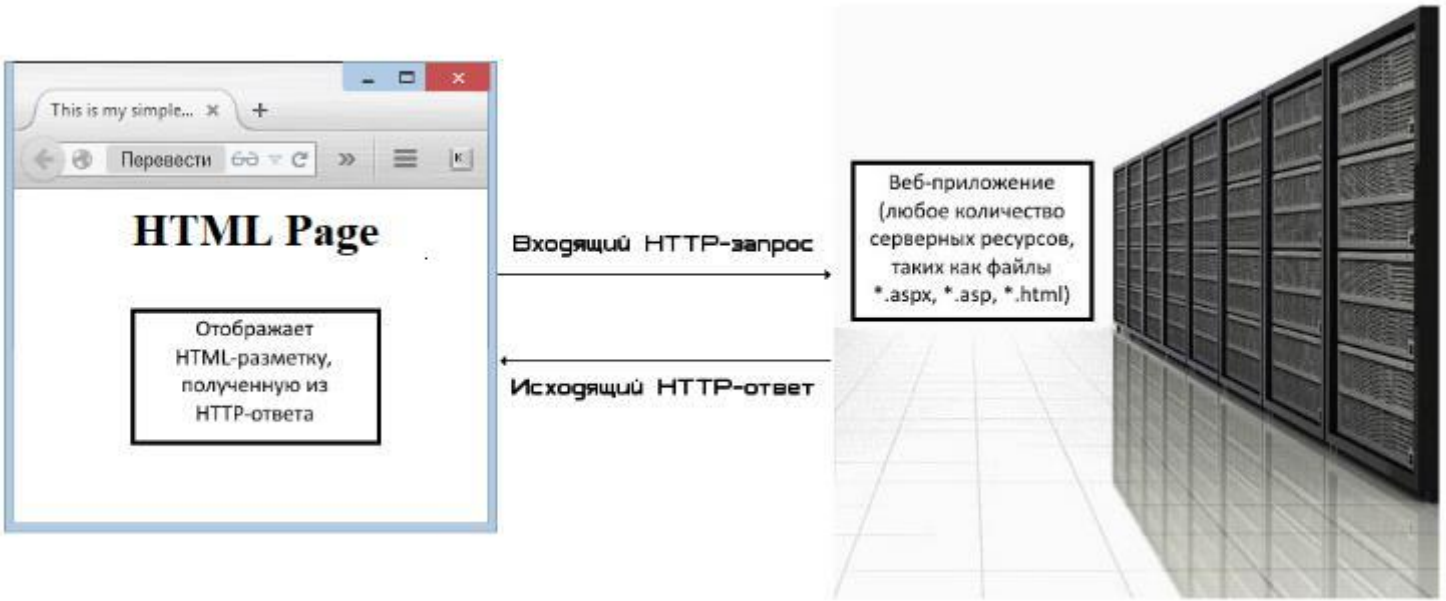


- Веб-сервер принимает входящий запрос HTTP и может обработать любые отправленные клиентом входные значения, формирующие нужный HTTP-ответ.
- Веб-программисты могут использовать любое количество технологий (PHP, ASP.NET, JSP и т.п.) для динамической генерации содержимого, посылаемого в HTTP-ответ.
- Браузер клиентской стороны отображает на экране HTML-разметку, отправленную веб-сервером.

Протокол Http

HTTP представляет собой сетевой протокол без хранения состояния

- После отправки с веб-сервера ответа клиентскому браузеру все их предыдущее взаимодействие забывается.
- На веб-разработчика возлагается ответственность за реализацию запоминания информации, связанной с пользователями, которые в данный момент зашли на сайт.
- ASP.NET предлагает многочисленные способы поддержки состояния: переменные сеанса, cookie-наборы и кэш приложения, а также API-интерфейс управления профилями ASP.NET.

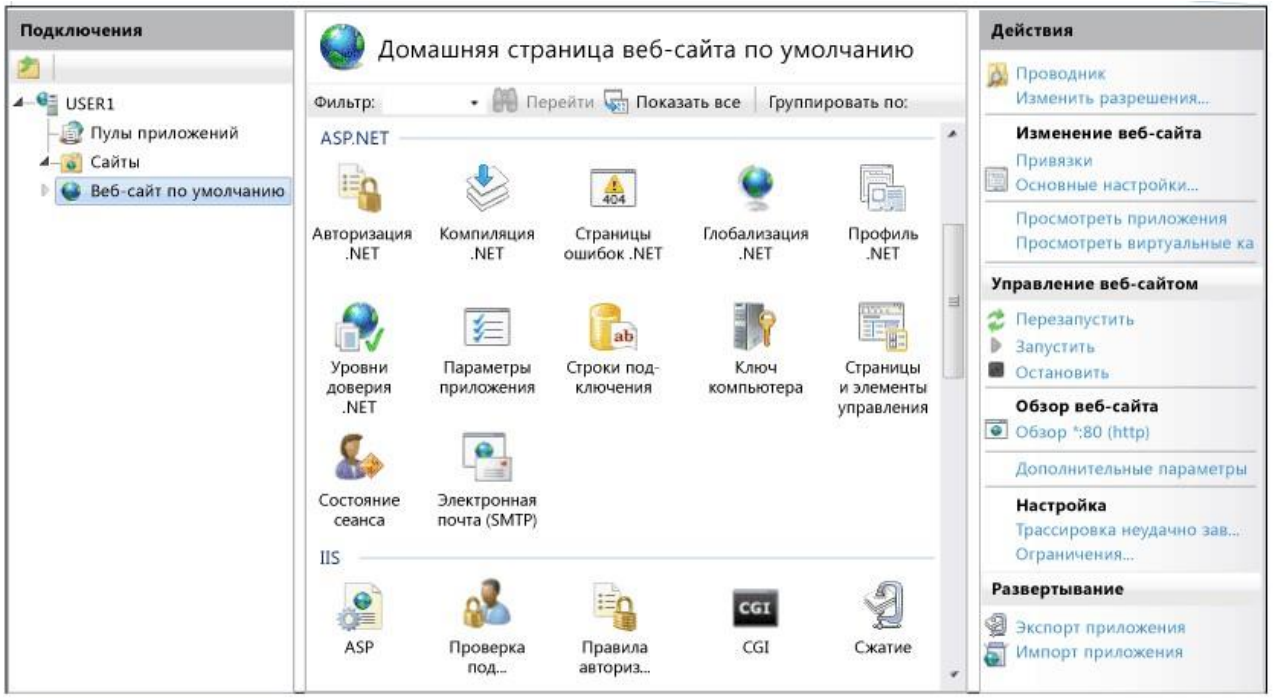


Веб-приложения и веб-серверы

Веб-приложение – это коллекция файлов (*.htm, *.aspx, файлов изображений, файлов данных XML и т.п.), и связанных с ними компонентов (таких как библиотека кода .NET), которые хранятся в определенном наборе каталогов на веб-сервере.

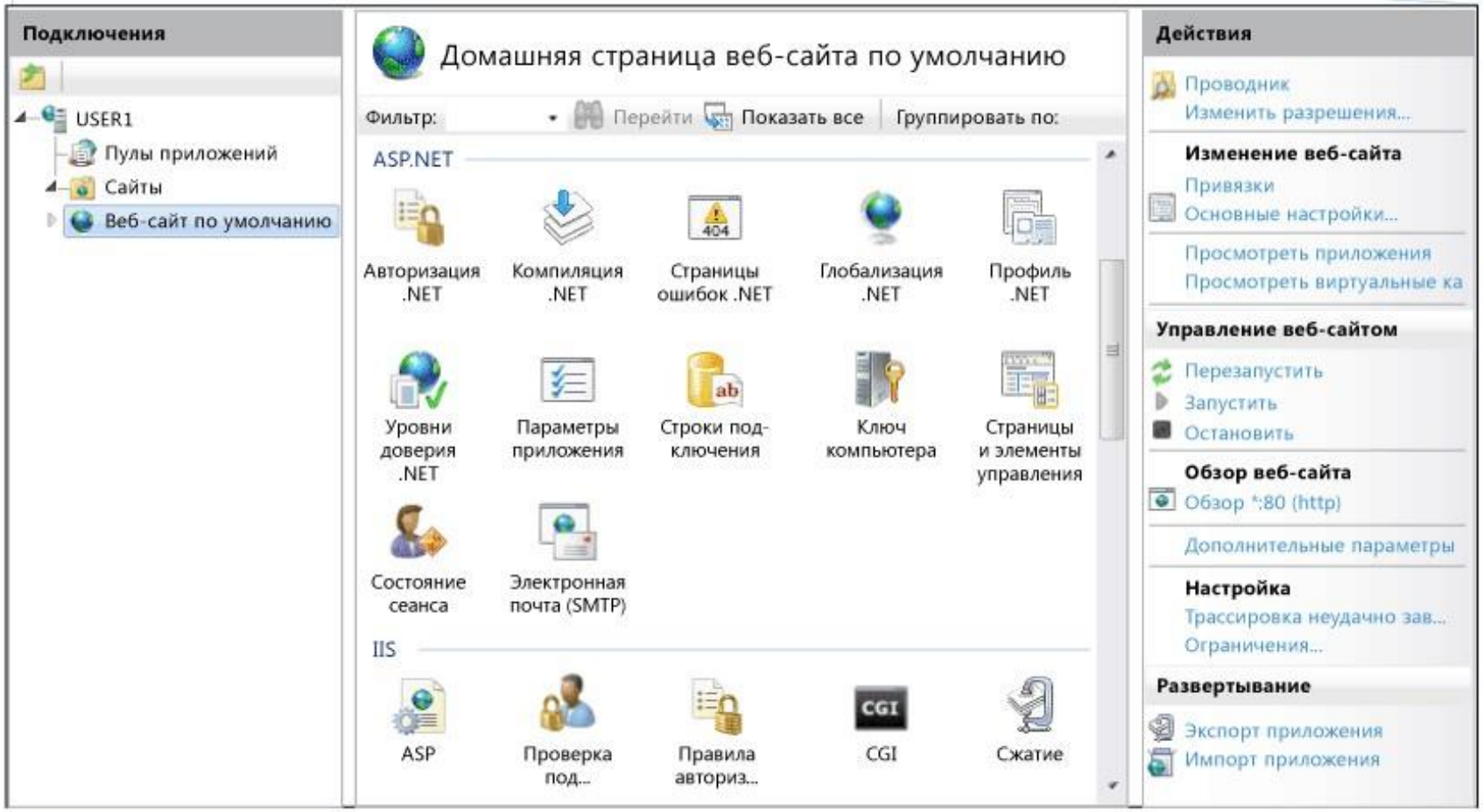
Веб-сервер – это программный продукт, отвечающий за размещение веб-приложений.

Службы **Internet Information Services (IIS)** – это веб-серверный продукт производственного уровня от Microsoft, обладающий встроенной поддержкой веб-приложений классического ASP и ASP.NET.

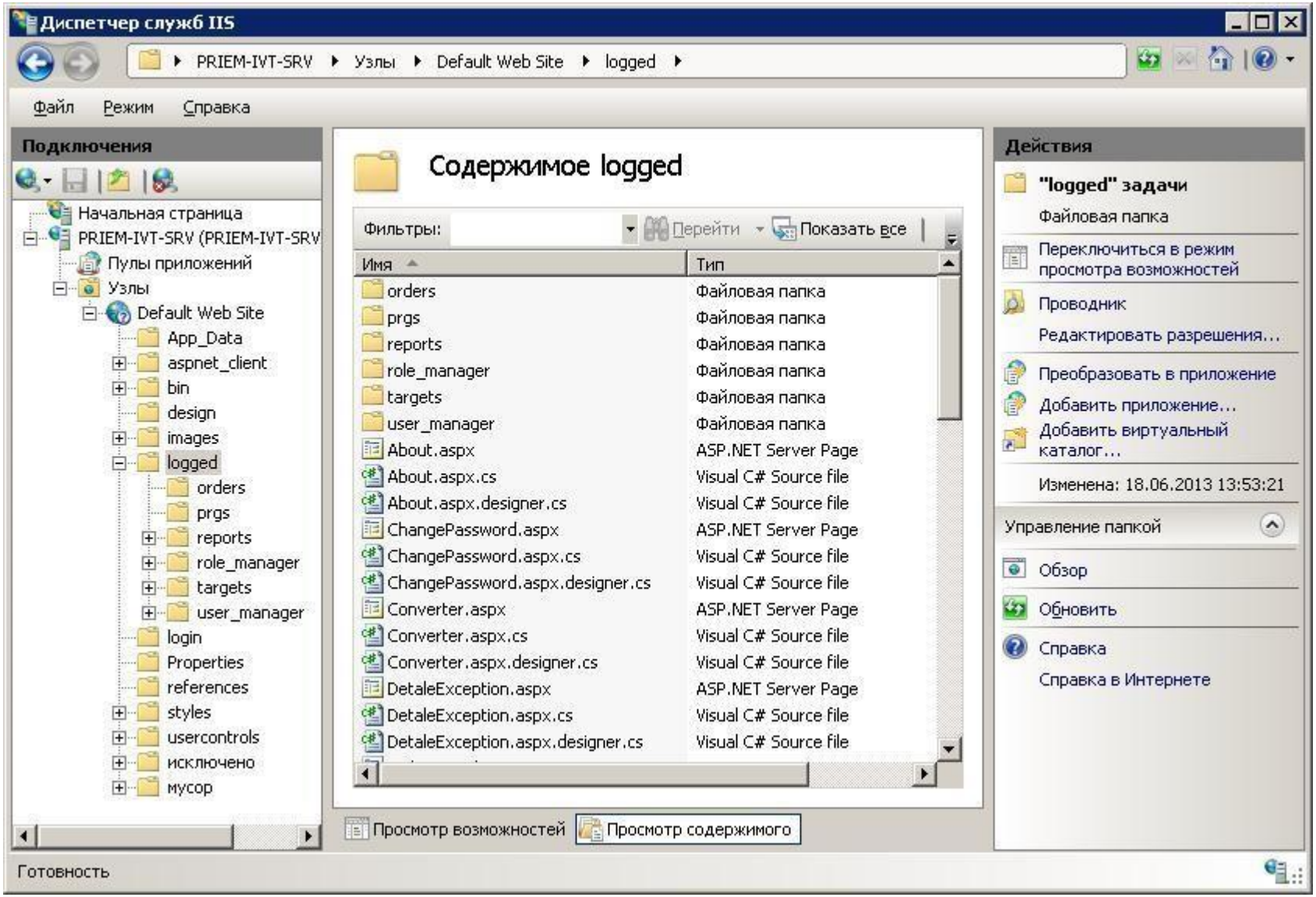


Веб-сервер

- Единственная установленная копия IIS может размещать многочисленные веб-приложения, каждое из которых располагается в своем *виртуальном каталоге*.
- Каждый виртуальный каталог отображается на физический каталог на жестком диске.

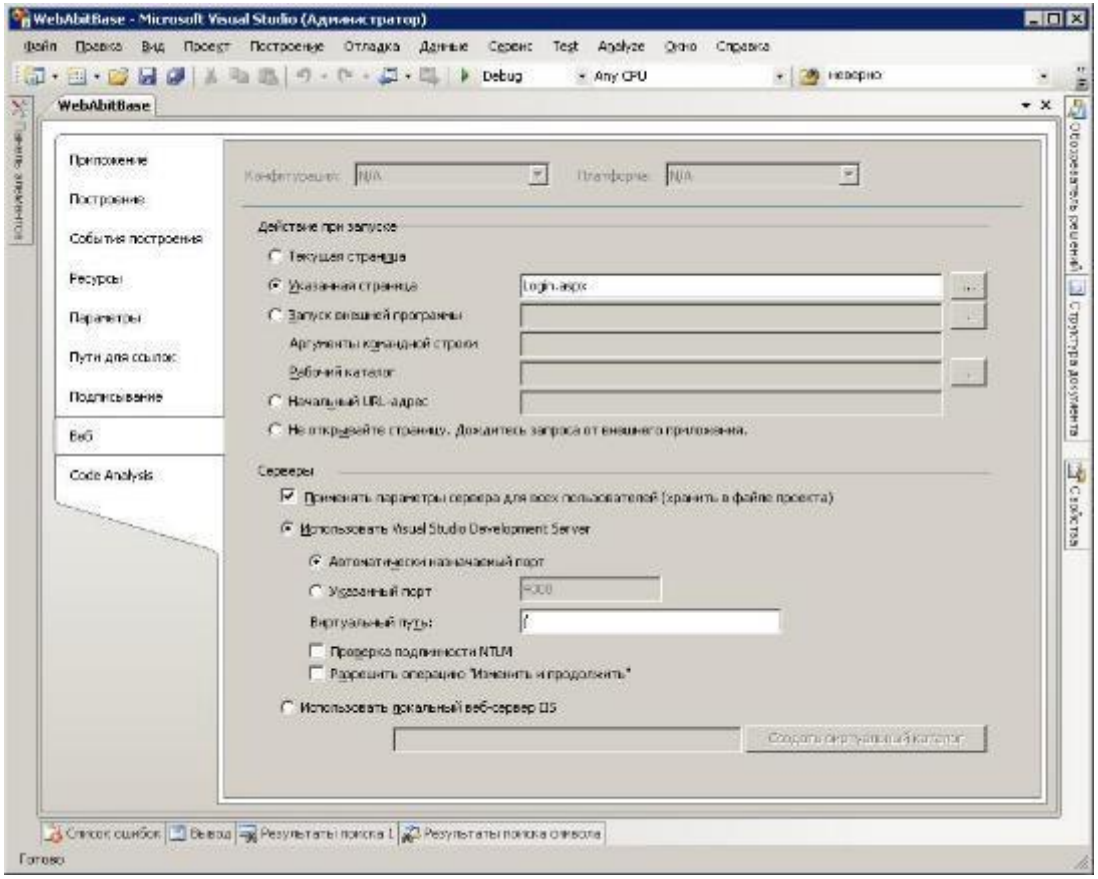


Система ЕИС «Абитуриент»



Веб-сервер разработки ASP.NET

Во время разработки и тестирования веб-приложений у разработчиков есть возможность использовать легковесный веб-сервер, который называется ASP.NET Development Web Server.



Эта утилита позволяет разработчикам размещать веб-приложения ASP.NET за пределами IIS.

Этот веб-сервер предназначен только для целей разработки и тестирования и не предназначен для размещения веб-приложений производственного уровня. Как только веб-приложение готово в первом приближении, сайт должен быть скопирован в виртуальный каталог IIS.

Язык HTML

HTML (Hypertext Markup Language) – стандартный язык разметки, используемый для описания того, как литеральный текст, изображения, внешние ссылки и разнообразные элементы управления HTML отображаются внутри браузера на стороне клиента.

```
<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="utf-8" />
    <title></title>
</head>
<body>

</body>
</html>
```

Файл HTML состоит из набора дескрипторов, описывающих внешний вид и поведение веб-страницы.

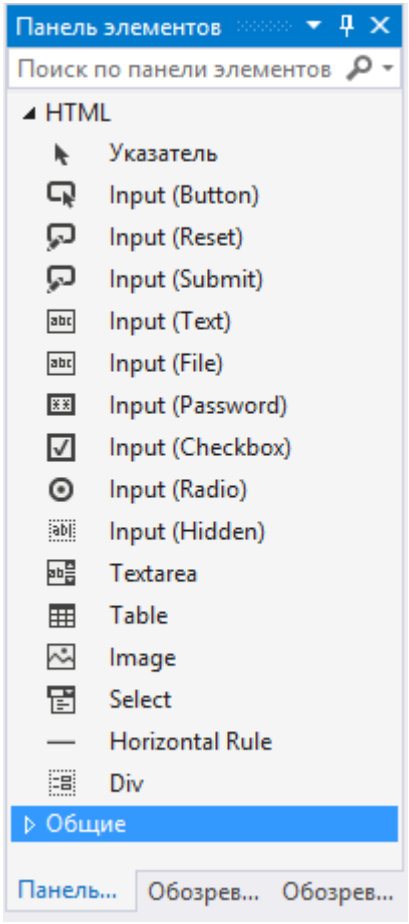
Базовая структура HTML-документа, как правило, не изменяется.

Реальная функциональность большинства файлов *.html находится в области элементов **<form>**.

Форма HTML – это именованная группа взаимосвязанных элементов пользовательского интерфейса, обычно служащих для ввода пользователем каких-нибудь данных.

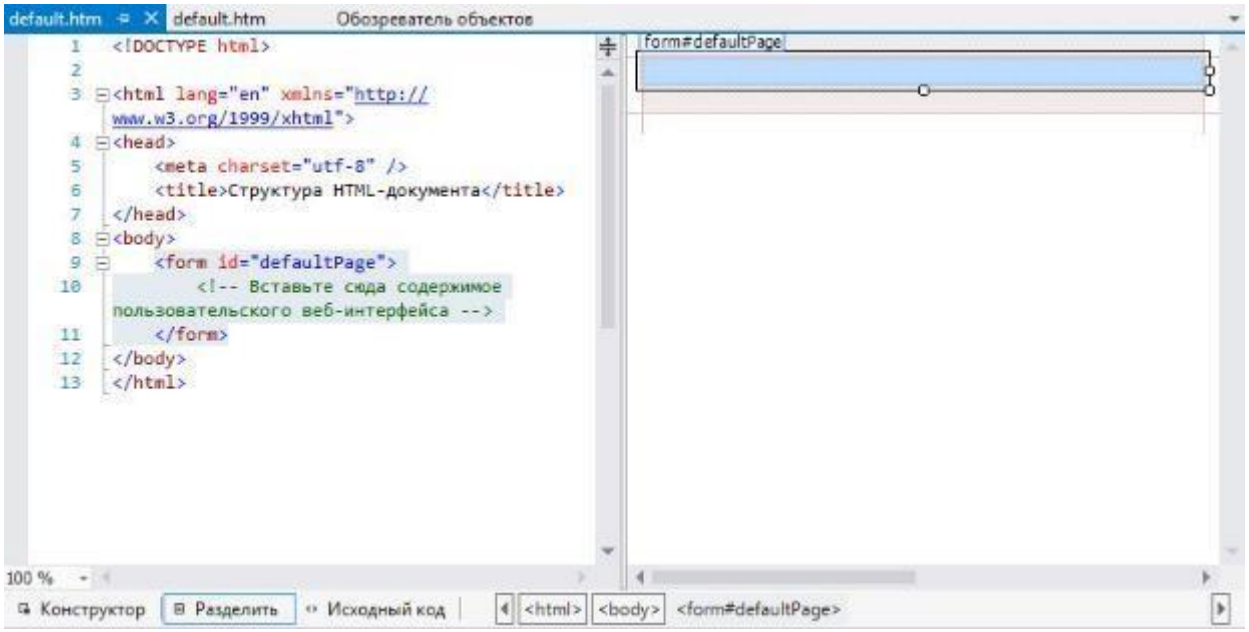
```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>This is my simple web page</title>
</head>
<body>
    <form id="defaultPage">
        <!-- Вставьте сюда содержимое пользовательского веб-интерфейса -->
    </form>
</body>
</html>
```

Среда Visual Studio



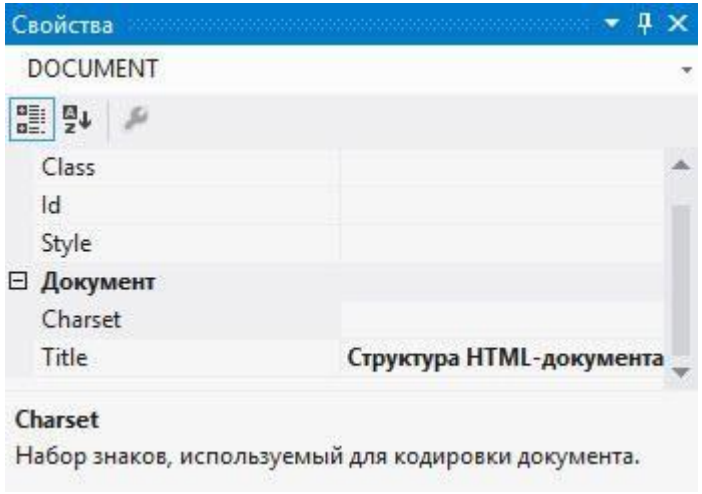
В панели инструментов среды Visual Studio имеется вкладка HTML, которая позволяет выбирать любой элемент управления HTML для помещения на поверхность визуального конструктора HTML.

Элементы управления HTML могут перетаскиваться на поверхность визуального конструктора HTML или непосредственно в HTML-разметку.



Среда Visual Studio

Среда Visual Studio позволяет редактировать внешний вид и поведение файла *.htm или определенного элемента управления HTML в <form> с использованием окна Свойства.



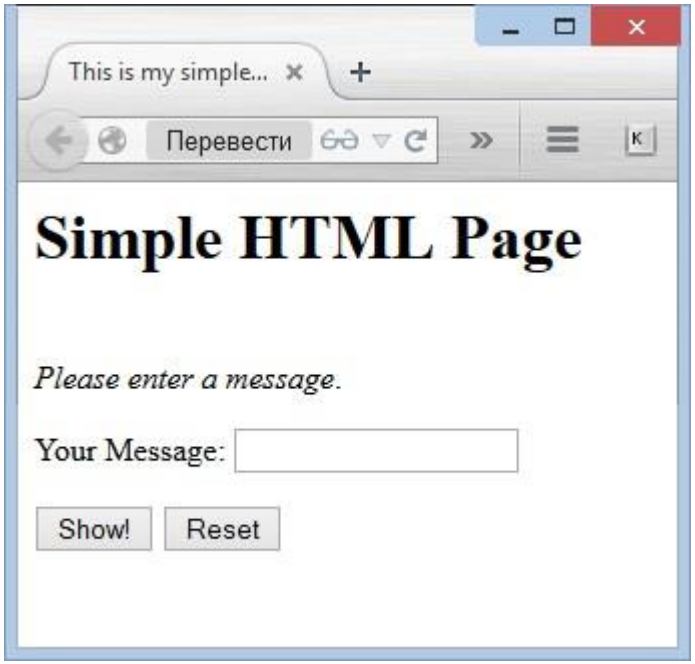
При использовании окна Properties для конфигурирования какого-то аспекта веб-страницы IDE-среда соответствующим образом изменяет HTML-разметку.

Построение формы HTML

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>This is my simple web page</title>
</head>
<body>
    <!-- Приглашение пользовательского ввода -->
    <h1>Simple HTML Page</h1>
    <p>
        <br />
        <i>Please enter a message</i>.
    </p>
    <!-- Форма для получения информации о пользователе -->
    <form id="defaultPage">
        <p>
            Your Message:
            <input id="txtUserMessage" type="text" /></p>
        <p>
            <input id="btnShow" type="button" value="Show!" />
            <input id="btnReset" type="reset" value="Reset" />
        </p>
    </form>
</body>
</html>
```

- Каждому элементу управления назначены подходящий идентификатор.
- Каждый элемент ввода имеет дополнительный атрибут type, который указывает, что делает элемент управления.

Создаваемый пользовательский интерфейс будет содержать одно текстовое поле и две кнопки. Первая кнопка будет служить для запуска сценария клиентской стороны, а другая – для сброса входных полей формы к своим значениям по умолчанию.



Сценарии клиентской стороны

Файл *.html может содержать блоки кода сценариев, которые будут обработаны запрашивающим браузером.

Причины использования сценариев на стороне клиента

- проверка достоверности пользовательского ввода перед отправкой данных обратно веб-серверу;
- взаимодействие с объектной моделью документов (Document Object Model – DOM) браузера.

Если обнаруживается ошибка (вроде отсутствующих данных в обязательном поле), можно уведомить пользователя об этом, не выполняя обратную отправку веб-серверу.

Когда браузер производит синтаксический анализ HTML-страницы, он строит в памяти дерево объектов, представляющих все содержимое веб-страницы (формы, элементы ввода, и т.п.).
Браузеры предоставляют API-интерфейс под названием DOM, который открывает доступ к дереву объектов и позволяет программно изменять его содержимое. Например, можно написать код JavaScript, который выполняется в браузере для получения значений определенных элементов управления, изменения цвета элемента, динамического добавления новых элементов на страницу и т.д.

Сценарии клиентской стороны

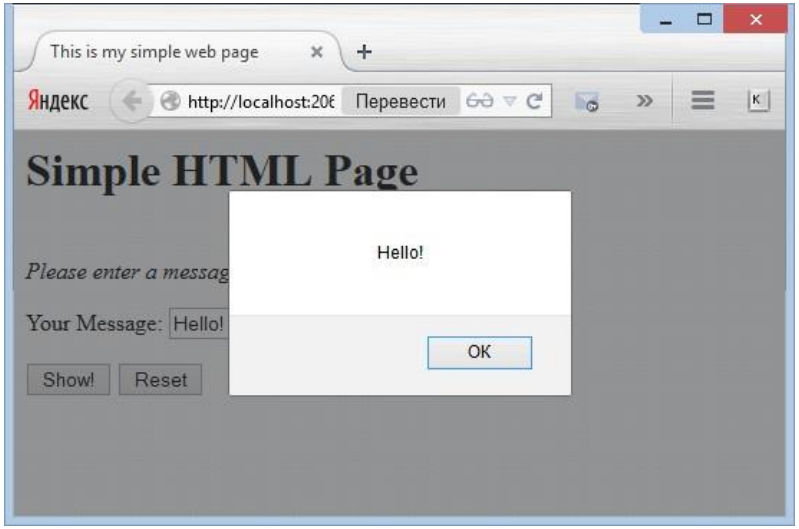
Для перехвата события click кнопки «Show!» добавим в определение виджета btnShow атрибут onclick и укажем в нем JavaScript-метод по имени btnShow_onclick()

```
<input id="btnShow" type="button" value="Show!" onclick="return btnShow_onclick()" />
```

Добавим сразу после открывающего элемента <head> код функции JavaScript, которая будет вызвана при щелчке пользователя на кнопке.

```
<script lang="javascript" type="text/javascript">
    // <![CDATA[
    function btnShow_onclick() {
        alert(txtUserMessage.value);
    }
    // ]]>
</script>
```

Блок сценария помещен в раздел CDATA чтобы в случае, если страница попадет в браузер, не поддерживающий JavaScript, этот код воспринимался как блок комментария и был проигнорирован.



Обратная отправка веб-серверу

Реальная веб-страница нуждается в обратной отправке ресурса на веб-сервер, одновременно передавая все введенные данные. Получив эти данные, ресурс серверной стороны может использовать их для динамического построения соответствующего ответа HTTP.

```
<form id="defaultPage"
      action="http://localhost/Cars/ClassicAspPage.asp" method="GET">
    <input id="btnPostBack" type="submit" value="Post to Server!" />
    ...
</form>
```

Атрибут action в открывающем дескрипторе <form> указывает получателя входных данных формы. К возможным получателям относятся почтовые серверы, другие HTML-файлы на веб-сервере, классические файлы Active Server Pages (ASP) (на основе COM), веб-страницы ASP.NET и т.д.

Если указан method="GET" в качестве режима передачи, данные формы присоединяются к строке запроса в виде набора пар "имя/значение", разделенных амперсандами.

```
http://www.google.com/search?hl=en&source=hp&q=vikings&cts=12643
70773666&aq= f&aql=&aqi=g1g-z1g1g-z1g1g-z1g4&oq=
```

При использовании метода POST данные формы не становятся непосредственно видимыми внешнему миру. Данные POST не имеют ограничения по длине строки.

```
<form id="defaultPage"
      action="http://localhost/Cars/ClassicAspPage.asp" method="POST">
</form>
```

Однофайловая веб-страница ASP.NET

В рамках модели однофайловой страницы код серверной стороны помещается внутрь области <script>, но сам код не является сценарием (например, на JavaScript).
Вместо этого код внутри блока <script> пишется на выбранном языке .NET (C#, Visual Basic и т.д.)

Достоинства

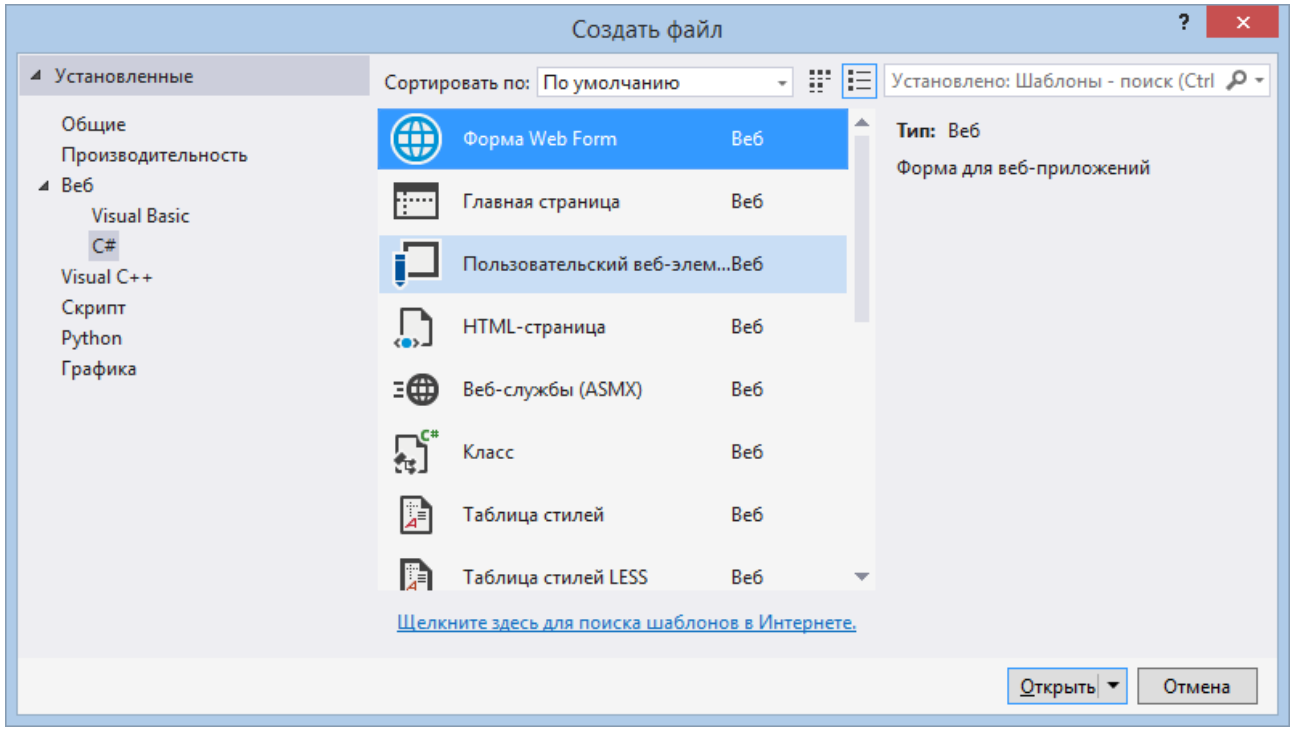
- Страницы, написанные в соответствии с однофайловой моделью, несколько легче развертывать или отправлять другим разработчикам.
- Такую страницу проще переименовывать, т.к. нет зависимости между несколькими файлами.
- Упрощается управление файлами в системе управления исходным кодом, поскольку все действия выполняются с единственным файлом.

Недостатки

- Единственный файл делает слишком много (определение разметки пользовательского интерфейса и программной логики в одном месте)

Однофайловая веб-страница ASP.NET

Для создания однофайловой веб-страницы необходимо выбрать пункт меню Файл\Создать\Файл, далее в диалоговом окне выбрать «Форма Web Form»



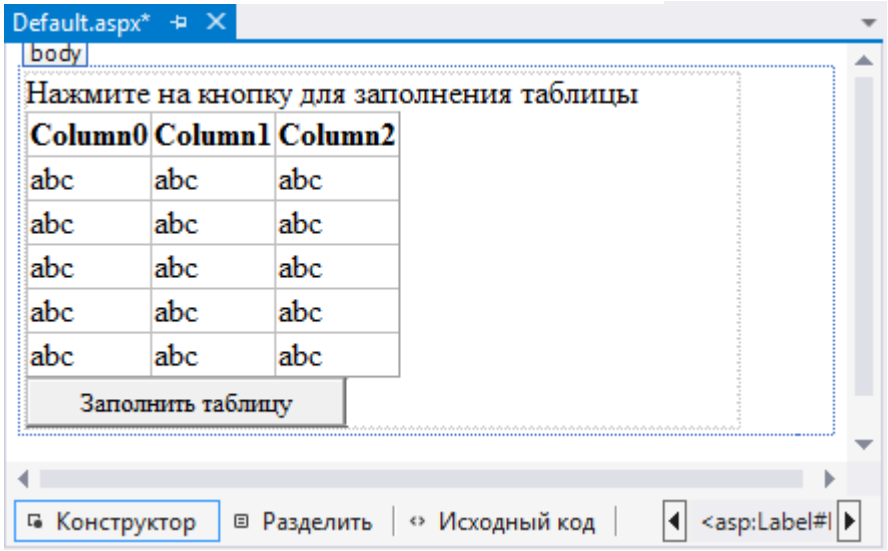
Однофайловая веб-страница ASP.NET

Веб-элементы управления – это объекты, обрабатываемые на веб-сервере, который автоматически возвращает их HTML-представление в исходящем HTTP-ответе.

Каждый веб-элемент управления определен с использованием дескриптора <asp:>. После этого префикса дескриптора указывается имя веб-элемента управления ASP.NET (Label, GridView и Button). Перед закрывающим дескриптором заданного элемента находится серия пар "имя/значение", которые соответствуют настройкам, проведенным в окне Свойств.

```
<form id="form1" runat="server">
    <asp:Label ID="lblInfo" runat="server" Text="Нажмите на кнопку для заполнения таблицы"></asp:Label>
    <asp:GridView ID="carsGridView" runat="server">
    </asp:GridView>
    <asp:Button ID="btnFillData" runat="server" Text="Заполнить таблицу" />
</form>
```

Веб-элементы управления ASP.NET имитируют модель программирования настольных приложений, при которой имена свойств, методов и событий обычно совпадают с их аналогами из Windows Forms/WPF.



Однофайловая веб-страница ASP.NET

В определении Button добавляем обработчик события Click

```
<asp:Button ID="btnFillData" runat="server"
Text="Заполнить таблицу" OnClick="btnFillData_Click" />
```

В блоке <script> серверной стороны внутри файла *.aspx необходимо реализовать обработчик события Click. Его входные параметры в точности соответствуют целевому методу делегата System.EventHandler.

С помощью директивы <%@Import%> нужно указать, что будет использоваться пространство имен AutoLotConnectedLayer.

Нужно также информировать исполняющую среду ASP.NET о том, что эта однофайловая страница ссылается на сборку AutoLotDAL.dll, через директиву <%@Assembly%>.

```
<%@ Page Language="C#" %>
<%@ Import Namespace = "AutoLotConnectedLayer" %>
<%@ Assembly Name ="AutoLotDAL" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
     void btnFillData_Click(object sender, EventArgs args)
     {
          InventoryDAL dal = new InventoryDAL();
          dal.OpenConnection(@"Data Source=(local)\SQLEXPRESS;" +
              "Initial Catalog=AutoLot;Integrated Security=True");
          carsGridView.DataSource = dal.GetAllInventory();
          carsGridView.DataBind();
          dal.CloseConnection();
     }
</script>
<html xmlns="http://www.w3.org/1999/xhtml">
...
</html>
```

Директивы ASP.NET

Типичный файл *.aspx обычно открывается набором *директив*.
Директивы ASP.NET всегда помечаются маркерами **<%@ ... %>** и могут быть квалифицированы различными атрибутами, информирующими исполняющую среду ASP.NET о том, как следует обрабатывать конкретную директиву.

Каждый файл *.aspx должен иметь, как минимум, директиву **<%@Page%>**, которая служит для определения управляемого языка, применяемого внутри страницы (в атрибуте language).

Избранные атрибуты директивы <%@Page%>

Атрибут	Назначение
CodePage	Указывает имя связанного файла отделенного кода
EnableTheming	Указывает, поддерживают ли элементы управления страницы *.aspx темы ASP.NET
EnableViewState	Указывает, поддерживается ли состояние представления между запросами страницы
Inherits	Определяет класс в файле отделенного кода, от которого наследуется страница; может быть любым классом, производным от System.Web.UI.Page
MasterPageFile	Устанавливает мастер-страницу, используемую с текущей страницей *.aspx
Trace	Признак включения трассировки

В дополнение к директиве **<%@Page%>**, заданный файл *.aspx может задавать различные директивы **<%@Import%>** для явной установки пространств имен, необходимых текущей странице, и директив **<%@Assembly%>** для указания внешних библиотек кода, используемых сайтом (обычно размещаемых в папке \bin веб-сайта).

Однофайловая веб-страница ASP.NET

В однофайловой модели страницы *.aspx-файл может содержать логику сценариев серверной стороны, которая выполняется на веб-сервере.

Все веб-элементы управления ASP.NET должны иметь в своем открывающем объявлении атрибут **runat="server"**.
Если атрибут **runat="server"** не указан, веб-элементы не будут генерировать соответствующую HTML-разметку в исходящий HTTP-ответ.

```
<script runat="server">
    void btnFillData_Click(object sender, EventArgs args)
    {
        InventoryDAL dal = new InventoryDAL();
        dal.OpenConnection(@"Data Source=(local)\SQLEXPRESS;" +
            "Initial Catalog=AutoLot;Integrated Security=True");
        carsGridView.DataSource = dal.GetAllInventory();
        carsGridView.DataBind();
        dal.CloseConnection();
    }
</script
```

Однофайловая веб-страница ASP.NET

Веб-виджеты ASP.NET помещаются в область элементов **<form>**, который помечен атрибутом **runat="server"**.
Если элемент управления помечен префиксом **asp:**, это означает, что этот элемент является членом библиотеки элементов управления ASP.NET и имеет соответствующее представление в виде класса C# в определенном пространстве имен .NET библиотеки базовых классов .NET.

```
<form id="form1" runat="server">
<div>
  <asp:Label ID="lblInfo" runat="server"
    Text="Click on the Button to Fill the Grid">
  </asp:Label>
  <br />
  <br />
  <asp:GridView ID="carsGridView" runat="server">
    ...
  </asp:GridView>
  <br />
  <asp:Button ID="btnFillData" runat="server" Text="Fill Grid"
OnClick="btnFillData_Click"/>
</div>
</form>
```

Построение веб-страницы ASP.NET с использованием файлов кода

Техника отделенного кода позволяет разделить код программы и логику HTML-представления, используя для них два разных файла.

Преимущества модели отделенного кода

- Страницы с отделенным кодом обеспечивают четкое разделение HTML-разметки и кода, поэтому можно организовать параллельную работу дизайнеров над разметкой и программистов – над кодом C#.
- Код не предоставляется дизайнерам страниц или прочему персоналу, который имеет дело только с разметкой страницы.
- Файлы кода можно использовать с несколькими файлами *.aspx.

Отладка и трассировка страниц ASP.NET

При построении веб-проектов ASP.NET можно размещать точки останова в файле отделенного кода (а также во встроенных блоках <script> внутри файла *.aspx), запускать сеанс отладки и пошагово выполнять код.

Чтобы отлаживать веб-приложения ASP.NET, сайт должен содержать свойство, сконфигурированное в файле Web.config.

```
<compilation debug="true" targetFramework="4.5"/>
```

Можно включить поддержку трассировки для файла *.aspx, установив атрибут Trace в true внутри директивы <%@Page%>

```
<%@ Page Language="C#" AutoEventWireup="true"
    CodeFile="Default.aspx.cs" Inherits="_Default" Trace="true" %>
```

После этого генерируемая HTML-разметка будет содержать множество деталей о предыдущем запросе/ответе HTTP (серверные переменные, переменные сеанса и приложения, запросы/ответы и т.п.).

Чтобы вставить собственные сообщения трассировки, можно воспользоваться свойством Trace из типа System.Web.UI.Page.

```
protected void btnFillData_Click(object sender, EventArgs e)
{
    Trace.Write("CodeFileTraceInfo!", "Filling the grid!");
}
```

Веб-сайты ASP.NET

Новый проект веб-сайта ASP.NET может содержать некоторое количество специфически именованных подкаталогов, каждый из которых имеет определенное значение для исполняющей среды ASP.NET.

Подкаталог	Назначение
App_Browsers	Папка для файлов определений браузеров, используемых для идентификации индивидуальных браузеров и определения их возможностей
App_Code	Папка исходного кода для компонентов или классов, которые должны компилироваться как часть приложения. ASP.NET компилирует код в этой папке при запросе страницы. Код в папке App_Code автоматически доступен приложению
App_Data	Папка для хранения файлов Access *.mdb, файлов SQL Express *.mdf, файлов XML или других источников данных
App_GlobalResources	Папка для файлов *.resx, доступных программно из кода приложения
App_LocalResources	Папка для файлов *.resx, привязанных к определенной странице
App_Themes	Папка, содержащая коллекцию файлов, которые определяют внешний вид веб-страниц и элементов управления ASP.NET
App_WebReferences	Папка для прокси-классов, схем и прочих файлов, ассоциированных с использованием веб-служб в приложении
Bin	Папка для скомпилированных приватных сборок (файлов *.dll). Сборки из папки Bin автоматически доступны приложению

Чтобы добавить любой из этих известных подкаталогов к текущему веб-приложению, выберите пункт меню Веб-сайт\Добавить папку.
Во многих случаях IDE-среда автоматически сделает это, когда вы естественным образом добавляете соответствующие файлы к сайту.

Взаимодействие с входящим запросом HTTP

Базовый поток веб-приложения начинается с запроса веб-страницы, возможного ввода информации пользователя и щелчка на кнопке Submit для отправки данных HTML-формы заданной веб-странице на обработку.

В большинстве случаев в открывающем дескрипторе оператора form присутствуют атрибуты action и method, указывающие файл на веб-сервере, которому будут отправлены данные из различных HTML-виджетов, и метод отправки данных (GET или POST).

```
<form name="defaultPage" id="defaultPage"
    action="http://localhost/Cars/ClassicAspPage.asp"
    method="GET">
</form>
```

Взаимодействие с входящим запросом HTTP

Все страницы ASP.NET поддерживают свойство System.Web.UT.Page.Request, которое обеспечивает доступ к экземпляру класса HttpRequest.
Основные члены класса HttpRequest представлены в таблице.

Член	Назначение
ApplicationPath	Получает виртуальный корневой путь приложения ASP.NET на сервере
Browser	Предоставляет информацию о возможностях клиентского браузера
Cookies	Получает коллекцию cookie-наборов, присланную клиентским браузером
FilePath	Показывает виртуальный путь для текущего запроса
Form	Получает коллекцию переменных формы HTTP
Headers	Получает коллекцию заголовков HTTP
HttpMethod	Показывает метод передачи данных, используемый клиентом (GET/POST)
IsSecureConnection	Показывает, является ли соединение HTTP защищенным (т.е. HTTPS)
QueryString	Получает коллекцию переменных строки запроса HTTP
RawUrl	Получает "сырой" URL текущего запроса
RequestType	Показывает тип текущего запроса HTTP
ServerVariables	Получает коллекцию переменных веб-сервера
UserHostAddress	Получает IP-адрес хоста удаленного клиента
UserHostName	Получает имя хоста удаленного клиента

Взаимодействие с входящим запросом HTTP

В классе HttpResponse определены свойства **Form** и **QueryString**. Они позволяют исследовать входные данные формы, используя пары "имя/значение".

```
protected void btnGetFormData_Click (object sender, System.EventArgs e)
{
    // Получить значение виджета с идентификатором txtFirstName.
    string firstName = Request.Form("txtFirstName");
    // Использовать полученное значение на странице...
}
```

Можно просто опросить виджет серверной стороны непосредственно через свойство Text, чтобы использовать его в программе:

```
protected void btnGetFormData_Click (object sender, System.EventArgs e)
{
    // Получить значение виджета с идентификатором txtFirstName.
    string firstName = txtFirstName.Text;
    // Использовать полученное значение на странице...
}
```

Работа с виджетом напрямую намного более безопасна к типам, т.к. ошибки неверного обращения к типу обнаруживаются во время компиляции, а не во время выполнения.

Свойство IsPostBack

Обычно необходимость в определении того, является ли текущий запрос HTTP обратной отправкой, чаще всего возникает, когда нужно организовать выполнение блока кода только при первом обращении к данной странице.

```
protected void Page_Load(object sender, System.EventArgs e)
{
   // Заполнить DataSet только при первом
   // входе пользователя на страницу.
   if (!IsPostBack)
   {
         // Заполнить DataSet и кэшировать его!
   }
   // Использовать кэшированный DataSet.
}
```

Взаимодействие с исходящим ответом HTTP

В ASP.NET свойство **Response** класса **Page** предоставляет доступ к экземпляру типа **HttpResponse**. В этом типе определен набор свойств, которые позволяют форматировать HTTP-ответ, отправляемый обратно клиентскому браузеру.

Свойство	Назначение
Cache	Возвращает семантику кэширования веб-страницы
ContentEncoding	Получает или устанавливает набор символов HTTP для выходного потока
ContentType	Получает или устанавливает тип HTTP MIME для выходного потока
Cookies	Получает коллекцию HttpCookie, отправленную текущим запросом
Output	Позволяет выполнять специальный вывод в тело исходящего содержимого HTTP
OutputStream	Позволяет выполнять двоичный вывод в тело исходящего содержимого HTTP
StatusCode	Получает или устанавливает код состояния HTTP вывода, возвращаемого клиенту
StatusDescription	Получает или устанавливает строку состояния HTTP вывода, возвращаемого клиенту
SuppressContent	Получает или устанавливает значение, указывающее, что HTTP не будет отправлен клиенту

Метод	Назначение
Clear()	Очищает все заголовки и вывод содержимого из буфера потока
End()	Посылает весь буферизованный вывод клиенту, после чего закрывает сокетное соединение
Flush()	Посылает весь текущий буферизованный вывод клиенту
Redirect()	Перенаправляет клиента по новому URL
Write()	Записывает значения в выходной поток HTTP-содержимого
WriteFile()	Записывает файл непосредственно в выходной поток HTTP-содержимого

Взаимодействие с исходящим ответом HTTP

Наиболее известным аспектом типа HttpResponse является способность записывать содержимое непосредственно в выходной поток HTTP.

```
protected void btnHttpResponse_Click(object sender, EventArgs e)
{
    Response.Write ("<b>My name is : </b><br>") ;
    Response.Write(this.ToString());
    Response.Write ("<br><br><b>Here was your last request: </b><br>");
    Response.WriteFile("MyHTMLPage.htm");
}
```

Метод HttpResponse.WriteFile() теперь издает содержимое файла *.htm, находящегося в корневом каталоге веб-сайта.

Еще одним аспектом класса HttpResponse является способность перенаправления пользователей на новый URL

```
protected void btnWasteTime_Click(object sender, EventArgs e)
{
    Response.Redirect("http://www.facebook.com");
}
```

Жизненный цикл веб-страницы ASP.NET

Каждая веб-страница ASP.NET имеет фиксированный жизненный цикл.

Когда исполняющая среда ASP.NET принимает входящий запрос определенного файла *.aspx, в памяти размещается ассоциированный с ней объект унаследованного от System.Web.UI.Page типа с помощью конструктора по умолчанию этого типа.

Затем платформа автоматически запускает последовательность событий.

По умолчанию инициируется событие Load, в обработчик которого можно поместить свой специальный код.

```
public partial class Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
          Response.Write("Load event fired!");
    }
}
```

Жизненный цикл веб-страницы ASP.NET

Избранные события типа Page

Событие	Назначение
PreInit	Платформа использует это событие для размещения в памяти всех веб-элементов управления, применения тем, установки мастер-страницы и настройки профилей пользователей. Это событие можно перехватить, чтобы вмешаться в процесс
Init	Платформа использует это событие для установки свойств веб-элементов в их предыдущие значения через обратную отправку или данные состояния представления
Load	Когда возникает это событие, страница и ее элементы управления полностью инициализированы и их предыдущее состояние восстановлено. В этот момент можно безопасно взаимодействовать с каждым веб-виджетом
Событие, инициировавшее обратную отправку	Конечно, события с таким именем нет. Это "событие" просто ссылается на любое событие, инициированное браузером для выполнения обратной отправки на веб-сервер (вроде щелчка на элементе Button)
PreRender	Вся привязка данных к элементам управления и конфигурирование пользовательского интерфейса произошло, и элементы управления готовы для визуализации своих данных в выходящий ответ HTTP
Unload	Страница и ее элементы управления завершили процесс визуализации, и объект страницы готов к уничтожению. В этот момент попытка взаимодействия с исходящим ответом HTTP вызовет ошибку. Однако это событие можно перехватить для выполнения любой очистки на уровне страницы (закрыть файл или соединение с базой данных, провести необходимое протоколирование, освободить объекты и т.д.)

Жизненный цикл веб-страницы ASP.NET

IDE-среда не поддерживает обработку других событий помимо Load.
В этом случае требуется вручную реализовать в файле кода метод по имени Page_ИмяСобытия.

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
          Response.Write("Load event fired!");
    }
    protected void Page_Unload(object sender, EventArgs e)
    {
        // Больше нельзя добавлять данные в ответ HTTP,
        // поэтому будем писать в локальный файл.
        System.IO.File.WriteAllText(@"C:\MyLog.txt", "Page unloading!");
    }
}
```

Роль атрибута AutoEventWireup

Атрибут AutoEventWireup (будучи включенным) приводит к генерации необходимой оснастки для событий внутри автоматически сгенерированного частичного класса

```
<%@ Page Language="C#" AutoEventWireup="true"
        CodeFile="Default.aspx.cs" Inherits="_Default" %>
```

При таком поведении по умолчанию каждый обработчик события уровня страницы будет автоматически добавляться, когда вводится соответствующий именованный метод.

Если же установить атрибут AutoPageWireup в false, то события уровня страницы больше перехватываться не будут

```
<%@ Page Language="C#" AutoEventWireup="false"
        CodeFile="Default.aspx.cs" Inherits="_Default" %>
```

В этом случае все равно можно будет обрабатывать события уровня страницы, используя логику обработки событий C#.

```
public _Default()
{
    // Явно привязаться к событиям Load и Unload.
    this.Load += new Page_Load;
    this.Unload += new Page_Unload;
}
```

Роль файла Web.config

По умолчанию все веб-приложения C# ASP.NET, созданные в Visual Studio, автоматически снабжаются файлом web.config, который инструктирует CLR о том, как обрабатывать запросы привязки, проверять сборки и реализовать прочие детали времени выполнения.

Элемент	Назначение
<appSettings>	Этот элемент служит для установки специальных пар "имя/значение", которые могут программно читаться в память для использования страницами через тип ConfigurationManager
<authentication>	Этот элемент, относящийся к безопасности, используется для определения режима аутентификации для веб-приложения
<authorization>	Этот элемент, связанный с безопасностью, используется для определения, какие пользователи к каким ресурсам имеют доступ на веб-сервере
<connectionStrings>	Этот элемент служит для хранения строк внешних соединений, используемых веб-сайтом
<customErrors>	Этот элемент позволяет сообщить исполняющей среде о том, как следует отображать ошибки, возникающие во время функционирования веб-приложения
<globalization>	Этот элемент используется для конфигурирования установок глобализации для веб-приложения
<namespaces>	Этот элемент документирует все пространства имен, которые нужно включить, если веб-приложение предварительно компилируется с использованием нового инструмента командной строки aspnet_compiler.exe
<sessionState>	Этот элемент используется для управления тем, как и где данные о состоянии сеанса будут храниться исполняющей средой .NET
<trace>	Этот элемент позволяет включать и отключать поддержку трассировки для данного веб-приложения