

\*\*\*

<%--Дериктива определяющая общие настройки страницы--%>

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="\_001\_SimpleWebApplication.Default" %>

<!DOCTYPE html>

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

<head runat="server">

<title>Hello ASP.NET</title>

</head>

<body>

<%--Форма для серверных элементов управления--%>

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

<div>

<%--Серверный элемент управления для вывода текста--%>

<asp:Label ID="Output" runat="server"></asp:Label>

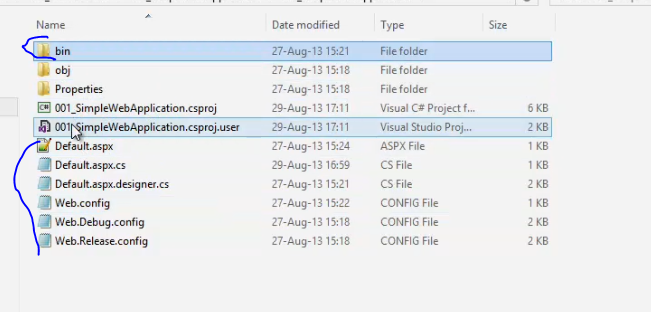
</div>

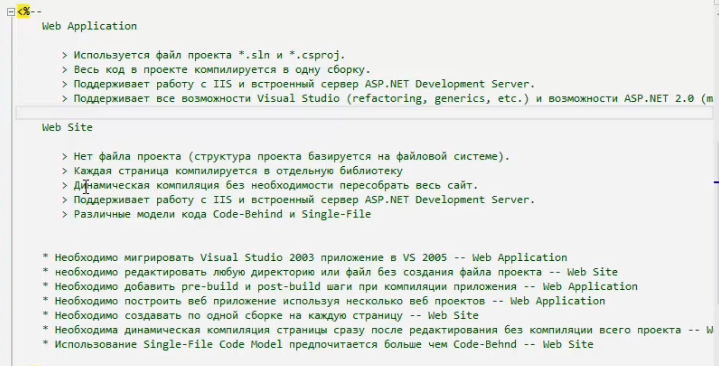
</form>

</body>

</html>

Чтобы разместить файлы на серверы нужны эти файлы





\*\*\*

Серверные теги

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="001\_ServerTag.aspx.cs" Inherits="\_003\_Samples.\_001\_ServerTag" %>

<!DOCTYPE html>

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

<head runat="server">

<title>Серверный тэг</title>

</head>

<body>

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

<div>

<!--Серверные тэги позволяют внедрять в разметку страницы блоки C# кода-->

<%

for (int i = 0; i < 10; i++)

{

Response.Write("hello world!<br />");

}

%>

Сегодня

<%Response.Write(DateTime.Now.ToLocalTime()); %>

<br />

Сегодня

<!--Знак = после открывающего серверного тэга указывает, что все выражение должно передаватся как параметр в метод

Write свойства Response и отображаться в указанном месте страницы-->

<%= DateTime.Now.ToLocalTime() %>

</div>

</form>

</body>

</html>

\*\*\*

Коменты

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

<div>

<!--HTML комментарий, передается в разметке клиенту-->

<%--ASP.NET комментарий, остается в исходном коде страницы. Не передается клиенту. --%>

<%--

Многострочный ASP.NET комментарий

Line 1

Line 2

--%>

<%

// Однострочный C# комментарий

/\*

Многострочный C# комментарий

Line 1

Line 2

\*/

%>

</div>

</form>

\*\*\*

Размер експлорера

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

{

FileInfo info = new FileInfo("C:\\windows\\explorer.exe");

long size = info.Length;

string output = string.Format("Размер фала explorer.exe <b>{0}</b> байт", size);

Output.Text = output;

}

\*\*\*

Класи

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

{

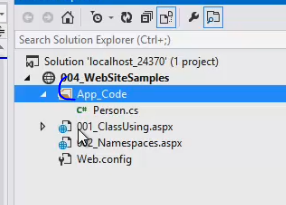
Person p1 = new Person("admin", "admin@example.com");

Output.Text = p1.GenerateHtml();

}

\*\*\*

Cs-файл нужно добавлять в папку App Code (Не в WebApllication)



\*\*\*

Импорт в одном файле

<%@ Import Namespace="System.IO" %>

<!DOCTYPE html>

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

<head runat="server">

<title>Импортирование пространств имен</title>

</head>

<body>

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

<div>

<%

FileInfo info = new FileInfo("C:\\windows\\explorer.exe");

// Без импортирования пространства имен следует использовать полное имя типа.

// System.IO.FileInfo info = new System.IO.FileInfo("C:\\windows\\explorer.exe");

long size = info.Length;

string output = string.Format("Размер фала explorer.exe <b>{0}</b> байт", size);

Response.Write(output);

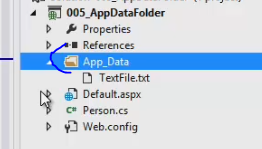
%>

</div>

</form>

\*\*\*

AppData для хранения файлов



<div>

<p>Login: <%LOGIN%></p>

<p>Email: <%EMAIL%></p>

</div>

public partial class Default : System.Web.UI.Page

{

// Данная переменная выводится на странице с помощью выражения <%=output %>

public string output;

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

{

// С помощью метода Server.MapPath() виртуальный путь можно преобразовать в физический.

// Относительный путь "App\_Data\\TextFile.txt" преобразовывается в подобный C:\wwwroot\mysite\App\_Data\TextFile.txt

string filename = Server.MapPath(@"App\_Data\TextFile.txt");

output = File.ReadAllText(filename);

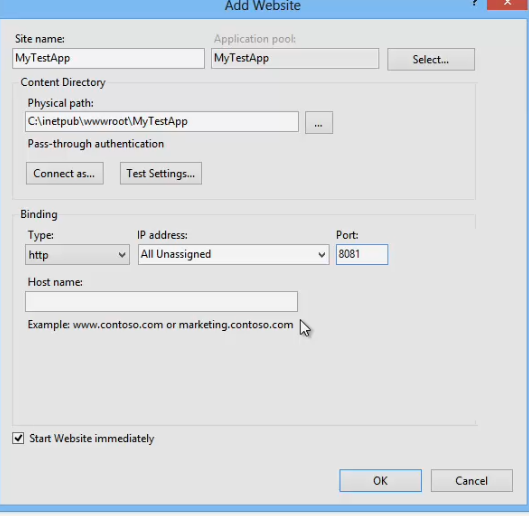
Person p = new Person("admin", "admin@example.com");

output = output.Replace("<%LOGIN%>", p.Login);

output = output.Replace("<%EMAIL%>", p.Email);

}

}



\*\*\*

PreLoad

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

{

Label1.Text += "Init <br />";

}

// При наличии атрибута AutoEventWireup="true" директивы @Page, методы с именем Page\_ИмяСобытия, автоматически становятся обработчиками событий страницы.

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

{

Label1.Text = "Сработал обработчик события Load";

}

\*\*\*

Привязка к событию

public EventHandling2()

{

this.Load += new EventHandler(EventHandling2\_Load);

}

void EventHandling2\_Load(object sender, EventArgs e)

{

Label1.Text = "Сработал обработчик события Load";

}

\*\*\*

События

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

{

Label1.Text += "PreInit <br />";

}

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

{

Label1.Text += "Init <br />";

}

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

{

Label1.Text += "InitComplete <br />";

}

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

{

Label1.Text += "PreLoad <br />";

}

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

{

Label1.Text += "Load <br />";

}

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

{

Label1.Text += "LoadComplete <br />";

}

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

{

Label1.Text += "PreRender <br />";

}

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

{

Label1.Text += "PreRenderComplete <br />";

}

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

{

Label1.Text += "SaveStateComplete <br />";

}

// Рендеринг страницы. Все элементы управления превращаются в HTML, CSS и JavaScript, который будет отправлен клиенту.

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

{

// Освобождение ресурсов, которые использовала страница.

}

\*\*\*

<asp:TextBox ID="TextBox1" runat="server" /> + <asp:TextBox ID="TextBox2" runat="server" /> = <asp:TextBox ID="TextBox3" runat="server" />

<asp:Button ID="Button1" Text="Сумма" runat="server" OnClick="Button1\_OnClick" />

IsPostBack

// Закомментировать

if (!IsPostBack)

{

// Устанавливать значения для полей ввода только при первой загрузке страницы.

// При postback запросе данный код не отработает.

TextBox1.Text = "0";

TextBox2.Text = "0";

}

}

protected void Button1\_OnClick(object sender, EventArgs e)

{

int a = Convert.ToInt32(TextBox1.Text);

int b = Convert.ToInt32(TextBox2.Text);

TextBox3.Text = Convert.ToString(a + b);

}

}

\*\*\*

Передача пераметров на страницу

1 страница

<div>

<a href="002\_RequestProp.aspx?param=1">Первая ссылка</a>

<br />

<a href="002\_RequestProp.aspx?param=2">Вторая ссылка</a>

</div>

2 страница код

// Request.QueryString[имя\_параметра] - это свойство дает возможность получить данные из адресной строки.

// В первую очередь, при работе с GET параметрами, нужно проверить наличие значений при получении запроса.

// После (если это требуется) проверить тип полученного значения. GET параметры могут быть изменены пользователем

// в следствии чего, неправильные данные в параметрах могут нарушить работу страницы.

string param = Request.QueryString["param"];

if (string.IsNullOrEmpty(param))

{

Label1.Text = "В адресной строке нет GET параметра с именем <i>param</i>";

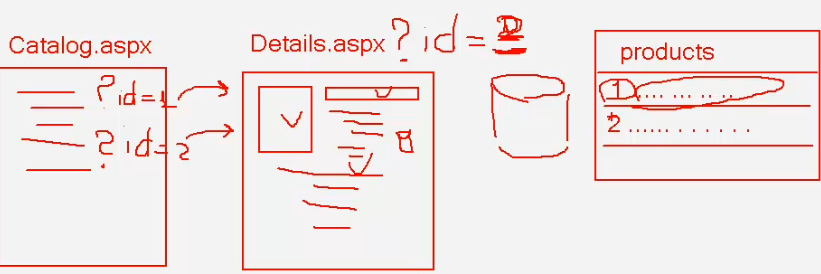
}

else

{

Label1.Text = param;

}



Передача данных на сервер

1 страница без кода

<!--

form - элемент HTML позволяющий отправлять сообщения со стороны клиента на сервер.

атрибуты:

action - адрес серверного ресурса, которому будет отправлен запрос

method - HTTP глагол

Каждый из ключей, который находиться в теле запроса, на стороне сервера можно

прочесть обратившись к коллекции Request.Form[имя\_ключа];

Обычно значения серверных элементов управления передаются в форме, но при этом

на стороне сервера они считываются из свойств объектов контролов, а не из коллекции Form

-->

<form action="002\_RequestProp.aspx" method="post">

Логин

<!--Поле ввода для логина-->

<input type="text" name="loginParam" /><br />

Пароль

<!--Поле ввода для пароля-->

<input type="text" name="passwordParam" />

<br />

<!--Кнопка, которая инициирует отправку данных на сервер по адресу указанному в атрибуте action-->

<input type="submit" value="Отправить" />

</form>

2 страница разметка

<div>

Логин: <asp:Label ID="LabelLogin" runat="server" />

<br />

Пароль: <asp:Label ID="LabelPassword" runat="server" />

</div>

2 страница код

string login, password;

// Request.Form[имя\_параметра] - свойство дает возможность получить доступ к коллекции переменных формы,

// которые передаются в HTTP заголовках вместе с запросом к странице.

login = Request.Form["loginParam"];

password = Request.Form["passwordParam"];

if (string.IsNullOrEmpty(login))

{

LabelLogin.Text = "Параметр loginParam не найден.";

}

else

{

LabelLogin.Text = login;

}

if (string.IsNullOrEmpty(password))

{

LabelPassword.Text = "Параметр passwordParam не найден.";

}

else

{

LabelPassword.Text = password;

}

}

\*\*\*

Свойства Request

Label1.Text += "ApplicationPath = " + Request.ApplicationPath + "<br />";

Label1.Text += "PhysicalApplicationPath = " + Request.PhysicalApplicationPath + "<br />";

Label1.Text += "Browser = " + Request.Browser.Type + "<br />";

Label1.Text += "Path = " + Request.Path + "<br />";

Label1.Text += "UserLanguages = " + Request.UserLanguages[0] + "<br />";

Label1.Text += "IsLocal = " + Request.IsLocal + "<br />";

Label1.Text += "<br /><br /><br /><br />";

Label1.Text += "Headers:<br />";

// Получение значений всех заголовков полученных от браузера.

foreach (string item in Request.Headers)

{

Label1.Text += item + " : " + Request.Headers[item] + "<br />";

}

}

\*\*\*

Response Write

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

{

// откройте в браузере исходный код странмцы и сравните куда в разметку попала

// строка "Hello ASP.NET (from code file)" и куда строка "Hello ASP.NET"

Response.Write("Hello ASP.NET (from code file)");

}  
\*\*\*

Response WriteFile

<div>

<%

Response.WriteFile("Item.txt");

%>

</div>

<ul>

<li>List Item 1</li>

<li>List Item 2</li>

<li>List Item 3</li>

<li>List Item 4</li>

</ul>

\*\*\*

Respones Redirect

<div>

<asp:Button ID="Button1" Text="edu.cbsystematics.com" runat="server" OnClick="Button1\_Click" />

</div>

// Перенаправление пользователя на другую страницу.

Response.Redirect("http://edu.cbsystematics.com");

\*\*\*

Server

// Имя компьтера на котором запускается страница.

Label1.Text += "MachineName = " + Server.MachineName + "<br />";

// Возрвращает физический путь соответствующий виртуальному.

Label1.Text += "MapPath('001\_MapPath.aspx') = " + Server.MapPath("001\_MapPath.aspx") + "<br />";

\*\*\*

HTML Encode

// Заменяет обычную строку, строкой допустимых символов HTML

// < - &lt;

// > - &gt;

// " - &guot;

// & - &amp;

Label1.Text += Server.HtmlEncode("<b>Hello World!</b>");

Label1.Text += "<br />";

Label1.Text += Server.HtmlDecode("&lt;b&gt;Hello World!&lt;/b&gt;");

\*\*\*

URL Encode

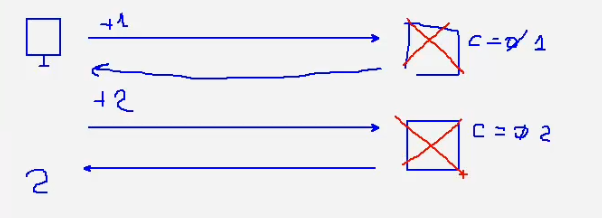
// Заменяет обычную строку, строкой допустимых символов URL

Label1.Text += "<b>Server.UrlEncode(http://example.com?p=123)</b> = " + Server.UrlEncode("http://example.com?p=123");

Label1.Text += "<br />";

Label1.Text += "<b>Server.UrlDecode(http%3a%2f%2fexample.com%3fp%3d123)</b> = " + Server.UrlDecode("http%3a%2f%2fexample.com%3fp%3d123");

Хранение состояния



<div>

<asp:Label ID="CounterLabel" runat="server" Text="0"></asp:Label>

<br />

<asp:Button ID="AddOneButton" runat="server" Text="+1" OnClick="AddOneButton\_Click" />

<asp:Button ID="AddTwoButton" runat="server" Text="+2" OnClick="AddTwoButton\_Click" />

</div>

</form>

// При каждом запросе экземпляр этого класса создается повторно,

// соответственно при вызове любого из обработчиков (AddOneButton\_Click или AddTwoButton\_Click)

// значение переменной \_counter будет равное 0.

int \_counter = 0;

protected void AddOneButton\_Click(object sender, EventArgs e)

{

\_counter += 1;

CounterLabel.Text = \_counter.ToString();

}

protected void AddTwoButton\_Click(object sender, EventArgs e)

{

\_counter += 2;

CounterLabel.Text = \_counter.ToString();

}

}

View State

protected void AddOneButton\_Click(object sender, EventArgs e)

{

int counter = 0;

// Чтение данных из ViewState

object obj = ViewState["counter"];

if (obj != null)

{

counter = (int)obj;

}

counter += 1;

// Запись данных во ViewState

ViewState["counter"] = counter;

}

protected void AddTwoButton\_Click(object sender, EventArgs e)

{

int counter = 0;

// Чтение данных из ViewState

object obj = ViewState["counter"];

if (obj != null)

{

counter = (int)obj;

}

counter += 2;

// Запись данных во ViewState

ViewState["counter"] = counter;

}

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

{

CounterLabel.Text = Convert.ToString(ViewState["counter"]);

}

Сереализация обьекта в ViewState

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="04\_ViewStateSaveObject.aspx.cs"

Inherits="StateManagment.\_01\_ViewState.ViewStateSaveObject" ViewStateEncryptionMode="Always" %>

<%--

По умолчани ViewState отправляется в не зашифрованном виде.

Что бы Включить шифрование используется следующий атрибут ViewStateEncryptionMode

"Always" - шифровать все данные во ViewStat при каждом запросе/ответе

"Never" - не шифровать данные

"Auto" - (по умолчанию) шифровать только де данные ViewState, которые относятся к контролам запросившим шифрование

свойе части информации во ViewState--%>

<!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>View State #4</title>

</head>

<body>

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

<div>

<asp:Button ID="WriteButton" Text="Записать во ViewState" runat="server" OnClick="WriteButton\_Click" />

<asp:Button ID="ReadButton" Text="Прочитать из ViewState" runat="server" OnClick="ReadButton\_Click" />

<br />

<br />

<b>UserName: </b>

<asp:Label ID="UserNameLabel" runat="server" /><br />

<b>Email: </b>

<asp:Label ID="EmailLabel" runat="server" /><br />

</div>

</form>

protected void WriteButton\_Click(object sender, EventArgs e)

{

// Объекты, сохраняемые в состояние вида, должны быть сериализуемого типа.

// Класс должен иметь атрибут Serializable

Customer cust = new Customer("admin", "admin@example.com");

ViewState["customer"] = cust;

}

protected void ReadButton\_Click(object sender, EventArgs e)

{

Customer cust = ViewState["customer"] as Customer;

if (cust != null)

{

UserNameLabel.Text = cust.UserName;

EmailLabel.Text = cust.Email;

}

else

{

UserNameLabel.Text = "<i>null</i>";

EmailLabel.Text = "<i>null</i>";

}

}

\*\*\*

URL передача и извлечение параметров

1

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

<div>

<ul>

<li><a href="02\_DescriptionPage.aspx?id=1">Mozzila Firefox</a></li>

<li><a href="02\_DescriptionPage.aspx?id=2">Google Chrome</a></li>

<li><a href="02\_DescriptionPage.aspx?id=3">Internet Explorer</a></li>

<li><a href="02\_DescriptionPage.aspx?id=4">Opera</a></li>

<li><a href="02\_DescriptionPage.aspx?id=5">Safari</a></li>

</ul>

</div>

</form>

2 код

Dictionary<int, string> browsers = new Dictionary<int, string>();

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

{

browsers.Add(1, “Mozilla Firefox is a free and open source web browser descended from the Mozilla Application Suite and managed by Mozilla Corporation. As of November 2011, Firefox is the second or third most widely used browser, according to different estimates, with approximately 25% of worldwide usage share of web browsers.”);

browsers.Add(2, “Google Chrome is a web browser developed by Google that uses the WebKit layout engine. It was first released as a beta version for Microsoft Windows on September 2, 2008, and the public stable release was on December 11, 2008. The name is derived from the graphical user interface frame, or ‘chrome’, of web browsers.”);

browsers.Add(3, “Windows Internet Explorer (formerly Microsoft Internet Explorer, commonly abbreviated IE or MSIE) is a series of graphical web browsers developed by Microsoft and included as part of the Microsoft Windows line of operating systems, starting in 1995.”);

browsers.Add(4, “Opera is a web browser and Internet suite developed by Opera Software with over 200 million users worldwide.[4] The browser handles common Internet-related tasks such as displaying web sites, sending and receiving e-mail messages, managing contacts, chatting on IRC, downloading files via BitTorrent, and reading web feeds. Opera is offered free of charge for personal computers and mobile phones.”);

browsers.Add(5, “Safari is a web browser developed by Apple Inc. and included with the Mac OS X and iOS operating systems. First released as a public beta on January 7, 2003[3] on the company’s Mac OS X operating system, it became Apple’s default browser beginning with Mac OS X v10.3 ‘Panther’.”);

try

{

string temp = Request.QueryString[“id”];

int browserId = Convert.ToInt32(temp);

DescriptionLabel.Text = browsers[browserId];

}

catch

{

DescriptionLabel.Text = «ERROR!»;

}

}

\*\*\*

Cookies

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

<div>

<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>

<asp:Button ID="SaveButton" Text="Сохранить в cookies" runat="server" OnClick="SaveButton\_Click" />

<a href="02\_CookieRead.aspx">Страница тестирования</a>

</div>

</form>

// Создаем cookie набор.

// Такой cookie-набор будет храниться до тех пор, пока пользователь не закроет браузер.

HttpCookie cookie = new HttpCookie("MyCookie", TextBox1.Text);

// Добавляем cookie-набор в текущий ответ.

Response.Cookies.Add(cookie);

Read Cookies

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

{

HttpCookie cookie = Request.Cookies["MyCookie"];

if (cookie != null)

{

OutputLabel.Text = cookie.Value;

}

else

{

OutputLabel.Text = "В запросе нет cookie-набора с именем 'MyCookie'";

}

}

\*\*\*

Cookies с датой

// Создаем cookie набор.

// Такой cookie-набор будет храниться до тех пор, пока пользователь не закроет браузер.

HttpCookie cookie = new HttpCookie("MyCookie", TextBox1.Text);

// Данный cookie-набор будет сохранен в браузере на 7 дней.

// Каждый раз когда web браузер будет делать запрос к данному сайту

// к запросу будут добавляться значения cookie-набора.

cookie.Expires = DateTime.Now.AddDays(7);

// Добавляем cookie-набор в текущий ответ.

Response.Cookies.Add(cookie);

Удаление Cookies

// Удаление значений cookie-набора.

protected void ClearCookies\_Click(object sender, EventArgs e)

{

HttpCookie cookie = new HttpCookie("MyCookie");

// Для удаления cookie-набора устанавливаем значение свойства Expires равное прошедшей дате.

cookie.Expires = DateTime.Now.AddDays(-1);

// Добавляем cookie в ответ.

Response.Cookies.Add(cookie);

// Делаем ответ браузеру и заставляем его выполнить запрос на эту же страницу.

Response.Redirect(Request.Url.PathAndQuery);

// При отправке сообщения браузеру в тело ответа добавляется просроченный cookie-набор.

// При повторном запросе к серверу браузер видит, что cookie-набор просрочен и удаляет его

// вместо того, что бы добавить в запрос.

}

ХРАНЕНИЕ СОСТОЯНИЯ НА СЕРВЕРЕ

Сессия

<div>

<asp:TextBox ID="TextBox1" runat="server" />

<asp:Button ID="SaveButton" Text="Сохранить в сеанс" runat="server" OnClick="SaveButton\_Click" />

<a href="02\_SessionRead.aspx">Страница для тестирования</a>

</div>

protected void SaveButton\_Click(object sender, EventArgs e)

{

// Запись значения в сессию.

Session["Key"] = TextBox1.Text;

// Значение будет храниться в памяти сервера 1 минуту. (по умолчанию значение храниться 20 минут)

Session.Timeout = 1;

}

2 страница

<div>

<asp:Label ID="OutputLabel" Text="Сеанс не содержит ключ 'Key'" runat="server" />

<asp:Button ID="ClearButton" Text="Очистить сеанс" runat="server" OnClick="ClearButton\_Click" />

</div>

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

{

string text = Session["Key"] as string;

if (text != null)

{

OutputLabel.Text = text;

}

}

protected void ClearButton\_Click(object sender, EventArgs e)

{

//Session.Clear(); // Удаляет все значения из объекта коллекции.

Session.Abandon(); // Удаляет объект коллекции.

Response.Redirect(Request.Url.PathAndQuery);

}

\*\*\*

Application

Общее данные (Количество людей онлайн)

private static string key = "counter";

public int Counter

{

get

{

// Данные, которые хранятся в Application - глобальные и доступны всем пользователям.

object obj = Application[key];

if (obj == null)

{

Application[key] = 0;

return 0;

}

return (int)obj;

}

set

{

Application[key] = value;

}

}

protected void AddOneButton\_Click(object sender, EventArgs e)

{

Counter += 1;

CounterLabel.Text = Counter.ToString();

}

protected void AddTwoButton\_Click(object sender, EventArgs e)

{

Counter += 2;

CounterLabel.Text = Counter.ToString();

}

}

\*\*\*

Монопольный доступ

protected void AddOneButton\_Click(object sender, EventArgs e)

{

// Запрашиваем монопольный доступ.

Application.Lock();

int counter = 0;

if (Application["counter"] != null)

{

counter = (int)Application["counter"];

}

Application["counter"] = ++counter;

CounterLabel.Text = counter.ToString();

// Отключаем монопольный доступ.

Application.UnLock();

}

Механизм авторизации

Страница авторизации

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

<div>

Логин <asp:TextBox ID="LoginTextBox" runat="server" />

<br />

Пароль <asp:TextBox ID="PasswordTextBox" TextMode="Password" runat="server" />

<br />

<asp:Button ID="LoginButton" Text="Вход" runat="server" OnClick="LoginButton\_Click" />

<br />

<asp:Label ID="ErrorLabel" runat="server" Text="Логин или пароль введены не правильно!" ForeColor="Red" Visible="false"></asp:Label>

</div>

</form>

Код страницы авторизации

protected void LoginButton\_Click(object sender, EventArgs e)

{

if (PasswordTextBox.Text == "qwerty" && LoginTextBox.Text == "admin")

{

HttpCookie cookieName = new HttpCookie("name", LoginTextBox.Text);

HttpCookie cookieSign = new HttpCookie("sign", CryptoProvider.GetMD5Hash(LoginTextBox.Text + "s@lt"));

// Для того, что бы гарантировать, что данный cookie-набор был создан нашей странице, а не злоумышленником,

// в ответ кроме имени мы отправляем cookie-набор подпись.

// Для того, что бы усложнить подбор значения, которое мы поместили в подпись используется "соль" - слово или

// набор символов.

// Каждый раз когда от пользователя будут приходить запросы, мы будем повторно генерировать подпись и проверять,

// что бы она совпадала с полученой.

Response.Cookies.Add(cookieName);

Response.Cookies.Add(cookieSign);

Response.Redirect("Default.aspx");

}

else

{

ErrorLabel.Visible = true;

}

}

Код дефолтной страницы

<div>

Добро пожаловать

<asp:Label ID="LoginLabel" Text="Гость" runat="server" />

<asp:HyperLink ID="LoginLink" NavigateUrl="Login.aspx" runat="server" >Вход</asp:HyperLink>

<asp:HyperLink ID="LogoutLink" NavigateUrl="Logout.aspx" Visible="false" runat="server" >Выход</asp:HyperLink>

<br />

<br />

<br />

<a href="Closed.aspx">Секретная страница</a>

</div>

Код дефолтной страницы

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

{

HttpCookie name = Request.Cookies["name"];

if (name != null)

{

// Прячем ссылку "Вход"

LoginLink.Visible = false;

// Отображаем ссылку "Выход"

LogoutLink.Visible = true;

LoginLabel.Text = name.Value;

}

}

Код секретной станицы

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

{

HttpCookie name = Request.Cookies["name"];

HttpCookie sign = Request.Cookies["sign"];

if (name != null && sign != null)

{

// Если полученная подпись правильная прекращаем дальнейшую обработку события Load

// и переходим к следующим этапам жизненного цикла страницы.

if (sign.Value == CryptoProvider.GetMD5Hash(name.Value + "s@lt"))

{

return;

}

}

Response.Redirect("Login.aspx");

}

