Крос-платформне програмування

Лекція 7

Компонентно-орієнтоване проектування. Основи розробки веб-застосувань за допомогою ASP.NET

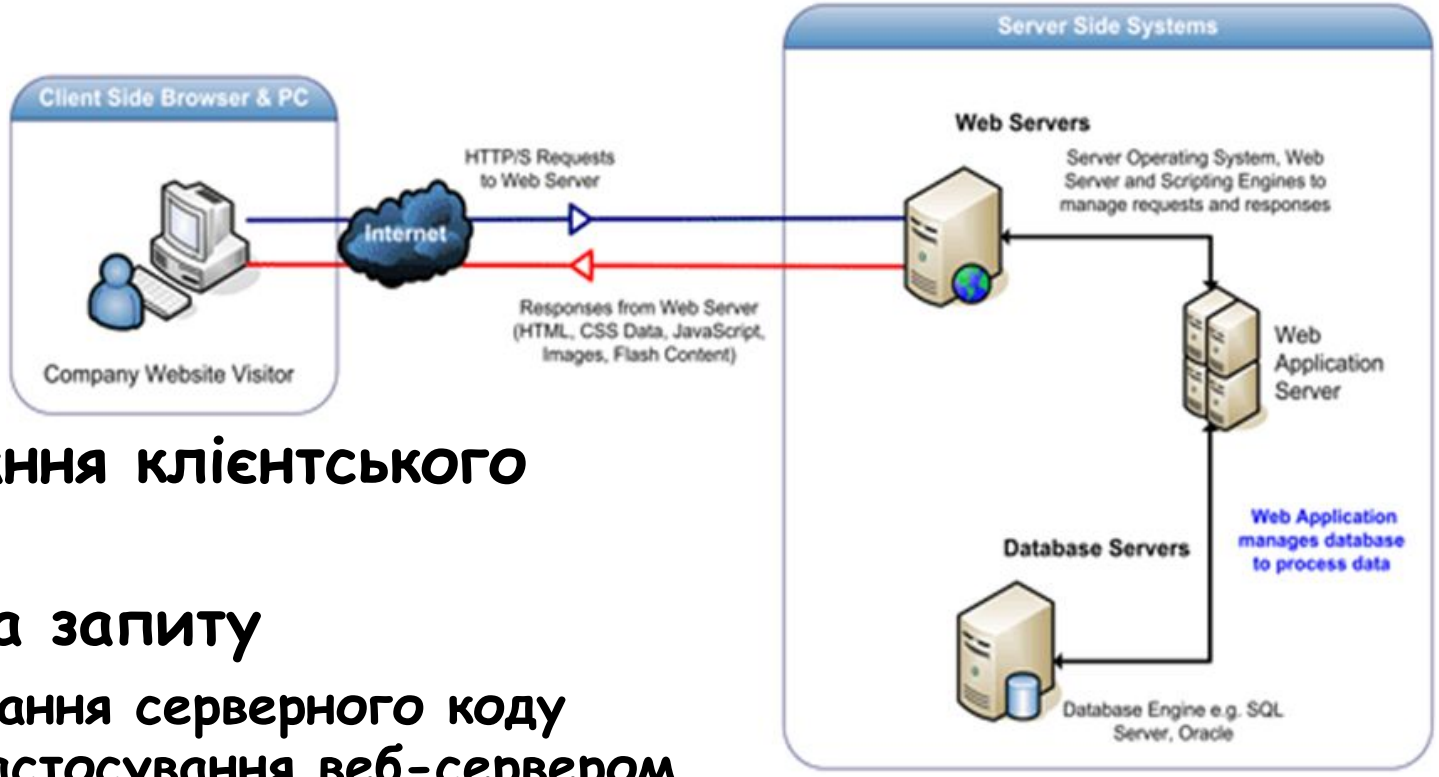
2 квітня, 2014

Примітка: слайди лекції підготовлені за матеріалами курсу В.К.Толстих www.tolstykh.com та http://jskreator.narod.ru/proaspnet20cs/glance.htm

План наступних лекцій

- Інсрументи розробки серверних веб-застосувань
- Огляд технології ASP.NET: веб-сайти, веб-застосування, веб-служби
- Клас Page, його властивості та методи. Групи елементів керування. Серверні елементи керування та елементи керування HTML
- Відправка даних іншій сторінці. Керування станом у ASP.NET
- Майстер-сторінка як шаблон веб-сторінки. Сторінка вмісту
- Динамічне створення елементів керування. Користувацькі елементи керування

Взаємодія клієнта та сервера



- Отримання клієнтського запиту
- Обробка запиту
  - Виконання серверного коду веб-застосування веб-сервером
  - Звернення до серверів баз данних (БД)
  - Звернення до веб-сервісів
- Генерація відповіді
  - HTML, ресурси (зображення, відео тощо), браузерні скрипти
- Видача відповіді клієнту

Клієнтські сценарії та застосування

- Клієнтський сценарій виконується на комп'ютері-клієнті
  - Виконуються браузером
  - Зазвичай без взаємодії з сервером
  - Написані на мові сценаріїв – JavaScript, VBScript
- Інші типи програм, що можуть завантажуватись та виконуватися на боці клієнта:
  - Java-аплети
  - програмні компоненти ActiveX Controls
  - інтерактивні мультимедійні об'єкти Flash

Серверні веб-застосування

- Виконуються на сервері
- Клієнт отримує лише результат
- Поділ частин веб-застосування
  - Статичний контент сторінок
  - Елементи управління
  - Ресурси
  - Код генерації динамічного контента
  - Робота з базами данних
  - Оформлення, дизайн
- Механізми, що реалізують серверну частину обробки даних
  - Internet Server Application Programming Interface (ISAPI), Common Gateway Interface (CGI)
- Мови (не обов'язково скриптові)
  - PHP, ASP, ASP.NET, JSP, Python тощо

Статичні та динамічні веб-сторінки

- **Статичні сторінки** – зберігаються на сайті у тому ж вигляді, в якому передаються клієнту
- **Динамічні сторінки** – перед відправкою клієнту проходять цикл обробки на сервері
- **Динамічне наповнення сторінки** – це інформація, яка відрізняється від перегляду до перегляду і зміст якої залежить від того, кому вона призначена
- Категорії технологій Web-розробки:
  - окремі застосування, що виконуються серверними викликами (напр., застосовується Perl у сценаріях CGI)
  - сценарії, що інтерпретуються серверним ресурсом (PHP, класичне середовище ASP)

CGI-скрипти

- **CGI** *(Common Gateway Interface)* – специфікація інтерфейсу взаємодії Web-сервера з зовнішніми прикладними програмами, яка забезпечує засоби створення динамічних Web-сторінок на основі даних, отриманих від користувача
- **CGI-скрипти** або шлюзи - програми, написані у відповідності зі специфікацією CGI
  - CGI-програма завантажується у пам'ять Web-сервера для виконання деякої роботи, повертає результати та при завершенні видаляється
  - Недолік – погана масштабованість

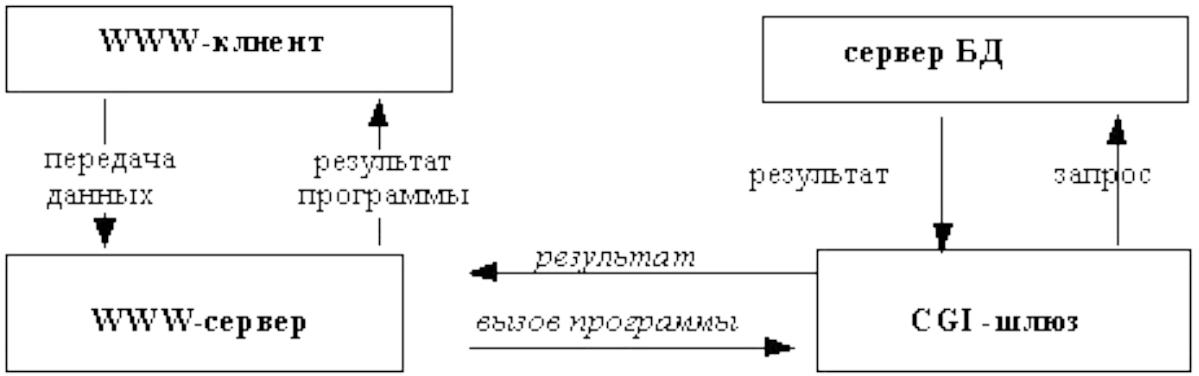
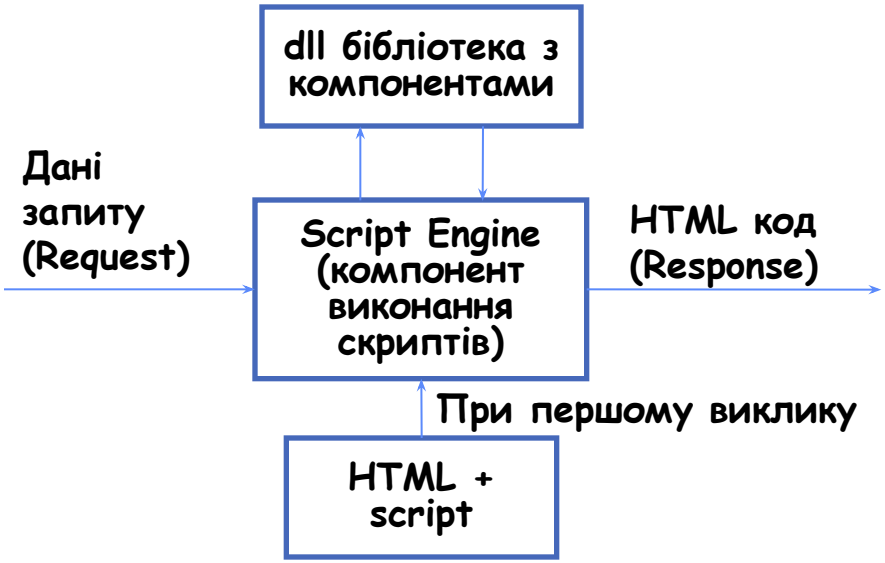


Схема взаємодії CGI-шлюза

ISAPI від Microsoft

- **ISAPI** *(Internet Server Application Programming Interface)* – код DLL-бібліотеки весь час знаходиться у пам'яті і для кожного запиту створює не процеси, а потоки виконання
  - Виконання в процесі IIS-сервера підвищує продуктивність і масштабованість



Технологія ASP

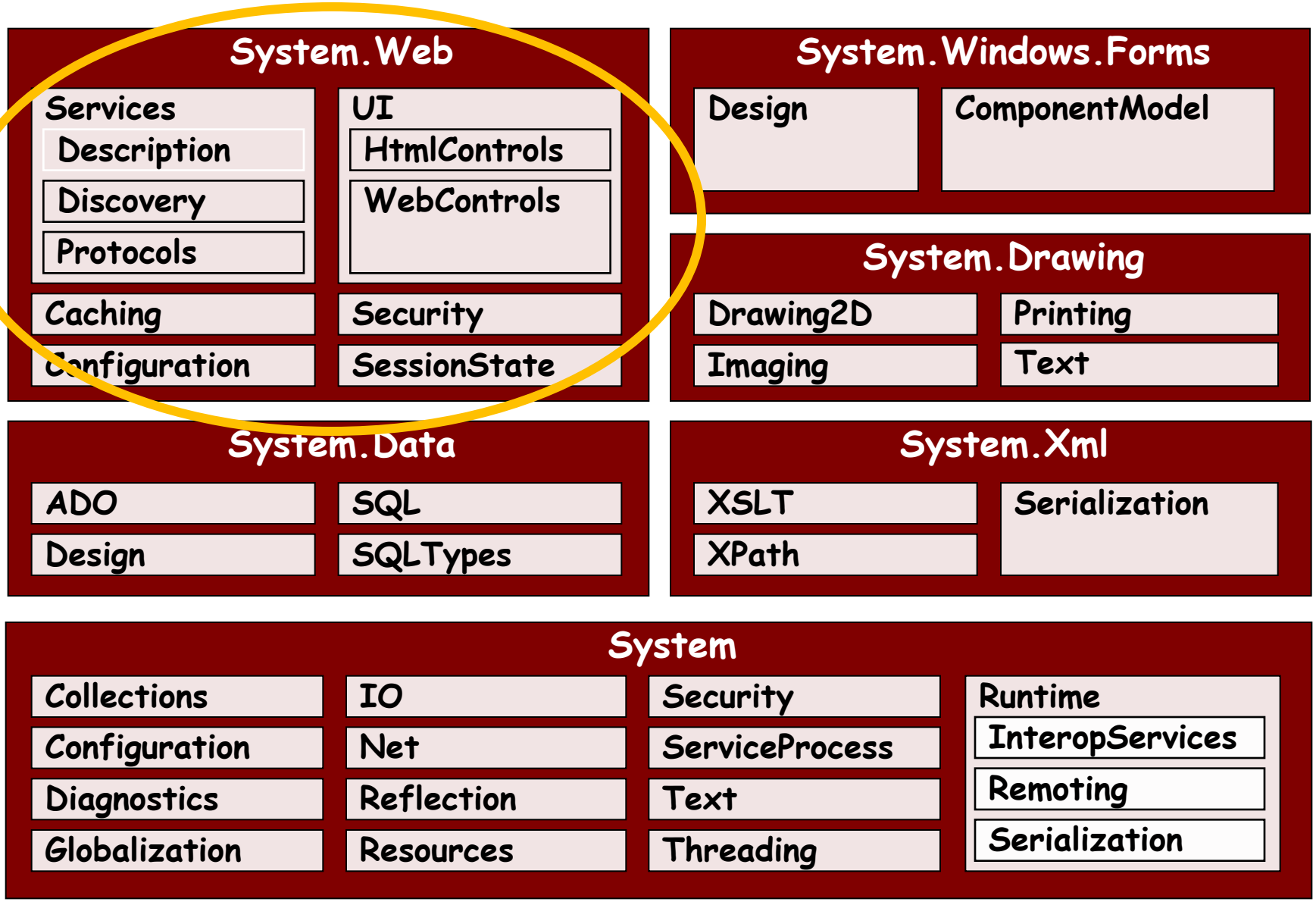
- ASP (Active Server Pages)
  - Підтримується ISAPI-розширенням сервера
  - Використовує скриптові мови
  - Код вбудований у HTML у вигляді спеціальних тегів
- Недоліки
  - заплутаний код
  - використання мов сценаріїв зумовлює низьку продуктивність
  - не підтримує багато можливостей ООП
  - неможливо повторно використовувати готові рішення в інших проектах

```
<%@ language="JScript"%></b>
<html>
<head>
<title>Наш первый asp-скрипт!</title>
</head>
<body>
<h3 align=center>Протестируем наш первый asp-скрипт...</h3>
<% var s="<H1 align=center>HELLO, WORLD!!!</H1>";%>
<%=s%>
</body>
</html>
```

Технологія ASP.NET

- ASP.NET – частина технології .NET, що використовується для написання клієнт-серверних інтернет-застосувань
- Основні риси
  - інтегрована з .NET Framework
  - вихідний код компілюється, а не інтерпретується
  - є багатомовною (C#, VB.NET, J# тощо)
  - виконується в середовищі (під управлінням) CLR (управління пам'яттю, автоматичне збирання сміття, безпека типів)
  - є об'єктно-орієнтованою
  - використовує бібліотеку класів .NET Framework
  - використовує технологію доступу до даних ADO.NET
  - підтримує різні типи браузеров
  - легко встановлюється та конфігурується

Простори імен FCL



Архітектура Web-застосувань у ASP.NET

- **Типи Web-застосувань у Visual Studio**
  - **Web-сайти – модель проекту, що використовує структуру каталогу**
  - **Web-застосування – модель проекту з використанням файлу проекту**
  - **Web-служби - компоненти веб-сервера, які клієнтське застосування може викликати, виконуючи HTTP-запити**
  - **серверні елементи керування AJAX**
- **Для розробки програм можна використовувати**
  - **Visual Studio .NET**
  - **Visual Web Developer**
  - **текстовий редактор, за умови наявності доступу до серверу IIS**
  - **WebMatrix тощо**

Модель поділу коду представлення та коду реалізації

- **Веб-застосування (веб-сайт) складається з:**
  - **інформаційної частини**
  - **програмного коду**
  - **відомостей про конфігурацію**
- **Варіанти організації Web-сторінок**
  - **весь код *інформаційної частини* та *програмна частина* зберігаються в одному файлі з розширенням .aspx**
  - **Web-сторінка розділяється на дві частини: Web-форму (міститься у файлі .aspx) та файл, що містить програмний код (файл з розширенням .cs)**

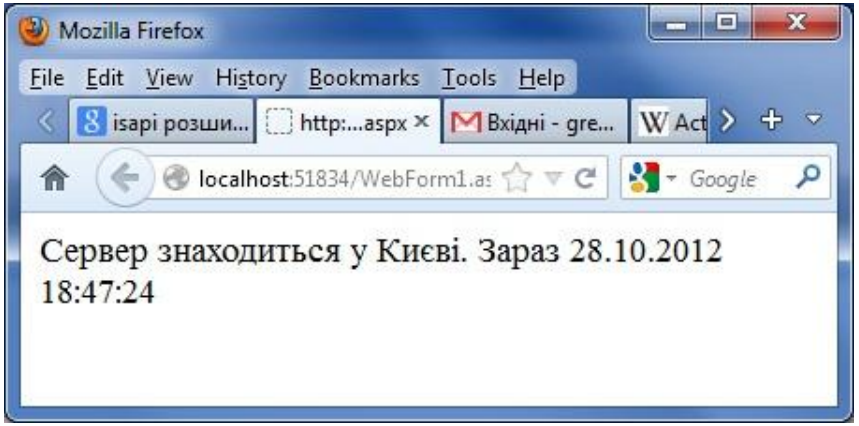
ASP.NET-сторінка при розділенні коду

- **Форма знаходиться у файлі WebForm1.aspx**

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication.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:Label ID="Time" runat="server" Text="Сервер знаходиться у Києві. Зараз "></asp:Label>
    </div>
    </form>
</body>
</html>
```

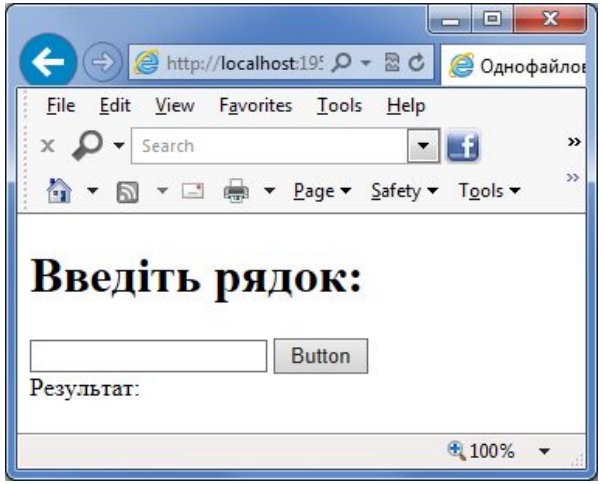
- **Клас сторінки на мові *C#* у файлі WebForm1.aspx.cs**

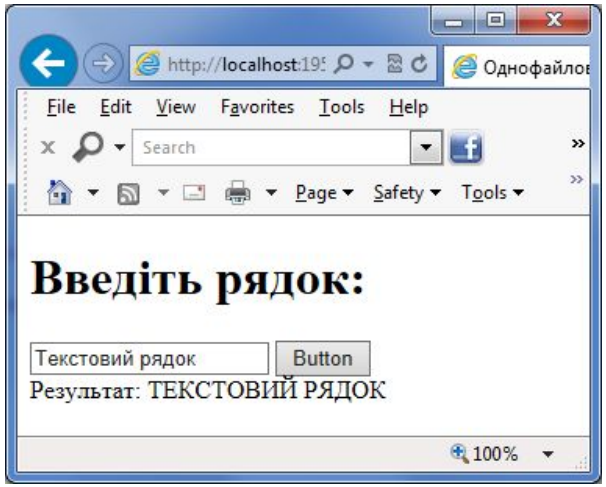
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace WebApplication
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            Time.Text += DateTime.Now.ToString ();
        }
    }
}
```



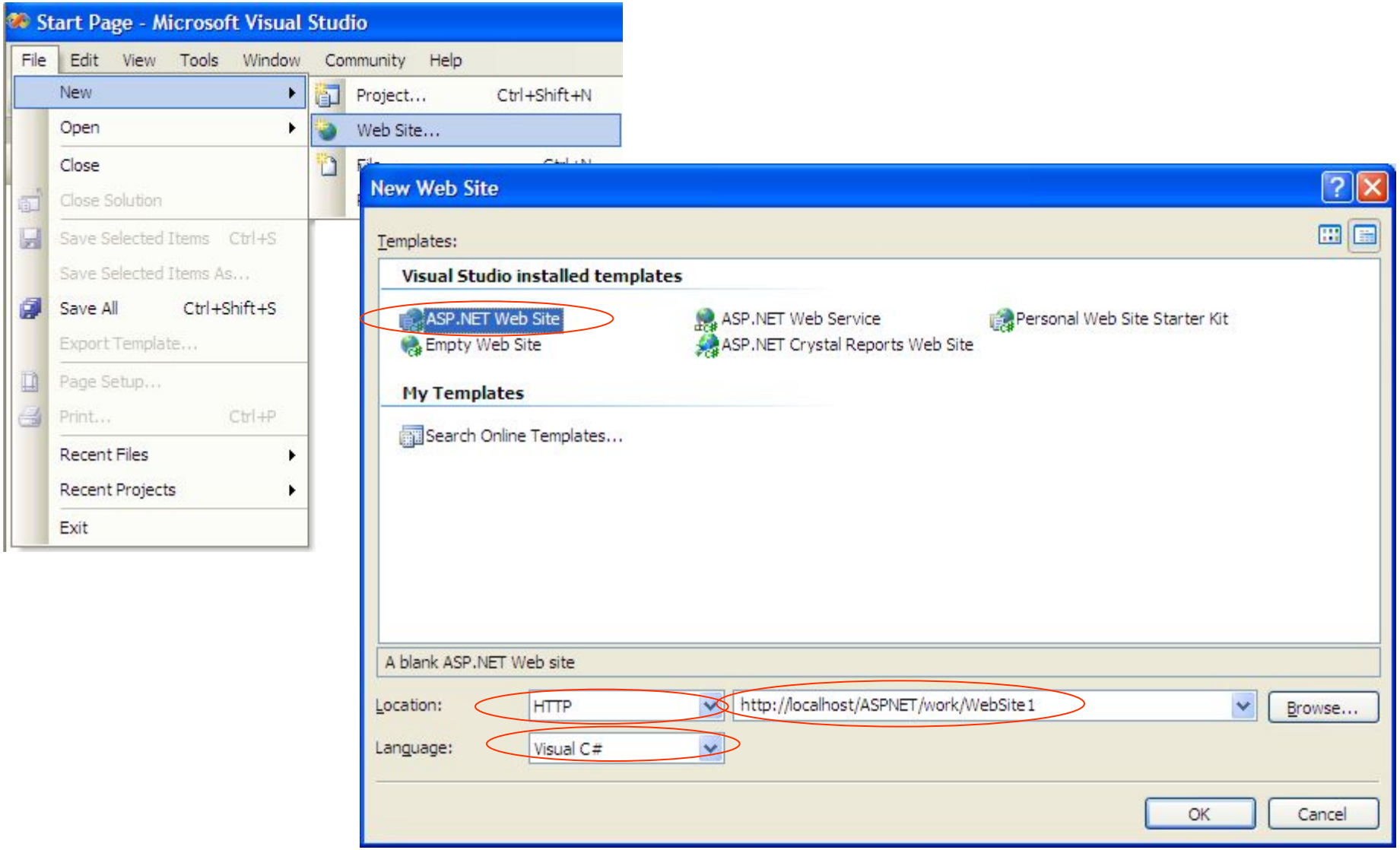
ASP.NET-сторінка з вбудованим кодом реалізації

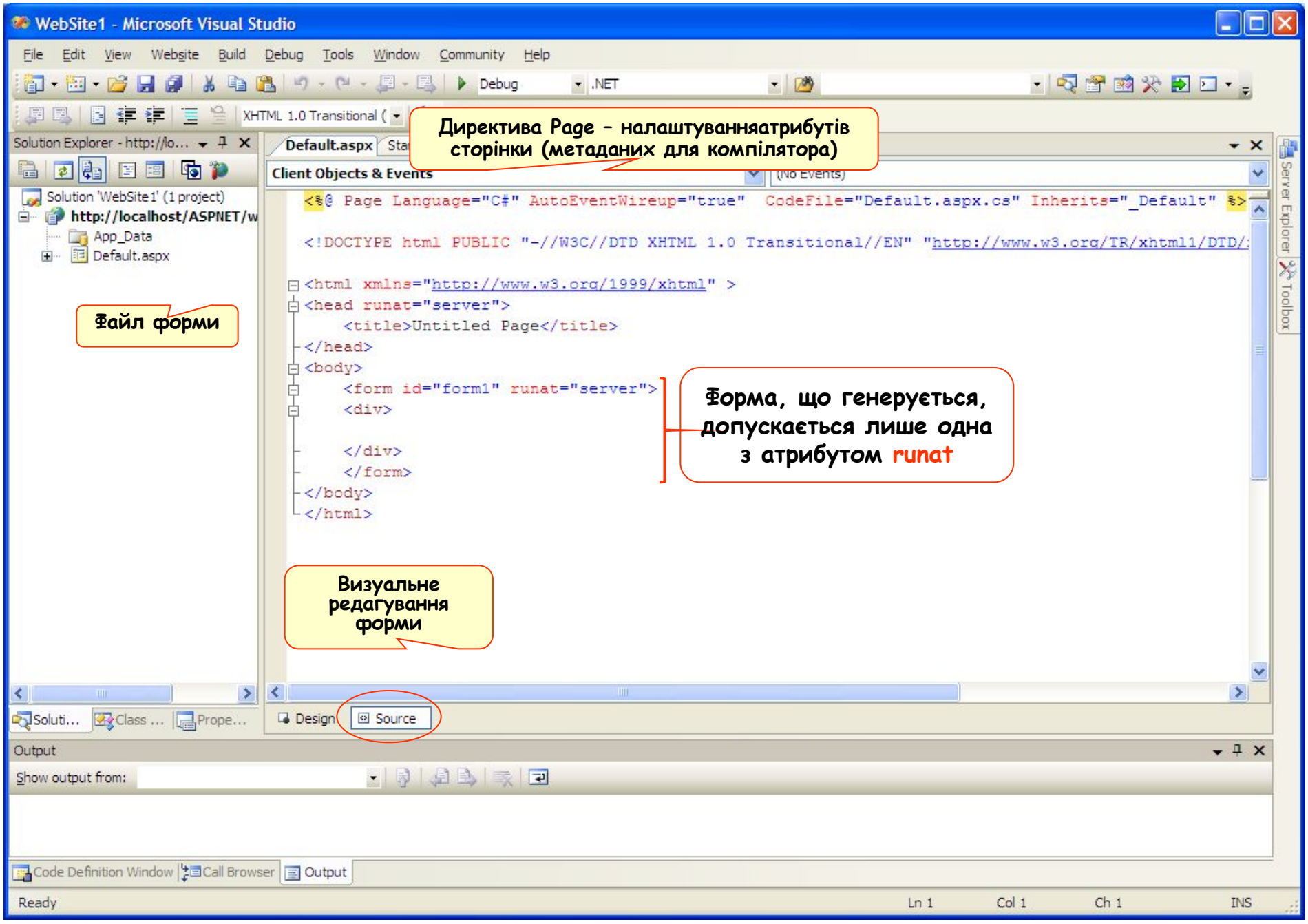
```
<!-- Розділ директив -->
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Розділ коду -->
<script runat="server">
    protected void Button1_Click(object sender, EventArgs e)
    {
        Label1.Text = "Результат: " + TextBox1.Text.ToUpper();
    }
</script>

<!-- Розділ користувацького інтерфейсу -->
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Однофайлова модель сторінки</title>
</head>
<body>
    <h1>Введіть рядок: </h1>
    <form id="form1" runat="server">
    <div>
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <asp:Button ID="Button1" runat="server" Text="Button" onclick="Button1_Click" />
        <br />
        <asp:Label ID="Label1" runat="server" Text="Результат: "></asp:Label>
    </div>
    </form>
</body>
</html>
```



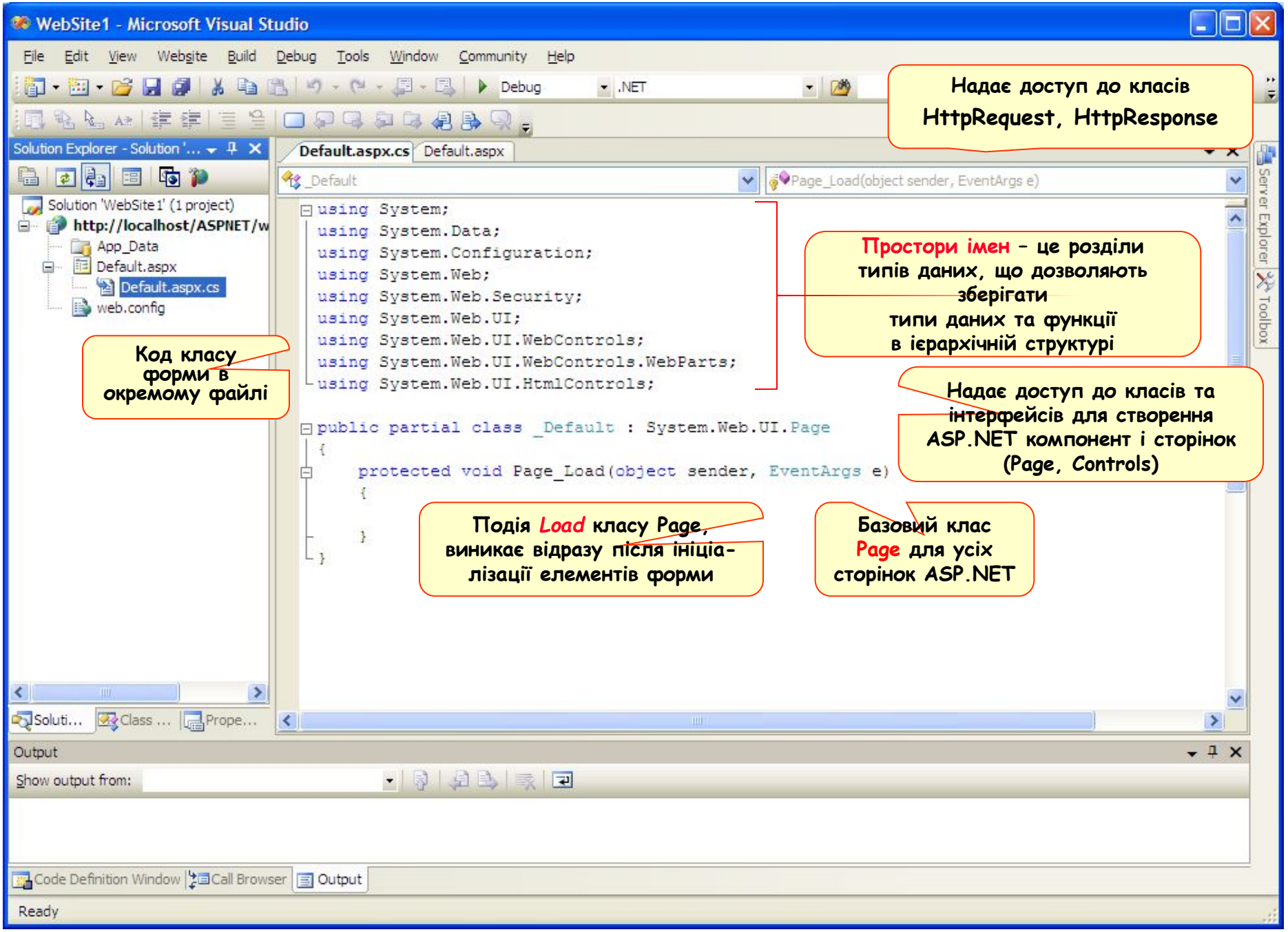


Створення Web сайту

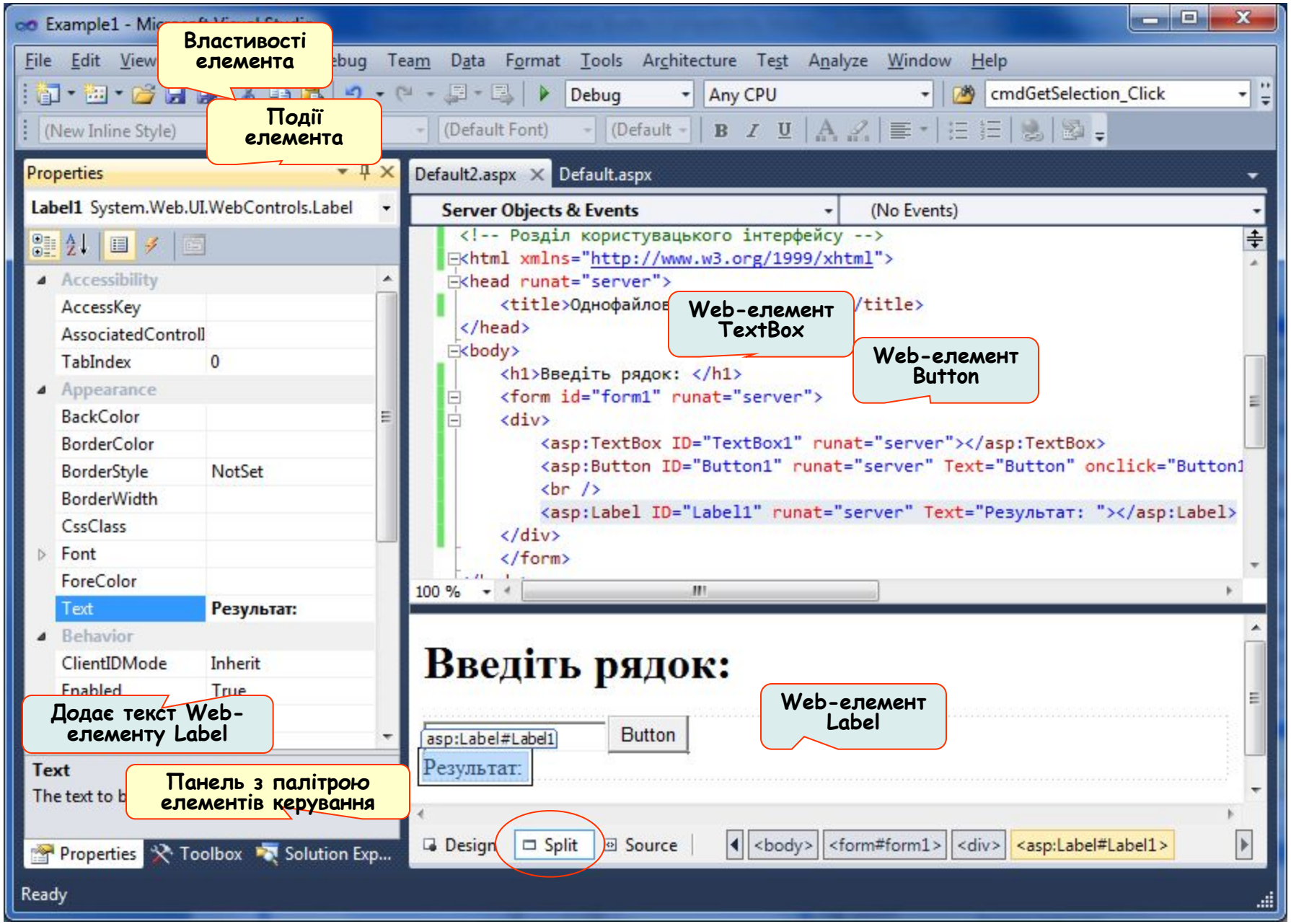



WebSite1 - Microsoft Visual Studio
File Edit View Website Build Debug Tools Window Community Help
Debug
.NET
XHTML 1.0 Transitional (
Директива Page – налаштуванняатрибутів сторінки (метаданих для компілятора)
Solution Explorer - http://lo...
Solution 'WebSite1' (1 project)
http://localhost/ASPNET/w
App_Data
Default.aspx
Файл форми
Default.aspx
Client Objects & Events
(No Events)
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
<title>Untitled Page</title>
</head>
<body>
<form id="form1" runat="server">

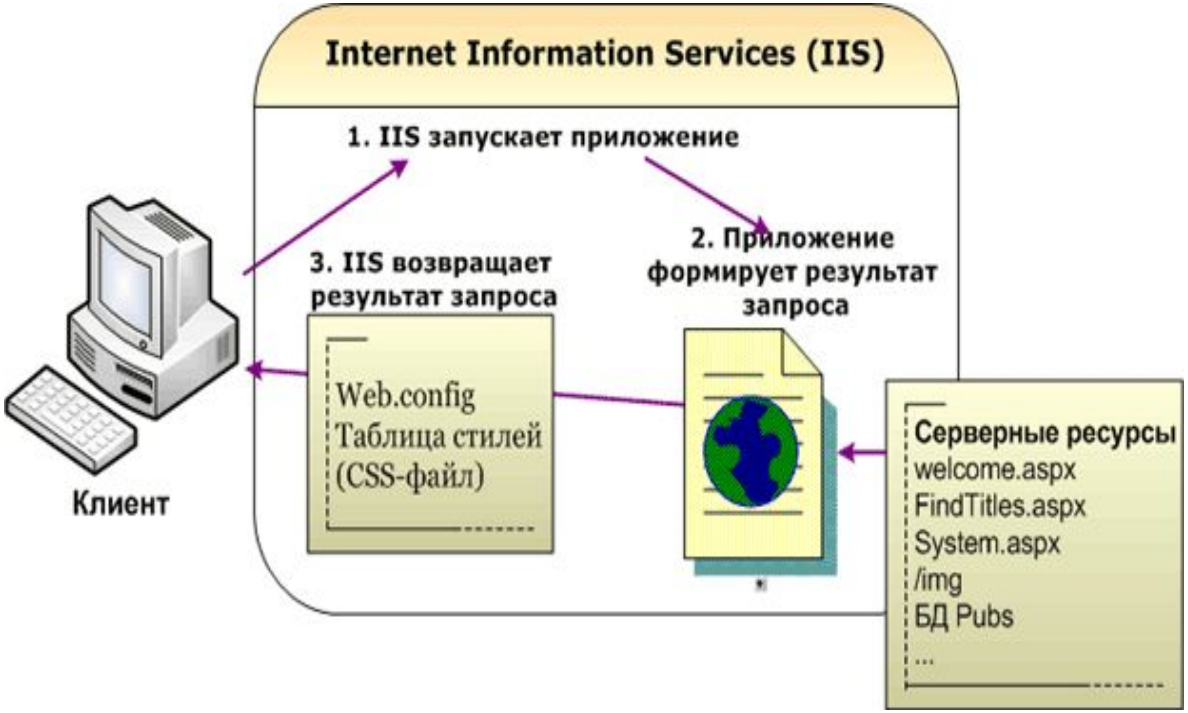
</form>
</body>
</html>
Форма, що генерується, допускається лише одна з атрибутом runat
Визуальне редагування форми
Design
Source
Output
Show output from:
Code Definition Window
Call Browser
Output
Ready
Ln 1
Col 1
Ch 1
INS


WebSite1 - Microsoft Visual Studio
File Edit View Website Build Debug Tools Window Community Help
Debug .NET
Solution Explorer - Solution '...
Solution 'WebSite1' (1 project)
http://localhost/ASPNET/w
App_Data
Default.aspx
Default.aspx.cs
web.config
Default.aspx.cs Default.aspx
_Default
Page_Load(object sender, EventArgs e)
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
public partial class _Default : System.Web.UI.Page
{
protected void Page_Load(object sender, EventArgs e)
{
}
}
Надає доступ до класів HttpRequest, HttpResponse
Простори імен – це розділи типів даних, що дозволяють зберігати типи даних та функції в ієрархічній структурі
Код класу форми в окремому файлі
Надає доступ до класів та інтерфейсів для створення ASP.NET компонент і сторінок (Page, Controls)
Подія Load класу Page, виникає відразу після ініціа-лізації елементів форми
Базовий клас Page для усіх сторінок ASP.NET
Server Explorer
Toolbox
Output
Show output from:
Code Definition Window
Call Browser
Output
Ready

Дизайн сторінки



Типовий сценарій взаємодії елементів Web-застосування з клієнтом



Клас Page. Його властивості та методи

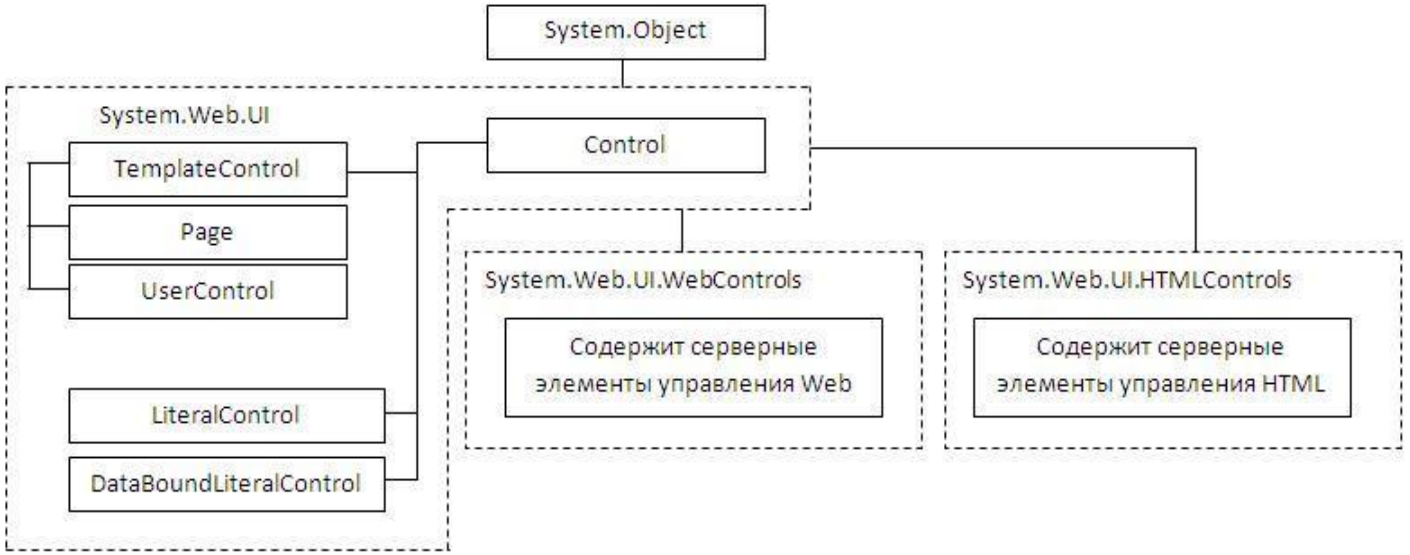
- **Усі Web-сторінки (Web-форми) є екземплярами класу `System.Web.UI.Page`**
- **Web-сторінка є контейнером елементів керування**

Application	Зберігання даних у змінних стану застосування.
Cache	Керування кешуванням відгуків на сервері.
Controls	Отримання елементів керування сторінки.
Request	Читання запитів та отримання з поточного запиту об'єктів Browser, Cookies, Files, ClientCertificates.
Response	Запис тексту та даних у відгук, отримання з поточного відгуку об'єктів Cache, Cookies, Output.
Server	Обробка запитів та відгуків, допоміжні методи кодування та декодування URL.
Session	Збереження елементів даних у змінних стану сеанса.
Trace	Включення і виключення трасування, запис подій трасування у журнал.

Групи елементів керування

- Серверні елементи керування HTML
  - `HtmlAnchor` (відповідає `<a>`), `HtmlSelect` (`<select>`) тощо
- Серверні елементи керування Web
  - `HyperLink`, `ListBox`, `Button` тощо
- Повнофункціональні елементи керування
  - `Calendar`, `AdRotator` та `TreeView`
- Елементи керування даними
  - Розширені елементи керування `GridView`, `DetailsView` та елементи керування джерелами даних, напр., `SqlDataSource`
- Елементи керування перевіркою достовірності
- Користувацькі елементи керування
- Елементи керування навігацією тощо

Ієрархія серверних елементів керування



Властивість	Опис
ClientID	Повертає ідентифікатор елемента, який є унікальним ім'ям, створеним ASP.NET при створенні екземпляра сторінки
Controls	Повертає набір дочірніх елементів керування
EnableViewState	Повертає або встановлює булеве значення, яке вказує на те, чи повинен елемент керування підтримувати стан виду за допомогою зворотніх посилань своєї батьківської сторінки. За промовчанням - true
ID	Повертає або встановлює ідентифікатор елементу керування
Page	Повертає посилання на об'єкт батьківської сторінки
Parent	Повертає посилання на батька елемента керування, яким може бути сторінка або інший контейнерний елемент керування
Visible	Повертає або встановлює булеве значення, яке вказує на те, чи потрібно генерувати елемент керування

Серверні елементи керування HTML

- Безпосередньо відображаються у вигляді елементів розмітки HTML
- Успадковують від класу `System.Web.UI.HtmlControls.HtmlControl`
- Звичайний тег

```
<input id="Reset1" type="reset" value="reset"/>
```

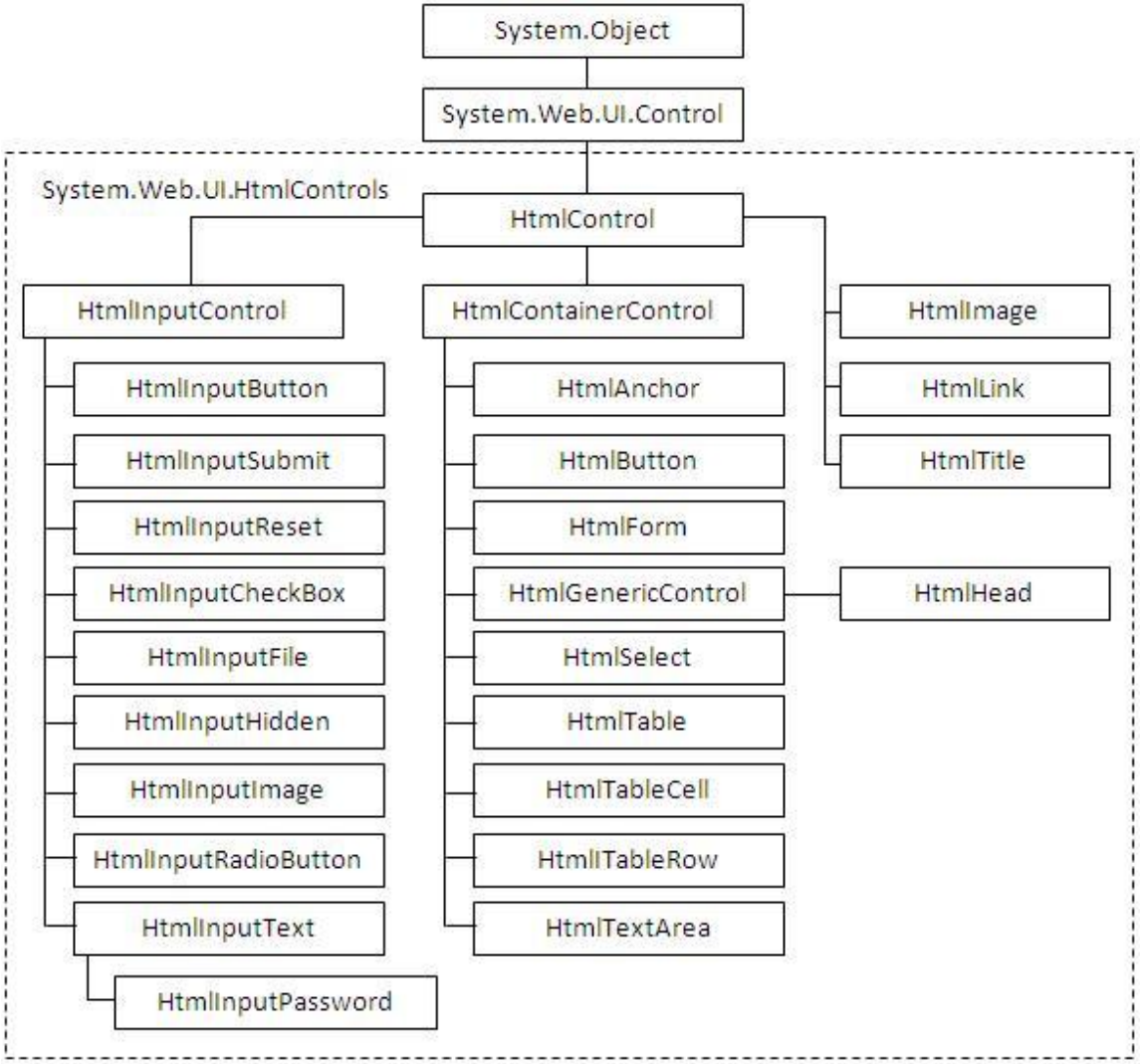
- Серверний елемент керування HTML

```
<input id="Reset1" runat="server" type="reset"
       value="reset" />
```

- Атрибут `runat` дозволяє ASP.NET обробляти серверні елементи керування HTML та транслювати їх у екземпляри відповідного класу .NET

Ієрархія серверних елементів керування HTML

- **`HtmlInputControl` - елементи керування введенням, що допускають взаємодію з користувачем**
- **`HtmlContainerControl` – будь-який HTML дескриптор**



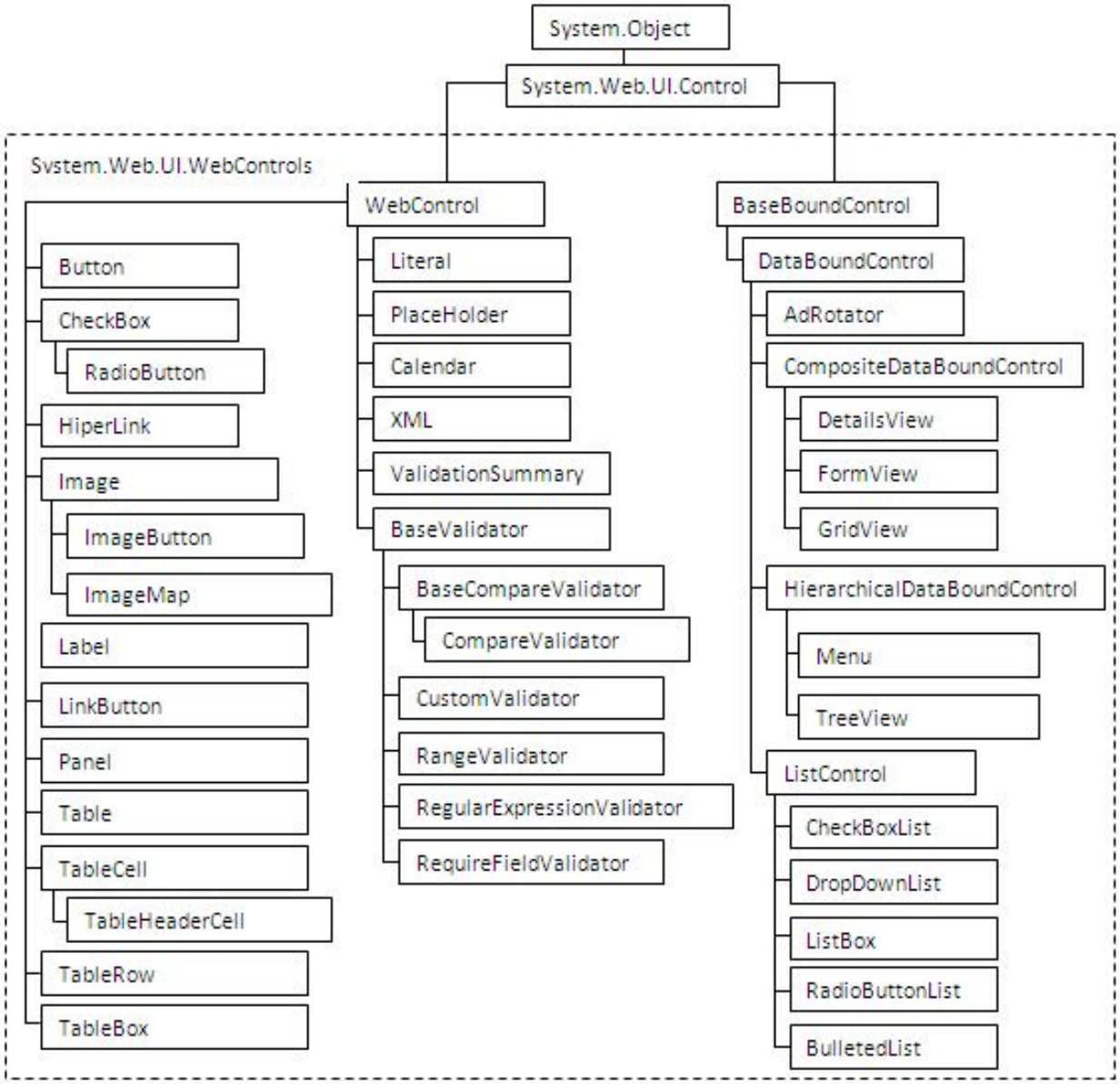
Стандартні або серверні елементи керування Web

- **Прив'язані не до розмітки, а до функціональності, яку потрібно забезпечити**
- **Успадковують від класу `System.Web.UI.WebControls.WebControl`**

```
<asp:Label ID="Label1" runat="server" Text = "Hello World">
  </asp: Label>
<asp:Label ID="Label1" runat="server" Text="Hello World"/>
```

Властивість	Опис
BackColor	Колір фону
BorderColor	Колір границі елемента керування
BorderStyle	Стиль границі — суцільний, пунктир, точковий, подвійний і інші
BorderWidth	Ширина границі
Enabled	Активність. Якщо false, неможна вводити дані, не отримує фокус
Font	Шрифт, складається з кількох атрибутів
ForeColor	Колір, яким відображається текст
Height	Висота елемента
Width	Ширина элемента
Visible	Чи видно елемент керування
ToolTip	Текст вікна підсказки

Ієрархія серверних елементів керування Web



Базові класи серверних елементів керування Web

Елемент керування ASP.NET	Відповідний тег HTML	Призначення
<asp:Label>		Відобразити текст
<asp:ListBox>	<select>	Список вибору
<asp:DropDownList>	<select>	Випадаючий список
<asp:TextBox>	<input type="text"> <input type="password"> <textarea>	Поле редагування
<asp:HiddenField>	<input type="hidden">	Невидиме поле
<asp:RadioButton>, <asp:RadioButtonList>	<input type="radio">	Перемикач, список перемикачів
<asp:CheckBox>, <asp:CheckBoxList>	<input type="checkbox">	Прапорець, список прапорців
<asp:Button>	<input type="button"> <input type="submit">	Командна кнопка
<asp:Image>	<IMG>	Зображення
<asp:ImageButton>	<INPUT TYPE="IMAGE">	Кнопка-зображення
<asp:Table>	<table>	Таблиця
<asp:Panel>		Контейнер
<asp:BulletedList>	,	Маркований список
<asp:HyperLink>	<a href>	Гіперпосилання

Відповіднності між елементами керування ASP.NET та тегами HTML

- ```
  <asp:TextBox runat="server" ID="Textbox1"
    Text="This is a test" ForeColor="red"
  BackColor="lightyellow" Width="250px"
    Font-Name="Verdana" Font-Bold="True"
    Font-Size="20" />
  ```
- ```
  <input name="Textbox1" type="text"
    value="This is a test" id="Textbox1"
  style="color:Red;background-color:LightYellow;
       font-family:Verdana;font-size:20pt;
       font-weight:bold;width:250px;" />
  ```
  - Елемент керування оголошується з використанням імені класу (`TextBox`) замість імені дескриптора HTML (`input`)
  - Вміст за промовчанням встановлюється за допомогою властивості `Text`, а не атрибуту `value`
  - Атрибути стилів встановлюються через прямі властивості замість групування їх в одному атрибуті стилю

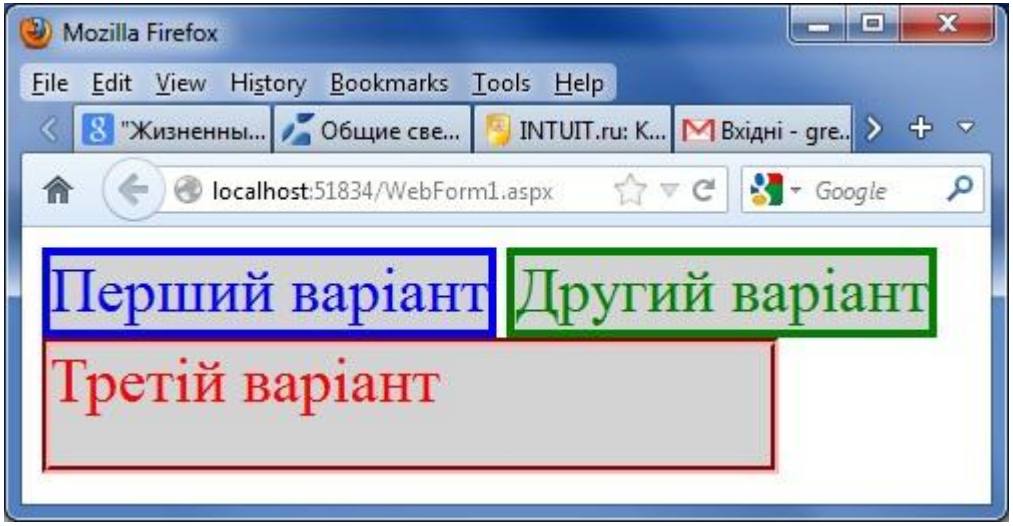
Елемент керування Label

Опис елемента Label у файлі WebForm1.aspx:

```
<asp:Label id="Label1" runat="server" Font-Size=20 ForeColor="blue" BackColor="lightgray" BorderWidth=4 >Перший варіант </asp:Label>
<asp:Label id="Label2" runat="server" Font-Size=20 ForeColor="green" BackColor="lightgray" BorderWidth=4  Text="Другий варіант"/>
```

Створення елемента Label прямо у програмі:

```
protected void Page_Load(object sender, EventArgs e)
{
    Label ShopNews = new Label();
    ShopNews.Text = "Третій варіант";
    ShopNews.Font.Size = 20;
    ShopNews.ForeColor = Color.Red;
    ShopNews.BackColor = Color.LightGray;
    ShopNews.BorderWidth = 4;
    ShopNews.BorderStyle = BorderStyle.Groove;
    ShopNews.Height = 50;
    ShopNews.Width = 300;

    form1.Controls.Add(ShopNews);
}
```



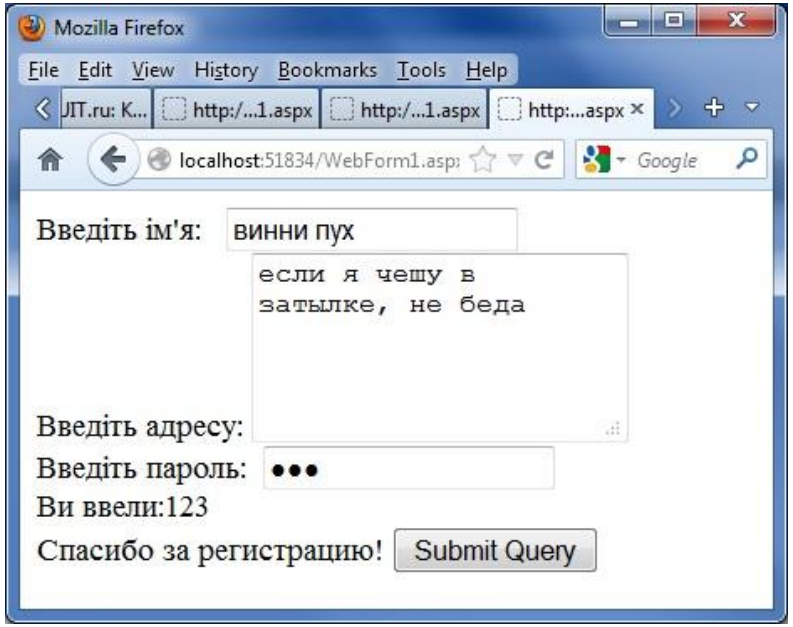
Елемент керування TextBox

Файл WebForm1.aspx:

```
    <form id="form1" runat="server">
    <div>
            <asp:Label ID="lblName" runat="server" Text="Введіть ім'я: "></asp:Label>
 <asp:TextBox ID="txtName" runat="server" style="margin-bottom: 0px"></asp:TextBox>
            <br />
            <asp:Label ID="lblAddress" runat="server" Text="Введіть адресу:"></asp:Label>
            <asp:TextBox ID="txtAddress" runat="server" TextMode="multiline" Rows="5"
Wrap="true"></asp:TextBox>
            <br />
            <asp:Label ID="lblPassword" runat="server" Text="Введіть пароль:"></asp:Label>
 <asp:TextBox ID="txtPassword" runat="server" TextMode="password"></asp:TextBox>
                    <br />
            <asp:Label ID="Label1" runat="server" Text="Результат: "></asp:Label>
           <input type="Submit"><br />
                    </div>
     </form>
```

Файл WebForm1.aspx.cs:

```
namespace WebApplication
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (txtName.Text != "")
                Label1.Text = "Ви ввели ім'я:" + txtName.Text;
            if (txtAddress.Text != "")
                Label1.Text = "Ви ввели адресу:" +
              txtAddress.Text;
            if (txtPassword.Text != "")
                Label1.Text = "Ви ввели:" +
                  txtPassword.Text + "<br>Спасибі за регистрацію!";
        }
    }
}
```



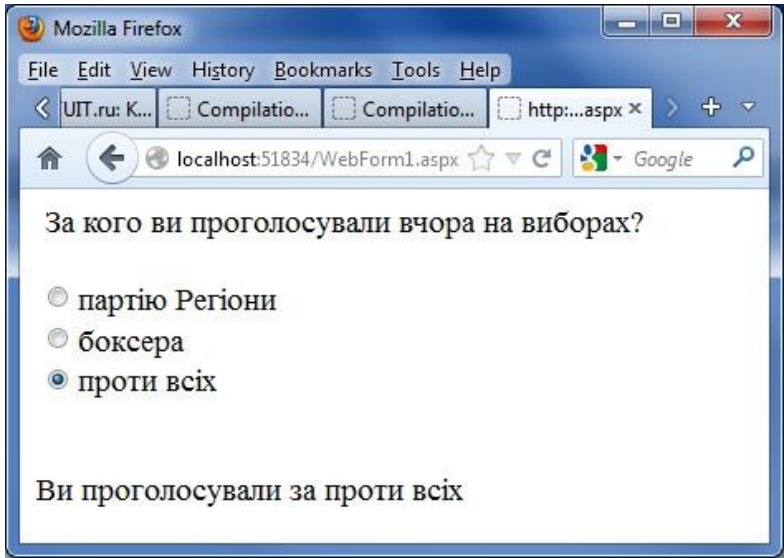
Елемент керування RadioButton

Файл WebForm1.aspx:

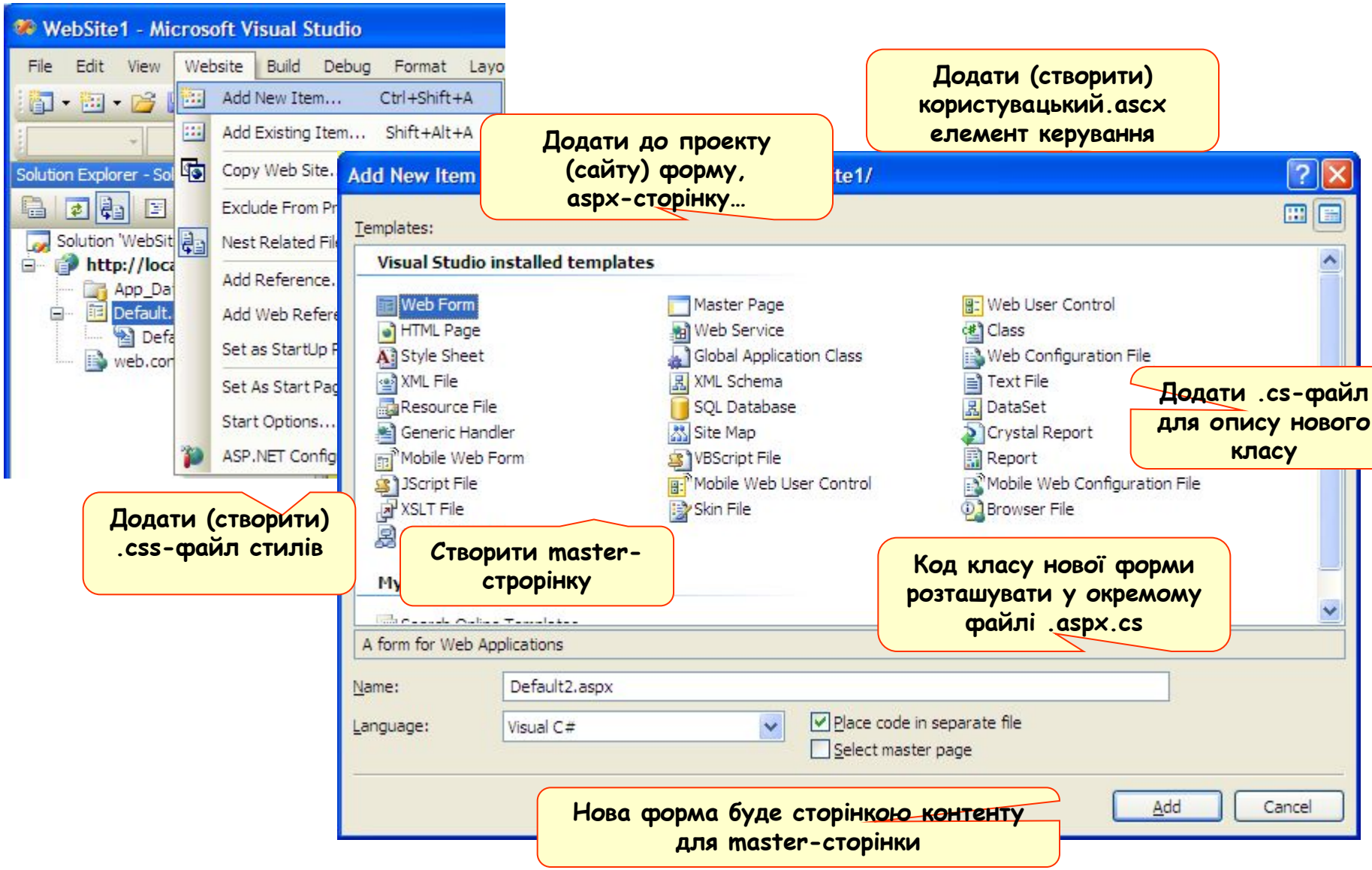
```
    <form runat="server" id="voting">
    <div>
 За кого ви проголосували вчора на виборах?<br />
            <br />
            <asp:RadioButton id="option1" runat="server" Text="партію Регіони"
OnCheckedChanged="option1_CheckedChanged" AutoPostBack = true GroupName="Choice" /><br />
            <asp:RadioButton id="option2" runat="server" Text="боксера"
OnCheckedChanged="option1_CheckedChanged" AutoPostBack = true GroupName="Choice" /> <br />
            <asp:RadioButton id="option3" runat="server" Text="проти всіх"
OnCheckedChanged="option1_CheckedChanged" AutoPostBack = true GroupName="Choice"/><br />
<br /><br />
</form>
<asp:Label id="Message" runat="server" />
```

Файл WebForm1.aspx.cs:

```
protected void option1_CheckedChanged(object sender, EventArgs e)
 {
     if (option1.Checked)
         Message.Text = "Ви проголосували за " + option1.Text;
     if (option2.Checked)
         Message.Text = "Ви проголосували за " + option2.Text;
     if (option3.Checked)
         Message.Text = "Ви проголосували за " + option3.Text;
 }
```



Зміна складу сайту



Використання MasterPage

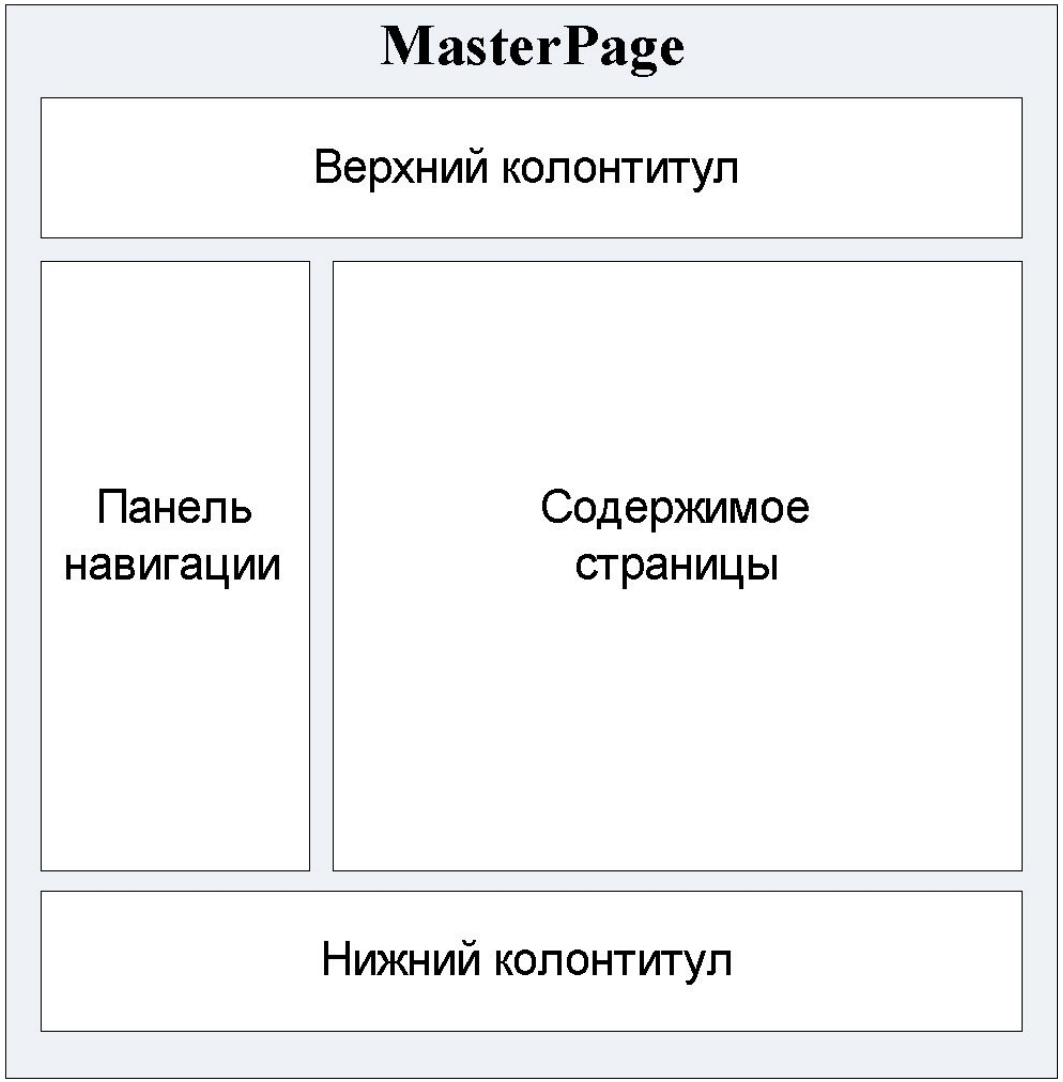
- **Майстер-сторінка** *(master pages)* – це шаблон сторінки, який може містити будь-які елементи, допустимі для звичайної сторінки, а також програмний код.

```
<%--Звичайна сторінка ASP.NET--%>
<%@ Page Title="" Language="C#" AutoEventWireup="true"
        CodeFile="Default.aspx.cs" Inherits="_Default" %>

<%--Майстер-сторінка ASP.NET--%>
<%@ Master Language="C#" AutoEventWireup="true"
        CodeFile="MainMasterPage.master.cs" Inherits="MainMasterPage" %>
```

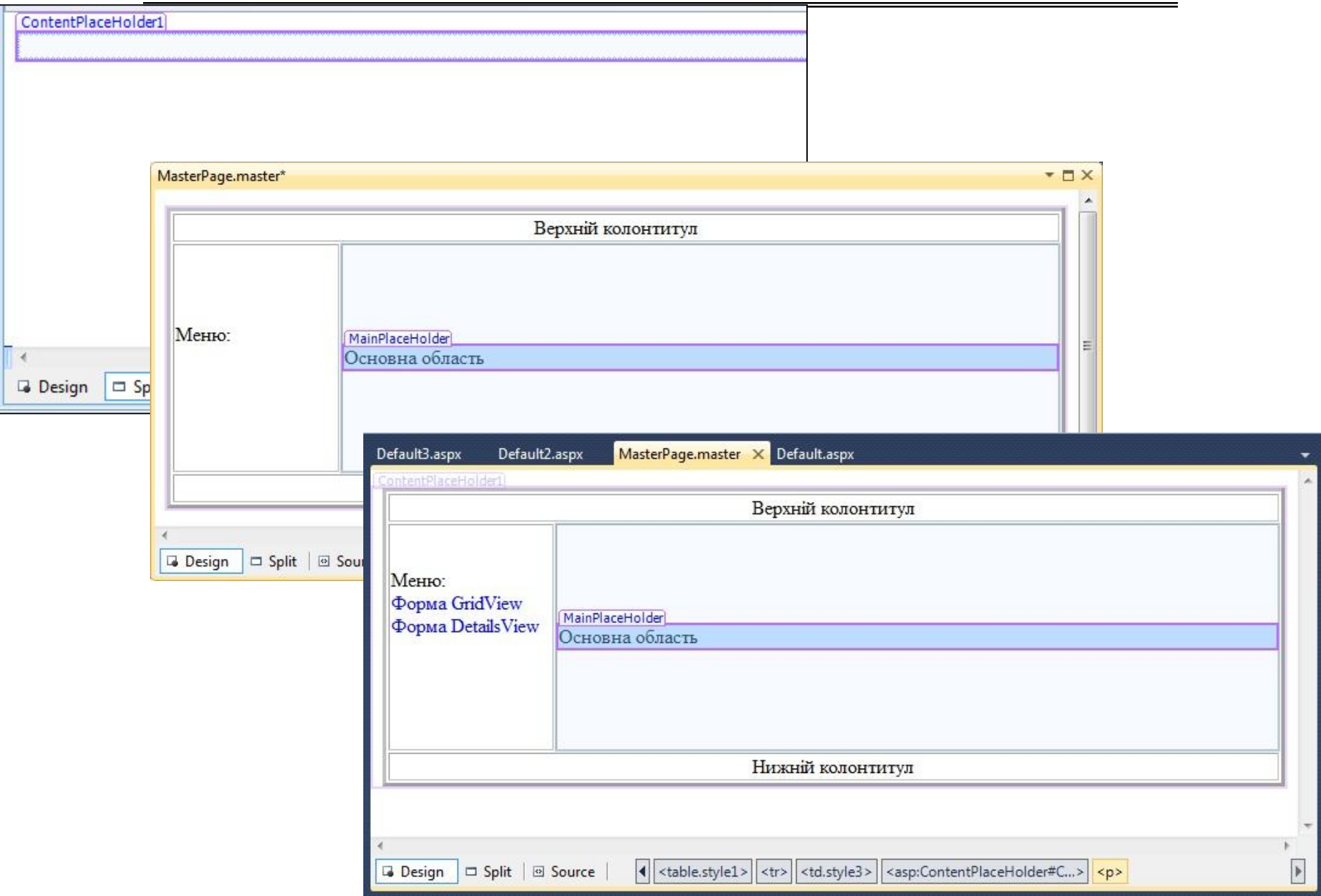
- **Сторінка вмісту** *(content pages)* – містить допустимі елементи керування, за допомогою яких визначає вміст, яким заповнюються спеціальні області майстер-сторінок

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default2.aspx.cs"
Inherits="Default2" MasterPageFile="~/MasterPage.master" %>
<asp:Content ContentPlaceHolderID="MainPlaceHolder" ID="Content1" runat="server">

</asp:Content>
```

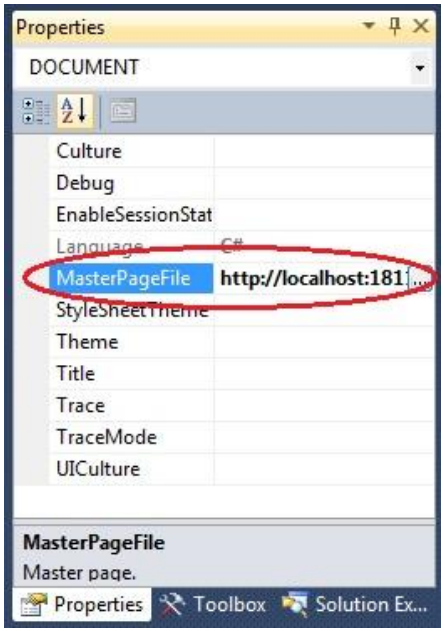
Структура MasterPage



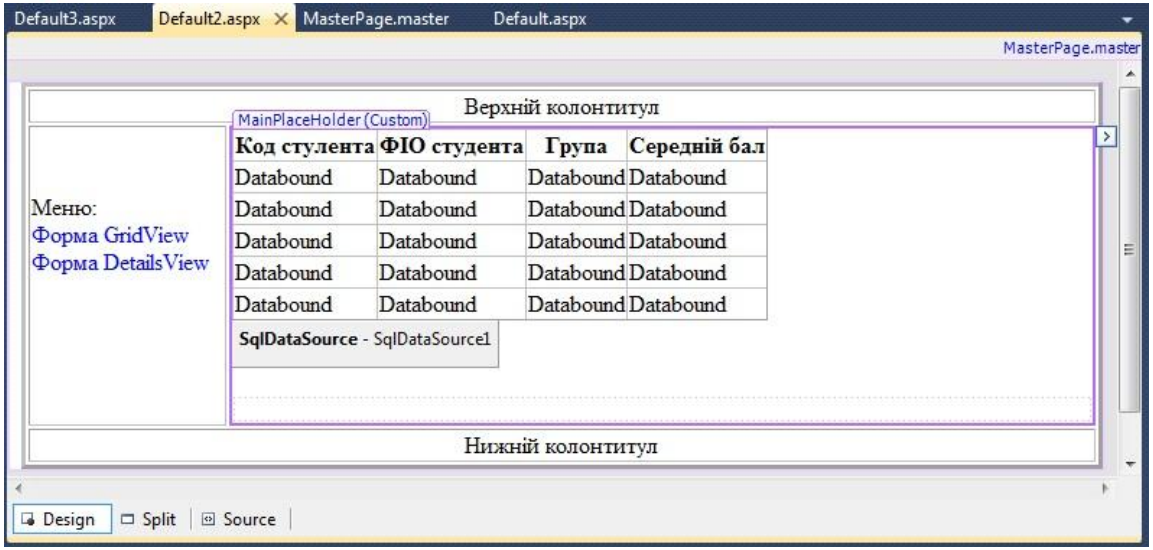
Приклад використання MasterPage



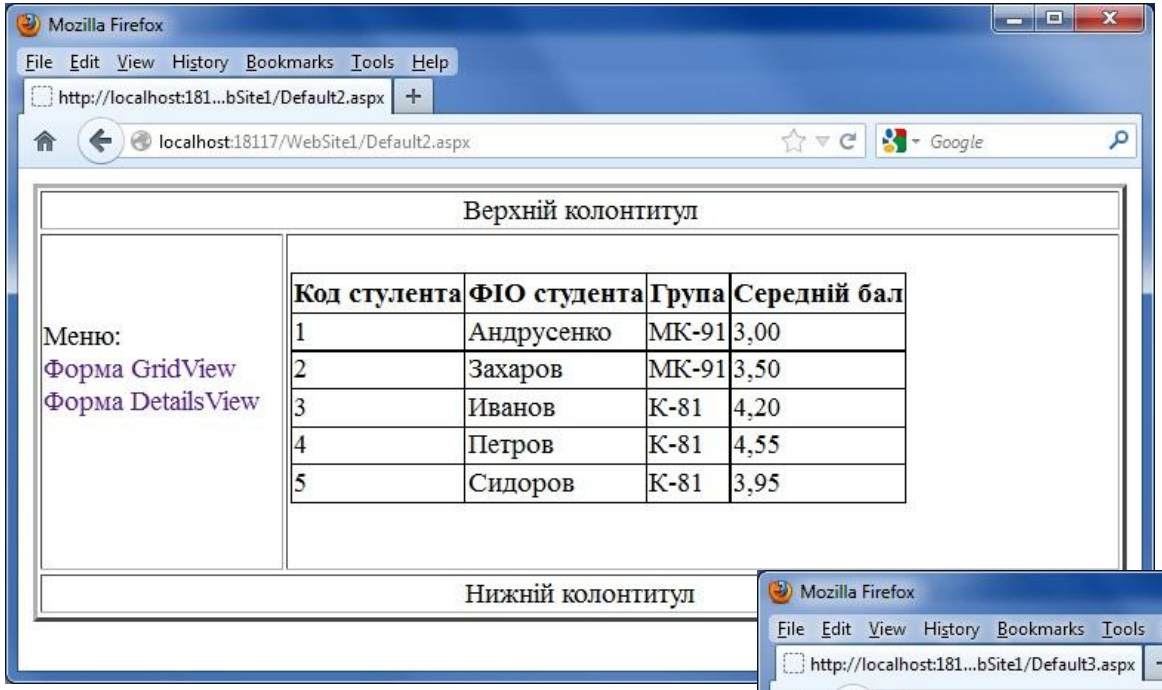
Приклад використання MasterPage

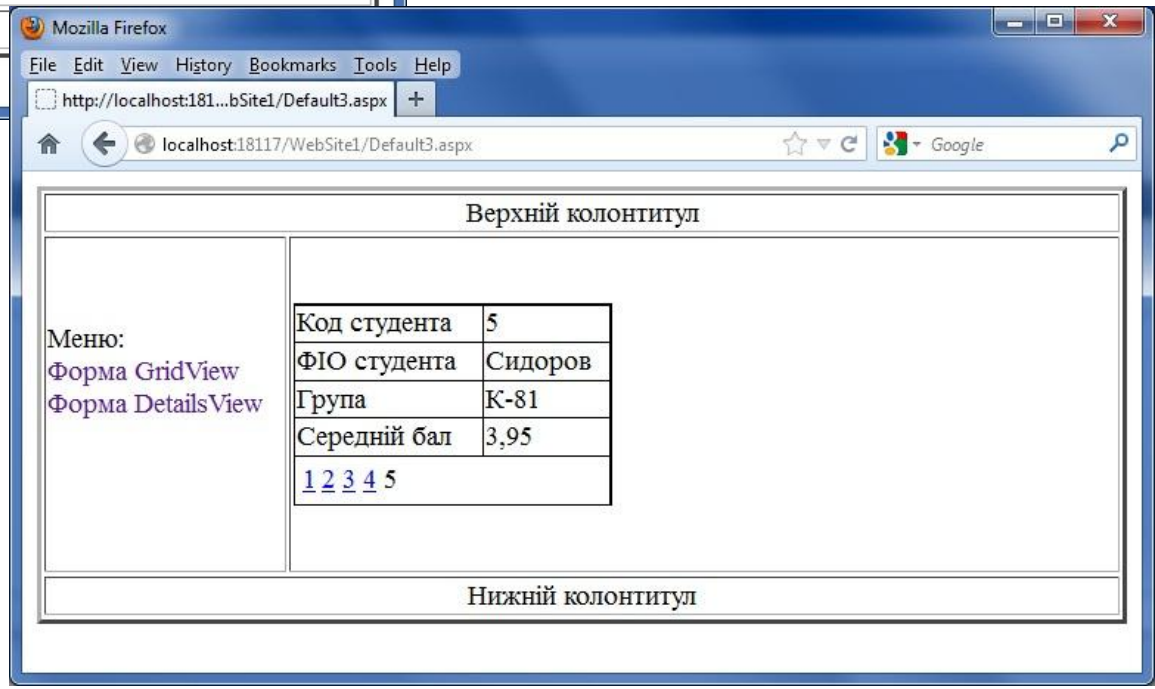



```
<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true"
CodeFile="Default2.aspx.cs" Inherits="Default2" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="MainPlaceHolder" Runat="Server">
    <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
    DataSourceID="SqlDataSource1" style="margin-right: 53px">
    <Columns>
        <asp:BoundField DataField="Student_ID" HeaderText="Код стулента"
            SortExpression="Student_ID" />
        <asp:BoundField DataField="St_Name" HeaderText="ФІО студента"
            SortExpression="St_Name" />
        <asp:BoundField DataField="St_group" HeaderText="Група"
            SortExpression="St_group" />
        <asp:BoundField DataField="St_rating" HeaderText="Середній бал"
            SortExpression="St_rating" />
    </Columns>
</asp:GridView>
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%$ ConnectionStrings:LabDBConnectionString %>"
    SelectCommand="SELECT [Student_ID], [St_Name], [St_group], [St_rating] FROM [Student]">
</asp:SqlDataSource>
<p>
</p>
</asp:Content>
```



Приклад використання MasterPage





Література

- *Мэтью Мак-Дональд, Марио Шпушта.* Microsoft ASP.NET 3.5 с примерами на C# 2008 и Silverlight 2 для профессионалов.: Пер. с англ. – М.: ООО "И.Д. Вильямс", 2009. – 1408 с.
- *Рихтер Дж.* CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C#. 3-е изд. – СПб.: Питер, 2012. – 928 с.
- *Троелсен Э.* Язык программирования C# 5.0 и платформа .NET 4.5. 6-е изд., – М.: ООО "И.Д. Вильямс", 2013. - 1312 с.
- Корисні ресурси
  - http://msdn.microsoft.com/ru-ru/library/
  - http://habrahabr.ru/post/165597/

Дякую за увагу

Крос-платформне програмування

Лекція 8

Керування станом у ASP.NET

09 квітня, 2014

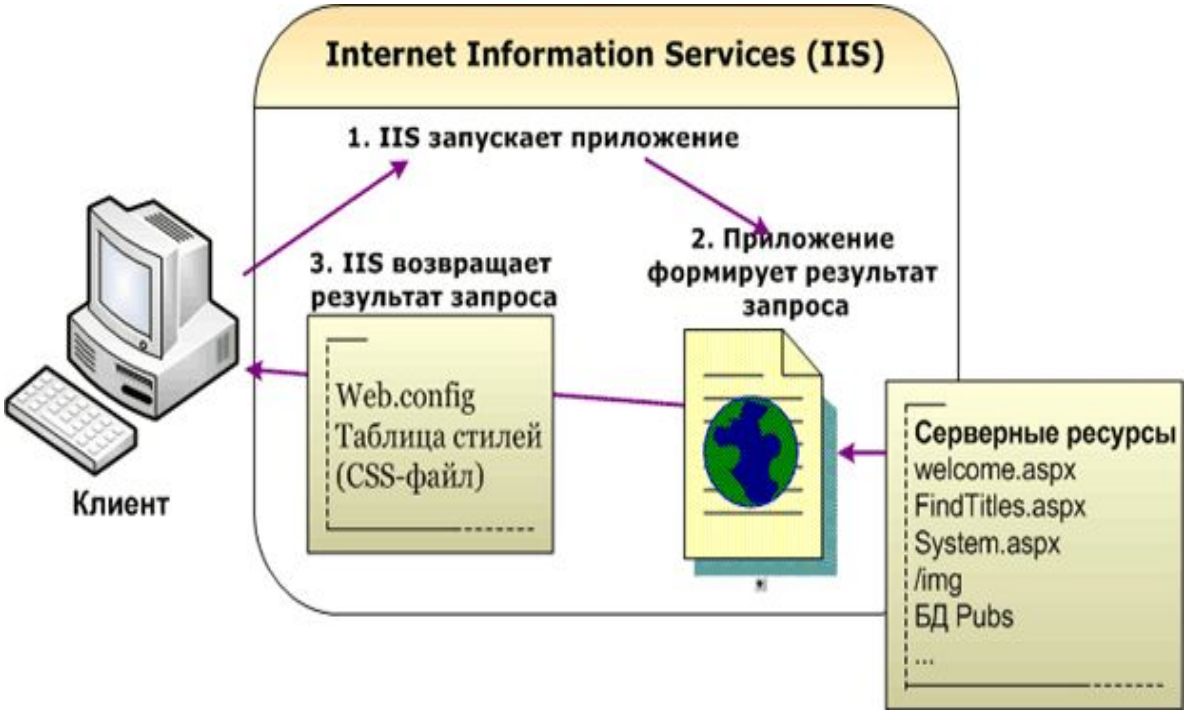
Примітка: слайди лекції підготовлені за матеріалами http://jskreator.narod.ru/proaspnet20cs/glance.htm

Порівняння Windows- та Web-застосувань

Windows-застосування	Web-застосування
Клієнт і застосування працюють на одному комп'ютері	Клієнт використовує свій комп'ютер, а застосування виконується на web-сервері
Один користовач – одна копія програми	Багато користувачів – одна копія програми (або декілька)
Набір об'єктів	Набір об'єктів
Виконання у середовищі CLR	Виконання у середовищі CLR
Стан об'єктів зберігається	Стан об'єктів очищується

Керування станом у ASP.NET

- HTTP – протокол без збереження інформації про стан (після першого веб-запиту клієнт відключається від сервера і механізм ASP.NET очищає об'єкти сторінки)
- Технології зберігання інформації між веб-запитами
  - стан перегляду або стан виду ViewState
  - рядок запиту
  - cookie
  - стан сеансу (Session)
  - використання об'єктів кешування



Колекція ViewState

- **Стан виду** *(ViewState)* **організований за принципом колекції типу словника у форматі "ім'я-значення"**
  - **Web-елементи керування використовують стан виду за промовчанням**
  - **Можна додавати у колекцію власні дані, зокрема прості типи даних та спеціальні об'єкти**
  - **Кожен елемент індексується за допомогою унікального рядкового імені**

```
ViewState["Name"] = "Іван";
ViewState["CurrentPage"] = 3;
```

  - **Усі елементи колекції зберігаються у вигляді об'єктів узагальненого типу, тому для отримати значення елементу виконується перетворення**
  - **При спробі відшукати значення, якого немає у колекції, буде видано виключення `NullReferenceException`**

```
string name;
if (ViewState["Name"] != null)
    name = (string)ViewState["Name"];
```

Зберігання колекції ViewState

- **Інформація стану виду зберігається як рядок у форматі Base64 у прихованих полях вводу**


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

</title></head>
<body>
    <form method="post" action="Default8.aspx" id="form1">
<div class="aspNetHidden">
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE" value="/wEPDwUKLTU2ODgwMzc3NWRkexl+WgokCtm3z24hTj2UgDha4Ub12oaY/K+pK4sVGWY=" />
</div>

<div class="aspNetHidden">

    <input type="hidden" name="__EVENTVALIDATION" id="__EVENTVALIDATION" value="/wEWAgLH+N+0AgKM54rGBhQxvyc5TI/XpPQhIJhR84ceSwWtUbgn0fCV3mYbLLmz" />
</div>
    <div>
        <input type="submit" name="Button1" value="Button" id="Button1" />
        <br />
        <span id="Label1">Hello, world!</span>
    </div>
    </form>
</body>
</html>
```

- **Перетворення рядка у форматі Base64 у масив ASCII-символів**

```
string viewStateString = "/wEPDwUKLTE2MjY5MTY1NQ9kFgICAw9kFgICAQ8PFgIeBFRleHQFDEhlbGxvIFdvcmxkIWRkZPsbiNOyNAufEt7OvNIbVYcGWHqf";
byte[] stringBytes = Convert.FromBase64String(viewStateString);

// Десериализация и отображение строки.
string decodedViewState = System.Text.Encoding.ASCII.GetString(stringBytes);
Label1.Text = decodedViewState;
```

Збереження об'єктів у стані виду

- Щоб клас піддавався серіалізації, він повинен:
  - мати атрибут Serializable
  - усі породжувані від нього класи теж повинні мати атрибут Serializable
  - усі індивідуальні змінні цього класу повинні мати типи даних, що піддаються серіалізації, а ті типи даних, що не піддаються серіалізації, повинні супроводжуватися атрибутом NonSerialized

```
[Serializable]
public class BookPage
{
    public int PageNum;
    public string Text;

    public BookPage(int pageNum, string text)
    {
        PageNum = pageNum;
        Text = text;
    }
}
```

- Збереження класу BookPage у стані виду

```
BookPage page = new BookPage(3, "Понеділок починається в суботу");
ViewState["CurrentPage"] = page;
```

- Отримання класу BookPage зі стану виду

```
cust = (BookPage)ViewState["CurrentPage"];
```

- Серіалізация - процес перетворення об'єкту у потік байтів

Використання стану виду

- **Стан виду краще не застосовувати:**
  - для збереження критично важливих даних, можливість зміни яких користувачем повинна бути повністю виключена
  - для збереження інформації, яка буде використовуватися кількома сторінками
  - для збереження надзвичайно великого обсягу інформації, щоб це вплинуло на швидкість передачі даних сторінки
- **Відключення стану виду для всієї сторінки та усіх її елементів керування**

```
<%@Page Language="C#" EnableViewState="false" ... %>
```

- **Захист інформації стану виду**
  - **Використання хеш-коду**

```
<%@Page EnableViewStateMAC="true" %>
```

  - **Застосування функції шифрування**

```
<%@Page ViewStateEncriptionMode="Always" %>
```

Використання рядка запиту

- **Рядок запиту** – частина URL-адреси, що знаходиться після знаку питання

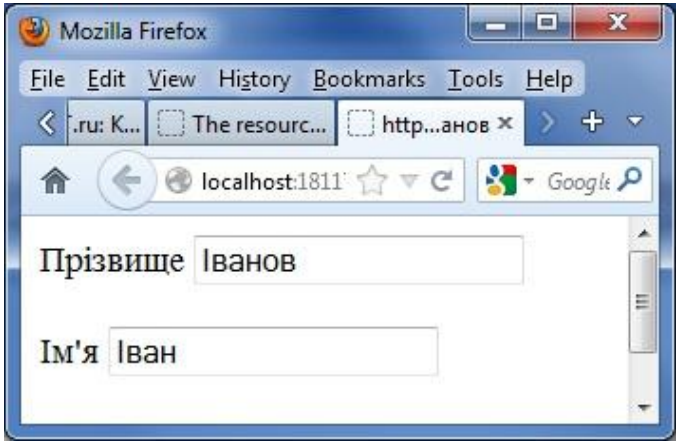
Пошукова система	URL для пошуку рядка «ASP.NET»
Яндекс	http://yandex.ua/yandsearch?text=ASP.NET&tld=ua&lr=143
Rambler	http://nova.rambler.ru/search?query=ASP.NET
Google	https://www.google.com.ua/search?hl=uk&biw=1120&bih=522&noj=1&sclient=psy-ab&q=ASP.NET&oq=ASP.NET&gs_l=serp.12...0.0.0.1266.0.0.0.0.0.0.0.0..0.0...0.0...1c.DUfOHoJVujs

- Відправка декількох параметрів

```
protected void Page_Load(object sender, EventArgs e)
{
    String FirstName = "Іван";
    String LastName = "Іванов";
    Response.Redirect("Default6.aspx?firstname=" + FirstName + "&lastname=" + LastName);
}
```

- Отримання результату

```
public partial class Default6 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        string FN = Request.QueryString["firstname"];
        string LN = Request.QueryString["lastname"];
        tb_FirstName.Text = FN;
        tb_LastName.Text  = LN;
    }
}
```



Обмеження використання рядка запиту

- Передача інформації у вигляді простих рядків, що містять лише припустимі для URL-адреси символи (літери, цифри, спец. символи $, -, _, ., +, !, *, ', (,), ,)
- Для неприпустимих символів використовувати URL-кодування
- Інформація візуально доступна для користувача та будь-кого, хто працює в Інтернеті
- Користувач може змінити рядок запиту і надати йому нові значення, які програма ніяк не очікує отримати і від яких вона не має захисту
- Багато браузерів мають свої обмеження щодо довжини URL-адреси, тому нобхідно виконувати перевірку на сумісність з браузерами

Файли cookie

- **Файли cookie або cookie-набори – це невеликі файли, які створюються на жорсткому диску клієнта або, якщо вони є тимчасовими, у пам'яті Веб-браузера**

- Перед використанням слід імпортувати простір імен для роботи з відповідними типами

```
using System.Net;
```

- Встановити cookie-набір (зберігається до моменту закриття вікна браузера)

```
// Створюється об’єкт cookie
HttpCookie cookie = new HttpCookie("Language");
// У ньому встановлюється значення
cookie["LanguagePref"] = "English";
// Об’єкт cookie додається до поточної Веб-відповіді
Response.Cookies.Add(cookie);
```

- Отримання cookie-набору

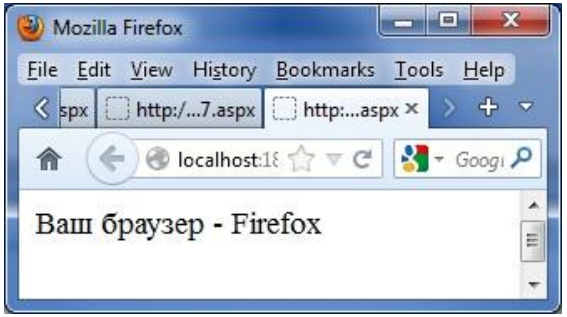
```
HttpCookie cookie = Request.Cookies["Language"];
// Перевіряємо, чи знайдено cookie с таким именем
string language;
if (cookie != null)
{
    language = cookie["LanguagePref"];
}
```

Об'єкт Request

Властивість	Опис
Browser	Визначення номера версії браузера, який запитується, можливість підтримки ним файлів cookie та іншої службової інформації
ClientCertificates	Аутентифікація клієнта
Cookies	Отримання файлів Cookies від клієнта
Files	Отримання файлів, переданих клієнтом
InputStream	Читання і запис переданого запиту у вигляді неструктурованих даних

```
if (!IsPostBack)
    if (Request.Browser.Cookies) Response.Write("Ваш Браузер підтримує cookies ");
    else Response.Write("Ваш Браузер не підтримує cookies ");
```

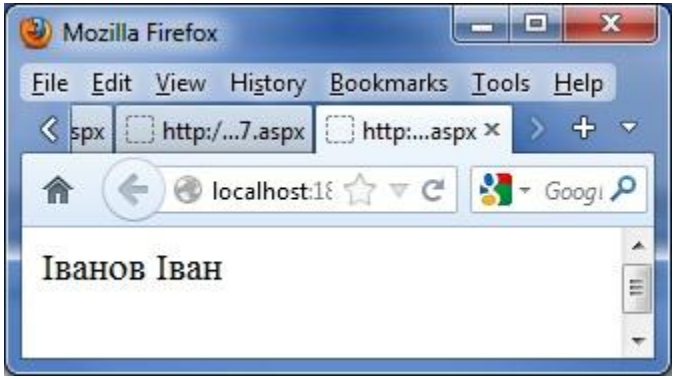

```
Response.Write("Ваш браузер - " + Request.Browser.Browser);
```



Об'єкти Response

Властивість	Опис
Cashe	Визначає кешування відгуків перед відправленням їх клієнтові.
Cookies	Дозволяє задавати вміст файлів cookie, переданих клієнтові.
Output	Дозволяє читати і записувати неструктуровані дані, що повертаються клієнту у вигляді відгуку.

```
if (!IsPostBack)
    if (Request.Browser.Cookies)
        if (Request.Cookies["UserName"] != null)
        {
            Session["User"] = Request.Cookies["UserName"].Value;
            Response.Write(Request.Cookies["UserName"].Value);
        }
        else
        {
            HttpCookie uname = new HttpCookie("UserName");
            uname.Value = "Іванов Іван";
            Response.Cookies.Add(uname);
        }
    else
        Response.Write("Ваш Браузер не підтримує cookies");
```



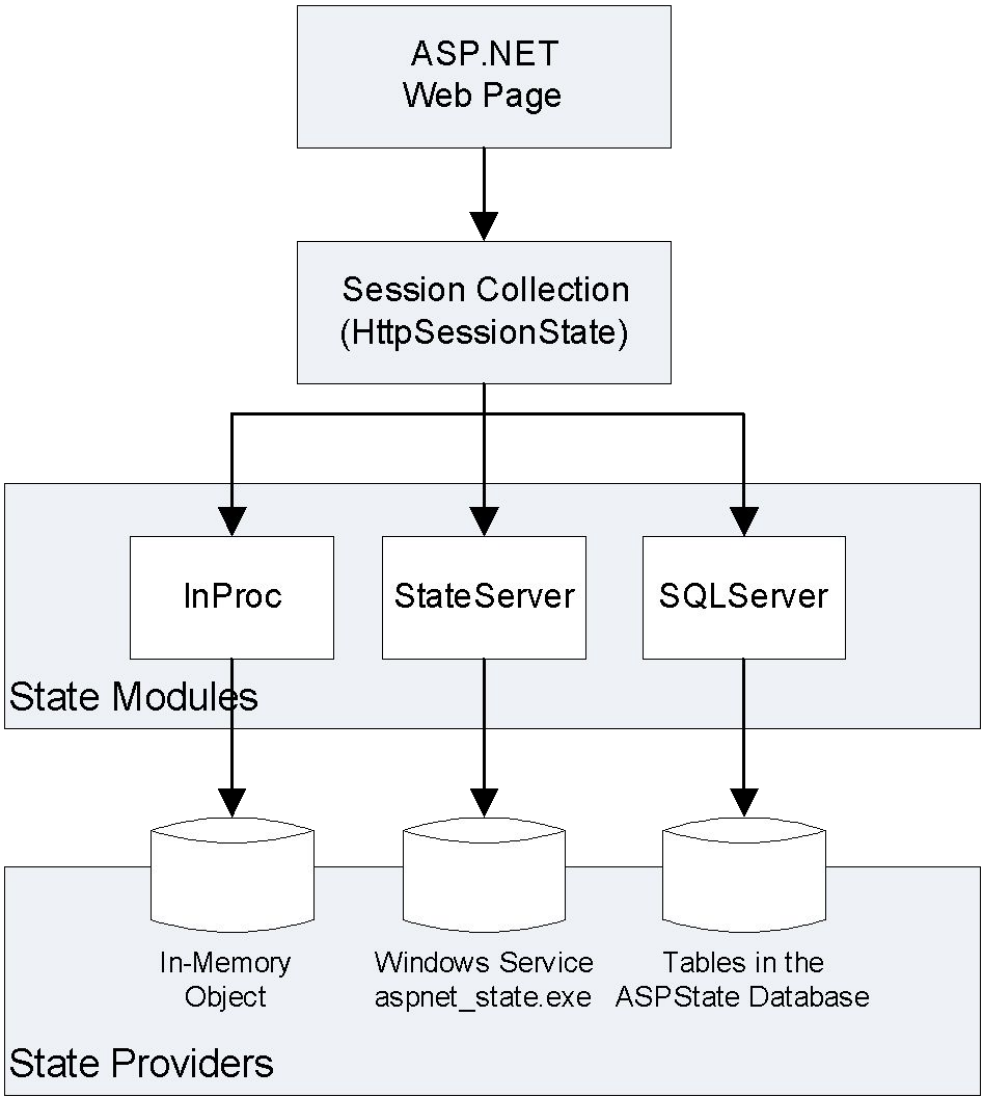
Порівняння опцій для керування станом

	Стан перегляду	Рядок запиту	Спеціальні cookie-набори
Допустимі типи даних	Всі піддаються серіалізації. NET-типи даних.	Обмежена кількість рядкових даних.	Рядкові дані.
Місце зберігання	Приховане поле на поточній Веб-сторінці.	Рядок URL-адреси у браузері.	Комп'ютер клієнта (у пам'яті або невеликому текстовому файлі).
Термін життя	Зберігається постійно для відправки даних на одну сторінку.	Втрачається, коли користувач вводить нову URL-адресу або закриває вікно браузера, але може зберігатися у посиланнях.	Визначається програмістом. Можуть використовуватися на декількох сторінках і зберігатися між відвідинами.
Контекст	Обмежується поточною сторінкою.	Обмежується цільової сторінкою.	Все ASP.NET-застосування цілком.
Безпека	За промовчанням є незахищеним, хоча можна примусово застосувати шифрування та хешування.	Доступна для перегляду і може бути змінена користувачем.	Ніяк не захищені і можуть змінюватися користувачем.
Складнощі, що впливають на продуктивність	При великій кількості інформації сповільнить процес передачі, але не позначиться на продуктивності сервера.	Ніяких, тому що кількість даних очевидна.	Ніяких, тому що кількість даних очевидна.
Застосовується для	Налаштування параметрів конкретної сторінки.	Відправки ідентифікаційного номера продукту зі сторінки каталогу на сторінку подробиць.	Визначення персоналізованих уподобань на Веб-сайті.

Стан сеанса

- Особливості
- Використовує заснованний на колекціях синтаксис
- Дозволяє зберігати інформацію на одній сторінці, а потім отримувати доступ до неї з іншої
- Підтримує об'єкти будь-якого типу
- Кожен клієнт, що отримує доступ до застосування, має свій сеанс та свою колекцію даних
- Керування сеансом не є частиною HTTP-стандарта
- Кожен сеанс має унікальний 120-бітний ідентифікатор, що передається між Web-сервером та клієнтом
- Постачальники сеанса – зовнішні компоненти, які зберігають стан (дані) сеансу

Архітектура стану сеанса у ASP.NET



- **Off – відключення функції керування станом сеанса для усіх сторінок у застосуванні**
- **InProc – зберігання інформації у поточному домені застосування**
- **StateServer – використання для керування станом окремої служби Windows**
- **SqlServer – використання для зберігання даних сеансу бази даних SQL Server**

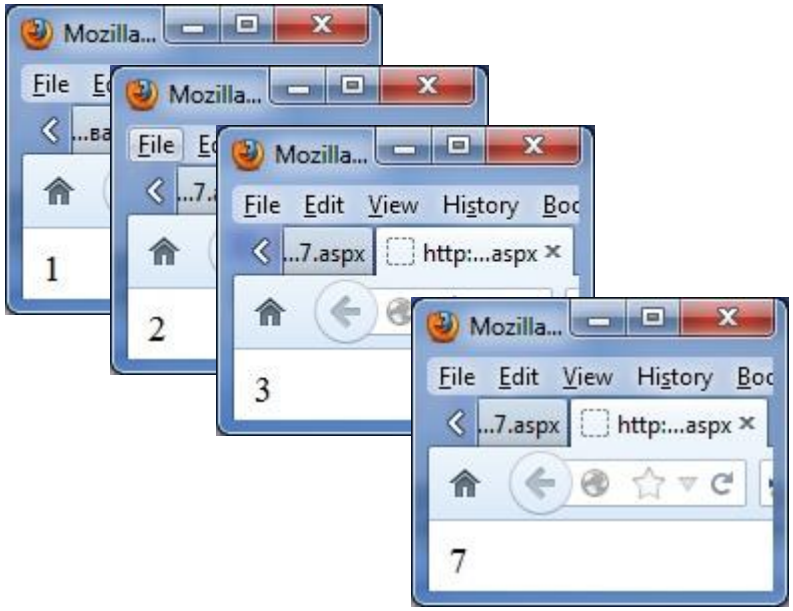
Об'єкт Session

- **Збереження об'єкта `user` у пам'яті сеансу**
  ```
  Session["user"] = user;
  ```
- **Відновлення збереженого об'єкта**
  ```
  user = (string) Session["user"];
  ```
- **Стан сеансу знищується у наступних випадках:**

- якщо користувач закриває браузер
- якщо користувач отримує доступ до тієї ж сторінки через інше вікно браузера при збереженні доступу до Веб-сторінки з вихідного вікна браузера
- після закінчення 20 хвилин з моменту останньої активності користувача
- при явному завершенні сеансу з програмного коду за допомогою виклику методу

Об'єкт Application

```
protected void Page_Load(object sender, EventArgs e)
{
    int count;
    if (Application["HitsNumber"] == null)
    {
        Application["HitsNumber"] = 1;
        count = (int)Application["HitsNumber"];
        Label1.Text = count.ToString();
        return;
    }
    count = (int)Application["HitsNumber"];
    count++;
    Application["HitsNumber"] = count;
    Label1.Text = count.ToString();
}
```



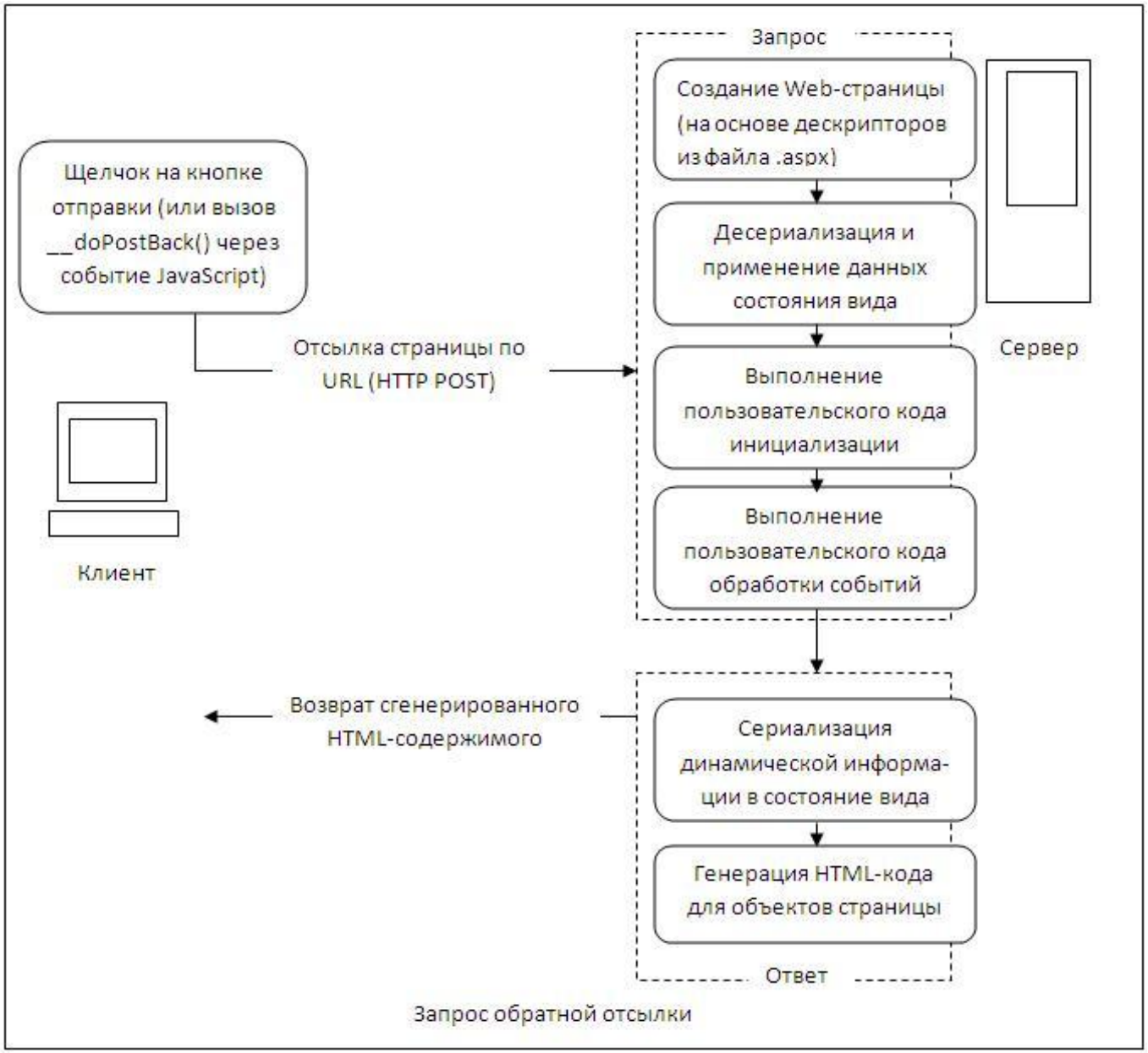
- **Стан застосування – дозволяє зберігати глобальні об'єкти, доступ до яких може отримати будь-який клієнт**
- **Зберігає інформацію на сервері**
- **Використовує синтаксис словаря**

```
protected void Page_Load(object sender, EventArgs e)
{
    int count;
    Application.Lock();
    if (Application["HitsNumber"] == null)
    {
        Application["HitsNumber"] = 1;
        count = (int)Application["HitsNumber"];
        Label1.Text = count.ToString();
        return;
    }
    count = (int)Application["HitsNumber"];
    count++;
    Application["HitsNumber"] = count;
    Application.UnLock();
    Label1.Text = count.ToString();
}
```

Порівняння опцій для керування станом

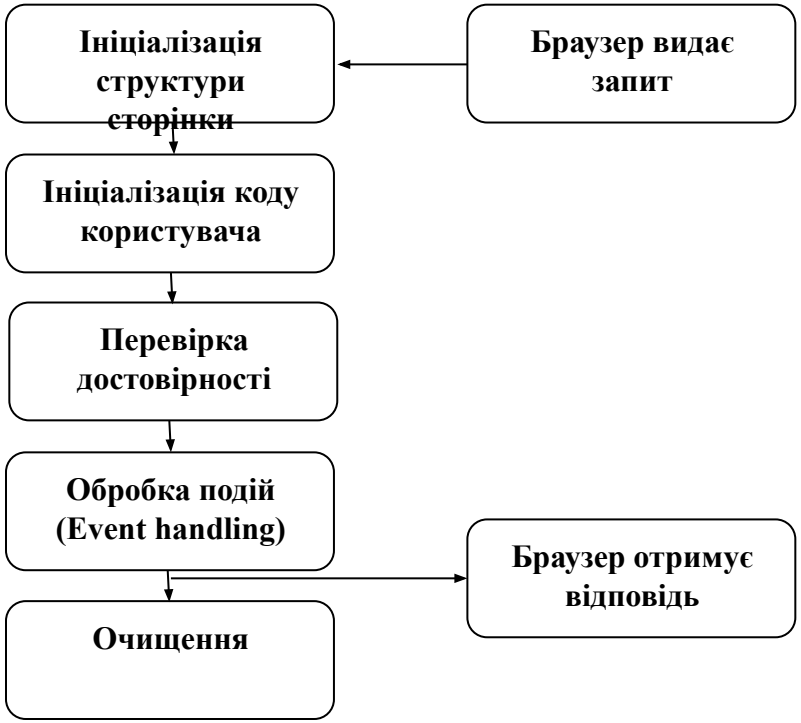
	Стан сеансу (Session)	Стан застосування(Application)
Допустимі типи даних	Усі .NET-типи даних, що піддаються сериализации	Усі .NET-типи даних
Місцезнаход-ження сховища та зберігання даних	У пам'яті сервера	У пам'яті сервера
Час життя	Закінчується по закінченні визначеного періоду часу (20 хвилин або може бути змінений програмно)	Збігається з терміном життя застосування (до перезавантаження сервера)
Контекст	Усе ASP.NET-застосування	Усе ASP.NET-застосування. На відміну від більшості інших типів методів, дані застосування є глобальними для усіх користувачів.
Безпека	Є безпечним/захищеним, тому що дані ніколи не передаються клієнту. Однак піддається атакам типу перехоплення сеансу	Є дуже безпечним, тому що дані ніколи не передаються клієнту
Складнощі, що впливають на продуктивність	Збереження великої кількості інформації може значно сповільнити роботу сервера, особливо у випадку, коли до нього підключається одночасно велика кількість користувачів, тому що для кожного користувача буде створюватися окрема	Збереження великої кількості інформації може сповільнити роботу сервера, тому що термін життя цих даних ніколи не буде закінчуватися, і вони ніколи не будуть видалятися

Запити сторінок ASP.NET



Основні етапи процесу обробки ASP.NET

- Ініціалізація середовища сторінки
- Ініціалізація коду користувача
- Перевірка достовірності (Validation)
- Обробка подій (Event handling)
- Автоматичне зв'язування з даними
- Очищення



Директиви

- Директиви - задають параметри, що використовуються компіляторами сторінок і користувацьких елементів керування
- Синтаксис оголошення директив

```
<%@ [Directive] [Атрибут=значення]%>
<%@ [Directive] [Атрибут=значення] [Атрибут=значення]%>
```

- Приклади

- імпорт просторів імен на сторінці aspx

```
<%@ Import Namespace= "System.XML " %>
```

- визначення властивостей сторінки, майстер-сторінки, користувацького елемента керування

```
<%@ Page Title="" Language="C#" AutoEventWireup="true"
  CodeFile="Default.aspx.cs" Inherits="_Default" %>
<%@ Master Language="C#" AutoEventWireup="true"
  CodeFile="MainMasterPage.master.cs"
  Inherits="MainMasterPage" %>
<%@ Control Language="C#" AutoEventWireup="true"
  CodeFile="WebUserControl.ascx.cs"
  Inherits="WebUserControl" %>
```

Атрибути директиви @Page

Атрибут	Опис
AutoEventWireup	Автоматична обробка подій сторінки
Buffer	Керує буферізацією сторінки. За промовчанням буферизується
ClassName	Дозволяє призначати ім'я класу, сгенерованого даною сторінкою
CodeFile	Назва файлу з відокремленим кодом для даної сторінки
Culture	Налаштовує набір регіональних параметрів, тобто мову, формат валюти, дати, чисел
Debug	Якщо true, на сторінку виводиться інформація відлагодження
Trace	Вивід інформації трасування
EnableViewState	Збереження стану сторінки. За промовчанням вона зберігається
EnableTheming	Дозволяє включити або виключити підтримку тем оформлення. За промовчанням включено
Inherits	Клас, від якого успадковуєся клас даної сторінки у технології відокремлення коду
IsAsync	Показує, чи обробляється сторінка асинхронно
Language	Мова, використовувана у вбудованому коді
WarningLevel	Найбільш допустимий рівень попереджень компілятора
CompilerOptions	Опції компілятора

Література

- *Мэтью Мак-Дональд, Марио Шпушта.* Microsoft ASP.NET 3.5 с примерами на C# 2008 и Silverlight 2 для профессионалов.: Пер. с англ. – М.: ООО "И.Д. Вильямс", 2009. – 1408 с.
- *Рихтер Дж.* CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C#. 3-е изд. – СПб.: Питер, 2012. – 928 с.
- *Троелсен Э.* Язык программирования C# 5.0 и платформа .NET 4.5. 6-е изд., – М.: ООО "И.Д. Вильямс", 2013. - 1312 с.
- Корисні ресурси
  - http://msdn.microsoft.com/ru-ru/library/
  - http://habrahabr.ru/post/165597/

Дякую за увагу