



Урок #1

План заняття:

1. Що таке ASP.NET та його можливості. Порівняльний аналіз існуючих серверних веб-технологій
2. Встановлення IIS 7.0
3. Пишемо першу сторінку ASP.NET
4. Структура сторінки ASP.NET. Директиви сторінки
5. Створюємо сторінку ASP.NET засобами Visual Studio 2010
6. Поняття Web-форми
 - 6.1. Форма HTML
 - 6.2. Життєвий цикл Web-форми. Сторінкова подія Load
 - 6.3. Клас System.Web.UI.Page
7. Поширені простори імен ASP.NET
8. Елементи управління
 - 8.1. Клас System.Web.UI.Control. Серверні елементи управління, їх класифікація
 - 8.2. Елементи управління Web. Простір імен System.Web.UI.WebControls
 - 9.2.1. Елементи управління Label та TextBox. Кнопки (Button)
 - 9.2.2. Елементи управління HyperLink та Image
 - 9.2.3. Елементи управління CheckBox та RadioButton
 - 9.2.4. Списки: ListBox, CheckBoxLayout, RadioButtonList, DropDownList
 - 9.2.5. Нумерований список BulletedList
 - 9.2.6. Таблиці
 - 8.3. Серверні елементи управління HTML. Простір імен System.Web.UI.HtmlControls
 - 8.4. Короткий огляд бібліотек сторонніх виробників
9. Web-форма як контейнер елементів управління. Динамічне створення елементів управління
10. Приклади використання елементів управління
11. Домашнє завдання

1. Що таке ASP.NET та його можливості. Порівняльний аналіз існуючих серверних веб-технологій

Сьогодні ми розпочнемо вивчення нової технології – ASP.NET. Але перед тим як розпочати знайомство з її основами, розглянемо коротенько що це за технологія і для чого її необхідно використовувати.

Сучасний світ вимагає від веб-додатків динаміки, яка повинна проявлятися у всьому: від динамічного дизайну (випадаючі меню, анімація, приховані блоки тощо) до динамічного інтерфейсу (взаємодія з користувачем, відображення вмісту бази даних тощо). І якщо для реалізації першого достатньо клієнтських скриптів, наприклад, JavaScript, то для здійснення, наприклад, взаємозв'язку з базою даних цього мало, оскільки база даних повинна розміщуватись на сервері.

Для вирішення таких задач використовують серверні скриптові мови та серверні технології створення веб-додатків. Одними з найпоширеніших на сьогоднішній день являються PHP, Perl, ASP.NET, JSP, Ruby on Rails, Python тощо. Всі вони мають свої переваги та недоліки. У всьому іншому вони дещо відрізняються. Наприклад, серверні скриптові мови кросплатформенні, але мають менше можливостей, чим повнофункціональні мови програмування. Крім того, скриптові мови не підтримують строгую типізацію, що з однієї сторони спрощує роботу, але з іншої призводить до помилок. Код PHP, Perl та інших серверних скриптових мов, вбудований в HTML у вигляді спеціальних тегів і розкиданий по самій веб-сторінці. Це доволі незручно і створює плутанину.

ASP.NET та JSP дозволяють відмежувати програмний код від коду HTML та розмістити їх в різних файлах (code behind). Код ASP.NET і JSP, на відміну від серверних скриптових мов, компілюється, а не інтерпретується. Отже, швидше виконується. Але багато в чому ці дві технології відрізняються.

Переваги ASP.NET над JSP:

- менш ресурсоємкий, ніж JSP (при інших рівних параметрах);
- ASP.NET дозволяє писати програмний код на різних мовах, наприклад, C#, VB.NET тощо;
- містить багато готових елементів управління, використовуючи які можна швидко створювати інтерактивні веб-сайти;
- інтеграція з MS SQL Server, який потужніше MySQL і дешевше, наприклад, за Oracle;
- має зручне середовище розробки.

**Недоліки ASP.NET:**

- більшість елементів управління генерують надмірно надлишковий код;
- в порівнянні навіть з PHP помітні затримки. Щодо швидкості, то JSP тут має суттєві переваги; це більш промислова технологія, яка розрахована на велику кількість звернень в хвилину.

Отже, **ASP.NET (Active Server Page .NET - активні серверні сторінки)** – це технологія, яка використовується для створення клієнт-серверних веб-додатків та веб-сервісів. ASP.NET виникла в результаті об'єднання застарівшої серверної скриптової мови ASP і .NET Framework.

ASP була розроблена корпорацією Microsoft у 1996 р. та належала до тих скриптових мов, які виконуються на стороні сервера. Зрозуміло, що вона мала обмежений набір можливостей. Тому в 2000 році на конференції розробників Microsoft презентувала нову технологію .NET Framework, до складу якої входила ASP+. З виходом .NET Framework 1.0 вона стала називатись ASP.NET. Але ASP.NET – це не продовження ASP. Це концептуально нова технологія Microsoft, створена в рамках ідеології .NET. В ASP.NET створено все для того, щоб зробити весь цикл розробки веб-додатку більш швидким, а підтримку – більш простою. ASP.NET основана на об'єктно-орієнтованій ідеології, але зберегла модель розробки ASP: ви створюєте програму і поміщаєте її в директорію, виділену сервером, і вона буде працювати.

ASP.NET використовує компілюємі мови, які являються типізованими. Компіляція відбувається на сервері в момент першого звернення користувача до сторінки. Якщо програміст змінив текст сторінки, програма перекомпілюється автоматично. При написанні самого коду можна використовувати набір компонентів, поставляємих з .NET. Зокрема для доступу до даних використовується технологія ADO.NET. Крім того, посиленна модель безпеки дозволяє забезпечити захист клієнта і сервера від несанкціонованого доступу.

В 2003 році зв'яється на світ .NET 1.1 разом з ASP.NET 1.1, а згодом (в 2005 р.) - версія ASP.NET 2.0. Серед нових можливостей версії 2.0 можна виділити: використання шаблонів дизайну сторінок (Master Page) та тем, спрощена локалізація web-додатків, більше 50 нових серверних елементів управління, використання профілів тощо. Ціллю даної технології було підвищення швидкості розробки сайтів, масштабування, легкість підтримки і адміністрування сайтів. З'явилась панель оснастки MMC (консоль управління Microsoft), яка надавала графічний інтерфейс для управління опціями ASP.NET. Змінити опції проекту тепер можна і через веб-інтерфейс. ASP.NET 2.0 підтримує роботу на 64-бітних процесорах.

Наприкінці 2006 р. побачила світ нова проміжна версія платформи .NET Framework 3.0, але для ASP.NET вона нічого нового не принесла. Але вже через рік виходить версія ASP.NET 3.5, яка розширена двома технологіями: LINQ (мова інтерпретованих запитів) та Ajax.

Останньою версією цієї технології являється ASP.NET 4.0 (2010 р.), яка містить багато покращених можливостей. Це розширений набір засобів для кешування, нові елементи управління та вдосконалення роботи деяких старих контролів, шаблони коду, підтримка CSS 2.1 тощо.

Разом з тим корпорація Microsoft випустила технологію Silverlight та кілька розширень для ASP.NET:

- ASP.NET MVC;
- ASP.NET Dynamic Data.

І хоча у вересні поточного 2011 р. корпорація Microsoft вже презентувала попередню версію ASP.NET 4.5, але повноцінно світ вона ще не побачила. Тому наш курс ми присвяtimo вивченню ASP.NET 4.0.

2. Встановлення IIS 7.0

Перед тим як перейти до практичної роботи необхідно сконфігурувати необхідне програмне забезпечення. Для цього, вам необхідний інформаційний веб-сервер та .NET SDK для ASP.NET. В якості веб-сервера для ASP.NET використовують, як правило, IIS.

IIS (Internet Information Server, Інформаційні Служби Інтернет) – це сервер компонентів для служб Інтернету від фірми Microsoft, який входить до складу Windows. Найпоширенішим компонентом IIS являється веб-сервер, який дозволяє розгорнути веб-сайт для доступу користувачів через Інтернет. Для цих цілей веб-сервер IIS підтримує протоколи HTTP, HTTPS, FTP, POP3, SMTP, NNTP, SSL тощо.

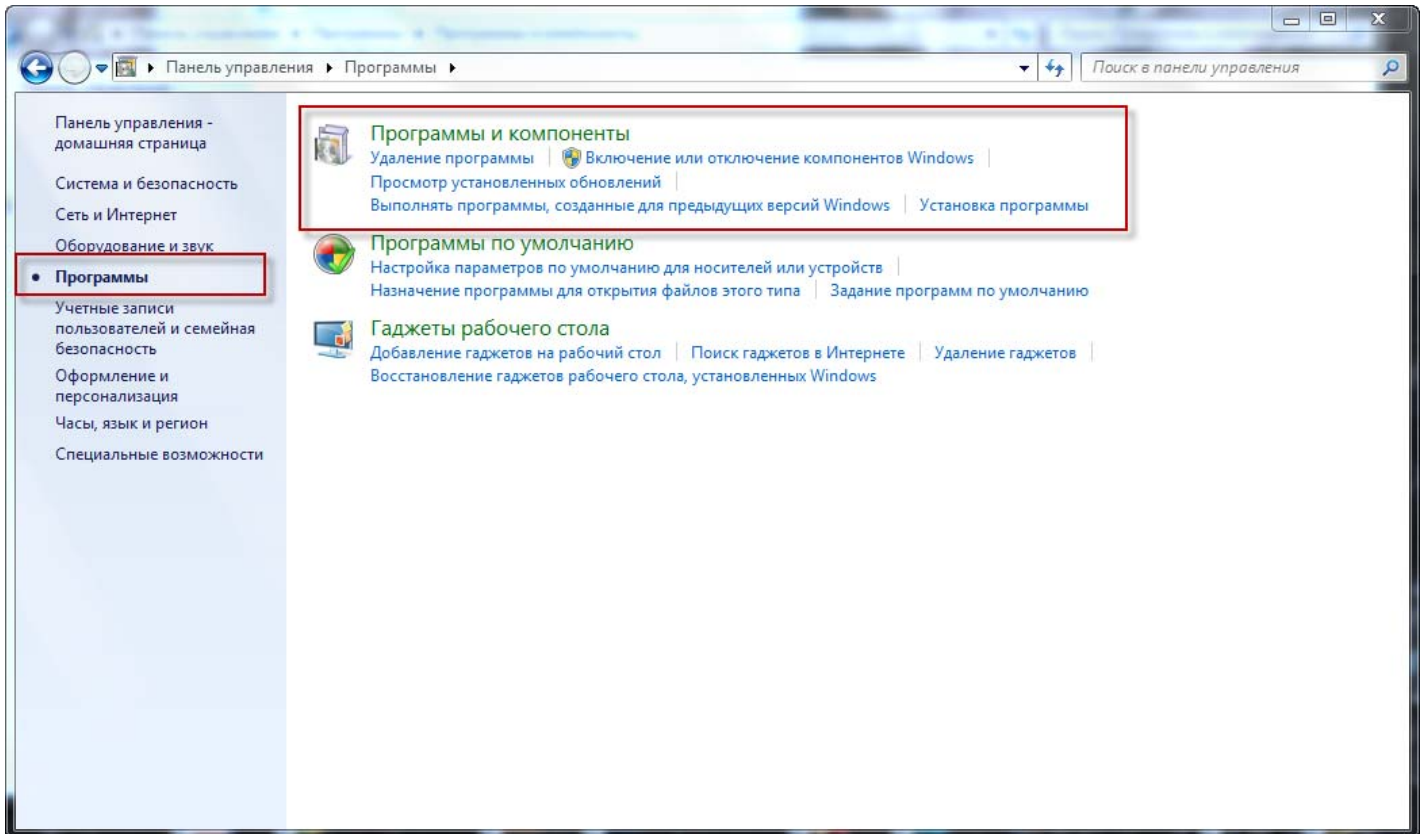
Для спрощення процесу інсталяції IIS або компонентів, які входять до її складу, використовується інструмент WebPI (Web Platform Installer). Цей інструмент дозволяє також швидко розгорнути робоче місце веб-розробника та встановити необхідний інструментарій для веб-розробки. Він входить до складу Windows, тому ви можете легко ним скористатись.

Поточною версією IIS являється IIS 7, яка по суті складається з двох версій IIS: 7.0 та 7.5. Відзначимо, що у версії IIS 7.5 було вдосконалено засоби управління сервером, служби захисту, управління обліковими записами, додані нові засоби адміністрування, розширені можливості FTP тощо.

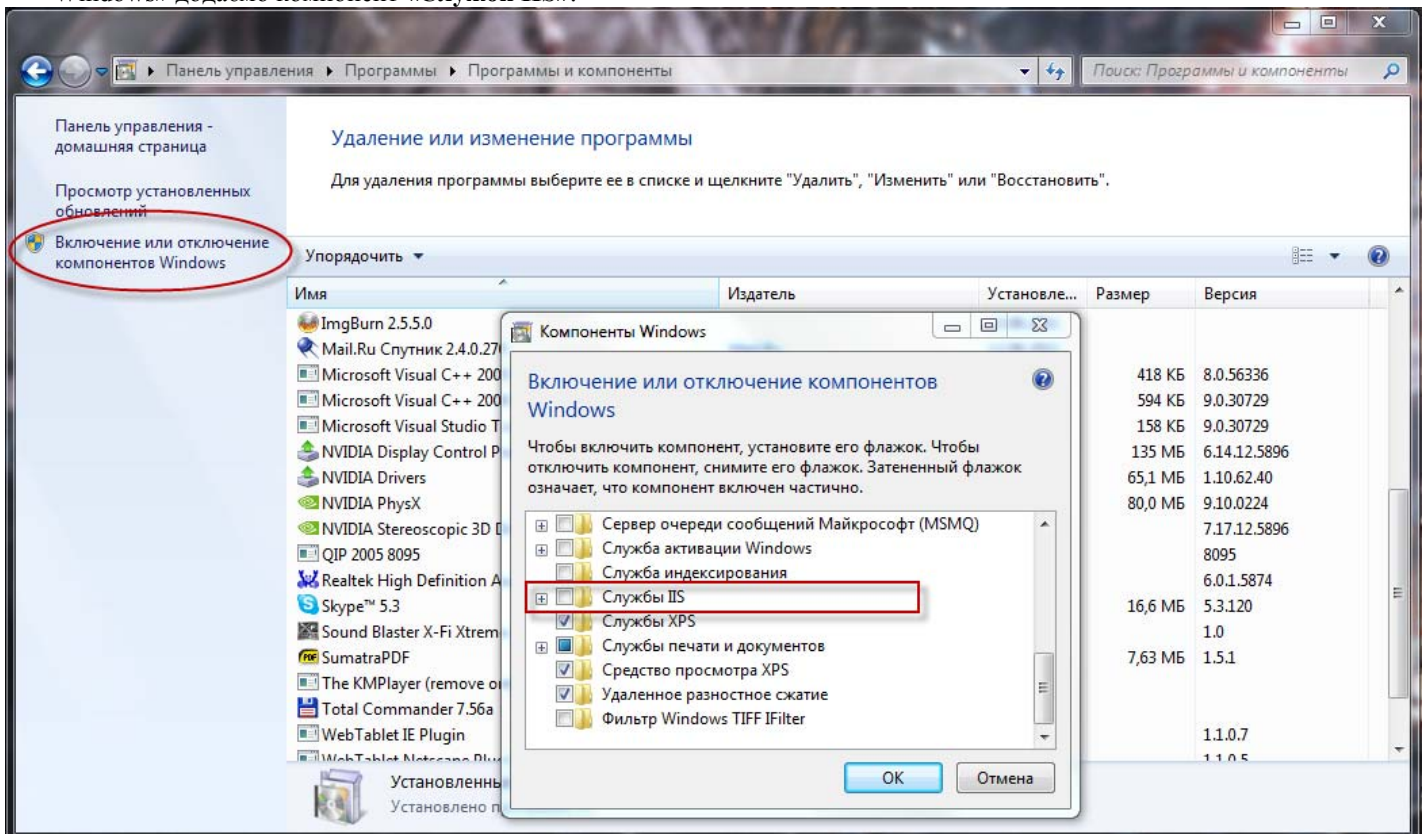
До складу Windows Server 2008 та Windows Vista входить версія IIS 7.0, а до складу Windows Server 2008 R2 та Windows 7 - IIS 7.5. Причому модернізувати версію IIS 7.0 до 7.5 неможливо.

Приведемо для початку приклад інсталяції веб-сервера IIS 7 на платформі Windows 7. Для цього необхідно здійснити наступні дії:

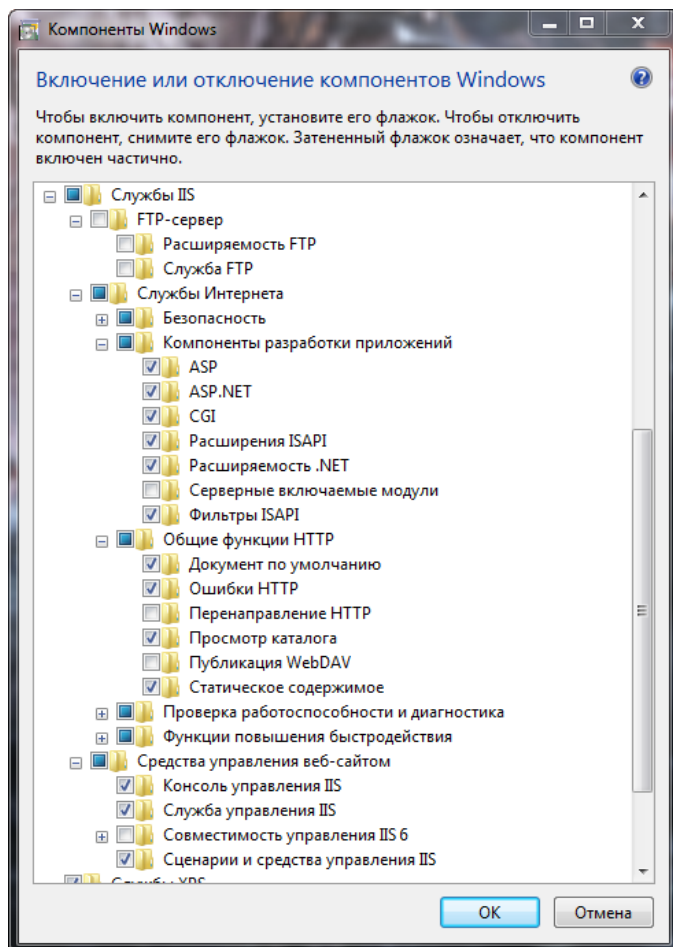
1. Оберіть в меню **Пуск** пункти **Параметри->Панель управління->Програми->Програми та компоненти**.



2. Далі обираємо пункт «Включення та відключення компонентів Windows» і в діалоговому вікні «Компоненти Windows» додаємо компонент «Служби IIS».



3. Щоб встановити окремі частини IIS, наприклад, службу FTP або SMTP, потрібно розгорнути вузол «Служби IIS» та обрати необхідні компоненти. При цьому, якщо необхідно використовувати ASP, ASP.NET, PHP тощо, ви можете легко встановити необхідні модулі, обравши відповідний пункт меню.



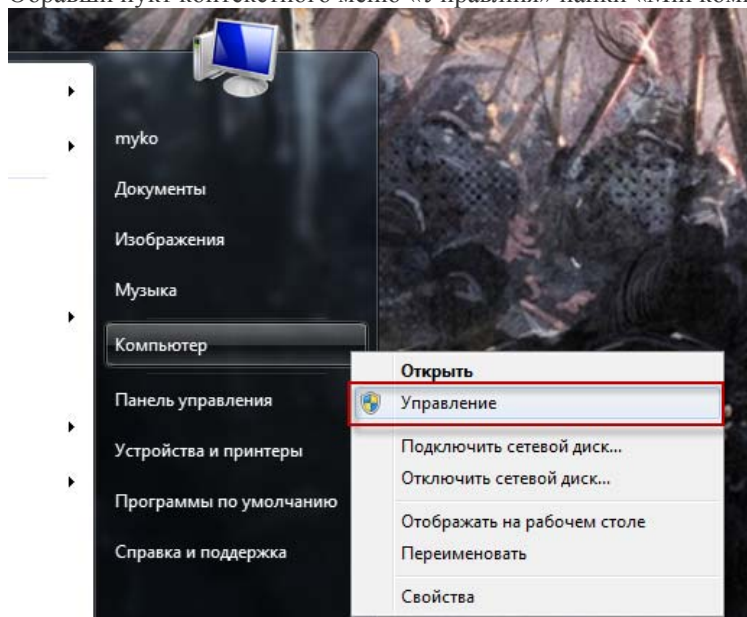
4. Після натиснення кнопки ОК, Windows розпочне збирати необхідну для установки інформацію, а потім розпочнеться процес інсталяції.
5. Після встановлення веб-сервера, необхідно перевірити його роботу. Для цього слід відкрити браузер і ввести рядок запити <http://localhost> або http://ім'я_комп'ютера, після чого відкриється ваш веб-вузол з сторінкою по замовчуванню, створеною IIS 7.0.



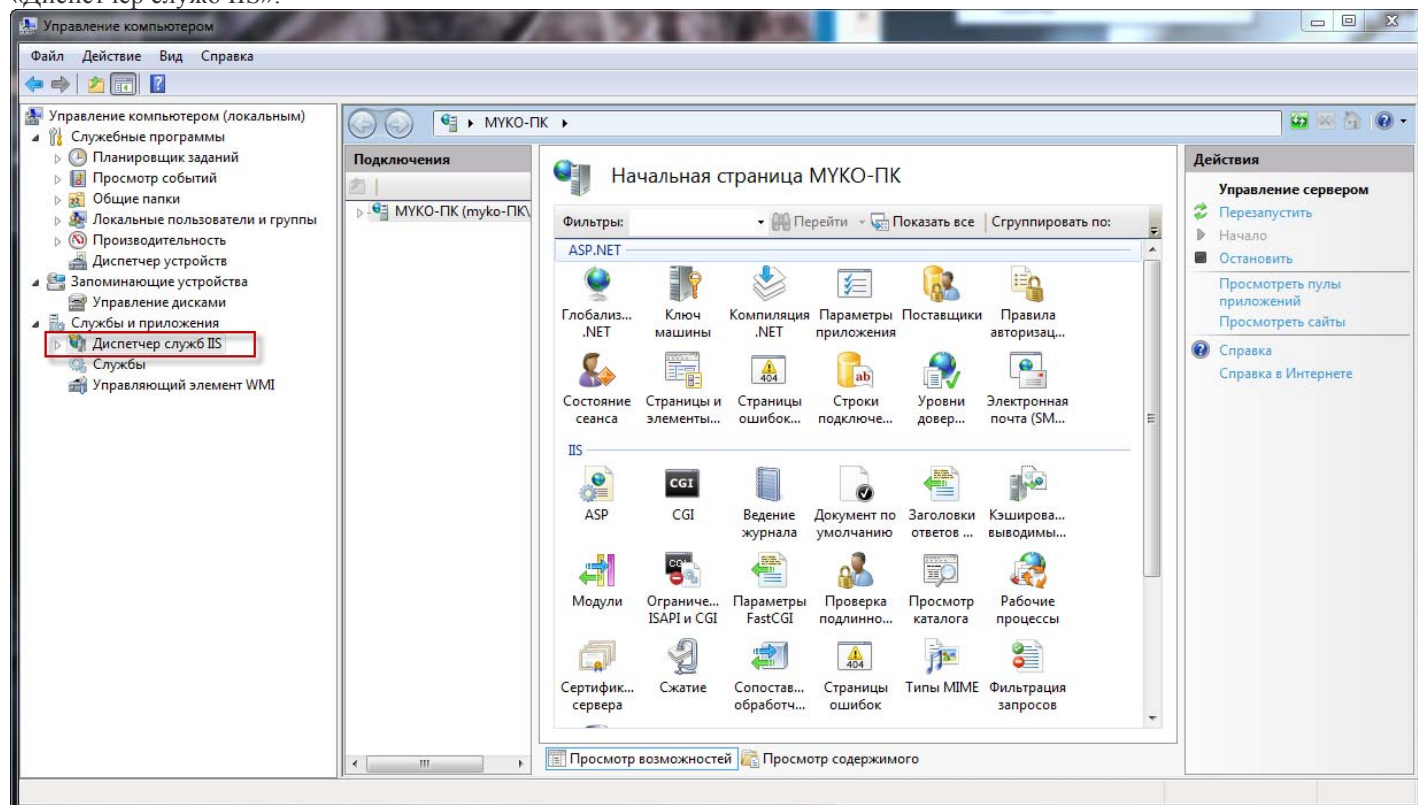


Для управління і конфігурування IIS 7.0 використовується Диспетчер служб IIS (Internet Information Services Manager), який можна запустити **трьома способами**:

1. Ввівши в командному рядку (Пуск->Виконати) команду **inetmgr**.
2. Обравши пункт контекстного меню «Управління» папки «Мій комп'ютер» або в меню «Пуск->Комп'ютер».

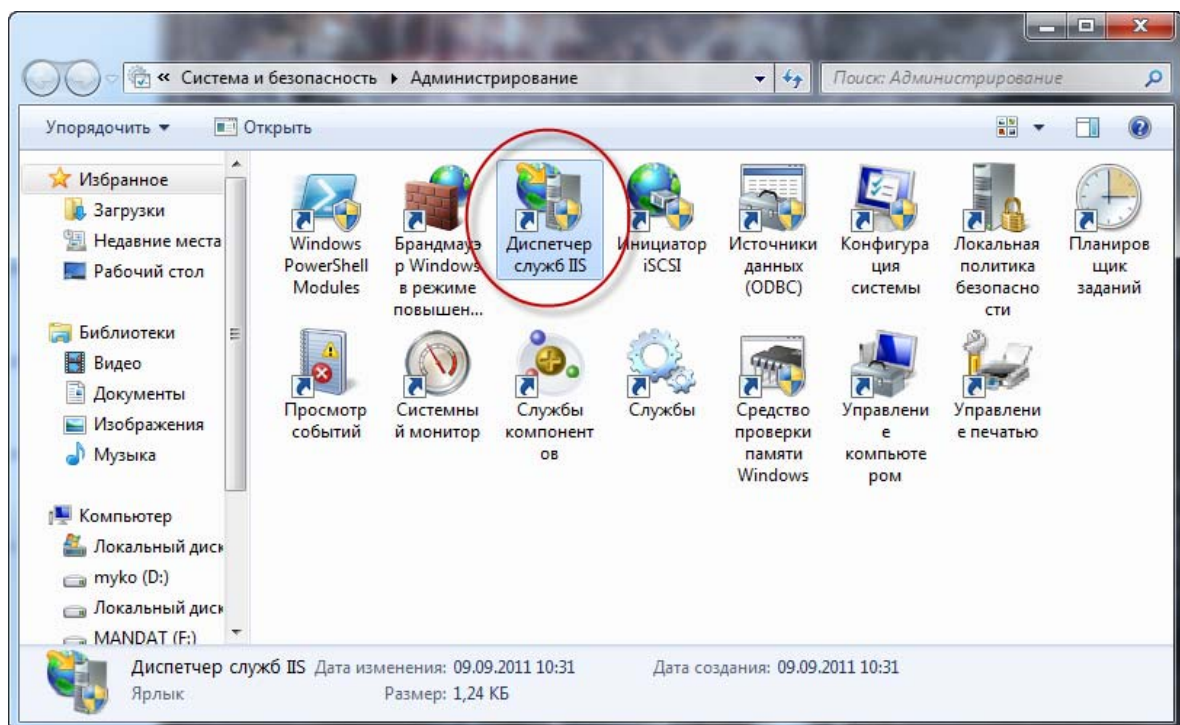
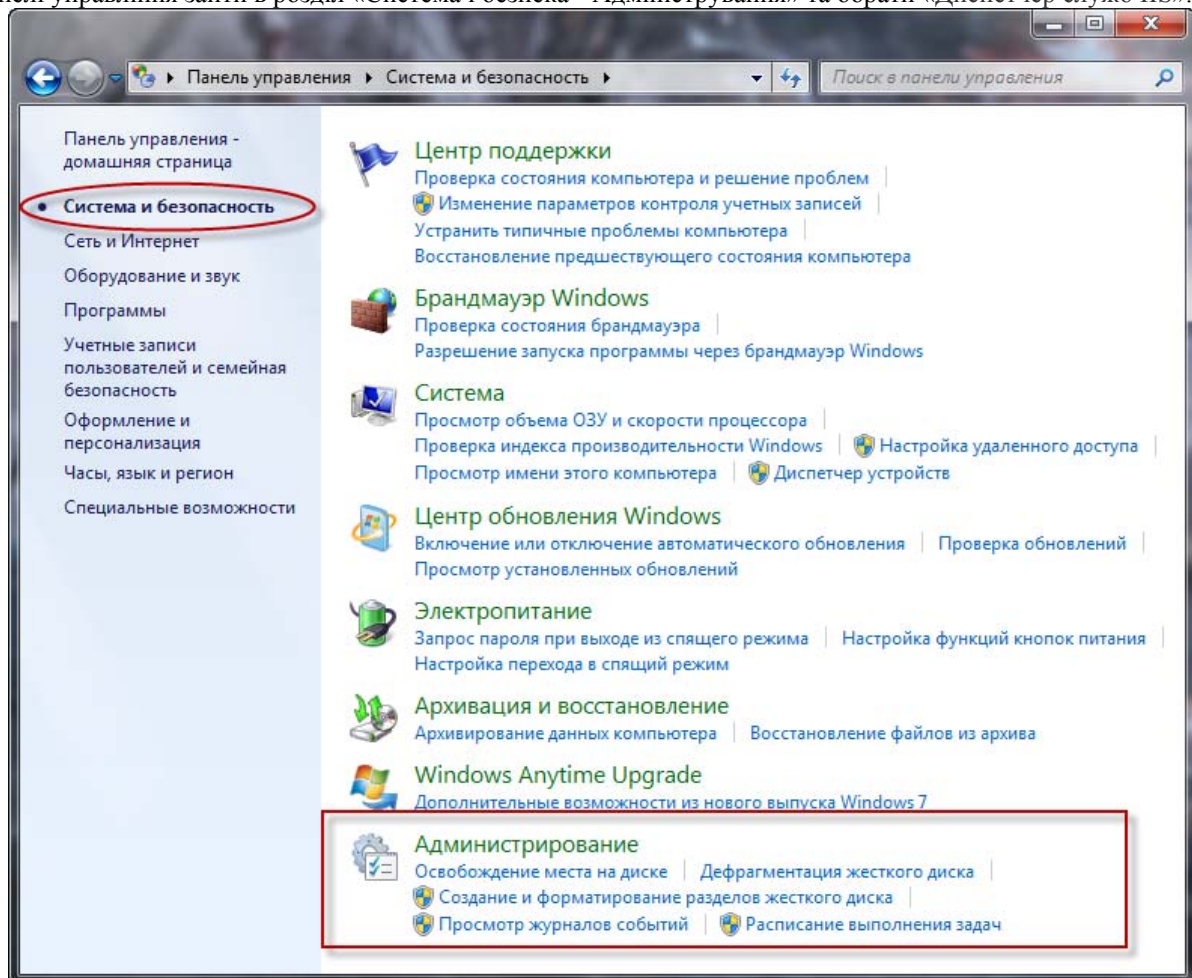


Після цього відкриється панель управління комп'ютером, де в розділі «Служби і додатки» необхідно обрати «Диспетчер служб IIS».



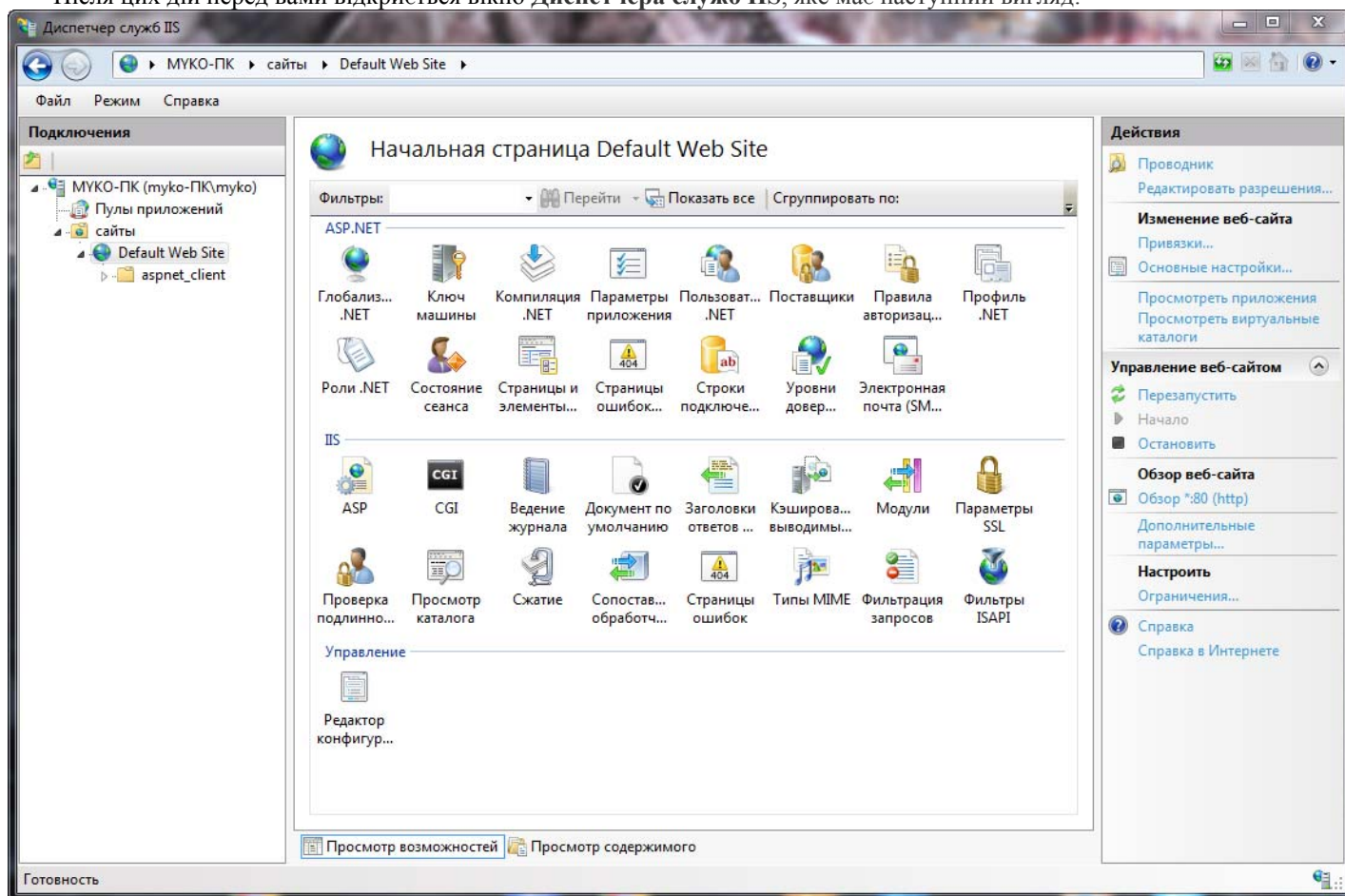


3. В Панели управления зайти в раздел «Система и безопасность» та обрати «Диспетчер служб IIS».





Після цих дій перед вами відкриється вікно Диспетчера служб IIS, яке має наступний вигляд:



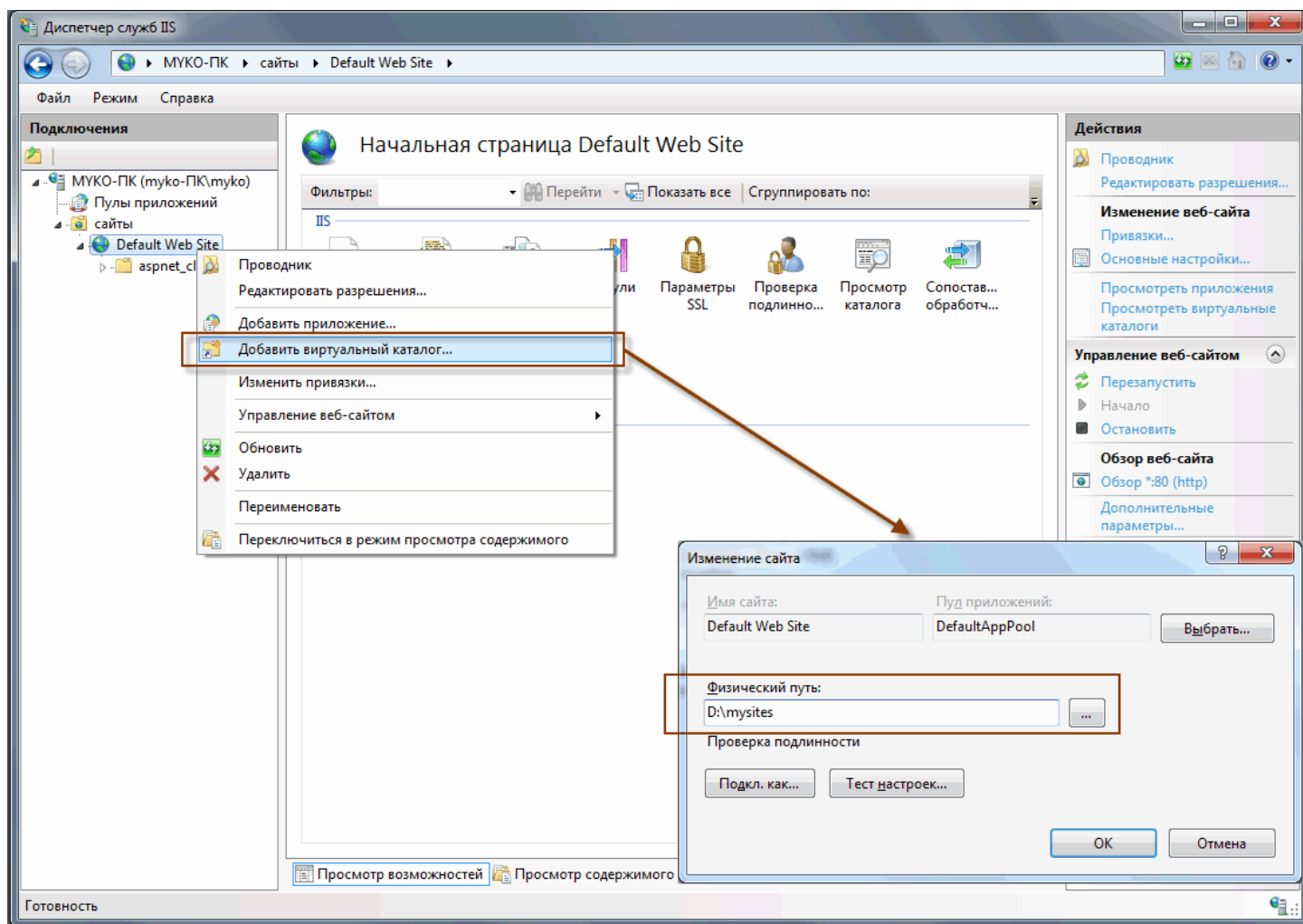
В лівій частині Диспетчера служб IIS відображається **область «Підключення»**, яка дозволяє підключатись до веб-серверів, сайтів та інших додатків.

В середній частині розміщується **робоча область** Диспетчера служб IIS з необхідним інструментарієм для конфігурування параметрів обраного сервера. Робоча область містить два представлення: Перегляд можливостей та Перегляд вмісту. **Перегляд можливостей** дозволяє переглянути фактичний вміст обраного в дереві підключень об'єкта. Наприклад, при виборі сайту в робочій області буде відображатись його вміст, тобто список фізичних і віртуальних каталогів та файлів. В режимі **перегляду вмісту** можна переглянути та налаштувати властивості окремих об'єктів, наприклад, окремо обраного сайту.

Права частина вікна містить **список дій** по управлінню сервером IIS, ASP.NET або самим Диспетчером служб IIS. Наприклад, тут можна здійснювати запуск, зупинку або перезапуск веб-сервера IIS або певного сайту.

Щоб здійснити запуск ваших ASP.NET додатків, необхідно створити віртуальний каталог на одному з сайтів і помістити в нього свої файли. **Віртуальний каталог** – це посилання на реально існуючу папку на жорсткому диску. При інсталяції IIS створює папку **Default Web Site** (Веб-сайт по замовчуванню), яка розміщується в списку сайтів та з якої буде здійснюватись запуск всіх ваших ASP.NET додатків. По замовчуванню такою папкою буде **%SystemDrive%\inetpub\wwwroot**. Тобто, коли ви вводите в рядку пошуку URL адресу, ваш веб-вузол насправді бачить сторінку, яка знаходиться саме в цій папці.

Ви також можете змінити директорію по замовчуванню для ваших сайтів, тобто створити свій власний віртуальний каталог. Для цього в IIS 7 необхідно викликати контекстне меню каталога **Default Web Site** та обрати в ньому пункт **«Додати віртуальний каталог»**. В новому діалоговому вікні **«Зміна сайту»**, в розділі **«Фізичний шлях»** потрібно вказати новий кореневий каталог.



Тепер для запуску сторінок ASP.NET потрібно буде прописати один з наступних рядків:

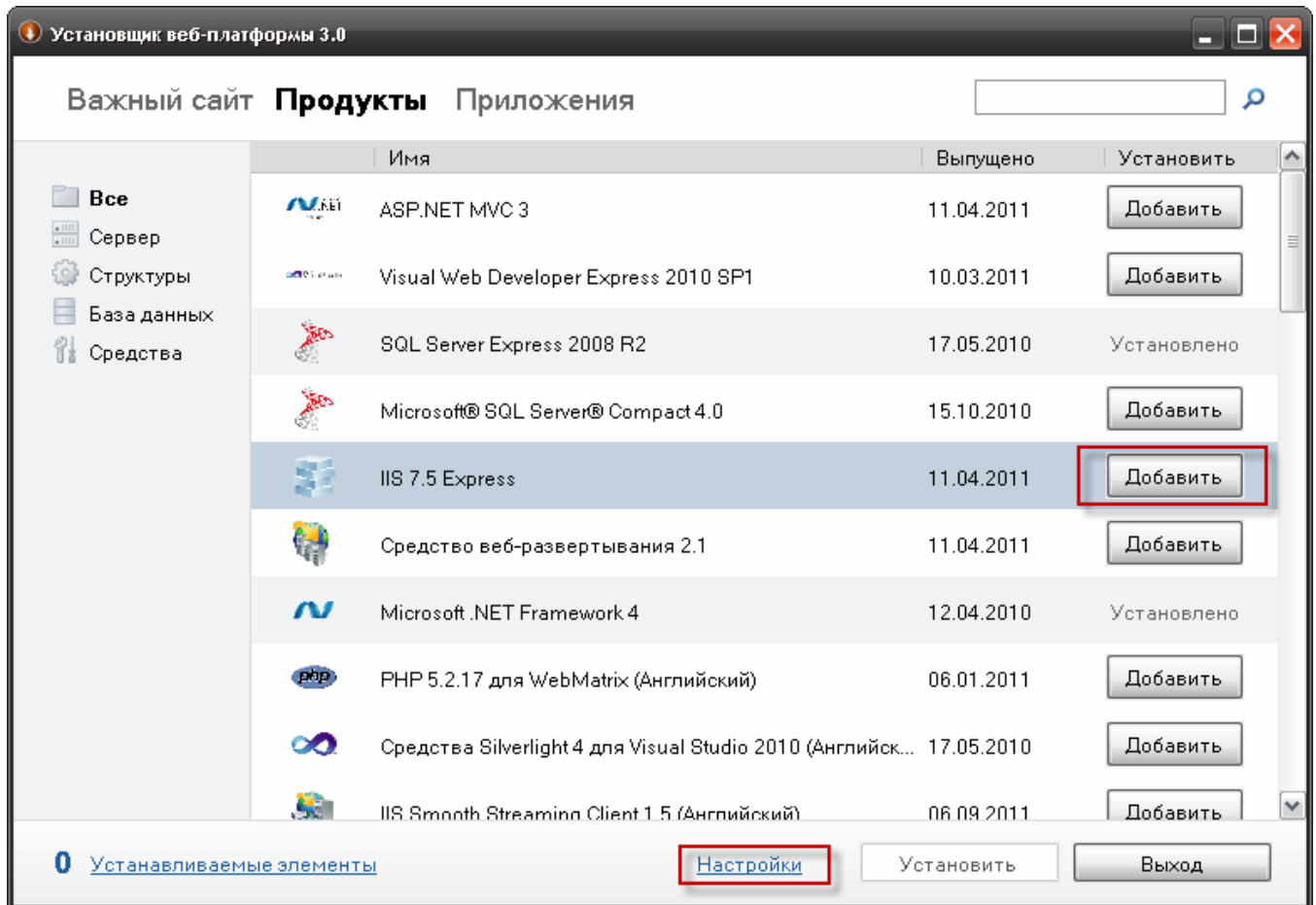
```
http://localhost/віртуальна_папка/ім'я_файла.aspx
http://ІРадреса_сервера/віртуальна_папка/ім'я_файла.aspx
http://ім'я_сервера/віртуальна_папка/ім'я_файла.aspx
```

Якщо ж ви користуєтесь версією Windows, до складу якої не входить IIS 7, тоді слід скачати WebPI з сайту розробника по адресі <http://www.microsoft.com/web/downloads/platform.aspx>. Тут слід відмітити наступне: на момент написання уроку поточною версією WebPI являється 3.0, до складу якого входить веб-сервер IIS 7.5 Express, а також ще ряд інших компонентів, зокрема і SQL Server Express 2008, PHP, Joomla, WordPress тощо. Версія IIS 7 входить до складу WebPI 2.0.

IIS 7.5 Express – це оптимізована для розробників версія IIS 7.5, яка містить засоби для створення та тестування веб-додатків в Windows. Для такого тестування він об'єднується з зручним веб-сервером, наприклад, сервером ASP.NET Development Server або Visual Studio 2010.

Для повноцінного функціонування IIS 7.5 Express, його слід також інсталиувати разом з MS WebMatrix, який надає інтегрований набір засобів для спрощення розробки веб-додатків в Windows.

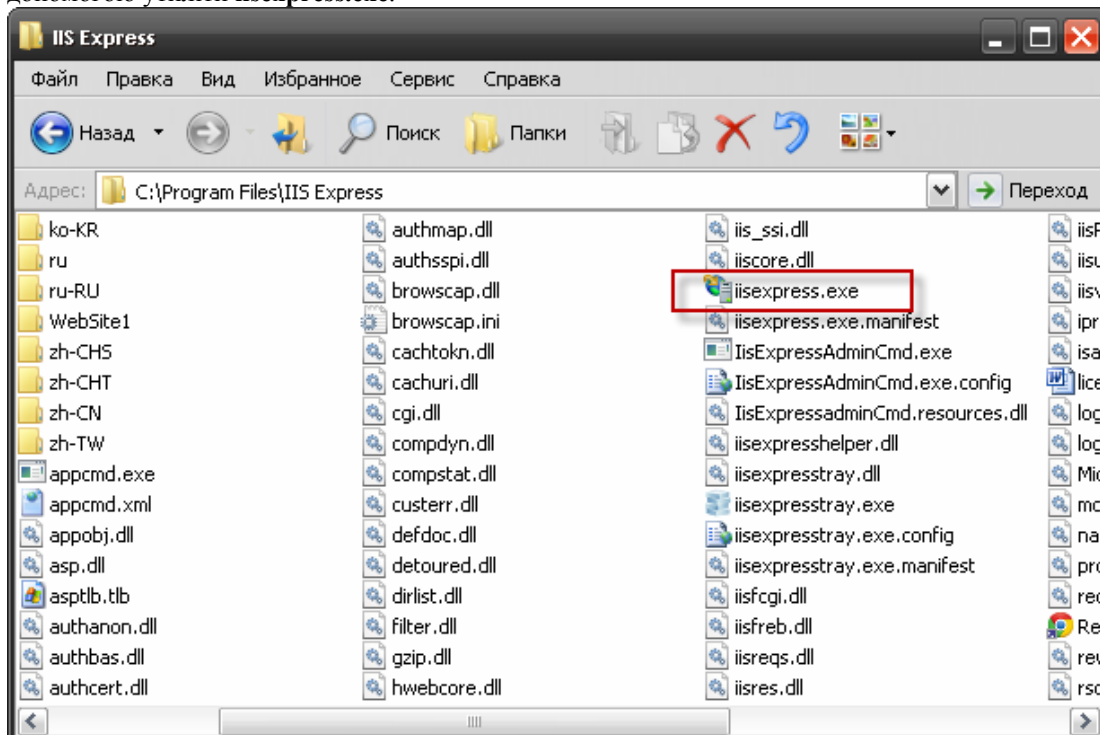
Отже, завантаживши WebPI 3.0, вам необхідно її запустити. Після цього в розділі «**Продукти**» оберіть та додайте компонент IIS 7.5 Express. Ви можете додатково налаштувати встановлюваний сервер за допомогою кнопки «Налаштування».



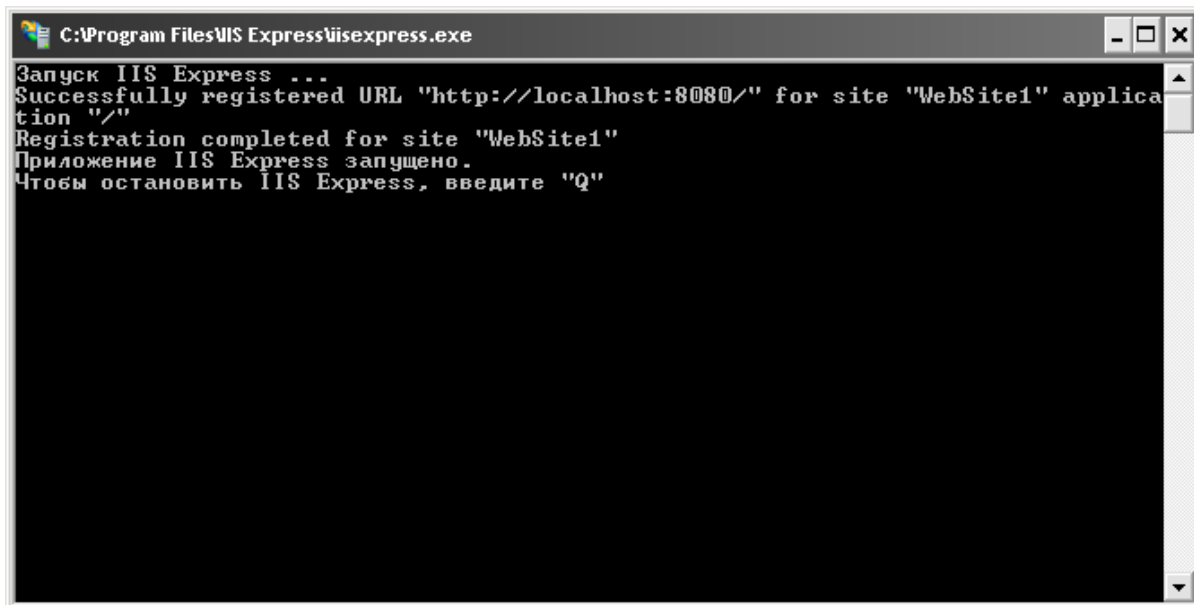
Запускаємо процес інсталяції.

Тепер, щоб перевірити коректність роботи встановлених компонентів, необхідно **зробити наступні кроки**:

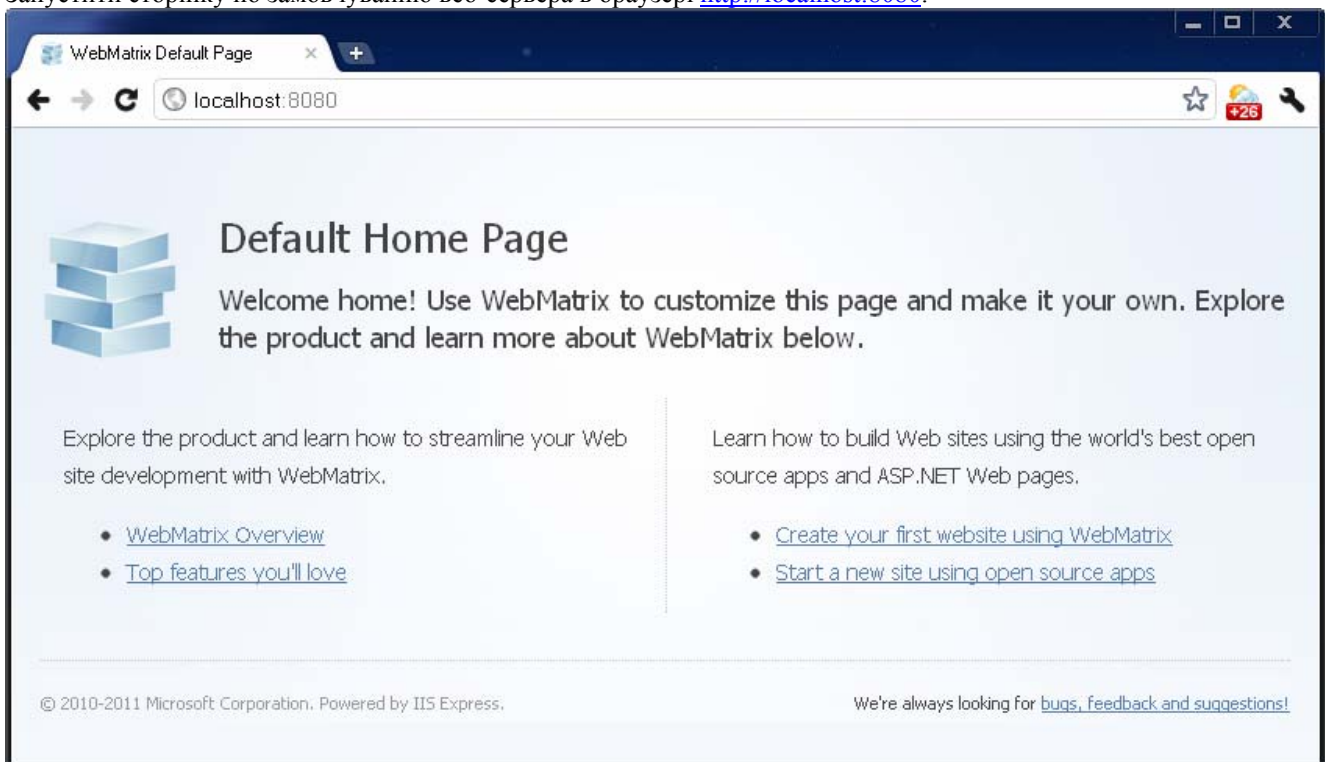
1. Перейти в батьківську директорію сервера. Як правило, це C:\Program Files\IIS Express та запустити сервер за допомогою утиліти **iisexpress.exe**.



У вікні **iisexpress** можна відслідковувати всі подальші дії з сервером, а також у випадку необхідності зупинити його, ввівши команду «Q» (Quit).



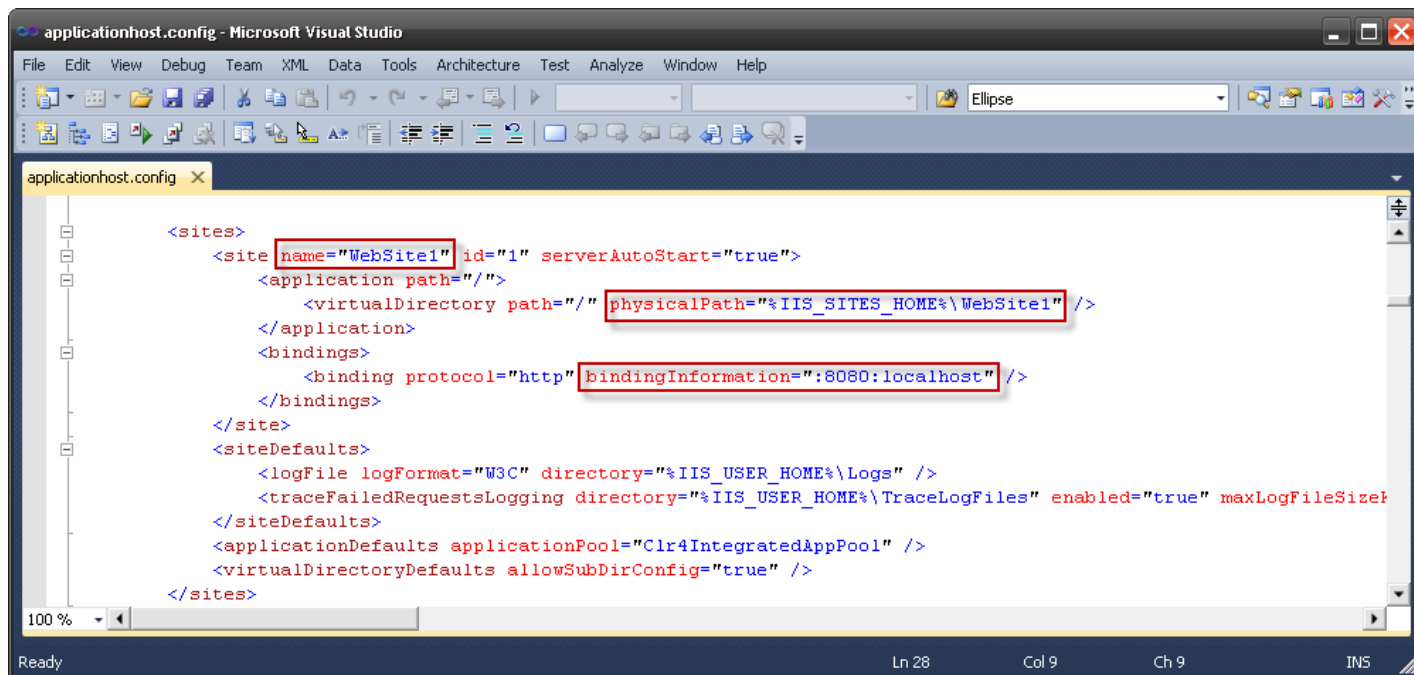
2. Запустити сторінку по замовчуванню веб-сервера в браузері <http://localhost:8080>.



Директорією по замовчуванню для IIS 7.5 Express буде

C:\Documents and Settings\[ім'я_користувача]\Мои документы\My Web Sites\WebSite1\

Але це можна легко змінити за допомогою конфігураційного файлу **applicationhost.config**, який розміщується в папці **C:\Documents and Settings\[ім'я_користувача]\Мои документы\IISExpress\config**. В цьому файлі, в розділі **sites** необхідно змінити значення в параметрах: **name** елемента **<site>**, **physicalPath** елемента **<virtualDirectory>** та **bindingInformation** елемента **<binding>** на зручні для вас.



Наприклад, нам необхідно встановити стартовою директорією для сайтів папку D:\mysites. В такому разі розділ sites набуде наступного вигляду:

```
<sites>
  <site name="mysites" id="1" serverAutoStart="true">
    <application path="/">
      <virtualDirectory path="/" physicalPath="D:\mysites" />
    </application>
    <bindings>
      <binding protocol="http" bindingInformation=":8080:localhost" />
    </bindings>
  </site>
  <siteDefaults>
    <logFile logFormat="W3C" directory="%IIS_USER_HOME%\Logs" />
    <traceFailedRequestsLogging directory="%IIS_USER_HOME%\TraceLogFiles" enabled="true"
maxLogFileSizeKB="1024" />
  </siteDefaults>
  <applicationDefaults applicationPool="Clr4IntegratedAppPool" />
  <virtualDirectoryDefaults allowSubDirConfig="true" />
</sites>
```

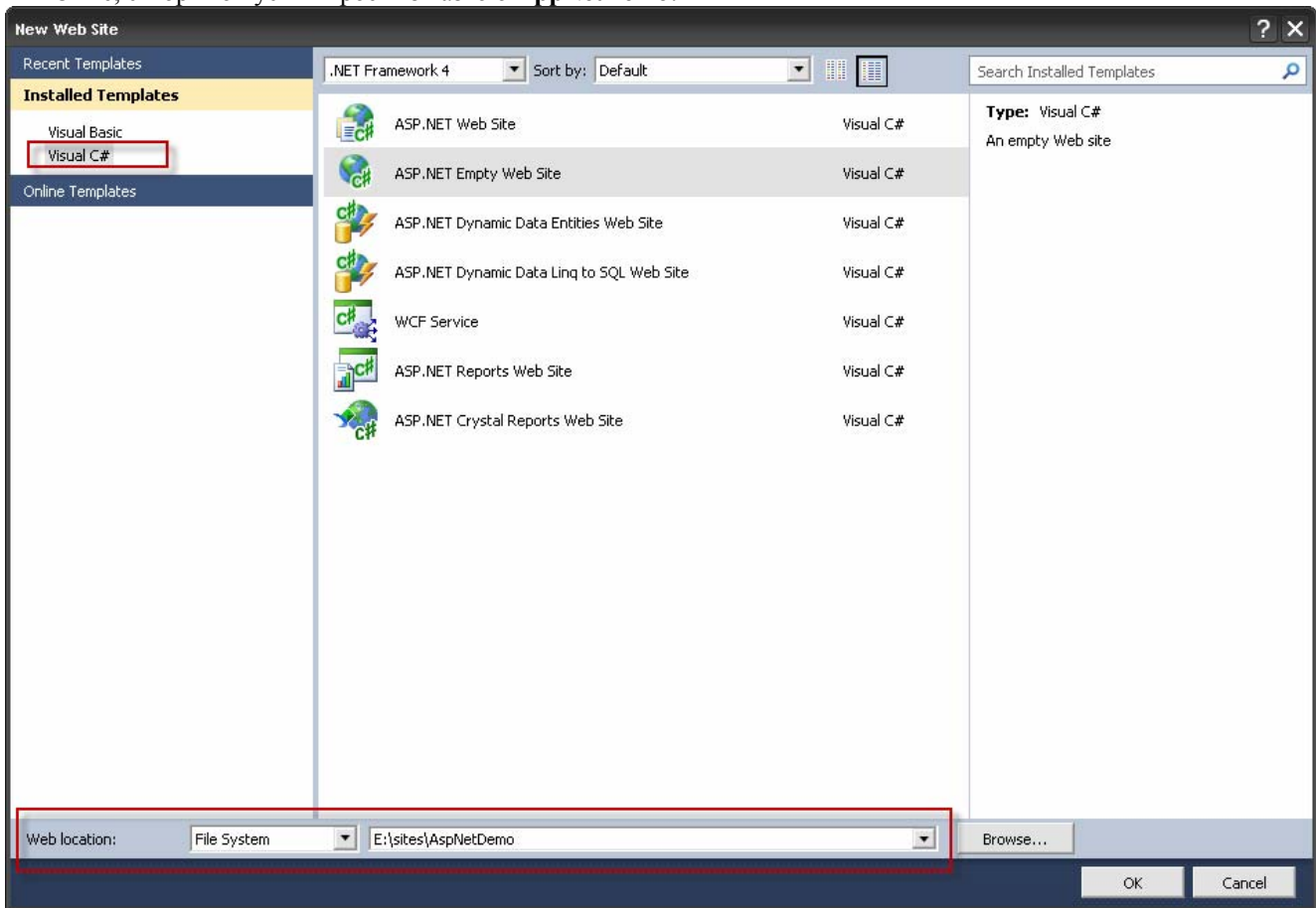
Якщо ваш веб-додаток використовує додаткові віртуальні каталоги, тоді їх слід додати в розділ <application>. Для кожного віртуального каталога свій дочірній елемент <virtualDirectory>.

3. Пишемо першу сторінку ASP.NET

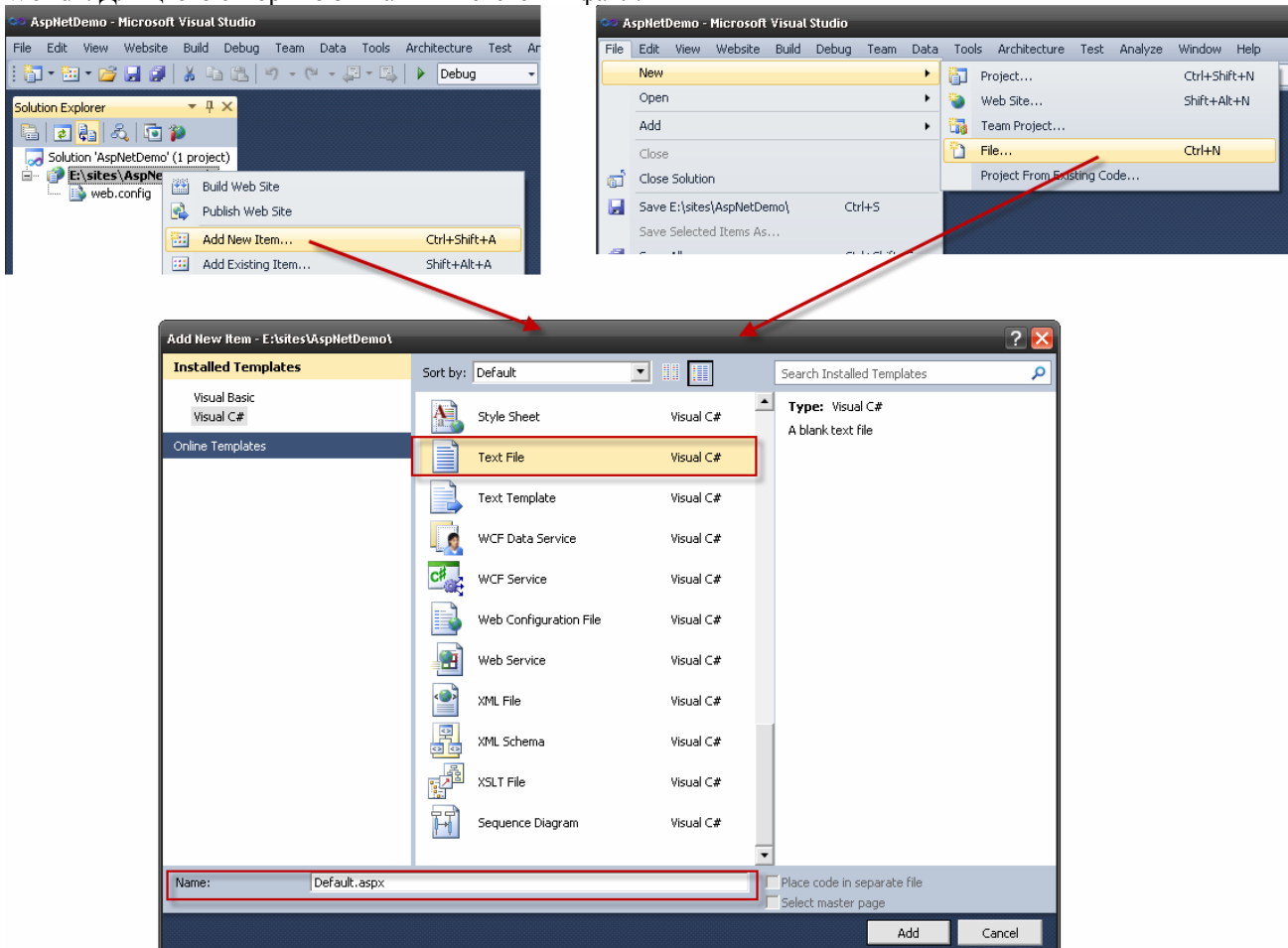
Від теорії до практики. Напишемо свою першу сторінку на ASP.NET. **Сторінка ASP.NET** – це простий текстовий файл з розширенням **.aspx**. Тому як тільки у вас запрацює IIS і .NET Framework SDK, ви легко зможете створювати сторінки ASP.NET в будь-якому текстовому редакторі.



Отже, створимо пустий проект з назвою **AppNetDemo**:



Тепер можна додати в наш проект файл з кодом, який буде сторінкою ASP.NET та виведе повідомлення «Hello World». Для цього створимо звичайний текстовий файл:





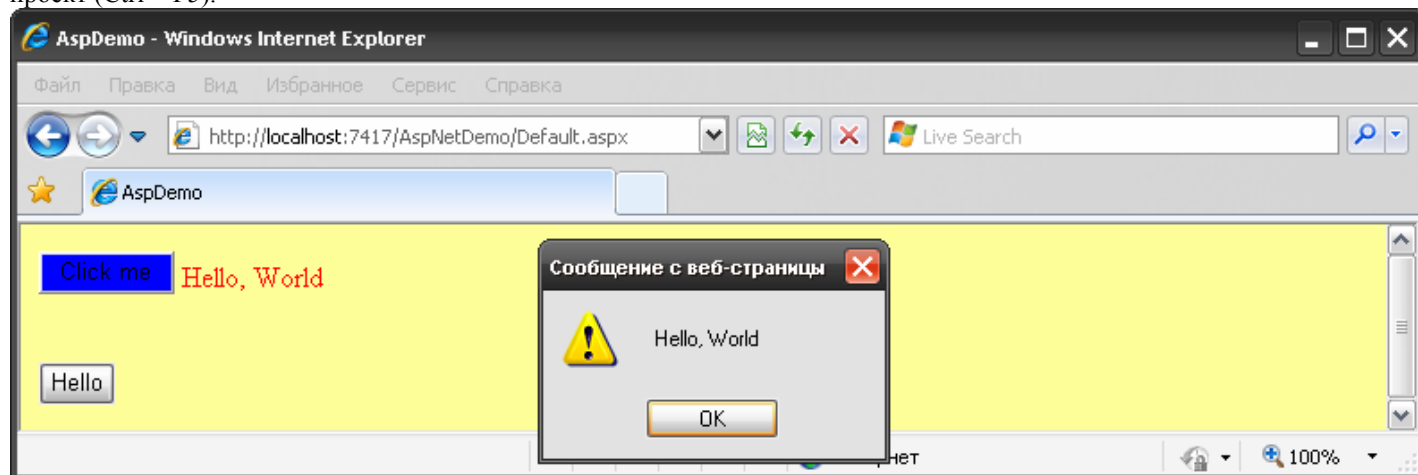
Збережемо цей файл з розширенням **.aspx** та напишемо наступний код:

DemoAsp.aspx

```
<%@ Page Language="C#" %>
<script language="C#" runat="server">
    void OnClick_Hello(object sender, EventArgs e){
        labelHello.ForeColor = System.Drawing.Color.Red;
        labelHello.Text = "Hello, World";
    }
</script>

<html>
<head>
    <title>AspDemo</title>
</head>
<body bgcolor="#ffff99">
    <form id="f" runat="server">
        <asp:Button ID="b" BackColor="Blue" Text="Click me"
            OnClick="OnClick_Hello" runat="server"/>
        <asp:Label ID="labelHello" Text="" runat="server" />
        <br /><br /><br />
        <input type="button" value="Hello" onclick="alert('Hello, World');" />
    </form>
</body>
</html>
```

Visual Studio містить власний вбудований веб-сервер, тому, щоб побачити результат, достатньо просто скопіювати проект (Ctrl + F5).



А тепер давайте розберемо код. Наша сторінка починається з директиви **@ Page**, яка дозволяє задати загальні атрибути і параметри компіляції для Web-форми. Далі йде код, який розділений на **дві частини**: перша частина містить код по роботі з елементами управління, що розміщуються на сторінці (в нашому випадку це обробник події OnClick), а друга – набір тегів html та власне опис серверних елементів управління ASP.NET. В принципі, код бажано виносити в окремий файл, але це ми зробимо пізніше.

ASP.NET дозволяє розміщувати на сторінці ряд елементів управління (наприклад, **Label**, **TextBox**, **Button** тощо), з якими власне і відбувається вся робота на сервері. Всі ці елементи управління ASP.NET в коді HTML будуть починатись з префікса **<asp:** та мати атрибут **runat** з значенням **server**. Доступ до серверних елементів управління можна здійснити як по імені, так і по ідентифікатору. Після чого можна змінювати їх властивості, призначати обробники подій тощо.

А тепер розглянемо отриманий HTML файл, який буде відображатись на стороні клієнта після запуску нашого ASP.NET додатку:

```
<html>
<head>
    <title>AspDemo</title>
</head>
<body bgcolor="#ffff99">
    <form name="f" method="post" action="Default.aspx" id="f">
<div class="aspNetHidden">
```



```
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwUKMTEwMjM4NDkyMQ9kFgICAQ9kFgICAw8PFgYeBFRleHQFDEhlbXvLCBxb3JsZB
4JRm9yZUNvbG9yCo0BHgRfIVNCAgRkZGQo8oQyKBfWPhl9CZaWg8uxZ5Bjyg==" />
</div>
<input type="submit" name="b" value="Click me" id="b"
style="background-color:Blue;" />
<span id="labelHello" style="color:Red;">Hello, World</span>
<br /><br /><br />
<input type="button" value="Hello" onclick="alert('Hello, World');" />
<div class="aspNetHidden">

<input type="hidden" name="__EVENTVALIDATION" id="__EVENTVALIDATION"
value="/wEWAgl4tsXrBAK+76ruDAtom55OhJF7yg6FOZWhEvay0E9j" />
</div>
<input type="submit" name="b" value="Click me" id="b"
style="background-color:Blue;" />
<span id="labelHello" style="color:Red;">Hello, World</span>
<br /><br /><br />
<input type="button" value="Hello" onclick="alert('Hello, World');" />
</form>
</body>
</html>
```

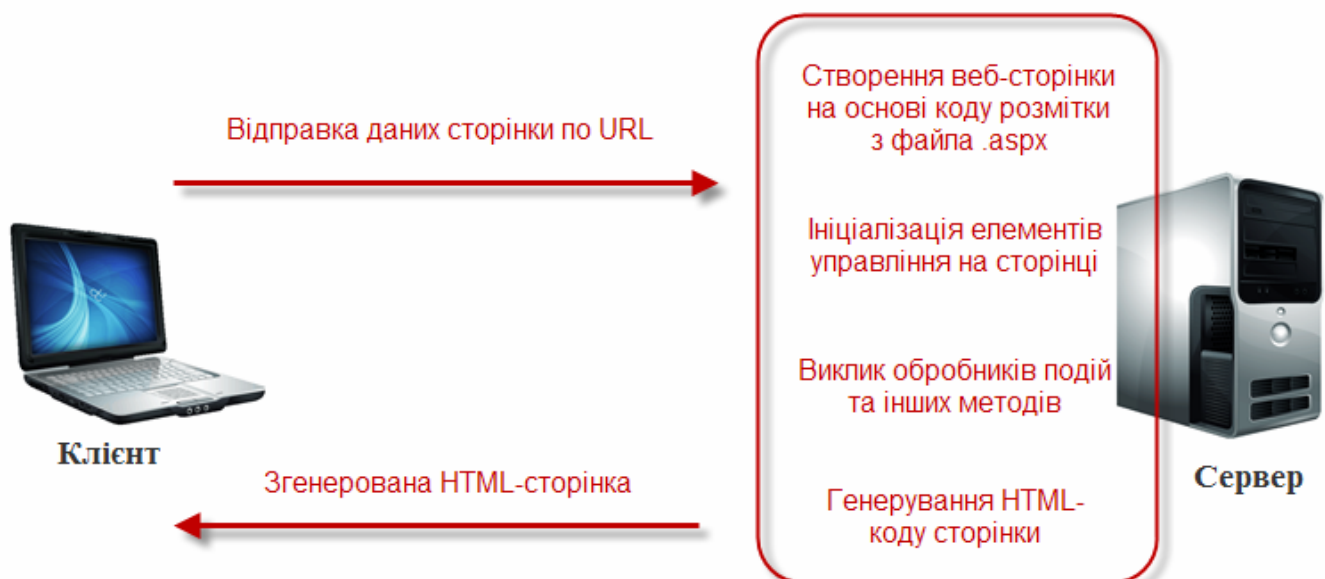
Як бачите, тут міститься лише HTML код, оскільки іншої мови браузер не розуміє. Тобто при генерації веб-сторінки, всі серверні елементи управління, які починаються з префіксами <asp: та мають встановлений атрибут runat="server" конвертуються у відповідні їм аналоги HTML. Якщо атрибут runat="server" відсутній, то співставлення такого серверного елемента управління не відбудеться, він буде відсутній на стороні клієнта і дані з нього обробити буде неможливо. Варто також відмітити, що вищепредставлений згенерований код буде **браузеронезалежним (!)**.

Якщо гарно подивитись на згенерований код, то можна побачити два прихованих поля з іменами **__VIEWSTATE** та **__EVENTVALIDATION**. Ці два поля необхідні для того, щоб відслідковувати зміни, які відбулись на стороні клієнта і обробити їх на сервері при поверненні, тобто викликати необхідні обробники.

В спеціальному полі **__VIEWSTATE** зберігається стан всіх серверних елементів управління. При поверненні сторінки на сервер, ASP.NET порівнює поточний стан елементів з тим, що був записаний у **__VIEWSTATE** і при необхідності викликає відповідні обробники.

Поле **__EVENTVALIDATION** служить для забезпечення додаткового рівня безпеки і містить вказівник на джерело події. Воно дозволяє здійснити додаткову перевірку того, вірна подія була викликана чи ні. Подібно до поля **__VIEWSTATE**, поле перевірки події містить хешоване значення, з ціллю запобігання змінам на сторінці клієнта.

Отже, серверні обробники потребують передачі форми на сервер. Після цього на формі виконуються всі необхідні методи, форма оновлюється і завантажується назад в браузер.





Варто відмітити, що для підвищення швидкодії ASP.NET оновлює лише необхідні елементи шляхом клієнтських методів JavaScript і DHTML. При цьому серверні обробники не призводять до перевантаження сторінки, а викликають лише клієнтський метод на JavaScript, який в якості параметрів приймає оновлені дані з сервера.

Більш детальне вивчення цього механізму відкладемо на потім. А зараз розглянемо структуру сторінки ASP.NET.

4. Структура сторінки ASP.NET. Директиви сторінки

А тепер перейдемо до більш детального знайомства з структурою сторінки ASP.NET. Отже, сторінка ASP.NET може містити:

- ✓ директиви сторінки;
- ✓ блоки серверного коду;
- ✓ блоки виконання коду;
- ✓ серверні елементи управління ASP.NET;
- ✓ серверні коментарі;
- ✓ серверні директиви включення;
- ✓ HTML-код.

Директиви сторінки використовуються для встановлення окремих параметрів сторінки ASP.NET та інших допоміжних цілей, які необхідні для компіляції сторінки. Наприклад, їх можна використовувати для вказування мови програмування по замовчуванню, для трасування, відлагодження коду, підключення просторів імен тощо. Початок сторінкової директиви позначається символами `<%@`, а закінчення - символами `%>`. Директиви можуть знаходитися в будь-якій частині сторінки, але як правило вони вказуються на її початку.

До найпоширеніших директив сторінки можна віднести наступні директиви:

- **`<%@ Page %>`** - використовується для задання загальних атрибутів та параметрів компіляції у веб-формі. Вона може мати ряд атрибутів, серед яких можуть бути наступні:
 - **Language** - вказує мову програмування по замовчуванню;
 - **CodeFile** - вказує файл з з розподіленням кодом;
 - **Inherits** - вказує простір імен в розподіленому файлі (використовується сумісно з CodeFile).
Наприклад, `<%@ Page Language="C#" CodeFile="Default.aspx.cs" Inherits="_Default" %>`
 - **Trace** - якщо трасування коду дозволено, на сторінку разом з вмістом виводиться додаткова інформація про її виконання;
 - **Debug** - вмикає можливість відлагодження сторінки.
- **`<%@ Import %>`** - використовується для явного підключення вказаного простору імен, оскільки по замовчуванню в сторінку ASP.NET включаються не всі простори імен. Наприклад, `<%@ Import Namespace="System.Web.Mail" %>`. Слід також відмітити, що підключення простору імен за допомогою директиви `@Import` потрібно використовувати лише у випадку, якщо програмний код знаходиться на сторінці ASP.NET (*.aspx), а не в зовнішньому файлі (*.cs).

З іншими сторінковими директивами та їх атрибутами ми познайомимось по ходу вивчення ASP.NET.

Блоки серверного коду по суті включають в себе всю логіку сторінки ASP.NET, всі оголошення глобальних змінних підпрограм та функцій. Ці блоки обмежені тегами `<script>` з обов'язковим атрибутом `runat="server"`.

```
<script runat="server">
    //код
</script>
```

Наприклад:

```
<script language="C#" runat="server">
    string buffer = "Hello ";

    void OnClick_Hello(object sender, EventArgs e){
        labelHello.ForeColor = System.Drawing.Color.Red;
        labelHello.Text = "Hello, World";
    }
</script>
```

Блоки серверного коду транслюються в члени класу, який відповідає сторінці. Для нашого прикладу, клас буде містити рядкове поле `buffer` та метод `OnClick_Hello()`.

Та розміщення таких блоків суперечить концепції приховання коду, яка передбачає розміщення зв'язаного з сторінкою коду в окремому класі, похідному від класу `Page`, та винесений в окремий файл. При дотриманні правил, в директиві сторінки або в тезі `<script>`, в **атрибуті `src`** слід вказати суміжний файл.



```
<%@ Page language="мова_кодування" src="ім'я_файлу" %>
або
<script runat="server" language="мова_кодування" src="ім'я_файлу"></script>
```

Та слід пам'ятати, що в останньому варіанті прив'язки зовнішнього файлу, код в самому блоці `<script>` буде проігнорований.

Якщо необхідно в HTML-код ASP.NET сторінки вбудувати код для виконання, то такий код необхідно помістити в **блок виконання коду**. Такі блоки можуть бути двох типів:

- **Внутрішньотекстовий код**, який виконує оператор або групу операторів. Такий тип коду починається з символів `<%` і закінчується символами `%>`.

```
<form id="form1" runat="server">
    <% Response.Write("Hello, world"); %>

    <% for (int i = 0; i < 4; i++)
        Response.Write(getData(i)); %>

    <% for (int i=0; i<10; i++) { %>
        <font size="<%=i %>"> Hello World! </font>
    <% } %>
</form>
```

- **Внутрішньотекстові вирази**, які обчислюють деякий вираз і результат передається в вихідний потік. Такий тип коду починається з символів `<%=` і закінчується символами `%>`.

```
<form id="form1" runat="server">
    <asp:TextBox ID="txtTypeAlign" Text="left" runat="server" AutoPostBack="true" />
    <br/><br/>
    <h1 align="<%= txtTypeAlign %>"> Hello, world </h1>
</form>
```

Елементи управління ASP.NET можуть розміщуватись в HTML-кодї сторінки ASP.NET, подібно до звичайних елементів HTML. Єдина умова – всі серверні елементи управління повинні знаходитись у формі та мати властивість `runat="server"`, власне як і сама форма. Але, не забувайте, що на сторінці ASP.NET не повинно бути більше ОДНІЄЇ форми типу `<form runat="server">`.

Серверні коментарі на сторінці ASP.NET починаються символами `<%--`, а закінчуються символами `--%>`.
Наприклад:

```
<form id="form1" runat="server">
    <asp:TextBox ID="txtTypeAlign" Text="left" runat="server" AutoPostBack="true" />
    <%-- <h1 align="<%= txtTypeAlign %>"> Hello world</h1> --%>
</form>
```

Такі коментарі, на відміну від звичайних HTML-коментарів неможна переглянути за допомогою команди **View Source** веб-браузера. Вони доступні для перегляду лише на сервері.

Серверні директиви включення дозволяють вставляти вміст вказаного файлу в ASP.NET сторінку. Таким файлом може бути інша веб-сторінка ASP.NET (*.aspx), файли користувацьких елементів управління (*.ascx) та файли додатку Global.asax.

Для цих цілей використовується директива **Include**, яка може використовувати одну з **двох форм запису**:

1. При підключенні файлу, ім'я якого являється **фізичним шляхом**. Тобто підключаємий файл знаходиться в тому ж каталозі, що і файл веб-сторінки, або в його підкаталозі. При цьому дозволяється вказувати відносний шлях.

```
<!-- #Include file="myfile.aspx" -->
```

2. При підключенні файлу, ім'я якого являється **віртуальним шляхом**, тобто задається по відношенню до віртуальної папки веб-вузла. Цей шлях також може бути відносним. Наприклад, якщо у віртуальному каталозі `wwwroot` знаходиться підкаталог `sites`, файл з цього підкаталога можна включити наступним чином:

```
<!-- #Include virtual="/sites/myfile.aspx" -->
```




Директива Include – це фактично директива препроцесора, тому її можна використовувати лише для статичного підключення файлів. В зв'язку з цим для підключення файлів рекомендується використовувати засоби ASP.NET.

Крім всіх вищеописаних структурних елементів, на сторінці ASP.NET можна розмішувати звичайний **HTML-код**, який компілюється разом з іншими її елементами. HTML-код визначає статичну частину веб-сторінки і включає в себе HTML-теги та звичайний текст.

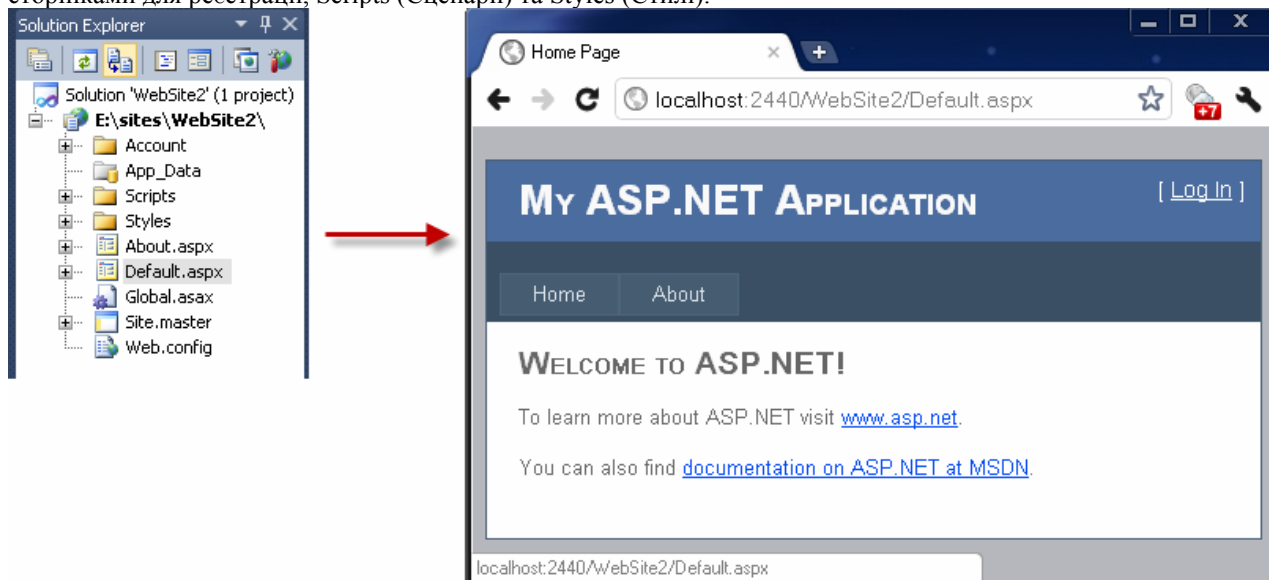
Якщо необхідно на веб-сторінці вивести звичайний статичний текст, то слід скористатись елементом управління **Literal**. Його властивість **Text** буде містити власне текст, який буде виводитись на веб-сторінку. Наприклад:

```
<form id="form1" runat="server">
    <asp:Literal ID="L1" runat="server" Text="Hello" />
</form>
```

5. Створюємо сторінку ASP.NET засобами Visual Studio 2010

А тепер спробуємо створити сторінку ASP.NET засобами Visual Studio 2010. Для цього оберемо пункт головного меню **File/New/Web Site**, після чого вказуємо мову кодування (в меню зліва) та необхідний тип проекту, тобто необхідний шаблон. У версії Visual Studio 2010 існує кілька видів шаблонів, відмінність між якими тільки в тому, які файли будуть створені по замовчуванню. Основними шаблонами для створення простих веб-сайтів є:

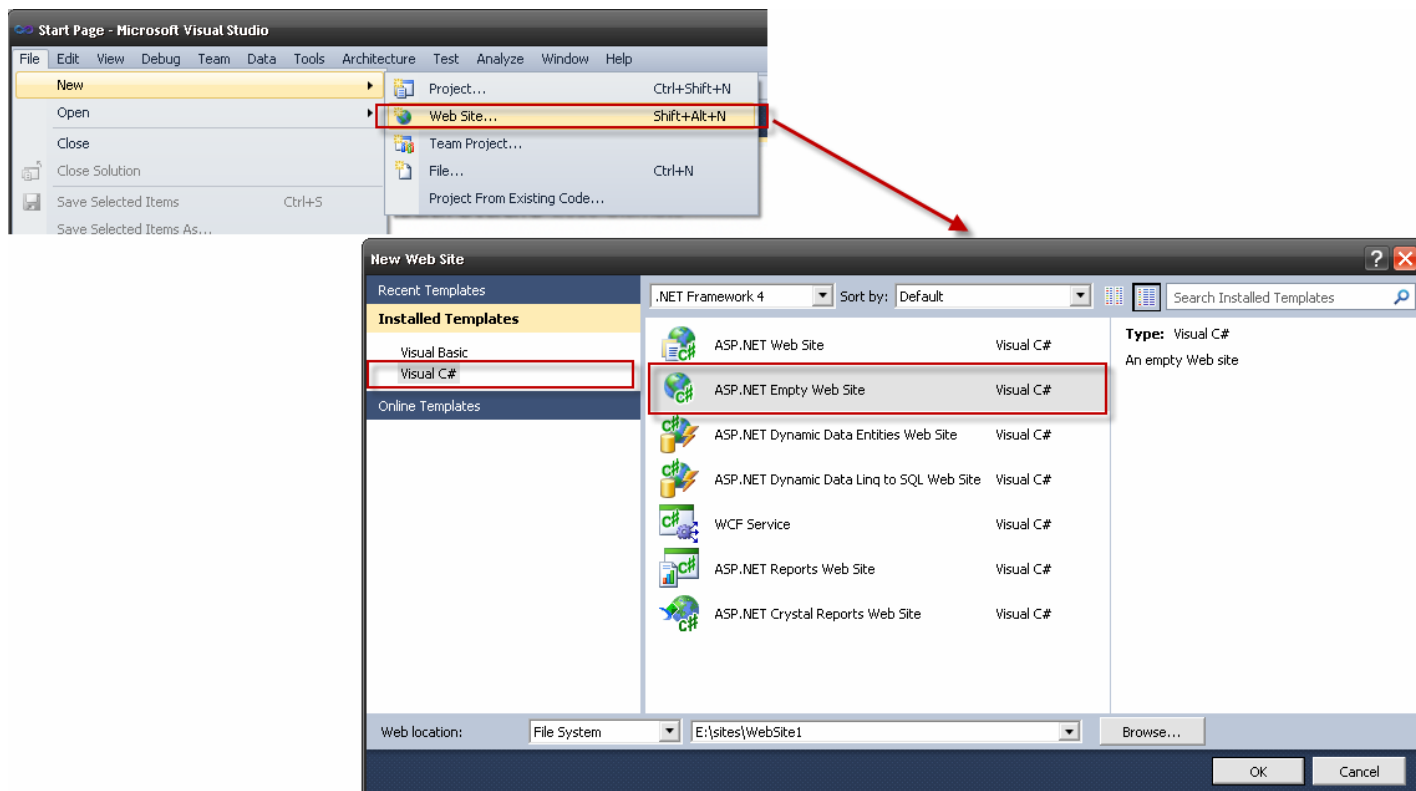
1. **ASP.NET Web Site (веб-сайт ASP.NET)** – створюється повнофункціональний сайт, який включає в себе головну сторінку, разом з сторінками Default.aspx та About.aspx. Він містить також папки Account (Облікові записи) з сторінками для реєстрації, Scripts (Сценарії) та Styles (Стили).



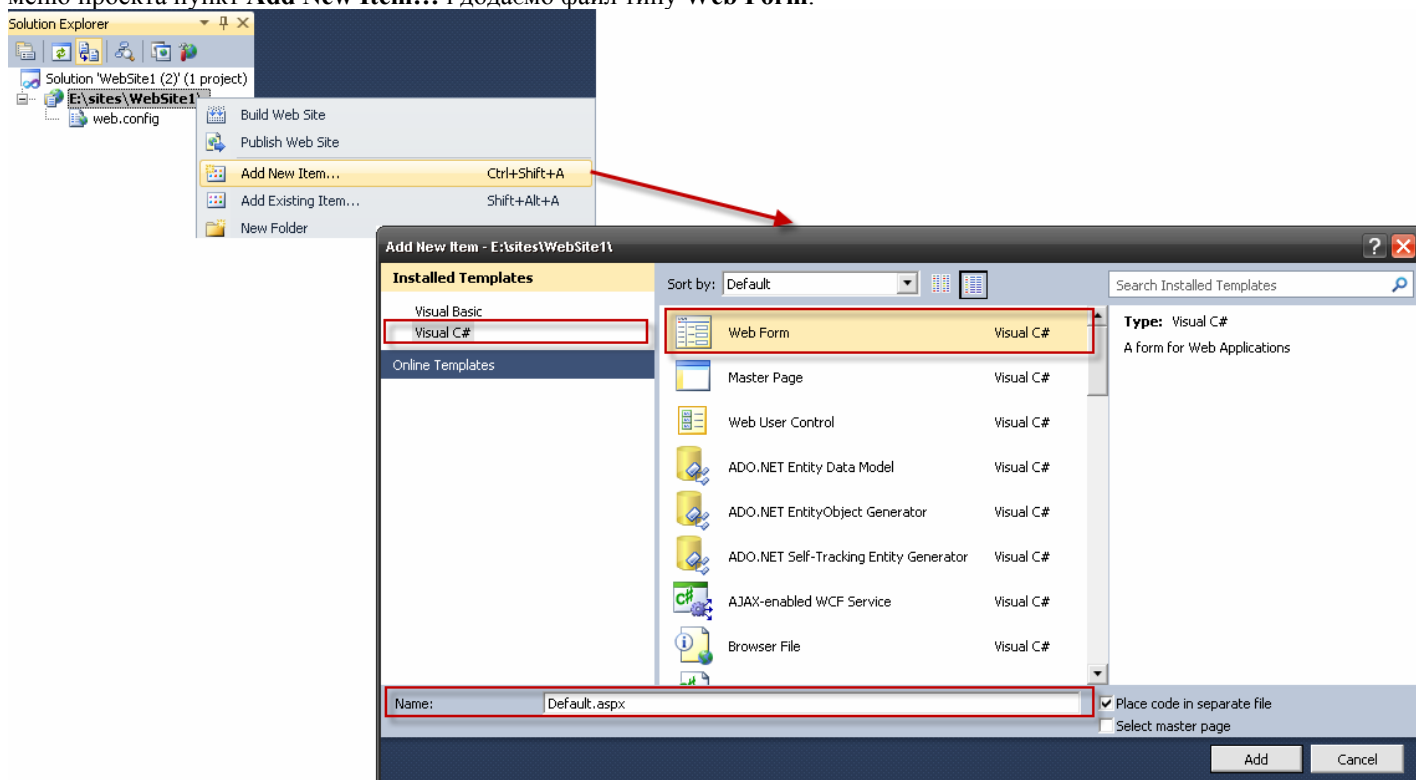
2. **ASP.NET Empty Web Site (пустий веб-сайт ASP.NET)** – створює майже пустий веб-сайт, який містить лише файл конфігурації web.config.



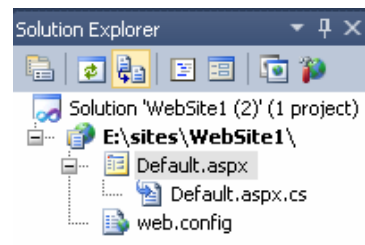
Оскільки ми хочемо хоч чогось навчитись, обираємо пустий шаблон для веб-сайту, його місцезнаходження (ним може бути файлова система, локальний IIS, FTP або веб-сайт), вводимо його назву та натискаємо кнопку «ОК».



Після цього додаємо файл веб-форми. Для цього обираємо пункт головного меню **File/New/File** або в контекстному меню проекту пункт **Add New Item...** і додаємо файл типу **Web Form**.



Після натиснення кнопки «ОК» створиться два файли: **default.aspx** з кодом сторінки (аналогічним тому, що ми писали в п.3 уроку) і **default.aspx.cs** з кодом на C#. Тобто Visual Studio використовує принцип розділення коду (code behind).





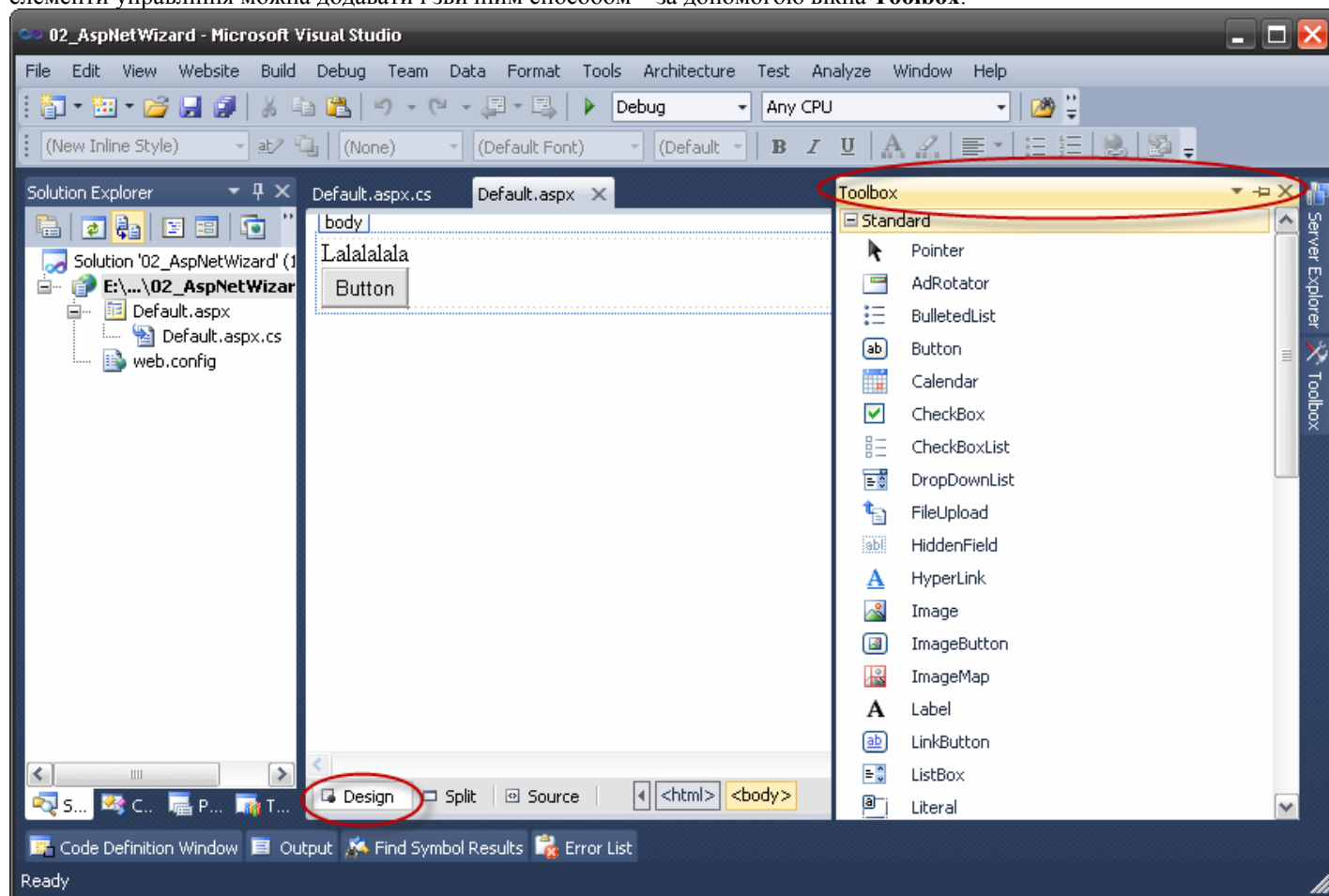
З кодом першого файлу (default.aspx) ви вже знайомі, а код файла **default.aspx.cs** буде виглядати так:

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

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

В цьому файлі розміщуються підключаємі простори імен та частковий клас **_Default** похідний від класу **Page**, який містить метод **Page_Load()**. Метод **Page_Load()** виконується ще до відображення сторінки, тому в нього рекомендується вставляти код, який виконує ініціалізацію полів форми.

Ну що ж, давайте додамо кілька серверних елементів управління на сторінку і трішки з ними попрацюємо. Доречі, елементи управління можна додавати і звичним способом – за допомогою вікна **Toolbox**.



Отже, після всього зробленого наш код обробки матиме наступний вигляд:

default.aspx

```
<%@ 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/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Asp Net Wizard</title>
</head>
```



```
<body>
  <form id="form1" runat="server">
    <div>
      <asp:Label ID="Label1" runat="server" Text="Lalalalala"></asp:Label><br />
      <asp:Button ID="Button1" runat="server" Text="Button" OnClick="Button1_Click" />
    </div>
  </form>
</body>
</html>
```

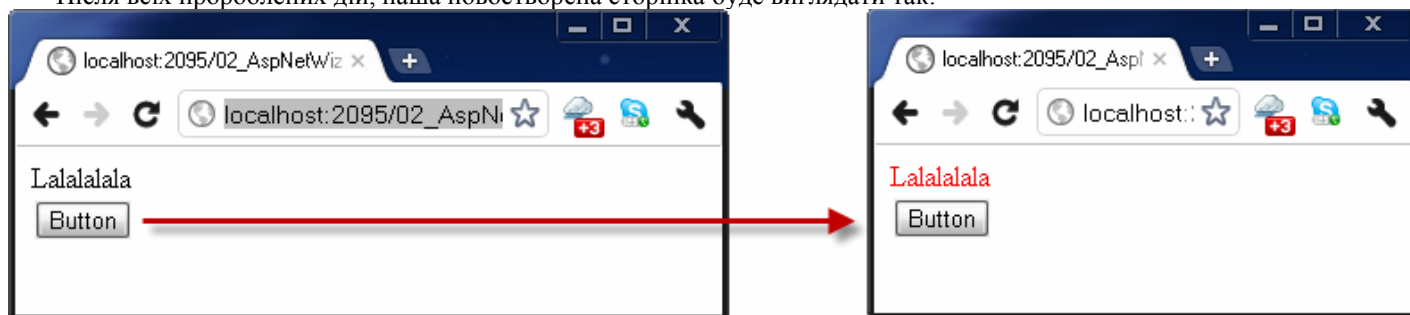
default.aspx.cs

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

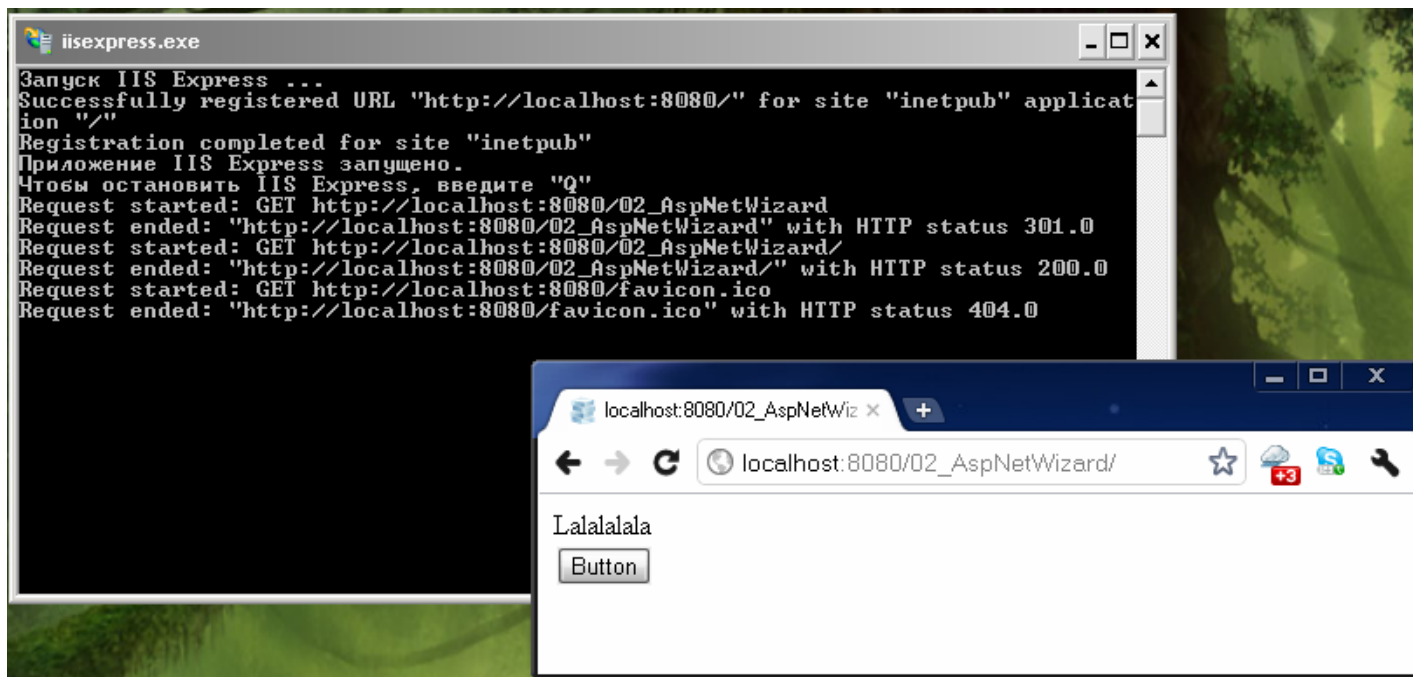
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e) {}
    protected void Button1_Click(object sender, EventArgs e)
    {
        Label1.ForeColor = System.Drawing.Color.Red;
    }
}
```

Як вже було сказано, Visual Studio 2010 має свій власний Web Server, під яким вона запускає ASP.NET сторінки. Тому активізувати IIS для таких цілей немає необхідності, достатньо лише запустити веб-сайт на виконання за допомогою комбінації клавіш Ctrl+F5.

Після всіх пророблених дій, наша новостворена сторінка буде виглядати так:



Якщо необхідно переглянути веб-сторінку без використання Visual Studio, тоді слід скористатись встановленим і налаштованим в системі веб-сервером IIS. Для цього необхідно запустити веб-сервер та розмістити вашу веб-сторінку у одному з створених віртуальних каталогів або в директорії по замовчуванню для веб-сайтів ([див. розділ 2. Встановлення IIS 7](#)). Після цього ввести в рядку пошуку браузера URL адресу сторінки:



І ще одне важливе зауваження і обмеження сторінок ASP.NET. На сторінці не може бути більше одного дескриптора `<form runat="server">`. Тобто на сторінці ASP.NET неможна групувати кілька форм, це призведе до помилки.

6. Поняття Web-форми

6.1. Форма HTML

Основою будь-якої веб-сторінки на ASP.NET являється веб-форма або HTML форма, яка фактично і визначає те, який графічний інтерфейс буде перед собою бачити користувач. Тому ASP.NET сторінку офіційно називають **веб-формою**. На веб-формі розміщуються елементи управління ASP.NET, HTML-теги та додатковий програмний код, який також включає в себе обробку подій, які генерують серверні елементи управління, подібно до подій Windows Forms. З цих трьох основних складових, як ви вже бачили, лише теги HTML однаково відображаються на стороні клієнта. Щодо серверних елементів управління, то після обробки їх сервером, механізм ASP.NET генерує для них відповідні HTML-теги, які потім відсилаються клієнтському браузеру.

Варто відмітити, що в ASP.NET 4.0 веб-сторінки на 100% відповідають стандарту XHTML Strict, якщо правда розробник сам буде дотримуватись правил стандарту при розробці.

6.2. Життєвий цикл Web-форми. Сторінкова подія Load

Після запуску веб-сторінки ASP.NET на виконання, вона проходить ряд етапів обробки: від ініціалізації та визначення елементів управління до виконання коду обробників подій та вивільнення ресурсів, зайнятих сторінкою. Тобто кожна веб-сторінка має свій життєвий цикл, який потрібно знати для грамотного управління даними сторінки. Прослідкувати за життєвим циклом вашої сторінки можна, додавши до директиви `@Page` атрибути `Trace="true"` та `TraceMode="SortByTime"`. От як він буде виглядати при початковому завантаженні нашої веб-сторінки:



Request Details

Session Id:	4uap2bboticrklackfdvtn0i	Request Type:	GET
Time of Request:	25.10.2011 16:04:40	Status Code:	200
Request Encoding:	Unicode (UTF-8)	Response Encoding:	Unicode (UTF-8)

Trace Information

Category	Message	From First(s)	From Last(s)
aspx.page	Begin PreInit		
aspx.page	End PreInit	8,82793762894446E-05	0,000088
aspx.page	Begin Init	0,000152253987587808	0,000064
aspx.page	End Init	0,000231314315087532	0,000079
aspx.page	Begin InitComplete	0,000283834956677455	0,000053
aspx.page	End InitComplete	0,000335796868037698	0,000052
aspx.page	Begin PreLoad	0,000444749262825303	0,000109
aspx.page	End PreLoad	0,000495593713726186	0,000051
aspx.page	Begin Load	0,000548952450660629	0,000053
aspx.page	End Load	0,000605104838743472	0,000056
aspx.page	Begin LoadComplete	0,000655390559414674	0,000050
aspx.page	End LoadComplete	0,000706793740545237	0,000051
aspx.page	Begin PreRender	0,0007581969216758	0,000051
aspx.page	End PreRender	0,000869663602496965	0,000111
aspx.page	Begin PreRenderComplete	0,000927771546383688	0,000058
aspx.page	End PreRenderComplete	0,000986717585614932	0,000059
aspx.page	Begin SaveState	0,00179296530704321	0,000806
aspx.page	End SaveState	0,00209272407526655	0,000300
aspx.page	Begin SaveStateComplete	0,00217066694230691	0,000078
aspx.page	End SaveStateComplete	0,00222402567924136	0,000053
aspx.page	Begin Render	0,0022759875906016	0,000052
aspx.page	End Render	0,00427819736865998	0,002002

Тепер наглядно видно, що навіть пуста веб-сторінка «живе бурхливим життям». Весь її «тернистий шлях» можна побачити в розділі **Trace Information**, в полі **Message**, де розміщується список тих подій, які відбулися для даної сторінки.

Отже, основні етапи життєвого циклу веб-сторінки ASP.NET 4.0 наступні:

Етап	Опис
Запит сторінки по HTTP	Він відбувається, коли користувач вводить URL адресу на запит сторінки ASP.NET. Фактично цей етап відбувається ще до початку життєвого циклу веб-сторінки.
Початок (запуск) життєвого циклу	<p>На даному етапі відбувається завантаження персональних даних користувача і схем, щоб сторінка відображалась згідно налаштувань користувача. Також заповнюються значеннями властивості Response, Request та властивості UICulture, яка містить ID користувацького інтерфейсу.</p> <p>В цей період встановлюється також властивість IsPostBack, яка дозволяє визначити була сторінка завантажена вперше чи в результаті зворотнього виклику (тобто користувач відправив дані на сервер).</p> <p>На початковому етапі життєвого циклу генерується подія PreInit. Це перша подія в життєвому циклі сторінки, яка фактично здійснює її попередню ініціалізацію. Тут можна динамічно створювати елементи управління, встановлювати тему сторінки та вказувати головну сторінку.</p>
Ініціалізація сторінки	На цьому етапі відбувається ініціалізація всіх елементів управління на сторінці, застосовуються теми оформлення тощо, тобто відбувається процес створення веб-сторінки.



	<p>Сторінка генерує наступні сторінкові події:</p> <ol style="list-style-type: none"> 1. Init – дозволяє зчитати або проініціалізувати властивості елементів управління. Але насправді цю подію рідко використовують для цих цілей, оскільки об'єкти елементів управління ще не створені і звернутись напругу до них складно. 2. InitComplete – виникає в кінці ініціалізації сторінки, тобто після того, як сторінка і всі елементи управління були проініціалізовані.
Завантаження	<p>На даному етапі здійснюється перевірка: сторінка завантажена вперше чи в результаті зворотнього виклику. Якщо поточний запит являється результатом зворотнього виклику, сервер, враховуючи інформацію про стани ViewState та ControlState, передає дані елементам управління.</p> <p>Сторінкові події на етапі завантаження:</p> <ol style="list-style-type: none"> 1. PreLoad - генерується на етапі попереднього завантаження, тобто після завантаження всіх відправлених даних, включених в Request. 2. Load – генерується при завантаженні сторінки ASP.NET. На цьому етапі можна проводити ініціалізацію змінних, динамічних елементів управління, створених раніше, або ж створювати динамічно нові елементи управління, відслідковувати зміни, які відбулись при роботі з елементами управління, і здійснювати відповідно до цих змін дії тощо. Визначити поточний стан сторінки можна за допомогою властивості Page.IsPostBack.
Перевірка достовірності даних (валідація)	<p>Якщо на сторінці присутні спеціальні елементи управління для перевірки коректності вводу даних, тоді сторінка проходить ще один етап життєвого циклу – валідацію. На даному етапі викликається метод Validate() цих елементів управління і перевіряється коректність відправлених даних.</p>
Обробка подій	<p>Оскільки сторінка вже повністю завантажена, на цьому етапі відбувається виклик всіх обробників подій, які здійснив користувач після останньої передачі даних на сервер. До таких подій відноситься зокрема відправка форми на сервер кнопкою Submit або дією на іншому елементі управління, який генерує відправку даних на сервер чи має встановлену властивість AutoPostBack.</p>
Попередня візуалізація	<p>На етапі попередньої візуалізації генеруються наступні події:</p> <ol style="list-style-type: none"> 1. LoadComplete – відбувається після завершення завантаження сторінки, тобто після виконання всіх обробників подій і вказує на початок формування представлення веб-сторінки. 2. PreRender – виникає після створення всіх елементів управління сторінки. В обробнику даної події можна внести кінцеві зміни, які стосуються зовнішнього вигляду сторінки. 3. PreRenderComplete – виникає після прив'язки даних до елементів управління (DataBinding).
Візуалізація	<p>Відбувається генерування сторінки, яка буде передана клієнту. На етапі візуалізації ASP.NET викликає метод Render() для кожного серверного елемента управління.</p>
Очистка сторінки	<p>Цей етап відбувається після створення веб-сторінки і завантаження її в браузер клієнта. Вона служить для вивільнення ресурсів, зайнятих сторінкою: закриття відкритих з'єднань з базою даних, файлів тощо. При цьому відбувається подія Unload, яка викликається спочатку для всіх елементів управління, а потім і для веб-сторінки в цілому.</p> <p>Після завершення вивантаження сторінки запускається подія Disposed для знищення сторінки. На цьому життєвий цикл веб-сторінки ASP.NET завершується.</p>

Більш детально переглянути список методів та подій класу Page, які використовуються на етапах життєвого циклу сторінки, можна в MSDN по адресі <http://msdn.microsoft.com/ru-ru/library/ms178472.aspx> або нижче:



	Методы Page	События Page	Методы и события элементов управления
Запуск	Construct ◊ ProcessRequest ◊ InitializeCulture ◊ DeterminePostBackMode ◊ OnPreInit ◊ OnInit ◊ TrackViewState ◊ OnInitComplete ◊	PreInit + Init + InitComplete +	Init + LoadViewState ◊ IPostBackDataHandler. LoadPostData ◊ Load +
Загрузка	LoadPageStateFrom PersistenceMedium ◊ LoadViewState ◊ ProcessPostData ◊ OnPreLoad ◊ OnLoad ◊	PreLoad + Load +	Load +
Обработка событий	RaisePostBackEvent ◊		События изменения элементов управления
Проверка	Validate ◊		
Предварительная визуализация	OnLoadComplete ◊ OnPreRender ◊ OnPreRenderComplete ◊ SaveViewState ◊ SavePageStateTo PersistenceMedium ◊ OnSaveStateComplete ◊	LoadComplete + PreRender + PreRenderComplete + SaveStateComplete +	PreRender + События привязки данных SaveViewState ◊
Визуализация	RenderControl ◊ Render ◊ RenderChildren ◊		Render ◊
Выгрузка	OnUnload ◊ Dispose ◊		Unload +

Підсумовуючи все вищесказане, наведемо невеликий приклад обробки однієї із сторінкових подій. Для цього обробимо подію **Load**, яку найчастіше використовують для ініціалізації елементів управління на сторінці, оскільки вона **відбувається кожен раз, коли сторінка завантажується на сервер**. Наші кроки будуть наступними.

По-перше, сторінки ASP.NET підтримують автоматичну прив'язку подій і методів-обробників при їх виклику, якщо останні відповідають правилам іменування: **Page_подія**. Наприклад, Page_Load, Page_Init тощо.

По-друге, щоб визначити була сторінка завантажена вперше чи вона викликається в результаті дій користувача, що призвели до повторного запиту на сервер, використовується властивість **IsPostBack**. У відповідності до її значення ми ініціалізуємо елементи управління. Наприклад:

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        //якщо сторінка завантажена вперше
        if (!Page.IsPostBack) {
            Label1.Text = "";    //мітка буде пуста
        }
    }
}
```




6.3. Клас System.Web.UI.Page

Як ви вже могли помітити, основою будь-якого веб-додатку ASP.NET являється сторінка, яка є об'єктом класу **Page** та розміщується в просторі імен **System.Web.UI**. Сам клас **Page** являється елементом управління, який діє як користувацький інтерфейс для веб-додатків.

Оскільки кожна сторінка ASP.NET походить від класу **System.Web.UI.Page**, створюючи свої сторінки, ви повинні зробити їх спадкоємцями цього класу. Таким чином, ваша сторінка успадковує всю функціональність, включаючи набір корисних методів та властивостей, якими володіє клас **System.Web.UI.Page**. Серед найбільш поширених її **властивостей** виділяють наступні:

- **Application** – містить посилання на поточний додаток, який асоціюється з об'єктом типу **HttpApplicationState**.
- **Cache** – містить посилання на кеш поточного веб-додатку, який представлений об'єктом типу **Cache**. Він дозволяє зберігати та отримувати довільні дані в послідовних запитах.
- **Server** – містить посилання на об'єкт типу **HttpServerUtility**, за допомогою якого можна отримати інформацію про помилки сервера, ім'я комп'ютера, на якому запускається сторінка тощо.
- **Session** – містить посилання на об'єкт типу **Session**, який надає відомості про поточний сеанс веб-додатку.
- **Controls** – містить посилання на колекцію серверних елементів управління, що розміщуються на поточній сторінці, тобто повертає об'єкт типу **ControlCollection**.
- **Request** – повертає запит (об'єкт типу **HttpRequest**), який клієнт передає серверу для виводу поточної сторінки. Він містить всю інформацію про клієнта, включаючи налаштування браузера, cookie-файли та дані форми.
- **Response** – відповідь веб-сервера на запит клієнта, яка представлена об'єктом типу **HttpResponse**.
- **User** – надає інформацію про користувача, який робить запит до сервера. Дана властивість реалізує інтерфейс **IPrincipal**, який в залежності від використовуваного користувачем типу аутентифікації надає необхідний об'єкт.
- **Trace** – об'єкт трасування типу **TraceContext** для поточного запиту, який містить всю інформацію про виконання поточного запиту.
- **IsPostBack** – містить значення, яке вказує на те, була сторінка завантажена вперше (**false**) чи повторно, тобто в наслідок певних дій користувача (**true**).

Це тільки невеликий перелік властивостей. Про них та про інші властивості класу **System.Web.UI.Page** ми ще не раз поговоримо на наступних заняттях.

7. Поширені простори імен ASP.NET

В бібліотеці .NET міститься певний набір просторів імен, які приймають участь у створенні веб-додатків на ASP.NET. Всі їх можна умовно поділити на **три групи**:

1. для роботи з основними елементами веб-додатків, наприклад, типи системи безпеки, типи для роботи з протоколом HTTP тощо;
2. для роботи з елементами управління;
3. для роботи з веб-службами.

До **просторів імен, які найчастіше використовуються для створення веб-додатків ASP.NET**, можна віднести:

- **System** – містить всі базові типи даних та ряд інших допоміжних класів.
- **System.Configuration** – містить типи для роботи з файлами конфігурації проекту (файлами **web.config**).
- **System.Web** – містить базові типи для організації взаємодії між браузером та веб-сервером: запит і відповідь (**HttpRequest** та **HttpResponse**), передача файлів, робота з cookie-файлами (**HttpCookie**) тощо.
- **System.Web.Caching** – простір імен, який містить типи, які використовуються для кешування вмісту веб-сторінок.
- **System.Web.Security** – містить набір типів, які реалізують засоби безпеки веб-додатків: аутентифікація і авторизація користувачів.
- **System.Web.Services** – містить набір типів для роботи з веб-службами.
- **System.Web.SessionState** – містить класи для реалізації стану сесії.
- **System.Web.UI** – містить базові типи для побудови користувацького інтерфейсу на веб-сторінці.
- **System.Web.UI.HtmlControls** – даний простір імен містить класи елементів управління HTML.
- **System.Web.UI.WebControls** – містить класи **Web** елементів управління.

Ці та інші простори імен ми будемо також розглядати з вами по мірі вивчення курсу ASP.NET 4.0.

8. Елементи управління

8.1. Клас System.Web.UI.Control. Серверні елементи управління, їх класифікація

А тепер перейдемо безпосередньо до створення елементів управління і роботі з ними. По-перше, всі елементи управління ASP.NET являються класами, які похідні від базового класу **Control** з простору імен **System.Web.UI**. По-друге, всіх їх можна **поділити на дві групи**:



- **Веб-елементи управління** – ці елементи управління належать ASP.NET. Вони розміщуються в просторі імен **System.Web.UI.WebControls** та починаються з префікса <asp:, але на сторінці клієнта вони присутні у вигляді стандартних HTML-тегів.
- **HTML-елементи управління** – ці елементи управління відповідають стандартним HTML-тегам. Відмінність від перших полягає в тому, що вони обов'язково повинні містити атрибут `runat="server"`, інакше на сервері він не буде створений і не можна буде до нього доступитись. Розміщуються ці елементи управління в просторі імен **System.Web.UI.HtmlControls**.

Оскільки HTML-елементам управління відповідають стандартні теги HTML, вони не потребують додаткового перетворення при відправленні даних на сторону клієнта, тому їх дуже зручно використовувати для оптимізації. Але веб-елементи управління значно зручніші у використанні та надають значно більше можливостей, ніж елементи управління HTML, зокрема і у налаштуванні зовнішнього вигляду.

Серверні елементи управління можна створювати засобами HTML, за допомогою вікна Toolbox або динамічно в коді, додаючи кожен окремий елемент управління в колекцію **Controls** певної форми.

8.2. Елементи управління Web. Простір імен System.Web.UI.WebControls

Як вже було сказано, всі класи веб-елементів управління розміщуються в просторі імен **System.Web.UI.WebControls** та всі вони похідні від базового класу **WebControl**. Цей клас містить набір загальних властивостей, методів та подій для всіх веб-елементів управління. До поширених **властивостей** класу **WebControl** можна віднести:

- **AccessKey** – дозволяє встановити комбінацію клавіш для швидкого доступу до елемента управління;
- **Attributes** – містить колекцію атрибутів елемента управління;
- **BackColor** та **ForeColor** – дозволяє встановити колір фону та тексту елемента управління;
- **BorderColor**, **BorderStyle**, **BorderWidth** – дозволяють встановити властивості рамки контролю, а саме, його колір, стиль та ширину;
- **CssClass** – дозволяє встановлювати стиль CSS для елемента управління;
- **Enabled** – дозволяє увімкнути (зробити активним; true) або вимкнути (false) елемент управління;
- **Font** – повертає властивості шрифту, який використовує елемент управління;
- **Height** та **Width** – дозволяє встановити висоту та ширину контролю.

Всі веб-елементи управління можна також розділити на окремі **підгрупи**:

- базові (прості) елементи управління Web;
- повнофункціональні елементи управління Web;
- елементи перевірки коректності вводу;
- спеціалізовані елементи управління, до яких можна віднести елементи управління даними, входом в систему, AJAX тощо.

Оскільки їх перелік доволі великий, вивчати їх будемо поступово. А розпочнемо ми з простих веб-елементів управління, до яких **відносять**:

- Label – мітка;
- TextBox – текстове поле;
- Button – кнопка та її різновиди: LinkButton та ImageButton;
- HyperLink – гіперпосилання;
- HiddenField – приховане поле;
- Image – зображення;
- CheckBox – флажок;
- RadioButton – перемикач;
- списки: ListBox, CheckBoxList, RadioButtonList, DropDownList;
- BulletedList - нумерований список;
- таблиці: Table, TableCell, TableRow.

8.2.1. Елементи управління Label та TextBox. Кнопки (Button)

Розпочнемо ми знайомство з таких елементів управління як підпис (мітка) та текстове поле.

За звичайний текст (підпис) на сторінці відповідає елемент управління **Label**, вміст якого відображається у властивості **Text**. Інших особливих властивостей у даного елемента управління немає.

```
<asp:Label ID="Label1" runat="server" Text="Привіт"></asp:Label>
<asp:Label ID="Label2" runat="server">Це маленька міточка</asp:Label>
```

Зазвичай, даний контрол використовують, якщо необхідно програмно управляти підписом на сторінці. В інших випадках, використовуються звичайні HTML теги.



За текстове поле відповідає клас **TextBox**, який має наступні корисні властивості та події:

- **Text** – вміст текстового поля.
- **TextMode** – режим відображення тексту, який має тип перелічувальної константи `TextBoxMode`:
 - `SingleLine` – однорядкове текстове поле (значення по замовчуванню);
 - `MultiLine` – багаторядкове текстове поле, яке складається з встановленої кількості рядків (властивість `Rows`) та стовпчиків (властивість `Columns`);
 - `Password` – текстове поле з можливістю приховання вмісту за допомогою символів. Як правило, використовується для введення пароля.
- **MaxLength** – максимальна кількість символів в текстовому полі.
- **AutoPostBack** – при встановленні значення даної властивості в `true`, дані текстового поля будуть автоматично відправлятися на сервер.
- **ReadOnly** – дозволяє встановити текстове поле тільки для читання.
- **Wrap** – встановлює перенос тексту в багаторядковому текстовому полі.
- ⚡ **TextChanged** – подія при зміні вмісту текстового поля.

Наприклад:

```
<asp:TextBox ID="tb1" runat="server" />
<asp:TextBox ID="tb2" runat="server" Text="" TextMode="Password" MaxLength="5" />
```

Результат:



Кнопки в ASP.NET представлені трьома класами:

1. **Button** – звичайна натискаема кнопка;
2. **LinkButton** – кнопка, яка має вигляд гіперпосилання;
3. **ImageButton** – кнопка з підтримкою зображення.

Різниця між цими кнопками лише у зовнішньому вигляді. Всі вони мають наступні спільні властивості та події:

- **Text** - визначає підпис на кнопці;
- **PostBackUrl** – містить URL сторінки, на яку потрібно передати дані з поточної сторінки, при натисненні кнопки. По замовчуванню приймаючою сторінкою є поточна сторінка;
- ⚡ **Click** – подія при натисненні на кнопку.

Лише клас `ImageButton` має додаткову властивість **ImageUrl**, в якій вказується шлях до зображення на кнопці.

Наприклад:

Default.aspx

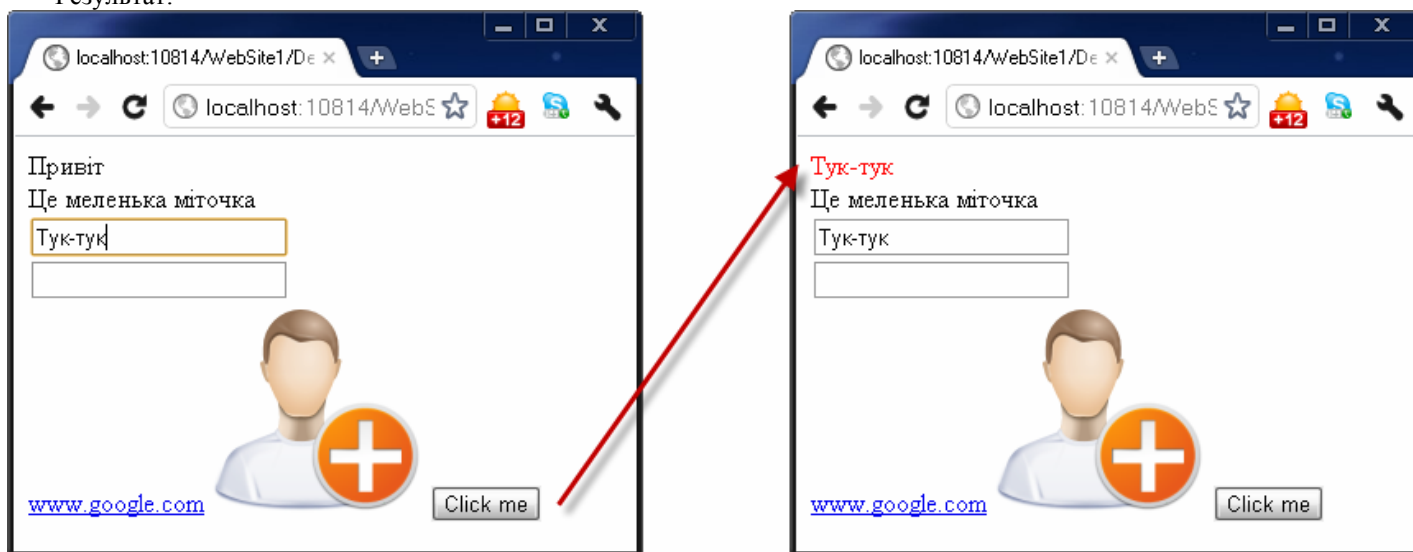
```
<asp:LinkButton ID="lb" runat="server" Text="www.google.com" />
<asp:ImageButton ID="imgb" runat="server" ImageUrl="contact-new.png" />
<asp:Button ID="Button1" runat="server" Text = "Click me" OnClick="Button1_Click" />
```

Default.aspx.cs

```
protected void Button1_Click(object sender, EventArgs e)
{
    Label1.ForeColor = System.Drawing.Color.Red;
    Label1.Text = tb1.Text;
}
```



Результат:



8.2.2. Елементи управління HyperLink та Image

Елемент управління **HyperLink** дозволяє створювати гіперпосилання, які на стороні клієнта транслюються в теги <a>, та має наступні **властивості**:

- ✓ **Text** – підпис на гіперпосиланні;
- ✓ **NavigateUrl** – задає адресу, на яку ссилається гіперпосилання;
- ✓ **ImageUrl** – шлях до зображення, якщо гіперпосилання являє собою зображення;
- ✓ **Target** – вказує, в якому вікні буде відображатись вміст веб-сторінки, яка відкривається при активації гіперпосилання:
 - **_blank** - ресурс завантажується в нове вікно;
 - **_parent** - ресурс завантажується в батьківському вікні;
 - **_top** - ресурс завантажується в topmost вікно;
 - **_self** - ресурс завантажується в поточному вікні (по замовчуванню);
 - **_search** - завантажується в панель пошуку браузера. Тільки в Internet Explorer 5 і вище.

Клас **HyperLink** доцільно використовувати тільки для створення гіперпосилання, текст або адреса якого буде динамічно змінюватись в ході роботи користувача.

Зображення представлені елементом управління **Image**:

- ✓ **ImageUrl** – шлях до зображення;
- ✓ **ImageAlign** – вирівнювання зображення на сторінці відносно інших елементів;
- ✓ **AlternateText** – альтернативний текст;
- ✓ **DescriptionUrl** - дозволяє вказати шлях до детального опису зображення.

Клас **Image** дуже добре використовувати при динамічному створенні великої кількості зображень і їх завантаженні на веб-сторінку чи веб-сайт.

Default.aspx

```
<asp:HyperLink runat="server" Text="Пошукова система" NavigateUrl="http://www.google.com" />
<asp:HyperLink runat="server" Text="Зареєструватись" NavigateUrl="~/Register.aspx" /> <br />
<asp:Image runat="server" ImageUrl="img1.jpg" Width="300px" />
```

Знак ~ вказує на кореневий каталог поточного сайту.

8.2.3. Елементи управління CheckBox та RadioButton

Для отримання вибору користувача використовуються флажки та перемикачі. Як вам вже відомо з попередніх курсів, **флажки (check boxes)** являють собою маленькі квадратні віконечка з текстом, який розміщується зазвичай справа від мітки. Як правило, даний елемент управління використовується для вказування різноманітних опцій програми. Флажки діють як вимикачі: один клік ставить мітку (галочку) в елементі управління, наступний – знімає.

Для роботи з флажками в ASP. NET слід скористатись елементом управління **CheckBox**, який має наступні корисні **властивості та події**:

- ✓ **Text** – підпис біля флажка;
- ✓ **TextAlign** – вирівнювання тексту підпису біля флажка;
- ✓ **Checked** – стан флажка (ввімкнений чи ні);
- ✓ **AutoPostBack** – автоматично відправляти дані про стан флажка на сервер (true) чи ні (false);



✓ ⚡ **CheckedChanged** – подія при зміні стану флажка.

Приведемо невеличкий приклад, в якому на сторінці буде відображатись невеличкий каталог розсилки з списком новини, на які можна підписатись:

Default.aspx

```
<%@ 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/xhtml1-transitional.dtd">

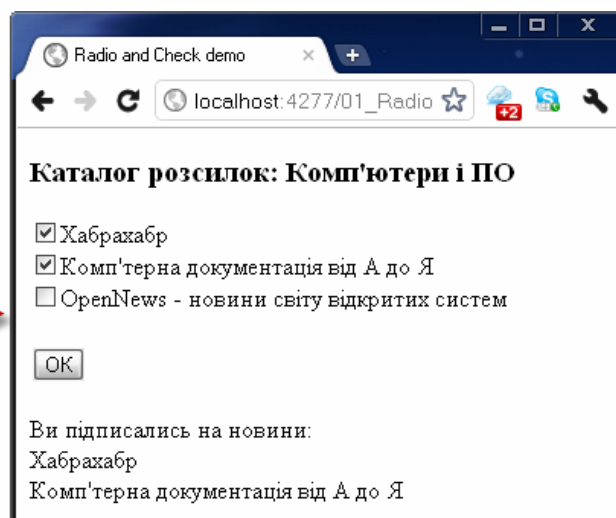
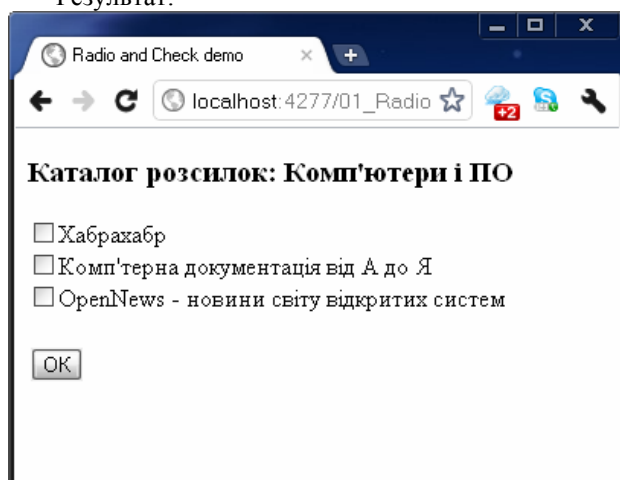
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server"><title>Radio and Check demo</title></head>
<body>
    <form id="form1" runat="server">
        <h3>Каталог розсилок: Комп'ютери і ПО</h3>
        <asp:CheckBox ID="cbNews1" runat="server" Text="Хабрахабр"
            OnCheckedChanged="cbNews_CheckedChanged" /><br />
        <asp:CheckBox ID="cbNews2" runat="server"
            Text="Комп'терна документація від А до Я"
            OnCheckedChanged="cbNews_CheckedChanged" /><br />
        <asp:CheckBox ID="cbNews3" runat="server"
            Text="OpenNews- новини світу відкритих систем"
            OnCheckedChanged="cbNews_CheckedChanged" />
        <br /><br />
        <asp:Button ID="Button1" runat="server" Text="OK" />
        <br /><br />
        <asp:Label ID="Label1" runat="server" Text="" />
    </form>
</body>
</html>
```

Default.aspx.cs

```
public partial class _Default : System.Web.UI.Page {
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Page.IsPostBack)
            Label1.Text = "Ви підписались на новини:<br/>";
    }

    protected void cbNews_CheckedChanged(object sender, EventArgs e)
    {
        CheckBox check = (CheckBox) sender;
        if (check.Checked)
            Label1.Text += check.Text + "<br/>";
    }
}
```

Результат:





Перемикачі (radio buttons) – це також кнопки, які схожі на флажки, але їх форма не квадратна, а кругла. Крапочка, яка знаходиться в середині перемикача, означає, що він “ввімкнутий”. Перемикачі використовують в основному групою для вибору одного з кількох взаємовиключаючих варіантів.

Перемикачі в ASP.NET представлені елементом управління **RadioButton**, який має основну властивість **GroupName** – ім'я групи перемикачів, а також властивість **Checked**, яка дозволяє отримати інформацію про активний перемикач. Всі інші властивості та події аналогічні класу **CheckBox**.

В якості прикладу, додамо до нашої програми, набір перемикачів:

Default.aspx

```
<%@ 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/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server"><title>Radio and Check demo</title></head>
<body>
    <form id="form1" runat="server">
        <h3>Каталог розсилок: Комп'ютери і ПО</h3>
        <asp:CheckBox ID="cbNews1" runat="server" Text="Хабрахабр"
            OnCheckedChanged="cbNews_CheckedChanged" /><br />
        <asp:CheckBox ID="cbNews2" runat="server"
            Text="Комп'терна документація від А до Я"
            OnCheckedChanged="cbNews_CheckedChanged"/><br />
        <asp:CheckBox ID="cbNews3" runat="server"
            Text="OpenNews - новини світу відкритих систем"
            OnCheckedChanged="cbNews_CheckedChanged"/>

        <h3>Підписатись на новини?: </h3>
        <asp:RadioButton ID="RadioButton1" GroupName="rgAge" runat="server" Text="Так"
            Checked="true" />
        <asp:RadioButton ID="RadioButton2" GroupName="rgAge" runat="server" Text="Ні" />

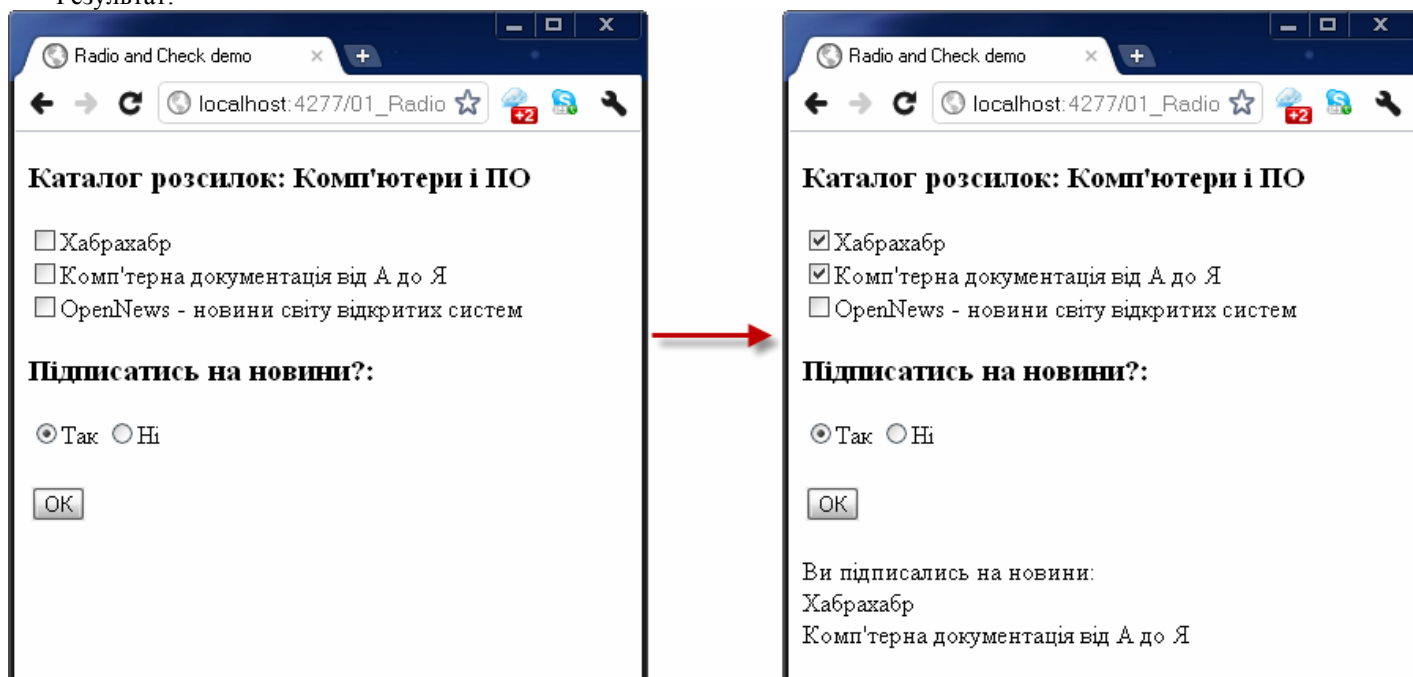
        <br /><br />
        <asp:Button ID="Button1" runat="server" Text="OK" />
        <br /><br />
        <asp:Label ID="Label1" runat="server" Text=""></asp:Label>
    </form>
</body>
</html>
```

Default.aspx.cs

```
public partial class _Default : System.Web.UI.Page{
    bool flag = true;
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Page.IsPostBack)
        {
            if (RadioButton1.Checked){
                flag = true;
                Label1.Text = "Ви підписались на новини:<br/>";
            }else if (RadioButton2.Checked){
                flag = false;
                Label1.Text = "Ви відмовились від підписки.";
            }
        }
    }
    protected void cbNews_CheckedChanged(object sender, EventArgs e)
    {
        CheckBox check = (CheckBox)sender;
        if (check.Checked & flag == true)
            Label1.Text += check.Text + "<br/>";
    }
}
```



Результат:



8.2.4. Списки: **ListBox**, **CheckBoxList**, **RadioButtonList**, **DropDownList**

Списки дозволяють зберігати набір текстових рядків, які виводяться у невеликій прямокутній області. Ви можете додавати нові елементи списку, видаляти існуючі або ж міняти їх місцями. В ASP.NET підтримується **5 видів списків**:

1. **ListBox** – звичайний список;
2. **DropDownList** – випадаючий список;
3. **CheckBoxList** – група флажків;
4. **RadioButtonList** – група перемикачів;
5. **BulletedList** – нумерований або маркерований список.

Кожен з списків включає в себе набір тегів, які являються пунктами списку. Для їх оголошення використовується спеціальний клас **Listltem**. Крім того, всі вони походять від класу **ListControl**, який надає їм **базовий набір властивостей**, зокрема таких як:

- **AutoPostBack** – для автоматичної відправки даних списків на сервер;
- **Items** – колекція елементів типу **Listltem**, яка має властивості:
 - **Text** – текст пункту списку;
 - **Value** – значення для пункту списку, наприклад, ключ;
 - **Selected** – вказує на активний пункт;
 - **Enabled** – дозволяє зробити пункт списку неактивним (false);
 - **Attributes** – колекція атрибутів для об'єкта **Listltem**.
- **SelectedIndex** – повертає найменший індекс серед обраних елементів списку;
- **SelectedItem** – повертає обраний елемент з найменшим індексом серед обраних;
- **SelectedValue** – повертає текст обраного елемента списку або пункт списку з вказаним значенням;
- **Text** – встановлює або повертає властивість **SelectedValue** списку;
- ⚡ **SelectedIndexChanged** – подія, яка генерується при зміні активного елемента списку;
- ⚡ **TextChanged** – подія, яка генерується при зміні значення властивостей **Text** та **SelectedValue**.

Відмінність між ними полягає лише в тому, що перші 4 види списків дозволяють користувачу здійснювати вибір значень (одного або кількох), а останній вид списку (**BulletedList**) служить лише для виведення статичної інформації на екран. Тому спочатку розглянемо списки з можливістю вибору значень, а потім перейдемо до **BulletedList**.

Отже, елементи управління **ListBox**, **DropDownList**, **CheckBoxList** та **RadioButtonList** мають нижчеописані унікальні **властивості та події**:

- ✓ **SelectionMode** – дозволяє встановити режим вибору для **ListBox**:
 - **Single** – одиничний вибір;
 - **Multiple** – множинний вибір.
- ✓ для списків типу **CheckBoxList** та **RadioButtonList**:
 - **RepeatLayout** – вказує на режим відображення списку:



- Table – у вигляді таблиці, яка складається з визначеної кількості стовпчиків (RepeatColumns) з визначеними інтервалами та вирівнюванням (CellPadding, CellSpacing, TextAlign);
- Flow – у вигляді рядка, в елементі ``;
- UnorderedList – у вигляді маркерованого списку, в елементі ``;
- OrderedList – у вигляді нумерованого списку, в елементі ``.

- **RepeatDirection** – показувати список горизонтально (Horizontal) чи вертикально (Vertical);
- **RepeatedItemCount** – кількість елементів в списку.

Наприклад:

Default.aspx

```
<%@ 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/xhtml1-transitional.dtd">

<script language="javascript" type="text/javascript">
    function Loading()
    {
        //встановлюємо колір фону сторінки в залежності від значення
        //активного пункта випадаючого списку
        element = form1.elements["ddList"];
        document.body.bgColor = "#" + element.options(element.selectedIndex).value;
    }
</script>

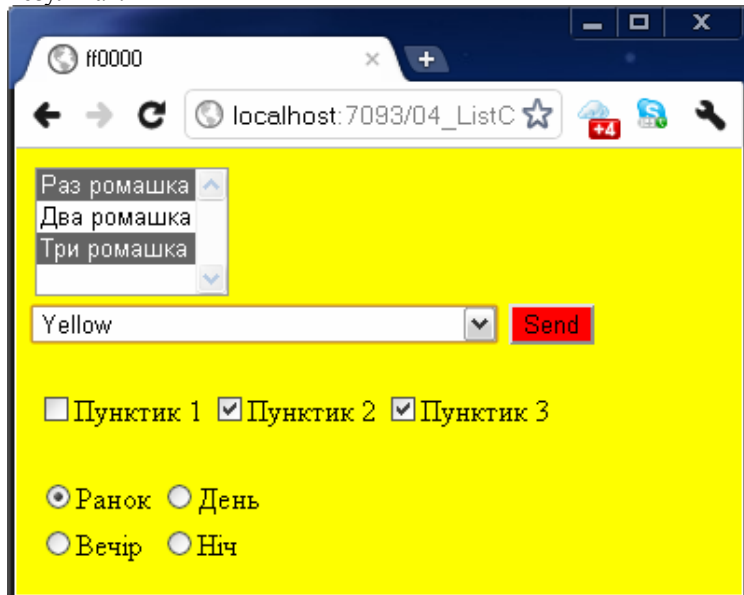
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>List control</title>
</head>
<body onclick="Loading()" >
    <form id="form1" runat="server">
        <div>
            <asp:ListBox ID="lb" runat="server" SelectionMode="Multiple">
                <asp:ListItem>Раз ромашка</asp:ListItem>
                <asp:ListItem>Два ромашка</asp:ListItem>
                <asp:ListItem>Три ромашка</asp:ListItem>
            </asp:ListBox>
            <br />
            <asp:DropDownList ID="ddList" runat="server" Width="250">
                <asp:ListItem Text="Red" Value="ff0000" Selected="True" />
                <asp:ListItem Text="Green" Value="00ff00" />
                <asp:ListItem Text="Blue" Value="0000ff" />
                <asp:ListItem Text="Yellow" Value="ffff00" />
            </asp:DropDownList>
            <asp:Button ID="b" runat="server" Text="Send" />
            <br /><br />
            <asp:CheckBoxList ID="cbl" runat="server" RepeatDirection="Horizontal">
                <asp:ListItem>Пунктик 1</asp:ListItem>
                <asp:ListItem Selected="True">Пунктик 2</asp:ListItem>
                <asp:ListItem>Пунктик 3</asp:ListItem>
            </asp:CheckBoxList>
            <br />
            <asp:RadioButtonList ID="CheckBoxList1" runat="server" RepeatColumns="2">
                <asp:ListItem>Ранок</asp:ListItem><asp:ListItem>Вечір</asp:ListItem>
                <asp:ListItem>День</asp:ListItem> <asp:ListItem>Ніч</asp:ListItem>
            </asp:RadioButtonList>
        </div>
    </form>
</body>
</html>
```



Default.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    this.Title = ddList.SelectedValue; //встановлюємо заголовок сторінки
    //встановлюємо колір кнопки в залежності від початкового значення випадаючого списку
    this.b.BackColor =
        System.Drawing.Color.FromName(ddList.Items[ddList.SelectedIndex].Text);
}
```

Результат:



8.2.5. Нумерованный список BulletedList

BulletedList – це нумерований або маркерований список, елементами якого можуть бути звичайний текст, гіперпосилання та елементи LinkButton. Елемент управління BulletedList є заміною списків HTML, які створюються за допомогою тегів ul та ol. Дуже часто він використовується для відображення даних з бази даних.

Властивості елемента управління BulletedList:

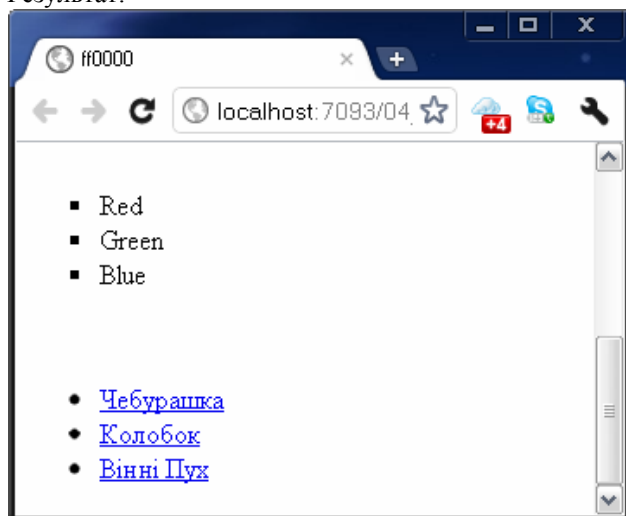
- **BulletStyle** – тип нумератора, який може приймати одне з наступних значень: Circle, Disk, Square, CustomImage, LowerAlpha, LowerRoman, Numbered, UpperAlpha, UpperRoman, NonSet.
- **DisplayMode** – задає режим відображення списку:
 - Text – в якості елементів списку виступає звичайний текст;
 - Hyperlink – в якості елементів списку використовуються гіперпосилання;
 - LinkButton – в якості елементів списку використовуються кнопки LinkButton.
- **FirstBulletNumber** – дозволяє задати номер, з якого починається нумерація в списку (для типів нумераторів LowerAlpha, LowerRoman, Numbered, UpperAlpha, UpperRoman).
- **BulletImageUrl** – якщо тип нумератора визначений як CustomImage, тоді даний атрибут визначає шлях до зображення.

Наприклад:

```
<asp:BulletedList ID="blist" runat="server" BulletStyle="Square">
    <asp:ListItem>Red</asp:ListItem>
    <asp:ListItem>Green</asp:ListItem>
    <asp:ListItem>Blue</asp:ListItem>
</asp:BulletedList>
<br />
<asp:BulletedList ID="bl" runat="server" BulletStyle="Disc" DisplayMode="HyperLink">
    <asp:ListItem Value="http://cheburator.com.ua">Чебурашка</asp:ListItem>
    <asp:ListItem Value="http://kolobok.net">Колобок</asp:ListItem>
    <asp:ListItem Value="http://vinni.ua">Вінні Пух</asp:ListItem>
</asp:BulletedList>
```



Результат:



Зверніть увагу на те, що в списку гіперпосилань адреси потрібно записувати у властивість Value елемента списка.

8.2.6. Таблиці

Для роботи з таблицями в ASP.NET призначений клас **Table**, якому відповідає аналогічний елемент управління. Даний елемент управління є альтернативою таблицям HTML. Використовується Table дуже рідко, оскільки для представлення даних бази даних частіше використовуються спеціалізовані елементи управління даними, наприклад, **GridView**. А для звичайного виведення табличних даних краще скористатись тегом HTML `<table>`, який являється більш оптимізованим. Фактично побудова таблиць за допомогою серверного елемента управління Table буде виправдана лише при побудові динамічно змінюваної таблиці.

З програмної точки зору, таблиця – це колекція рядків **Rows**, кожен з яких описується класом **TableRow**. В середині кожного рядка міститься колекція **Cells** (комірки) з об'єктами **TableCell**. Вміст таблиці програмно можна записати за допомогою властивості **Text** окремої комірки.

Рядки таблиці можуть бути також типу **TableHeaderRow** і **TableFooterRow**. Такі рядки завжди відображаються на мобільних пристроях з невеликим екраном, навіть якщо таблиця велика і для її перегляду потрібна прокрутка. Елементи таблиці можуть бути типу **TableHeaderCell** – спадкоємця **TableCell**. Текст в них відображається виділеним напівжирним шрифтом і центрований.

Крім вищеописаних, клас Table містить і інші **властивості**:

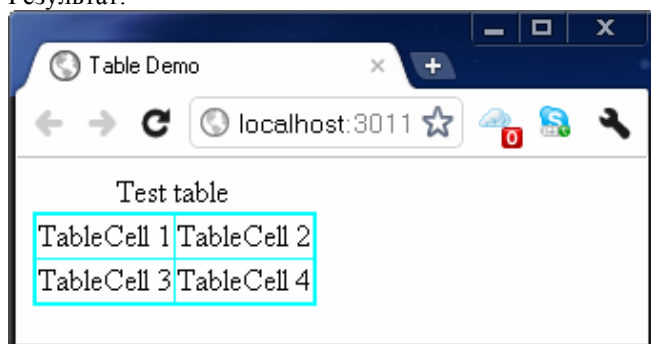
- **Caption** – заголовок таблиці, місцерозташування якого визначається властивістю **CaptionAlign**;
- **CellSpacing** та **CellPadding** – інтервал між рядками та стовпчиками таблиці;
- **BackColor** – шлях до фонових зображення таблиці;
- **GridLines** – стиль сітки, який визначає які проміжні лінії сітки таблиці необхідні:
 - Horizontal або Vertical – горизонтальні чи вертикальні;
 - Both – обидві: горизонтальні і вертикальні;
 - None – лінії сітки відсутні (по замовчуванню).

Приведемо невеличкий приклад побудови таблиці:

```
<form id="form1" runat="server">
<div>
  <asp:Table Caption="Test table" CaptionAlign="Top"
    BorderWidth="2" BorderStyle="Double" BorderColor="Aqua"
    GridLines="Both" runat="server">
    <asp:TableRow>
      <asp:TableCell>TableCell 1</asp:TableCell>
      <asp:TableCell>TableCell 2</asp:TableCell>
    </asp:TableRow>
    <asp:TableRow>
      <asp:TableCell>TableCell 3</asp:TableCell>
      <asp:TableCell>TableCell 4</asp:TableCell>
    </asp:TableRow>
  </asp:Table>
</div>
</form>
```




Результат:



8.3. Серверні елементи управління HTML. Простір імен System.Web.UI.HtmlControls

HTML елементи управління походять від класу **HtmlControls**, який знаходиться в просторі імен **System.Web.UI.HtmlControls**. Цей клас містить набір загальних властивостей та методів для елементів управління HTML:

- **Attributes** – посилання на колекцію атрибутів елемента управління;
- **Disabled** – дозволяє визначити доступність елемента управління (true – неактивний, недоступний; false - активний);
- **Style** – колекція стилів CSS, зв'язаних з елементом управління;
- **TagName** – повертає ім'я тега для елемента управління.

До елементів управління HTML відносять:

Клас елемента управління	Опис
HtmlAnchor	Заміна HTML елемента <a>
HtmlButton	Заміна HTML елемента <button>
HtmlForm	Заміна HTML елемента <form>
HtmlGenericControl	Заміна іншим HTML елементам, які не використовуються в якості серверних елементів управління HTML. Наприклад, <body>, <div>, тощо.
HtmlImage	Заміна HTML елемента <image>
HtmlInputButton	Заміна HTML елемента <input type="button">, <input type="submit"> та <input type="reset">
HtmlInputCheckBox	Заміна HTML елемента <input type="checkbox">
HtmlInputFile	Заміна HTML елемента <input type="file">
HtmlInputHidden	Заміна HTML елемента <input type="hidden">
HtmlInputImage	Заміна HTML елемента <input type="image">
HtmlInputRadioButton	Заміна HTML елемента <input type="radio">
HtmlInputText	Заміна HTML елемента <input type="text"> і <input type="password">
HtmlSelect	Заміна HTML елемента <select>
HtmlTable	Заміна HTML елемента <table>
HtmlTableCell	Заміна HTML елемента <td> і <th>
HtmlTableRow	Заміна HTML елемента <tr>
HtmlTextArea	Заміна HTML елемента <textarea>

Вони відображаються у вигляді елементів розмітки HTML і не залежать від типу браузера. Властивості таких контролів повністю відповідають атрибутам тегів HTML. Для розуміння різниці між звичайними тегами та елементами управління HTML, розглянемо маленький приклад:

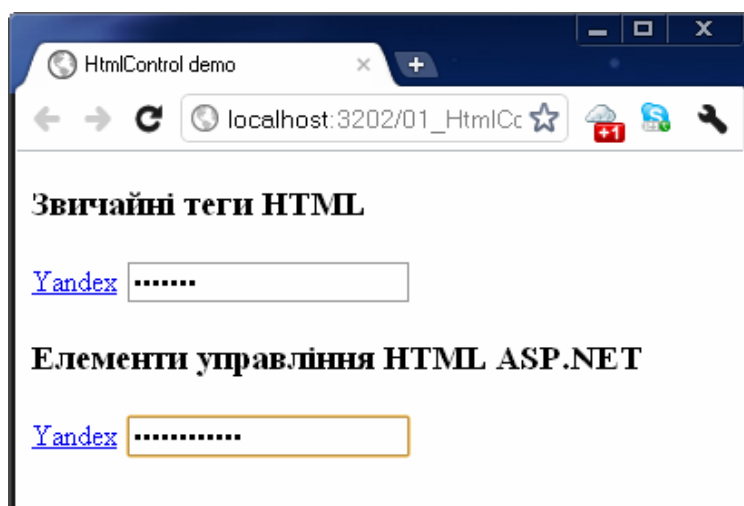
```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```



```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>HtmlControl demo</title>
</head>
<body>
  <h3>Звичайні теги HTML</h3>
  <a href="http://www.yandex.ru">Yandex</a>
  <input id="p1" type="password" value="тук-тук" />

  <h3>Елементи управління HTML ASP.NET</h3>
  <form id="form1" runat="server">
    <div>
      <a runat="server" href="http://www.yandex.ru">Yandex</a>
      <input id="p2" runat="server" type="password" value="" />
    </div>
  </form>
</body>
</html>
```

Результат:



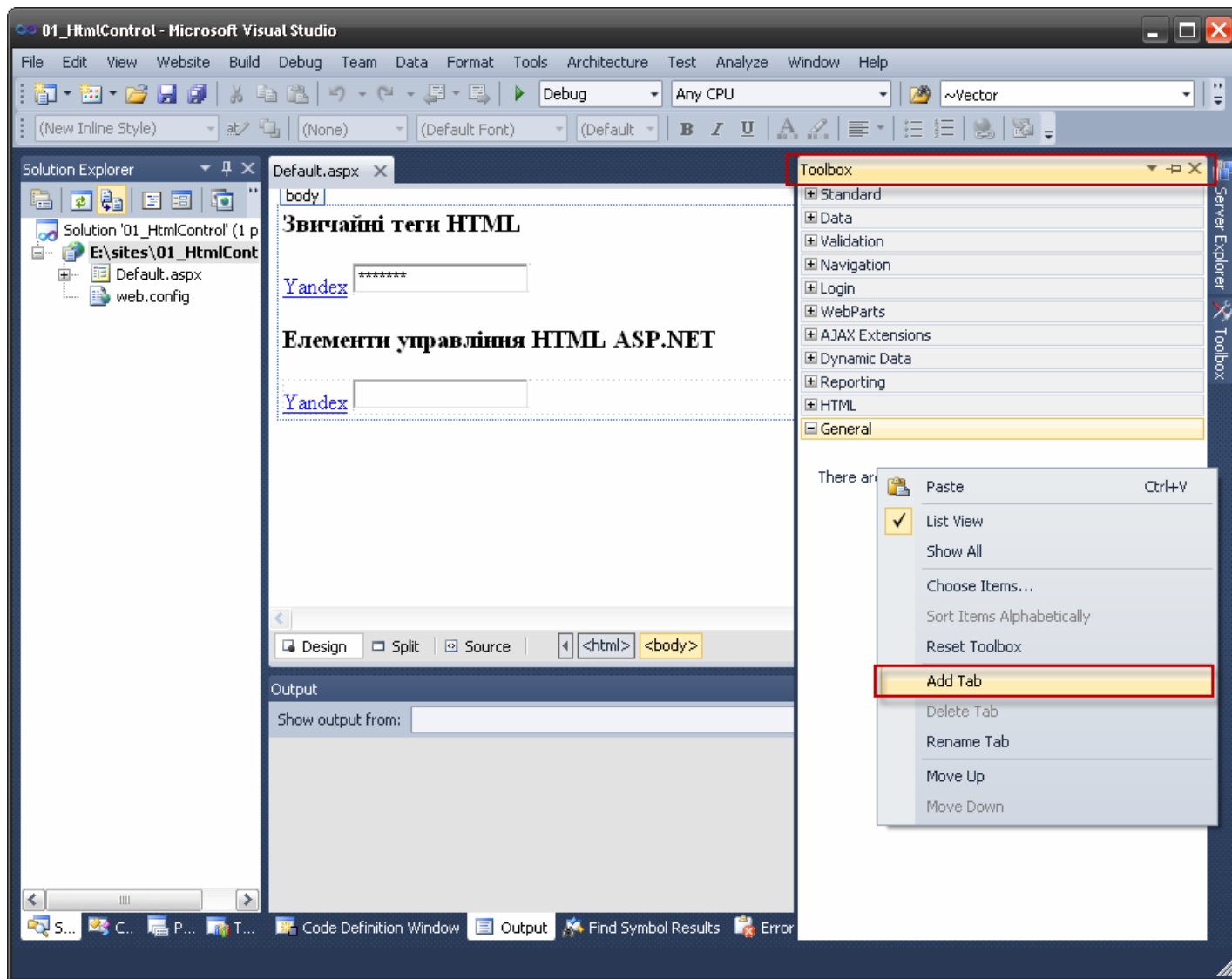
Як бачите, різниця полягає лише в існуванні атрибута **runat = "server"**. Але ця різниця суттєва, адже тепер – це серверний елемент управління HTML і ним можна маніпулювати на програмному рівні.

Коли ж використовувати серверні елементи управління HTML? Ці класи в основному використовуються, якщо необхідна оптимізація, але потрібно обробляти деякі теги HTML, а також якщо потрібно конвертувати старі сторінки ASP.

8.4. Короткий огляд бібліотек сторонніх розробників

Крім елементів управління, які нам надає Visual Studio 2010 та корпорація Microsoft, існують і інші бібліотеки елементів управління для ASP.NET від сторонніх фірм-розробників. Наприклад, до таких розробників можна віднести компанію Telerik Corporation (<http://www.telerik.com/>), яка являється на сьогоднішній день одним з лідерів розробок та продажу User Interface (UI) компонентів для ASP.NET і Windows Forms, та продукти компанії Developer Express (<https://www.devexpress.com/>), яка також займає вагомую частину ринку користувацьких елементів управління для ASP.NET.

Щоб користуватись такими елементами управління, достатньо зайти на сайт компанії-розробника, купити їх або скачати безкоштовні бібліотеки та проінсталювати. Після цього активувати вікно панелі інструментів Toolbox в Visual Studio та додати новий розділ (пункт контекстного меню **Add Tab**), де будуть розміщуватись елементи управління стороннього розробника.



Після цього, в новоствореному розділі, викликати пункт контекстного меню **Choose Items...** та обрати бібліотеку з необхідними елементами управління. Ось і все.

9. Web-форма як контейнер елементів управління. Динамічне створення елементів управління

Сторінка ASP.NET являється контейнером для елементів управління, які на ній розміщуються. Тому переглянути набір серверних елементів управління (з атрибутом `runat="server"`) будь-якої сторінки можна за допомогою колекції **Controls** класу `Page`. Причому, якщо заголовок `<head>` має встановлений атрибут `runat="server"`, то він також буде доданий до цієї колекції.

Сам процес візуалізації сторінки проходить ряд етапів. Спочатку візуалізується сама веб-форма, потім візуалізуються всі елементи управління, які на ній знаходяться. При цьому, якщо якийсь елемент управління містить інші дочірні контроли, то вони також окремо візуалізуються. Таким чином, по ходу візуалізації всіх елементів управління і самої веб-форми зокрема, ASP.NET генерує вихідний HTML код.

Підсумовуючи весь сьогоднішній урок, скажемо, що всі елементи управління можна створювати не лише за допомогою HTML розмітки, але і програмно, додаючи їх динамічно до колекції **Controls**. Код створення та додавання нових контролів можна виконувати в довільному обробнику подій, але пам'ятайте, що новий елемент управління буде додаватися в кінець сторінки. Крім того, він буде існувати лише до наступної відправки даних на сервер. Тому, якщо необхідно створювати постійні елементи управління, це потрібно робити в обробнику подій `Page.Load`.

Default.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default2.aspx.cs"
Inherits="Default2" %>
```



```
<!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">
    </form>
</body>
</html>
```

Default.aspx.cs

```
using System;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class Default2 : System.Web.UI.Page
{
    ListBox lb = new ListBox();
    Button btnSend = new Button();
    Label lText = new Label();

    protected void Page_Load(object sender, EventArgs e)
    {
        Page.Header.Title = "Динамич контроль"; //встановлюємо заголовок сторінки

        //створюємо список
        lb.BackColor = System.Drawing.Color.Gold;
        lb.Items.AddRange(new ListItem[] { new ListItem("Item1"),
                                            new ListItem("Item2"),
                                            new ListItem("Item3") });

        lb.SelectionMode = ListSelectionMode.Multiple;
        lb.SelectedIndex = 0;
        lb.Width = new Unit(120, UnitType.Pixel);
        lb.Height = new Unit(100, UnitType.Pixel);

        //створюємо кнопку
        btnSend.ID = "bSend";
        btnSend.Text = "Відправити";
        btnSend.ForeColor = System.Drawing.Color.FromArgb(250, 200, 5);
        btnSend.BackColor = System.Drawing.ColorTranslator.FromHtml("Coral");
        btnSend.Click += new EventHandler(this.btnSend_OnClick);

        //створюємо підпис
        lText.ID = "label1";
        lText.Text = "";

        //додаємо елемент управління на сторінку
        this.Controls.Add(new LiteralControl("<h2>Hello students</h2>"));
        //додаємо елементи управління на форму
        this.form1.Controls.Add(lb);
        this.form1.Controls.Add(btnSend);
        this.form1.Controls.Add(new LiteralControl("<br/>"));
        this.form1.Controls.Add(lText);
    }

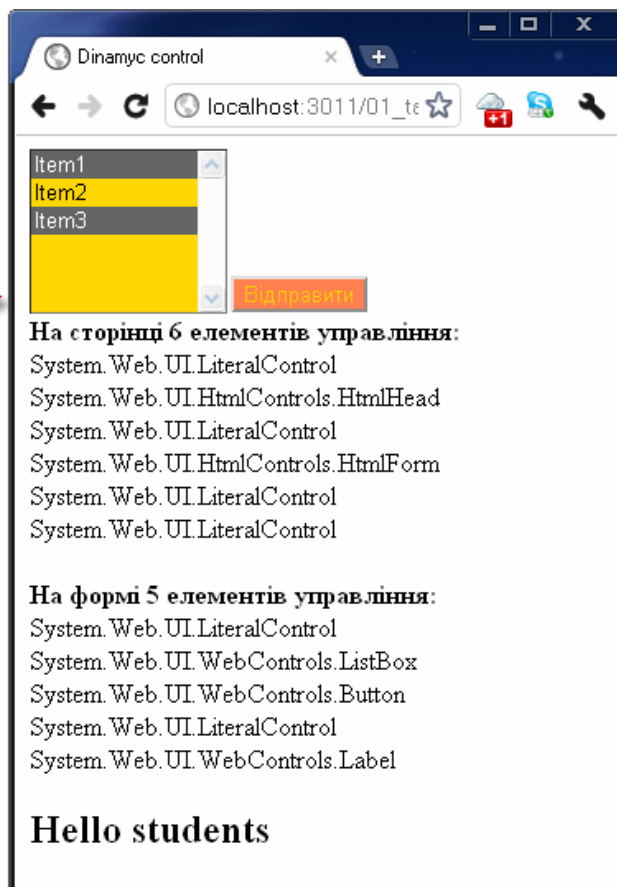
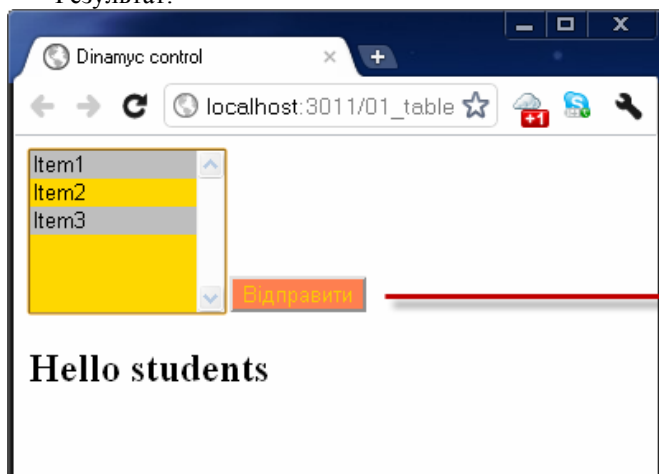
    public void btnSend_OnClick(object sender, EventArgs e)
    {
        lText.Text = "<b>На сторінці " + this.Controls.Count +
                    " елементів управління:</b><br/>";
    }
}
```



```
foreach (Control item in Page.Controls)
    lText.Text += item.GetType().ToString() + "<br/>";

lText.Text += "<br/><b>На формі " + this.form1.Controls.Count +
    " елементів управління:</b><br/>";
foreach (Control item in this.form1.Controls)
    lText.Text += item.GetType().ToString() + "<br/>";
}
```

Результат:



10. Приклади використання елементів управління

Розглянемо ще кілька прикладів використання елементів управління.

1. Веб-сторінка, яка дозволить забронювати номер в готелі певної країни. Для цього слід вказати свої реєстраційні дані, обрати країну та місто, готель, який знаходиться на обраній території, та іншу додаткову інформацію.

Hotel.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Hotel.aspx.cs" Inherits="Hotel" %>
<!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>Hotel Samples</title>
    <style type="text/css">
        body { background-color:#ffcc00; }
        .col1 { width: 150px; }
    </style>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="Label1" runat="server" Font-Size="20" ForeColor="DarkBlue">
```




```

        BorderWidth="2" BorderStyle="Double" Height="40" Width="300"
        Text="Бронювання готелів" /><br />
<table><tbody>
    <tr><td class="col1">Повне ім'я:</td>
        <td><asp:TextBox ID="tbFullName" runat="server" Width="220px" /></td>
    </tr>
    <tr><td class="col1">Місто: </td>
        <td><asp:DropDownList ID="ddlCities" runat="server"
            OnSelectedIndexChanged="ddlCities_OnSelectedIndexChanged"
            AutoPostBack="true"></asp:DropDownList></td>
    </tr>
    <tr><td class="col1">Готель: </td>
        <td><asp:DropDownList ID="ddlHotels" runat="server" Width="250px">
            </asp:DropDownList></td>
    </tr>
    <tr><td colspan="2">Дата заїзду і виїзду:</td></tr>
    <tr><td colspan="2"><asp:TextBox ID="tbDateOfArrival" runat="server" /> -
        <asp:TextBox ID="tbDateOfLeaving" runat="server" /></td>
    </tr>
    <tr><td colspan="2"><asp:CheckBox ID="checkDate" runat="server" />
        Точна дата поїздки поки не відома</td>
    </tr>
    <tr><td class="col1">Кількість номерів: </td>
        <td><asp:DropDownList ID="ddlNumbersOfRoom" runat="server" Width="50">
            </asp:DropDownList></td>
    </tr>
</tbody></table>
<asp:Button ID="bReservation" runat="server" Text="Забронювати"
    OnClick="bReservation_OnClick" />
</div>
</form>
</body>
</html>

```

Hotel.aspx.cs

```

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

public partial class Hotel : System.Web.UI.Page
{
    #region InitializeComponent
    String[] cities = new String[] { "Лондон, Англія", "Мілан, Італія",
        "Париж, Франція", "Київ, Україна" };

    String[][] hotels = new String[][]
    {
        new String[] { "L' Hotel de Sers", "Hotel Mayfair", "Citadines Prestige Opéra Vendôme" },
        new String[] { "Hotel Berna", "Enterprise Hotel", "Ramada Plaza Milano",
            "The Westin Palace" },
        new String[] { "The Westin Palace", "Hotel Le Littre" },
        new String[] { "Готель Мир", "Готель Салют", "Готель Либідь" }
    };

    override protected void OnInit(EventArgs e)
    {
        InitializeComponent(); //для ініціалізації елементів управління
        base.OnInit(e);
    }

    protected void Page_Load(object sender, EventArgs e) {}
    private void InitializeComponent()
    {
        //ініціалізація списку міст та країн
    }
}

```



```

for (int i = 0; i < cities.Length; i++)
    ddlCities.Items.Add(new ListItem(cities[i]));
ddlCities.SelectedIndex = 0;

//початкова ініціалізація списку готелів
for (int i = 0; i < hotels[0].Length; i++)
    ddlHotels.Items.Add(new ListItem(hotels[0][i]));

//ініціалізація списку номерів
for (int i = 1; i < 4; i++)
    ddlNumbersOfRoom.Items.Add(new ListItem(i.ToString()));
}
#endregion

protected void ddlCities_OnSelectedIndexChanged(object sender, EventArgs e)
{
    //int indx = ddlCities.SelectedIndex;           //можна так
    int indx = ((DropDownList)sender).SelectedIndex; //а можна і так
    ddlHotels.Items.Clear();           //очищаємо список

    //заповнюємо його новими значеннями, в залежності
    //від обраної країни та міста
    for (int i = 0; i < hotels[indx].Length; i++)
        ddlHotels.Items.Add(new ListItem(hotels[indx][i]));
}

protected void bReservation_OnClick(object sender, EventArgs e)
{
    //виводимо результат на веб-сторінку
    Label lRes = new Label();
    lRes.ForeColor = System.Drawing.Color.DarkBlue;
    lRes.Font.Size = 14;
    lRes.BorderWidth = 2;
    lRes.BorderStyle = BorderStyle.Double;

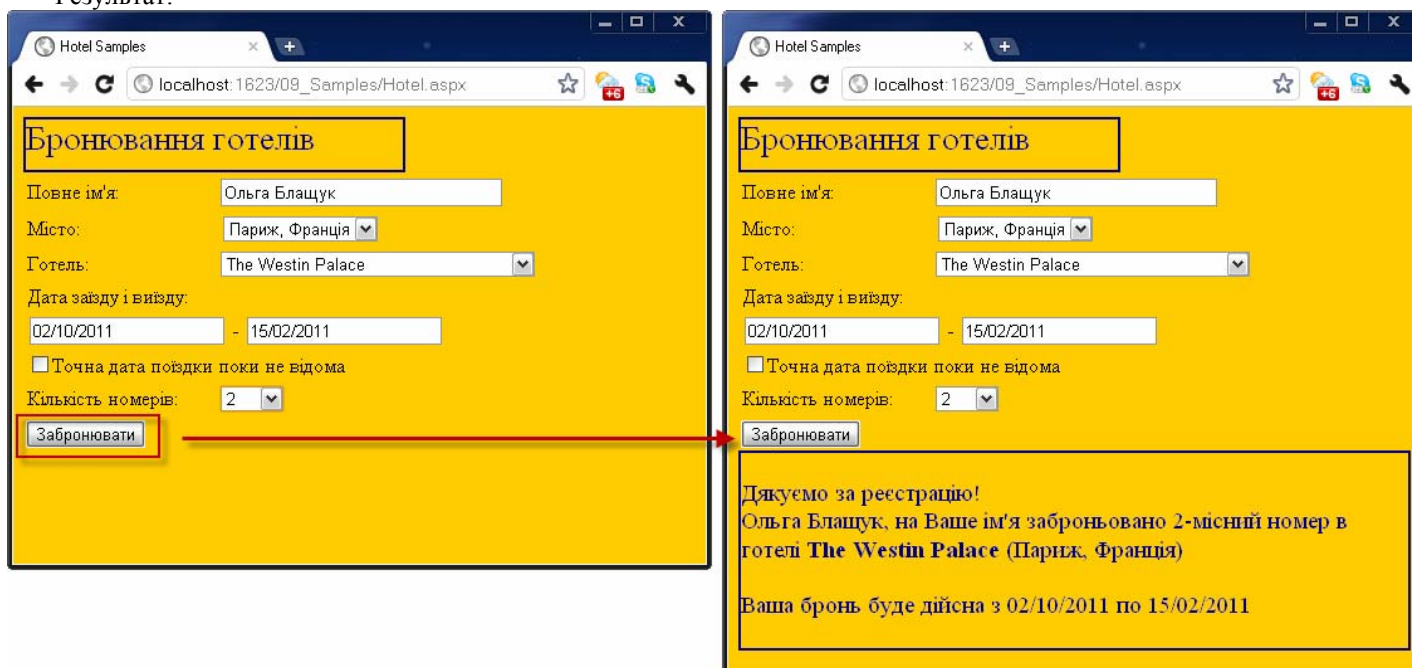
    lRes.Text = "<br/>Дякуємо за реєстрацію!<br/>";
    lRes.Text += tbFullName.Text + ", на Ваше ім'я заброньовано " +
        ddlNumbersOfRoom.SelectedValue + "-місний номер в готелі <b>" +
        ddlHotels.SelectedValue + "</b> (" +
        ddlCities.SelectedItem.Text + ") <br/><br/>";
    if(!checkDate.Checked)
        lRes.Text += "Ваша бронь буде дійсна з " + tbDateOfArrival.Text + " по " +
            tbDateOfLeaving.Text + "<br/><br/>";

    form1.Controls.Add(lRes);
}
}

```



Результат:



Бронювання готелів

Повне ім'я:

Місто:

Готель:

Дата заїзду і виїзду: -

☐ Точна дата поїздки поки не відома

Кількість номерів:

Дякуємо за реєстрацію!
Ольга Блашук, на Ваше ім'я заброньовано 2-місний номер в готелі The Westin Palace (Париж, Франція)
Ваша бронь буде дійсна з 02/10/2011 по 15/02/2011

2. Розглянемо приклад на динамічне створення маркерovanого списку типу BulletedList, вигляд якого буде визначатись користувачем:

DynBulletList.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="DynTable.aspx.cs" Inherits="DynTable"%>
<!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>Dynamic BulletList</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        Оберіть тип нумератора :<br/>
        <asp:ListBox ID="lbStyle" runat="server" Width="150px"
            AutoPostBack="true"
            OnSelectedIndexChanged="lbStyle_OnSelectedIndexChanged"></asp:ListBox><br />
        Оберіть режим відображення:<br/>
        <asp:RadioButtonList ID="rblDisplayMode" runat="server">
            <asp:ListItem Value="Text">Текст</asp:ListItem>
            <asp:ListItem Value="HyperLink">Гіперпосилання</asp:ListItem>
            <asp:ListItem Value="LinkButton">Кнопки</asp:ListItem>
        </asp:RadioButtonList><br />

        Початок відліку списку : <asp:TextBox ID="tbFirstNumber" runat="server" Width="125px">
            </asp:TextBox><br />

        Оберіть зображення :
        <asp:DropDownList ID="ddlImageUrl" runat="server" Width="150px"
            AutoPostBack="true"
            OnSelectedIndexChanged="ddlImageUrl_OnSelectedIndexChanged">
        </asp:DropDownList><br />

        <asp:Image ID="imgBullet" runat="server" Width="50" Height="50" /><br/><br/>
        <asp:Button ID="bSend" runat="server" Text="Згенерувати" OnClick="bSend_OnClick" />
    </div>
    </form>
</body>
</html>
```



DynBulletList.aspx.cs

```

using System;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class DynTable : System.Web.UI.Page
{
    String[] styles = new String[] { "Circle", "Disc", "Square", "CustomImage",
        "LowerAlpha", "LowerRoman", "UpperAlpha", "UpperRoman",
        "Numbered", "NonSet" };

    protected void Page_Load(object sender, EventArgs e){}

    override protected void OnInit(EventArgs e)
    {
        InitializeComponent(); //для ініціалізації елементів управління
        base.OnInit(e);
    }

    private void InitializeComponent()
    {
        //ініціалізація списку стилів для нумератора
        for (int i = 0; i < styles.Length; i++)
            lbStyle.Items.Add(new ListItem(styles[i]));

        //початкова ініціалізація списку зображень
        for (int i = 1; i <= 4; i++)
            ddlImageUrl.Items.Add(new ListItem(i.ToString(), "images/"+i+".png"));

        //встановлюємо початкові значення
        imgBullet.ImageUrl = "images/1.png";
        rblDisplayMode.SelectedIndex = 0;
        lbStyle.SelectedIndex = 0;
    }

    protected void lbStyle_OnSelectedIndexChanged(object sender, EventArgs e)
    {
        String value = ((ListBox)sender).SelectedValue;

        //активація допоміжних полів, в залежності від обраного
        //типу нумератора для BulletedList
        if (value == "NonSet" || value == "Circle" || value == "Disc" || value == "Square")
        {
            tbFirstNumber.Enabled = false;
            ddlImageUrl.Enabled = false;
        }else if (value == "CustomImage"){
            tbFirstNumber.Enabled = false;
            ddlImageUrl.Enabled = true;
        }else{
            tbFirstNumber.Enabled = true;
            ddlImageUrl.Enabled = false;
        }
    }

    protected void ddlImageUrl_OnSelectedIndexChanged(object sender, EventArgs e)
    {
        //змінюємо відображаєме зображення в залежності від пункту списку
        imgBullet.ImageUrl = ((DropDownList)sender).SelectedValue;
    }

    protected void bSend_OnClick(object sender, EventArgs e)
    {
        //створюємо список на основі вказаних користувачем даних
        BulletedList bullet = new BulletedList();
        bullet.BulletStyle = (BulletStyle)Enum.Parse(typeof(BulletStyle),
            lbStyle.SelectedValue);
    }
}

```



```

bullet.DisplayMode = (BulletedListDisplayMode)Enum.Parse(
                                                                    typeof(BulletedListDisplayMode),
                                                                    rblDisplayMode.SelectedValue);

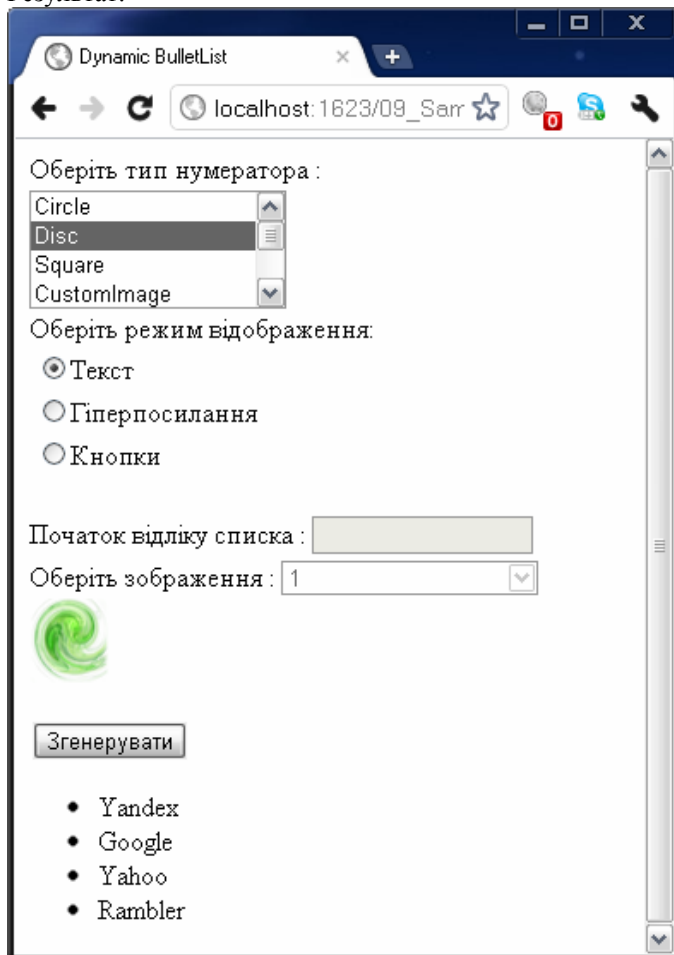
if (ddlImageUrl.Enabled)
    bullet.BulletImageUrl = ddlImageUrl.SelectedValue;
if (tbFirstNumber.Enabled)
    bullet.FirstBulletNumber = System.Convert.ToInt32(tbFirstNumber.Text);

//заповнюємо список значеннями
String[] sItems = new String[] { "Yandex", "Google", "Yahoo", "Rambler" };
for (int i = 0; i < sItems.Length; i++)
    bullet.Items.Add(new ListItem(sItems[i]));

//додаємо новостворений список на сторінку
form1.Controls.Add(bullet);
    }
}

```

Результат:



Dynamic BulletList

localhost:1623/09_Sarr

Оберіть тип нумератора:


Circle
Disc
Square
CustomImage

Оберіть режим відображення:

☒ Текст
☐ Гіперпосилання
☐ Кнопки

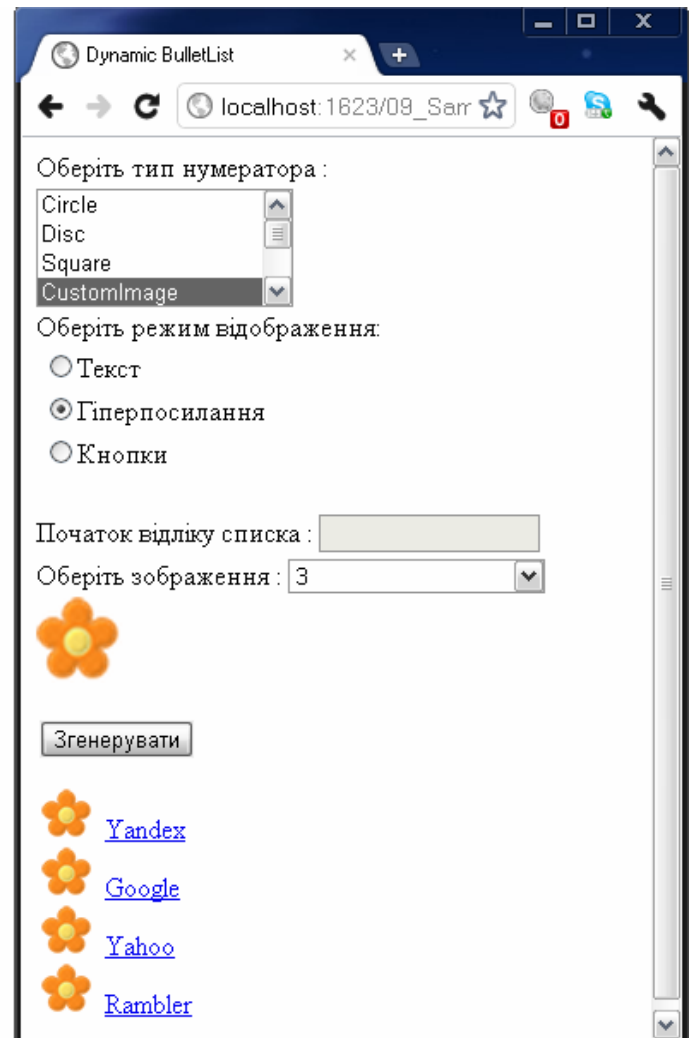
Початок відліку списка:

Оберіть зображення: 1



Згенерувати

- Yandex
- Google
- Yahoo
- Rambler



Dynamic BulletList

localhost:1623/09_Sarr

Оберіть тип нумератора:


Circle
Disc
Square
CustomImage

Оберіть режим відображення:





☐ Текст
☒ Гіперпосилання
☐ Кнопки

Початок відліку списка:

Оберіть зображення: 3



Згенерувати

-  [Yandex](#)
-  [Google](#)
-  [Yahoo](#)
-  [Rambler](#)



11. Домашнє завдання

1. Розробити просту форму реєстрації користувача на сайті, яка має наступний вигляд:

The screenshot shows a web form for user registration. It includes fields for: *Ім'я: (text), *Прізвище: (text), *Логін: (text), *Пароль: (text), *Підтвердження паролю: (text), *День народження: (three dropdown menus for day, month, and year), *Країна: (dropdown), *Повна адреса: (text), and *Е-mail: (text). There are radio buttons for 'Жінка' (selected) and 'Чоловік'. Below the form is a scrollable area containing the site's rules, followed by two checkboxes: 'Я погоджуюсь з усіма правилами' and 'Я, НЕ згоден з правилами'. A blue 'Зареєструватись' button is at the bottom. A note at the bottom left states: 'Примітка! Поля з * являються обов'язковими для заповнення.'

Після проведення реєстрації на сторінку в табличному вигляді виводиться вся введена інформація про користувача.

2. Написати простий варіант фотогалереї. Для цього:
- підготувати папку з зображеннями;
 - використати елемент управління Image, який посилається на будь-який файл із вищезгаданої директорії;
 - створити дві кнопки ImageButton для зміни зображень (Prev та Next). Якщо в будь-якому напрямку закінчуються файли (самий перший або останній), то кнопка змінює своє зображення, наприклад, з кольорового на чорно-біле або сіре;
 - над зображенням, в елементі Label відображається ім'я файлу.
3. Сьогодні ми розпочнемо роботу над курсовим проектом. Ціллю його буде розробити веб-сайт по автоматизації складського обліку магазину. Для цього необхідно:
- 3.1. Зробити опис історії (user story) з використання системи автоматизації складського обліку магазину будівельних матеріалів, або з іншої предметної області на розсуд викладача. Наприклад:
- спеціалізований магазин із продажу аудіо-, відеотехніки і деталей до них;
 - магазин комп'ютерної техніки;
 - агенство обміну і продажу нерухомості тощо.

Можна також скористатись вже розробленою на попередньому курсі по MS SQL Server 2008 базою даних.

- 3.2. Розробити інфологічну модель бази даних.
- 3.3. Розробити DDL-скрипти для створення структури бази даних на MS SQL Server.
- 3.4. Розробити Insert-скрипти для наповнення бази даних тестовими значеннями.