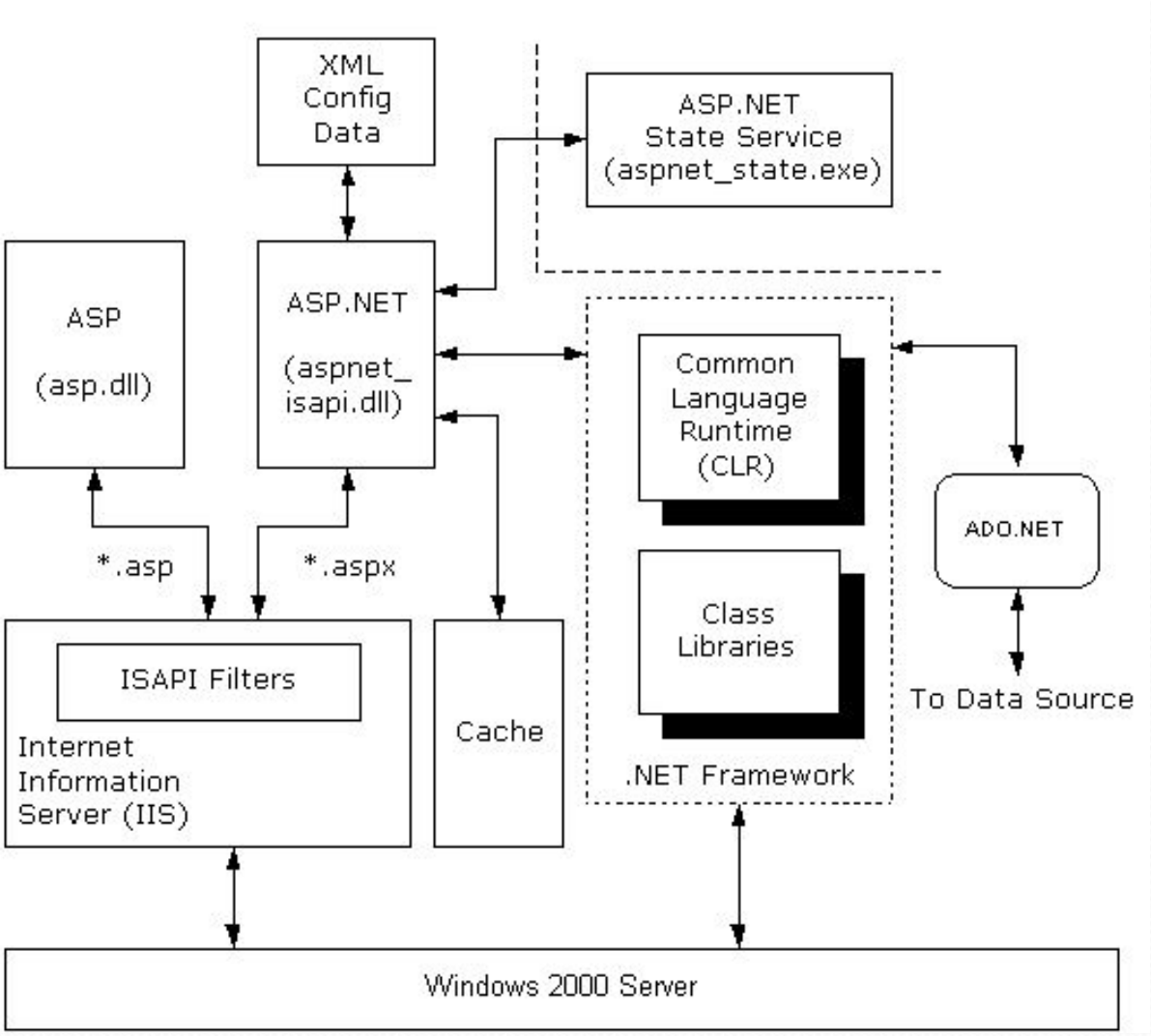
ЛЕКЦІЯ № 10

ASP.NET

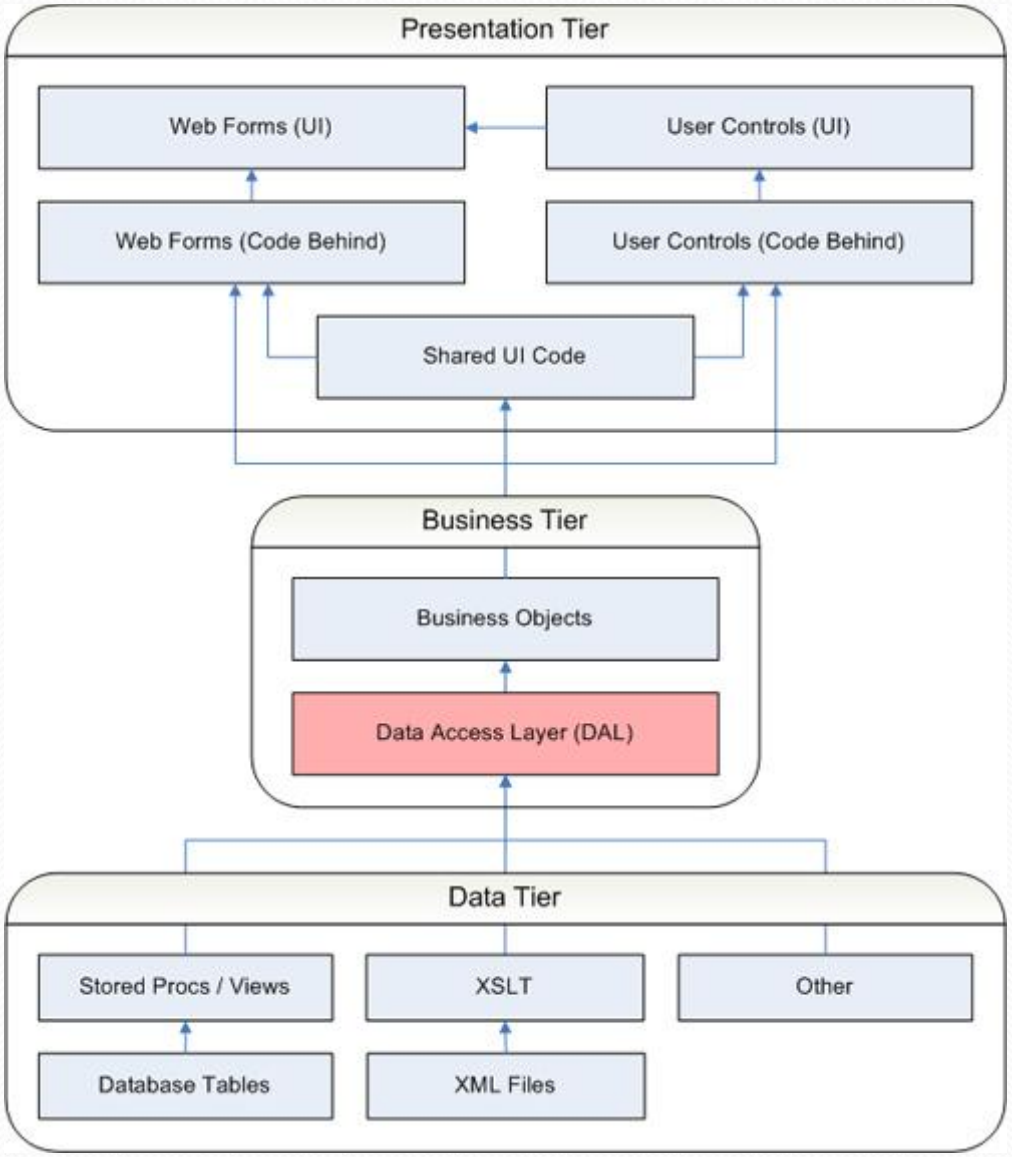
ASP.NET

ASP.NET – технологія створення веб-додатків та веб-сервісів. Вона є складовою частиною платформи Microsoft.NET та вдосконаленням більш старої технології Microsoft ASP.

Архітектура ASP.NET



Архітектура додатку ASP.NET



Життєвий цикл сторінки ASP.NET

Етап 1

Підготовка сторінки до виконання

Етап 2

Підготовка даних, зв’ язаних з поверненням форми

Етап 3

Завершальний етап виконання сторінки

Термінологія

Повернення форми – стандартний механізм, що дозволяє повернути параметри форми самій собі при перезавантаженні сторінки.

Міжсторінкове повернення форми – механізм, що дозволяє змінити стандартний цикл обробки сторінки і тим самим не дозволити повернути дані самій собі, а передати їх іншій сторінці.

Зворотній виклик сценарію – запуск на javascript функції, яка звертається до сервера за даними.

Етап 1. Підготовка сторінки до виконання

- ✔ Створення екземпляру класу сторінки для обслуговування поточного запиту;
- ✔ Формування дерева елементів сторінки;
- ✔ Підготовка та ініціалізація всіх дочірніх елементів управління та внутрішніх об'єктів сторінки (контекст HTTP, об’єкти Request та Response);
- ✔ З'ясування причини запуску сторінки (отримання звичайного запиту, повернення форми, міжсторінкове повернення форми, зворотній виклик сценарію).

Етап 2. Підготовка даних, зв’язаних з поверненням форми

- ✔ Перехід **серверної сторінки** у стан, у якому вона знаходилась, коли генерувала **клієнтську сторінку**;
- ✔ Коригування стану **серверної сторінки** в залежності від даних, введених на **клієнтській сторінці;**
- ✔ Обробка серверної події повернення форми.

Етап 3. Завершальний етап виконання сторінки

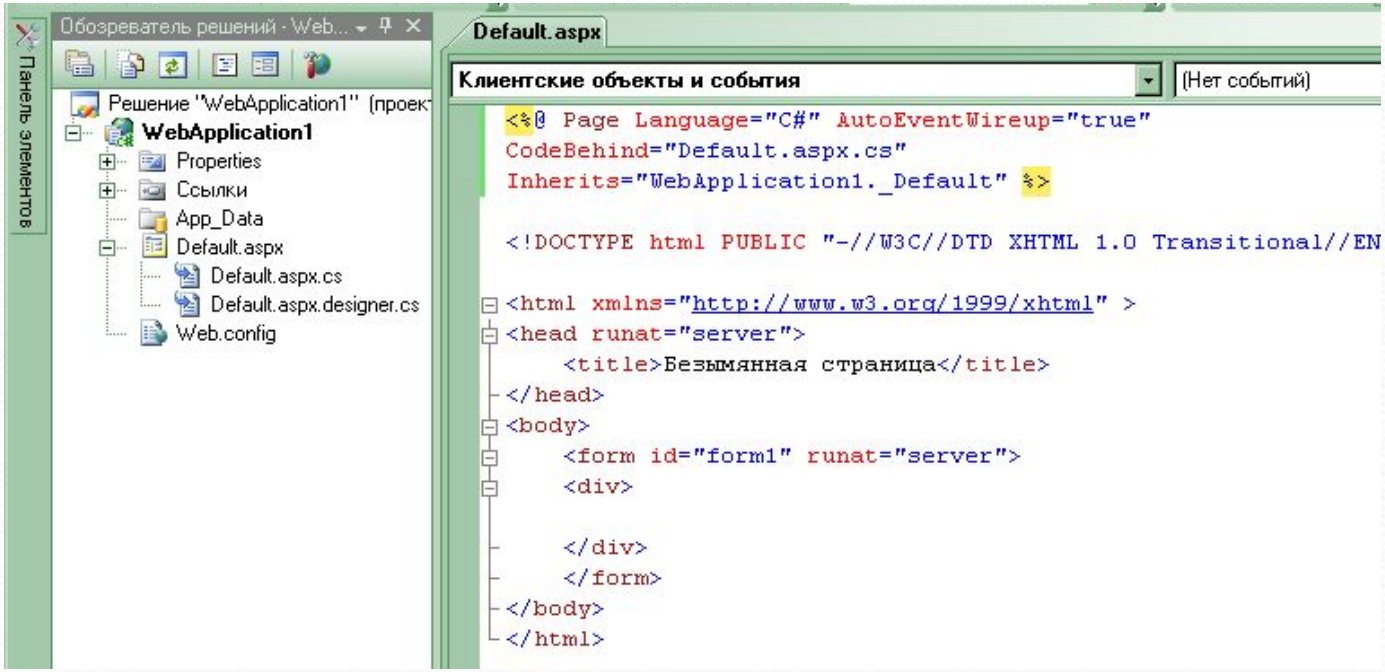
- ✔ Внесення необхідних змін у стан серверної сторінки;
- ✔ Збереження поточного стану серверної сторінки;
- ✔ Генерування розмітки для браузеру.

Форма – клас HtmlForm

Сторінка може мати лише одну видиму серверну форму. Не серверних (без атрибуту **runat="server"**) та невидимих (атрибут **visible="false"**) форм може бути скільки завгодно.

Сторінка виконує повернення форми самій собі. Серверна форма не може мати атрибут **action** та **name**. Не серверні форми можуть підтримувати технологію ASP, але не ASP.NET.

За замовчуванням атрибут **method="POST"**. При використанні **method="GET"** необхідно пам’ятати, що рядок запиту у заголовку HTTP обмежений до 2 кб.



Обробка даних форми

Для відправки даних форми на сервер необхідно всередину форми помістити будь-який елемент, який реалізує звернення до сервера, наприклад, кнопку **<asp:Button ID="Button1" runat="server" Text="Button" />.**

Дані форми за замовчуванням повертаються POST-методом на сервер тій же самій **.aspx**-сторінці, яка і створювала поточну форму. Вилучення та обробка отриманих даних зазвичай здійснюється у момент повторних завантажень **.aspx**-сторінки, тобто всередині обробника події **Page_Load**, коли сторінка перебуває у стані **IsPostback == "true" .**

Вилучення даних з елементів управління форми здійснюється простим зверненням до властивостей відповідних ідентифікаторів цих елементів, наприклад, **txt = "TextBox1.Text".**

Приклад

```
protected void Page_Load(object sender, EventArgs e)
{
   if (IsPostBack) // При повторній та наступних завантаженнях цієї сторінки...
   {
      Response.Write("<h3>Дякую, " + TextBox1.Text + "</h3>");
   }
}
```

Міжсторінкове повернення форми

Починаючи з версії ASP.NET 2.0 є присутнім вбудований механізм, що дозволяє змінити стандартний цикл обробки сторінки і тим самим перешкодити їй повернути дані самій собі. Це називається **міжсторінкове повернення форми**.

Повернення даних на іншу сторінку організовується додаванням властивості **PostBackUrl = "**адреса нової **.aspx**-сторінки**"** попередній кнопці. Для доступу на новій сторінці до всіх елементів управління попередньої сторінки, яка виконала постінг (методом **Transfer**), можна використати властивість **PreviousPage**. При цьому постінгу створюється додаткове приховане поле __PREVIOUSPAGE, яке і містить інформацію про стан елементів форми. Ця інформація доступна сторінці-одержувачу через властивість **PreviousPage**.

Наприклад, для отримання елемента **TextBox1** попередньої форми у змінну **txt** типу **TextBox** нової форми необхідно написати:

```
TextBox txt = (TextBox)PreviousPage.FindControl("TextBox1");
```

Далі можна отримати текст із елементу **txt** та вивести його на нову форму, наприклад, за допомогою мітки:

```
Label1.Text = txt.Text;
```

Зверніть увагу, що метод **Page.FindControl()** шукає елементи тільки кореневого контейнера сторінки, але не шукає елементи вкладених контейнерів!

Використання контейнерів з PreviousPage

Якщо Вам не відома контейнерна структура сторінки, що здійснила перенаправлення, але Ви знаєте, що там повинен бути потрібний Вам елемент, наприклад, **DropDownList** з **id="MyControl"**, то його пошук можна організувати перебором всіх контейнерів (на обмежену глибину N) та їх елементів:

Приклад

```
Page prevPage = (Page)PreviousPage;
foreach (Control child1 in prevPage.Controls){
   if (child1.ID == "MyControl"){
      Label1.Text = ((DropDownList)child1).SelectedValue;
      break;
   }
   else{
      foreach (Control child2 in child1.Controls){
         if (childN.ID == "MyControl"){
            Label1.Text = ((DropDownList)childN).SelectedValue;
            break;
         }
      }
   }
}
```

Перенаправлення користувача на іншу сторінку

Існує ще один механізм **міжсторінкового постінгу** – метод Transfer:

Server.Transfer("~/… нова сторінка .aspx")

Перенаправлення виконується на сервері. У браузері клієнта не відображається URL-адреса нової сторінки. Після цього методу ніякий код поточної сторінки не виконується, однак, все, що було до цього виконано, буде вбудоване у відповідь клієнту разом з HTML-потоком від нової сторінки. Компоненти та їхні значення на старій сторінці будуть доступні також й на новій сторінці через властивість **PreviousPage**. Обидві сторінки (відправляюча та отримуюча) повинні належати одному додатку з віртуальною адресою **"~/…"**.

Перенаправлення браузером клієнта:

Response.Redirect("нова сторінка .aspx")

Працює як звичайне гіперпосилання, активізоване на стороні клієнта.

При будь-якому перенаправленні можна передати **GET**-параметри, наприклад:

Response.Redirect("page2.aspx?name=value&…")

Отримання значень **value** параметрів **name** із заголовку запиту здійснюється таким чином:

Request.QueryString["name"]

Application state

Application state – колекція визначених користувачем змінних, які доступні з будь-якого місця програми ASP.NET. Вони встановлюються та ініціалізуються при події **Application_OnStart** на етапі завантаження першого екземпляру програми та залишаються доступними до тих пір, поки існує останній екземпляр додатку. Змінні Application state доступні через колекцію Applications, яка забезпечує «обгортку» для змінних application state. Змінні Application state ідентифікуються за їхніми іменами.

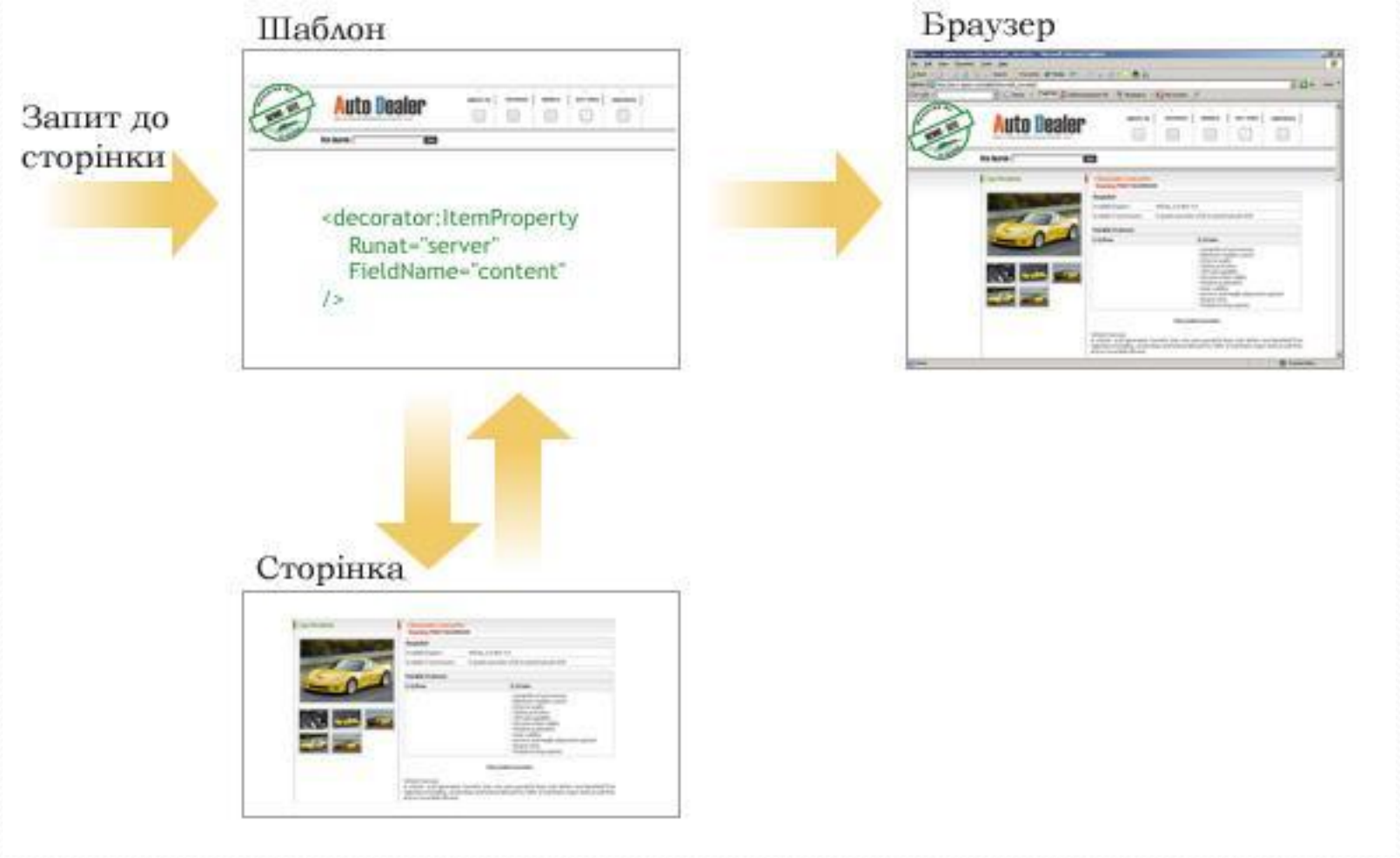
Session state

Session state – набір користувацьких змінних, які існують протягом сеансу користувача. Ці змінні унікальні для різних екземплярів сесії користувачів та доступні через колекцію Session. Такі змінні (Session variables) можна налаштувати таким чином, щоб вони автоматично знищувалися після спливання певного часу неактивності користувача, навіть якщо сесія ще не завершена. На стороні клієнта сесія користувача ідентифікується за допомогою cookie або ж ID сесії у її URL.

View state

View state посилається на рівень сторінки механізму управління станом (state management mechanism), який використовується HTML-сторінками для збереження стану компонентів веб-форми та її віджетів. При запиті на сервер поточний стан компонентів (controls) кодується та відправляється на сервер у прихованому полі __VIEWSTATE. Сервер відправляє назад змінну, тому коли сторінка повторно промальовується (re-rendered), компоненти (controls) відображаються (render) з їхнім останнім станом. На серверній стороні програма може змінити viewstate, якщо результати обробки оновлюють хоча б один елемент управління на сторінці. Стани кожного компонента декодуються на сервері та доступні для використання в сторінках ASP.NET через колекцію ViewState.

Шаблони сторінок



Шаблони сторінок (приклад шаблону)

```
<%@ Master Language="C#"%>
<html>
    <head runat="server">
        <title>Example template</title>
    </head>
    <body>
        <form id="form1" runat="server">
            <div>
         <asp:Image ID="Image1" runat="server" ImageUrl="~/Images/1.png" />
                <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">

                </asp:ContentPlaceHolder>
            </div>
        </form>
    </body>
</html>
```

Шаблони сторінок (приклад сторінки шаблону)

```
<%@ Page Language="C#" MasterPageFile="~/ExampleTemplate.Master"%>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <asp:Image ID="Image2" runat="server" ImageUrl="~/Images/2.jpg" />
</asp:Content>
```