BÀI 2: ASP.NET Pages

- ✓ ASP Web Form Pages
- ✓ Page Class và Master Page
- ✓ Client-Side Scripts Manager

ASP.NET Pages

I. ASP.NET Web Form Pages

- Khi sử dụng ASP.NET Web Forms pages để lập trình giao diện cho ứng dụng Web, Web Forms hiển thị dữ liệu cho người dùng trên trình duyệt ở bất cứ thiết bị nào và thực thi các nghiệp vụ logic.

- Trong Web Form pages, lập trình giao diện bao gồm 2 phần

  - **Visual Components: Phần hiển thị**
  - **Logic: Phần nghiệp vụ**

Nghiệp vụ logic sẽ tương tác với phần hiển thị visual thông qua các đoạn mã chương trình server - side

ASP.NET Pages

Visual components:

- Phần hiển thị bao gồm một tệp chứa ngôn ngữ đánh dấu HTML hoặc mã ASP.NET hoặc cả hai.
- The ASP.NET Web Forms page thực thi như một vùng chứa cho các dữ liệu tĩnh (static) và các điều khiển mà ta muốn hiển thị

Logic:

- Phần nghiệp vụ logic của trang Web Form bao gồm đoạn mã chúng ta thiết lập để tương tác với dữ liệu hiển thị trên trang. Đoạn mã có thể được đặt trong khối mã lệnh **script** trong trang web hoặc trong một file với lớp dữ liệu riêng.
- Nếu khối mã lệnh trong trang thì được gọi là chế độ **code-inline**, ngược lại nếu đoạn mã trong một file định nghĩa lớp dữ liệu riêng sẽ được gọi là chế độ **code-behind**

Chế độ code-behind có thể được viết bởi bất kỳ NNLT nào trên .Net framework như C# hay VB.NET

ASP.NET Pages

1.1. Đặc trưng của ASP.NET Web Form pages

Một số đặc trưng khác biệt trong lập trình ứng dung Web

- *Implementing a rich Web user interface:* Giao diện phức tạp với nhiều loại dữ liệu
- *Separation of client and server:* Tách biệt giữa Client và Server
- *Stateless execution:* Không duy trì trạng thái
- *Unknown client capabilities:* Nhiều loại Client khác nhau
- *Complications with data access:* Truy cập dữ liệu phức tạp
- *Complications with scalability:* phức tạp khi ứng dụng cần mở rộng

ASP.NET Pages

1.2. Cấu trúc và chế độ Web Form Pages

Cấu trúc (.aspx)

- **Directive:** Chỉ thị tiền xử lý (thiết lập cấu hình tính năng và dữ liệu)
  <%@ Directive attribute = "value" %>
- **Content:** Nội dung dữ liệu hiển thị (HTML)

```
<html>
    <head> … </head>
    <body>
        <form runat="server">
            Các điều khiển trên server
        </form>
    </body>
</html>
```

- **Code:** Mã chương trình server (code-inline hoặc code-behind)
  Code-inline: <script runat="server"></script>
  Code-behind: *.cs/*.vb

ASP.NET Pages

1.2. Cấu trúc và chế độ Web Form Pages

Chế độ code-inline: sử dung 1 tệp .aspx

```
<%@ Page Language="C#" %>
<html>
<head runat="server"><title>Basic ASP.NET Web Page</title></head>
<body>
    <form id="form1" runat="server">
        <h1>Welcome to ASP.NET</h1>
        <p>Type your name and click the button.</p>
        <p><asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
                <asp:Button ID="Button1" runat="server" Text="Click"  OnClick="Button1_Click" />
        </p>
        <p> <asp:Label ID="Label1" runat="server"></asp:Label> </p>
    </form>
</body>
</html>
<script runat="server">
                protected void Button1_Click(object sender, System.EventArgs e) {
                                Label1.Text = ("Welcome, " + TextBox1.Text);
                }
</script>
```

ASP.NET Pages

1.2. Cấu trúc và chế độ Web Form Pages

Chế độ code-behind: sử dung 2 tệp .aspx và .aspx.cs/vb

Tệp .aspx

```
<%@ Page Language="C#" CodeFile="Tệp.cs" Inherits="MyPage" %>
<html>
<head runat="server">
    <title>Basic ASP.NET Web Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <h1>Welcome to ASP.NET</h1>
        <p>Type your name and click the button.</p>
        <p><asp:TextBox ID="TextBox1"                runat="server">
</asp:TextBox>
        <asp:Button ID="Button1" runat="server"  Text="Click"
OnClick="Button1_Click"/>
        </p>
        <p> <asp:Label ID="Label1"
                    runat="server"></asp:Label> </p>
    </form>
</body>
</html>
```

Tệp .cs

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

public partial class MyPage : System.Web.UI.Page {
    void Button1_Click(object sender, System.EventArgs e) {
                Label1.Text = ("Welcome, " + TextBox1.Text);
    }
}
```

Page Class

II. Đối tượng Page

Khi trang ASP.NET được gọi và kết xuất về trình duyệt, ASP.NET khởi tạo một đối tượng của lớp dữ liệu Page tham chiếu trang web. Lớp Page bao gồm:

- Chương trình (code) trong trang aspx
- Chương trình được khởi tạo bởi ASP.NET

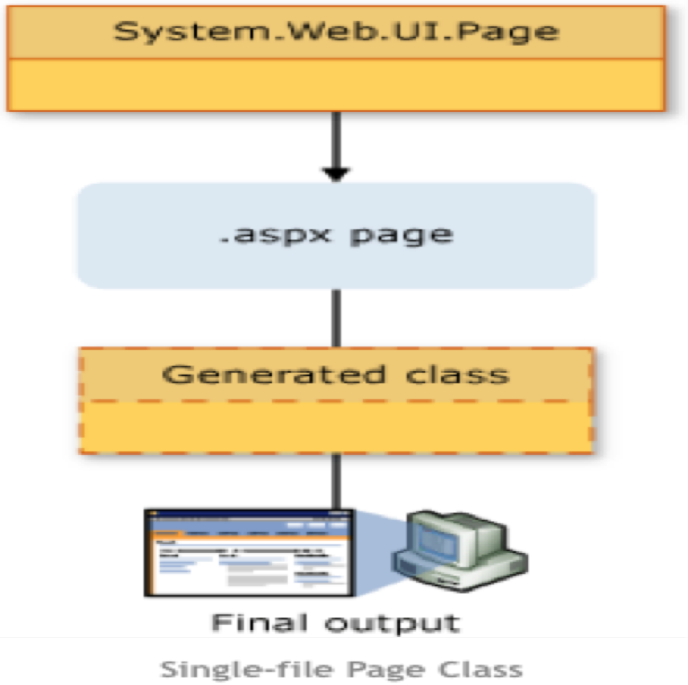
Khi thực thi, trang ASP.NET được đóng gói trong một lớp dữ liệu Page bao gồm các thành phần điều khiển server trong trang với các trình xử lý sự kiện.

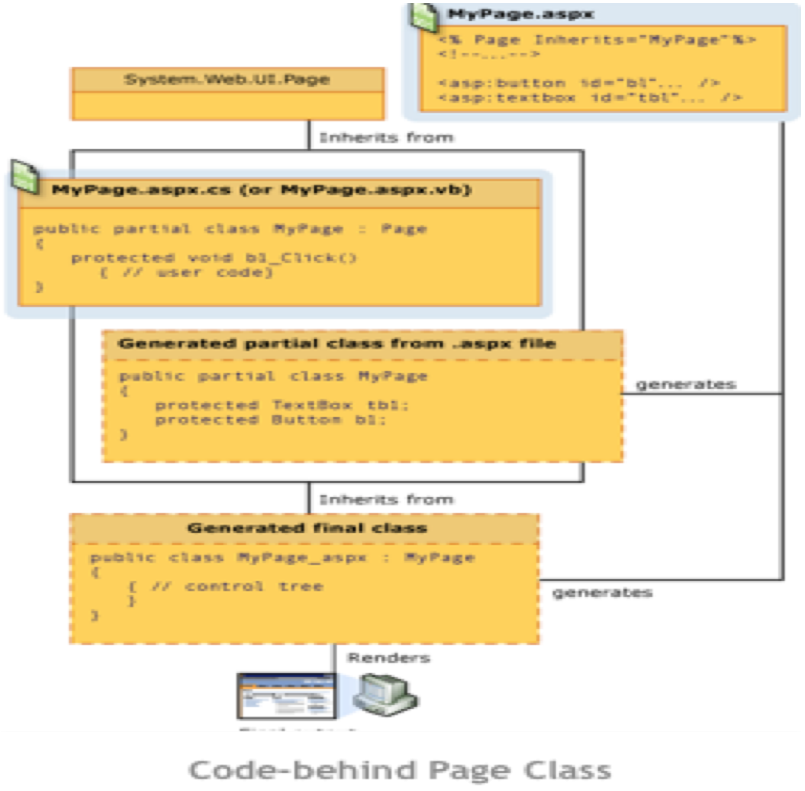
Đối với Website Project, ứng dụng không được biên dịch sẵn thành assembly nên ASP.NET sẽ biên dịch khi trang aspx được gọi. Nếu có sự thay đổi trong mã nguồn chúng sẽ được tự động biên dịch lại trong những lần tiếp theo

Website Application phải biên dịch trước khi triển khai

Page Class

II. Đối tượng Page





Mô hình khởi tạo đối tượng Page trong ASP.NET

Page Class

2.1. Giai đoạn hoạt động của Page Class

Khi trang ASP.NET thực thi đối tượng Page trải qua một vòng đời để thực hiện các bước xử lý dữ liệu. Các giai đoạn gồm:

- Initialization,
- Instantiating controls,
- Restoring and maintaining state,
- Running event handler code, and
- Rendering.

Vòng đời của Page cho cho ta biết giai đoạn thực hiện để áp dụng chương trình hiệu quả. Đặc biệt, khi phát triển custom controls, vòng đời của điều khiển phụ thuộc vòng đời của Page

Page Class

2.1. Giai đoạn hoạt động của Page Class

Giai đoạn	Mô tả
Page Request	Page được gọi, ASP.NET xác định có biên dịch hay dùng Cache. Nếu biên dịch sẽ bắt đầu vòng đời Page
Start	Giai đoạn bắt đầu sẽ thiết lập thuộc tính Request, Response và kiểm tra trạng thái Postback
Initialization	Giai đoạn khởi tạo các điều khiển trong trang, áp dụng Master và Theme (nếu có). Nếu là Postback các điều khiển chưa được cập nhật dữ liệu từ ViewState
Load	Trong giai đoạn Load, nếu là Postback thì sẽ cập nhật dữ liệu cho các điều khiển từ ViewState và Control State
Postback event handling	Nêu là Postback thì các handlers xử lý điều khiển được gọi, sau đó gọi Validate cho tất cả các điều khiển
Rendering	Giai đoạn kết xuất dữ liệu về trình duyệt, trước đó sẽ cập nhật lại trạng thái ViewState và các điều khiển và thiết lập outputStream cho đối tượng Response
Unload	Sau khi đã hoàn tất kết xuất dữ liệu về trình duyệt, giai đoạn này huỷ bỏ các Controls, Request, Response và huỷ Page

Page Class

2.1. Sự kiện và vòng đời của Page Class

Sự kiện	Ứng dụng
Pre Init	Sự kiện phát sinh sau giai đoạn bắt đầu, thường dùng để thiết lập Master Page hoặc Theme
Init	Thường sử dụng để đọc hoặc thiết lập thuộc tính cho các điều khiển
InitComplete	Chỉ sử dụng để bật ViewState
PreLoad	Cập nhật dữ liệu từ ViewState cho các điều khiển
Load	Đối tượng Page gọi sự kiện OnLoad cho Page và cho các điều khiển. Sự kiện chính để cập nhật dữ liệu và điều khiển
Control Event	Sự kiện của các điều khiển, và gọi các trình xử lý thực hiện. Thường sử dụng trong chế độ Postback
LoadComplete	Chỉ sử dụng với các chức năng yêu cầu tất cả các điều khiển đã hoàn tất dữ liệu

Page Class

2.1. Sự kiện và vòng đời của Page Class

Sự kiện	Ứng dụng
Prerender	Sử dụng để hiệu chỉnh nội dung dữ liệu cuối cùng sẽ được gửi về Client
Prerender Complete	Thường sử dụng với DataBinding khi các điều khiển hoàn tất gắn kết dữ liệu
SaveState Complete	Lưu lại trạng thái dữ liệu của ViewState cho vòng Postback tiếp theo
Render	Đây không phải sự kiện mà là giai đoạn các điều khiển và Page gọi phương thức Render để ghi dữ liệu lên Client
UnLoad	Huỷ bỏ các điều khiển, sau đó huỷ bỏ Page

Page Class

2.1. Sự kiện và vòng đời của Page Class

Ví dụ: PageCycle.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="PageCycle.aspx.cs"
Inherits="PageFlow" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
        <title>Page Flow</title>
</head>
<body>
  <form id="form1" runat="server">
  <div>
        <asp:Label id="lblInfo" runat="server" EnableViewState="False"></asp:Label>
        <asp:Button id="Button1" runat="server"Text="Button"  OnClick="Button1_Click">
        </asp:Button>
  </div>
  </form>
</body>
</html>
```

Page Class

2.1. Sự kiện và vòng đời của Page Class

Ví dụ: PageCycle.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;


public partial class PageFlow : System.Web.UI.Page {
  private void Page_Load(object sender, System.EventArgs e) {
      lblInfo.Text += "Page.Load event handled.<br/>";
      if. (Page.IsPostBack) {
          lblInfo.Text += "<b>second time</b><br/>";
      }
  }
  private void Page_Init(object sender, System.EventArgs e) {
      lblInfo.Text += "Page.Init event handled.<br/>";  }
  protected void Button1_Click(object sender,
              System.EventArgs e){
      lblInfo.Text += "Button1.Click event handled.<br/>";
  }
  private void Page_PreRender(object sender,
              System.EventArgs e)
  {
      lblInfo.Text += "Page.PreRender event handled.<br/>";
  }
  private void Page_Unload(object sender,
              System.EventArgs e)
  {
      lblInfo.Text += "Page.Unload event handled.<br/>";
  }
}
```

Page Class

2.3. Postback và ViewState

ViewState là phương pháp để ASP.NET tự động lưu trữ trạng thái của trang Web và các điều khiển sau mỗi quá trình **Postback** từ Client tới Server

ViewState là đối tượng được tự động khởi tạo bởi ASP.NET và có kiểu **System.Web.UI.StateBag** là đối tượng để lưu trữ trạng thái. StateBag thực thi interface **IDictionary** nên ta có thể thêm hoặc xoá dữl liệu trong ViewState theo từng cặp key value có kiểu object

Dạng thức:

Thêm dữ liệu:

```
ViewState [“Key”] = objectValue;
```

Xoá dữ liệu

```
ViewState.RemoveString(Key);
```

Đọc dữ liệu:

```
object o    = ViewState[“Key”];
type o      = (type) ViewState[“Key”];
```

ViewState không lưu trữ các giá trị mặc định của dữ liệu trong các điều khiển

Page Class

2.3. PostBack và ViewState

PageCounter.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile=”PageCounter.aspx.cs" Inherits="SimpleCounter" %>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title>Untitled Page</title>
</head>
<body>
   <form id="form1" runat="server">
   <div>
      <asp:Button ID="cmdIncrement" runat="server”
       OnClick="cmdIncrement_Click"  Text="Increment" />
      <br />
      <asp:Label ID="lblCount” runat="server"></asp:Label>
  </div>
  </form>
</body>
</html>
```

Page.Counter.aspx.cs

```
using System;
using System.Collections;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
public partial class SimpleCounter : System.Web.UI.Page {
   int counter;

   protected void cmdIncrement_Click(object sender,
      EventArgs e) {
      if (ViewState["Counter"] == null) {
         counter = 1;
      }
      else {
         counter = (int)ViewState["Counter"] + 1;
      }
      // ViewState["Counter"] = counter;
      lblCount.Text = "Counter: " + counter.ToString();
   }
  protected void Page_PreRender(object sender, EventArgs e) {
     ViewState.Add(”Counter", counter);
  }
}
```

Master Page

III. Master và Content

Master Page:

ASP.NET master pages cho phép ta tạo các layout thống nhất về UI, UX (look and feel) và các hoạt động cơ bản của một nhóm hoặc toàn bộ các trang web trong ứng dụng. Các trang web chứa nội dung cần hiển thị sẽ được tích hợp cùng với trang master khi users gọi và hiển thị với layout thống nhất. Trang master định nghĩa các vùng chứa dữ liệu hiển thị cho trang nội dung bởi điều khiển **ContentPlaceHolder**

Cấu trúc trang master: ***.master**

Directive:

```
<%@ Master Language="C#" %>
<%@ Master Language="C#" CodeFile="MasterPage.master.cs" Inherits="MasterPage" %>
```

Content:

```
<form runat=“server”>
        <asp:ContentPlaceHolder id=“ID” runat=“server”/>
        <asp:ContentPlaceHolder id=“ID” runat=“server”/>
</form>
```

Trang master chứa đầy đủ các thành phần content HTML (html, head, body, form) và ASP.NET. Thẻ head có thuộc tính runat=“server”

Master Page

III. Master và Content

Ví dụ: MySite.master

```
<%@ Master Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
    <head runat="server" >
        <title>Master page title</title>
    </head>
    <body>
        <form id="form1" runat="server">
            <table>
                <tr>
                    <td><asp:contentplaceholder id="Main" runat="server" /></td>
                    <td><asp:contentplaceholder id="Footer" runat="server" /></td>
                </tr>
            </table>
        </form>
    </body>
</html>
```

Master Page

III. Master và Content

Content Page

Trang nội dung là những trang .aspx được liên kết với trang .master để tích hợp nội dung và layout khi hiển thị:

```
<%@ Page Language="C#" MasterPageFile="~/MasterPages/MySite.master"
                                    Title="Content Page"%>
```

Trong trang nội dung ta xác định dữ liệu hiển thị bởi điều khiển **Content** cho các vùng chứa dữ liệu đã được thiết lập bởi **ContentPlaceHolder** trong trang master

```
<asp:Content ID="Content1" ContentPlaceHolderID="Main" Runat="Server">
        Main content.
</asp:Content>
```

Trang nội dung không chứa các thẻ chứa HTML (html, head, body, form).
Tiêu đề trang được xác định trong thuộc tính Title

Master Page

III. Master và Content

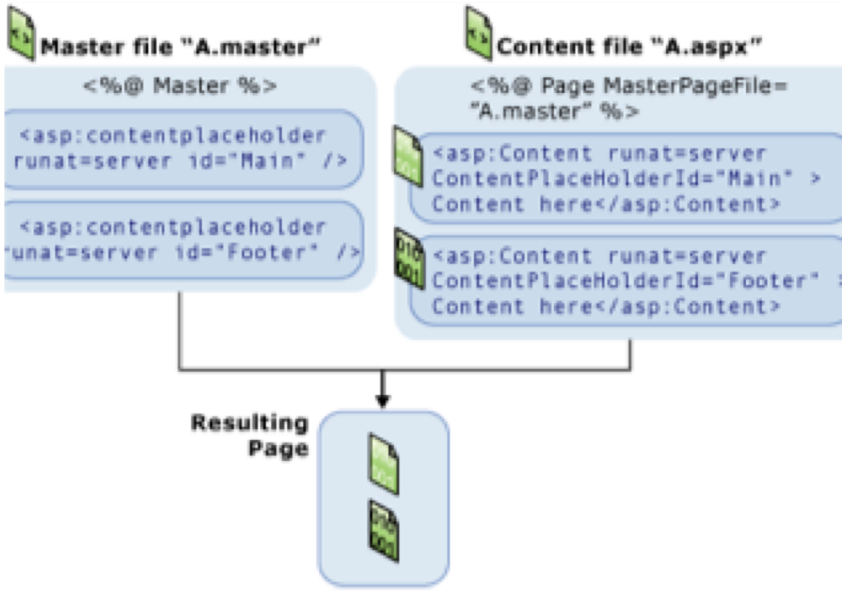
Ví dụ: main.aspx

```
<% @ Page Language="C#" MasterPageFile="~/MySite.master" Title="Content Page 1" %>
    <asp:Content ID="Content1" ContentPlaceHolderID="Main” Runat="Server">.
            Main content
    </asp:Content>
    <asp:Content ID="Content2" ContentPlaceHolderID="Footer” Runat="Server" >
            Footer content.
    </asp:content>
```

Master Page

III. Master và Content

Mô hình tích hợp hiển thị



Master page khi thực thi được tạo thành một bộ phận của content page.

Thực tế, nó hoạt động như một điều khiển server của trang content và là thành phần chứa của các điều khiển các trong trang

Master Page

Master và Content Pages

MasterPage.master

```
<%@ Master Language="C#" ClassName="MasterExample" %>
<script runat="server">
      public string SiteName   {
         get { return "My Site Name"; }
      }
</script>
<html >
<head runat="server">
   <title>MasterPage Example</title>
</head>
<body>
   <form id="form1" runat="server">
   <div>
      <asp:contentplaceholder id="ContentPlaceHolder1" runat="server">
      </asp:contentplaceholder>
   </div>
   </form>
</body>
</html>
```

ContentPage.aspx

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master" Title="MasterPage Example" %>
<script runat="server">
  protected void Page_Load(object sender, EventArgs e)  {
     MasterExample m = (MasterExample)Page.Master;
     mylabel.Text = m.SiteName;
  }
</script>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="Server"> Hello, Master Pages!
<asp:Label runat="server" Text="Label" ID="mylabel"></asp:Label>
</asp:Content>
```

Client Script

IV. Quản lý chương trình kịch bản Client

ASP.NET không bị giới hạn các công cụ và ngôn ngữ trên server. Ta có thể tích hợp mã Client ECMAScript (JavaScript hay JScript) trong trang ASP.NET để tăng cường chức năng trên Client. Hai cách tiếp cận để tích hợp mã Client

- Tạo và tích hợp mã Client: Sử dụng với đoạn mã nhỏ, có thể dùng với các điều khiển như CustomValidator
- Sử dụng AJAX: gồm cả framework cho phép Postback không đồng bộ (Bài Ajax)

Client Script

4.1. Custom Client-Side Script

ASP.NET kết xuất mã HTML về Client nên ta có thể tích hợp Client-Side Script trong trang ASP.NET. Nếu trang web chạy trên trình duyệt khác nhau của các thiết bị khác nhau thì khả năng hỗ trợ có thể thay đổi. 3 Tuỳ chọn để tích hợp

- Tích hợp mã Client tĩnh trong thẻ <script></script>: khi không cần dữ liệu phía server
- Tạo mã Client động với ClientScriptManager: khi cần có dữ liệu trên server trong lúc runtime
- Sử dụng ScriptManager của AJAX (Bài Ajax)

- **RegisterScirptBlock key type**
- **isScriptBlockRegister type key …**

Client Script

4.1.1. Static Script

Mã Client tính được tích hợp trong phần content của trang ASP.NET và kết xuất về trình duyệt như mã HTML thông thường. File .js cũng có thể được sử dụng và sẽ được lưu trong bộ nhớ Cache của trình duyệt như các dữ liệu khác.

Ví dụ:

MyScritpt.js

```
function sayHello() {  alert("Hello");  }
```

MyPage.aspx

```
<%@ Page Language="C#" %>
<html>
<head><title>Static Client</title>
        <script src="MyScript.js"></scritpt>
</head>
<body>
    <input type="button" value="Click" onClick="sayHello();" />
    <form runat="server"> </form>
</body>
</html>
```

Client Script

4.1.2. Dynamic Script

Trong trường hợp đoạn mã Client-Script dùng để xử lý dữ liệu được kết xuất từ Server lúc Runtime, ta phải tạo các thành phần Dynamic Script qua lớp ClientScriptManager. Các phương thức:

- **RegisterClientScriptBlock:** tạo khối script ở phần đầu trang web
- **RegisterStartupScript:** tạo khối script ở cuối trang web

ClientScriptManager xác định khối mã duy nhất thông qua cặp Key và Type để tránh xung đột. Trong trường hợp cần kiểm tra khối mã:

- **isClientScriptBlockRegister:** kiểm tra khối mã đầu đã đăng ký với Page
- **isClientScriptIncludeRegister:** Kiểm tra tệp **.js** đã đăng ký với Page
- **isStartupScriptRegister:** kiểm tra khối mã cuối đã đăng ký với Page

Client Script

BlockScript.aspx

```
<%@ Page Language="C#"%>

<script runat="server">
 public void Page_Load(Object sender, EventArgs e)
 {
  String csname1 = "PopupScript";
  String csname2 = "ButtonClickScript";
  Type cstype = this.GetType();

  ClientScriptManager cs = Page.ClientScript;

 if (!cs.IsStartupScriptRegistered(cstype, csname1))  {
    String cstext1 = "alert('Hello World');";
    cs.RegisterStartupScript(cstype, csname1, cstext1, true);
  }

if (!cs.IsClientScriptBlockRegistered(cstype, csname2))  {
    StringBuilder cstext2 = new StringBuilder();
    cstext2.Append("<script type=\"text/javascript\">
                function DoClick() {");
    cstext2.Append("Form1.Message.value='Text from client
                script.'} </");
    cstext2.Append("script>");
    cs.RegisterClientScriptBlock(cstype, csname2,
                cstext2.ToString(), false);
  }
 }
</script>

<html xmlns="http://www.w3.org/1999/xhtml" >
 <head>
  <title>ClientScriptManager Example</title>
 </head>
 <body>
  <form id="Form1"
     runat="server">
    <input type="text" id="Message" />
    <input type="button" value="ClickMe"
                onclick="DoClick()" />
  </form>
 </body>
</html>
```

Client Script

Include.js

```
function DoClick() {
    Form1.Message.value='Text from include script.'
}
```

IncludeJS.aspx

```
<%@ Page Language="C#"%>
<script runat="server">
   public void Page_Load(Object sender, EventArgs e) {
      String csname = "ButtonClickScript";
      String csurl = "~/ncludeJS.js";
      Type cstype = this.GetType();

      ClientScriptManager cs = Page.ClientScript;
      if (!cs.IsClientScriptIncludeRegistered(cstype, csname) {
         cs.RegisterClientScriptInclude(cstype, csname,
                                        ResolveClientUrl(csurl));
      }
   }
</script>
```

```
<html xmlns="http://www.w3.org/1999/xhtml" >
  <head>
    <title>ClientScriptManager Example</title>
  </head>
  <body>
    <form id="Form1" runat="server">
    <div>
      <input type="text"
           id="Message"/>
      <input type="button"
           value="ClickMe"
           onclick="DoClick()"/>
    </div>
    </form>
  </body>
</html>
```