

Bài 1. Tổng quan về ASP.Net

- **Mục đích, yêu cầu:** Bài học cung cấp kiến thức cơ sở lý thuyết tổng quan về ASP.NET, kiến trúc ASP.NET, Code phía server, cách thức truyền dữ liệu giữa các trang, chuyển trang. Sau khi học xong bài học này sinh viên có thể xây dựng được các trang Web Form sử dụng điều khiển Html, Server và biết cách truyền dữ liệu giữa các trang Web.

- **Hình thức tổ chức dạy học:** Lý thuyết + tự học

- **Thời gian:** Lý thuyết(trực tiếp: 3). Tự học, tự nghiên cứu: 06

- **Nội dung:**

1. ASP.NET VÀ WEB FORM.....	2
1.1. Mô hình lập trình phía máy chủ	2
1.2. Cơ chế xử lý file ASP.NET phía máy chủ.....	5
1.3. Cấu trúc một trang ASP.Net	7
1.4. Tổng quan về ASP.Net Server controls.....	10
1.5. Các điều khiển Web Server Controls cơ bản	12
2. CÁC ĐỐI TƯỢNG TRONG ASP.NET	21
2.1. Request Object	21
2.2. Response Object	23
2.3. Server Object.....	26
2.4. Session Object.....	26
2.5. Application Object.....	26

1. ASP.NET VÀ WEB FORM

1.1. Mô hình lập trình phía máy chủ

Trong thế giới web, tất cả các giao tiếp giữa Client (trình duyệt) và Server (web server) đều được thực hiện theo cơ chế “***Request and Response***”. Tức là, trước tiên phía máy khách cần phải “request” (gửi yêu cầu) tới Server, sau đó phía server sẽ “response” (hồi đáp) lại yêu cầu.

Cùng một cơ chế này, người ta có 2 cách tiếp cận để xử lý “request trang web” từ máy khách:

Cách 1: Khi máy khách yêu cầu một trang – ví dụ trang **abc**. – thì máy chủ sẽ đọc toàn bộ nội dung của trang và gửi về cho phía máy khách mà không thực hiện bất kỳ xử lý nào. Nó hoàn toàn không qua tâm đến ý nghĩa bên trong của trang **abc**. Nội dung trang này sau đó sẽ được phía trình duyệt xử lý.

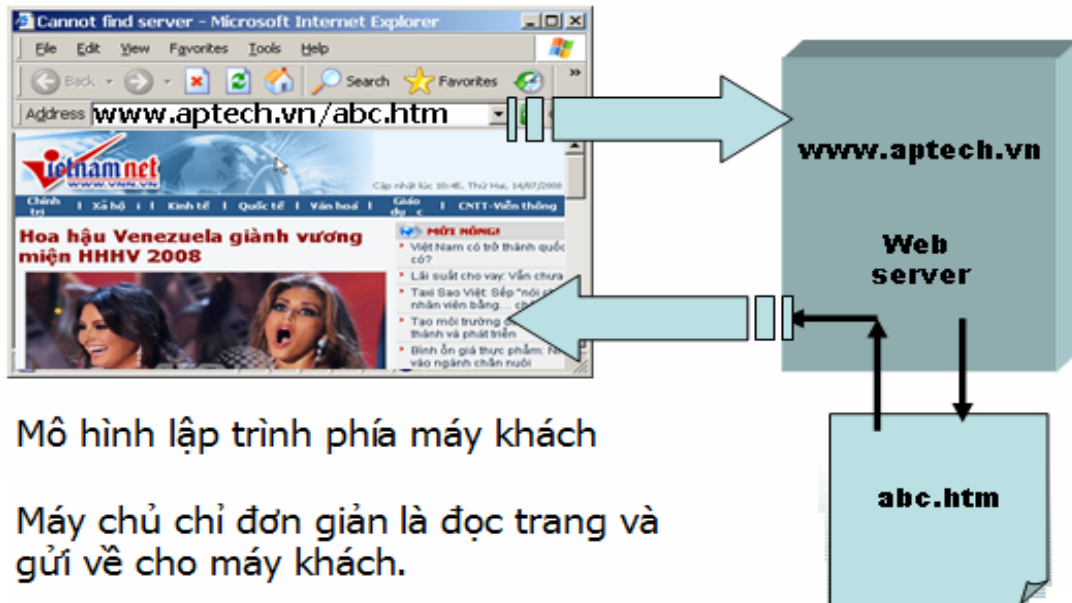
Cách 2: Khi máy khách yêu cầu một trang – ví dụ trang **xyz**. – thì máy chủ sẽ đọc toàn bộ nội dung của trang đó và **xử lý tại Server** (trước khi gửi về cho client) **để được kết quả**, tiếp theo lấy kết quả xử lý được gửi về cho phía máy khách. Kết quả trả về cho máy khách có thể chứa các phần tử HTML, các câu lệnh JavaScript, các định nghĩa kiểu CSS....và tiếp tục được phía client (trình duyệt) xử lý như cách 1.

Với cách 1, do việc xử lý không diễn ra bên phía server nên trang web không thể đọc/ ghi các dữ liệu trên Server được (ví dụ Danh sách khách hàng, danh mục sản phẩm,...). Vì vậy nó chỉ phù hợp với các trang web đơn giản, không đòi hỏi xử lý chi tiết.

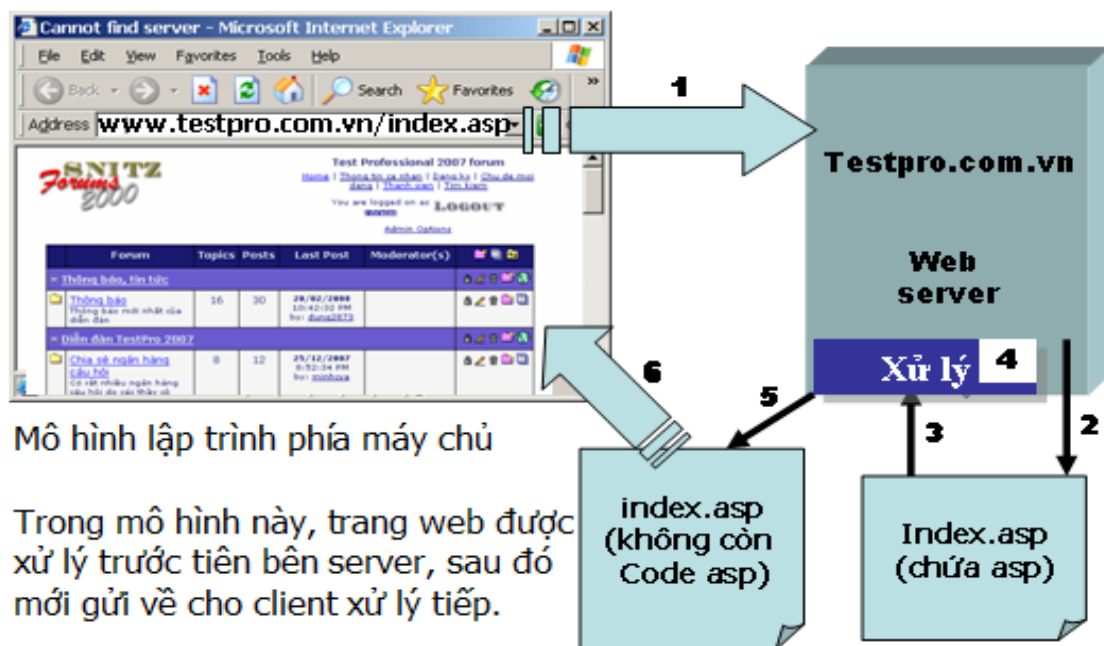
Với cách 2, do việc xử lý thông tin ở tại server nên hoàn toàn có thể đọc/ ghi dữ liệu trên chính server đó. Vì vậy, nó phù hợp với các dự án lớn và tính bảo mật cao. Mô hình theo cách này gọi là mô hình lập trình phía máy chủ.

Dưới đây là hình ảnh minh họa cho 2 mô hình này:

❖ Mô hình lập trình phía máy khách (Client side)



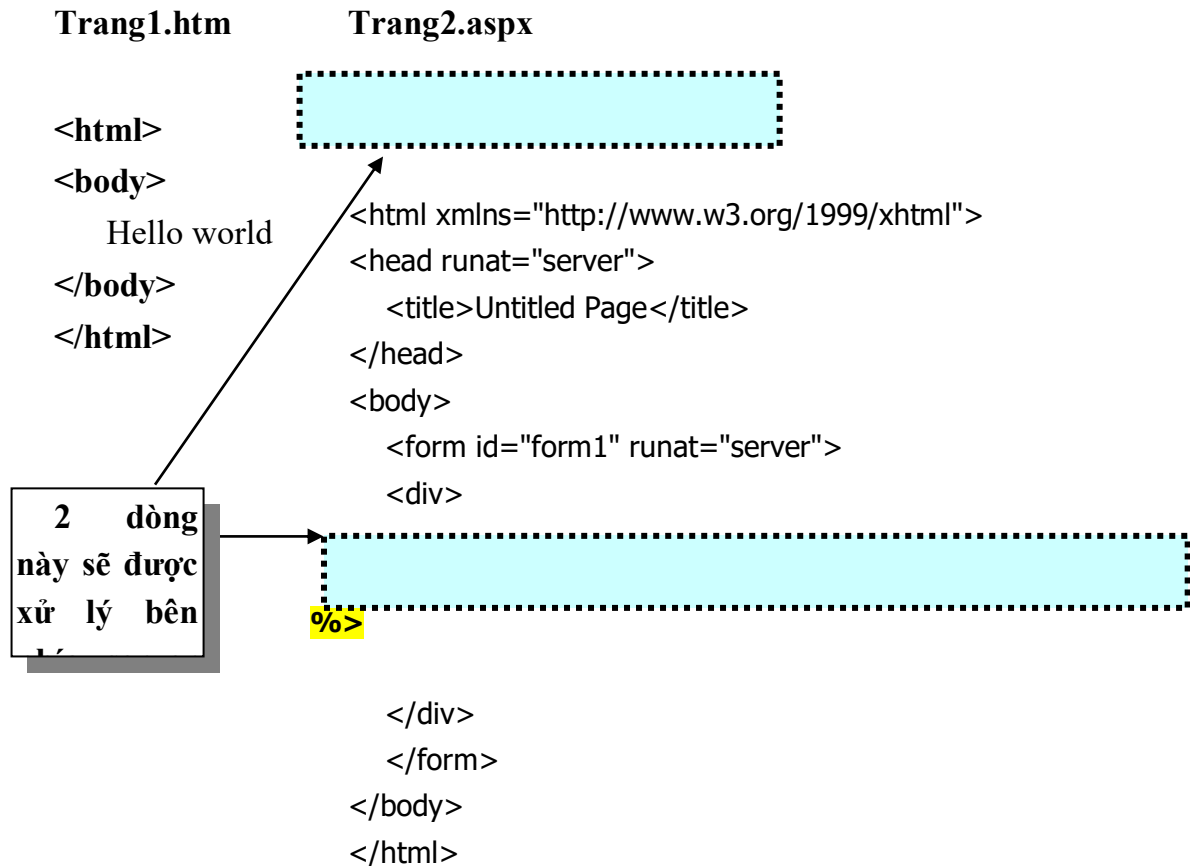
❖ Mô hình lập trình phía máy chủ



Câu hỏi: Khi nào thì một trang sẽ được xử lý ở bên Server trước ?. hay nói cách khác là khi nào thì được gọi là xử lý theo mô hình phía server?

Trả lời: Các trang (file) có đuôi mở rộng mà server có thể xử lý, ví dụ: asp, php, jsp, aspx...

Câu hỏi: Có thể lấy một ví dụ về một trang sẽ được xử lý phía server và trang sẽ không được xử lý phía server ?



Câu hỏi: Chương trình Client và server có nhất thiết phải nằm trên hai máy tính riêng biệt không ? và Client là các trình duyệt rồi (IE, FireFox...), còn server là chương trình nào ?

Trả lời: Hai chương trình này hoàn toàn có thể nằm trên cùng một máy tính. Chương trình server thực chất là một chương trình có tên là IIS (Internet Information Service).

Câu hỏi: Phải viết như thế nào để server hiểu là cần phải xử lý bên phía server trước khi gửi về cho phía Client ?

Trả lời: Trước tiên phải đặt phần mở rộng cho file (ví dụ .aspx), sau đó trong trình duyệt cần phải đặt những nội dung muốn xử lý bên phía server trong cặp thẻ đặc biệt, ví dụ:

```
<% Response.Write (DateTime.Today.Date.ToString ()); %>
```

Hoặc:

```
<form id="form1" runat="server">
  <asp:Calendar runat="server" ID="Lịch"> </asp:Calendar>
</form>
```

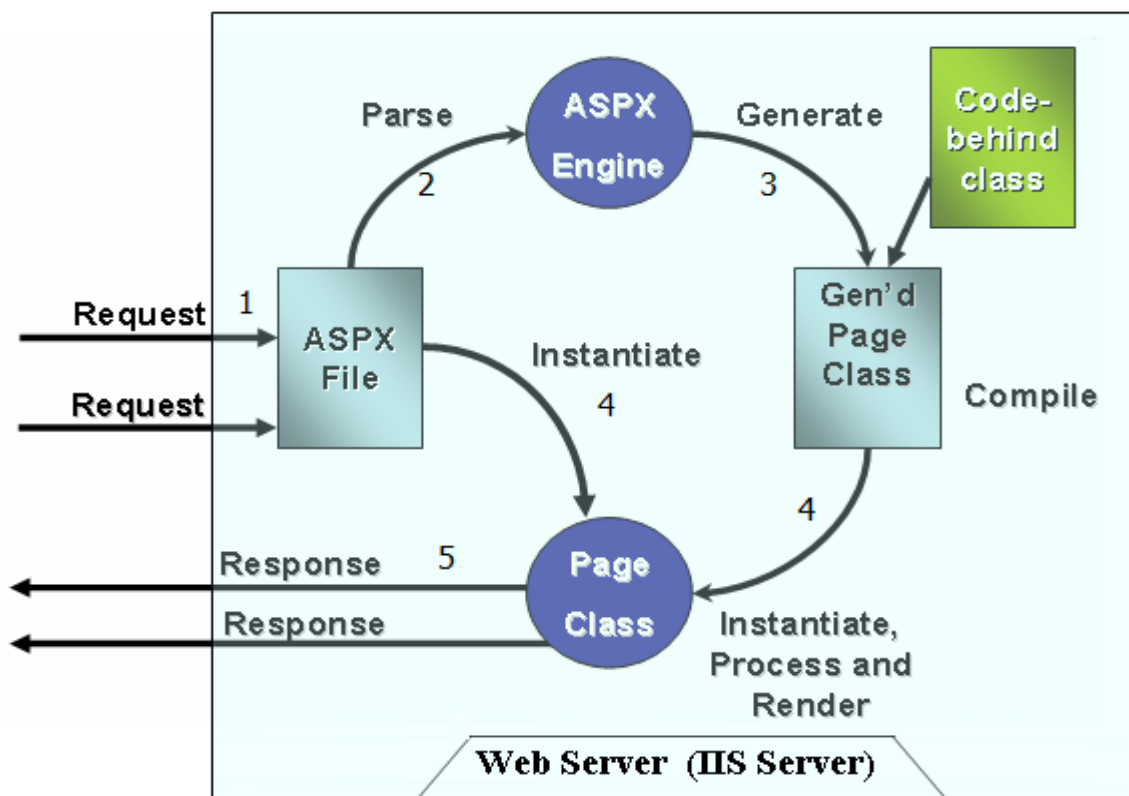
*** Chính các ký hiệu `<% %>` và **Runat = "Server"** đã "mách bảo" Server là : "Hãy xử lý nội dung đó bên phía server đi"! . Nếu không có những ký hiệu này thì mặc nhiên server làm mỗi việc là gửi trả lại cho trình duyệt xử lý.

Câu hỏi: Sao không gửi ngay cho trình duyệt xử lý như trước đây mà cứ phải để server xử lý ...!. Để Client xử lý sẽ giảm tải cho server, điều này chẳng tốt hơn sao ?

Trả lời: Vì trình duyệt chỉ có thể hiểu và xử lý được các thẻ HTML và Javascript thôi, còn nó không thể xử lý được các nội dung phức tạp. Ví dụ nó không hiểu **asp:Calendar** là gì ?

1.2. Cơ chế xử lý file ASP.NET phía máy chủ.

Đối với các trang ASP.NET, thì cơ chế xử lý giống như đã mô tả ở trên, tức là theo mô hình xử lý bên phía server. Nhưng có bổ sung thêm tính năng Compile and Cache:

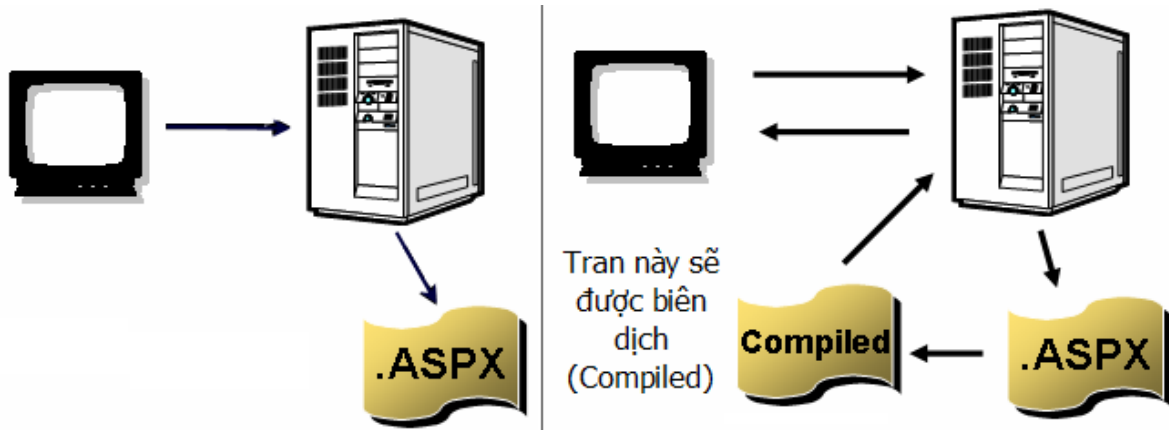


Giải thích cơ chế xử lý ở trên:

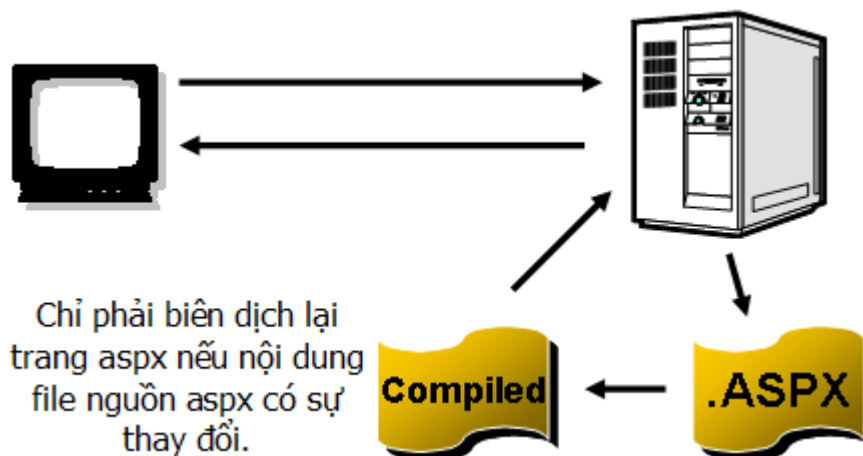
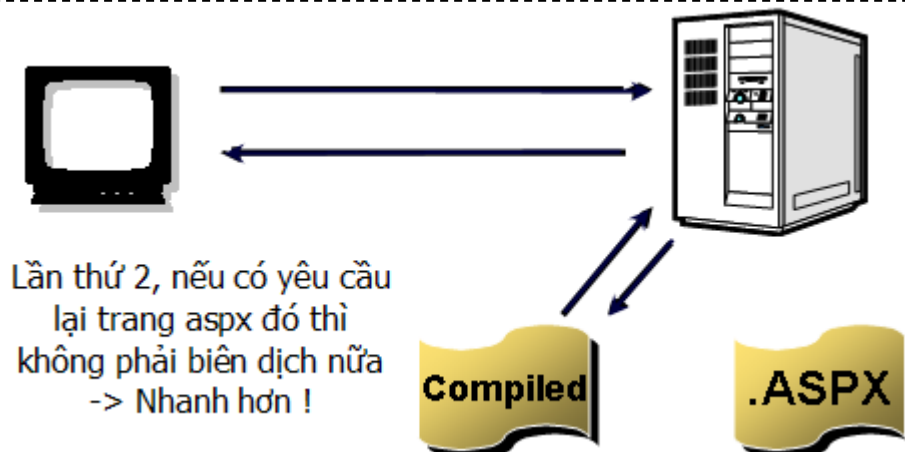
- Bước 1: Người lập trình phải tạo các trang ASPX (giả sử tên trang đó là **abc.aspx**) và đặt nó vào trong thư mục web của web server (có tên là www.server.com). Trên thanh địa chỉ của trình duyệt, người dùng nhập trang www.server.com/abc.aspx.
- Bước 2: Trình duyệt gửi yêu cầu tới server với nội dung: **"Làm ơn gửi cho tôi trang abc.aspx thì tốt !"**.

- Bước 3: web server sẽ biên dịch code của trang aspx (bao gồm cả các mã code vb.net/ c# - gọi là code behind hay code file) thành class.
- Bước 4: Lớp sau khi được biên dịch sẽ thực thi.
- Bước 5: trả kết quả về cho trình duyệt

Riêng với ASP.NET thì việc biên dịch sẽ được thực hiện “thông minh hơn”, như sau:



Lần đầu tiên trang aspx được yêu cầu.



1.3. Cấu trúc một trang ASP.Net

Một trang ASP.NET bao gồm cả phần giao diện người dùng và phần xử lý logic bên trong. Giao diện người dùng chịu trách nhiệm hiển thị các thông tin và tiếp nhận dữ liệu từ người dùng, trong khi đó phần xử lý (lập trình) đảm nhiệm việc điều khiển sự tương tác của người dùng với trang web. Phần giao diện người dùng bao gồm một file chứa ngôn ngữ đánh dấu – như HTML hoặc XML và server controls chẳng hạn. File này được gọi là một **Trang (Page)** và có đuôi mở rộng là **aspx**.

Phần đáp ứng các tương tác của người dùng với trang web được thực hiện bởi một ngôn ngữ lập trình chẳng hạn như Visual Basic.NET và C#. Chúng ta có thể thực hiện việc viết code bằng bất kỳ ngôn ngữ lập trình nào được hỗ trợ bởi CLR ở ngay trong trang ASPX hoặc tách ra một file riêng. File tách riêng này được gọi là file Code Behind hay mới đây gọi là Code file. Đuôi mở rộng của Code file là **.VB** (Nếu dùng ngôn ngữ Visual Basic) hoặc **.CS** (nếu dùng ngôn ngữ C#).

Một trang ASP.Net gồm 2 thành phần:

- Phần giao diện (file *.aspx)
- Phần xử lý - lập trình (file *.aspx.cs)

Để viết code (C#,VB.Net,..) xây dựng một trang web asp.net ta có 2 cách sau:

- **Cách 1:** Viết code trực tiếp trong trang giao diện *.aspx:

- ✓ Thông qua cặp thẻ <% ... %>
- ✓ Thông qua cặp thẻ <script ... > </script>

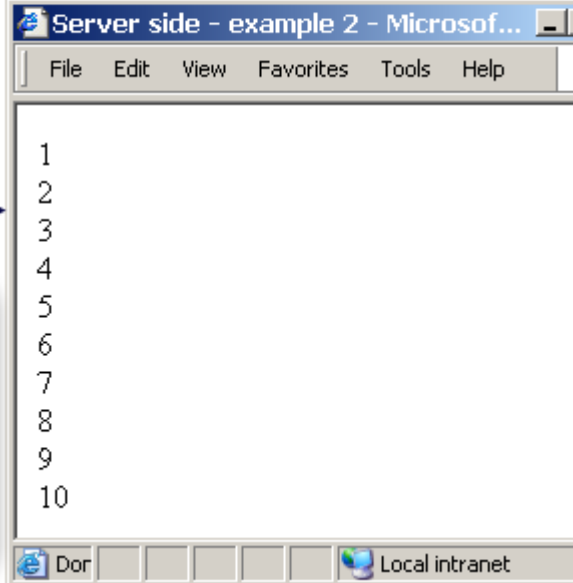
- **Cách 2:** Viết code trong trang code *.aspx.cs (code-behind thường dùng)

Ví dụ 1: viết code trực tiếp trong trang *.aspx thông qua cặp thẻ <% %>

```

<%@ Page Language="C#" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Server side - example 2</title>
</head>
<body>
  <form id="form1" runat="server">
    <% int i;
       for (i = 1; i <= 10; i++)
         Response.Write (i + "<br>");
    %>
  </form>
</body>
</html>

```



Khối lệnh ở trên sẽ được phía server xử lý trước, sau đó mới lấy kết quả xử lý được gửi trả lại cho phía Client (trình duyệt) :

Câu hỏi: Kết quả trả về cho client trong trường hợp này cụ thể là gì ???

Ví dụ 2: viết code trực tiếp trong trang *.aspx thông qua cặp thẻ <script>

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TinhTong_Script.aspx.cs" Inherits="TinhTong" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<asp:TextBox ID="txtA" runat="server" Width="50px" />
<asp:TextBox ID="txtB" runat="server" Width="50px" />
<asp:Button ID="cmdTinhTong" Text="Tính" runat="server" OnClick="Tong" />
<asp:TextBox ID="txtKetQua" runat="server" Width="50px" />
<script runat="server">
public void Tong(object s, EventArgs e)
{
txtKetQua.Text = (int.Parse(txtA.Text) + int.Parse(txtB.Text)).ToString();
}
</script>
</form>
</body>
</html>
```



Ví dụ 3: viết code trong trang code behind *.aspx.cs

```

*File *.aspx
<body>
  <form id="form1" runat="server">
    <asp:TextBox ID="txtA" runat="server" Width="50px" />
    <asp:TextBox ID="txtB" runat="server" Width="50px" />
    <asp:Button ID="cmdTinhTong" Text="Tính" runat="server" OnClick="Tong" />
    <asp:TextBox ID="txtKetQua" runat="server" Width="50px" />
  </form>
</body>
</html>

*File *.aspx.cs
using System.Web.UI.WebControls;

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

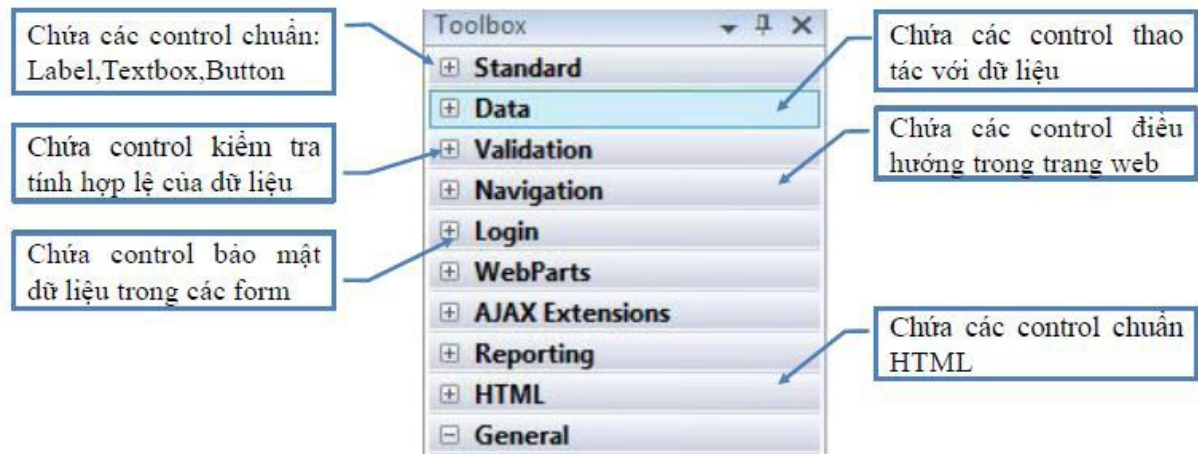
  protected void Tong(object sender, EventArgs e)
  {
    txtKetQua.Text = (int.Parse(txtA.Text) + int.Parse(txtB.Text)).ToString();
  }
}

```

1.4. Tổng quan về ASP.Net Server controls

1.4.1 Giới thiệu

Để giúp cho việc phát triển các ứng dụng web nhanh chóng và thuận tiện, ASP.NET cung cấp cho chúng ta một tập hợp các điều khiển sẵn có để thực hiện hầu hết các công việc phổ biến hàng ngày. Các điều khiển này chia làm 2 loại: **HTML Server Controls** và **Web server controls**.



1.4.2 HTML Server controls

a) HTML controls

- Được tạo ra từ các thẻ HTML tĩnh
- Thường được sử dụng lập trình phía client

b) Chuyển đổi HTML controls thành HTML Server controls

- Sử dụng **HTML controls** để lập trình phía server, ta phải thêm thuộc tính **runat="server"**.
- Điều khiển có thuộc tính **runat="server"** gọi là **HTML Server controls**

***Lưu ý:** thường dùng **HTML Server controls** khi:

- Điều khiển cần phải có đoạn **JavaScript** kèm theo sự kiện
- Có nhiều code **JavaScript** tham chiếu đến điều khiển đó

1.4.3 Web Server controls

- **Web Server controls** là đối tượng của .Net Framework
- Được chuyển đổi sang dạng HTML tĩnh lúc thực thi
- Thường được sử dụng lập trình phía server
- HTML Source lúc thiết kế có dạng:

<asp: Kiểu_đk ds_thuộc_tính runat = "server" />

Ví dụ: **<asp:TextBox ID="txtHVT" runat="server" />**

***Lưu ý:** **Web Server controls** có nhiều thuộc tính hơn, thực hiện được nhiều chức năng phức tạp hơn **HTML Server controls**

1.5. Các điều khiển Web Server Controls cơ bản

Dưới đây là các lý do bạn nên sử dụng Web Server Control:

- ☐ + Đơn giản, tương tự như các điều khiển trên Windows Form.
- ☐ + Đồng nhất: Các điều khiển Web server có các thuộc tính giống nhau nên dễ tìm hiểu và sử dụng.
- ☐ + Hiệu quả: Các điều khiển Web Server tự động phát sinh ra các tag HTML theo từng loại Browser.

Bảng liệt kê các thuộc tính chung của các Web control

Thuộc tính	Kiểu	Ý nghĩa
(ID)	Chuỗi	Qui định tên của điều khiển. Tên của điều khiển là duy nhất.
AccessKey	String	Qui định ký tự để di chuyển nhanh đến điều khiển - ký tự xử lý phím nóng.
Attributes	AttributeCollection	Tập hợp các thuộc tính của điều khiển HTML.
BackColor	Color	Qui định màu nền của điều khiển.
BorderColor	Color	Qui định màu đường viền của điều khiển.
BorderStyle	BorderStyle	Qui định kiểu đường viền của điều khiển.
BorderWidth	Unit	Qui định độ rộng của đường viền.
CssClass	String	Qui định hình thức hiển thị của điều khiển qua tên CSS.
Enabled	Boolean	Qui định điều khiển có được hiển thị hay không. Giá trị mặc định của thuộc tính này là True – được phép hiển thị.
Font	FontInfo	Qui định Font hiển thị cho điều khiển.
ForeColor	Color	Qui định màu chữ hiển thị trên điều khiển
Height	Unit	Qui định chiều cao của điều khiển.
ToolTip	String	Dòng chữ sẽ hiển thị khi rê chuột vào điều khiển.
Width	Unit	Qui định độ rộng của điều khiển.

1.5.1 Label

Label thường được sử dụng để hiển thị và trình bày nội dung trên trang web. Nội dung được hiển thị trong label được xác định thông qua thuộc tính Text. Thuộc tính Text có thể nhận và hiển thị nội dung với các tag HTML.

Ví dụ:

```
lblA.Text = "Đây là chuỗi văn bản thường" ;
```

```
lblB.Text = "<B>Còn đây là chuỗi văn bản được in đậm</B>" ;
```

1.5.2 HyperLink

Điều khiển này được sử dụng để tạo ra các liên kết siêu văn bản.

Các thuộc tính

- ☐ + ImageURL: Qui định hình hiển thị trên điều khiển.
- ☐ + Text: Chuỗi văn bản sẽ được hiển thị trên điều khiển. Trong trường hợp cả 2 thuộc tính ImageURL và Text được thiết lập, thuộc tính ImageURL được ưu tiên, thuộc tính Text sẽ được hiển thị như Tooltip.
- ☐ + NavigateUrl: Đường dẫn cần liên kết đến.
- ☐ + Target: Xác định cửa sổ sẽ hiển thị cho mỗi liên kết
- ☐ + _blank: Hiển thị trang liên kết ở một cửa sổ mới.
- ☐ + _self: Hiển thị trang liên kết tại chính cửa sổ chứa liên kết đó.
- ☐ + _parent: Hiển thị trang liên kết ở frame cha.

Ví dụ:

```
hplASP_net.Text = "Trang chủ ASP.Net" ;  
hplASP_net.ImageUrl = "Hinh\Asp_net.jpg";  
hplASP_net.NavigateUrl = "http://www.asp.net" ;  
hplASP_net.Target = "_blank";
```

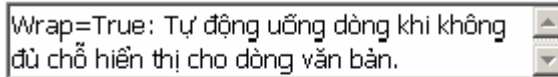
1.5.3 TextBox

TextBox là điều khiển được dùng để nhập và hiển thị dữ liệu. TextBox thường được sử dụng nhiều với các ứng dụng trên windows form.

Các thuộc tính

- + Text: Nội dung chứa trong Textbox
- + TextMode: Qui định chức năng của Textbox, có các giá trị sau:
 - ☐ -SingleLine: Hiển thị và nhập liệu 1 dòng văn bản
 - ☐ - MultiLine: Hiển thị và nhập liệu nhiều dòng văn bản
 - ☐ - Password: Hiển thị dấu * thay cho các ký tự có trong Textbox.
- + Rows: Trong trường hợp thuộc tính TextMode = MultiLine, thuộc tính Rows sẽ qui định số dòng văn bản được hiển thị.
- + Maxlength: Qui định số ký tự tối đa được nhập vào cho TextBox
- +Wrap: Thuộc tính này qui định việc hiển thị của văn bản có được phép tự động xuống dòng khi kích thước ngang của cửa điều khiển không đủ để hiển thị dòng nội dung văn bản. Giá trị mặc định của thuộc tính này là True - tự động xuống dòng.

Ví dụ:


 Wrap=True: Tự động uốn dòng khi không đủ chỗ hiển thị cho dòng văn bản.


 Wrap=False: Văn bản được hiển thị bình thường.

- + AutoPostBack: Thuộc tính này qui định điều khiển có được phép tự động PostBack về Server khi nội dung trong Textbox bị thay đổi hay không. Giá trị mặc định của thuộc tính này là False - không tự động Postback.

1.5.4 Image

Điều khiển này được dùng để hiển thị hình ảnh lên trang Web.

Thuộc tính

- + ImageURL: Đường dẫn đến tập tin hình ảnh cần hiển thị.
- +AlternateText: Chuỗi văn bản sẽ hiển thị khi tập tin được thiết lập trong thuộc tính ImageURL không tồn tại.
- +ImageAlign: Vị trí hiển thị giữa hình và nội dung văn bản.

1.5.5 Button, ImageButton, LinkButton

Các điều khiển Button, ImageButton, LinkButton mặc định đều là các nút Submit Button, mỗi khi được nhấn vào sẽ PostBack về Server.

Khi chúng ta thiết lập giá trị thuộc tính CommandName cho các điều khiển này, chúng ta gọi tên chung cho các điều khiển này là Command Button.

Các thuộc tính chung của Button, ImageButton, LinkButton	
Thuộc tính	Ý nghĩa
Text	Chuỗi văn bản hiển thị trên điều khiển.
CommandName	Tên lệnh. Được sử dụng trong sự kiện Command.
CommandArgument	Thông tin bổ sung cho sự kiện Command.
CausesValidation	Trang web mặc định kiểm tra tính hợp lệ dữ liệu mỗi khi được PostBack. Các điều khiển Button, ImageButton, LinkButton luôn PostBack về Server mỗi khi được nhấn → luôn kiểm tra tính hợp lệ dữ liệu trên trang web. Muốn trang Web bỏ qua việc kiểm tra dữ liệu khi được nhấn, gán trị cho thuộc tính này = False. Giá trị mặc định của thuộc tính này là True.

Chúng ta sẽ tìm hiểu 2 thuộc tính CommandName và CommandArgument ở phần sau.

Ngoài những thuộc tính trên, điều khiển ImageButton còn có các thuộc tính ImageURL, ImageAlign và AlternateText như điều khiển Image.



Button, LinkButton và ImageButton

Ví dụ: Các điều khiển: Label, Textbox, Button



* Trang giao diện (file *.aspx)

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server"> <title></title> </head>
<body>
  <form id="form1" runat="server">
    <div>
      <table border="1" cellpadding="0" cellspacing="0">
        <tr>
          <td style="width:40%">
            <asp:Label ID="Label1" runat="server" Text="Nhập số A:" Width="100px"></asp:Label>
          </td>
          <td> <asp:TextBox ID="txtSoA" runat="server" Width="100px"></asp:TextBox></td>
        </tr>
        <tr>
          <td> <asp:Label ID="Label2" runat="server" Text="Nhập số B:" Width="100px"></asp:Label> </td>
          <td> <asp:TextBox ID="txtSoB" runat="server" Width="100px"></asp:TextBox> </td>
        </tr>
        <tr>
          <td> <asp:Label ID="Label3" runat="server" Text="Tổng:"></asp:Label> </td>
          <td> <asp:TextBox ID="txtTong" runat="server" Width="100px"></asp:TextBox> </td>
        </tr>
        <tr>
          <td colspan="2" align="center">
            <asp:Button ID="ButTinh" runat="server" Text="Tính tổng" OnClick="ButTinh_Click" />
          </td>
        </tr>
      </table>
    </div>
  </form>
</body>
</html>
```

*** Trang xử lý (file *.aspx.cs)**

```
protected void btTong_Click(object sender, EventArgs e)
{
    int so1,so2,tong;
    so1 = Int32.Parse(txtSoA.Text);
    so2 = Int32.Parse(txtSoB.Text);
    tong = so1 + so2;
    txtTong.Text = tong.ToString();
}
```

1.5.6 Listbox và DropDownList

ListBox và DropDownList là điều khiển hiển thị danh sách lựa chọn mà người dùng có thể chọn một hoặc nhiều (chỉ dành cho ListBox). Các mục lựa chọn có thể được thêm vào danh sách thông qua lệnh hoặc ở cửa sổ thuộc tính (Property Windows).

a. Các thuộc tính

- + AutoPostBack: Thuộc tính này qui định điều khiển có được phép tự động PostBack về Server khi chỉ số của mục chọn bị thay đổi. Giá trị mặc định của thuộc tính này là False - không tự động Postback.
- + Items: Đây là tập hợp chứa các mục chọn của điều khiển. Ta có thể thêm vào mục chọn vào thời điểm thiết kế thông qua cửa sổ ListItem Collection Editor, hoặc thông qua lệnh.
- + Rows: Qui định chiều cao của ListBox theo số dòng hiển thị.
- + SelectionMode: Thuộc tính này xác định cách thức chọn các mục trong ListBox. SelectionMode chỉ được phép thay đổi trong quá trình thiết kế, vào lúc thực thi chương trình, thuộc tính này chỉ đọc.
 - ☐ - Single: Chỉ được chọn một mục có trong danh sách (mặc định).
 - ☐ - Multiple: Cho phép chọn nhiều lựa chọn.

b. Xử lý mục chọn

Các thuộc tính sau sẽ giúp bạn xác định chỉ số, giá trị của mục đang được chọn. Trong trường hợp điều khiển cho phép chọn nhiều, ta duyệt qua các Item trong tập hợp Items, sử dụng thuộc tính Selected của đối tượng Item để kiểm tra xem mục đó có được chọn hay không.

- + SelectedIndex: Cho biết chỉ số của mục được chọn. Trong trường hợp chọn nhiều mục, SelectedIndex sẽ trả về chỉ số mục chọn đầu tiên.
- + SelectedItem: Cho biết mục được chọn. Trong trường hợp chọn nhiều mục, SelectedItem sẽ trả về mục chọn đầu tiên.

– + SelectedValue: Cho biết giá trị của mục được chọn. Trong trường hợp chọn nhiều mục, SelectedValue sẽ trả về giá trị mục chọn đầu tiên.

c. Tìm hiểu về tập hợp Items

– + Add: Thêm mục mới vào cuối danh sách, sử dụng phương thức Items.Add
 Items.Add(<String>)
 Items.Add(<ListItem>)
 – + Insert: Thêm mục mới vào danh sách tại một vị trí nào đó, sử dụng phương thức Items.Insert

Items.Insert(<index>, <ListItem>)

Items.Insert(<index>, <String>)

– + Count: Trả về số mục (Item) có trong danh sách. Items.Count

– + Contains: Kiểm tra xem một Item đã có trong tập hợp Items hay chưa, nếu có, phương thức này sẽ trả về giá trị True, ngược lại, trả về False.

Items.Contains(<ListItem>)

– + Remove: Xóa đối tượng Item tại ra khỏi danh sách.

Items.Remove(<ListItem>)

Items.Remove(<Chuoi>)

□ Trong trường hợp các đối tượng Item là kiểu chuỗi, ta truyền vào một chuỗi để xóa. Nếu có nhiều giá trị giống nhau trong danh sách, chỉ có mục chọn đầu tiên bị xóa.

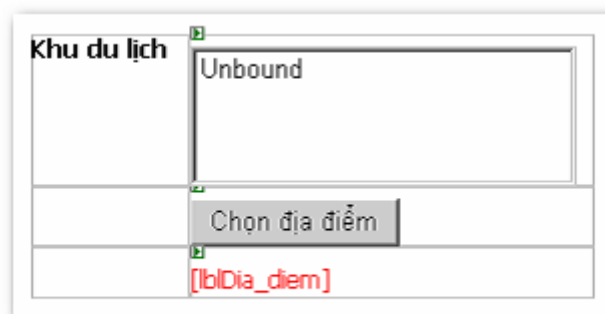
□ Trong trường hợp các đối tượng Item là đối tượng, ta truyền vào một biến tham chiếu đến item cần xóa.

– + RemoveAt: Xóa một item tại vị trí index ra khỏi danh sách.

Items.RemoveAt(<index>)

– + Clear: Phương thức Clear của tập hợp Items được dùng để xóa tất cả những Item có trong danh sách. Cú pháp: Items.Clear

Ví dụ: Điều khiển danh sách ListBox: ID=lstKhu_dl; SelectionMode=Multiple; Rows=4.



Khi thiết kế

*** Trang giao diện (file *.aspx)**

```
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
<form id="form1" runat="server">
<div>
<table border="1" cellpadding="0" cellspacing="0">
<tr>
<td style="width:40%"> <asp:Label ID="Label1" runat="server" Text="Khu du lịch"></asp:Label>
</td>
<td> <asp:ListBox ID="lstKhu_dl" SelectionMode="Multiple" Rows="4" runat="server"
Width="250"></asp:ListBox> </td>
</tr>
<tr>
<td colspan="2" align="center">
<asp:Button ID="BtChon" runat="server" Text="Chọn địa điểm" OnClick="BtChon_Click" />
</td>
</tr>
<tr>
<td colspan="2"> <asp:Label ID="LblDia_Diem" runat="server" Text=""></asp:Label> </td>
</tr>
</table>
</div>
</form>
</body>
</html>
```

*** Trang xử lý (file *.aspx.cs)**

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        lstKhu_dl.Items.Add("Vịnh Hạ Long");
        lstKhu_dl.Items.Add("Phan Thiết - Mũi Né");
        lstKhu_dl.Items.Add("Nha Trang");
        lstKhu_dl.Items.Add("Đà Lạt");
    }
}

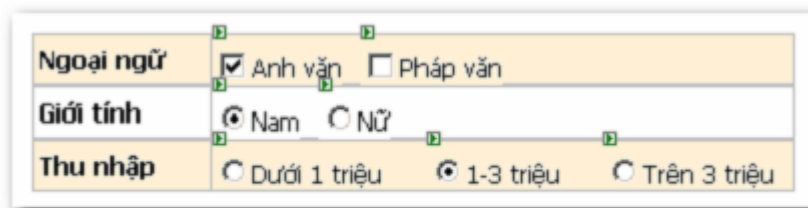
protected void btChon_Click(object sender, EventArgs e)
{
    LblDia_Diem.Text = "";
    if (lstKhu_dl.SelectedItem.Selected)
        LblDia_Diem.Text= "Bạn đã chọn: " + lstKhu_dl.SelectedValue ;
}
```

1.5.7 Checkbox, RadioButton

a. Các thuộc tính

- + Checked: Cho biết trạng thái của mục chọn - có được chọn hay không
- + TextAlign: Qui định vị trí hiển thị của điều khiển so với chuỗi văn bản.
- + AutoPostBack: Thuộc tính này qui định điều khiển có được phép tự động PostBack về Server khi các mục chọn của điều khiển bị thay đổi. Giá trị mặc định của thuộc tính này là False - không tự động Postback.
- + GroupName (RadioButton): Tên nhóm. Thuộc tính này được sử dụng để nhóm các điều khiển RadioButton lại thành 1 nhóm.

b. Ví dụ



Nhóm các RadioButton Giới tính, Thu nhập

Danh sách các điều khiển:

Điều khiển	Loại	Thuộc tính	Giá trị
chkAnh_van	CheckBox	Checked	True
chkPhap_van	CheckBox		
rbtNam	RadioButton	Checked	True
		GroupName	Gioi_tinh
rbtNu	RadioButton	GroupName	Gioi_tinh
rbtThu_nhapA	RadioButton	GroupName	Thu_nhap
rbtThu_nhapB	RadioButton	Checked	True
		GroupName	Thu_nhap
rbtThu_nhapC	RadioButton	GroupName	Thu_nhap

1.5.8 CheckBoxList, RadioButtonList

Hai điều khiển này được dùng để tạo ra một nhóm các CheckBox/Radio Button. Do đây là điều khiển danh sách nên nó cũng có thuộc tính Items chứa tập hợp các mục chọn như ListBox/DropDownList. Các thao tác trên tập hợp Items, xử lý mục chọn cũng tương tự như ListBox/DropDownList.

a. Các thuộc tính

- + RepeatColumns: Qui định số cột hiển thị.
- + RepeatDirection: Qui định hình thức hiển thị

- ☐ -Vertical: Theo chiều dọc
- Horizontal: Theo chiều ngang
- ☐ + AutoPostBack: Thuộc tính này qui định điều khiển có được phép tự động PostBack về Server khi các mục chọn của điều khiển bị thay đổi. Giá trị mặc định của thuộc tính này là False - không tự động Postback.

b. Ví dụ

Điều khiển RadioButtonList: ID=rblThu_Nhap; AutoPostBack=true; RepeatColumns=2; Items=...



* Trang giao diện (file *.aspx)

```
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
<form id="form1" runat="server">
<div>
<table border="1" cellpadding="0" cellspacing="0" >
<tr>
<td style="width:30%"> <asp:Label ID="Label1" runat="server" Text="Thu nhập"></asp:Label> </td>
<td>
<asp:RadioButtonList ID="rblThu_Nhap" runat="server" AutoPostBack="True" RepeatColumns="2"
Width="280px">
<asp:ListItem>Dưới 1 triệu</asp:ListItem>
<asp:ListItem>1-3 triệu</asp:ListItem>
<asp:ListItem>Trên 3 triệu</asp:ListItem>
<asp:ListItem>Trên 10 triệu</asp:ListItem>
<asp:ListItem>Trên 15 triệu</asp:ListItem>
<asp:ListItem>Trên 20 triệu</asp:ListItem>
</asp:RadioButtonList>
</td>
</tr>
<tr>
<td colspan="2">
<asp:Label ID="lblThu_Nhap" runat="server"></asp:Label>
</td>
</tr>
</table>
</div>
```

```

    </form>
  </body>
</html>

```

* Trang xử lý (file *.aspx.cs)

```

protected void rblThu_Nhap_SelectedIndexChanged(...)
{
    lblThu_Nhap.Text = "Bạn chọn thu nhập: " + rblThu_Nhap.SelectedItem.Text;
}

```

2. CÁC ĐỐI TƯỢNG TRONG ASP.NET

Trong bất kỳ ứng dụng nào, dù là winform based hay webform based thì việc giao tiếp (tương tác) với người dùng và giữa các webform với nhau là điều bắt buộc. Ví dụ ta cần phải lấy thông tin đặt hàng do người dùng nhập vào và hiển thị trở lại người dùng một số thông tin hữu ích khác, như kết quả thanh toán...hay một trang chuyển tiếp kết quả cho một trang khác để xử lý v.v...

Ở bài trước, để làm điều này chúng ta thực hiện dễ dàng thông qua các server controls như textbox, listbox, dropdownlist, label,... Tuy nhiên những điều khiển này chỉ có tác dụng trong một Page còn các trang khác thì hoàn toàn không thể đọc/ghi giá trị nằm trong các điều khiển này.

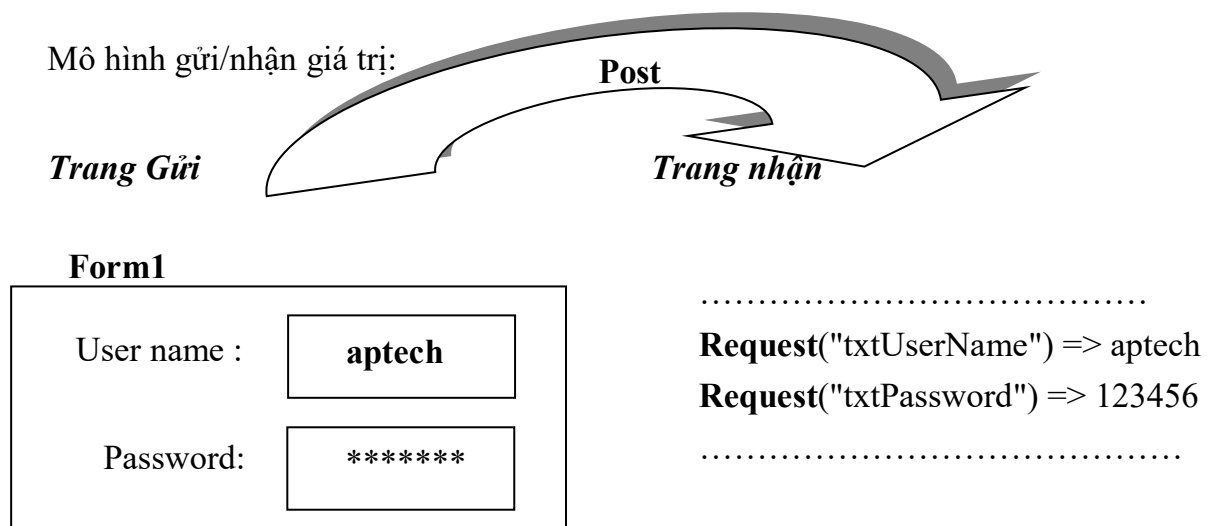
Để thực hiện việc giao tiếp (truyền dữ liệu) giữa các webform ASP.NET cung cấp một tập các điều khiển giúp ta làm việc đó một cách dễ dàng, đó là: Đối tượng Request và đối tượng Response.

Trong bài học này, chúng ta cũng tìm hiểu thêm một số đối tượng khác cũng rất hay dùng khi xây dựng ứng dụng là đối tượng Server, Application và Session.

2.1. Request Object

2.1.1 Đối tượng Request dùng để làm gì ?

Request là một đối tượng của ASP.NET, dùng để nhận các thông tin từ Client gửi về cho Web server.

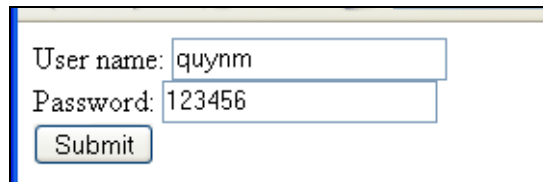


2.1.2 Các thành phần (thuộc tính và phương thức) chính

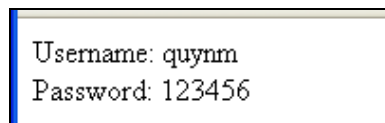
- Phương thức **Request.QueryString.Get("Tên_Phần tử cần đọc")**: Để đọc giá trị của một phần tử được gửi theo phương thức Get (Method = "Get")
- Phương thức **Request.Form.Get("Tên_Phần tử cần đọc")**: Để đọc giá trị của một phần tử được gửi theo phương thức Post (Method = "Post").
- Thuộc tính **QueryString**: cho phép nhận các giá trị truyền qua chuỗi tham số

Cú pháp: `Biến=Request.QueryString["Tên_tham_số"];`

Ví dụ 1: Xây dựng 2 trang web : trang Webform1.aspx, trong đó có 2 textbox chứa tên và mật khẩu. Khi người dùng click vào nút *submit* thì gửi tên và mật khẩu sang trang Webform2.aspx để hiển thị.



Trang nguồn (gửi): Webform1.aspx



Kết quả nhận về: trang Webform2.aspx

Code của 2 trang sẽ như sau:

Webform1.aspx

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Trang gửi</title>
</head>
<body>
  <form id="form1" runat="server" method="post" >
  <div>
    User name:
    <asp:TextBox runat="server" ID="txtUserName"></asp:TextBox><br />
    Password:
    <asp:TextBox runat="server" ID="txtPassword"></asp:TextBox><br />
    <asp:Button runat="server" ID="cmdSubmit" Text="Submit"
PostBackUrl="~/Bai1/WebForm2.aspx" />
  </div>
</form>
```

```
</body>
```

```
</html>
```

WebForm2.aspx

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

```
<head runat="server">
```

```
<title>Trang nhận</title>
```

```
</head>
```

```
<body>
```

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

```
<asp:Label ID="lblKetQua" runat="server"></asp:Label>
```

```
</form>
```

```
</body>
```

```
</html>
```

WebForm2.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
```

```
{
```

```
lblKetQua.Text = "Username: " + Request.Form.Get("txtUserName")
```

```
+ "<br>Password: " + Request.Form.Get("txtPassword");
```

```
}
```

Ví dụ 2: (tự làm)

Xây dựng 2 trang web : trang **TienDien.aspx**, trong đó có 2 textbox chứa chỉ số điện cũ và mới. Khi người dùng click vào nút **Tính** thì gửi chỉ số điện cũ và mới sang trang **XL_TienDien.aspx** để hiển thị số tiền phải trả. Biết rằng: 100 kw đầu giá là: 2000đ/kw

50 kw tiếp giá là: 2500đ/kw

50 kw tiếp giá là: 2800đ/kw

>200 kw giá là: 3500đ/kw

Vào chỉ số điện cũ:	<input type="text" value="100"/>
Vào chỉ số điện mới:	<input type="text" value="300"/>
<input type="button" value="Tính tiền"/>	

***XL_TienDien.aspx.cs**

2.2. Response Object

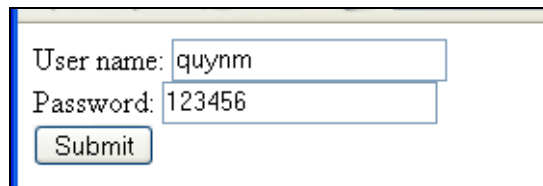
2.2.1 Đối tượng Response dùng để làm gì ?

Đối tượng này được dùng để gửi nội dung (một chuỗi) bất kỳ về cho trình duyệt. Quản lý và điều phối thông tin từ Web Server đến trình duyệt của người dùng.

2.2.2 Các thành phần (thuộc tính và phương thức) chính

- Phương thức: `Response.write(<Biểu thức>)` dùng để gửi giá trị biểu thức truyền vào cho phía trình duyệt.
- Phương thức: `Flush` dùng để đưa dữ liệu còn trong bộ đệm phía server về cho phía trình duyệt.
- Phương thức `Response.Redirect("địa chỉ URL")`: Chuyển tới một trang khác.

Ví dụ1: Tạo một trang Login như ví dụ 1, sử dụng đối tượng `Response` không dùng `Method = "Post/Get"`.



Code của 2 trang sẽ như sau:

Webform1.aspx

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Trang gửi</title>
</head>
<body>
  <form id="form1" runat="server" >
    <div>
      User name:
      <asp:TextBox runat="server" ID="txtUserName" runat="server"></asp:TextBox><br />
```

Password:

```
      <asp:TextBox runat="server" ID="txtPassword"></asp:TextBox><br />
      <asp:Button runat="server" ID="cmdSubmit" Text="Submit" />
    </div>
  </form>
</body>
</html>
```

Webform1.aspx.cs

```
protected void cmdSubmit_Click(object sender, EventArgs e)
{
    string sName = txtUserName.Text;
    string sPass = txtPassword.Text;
    Response.Redirect("WebForm2.aspx?Name=" + sName + "&Pass=" + sPass);
}
```


WebForm2.aspx

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Trang nhận</title>
</head>
<body>
  <form id="form1" runat="server">
    <asp:Label ID="lblKetQua" runat="server"></asp:Label>
  </form>
</body>
</html>
```

WebForm2.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    lblKetQua.Text = "Username: " + Request.QueryString["Name"].ToString() +
        "<br> Password: " + Request.QueryString["Pass"].ToString();
}
```

Ví dụ 2: (tự làm theo 2 cách: Request và Response)

Xây dựng 2 trang web : trang **TienLuong.aspx**.

Mã NV:	NV04	
Bậc lương:	4	
Ngày công:	27	
Chức vụ:	Trưởng phòng ▼	
<input type="button" value="TÍNH"/> <input type="button" value="XÓA"/>		

Khi người dùng click vào nút **Tính** thì gửi kết quả sang trang **XL_TienLuong.aspx**

Mã NV:	NV04	
Có ngày công:	27	
Tiền được lĩnh là:	79500000	

$TienLinh = BacLuong * 650000 * NCTL + PhuCap$

Với: + $NCTL = NgayCong$ nếu $NgayCong < 25$

$= (NgayCong - 25) * 2 + 25$ nếu $NgayCong \geq 25$

+ $PhuCap = 500000$ nếu chức vụ là Trưởng phòng

$= 300000$ nếu chức vụ là Phó phòng

=100000 nếu chức vụ là Nhân viên

2.3. Server Object

- Dùng để cung cấp thông tin của Web Server cho ứng dụng
- Phương thức:
 - Transfer(“URL”): ngừng thi hành trang hiện hành, gửi yêu cầu mới tới trang khác
 - MapPath(): trả về đường dẫn vật lý tương ứng với đường dẫn ảo trên Web Server

Ví dụ:

- In ra tên của máy chủ hiện hành: **Response.Write(Server.MachineName);**
- Cho biết đường dẫn thực sự trên ổ cứng (thư mục vật lý) của trang hiện hành (trang default.aspx) : **Server.MapPath(“default.aspx”);**
- Cho biết đường dẫn vật lý ứng với tệp QLCB.Mdb, biết rằng tệp này nằm trong một thư mục con là “App_Data”: **Server.MapPath(“App_Data/QLDB.MDB”);**

2.4. Session Object

- Dùng để lưu trữ thông tin trong một phiên làm việc cụ thể.
- Được tạo ra khi người dùng kết nối tới Web Server lần đầu tiên.

Nói cách khác, đối tượng (biến) **session** là một biến mà mọi trang trong một phiên (Session) đều có thể truy xuất

+ Cú pháp để **tạo biến** Session:

Session[“Tên_Biến”]=Giá_trị;

Ví dụ : Tạo một biến tên là MaNguoiDung và gán giá trị là TK34

Session[“MaNguoiDung”]=“TK34”;

+ Cú pháp để **lấy giá trị** từ biến session:

Biến=Session[“Tên_Biến”];

Riêng với đối tượng **Session**, nó còn có các sự kiện. Các sự kiện này tự động được gọi mỗi khi một phiên làm việc được tạo ra. Các sự kiện này có tên là **_Start** và **_End**. Các sự kiện này được đặt trong file **Global.asax**.

Lưu ý : Tập tin **Global.asax** được dùng để:

- + Khai báo và khởi tạo giá trị cho các biến Application, Session.
- + Viết xử lý cho các sự kiện của 2 đối tượng Application và Session.

2.5. Application Object

- Đối tượng toàn cục, quản lý toàn bộ ứng dụng Web

- Thông tin được lưu trong đối tượng (biến) Application được “hiểu” ở tất cả các trang aspx trong suốt thời gian “sống” của ứng dụng.
- Chỉ bị đóng/hủy khi tắt Web Server.
- Tạo biến Application:

Application["tên_biến"]=giá_trị;

- Lấy giá trị:

<biến>=Application["tên_biến"];

+ Ví dụ: Tạo biến So_Nguoi_Truy_Cap

Application["So_Nguoi_Truy_Cap"] = 0;

Ngoài ra, đối tượng Application còn có 2 phương thức thường dùng là **Application.Lock()**: Để khóa không cho người khác sửa đổi các biến toàn cục và **Application.Unlock()** để mở khóa .

Đối tượng Application cũng có 2 sự kiện đó là **Application_Start** và **Application_END**. Sự kiện **Start** chỉ được kích hoạt duy nhất một lần khi yêu cầu đầu tiên phát sinh. Sự kiện **END** được kích hoạt khi dịch vụ web dừng (unload).

Đối tượng Application có 2 phương thức là Lock và Unlock. Khi gọi phương thức Lock (khóa) thì tất cả các ứng dụng không được phép thay đổi các giá trị Application. Để các ứng dụng khác được phép thay đổi các biến Application thì gọi phương thức Unlock. Mã lệnh viết cho 2 sự kiện này cũng được đặt trong file **Global.asax**.

Ví dụ 1: Tạo một trang **Login**, nếu người dùng nhập user name và mật khẩu tương ứng là **son** và **123** thì được phép truy cập các trang **Home.aspx**, trái lại mỗi lần người dùng truy cập đến trang **Home.aspx** thì đều được chuyển tới trang **Login.aspx**.

Cần tạo 3 trang là Home.aspx/cs, Login.aspx/cs và Global.asax như sau:

Global.asax

...

Void Session_Start(object sender, EventArgs e)

{ Session["TrangThai"]=""; }

...

Home.aspx

. . .

<body>

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

<h1><asp:Label ID="lblThongbao" runat="server"> </asp:Label> </h1>

</form>

</body>

</html>

Home.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["TrangThai"] == "roi")
        lblThongbao.Text = "Chào mừng bạn đến Website của chúng tôi!";
    else
    {
        lblThongbao.Text = "Bạn chưa đăng nhập!";
        Response.Write("<a href='Login.aspx'> Login </a>");
    }
}
```

Login.aspx

```
...
<body>
    <form id="form1" runat="server">
        <table align="center">
            <tr>
                <td>User name (son): </td>
                <td>
                    <asp:TextBox runat="server" ID="txtUserID" >
                    </asp:TextBox>
                </td>
            </tr>
            <tr>
                <td>Password (123): </td>
                <td><asp:TextBox runat="server" ID="txtPassword"
TextMode="Password"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td colspan="2" align="center"> <asp:Button runat="server" Text="Login"
ID="cmdLogin" onclick="cmdLogin_Click"/> </td>
            </tr>
        </table>
    </form>
</body>
</html>
```

Login.aspx.cs

```
protected void cmdLogin_Click(object sender, EventArgs e)
{

```

```
if (txtUserID.Text == "son" && txtPassword.Text == "123")
{
    Session["TrangThai"] = "roi";
    Response.Redirect("Home.aspx");
}
else
{
    Session["TrangThai"] = "chua";
    Response.Write("Bạn đăng nhập sai!");
}
```

Ví dụ 2: Tạo một trang đếm số lượng người truy cập. Dùng biến tệp text để lưu.

- Tạo file **SL.txt** nhập giá trị 0
- Tạo 2 trang là Index.aspx/cs và Global.asax với nội dung sau:

Trang Index.aspx

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

```
<!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>Home Page - Hit counter</title>
</head>
<body>
    <form id="form1" runat="server">
        <h1>Chào mừng bạn đã đến website của chúng tôi</h1>
        <asp:Label runat="server" ID="lblSLKhach"></asp:Label>
    </form>
</body>
</html>
```

Trang Index.aspx.cs

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

```
{  
    lblSLKhach.Text="Bạn là vị khách thứ: " +  
        Application["SLTruyCap"].ToString();  
}  
}
```

Trang
Global.asax

```
<%@ Application Language="C#" %>  
  
<script runat="server">  
    void Application_Start(object sender, EventArgs e)  
    { //Đọc file SL.txt lấy giá trị gán cho biến S  
        Application.Lock ();  
        System.IO.StreamReader sr;  
        sr = new System.IO.StreamReader (Server.MapPath ("SL.txt"));  
        string S = sr.ReadLine ();  
        sr.Close ();  
        Application.UnLock ();  
        //Tạo một biến Applcation là SLTruyCap và khởi tạo giá trị lấy từ biến S  
        Application["SLTruyCap"]=S;  
    }  
    ...  
    void Session_Start(object sender, EventArgs e)  
    {  
        //Tăng số lượng người truy cập lên 1 khi có một người mới thăm  
        Application["SLTruyCap"] =  
            int.Parse (Application["SLTruyCap"].ToString ()) + 1;  
        //Lưu vào file SL.txt (mở và ghi đè)  
        System.IO.StreamWriter sw;  
        sw = new System.IO.StreamWriter (Server.MapPath ("SL.txt"));  
        sw.Write (Application["SLTruyCap"].ToString ());  
        sw.Close ();  
    }  
</script>
```

Sau khi tạo, chạy file **Index.aspx** để kiểm chứng sẽ thấy rằng số lượng người truy cập luôn luôn tăng lên bất kể là server có tắt hay máy tính bị trục trặc.