Bài 6: Session và Application

1. Đối tượng Session

Session là đối tượng được khởi động ngay khi người sử dụng gửi **request** tới ứng dụng Web và kết thúc khi người sử dụng ra khỏi ứng dụng. Mỗi người sử dụng truy nhập vào Web Site sẽ có một đối tượng Session riêng biệt.

Session được xây dựng cho server để bổ trợ cho giao thức HTTP trong việc quản lý users. Trong giao thức HTTP, server nhận request từ user và trả lại một response và sau đó không có khả năng xác định user. Trong trường hợp này đối với server, mỗi request được gửi tới bởi một user. Vì vậy để xác định được user trong Web Site gồm nhiều trang là không thể.

Đối tượng session được thiết kế để lưu trữ các thông tin của từng người sử dụng truy nhập vào các trang của Web Site. Các thông tin này có thể sử dụng vào nhiều mục đích khác nhau ví dụ như theo dõi được user truy nhập vào những trang Web nào và làm những gì.

2. Các thuộc tính và phương thức

2.1. Các thuộc tính

- **Contents:** Tập hợp các thành phần (biến và đối tượng) được khai báo thuộc loại session, ngoại trừ sử dụng thẻ <object>
- **StaticObjects:** Tập hợp các đối tượng khai báo thuộc loại session bằng thẻ <object>
- **CodePage:** Xác định mã văn bản mà sever sử dụng để hiển thị
- **Mode:** Chế độ kiểm soát session (InProc, StateServer, SQL-Server, Off)
- **SessionID:** Trả lại mã ID của mỗi session
- **Timeout:** Thiết lập thời gian để kết thúc một session không làm việc (20 mins)

2.2. Phương thức và sự kiện

- **Abandon():** Kết thúc một phiên làm việc
- **Remove(string)** Xoá một thành phần theo key
- **RemoveAll()** Xoá toàn bộ Contents
- **RemoveAt(int)** Xoá theo vị trí
- **OnStart:** Sự kiện xảy ra khi bắt đầu một phiên làm việc của user
- **OnEnd:** Sự kiện xẩy ra khi kết thúc một phiên làm việc

Chú ý: Sự kiện OnStart và OnEnd phải được khai báo trong tệp Global.asax

3. Sử dụng Session

3.1. Khai báo Session

- Khai báo biến đối tượng thuộc loại Session: (phạm vi)
  Session [“member”] = value;
- Trong tệp Global.asax ta có thể khai báo đối tượng như sau:
  <object name=”TênĐốiTượng” Scope=”Session”></object>
- Lấy dữ liệu từ biến Session phải xác định kiểu
  type obj = (type)Session["member"];

Ví dụ:
Page1.aspx

```
<html>
<head><title>Sử dụng Session</title>
</head>
<body>
        <%    Session[“userName”] = “Nguyễn Văn A”;
              Session[“Greeting”] = “Hello”;
              Response.redirect(“Page2.aspx”);
        %>
</body>
</html>
```

Page2.aspx

```
<html>
<head><title>Sử dụng Session</title></head>
<body>
        <% Response.write(Session[“Greeting”]  + “ “) +
<%=Session(“userName”)%>
</body>
</html>
```

3.2. Nội dung của Session

Các thành phần của Session, ngoại trừ được tạo ra bởi thẻ <object></object> đều được lưu trữ trong thuộc tính Contents. Tập hợp **Contents** có một thuộc tính **Count** để xác định số lượng các thành phần và các phương thức để huỷ bỏ dữ liệu.

Ví dụ: Tạo và lấy dữ liệu từ phiên làm việc

```
<%@ Page Language="C#" %>
<%@ Import Namespace="System.Collections" %>

<script runat="server">
 public void Page_Load(object sender, EventArgs args)
 {
  if (!IsPostBack)
  {
   if (Session["address"] = = null)
   {
    enterUserInfoPanel.Visible = true;
    userInfoPanel.Visible = false;
   }
   else
   {
    enterUserInfoPanel.Visible = false;
    userInfoPanel.Visible = true;

    SetLabels();
   }
  }
 }

 protected void SetLabels()
 {
  firstNameLabel.Text = Session["firstName"].ToString();
  lastNameLabel.Text = Session["lastName"].ToString();
  addressLabel.Text = Session["address"].ToString();
  cityLabel.Text = Session["city"].ToString();
  stateOrProvinceLabel.Text = Session["stateOrProvince"].ToString();
  zipCodeLabel.Text = Session["zipCode"].ToString();
  countryLabel.Text = Session["country"].ToString();
 }

 protected void EnterInfoButton_OnClick(object sender, EventArgs e)
 {
  Session["firstName"] = Server.HtmlEncode(firstNameTextBox.Text);
  Session["lastName"] = Server.HtmlEncode(lastNameTextBox.Text);
  Session["address"] = Server.HtmlEncode(addressTextBox.Text);
  Session["city"] = Server.HtmlEncode(cityTextBox.Text);
  Session["stateOrProvince"]                                    =
Server.HtmlEncode(stateOrProvinceTextBox.Text);
```

```
    Session["zipCode"] = Server.HtmlEncode(zipCodeTextBox.Text);
    Session["country"] = Server.HtmlEncode(countryTextBox.Text);

    enterUserInfoPanel.Visible = false;
    userInfoPanel.Visible = true;

    SetLabels();
  }

  protected void ChangeInfoButton_OnClick(object sender, EventArgs args)
  {
    enterUserInfoPanel.Visible = true;
    userInfoPanel.Visible = true;
  }
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
  <meta http-equiv="Content-Type" content="text/html" />
  <title>User Information</title>
</head>
<body>
  <form id="form1" runat="server">
    <h3>
      User information</h3>
    <asp:Label ID="Msg" ForeColor="maroon" runat="server" /><br />
    <asp:Panel ID="enterUserInfoPanel" runat="server">
      <table cellpadding="3" border="0">
        <tr>
          <td>
            First name:</td>
          <td>
            <asp:TextBox ID="firstNameTextBox" runat="server" /></td>
        </tr>
        <tr>
          <td>
            Last name:</td>
          <td>
            <asp:TextBox ID="lastNameTextBox" runat="server" /></td>
        </tr>
        <tr>
          <td>
            Address:</td>
```

```
        <td>
          <asp:TextBox ID="addressTextBox" runat="server" /></td>
        </tr>
        <tr>
          <td>
            City:</td>
          <td>
            <asp:TextBox ID="cityTextBox" runat="server" /></td>
        </tr>
        <tr>
          <td>
            State or Province:</td>
          <td>
            <asp:TextBox ID="stateOrProvinceTextBox" runat="server" /></td>
        </tr>
        <tr>
          <td>
            Zip Code/Postal Code:</td>
          <td>
            <asp:TextBox ID="zipCodeTextBox" runat="server" /></td>
        </tr>
        <tr>
          <td>
            Country:</td>
          <td>
            <asp:TextBox ID="countryTextBox" runat="server" /></td>
        </tr>
        <tr>
          <td>
             </td>
          <td>
            <asp:Button ID="enterInfoButton" runat="server" Text="Enter user
information" OnClick="EnterInfoButton_OnClick" /></td>
        </tr>
      </table>
    </asp:Panel>
    <asp:Panel ID="userInfoPanel" runat="server">
      <table cellpadding="3" border="0">
        <tr>
          <td>
            Name:</td>
          <td>
            <asp:Label ID="firstNameLabel" runat="server" />
```

```
          <asp:Label ID="lastNameLabel" runat="server" />
        </td>
      </tr>
      <tr>
        <td valign="top">
          address:</td>
        <td>
          <asp:Label ID="addressLabel" runat="server" /><br />
          <asp:Label ID="cityLabel" runat="server" />,
          <asp:Label ID="stateOrProvinceLabel" runat="server" />
          <asp:Label ID="zipCodeLabel" runat="server" /><br />
          <asp:Label ID="countryLabel" runat="server" />
        </td>
      </tr>
      <tr>
        <td>
           </td>
        <td>
          <asp:Button ID="changeInfoButton" runat="server" Text="Change user
information" OnClick="ChangeInfoButton_OnClick" /></td>
      </tr>
    </table>
  </asp:Panel>
 </form>
</body>
</html>
```

3.3. Xác định phiên làm việc

Thuộc tính **SessionID** được tạo ra cho mỗi phiên làm việc của người sử dụng và tồn tại trong suốt quá trình user truy nhập Web Site. Thuộc tính này được sử dụng để biết được user đang truy nhập vào trang Web nào.

Để xác định kết thúc một phiên làm việc ta có thể sử dụng thuộc tính **Timeout** hoặc phương thức **Abandon**. Mặc định thời gian chờ của các ứng dụng là 20 phút nếu không có request gì thì server sẽ kết thúc phiên làm việc và giải phóng bộ nhớ.

Ví dụ:

```
<html>
<head><title>Sử dụng SessionID</title></head>
<body>
<br/>The UserID is: <%=Session.SessionID %>
```

```
<% Session.Abandon(); %>
<br>The UserID is: <%=Session.SessionID %>
</body>
</html>
```

Trên thực tế, Session sử dụng cookies để thực hiện. Khi user yêu cầu một trang Web, server sẽ tạo ra một cookie trên trình duyệt để theo dõi Session. Khi Session kết thúc thì cookie cũng kết thúc.

Cookie được tạo ra cho mỗi user được gọi là ASPSESSIONID và giá trị của nó khác với giá trị của SessionID bởi thuật toán phức tạp để tránh việc đoán được ID của user. Thông qua ASPSESSIONID server nhận biết các biến session của từng user được lưu trữ.

Ví dụ: <%= Request.ServerVariables["HTTP_COOKIE"] %>

3.4. Làm việc với Cookie

Để bắt đầu thực hiện cookie được gửi từ Server tới Browser thông qua HTTP header qua đối tượng response và sau đó những requests được gửi tới từ Browser đều trả lại Cookie đó cho Server trong header.

Ví dụ:
Set-Cookie: UserName=Nguyen+Van+A; path='/'; expires = Tueday, 01-Jan-2004 GMT

Để thực hiện Cookie bằng ASP ta sử dụng tập hợp Cookies của đối tượng Response và Request.

Response.Cookies(TênCookie)[**.***attribute*] **=** *value*
Attribute có các giá trị sau:

- Domain: Cookie chỉ được gửi tới các request từ server chỉ định trong value
- Expires: Ngày mà Cookie sẽ hết tác dụng. Nếu không chỉ định thì Cookie sẽ hết tác dụng ngay sau khi Session kết thúc
- Path: Cookie chỉ được gửi cho request từ đường dẫn xác định. Nếu không thì đường dẫn của ứng dụng được sử dụng

Ví dụ:

```
<%
        Response.Cookies("UserName") = "Thái Thanh Tùng";
        Response.Cookies("UserName").Expires = "Jan 1, 2004";
        Response.Cookies("UserName").Path = "/examples";
```

```
        Response.Cookies("UserName").Path = "fithou.edu.vn";
%>
```

Để đọc các Cookie ta sử dụng đối tượng Request

Ví dụ:

```
<%
        foreach (string value in Request.Cookies) {
                Response.Write("<br/>" + value + "=" + Request.Cookies(value));
        }
%>
```

4. Đối tượng Application

4.1. Giới thiệu

ASPX được coi như những module chương trình thông thường. Khi chúng ta tạo ra một trang ASPX thì tương tự như ta tạo ra hàm hoặc thủ tục trong chương trình. Một tập hợp các trang ASPX sẽ tạo nên một chương trình hay ứng dụng.

Nhưng ứng dụng trong ASPX có đặc điểm khác biệt với chương trình thông thường

- Dữ liệu có thể sử dụng trong các trang riêng biệt và với từng user riêng biệt
- Ứng dụng có các sự kiện
- Có thể thiết lập server để tạo các ứng dụng khác nhau thực hiện trên các vùng nhớ khác nhau.

Một Website có thể bao gồm nhiều ứng dụng. Ví dụ như ta có thể xây dựng ứng dụng gồm các trang sử dụng cho khách hàng và ứng dụng dành riêng cho người quản trị. Ứng dụng được thiết lập bởi Web Server khi xác định thư mục gốc cho ứng dụng đó.

Xác định một ứng dụng:

- Khởi tạo IIS
- Chọn Default Web Site (Nếu chưa đổi tên)
- Chọn hoặc tạo thư mục ảo
- Nhắp chuột phải chọn Properties
- Virtual Directory hoặc Home Directory thay đổi các Settings của ứng dụng

Sau khi đã tạo xong các ứng dụng ta có thể tạo tệp Global.asax bao gồm các trình kịch bản sử dụng cho toàn bộ ứng dụng và đặt tại thư mục gốc của ứng dụng đó.

4.2. Các thuộc tính và phương thức

P/M/E	Tên	Chú thích
Property	Contents (Key)	Các thành phần dữ liệu có phạm vi Application trên toàn ứng dụng
Property	StaticObjects (Key)	Các thành phần dữ liệu có phạm vi Application được tạo bởi thẻ <object>
Method	Lock()	Khoá thành phần dữ liệu không cho các phiên khác truy cập
Method	Unlock()	Mở thành phần dữ liệu cho các phiên khác truy cập
Event	Onstart	Sự kiên khi Application được khởi tạo
Event	Onend	Sự kiện trước khi huỷ Application

5. Tệp Global.asa

ASAX viết tắt của Active Server Application .NET. Đây là tệp ứng dụng (quản lý trạng thái ứng dụng Web), tất cả các sự kiện của Application và Session được khai báo và kích hoạt từ tệp này.

Khi một ứng dụng Web được truy cập thì server sẽ kích hoạt ứng dụng, nạp tệp này vào bộ nhớ và khởi tạo đối tượng Application. Các hàm sự kiện định nghĩa trong tệp này sẽ tự động thực hiện. Khi ứng dụng hết kích hoạt (không còn phiên làm việc hoặc đóng server) sẽ huỷ Application.

Cấu trúc tệp Global.asax

Tương tự như ASPX nhưng chỉ gồm 2 phần

<%@ Dỉrective %> : Chỉ hỗ trợ 3 loại là Application, Assembly, Import

<script runat = “server”>

Code, Hàm sự kiện của Application, Session

</script>

Ví dụ: Global.asax

```
<%@ Application Language="C#" %>
<%@ Import Namespace="System.Diagnostics" %>

<script runat="server">
   public Guid AppId = Guid.NewGuid();

   public String TestMessage;

   public String GetAppDescription(String eventName)
   {
      StringBuilder builder = new StringBuilder();

      builder.AppendFormat("----------------------------{0}",Environment.NewLine);
      builder.AppendFormat("Event: {0}{1}", eventName,Environment.NewLine);
      builder.AppendFormat("Guid: {0}{1}", AppId, Environment.NewLine);
      builder.AppendFormat("Thread Id: {0}{1}",
        System.Threading.Thread.CurrentThread.ManagedThreadId,
        Environment.NewLine);
      builder.AppendFormat("Appdomain: {0}{1}",
        AppDomain.CurrentDomain.FriendlyName, Environment.NewLine);
      builder.AppendFormat("TestMessage: {0}{1}", TestMessage,
        Environment.NewLine);
      builder.Append((System.Threading.Thread.CurrentThread.IsThreadPoolThre
        ad ? "Pool Thread" : "No Thread") + Environment.NewLine);

      return builder.ToString();
   }

void Application_Start(object sender, EventArgs e)
   {
      TestMessage = "not null";
      Trace.WriteLine(GetAppDescription("Application_Start"));
   }

   void Application_End(object sender, EventArgs e)
   {
      Trace.WriteLine(GetAppDescription("Application_End"));
   }

   void Application_Error(object sender, EventArgs e)
```

```
    {
        Trace.WriteLine(GetAppDescription("Application_Error"));
    }

    void Application_BeginRequest(object sender, EventArgs e)
    {
        Trace.WriteLine(GetAppDescription("Application_BeginRequest"));
    }

    void Application_EndRequest(object sender, EventArgs e)
    {
        Trace.WriteLine(GetAppDescription("Application_EndRequest"));
    }
</script>
```

Các lưu ý:

- Global.asax mỗi ứng dụng chỉ gồm duy nhất 1 tệp đặt tại thư mục gốc
- Không sử dụng cặp thẻ <% %>, chỉ được phép sử dụng cặp thẻ <SCRIPT RUNAT=”Server “> </SCRIPT>.
- Global.asax được bảo vệ bởi Web Server, không nhận Request từ phía Client, và không trả lại Response cho Browser. Bởi vậy không được chứa mã dữ liệu HTML, không chứa mã lệnh trả dữ liệu, ví dụ Response.Write();
- Tất cả các dữ liệu và module khai báo trong tệp này phải có phạm vi Sesstion hoặc Application (ngoại trừ biến trung gian cục bộ), vì các trang Web khác không gọi được tệp này.

Cấu trúc Web.config

Web.config có cấu trúc là tệp XML, bao gồm các thẻ tags, mỗi thẻ tags có thể chứa tập hợp các thẻ con và các thuộc tính thiết lập cấu hình. Cấu trúc cấu hình cơ bản tệp web.config bao gồm thành phần gôc là thẻ <configuration> tag. Trong đó chưa các phần:

• **configSections:** Cho phép ta định nghĩa các thẻ của người dùng sử dụng <section> trong phần <configSection>. Trong ví dụ trên ta định nghĩa cặp thẻ <sampleCustomTag> và định nghĩa các cặp key, value với thẻ <add/>.

• **system.net:** Sử dụng nếu chúng ta cần thiết lập cấu hình cho các lớp dữ liệu Networking.
• **system.web:** Là thành phần chính thiết lập cấu hình cho các lớp dữ liệu ASP.NET. Các thiết lập quan trọng và thường sử dụng gồm:

- ***compilation:*** Xác định ngôn ngữ sử dụng, cho phép debug trong ứng dụng… Khi cho phép debug, ứng dụng sẽ chạy chậm hơn.
- ***customErrors:*** Khi được thiết lập ta sẽ xác định tệp hiển thị lỗi mặc định, trong trường hợp có lỗi trên ứng dụng không được xử lý sẽ hiển thị tệp này. Ta còn có thể thiết lập lỗi trang sử dụng thẻ con <errror/>.
- ***authentication:*** Xác định cơ chế xác thực định danh sử dụng trong ứng dụng và bao gồm cả cấu hình chi tiết cho từng loại xác thực định danh. Có thể sử dụng cơ chế xác thực định danh theo forms, windows, .net passport.
- ***authorization:*** Phần này định nghĩa cơ chế phân quyền cho người dùng truy cập toàn bộ, hoặc thư mục xác định trong ứng dụng. Ta có thể cho phép, không cho phép các users truy cập.
- ***trace:*** Ghi nhật ký theo dõi toàn bộ ứng dụng và có thể xem qua tiện ích trace.axd hoặc thông tin có thể được theo dõi cho từng trang.
- ***sessionState:*** Giao thức HTTP sử dụng với ứng dụng ASP.NET là giao thức không trạng thái. Tuy nhiên ta có thể sử dụng phiên làm việc để duy trì trạng thái.
- ***httpModules:*** Nếu ứng dụng có các httpModules xác định, ta có thể thiết lập cấu hình cho các modules đó (class và assembly).
- ***httpHandlers:*** Trong trường hợp có các trình xử lý handers của ứng dụng, ta định nghĩa trong phần này (handler, và assembly).
- ***httpRunTime:*** Thiết lập cấu hình thực thi ứng dụng, bao gồm xác định appRequestQueueLimit, số lượng tối đa requests được chờ server xử lý. Xác định thời gian kết thúc thực thi executionTimeout.

• **appSettings:** Dùng để định nghĩa các dữ liệu toàn cục trong ứng dụng, có thể truy cập dữ liệu toàn cục thông qua đối tượng

ConfigurationSettings.AppSettings[“key”];

• **Customized tags:** Là phần độc lập cho phép người dùng định nghĩa các tính năng riêng, và có thể truy cập bởi

ConfigurationSettings.GetConfig("sampleCustomTag")("sampleKey");

Ví dụ: web.config

```
<?xml version=’1.0” encoding=”utf-8”?>

<configuration>
<configSections>
        <section name="SampleCustomTag"
        type="System.Configuration.NameValueFileSectionHandler " />
</configSections>
```

```
<system.net>
        <!—Các lớp dữ liệu Networking -->
</system.net>
<system.web>
        <compilation defaultLanguage="c#" debug="false"/>
        <customErrors mode="On" defaultRedirect="defaultCusErrPage.htm"/>
        <authentication mode=”Forms”>
                <forms loginUrl = "frmSampleLogin.aspx" timeout="20"/>
        </authentication>
        <authorization>
                <allow users="User1, User2" />
                <deny users=”*” />
        </authorization>
        <trace enabled="true" requestLimit="20" />
        <sessionState mode="SQLServer" stateConnectionString=”tcpip
        =127.0.0.1:8040”
                sqlConnectionString="data source=127.0.0.1; Integrated
        Security=SSPI
                Trusted_Connection=yes”; cookieless="true" timeout="40" >
        </sessionState>
        <httpModules>
                <add type="sampleClass, sampleAssembly" name="sampleModule"
        />
                <remove name="modulename"/>
        </httpModules>
        <httpHandlers>
                <add verb="*" path="sampleFolder/*.aspx"
                type="sampleHttpHandler,sampleAssembly" />
        </httpHandlers>
        <httpRuntime appRequestQueueLimit="200" executionTimeout="500" />
</system.web>
<appSettings>
        <add key="DBConnString" value="server=127.0.0.1;uid=usr;pwd=pwd;
        database=DBPerson" />
</appSettings >
<sampleCustomTag >
        <add key="sampleKey" value="sample key value" />
</sampleCustomTag >
</configuration>
```

Các lưu ý:

- Mỗi ứng dụng có thể có nhiều tệp web.config đặt tại các cấp thư mục khác nhau và có tác dụng với cây thư mục được đặt
- Nếu không tạo tệp web.config sẽ lấy tệp mặc định của .NET. Ví dụ c:\Windows\Microsoft.NET\Framework\v4.0.30319\web.config