CS-32: Programming with ASP.NET

UNIT – 2

State Management

Dharmesh Kapadiya
Dhruvita Savaliya

TOPICS :

- What is State?
- Why is it required in Asp.Net?
- Client Side State Management
- Server Side State Management
- Various State Management Techniques
  - View State
  - Query String
  - Cookie
  - Session State
  - Application State

Stateless ?

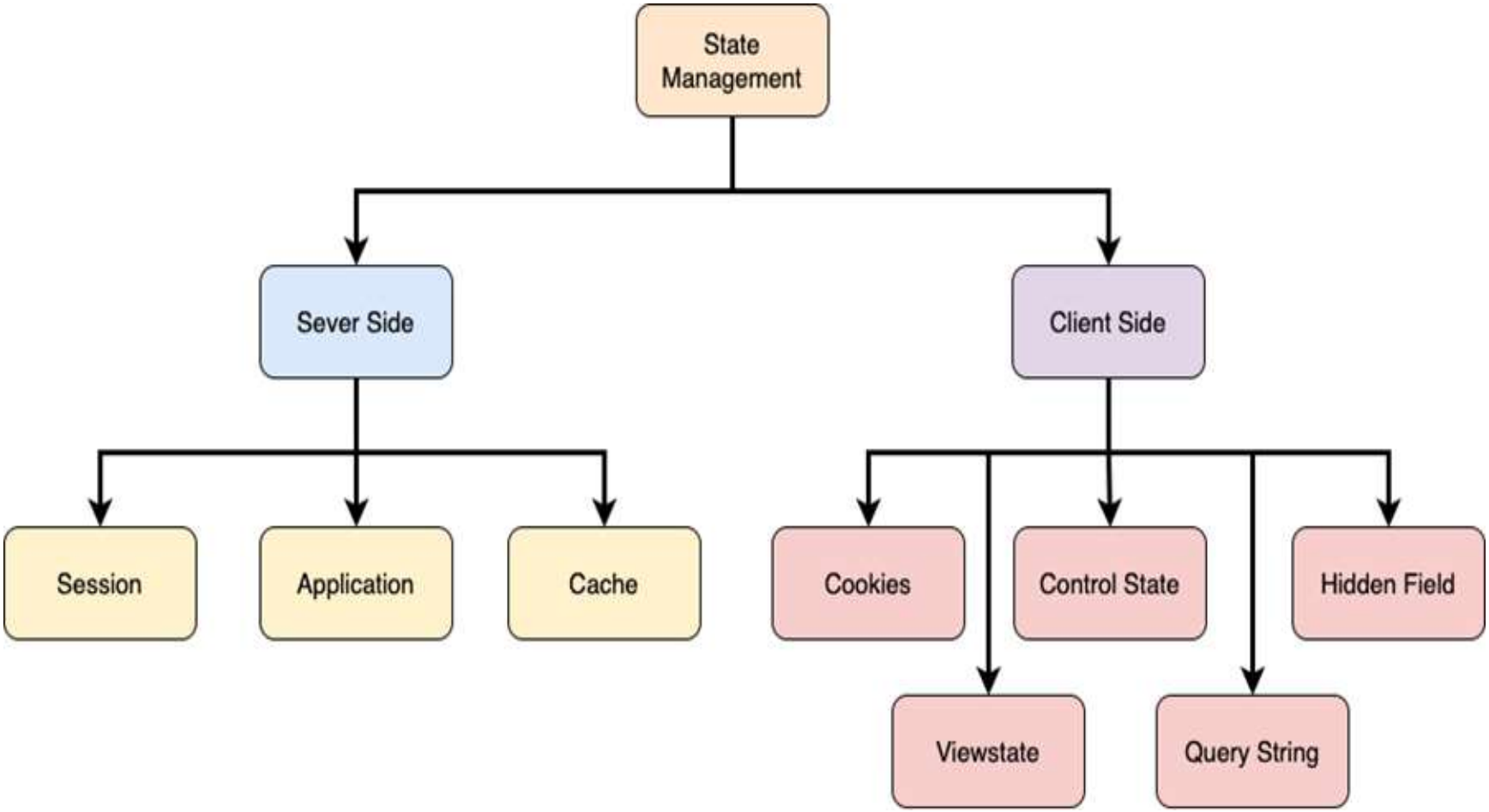
- Stateless means whenever we visit a website, our browser communicates with the respective server depending on our requested functionality or the request. The browser communicates with the respective server using HTTP.
- When we visit the same website again, HTTPs doesn't remember what website or URL we visited, or we can say it doesn't hold the state of a previous website that we visited before closing our browser, that is called stateless.
- HTTP is a stateless protocol. It just cleans up or we can say removes all the resources/references that were serving a specific request in the past. These resources can be:
  1. Objects
  2. Allocated Memory
  3. Sessions ID's
  4. Some URL info and so on

What is State Management ?

- State Management simply means **Saving Something for future use.**
- You all are aware that Http is a **Stateless protocol.**
- Stateless protocol means you can not save any information on Http.
- In other words, when you transfer from one web page to another web page, you can't store any information on Http.
- Even though you can pass information from one page to another page. **How ???** Answer is we pass information from one page to another page by using State Management.
- Let us understand the above words with an easy example :
- While developing web applications you may want to transfer some information from one web page to another web page.

- **For example, from the Login page you want to transfer "UserName" to** another page, so that you can write UserName on the welcome page.
- This solution can be done using State Management.
- ASP.NET State management is to preserve state control and objects in an application because ASP.NET web applications are stateless.
- A new instance of the Web page class is created each time the page is posted to the server.
- If a user enters information into a web application, that information would be lost in the round trip from the browser.
- **In a single line, State management maintains and stores the information of any user till the end of the user session.**

Types of State Management :

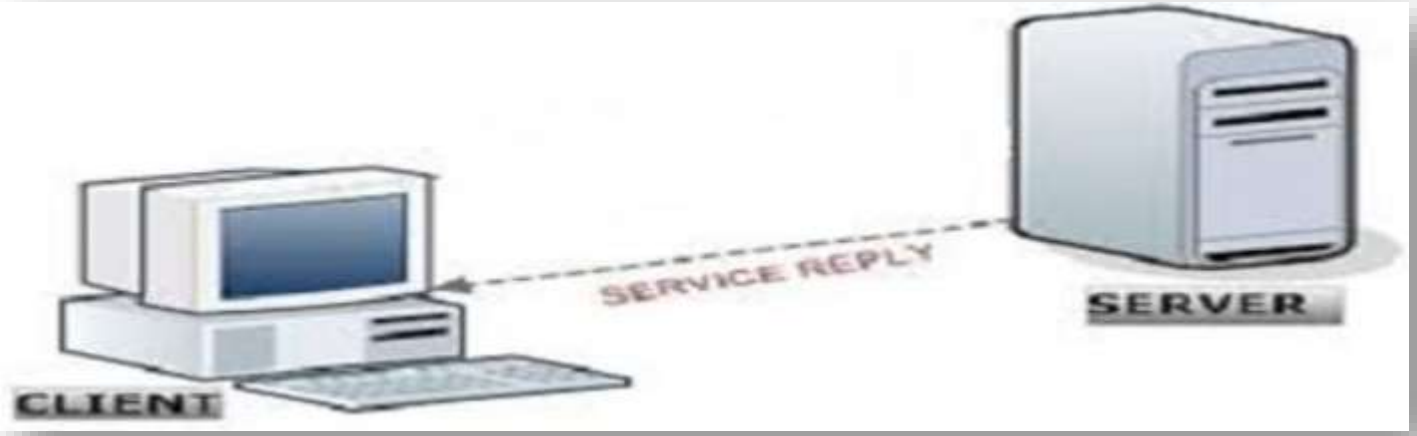


Client Side State Management :

- Client-Side State Management Techniques - To maintain the state of values on the client's machine, we use the client-side state management techniques.
- Whenever we use Client-Side State Management, the state related information will directly get stored on the client-side.
  - Cookies
  - Viewstate
  - Control state
  - Query String
  - Hidden Field

Server Side State Management :

- Server-Side State Management - To maintain the state of values on the server's machine, we use the server-side state management techniques.
- It is another way which ASP.NET provides to store the user's specific information or the state of the application on the server machine.
- It completely makes use of server resources (the server's memory) to store information.
  - Session
  - Application
  - Cache



Control State :

- The purpose of the control state repository is to cache data necessary for a control to properly function.
- ControlState is essentially a private ViewState for your control only, and it is not affected when ViewState is turned off.
- ControlState is used to store small amounts of critical information.
- Heavy usage of ControlState can impact the performance of application because it involves serialization and deserialization for its functioning.
- There are two methods you have to implement in your custom control.

1. Load Control State
2. Save Control State

View State :

- This is the Client Side Method.
- Viewstate is a very useful client side property.
- It is used for page level state management.
- Viewstate stores any type of data and is used for sending and receiving information.
- ViewState is used to store user data on a page at the time of post back to the web page.
- The view state is the state of the page and all its controls.
- It is automatically maintained across posts by the ASP.NET framework.
- ViewState does not hold the controls, it holds the values of controls.
- It does not restore the value to control after page post back.
- ViewState can hold the value on a single web page, if we go to another page using response.redirect then ViewState will be null.
- The property **EnableViewState="false"** is used in code when it is required to disable view state

- **How to Enable and Disable View State :**
- You can enable and disable View State for a single control as well as at the page level also. To turn off View State for a single control, set the EnableViewState property of that control to false.
- TextBox1.**EnableViewState=false**;
- To turn off the View State for an entire page, we need to set EnableViewState to false of the page directive as shown below:
- <%Page Language="C#" **EnableViewState**="false“ %>
- For enabling the same, you need to use the same property just set it to "True".

File.aspx :

```
<form id="form1" runat="server">
   <div>
        <asp:TextBox ID="txt1" runat="server"> </asp:TextBox>
        <asp:TextBox ID="txt2" runat="server"> </asp:TextBox>
        <asp:Button ID="sub" runat="server" Text="Submit"
                OnClick="sub_Click" />
        <asp:Button ID="res" runat="server" Text="Restore"
                OnClick="res_Click" />
   </div>
   </form>
```

File.aspx.cs :

```
protected void sub_Click(object sender, EventArgs e)
   {
        ViewState["one"]=txt1.Text;
        ViewState["two"] = txt2.Text;
        txt1.Text = string.Empty;
        txt2.Text = string.Empty;
   }
protected void res_Click(object sender, EventArgs e)
   {
      if (ViewState["one"] != null)       {
                txt1.Text = ViewState["one"].ToString();
      }
      if (ViewState["two"] != null)       {
                txt2.Text = ViewState["two"].ToString();
      }
   }
```

Query String :

- A query string is a collection of characters input to a computer or web browser.
- A Query String is helpful when we want to transfer a value from one page to another.
- When we need to pass content between the HTML pages or aspx Web Forms in the context of ASP.NET, a Query String is very easy to use and the Query String follows a separating character, usually a Question Mark (?).
- It is basically used for identifying data appearing after this separating symbol.
- A Query String Collection is used to retrieve the variable values in the HTTP query string.

- If we want to transfer a large amount of data then we can't use the Request.QueryString.
- Query Strings are also generated by form submission or can be used by a user typing a query into the address bar of the browsers.
- It enable us to retrieve the QUERY_STRING variable by name.
- When we use parameters with Request.QueryString, the server parses the parameters sent to the request and returns the effective or specified data.
- File_name.aspx**?**par1**=**value1**&**par2**=**value2**&**par3**=**value3

Example of Send the Data to the URL : First File

First.aspx

```
<form id="form1" runat="server">
  <div>
    <asp:TextBox ID="txt1"
  runat="server"></asp:TextBox>
    <asp:TextBox ID="txt2"
  runat="server"></asp:TextBox>
    <asp:Button ID="btn"
  runat="server" OnClick="btn_Click"
  Text="Send data"/>
  </div>
  </form>
```

First.aspx.cs

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

    Response.Redirect("second.aspx?nm=
    "+txt1.Text+"&pwd="+txt2.Text);
  }
```

Example of get Data from URL : Second File

second.aspx

```
<form id="form1" runat="server">
  <div>
    <asp:Label ID="l1"
  runat="server"></asp:Label><br />
    <asp:Label ID="l2"
  runat="server"></asp:Label>
  </div>
  </form>
```

second.aspx.cs

```
protected void Page_Load(object
    sender, EventArgs e)
{
 l1.Text = Request.QueryString[“nm"];
 l2.Text = Request.QueryString["pwd"];
}
```

Hidden Form Field :

- This is client side methods for state management.
- You can save the state of the data by using Hidden Form Fields.
- A hidden field stores text data with the HTML <input type=”hidden”> or by using <asp:HiddenField>.
- A hidden form field is not visible in the browser but you can set its properties like other controls.
- When page is posted to the server ,the content of the hidden form field is sent in the HTTP Form collection along with the values of other controls.
- In order to make available hidden form field value during page processing ,you must submit the page using HTTP post method.
- That means you can not use hidden form field,if the page is processed in response to a link or HTTP GET method.

Property	Description
Value	Sets the value which is stored in HiddenField.
Visible	Sets Hidden Form Field is visible or not.

Example :

```
<asp:HiddenField ID="HiddenField1" runat="Server"  Value="This
   is the Value of Hidden field" />
protected void btn2_Click(object sender, EventArgs e)
   {
      txt_hiden_field.Value = txt.Text;
      Response.Redirect("welcome.aspx?par1=" +
   HiddenField1.Value);
   }
```

This data is not visible in the page.

We can also pass this data like a query string.

Cookie :

- This is client side methods for state management.
- ASP.NET Cookie is a small bit of text that is used to store user-specific information.
- This information can be read by the web application whenever user visits the site.
- When a user requests for a web page, web server sends not just a page, but also a cookie containing the date and time.
- This cookie stores in a folder on the user's hard disk.
- When the user requests for the web page again, browser looks on the hard drive for the cookie associated with the web page.
- Browser stores separate cookie for each different sites user visited.
- ***Note:*** *The Cookie is limited to small size and can be used to store only 4 KB (4096 Bytes) text.*

- There are two ways to store cookies in ASP.NET application.
  1. Cookies collection
  2. HttpCookie
- **Persist Cookie** - A cookie that has not expired is called a Persist Cookie.
- **Non-Persist Cookie** - A cookie has expired, called a Non-Persist Cookie.
- **Cookie's Common Properties**
  - **Domain:** We use it to associate cookies to the domain.
  - **Secure:** We can enable the secure cookie to set valid (HTTP's).
  - **Value:** We can manipulate individual cookies.
  - **Values:** We can influence cookies with key/value pairs.
  - **Expires:** We use it to set expiration dates for the cookies.

- **Advantages of Cookie**
  - Its clear text so users can able to read it.
  - It can store users' preference information on the client machine.
  - It's an easy way to maintain.
  - Fast accessing.
- **Disadvantages of Cookie**
  - If the user clears cookie information, we can't get it back.
  - No security.
  - Each request will have cookie information with the page.
- **HttpCookie :**
- We can add Cookie either to Cookies collection or by creating instance of HttpCookie class. both work same except that HttpCookie require Cookie name as part of the constructor.
- **Example :**

```
HttpCookie cokie = new HttpCookie("student"); //creating object
cokie.Value = "Rahul Kumar";  //set value to the cookie
 Response.Cookies.Add(cokie);//add cookie
Response.Write(Response.Cookies["student"].Value);//printing of cookie
```

Example :

File.aspx :

```
<asp:TextBox ID="t1" runat="server"></asp:TextBox>

<asp:TextBox ID="t2" runat="server"></asp:TextBox>

<asp:Button ID="create" runat="server" Text="create cookie"
   OnClick="create_Click" />
<asp:Button ID="show" runat="server" Text="show cookie"
   OnClick="show_Click“ />
```

File.aspx .cs:

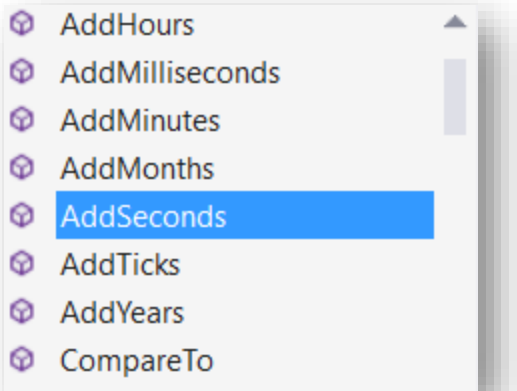
```
//Create cookies and deleting cookie :
protected void create_Click(object sender, EventArgs e)
   {
Response.Cookies["name"].Value = t1.Text;
Response.Cookies["name"].Expires =
    DateTime.Now.AddSeconds(50);

Response.Cookies["surname"].Value = t2.Text;
Response.Cookies["surname"].Expires = DateTime.Now.AddSeconds(50);
                    //Creating cookie collection
      //Response.Cookies["name"]["abc"] = t1.Text;
   }
```

Expires Cookie DateTime's methos :

We can also delete cookie permanently like :

```
Response.Cookies["name"].Expires =
    DateTime.Now.AddSeconds(-1);
```



```
//Display cookie
protected void show_Click(object sender, EventArgs e)
   {
      if (Request.Cookies["name"] != null)
      {
         Response.Write(Request.Cookies["name"].Value);
      }
      if (Request.Cookies["surname"] != null)
      {
         Response.Write(Request.Cookies["surname"].Value);
      }
   }
```

Session State Management :

- This is Server Side Method.
- Session state is a period of time to visit a website for a particular user.
- Session can store the client data on a server.
- Session is a best state management feature to store the client data on a server separately for each user.
  - **Session can store value across multiple pages of a website.**
  - **Viewstate can store value on a single page.**
- In simple words we can say, At the same time more than one user login to the system, all user identification names store separately until they logout.
- session can store username or any unique identification of user for a login period time.
- Session value can be accessible from all pages from the website.
- We can store some information in a session in one page and can access the same information on the rest of all pages by using session.

- When you login to any website with username and password.
- Your username shows on all other pages.The username will be stored in session and on page we access session value to display username.
- **Syntax :**

  Session["session_name"] = "session value";
- **Session Timeout in asp.net :**
- Session is a length/period of time that a particular browser instance spends accessing a website.
- The Session object is used to persist data across a user session during the user's visit to a website.
- In asp.net **by default session timeout = 20 minutes, but in some cases** we need to change session time increment or decrement by changing **web.config file setting.**
- We can also set manually by write c# code at code behind .aspx page in asp.net.

- **Set Session Timeout :**
- Session Timeout using web.config file
- Session timeout in each page using C# code
- Session timeout using IIS Server for website
- Using **Web.config** file :
- Set session timeout = 3 minute using web.config file.

```
<system.web>
        <sessionState timeout="3"></sessionState>
</system.web>
```

Properties	Description
SessionID	The unique session identifier.
Item(name)	The value of the session state item with the specified name. This is the default property of the HttpSessionState class.
Count	The number of items in the session state collection.
TimeOut	Gets and sets the amount of time, in minutes, allowed between requests before the session-state provider terminates the session.

Methods	Description
Add(name, value)	Adds an item to the session state collection.
Clear	Removes all the items from session state collection.
Remove(name)	Removes the specified item from the session state collection.
RemoveAll	Removes all keys and values from the session-state collection.
RemoveAt	Deletes an item at a specified index from the session-state collection.

- **Session.Clear( ) and Session.Abandon( )**
- We learn about session.clear and session.abandon methods in detail.
- **Session.Abandon( )**
- Session.Abandon( ) method used to destroys the session by generating a Session_End event
- Session.Abandon( ) method used to destroy the current session.
- It is the same thing to dispose of an object.
- It triggers the Session_End event Global.asax.
- Session.Abandon( ) method clear occupied memory location.
- **Session.Clear( ) :**
- Session.Clear( ) used to just clear all values of session without destroying it.
- It is the same thing as assigning null value to a variable.
- It doesn't trigger Session_End in Global.asax.
- The Session.Clear( ) method just clears session values.

loginpage.aspx :

```
<div>
  <div>
    <asp:Label ID="lblname" runat="server"
   Text="UserName"></asp:Label>
    <asp:TextBox ID="txtname"
   runat="server"></asp:TextBox>
  </div>
   <div>
     <asp:Label ID="lblpass" runat="server"
   Text="Password"></asp:Label>
     <asp:TextBox ID="txtpass"
   runat="server"></asp:TextBox>
  </div>
  <div>
    <asp:Button ID="btnlogin"
   runat="server" Text="Login"
   OnClick="btnlogin_Click" />
  </div>
   <div>
    <asp:Label ID="lblerror"
   runat="server"></asp:Label>
   <div>
</div>
```

loginpage.aspx.cs :

```
protected void btnlogin_Click(object sender,
    EventArgs e)
  {
    if (txtname.Text == "admin" && txtpass.Text
  == "admin@123")
    {
      Session["username"] = txtname.Text;

  Response.Redirect("welcomepage.aspx");
    }
    else
    {
      lblerror.Text = "wrong id and password";
    }
  }
```

welcomepage.aspx :

```
<div>
   <asp:Label ID="lbl1" runat="server"
   ></asp:Label>
 </div>

<div>
    <asp:Button ID="btnlogout"
   runat="server" Text="Log Out"
   OnClick="btnlogout_Click"/>
</div>
```

welcomepage.aspx.cs :

```
protected void Page_Load(object sender,
    EventArgs e)
   {
      if (Session["username"] == null)
      {
Response.Redirect("loginpage.aspx");
      }
      else
      {
         lbl1.Text =
    Session["username"].ToString();
      }
   }
   protected void btnlogout_Click(object
    sender, EventArgs e)
   {
      Response.Redirect("logout.aspx");
   }
```

logout.aspx :

```
protected void Page_Load(object sender, EventArgs e)
   {
      Session.Abandon();
      Session.Clear();
      Session.RemoveAll();
      Response.Redirect("loginpage.aspx");
   }
```

Application State :

- This is Server Side Method.
- Application Level State Management is used to maintain the state of all the users accessing the webforms present within the website.
- The value assigned for an application is considered an object.
- Application object will not have any default expiration period.
- Whenever the webserver has been restarted or stopped then the information maintained by the application object will be lost.
- If any data is stored on the application object then that information will be shared upon all the users accessing the webserver.
- Since the information is shared among all the users, it is advisable to lock and unlock the application object as per requirement.
- Technically the data is shared amongst users by aHTTPApplcationState class and the data can be stored here in a key/value pair.
- It can also be accessed using the application property of the HTTPContext class.

- **Syntax of Application State**
- Store information in application state :
  Application["name"] ="tybca";
- Retrieve information from application state :
  string str = Application["name"].ToString();

- **Global Application Class(Global.asax) :**

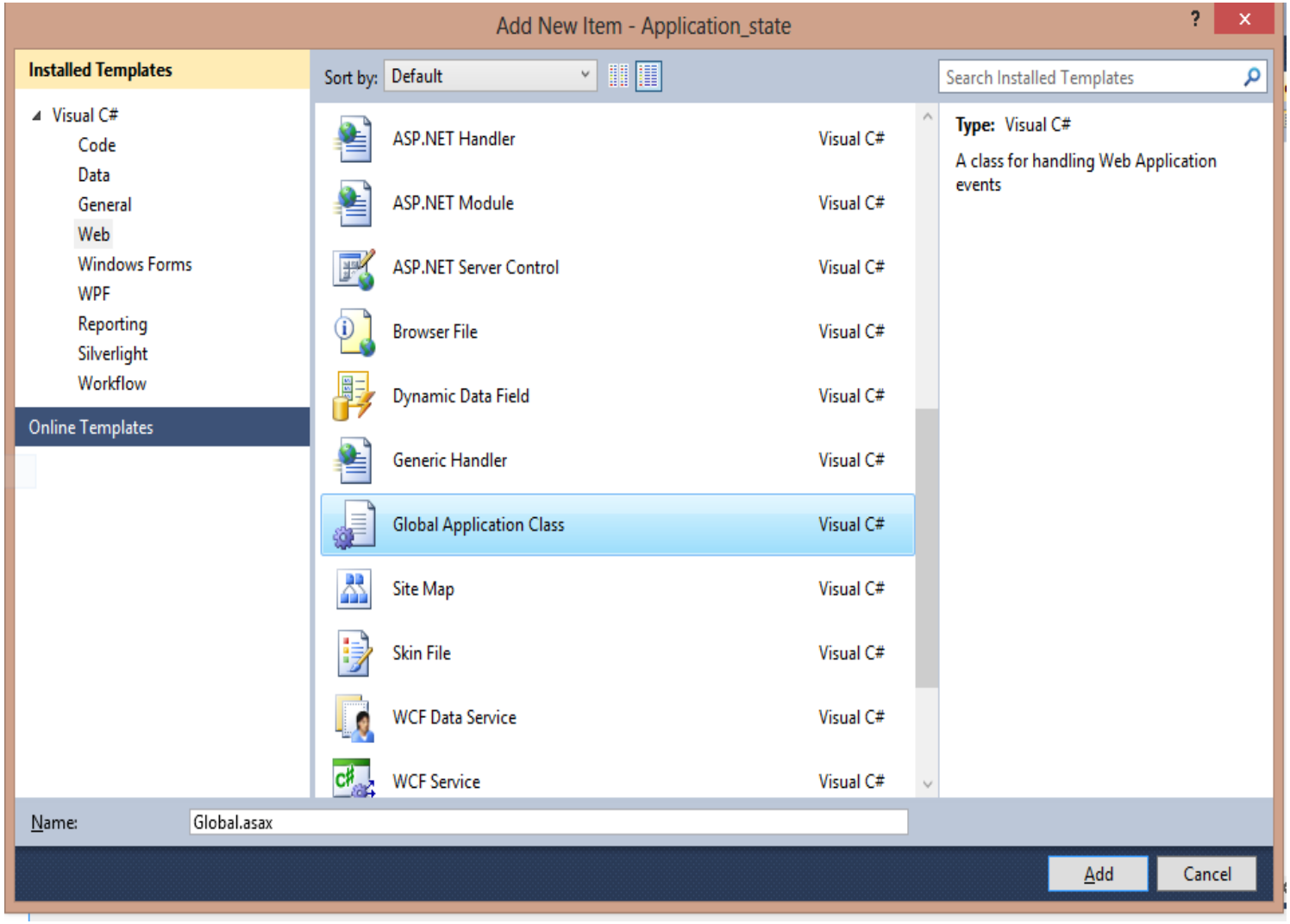
- It is a Class that consists of event handlers which executes the code implicitly whenever a relevant task has been performed on the web server.
- It always exists in the root level. Events are one of the following of the 2 types in the Global application:

1. Events that will be raised on a certain condition.
2. Events that will be raised on every request.

- **Global Application Class has following methods :**

1. **Application_Start() :** This method is invoked initially when first application domain is created.
2. **Session_Start() :** This method is called every time a session is start.
3. **Application_BeginRequest() :** After an application has started the first method Application_BeginRequest() is executed for every user.
4. **Application_AuthenticateRequest() :** It checks to determine whether or not the user is valid.
5. **Application_Error() :** Whenever an unhandled exception occurs then this event will be called.
6. **Session_End() :** When a user session is ended and all the data related to a specific user is cleared then the Session_End() event is called.
7. **Application_End() :** This method is called before the application ends. This can take place if IIS is restarted or the application domain is changing.
8. **Application_Disposed() :** This event is called after the application will be shut down and the .NET GC is about to reclaim the memory it occupies. Although this is very late to perform any clean-up but we can use it for safety purposes.

Right Click on your project → Add → Add New Item → Global Application Class



File.aspx :

```
<form id="form1" runat="server">
   <div>
      <asp:Label ID="Label1"
    runat="server" Text="">
    </asp:Label>
   </div>
   </form>
```

File.aspx.cs :

```
protected void Page_Load(object
    sender, EventArgs e)
{
    Label1.Text =
    Application["user"].ToString();
}
```

Global.asax file

```
<%@ Application Language="C#" %>
<script runat="server">
    void Application_Start(object sender,
     EventArgs e)
    {
        // Code that runs on application startup
       Application["user"] = 0;
    }
    void Application_End(object sender,
     EventArgs e)
    {
        //  Code that runs on application shutdown
    }
    void Application_Error(object sender,
     EventArgs e)
    {       // Code that runs when an unhandled
     error occurs
    }
    void Session_Start(object sender,
     EventArgs e)
    {
   // Code that runs when a new session is started
        Application.Lock();
       Application["user"] =
     (int)Application["user"] + 1;
       Application.UnLock();
    }
    void Session_End(object sender,
     EventArgs e)
    {       // Code that runs when a session ends.
       Application.Lock();
       Application["user"] =
     (int)Application["user"] - 1;
       Application.UnLock();
    }
</script>
```

- **Lock Method :**
- The Lock method prevents other users from modifying the variables in the Application object (used to ensure that only one client at a time can modify the Application variables).
- **Unlock Method :**
- The Unlock method enables other users to modify the variables stored in the Application object (after it has been locked using the Lock method).
- **Syntax :**

```
Application.Lock();
Application.Unlock();
```