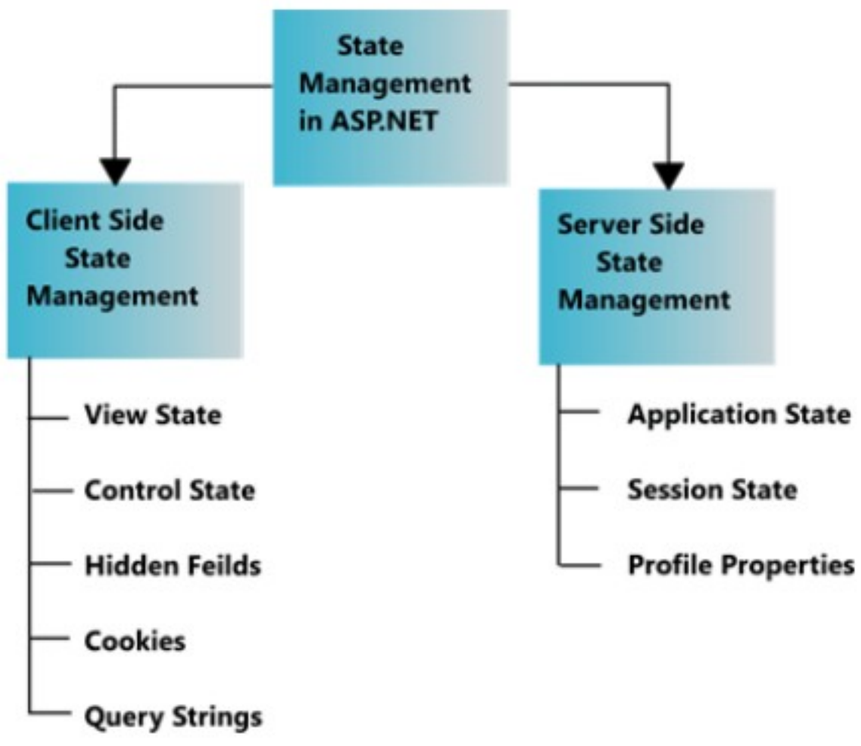
STATE MANAGEMENT

- Web is Stateless. It means a new instance of the web page class is re-created each time the page is posted to the server.
- As we all know HTTP is a stateless protocol, it can't hold the client information on page.
- As for example: if we enter a text and client presses submit button, text does not appear after post back, only because of page is recreated on its round trip.
- All the controls of the Web Page are created and after the round trip the server destroys all the instances. So, to retain the values of the controls we use state management techniques.

- As given in above pages, page is recreated before it's comes to clients and happened for each and every request. So, it is a big issue to maintain the state of the page and information for a web application. That is the reason to start concept of State Management.
- Whenever we visit a website, our browser communicates with the respective server depending on our request. The browser communicates with the respective server using the HTTP.
- But after that response, what's next or what will happen when we visit that website again after closing our web browser?
- It doesn't hold the state of a previous website that we visited before closing our browser that is called stateless.
- If we have to track the users' information between page visits and even on multiple visits of the same page, then we need to use the State management techniques provided by ASP.NET.
- State management is the process by which ASP.NET let the developers maintain state and page information over multiple requests for the same or different pages.



VIEW STATE*

- View State is client-side state management technique. In Asp.Net, the View State property is used to store the page-level state of a web page, the View State property the stores information of a single user, as long as the user is working with the page.
- The View State property is used when a form is submitted and presented second time to a user, as it retains the information entered in the form's controls for the first time.
- The VIEWSTATE property at each page is initialized and stored in View State.
- The string is assigned to the Value attribute of VIEWSTATE field and is sent as a part of the web page.

ViewState.aspx

```
<table><tr><td><asp:Label ID="lblUserName" runat="server" Text="User
Name"></asp:Label></td>
<td><asp:TextBox ID="txtUserName" runat="server"></asp:TextBox></td></tr>
<tr><td><asp:Label ID="lblPassword" runat="server"
Text="Password"></asp:Label></td>
     <td><asp:TextBox ID="txtPassword"
     runat="server"></asp:TextBox></td></tr>
<tr><td><asp:Button ID="btnSubmit" runat="server" Text="Submit" /></td>
<td><asp:Button ID="btnRestore" runat="server" Text="Restore" /></td></tr>
</table>
```

ViewState.aspx.vb

```
Public username As String
Public password As String

Protected Sub btnSubmit_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnSubmit.Click
        ViewState("username") = txtUserName.Text
        ViewState("password") = txtPassword.Text
        txtUserName.Text = ""
        txtPassword.Text = ""
End Sub

Protected Sub btnRestore_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnRestore.Click
        txtUserName.Text = ViewState("username").ToString
        txtPassword.Text = ViewState("password").ToString
End Sub
```

- **Disadvantages of VIEW STATE:**

1. **Security Risk**: The Information of View State can be seen in the page output source directly. You can manually encrypt and decrypt the contents of a Hidden Field, but it requires extra coding.

2. **Performance:** Performance is not good if we use a large amount of data because View State is stored in the page itself and storing a large value can cause the page to be slow.
3. **Device limitation**: Mobile Devices might not have the memory capacity to store a large amount of View State data.
4. It can store values for the same page only.

QUERYSTRING

- Query String is client-side state management technique. A **QueryString** 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 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 (?).
- If we want to transfer a large amount of data (more than 100kb) then we can't use the **Request.QueryString**.
- Multiple query strings can be specified in the URL by separating them by an ampersand ('&').
- If you need to send a variable which contains & such as "DARSHAN & COMPUTER", then we have to modify the code little bit which is shown in the below example.

QueryString.aspx

```
<div><b>QueryString Example</b></div><br />
<table><tr><td><b>UserId:</b></td><td><asp:TextBox ID="txtUserId"
runat="server"/></td></tr>
     <tr><td><b>UserName:</b></td><td><asp:TextBox ID="txtUserName"
runat="server"/></td></tr>
     <tr><td></td><td><asp:Button ID="btnSend" Text="Send Values"
runat="server"/></td></tr></table>
```

QueryString.aspx.vb

```
Protected Sub btnSend_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnSend.Click
     Dim susername As String = txtUserName.Text.Replace("&", "%26")
       Dim suserid As String = txtUserId.Text.Replace("&", "%26")
       Response.Redirect("Default.aspx?UserId=" + susername + "&UserName="
+ suserid)
End Sub
```

Default.aspx

```
<div><b>QueryString parameter Values in Default.aspx Page</b></div><br />
<div><b>UserId:</b><asp:Label ID="lblUserId" runat="server"/></div><br />
<div><b>UserName:</b><asp:Label ID="lblUserName" runat="server"/></div>
```

Default.aspx.vb

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        If Not Page.IsPostBack Then
            lblUserId.Text = Request.QueryString("UserId")
            lblUserName.Text = Request.QueryString("UserName")
        End If
End Sub
```

CONTROL STATE:

- Control State is client-side state management technique.
- Whenever we develop a custom control and want to preserve some information, we can use view state but suppose view state is disabled explicitly by the user, the control will not work as expected.
- For expected results for the control, we have to use Control State property. Control state is separate from view state.
- If you create a custom control that requires view state to work properly, you should use control state to ensure other developers don't break your control by disabling view state.

Default.aspx

```
<table><tr><td>
<asp:Label ID="lblname" runat="server" Text="Name"></asp:Label>
</td><td><asp:TextBox ID="txtname" runat="server"></asp:TextBox></td></tr>
<tr><td><asp:Button ID="btnsubmit" Text="Submit" runat="server" />
</td></tr></table>
```

Default.aspx.vb

```
Protected Sub btnsubmit_Click(sender As Object, e As EventArgs) Handles
btnsubmit.Click
        Server.Transfer("WebForm.aspx")
    End Sub
```

WebForm.aspx

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

WebForm.aspx.vb

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Dim msg As TextBox = CType(PreviousPage.FindControl("txtname"),
TextBox)
        lblmsg.Text = "Enterd Name is: " & msg.Text
End Sub
```

HIDDEN FIELDS:

- Hidden Field is client-side state management technique. The HiddenField control provides you with a way to store information in the page without displaying it.
- This control enables a developer to store a non-displayed value. You can use the Hidden Field control to store state values.
- Note, that because the value of a Hidden Field is rendered to the client browser, it is not suitable for storing security-sensitive values.

- **Default.aspx**

```
<table><tr><asp:HiddenField ID="hdn" runat="server" />
<td><asp:TextBox ID="txtname" runat="server"></asp:TextBox></td></tr>
<tr><td><asp:Button ID="btnsubmit" runat="server" Text="Submit" />
</td><td><asp:Label ID="lblname" runat="server" Text=""></asp:Label>
</td></tr></table>
```

- **Default.aspx.vb**

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        hdn.Value = txtname.Text
 End Sub
Protected Sub btnsubmit_Click(sender As Object, e As EventArgs) Handles
btnsubmit.Click
        lblname.Text = hdn.Value
    End Sub
```

COOKIES

- Cookie is client-side state management technique.
- Cookies is a State Management Technique that can store the values of control after a post-back.
- Cookies is a small piece of information stored on the client machine which was sent by web server.
- It is used to store user preference information like Username, Password, City and Phone No etc. on client machines.
- **Types of Cookies**

1. Persistent Cookie: A cookie does not have expiry time which is called as Persistent Cookie
2. Non-Persistent Cookie: A cookie has expired time which is called as Non-Persistent Cookie

- It is easy to create cookie in asp.net with the help of Response object or HttpCookie.
- Following is the example of how we can create the cookie.

1. **Response object Type:**

```
Response.Cookies("UserName").Value = txtUserName.Text
```

2. **HttpCookie Type:**

```
Dim nameCookie As New HttpCookie("UserName")
nameCookie.Values("UserName") = txtUserName.Text
```

CookiesDemo.aspx

```
<asp:Label ID="lblCookie" runat="server"></asp:Label>
<asp:TextBox ID="txtUserName" runat="server"></asp:TextBox>
<asp:Button ID="btnCreateCookie" runat="server" Text="Create Cookie" /><br />
<asp:Button ID="btnRetrieve" runat="server" Text="Retrieve Cookie" />
```

CookiesDemo.aspx.vb

```
Protected Sub btnCreateCookie_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnCreateCookie.Click
        Response.Cookies("UserName").Value = txtUserName.Text
        Response.Cookies("UserName").Expires = DateTime.Now.AddSeconds(30)
        lblCookie.Text = "Cookies Created"
        txtUserName.Text = ""
End Sub
Protected Sub btnRetrieve_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnRetrieve.Click
        If (Request.Cookies("UserName") Is Nothing) Then
            lblCookie.Text = "There is no cookie"
        Else
            lblCookie.Text = "Your cookie contains:" &
Request.Cookies("UserName").Value
        End IfEnd Sub
```

- **IMPORTANT PROPERTIES OF COOKIES:**

Name	Description
Domain	Using these properties, we can set the domain of the cookie.
Expires	This property sets the Expiration time of the cookies.
Value	Contains the value of the cookies.

HOW CAN WE CONTROL COOKIES SCOPE?

- By default, all cookies for a site are stored together on the client and all cookies are sent to the server with any request to that site. There are following 2 technique by which you can apply scope to cookies' scope

1. LIMIT THE SCOPE OF COOKIES TO A FOLDER ON THE SERVER

- In practical terms allows you to limit cookies to an application on the site.

```
Dim appCookie As New HttpCookie("AppCookie")
appCookie.Value = "written " & Now.ToString
appCookie.Expires = Now.AddDays(1)
appCookie.Path = "/Application1"
Response.Cookies.Add(appCookie)
```

- The effect will be that the cookie is available only to pages in the Application1 folder.

- For example, if your site is called www.dashan.ac.in, the cookie created in the previous example will be available to pages with the path http://www.dashan.ac.in/Application1/ and to any pages beneath that folder. However, the cookie will not be available to pages in other applications such as http:// www.dashan.ac.in/Application2/ or just http:// www.dashan.ac.in/

2. Set scope to a domain

- By default, cookies are associated with a specific domain
- For example: if your site is www.dashan.ac.in, the cookies you write are sent to the server when users request any page from that site.
- If your site has sub domains — for example, degree.darshan.ac.in and diploma.darshan.ac.in — then you can associate cookies with a specific subdomain.

```
Response.Cookies("domain").Domain = "diploma.darshan.ac.in"
```

SESSION STATE*

- The session state is used to maintain the session of each user throughout the application. Session allows information to be stored in one page and access in another page and support any type of object.
- In many websites we will see the functionality like once if we login into website they will show username in all the pages for that they will store username in session and they will access that session username in all the pages.
- Important Properties of Session:
  - ✓ **Timeout** – Sets or returns the timeout period (in minutes) for the Session object in this application.
- Important Method of Session:
  - ✓ **Abandon** – Destroys a user session.

- **Modes of Session:**

Sr. No.	Mode Name	Description
1	InProcMode	This is default session state mode which store the session data in memory on the web server.
2	OutProcMode (State Server Mode)	This stores the session data in separate memory called the ASP.NET Service.
3	SQL Server Mode	In this mode session data is stored in SQL Server Database
4	Custom Mode	This option enables you to specify the custom storage option
5	Off Mode	This mode disables the session state.

Login.aspx

```
<table><tr><td>User Name</td><td><asp:TextBox ID="txtUserName"
runat="server"></asp:TextBox></td>
     </tr><tr><td>Password</td><td><asp:TextBox ID="txtPassword"
runat="server"></asp:TextBox></td>
```

```
</tr><tr><td><asp:Label ID="lblMessage" runat="server"
Text="Label"></asp:Label></td>
<td><asp:Button ID="btnLogin" runat="server" Text="Button" /></td></tr>
</table>
```

Login.aspx.vb

```
Protected Sub btnLogin_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnLogin.Click
Dim objConn As New SqlConnection
        objConn.ConnectionString = "Data Source=ANS-PC\SQLEXPRESS;Initial
Catalog=Diploma;Integrated Security=True"
        objConn.Open()
        Dim objcmd As New SqlCommand
        objcmd.Connection = objConn
        objcmd.CommandType = CommandType.Text
        objcmd.CommandText = "select * from UserDetail where User_Name='"
+ txtUserName.Text + "' and Password='" + txtPassword.Text + "'"
        Dim objDR As SqlDataReader = objcmd.ExecuteReader()
        If objDR.HasRows = True Then
            Session("UserName") = txtUserName.Text
            Response.Redirect("Home.aspx")
        Else
            lblMessage.Text = "Please Enter Correct User Name and
Password"
        End If
End Sub
```

Home.aspx

```
<table><tr><td><asp:Label ID="lblShowUserName" runat="server"
Text="Label"></asp:Label></td>
<td><asp:LinkButton ID="lbLogout"
runat="server">Logout</asp:LinkButton></td></tr>
</table>
```

Home.aspx.vb

```
Protected Sub lbLogout_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles lbLogout.Click
        Session.Abandon()
        Response.Redirect("Login.aspx")
End Sub
Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        If (Session("UserName") Is Nothing) Then
```

```
        Response.Redirect("Login.aspx")
    Else
        lblShowUserName.Text = "Welcome " + Session("UserName").ToString
    End IfEnd Sub
```

Application State and Global.asax

- Application State is a state management technique. Application State is stored in the memory of the server and is faster than storing and retrieving information in a database.
- Session state is specific for a single user session but Application State is for all users and sessions. Application State does not have a default expiration period.
- Important Methods of Application:
  - **Lock** – 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**–The Unlock method enables other users to modify the variables stored in the Application object (after it has been locked using the Lock method).

- Important Methods of Global.asax:

Name	Description
Application_Start()	Fired when the first resource is requested from the web server and the web application starts.
Session_Start()	Fired when session starts on each new user requesting a page
Application_Error()	Fired when an error occurs
Session_End()	Fired when the session of a user ends
Application_End()	Fired when the web application ends
Application_Disposed()	Fired when the web application is destroyed

Global.asax

```
Sub Application_Start(ByVal sender As Object, ByVal e As EventArgs)
        Application("SiteVisitedCounter") = 0
End Sub
Sub Session_Start(ByVal sender As Object, ByVal e As EventArgs)
        Application.Lock()
        Application("SiteVisitedCounter") =
Convert.ToInt32(Application("SiteVisitedCounter")) + 1
        Application.UnLock()
End Sub
```

CountUser.aspx

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

CountUser.aspx.vb

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        lblSiteVisited.Text = "No of times site visited=" +
Application("SiteVisitedCounter").ToString()
End Sub
```

Web.Config file *

- Configuration file is used to manage various settings that define a website. The settings are stored in XML files that are separate from your application code. Generally, a website contains a single Web.config file stored inside the application root directory.
- There are number of important settings that can be stored in the configuration file. Some of the most frequently used configurations, stored inside Web.config file are: Database connections, Caching settings, Session States, Error Handling, Security.

- **Configuration for Connection String:**

```
    <configuration>
    <connectionStrings>
<add name ="myCon" connectionString ="server=ANS-PC\SQLEXPRESS;Initial
Catalog=Diploma;Integrated Security=True"
    </connectionStrings></configuration/>
```

- **Reading connectionstring values**

```
    Dim con As string =
ConfigurationManager.ConnectionStrings("myCon").ConnectionString
```

- **Configuration for Custom Error Handling:**

```
    <customErrors defaultRedirect ="Error.aspx">
        <error statusCode ="404" redirect ="NotFound.aspx"/>
    </customErrors>
```

- **Session State and View State Settings:**

```
<Pages EnableViewState="false" />  <sessionState mode="InProc" />
```

Cross Posting

- If we want to transfer data of one page to another page without using session, object or anything else, we can use cross post/page.
- Cross page posting is the concept which is used to submit one page controls to another page and access those page control value in another page.

ADVANTAGES & DISADVANTAGES OF CLIENT SIDE AND SERVER SIDE:

- **CLIENT SIDE:**

✓ ADVANTAGES:

1. No server resources are required
2. Simple implementation
3. Enhanced Security features
4. Widespread support
5. Configurable expiration rules in cookie

✓ DISADVANTAGES:

1. Cannot store large values, because of that page displaying process to user becomes slow
2. Mobile devices might be support or not (View State)
3. No Security for sensitive data
4. If in browser user had disable the cookies then we cannot use that technique also
5. The information in query string is directly visible to the user via the browser's user interface, therefore it is not useful for sensitive data
6. The cookies are stored on the client's machine, there are chances that if the client's machine is hacked then the hacker can view these cookie values.

- **SERVER SIDE:**

✓ ADVANTAGES:

1. If the information that we want to be accessed or stored globally throughout the application, even if multiple users access the site or application at the same time, then we can use an Application Object for such purposes.
2. Session management events can be raised and used by your application.
3. Session state can be used in both multi-computer and multi-process configurations.
4. Session state works with browsers that do not support HTTP cookies

✓ DISADVANTAGES:

1. Session-state& Application state required server memory, therefore can degrade server performance.
2. Variables stored in application state are global only to the particular process the application is running in, and each application process can have different values. Therefore, you cannot rely on application state to store unique values or update global counters
3. Data maintenance
4. Complexity in State management technique