ASP.NET –
STATE MANAGEMENT TECHNIQUES

SHIVRATNA P HONRAO

ASP.NET – STATE MANAGEMENT

- State Management is very important feature in ASP.NET.
- State Management maintains & stores the information of any user till the end of the user session.
- State management can be classified into two types.
  - **Client side state management**
  - **Server side state management**

Client side state management

- Client side state management can be handled by
  - **Hidden Fields**
  - **View State**
  - **Cookies**
  - **Query String**

Server side state Management

- Server side state management can be handled by
  - **Session state**
  - **Application state**

Hidden Field control

- Hidden Field is a control provided by ASP.NET, which is used to store small amount of data on the client
- Hidden Field Control is not rendered to the browser and it is invisible on the browser.

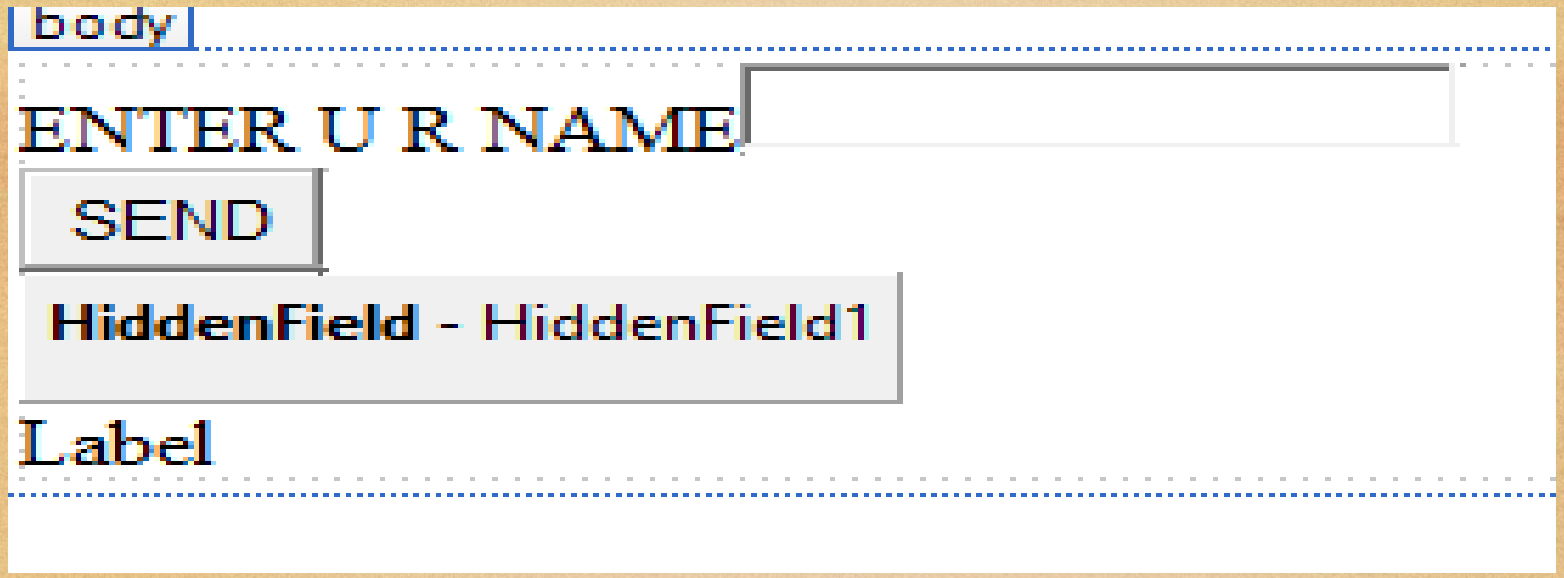
Syntax:

<asp: HiddenField ID="HiddenField1"

runat="server/>

Hidden Field Control Example

HiddenField.aspx



HiddenField.aspx.cs

```
protected void Button1_Click(object sender, EventArgs e)
   {
      HiddenField1.Value = TextBox1.Text; // TO STORE INFO.
      Label2.Text = HiddenField1.Value;      // RETRIVING INFO.
     }
```

ViewState

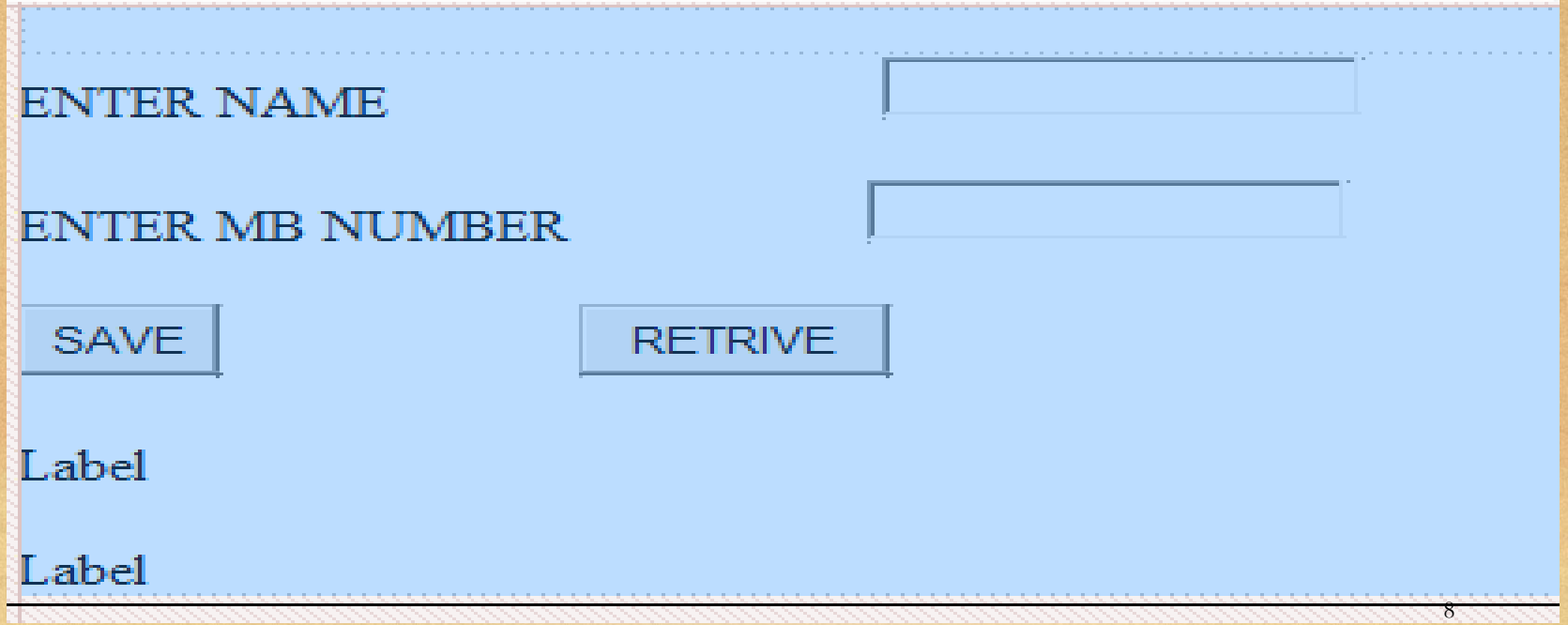
- View state is used to store user's data, which is provided by Asp.net client side state management mechanism.
- ViewState stores data in the generated HTML using hidden field , not on the server.
- ViewState provides page level state management.
  - ie., as long as the user is on the current page, the state will be available; Once the user redirects to the next page , the current state will be lost.
- ViewState can store any type of data & it will be enabled by default for all the serverside control by setting true to "EnableViewState" property.

Client Side State Management - View State - Example

- View State is ana another client side state management mechanism, which is used to store user's data.

* View State Provides Page level StateManagement.

* View state Can Store Any type of data.

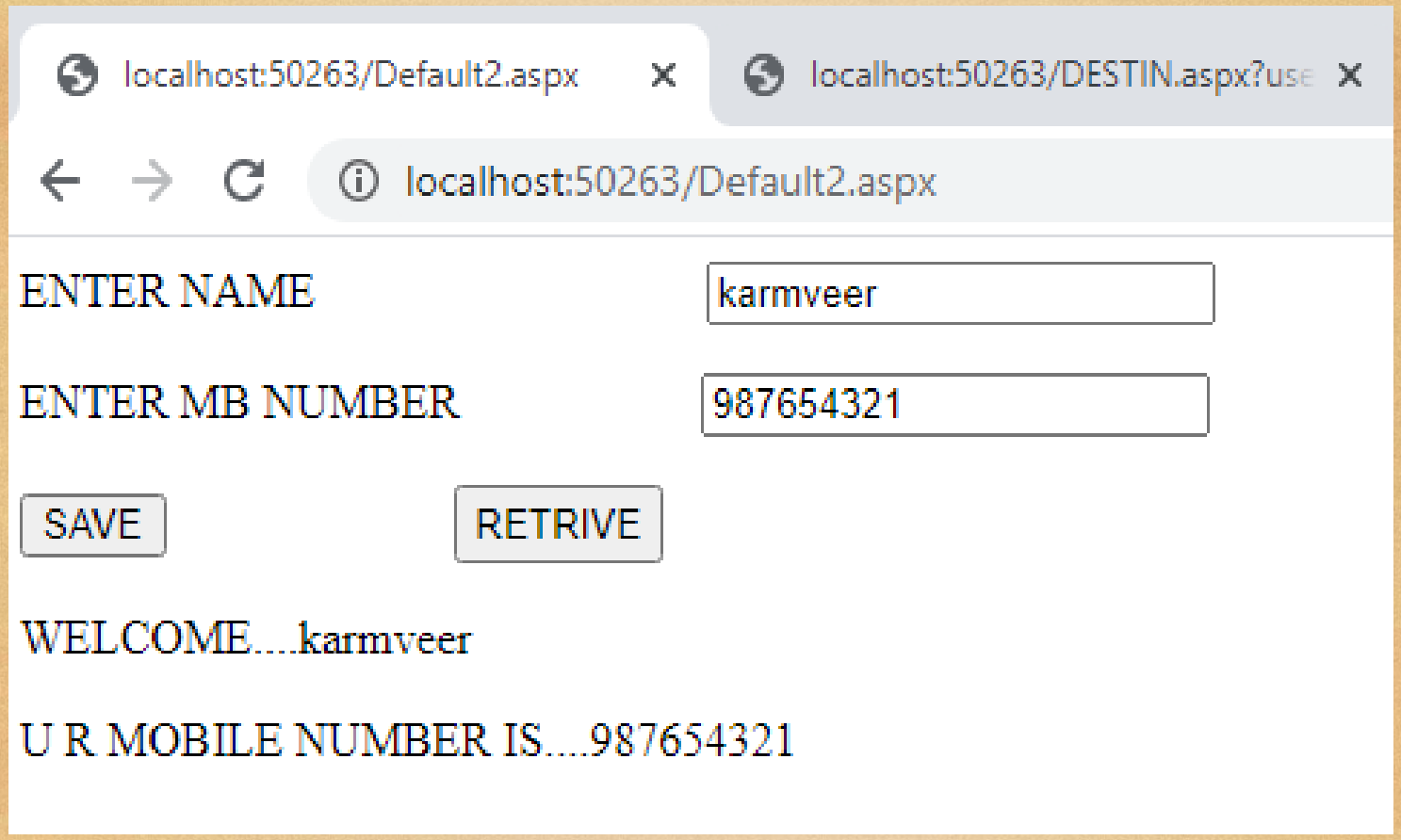


Viewstate.aspx.cs

```
protected void Button1_Click(object sender, EventArgs e)//TO ADD
    {
        ViewState.Add("UNAME", TextBox1.Text);
        ViewState.Add("MB_NUM",TextBox2.Text);
    }

    protected void Button2_Click(object sender, EventArgs e)//TO
RETRIEVE
    {
        Label3.Text = "WELCOME...." + ViewState["UNAME"].ToSting();
        Label4.Text = "U R MOBILE NUMBER IS...." +
ViewState["MB_NUM"].ToString();
    }
```

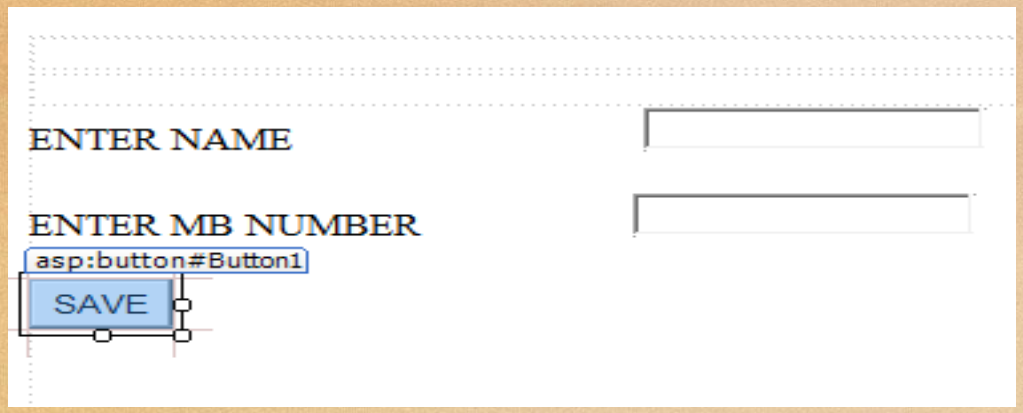
OUTPUT---


localhost:50263/Default2.aspx
localhost:50263/DESTIN.aspx?use
localhost:50263/Default2.aspx
ENTER NAME
karmveer
ENTER MB NUMBER
987654321
SAVE
RETRIVE
WELCOME....karmveer
U R MOBILE NUMBER IS....987654321

QUERYSTRING

- A Querystring is a collection of character input to a computer or a browser.
- In Querystring , we can sent value to only desired page & the value will be temporarily stored.
- Querystring increase the overall performance of web application.
- When we pass content between webforms, the Querystring followed with a separating character (?).
- The question mark(?) is, basically used for identifying data appearing after the separating symbol.

Example : QueryHome.aspx



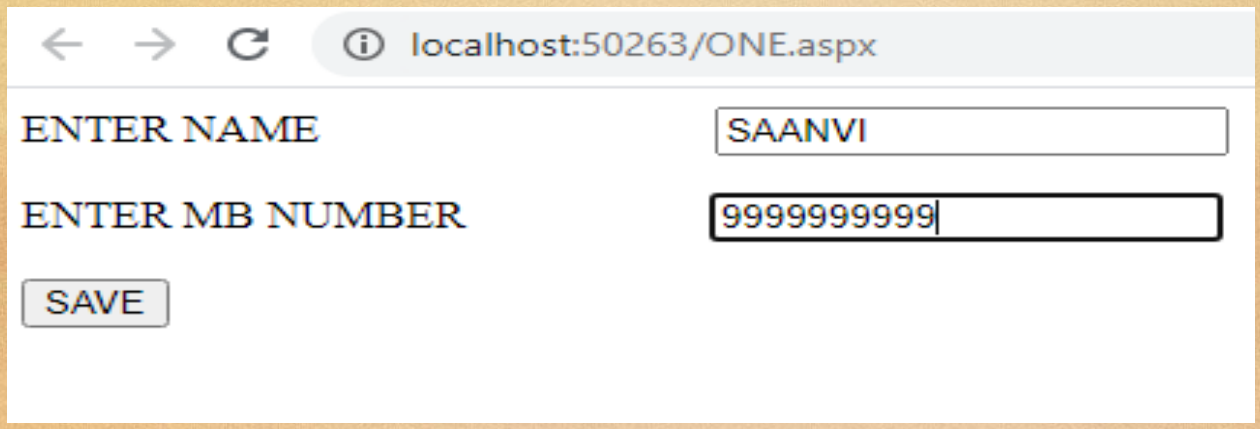
Example : QueryHome.aspx.cs

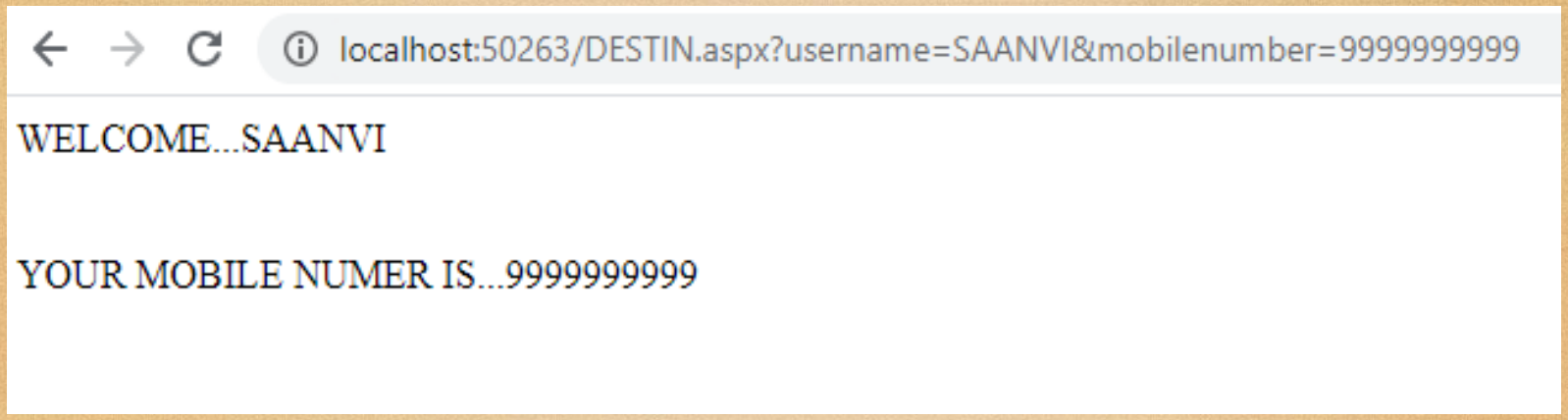
```
protected void Button1_Click(object sender, EventArgs e)
  {
     Response.Redirect("DESTIN.aspx?username="+TextBox1.Text
+"&MobileNumber="+TextBox2.Text);
  }
```

DESTIN.aspx.cs

DESTIN.aspx.cs

```
using System;
using System.Collections.Generic;  using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
public partial class QueryWelcome : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
      {
         Label3.Text = "WELCOME...."+Request.QueryString["username"].ToString();
         Label4.Text = "YOUR MB number IS ..."+
   Request.QueryString["MobileNumber"].ToString();
      }
   }
```


localhost:50263/ONE.aspx
ENTER NAME
SAANVI
ENTER MB NUMBER
9999999999
SAVE


localhost:50263/DESTIN.aspx?username=SAANVI&mobilenumber=9999999999
WELCOME...SAANVI
YOUR MOBILE NUMER IS...9999999999

COOKIES

- Cookies are small piece of text which is stored on the client's computer by the browser.
  - i.e. ., Cookies allows web applications to save user's information to reuse , if needed.

Cookies can be classified into 2 types

1. Persistence Cookie
2. Non-Persistence Cookie

1. Persistence Cookie

- This types of cookies are permanently stored on user hard drive. Cookies which have an expiry date time are called persistence cookies.
- This types of cookies stored user hard drive permanently till the date time we set.

To create persistence cookie:

```
Response.Cookies[“name”].Value = “Nithiyapriya”;
Response.Cookies[“Nithiyapriya”].Expires = DateTime.Now.AddMinutes(2);
```

(or)

```
HttpCookie strname = new HttpCookie(“name”);
strname.Value = “Nithiyapriya”;
strname.Expires = DateTime.Now.AddMinutes(2);
Response.Cookies.Add(strname);
```

Here, the **Cookie Expire time is set as 2 minutes; so that** we can access the cookie values up to 2 minutes, after 2 minutes the cookies automatically expires.

2. Non-Persistence Cookie

- This types of cookies are not permanently stored on user hard drive.
- It stores the information up the user accessing the same browser.
- When user close the browser the cookies will be automatically deleted.

To create non-persistence cookie:

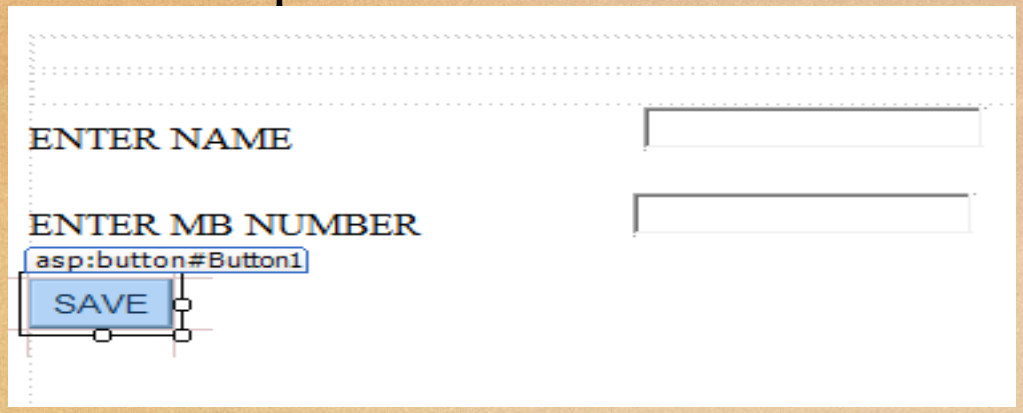
```
Response.Cookies["name"].Value = "Nithiyapriya";
```

(OR)

```
HttpCookie strname = new HttpCookie("name");
strname.Value = "Nithiyapriya";
Response.Cookies.Add(strname);
```

Example

Cookies.aspx

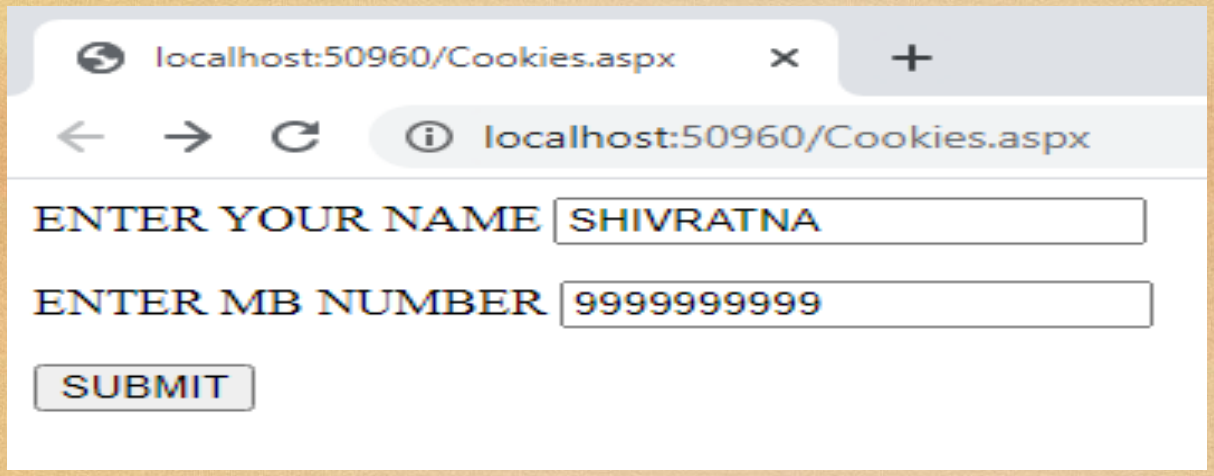


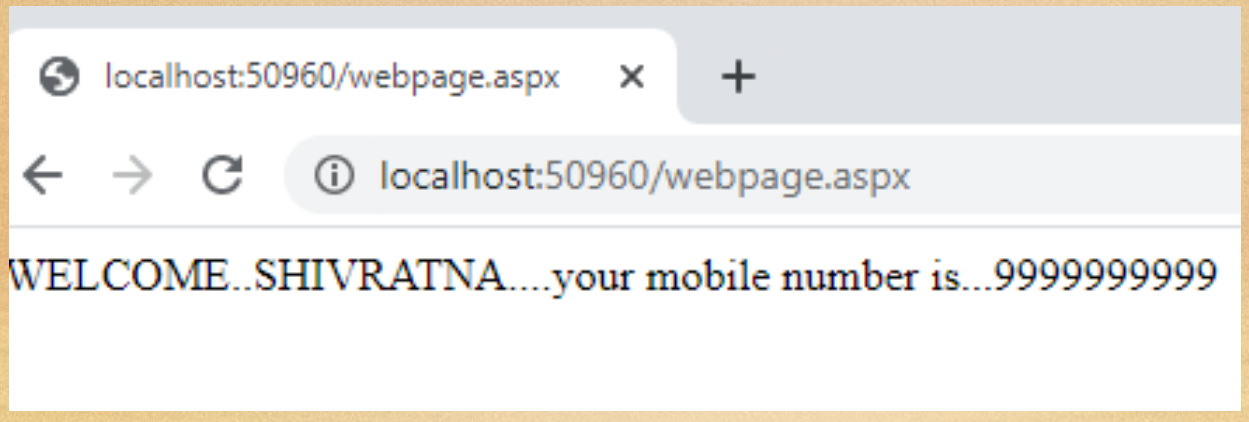
Cookies.aspx.cs

```
protected void Button1_Click(object sender, EventArgs e)
    {
        Response.Cookies["username"].Value = TextBox1.Text;
        Response.Cookies["Mobilenumber"].Value = TextBox2.Text;
        Response.Redirect("webpage.aspx");
    }
```

webpage.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
    {
        Label1.Text = "WELCOME.." +
Request.Cookies["username"].Value + "....your mobile number is..."
+
            Request.Cookies["Mobilenumber"].Value;
    }
```


localhost:50960/Cookies.aspx
localhost:50960/Cookies.aspx
ENTER YOUR NAME
SHIVRATNA
ENTER MB NUMBER
9999999999
SUBMIT


localhost:50960/webpage.aspx
localhost:50960/webpage.aspx
WELCOME..SHIVRATNA....your mobile number is...9999999999

Server side State Management

Session State

- The session is the important features of asp.net
- Session state is used to store value for the particular period of time.
- Session can store the client data on the sever separately for each user & it keeps value across multiple pages of website.
- i.e. user can store some information in session in one page & it can be accessed on rest of all other pages by using session.
- For example ,When a user login to any website with username & password, the username will be shown to all other pages.

Syntax

```
Session["session_name"] = "session value";
```

To Declare session in asp.net

```
Session["name"]="Piyush";
Response.Redirect("Welcomepage.aspx");
```

To Retrieve session value on Welcomepage

```
string myvalue= Session["name"].ToString();
Response.Write("Name = " + myvalue);
```

Example:

To set Session Timeout add this code to Web.config

```
<system.web>
  <compilation debug="true" targetFramework="4.5" />
  <httpRuntime targetFramework="4.5" />
 <sessionState timeout="1"></sessionState>
 </system.web>
```

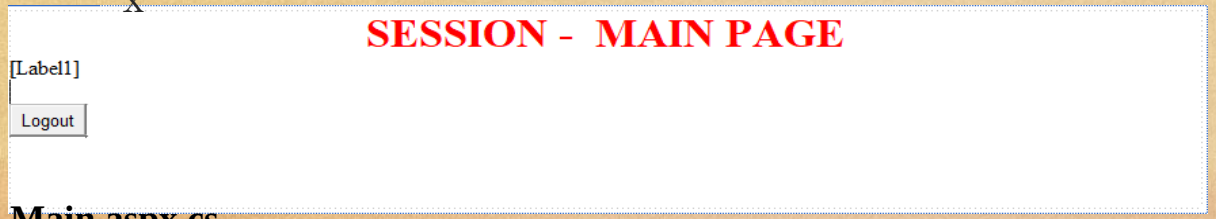
Login.asp
x

Login.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
public partial class Login : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        if (txtusername.Text == "admin" && txtpassword.Text == "admin")
        {
            Session["uname"] = txtusername.Text;
            Response.Redirect("Main.aspx");
     } else
        {
            Label1.Text = "Wrong UserName/Password";
        }    }     }
```

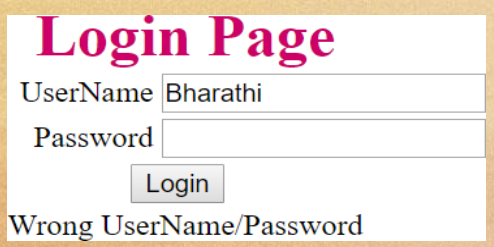
Main.asp

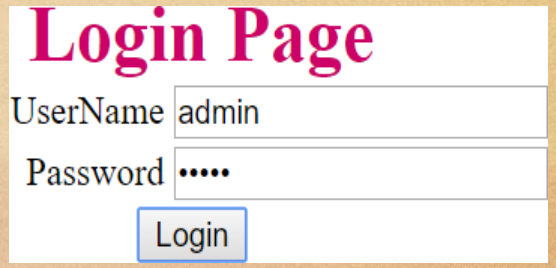
x

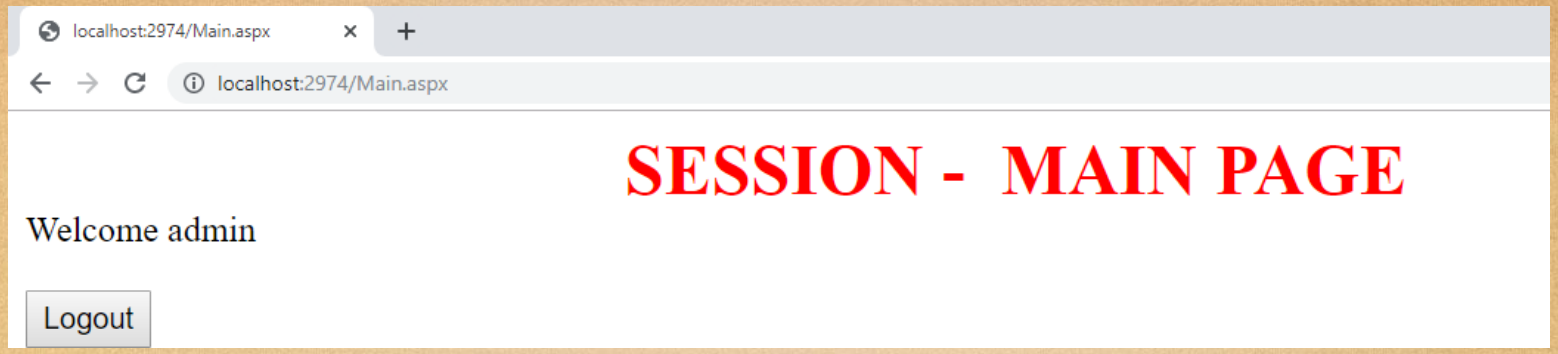


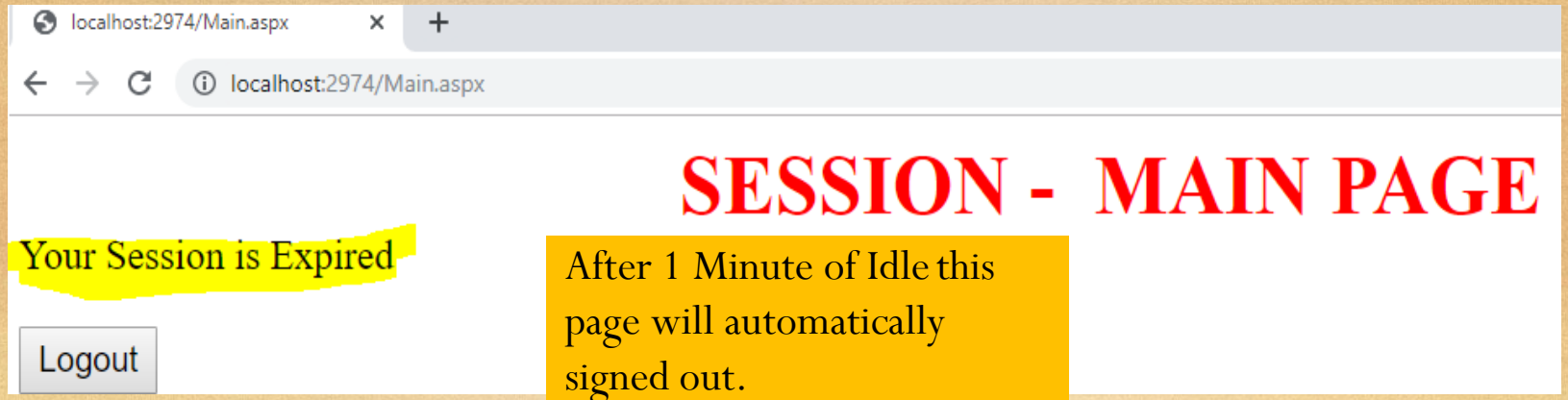
Main.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
public partial class Main : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["uname"] == null)
        {
           // Response.Redirect("Login.aspx");
            Label1.Text = "Your Session is Expired";
    } else
        {
            Label1.Text = "Welcome " + Session["uname"].ToString();
        }    }
    protected void btn_logout_Click(object sender, EventArgs e)
    {
        Session["uname"] = null;
        Response.Redirect("Login.aspx");
    }              }
```









After 1 Minute of Idle this page will automatically signed out.

Application State

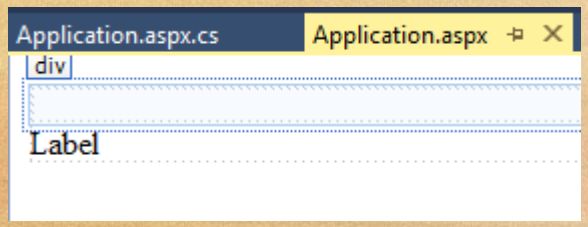
- Application state is server side state management mechanism.
- Application state is used to store data on server & shared for all the users and it can be accessible anywhere in the application.
- The application state is used same as session, but the difference is, the session state is specific for a single user ; where as an application state is common for all users.
- **To Store value in application state**

```
Application["name"] = "Piyush";
```

*__To get value from application state__

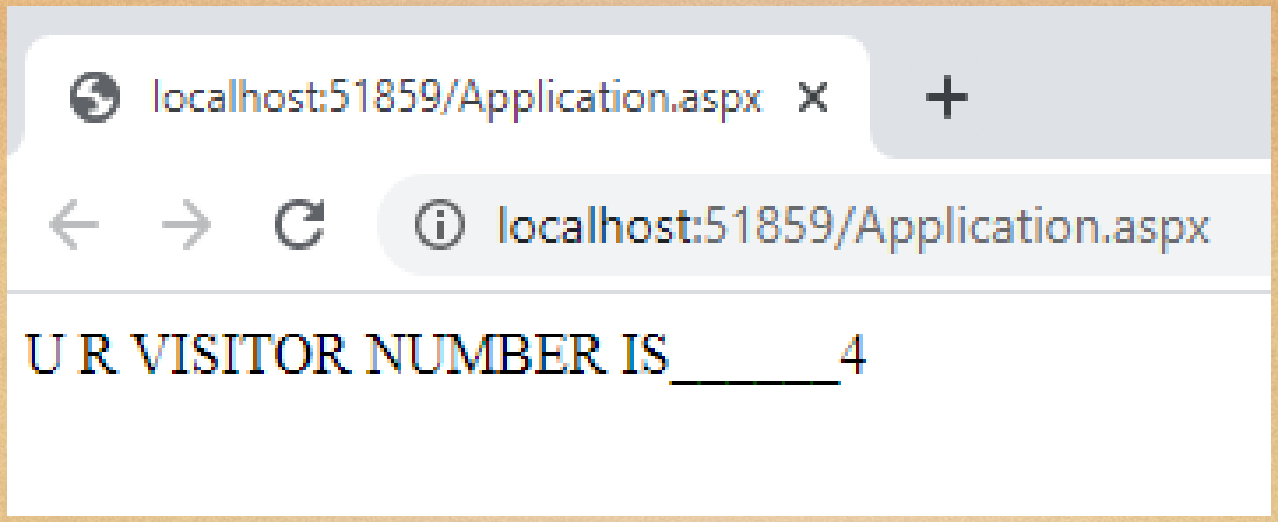
```
string str = Application["name"].ToString();
```

Application.aspx



Application.aspx.cs

```
0 references
protected void Page_Load(object sender, EventArgs e)
{
    Application.Lock();
    int count = 0;
    if(Application["hitcnt"]!=null)
    {
        count = (int)Application["hitcnt"];
    }
    count++;
    Application["hitcnt"] = count;
    Application.UnLock();
    Label1.Text = "U R VISITOR NUMBER IS_____"+count.ToString();
}
```


localhost:51859/Application.aspx
localhost:51859/Application.aspx
U R VISITOR NUMBER IS_____4

Cache OR Caching-

Definition-**Cache** or **Output Caching** means SAVING final rendered web page at clients Browser for certain period of time.

When the next client requests for this page, instead of regenerating the page, a **cached** copy of the page is sent, thus results in saving server execution time.

NEED of OutPut Caching---In order to build scalable web sites it is important to optimize the way your web pages are served to the user. Consider a web site that provides a list of certain products to the user. The products are not going to change frequently. In traditional ASP such product catalog page will be processed each time a new request comes. Event though there is nothing dynamic in the page it will be processed every time. This is clearly not efficient. This is where ASP.NET caching features come handy. Caching allows you to store a particular version of ASP.NET page on the server. The requests to the same page are served with this cached version of page rather than processing the page every time. This is called as Output Caching.

-To perform caching we can use OutputCache Directive.

The @OutputCache directive

- The OutputCache directive is used to enable caching for a ASP.NET page or even user control. This directive tells ASP.NET the duration in seconds for which the page is to be cached. Following markup shows the usage of the directive:

```
<%@ OutputCache duration="secs“ varybyparam="param" %>
```

Here, the duration attribute specifies the time in seconds for which the page will be cached. The VaryByXXXX attributes tells how ASP.NET should cache multiple versions of the page..

Example 1 - Simple OutputCaching

In this example we will see the simplest form of caching syntax. Here, we are caching page output for 30 seconds. Also, we are caching only one version of the page.

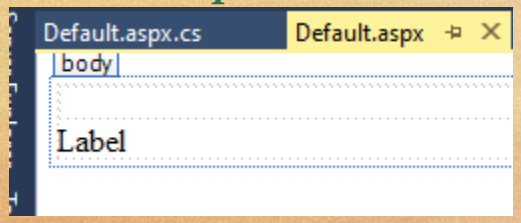
```
<%@ OutputCache duration="30" varybyparam="none" %>
```

Example 2 - Caching multiple versions based on form field

Consider an example where want to cache output of a book catalog for different values of categories. Here, you will provide a textbox or dropdownlist to select category. Depending on the category the books are fixed and you will cache that version. This can be achieved by using the VaryByParam attribute.

```
<%@ OutputCache duration="30" VaryByParam="TextBox1" %>
```

Default.aspx




```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default"  %>
<%@ OutputCache Duration="60"  VaryByParam="none" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

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

Default.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    Label1.Text = "current date and time is ...."+DateTime.Now.ToString();
}
}
```

Thank You