**ASP.net**

**Active Server Page**

**its a web application**

**Definition:**

Asp.net is server side technologies.

for creating Dynamic web page. Using dotnet programming language

**classical ASP :**

using scripting Language

(javascript)

(vbscript)

WebPage :

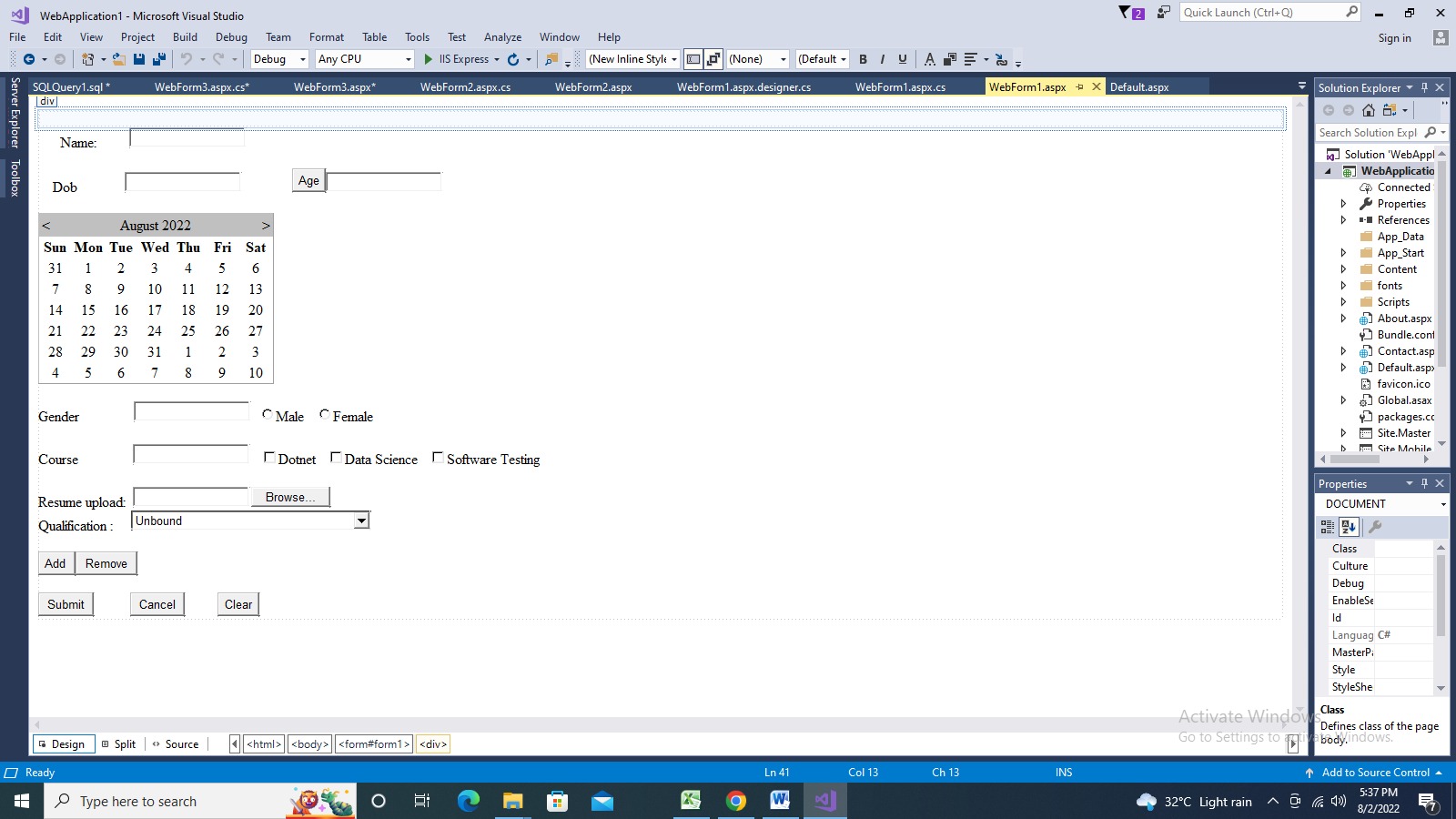
1. Static webpage
2. Dynamic webPage

IIS :Internet Information Service (or) system asp.net Webserver

**PostBack:**

PostBack is the name given to the process of submitting an ASP.NET page to the server for processing. PostBack is done if certain credentials of the page are to be checked against some sources (such as verification of username and password using database).

webForm



protected void Page\_Load(object sender, EventArgs e)

{

}

//To add Item in dropdownlist

protected void Button1\_Click(object sender, EventArgs e)

{

DropDownList1.Items.Add("ECE");

DropDownList1.Items.Add("Civil");

DropDownList1.Items.Add("EEE");

DropDownList1.Items.Add("CSC");

DropDownList1.SelectedItem.Text = "CSC";

}

//to remove Item

protected void Button2\_Click(object sender, EventArgs e)

{

DropDownList1.Items.Remove(DropDownList1.SelectedItem.Text);

}

//find the age

protected void Button3\_Click(object sender, EventArgs e)

{

DateTime d1 =DateTime .Parse ( TextBox2.Text);

int year = d1.Year;

int curentyear = DateTime.Now.Year;

int age = curentyear - year;

TextBox3.Text = age.ToString();

}

//initcap

//set autopostback property true

protected void TextBox1\_TextChanged(object sender, EventArgs e)

{

string s = TextBox1.Text;

string s1 = s.Substring(1);

string s2 = s[0].ToString().ToUpper();

TextBox1.Text = s2 + s1;

}

//calender selection mode=day

protected void Calendar1\_SelectionChanged(object sender, EventArgs e)

{

TextBox2.Text = Calendar1.SelectedDate.ToShortDateString();

}

//set autopostback property true

protected void RadioButton1\_CheckedChanged(object sender, EventArgs e)

{

if (RadioButton1 .Checked ==true)

{

TextBox4.Text = RadioButton1.Text;

RadioButton2.Checked = false;

}

else

{

TextBox4.Text = "";

}

}

//set autopostback property true

protected void RadioButton2\_CheckedChanged(object sender, EventArgs e)

{

if (RadioButton2.Checked == true)

{

TextBox4.Text = RadioButton2.Text;

RadioButton1.Checked = false;

}

else

{

TextBox4.Text = "";

}

}

//set autopostback property true

protected void CheckBox1\_CheckedChanged(object sender, EventArgs e)

{

if (CheckBox1 .Checked ==true)

{

TextBox5.Text = CheckBox1.Text;

}

}

//set autopostback property true

protected void CheckBox2\_CheckedChanged(object sender, EventArgs e)

{

if (CheckBox2.Checked == true)

{

TextBox5.Text = CheckBox2.Text;

}

}

//set autopostback property true

protected void CheckBox3\_CheckedChanged(object sender, EventArgs e)

{

if (CheckBox3.Checked == true)

{

TextBox5.Text = CheckBox3.Text;

}

}

}

}

Validation Controls

Validation controls validate the user input data

ASP.NET provides the following validation controls:

* RequiredFieldValidator
* Range Validator
* CompareValidator
* RegularExpressionValidator
* CustomValidator
* ValidationSummary

## RequiredFieldValidator Control

The RequiredFieldValidator control ensures that the required field is not empty. It is generally tied to a text box to force input into the text box.

<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>

&nbsp;<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"

ControlToValidate="TextBox1"

ErrorMessage="Should be value"></asp:RequiredFieldValidator>

## CompareValidator Control

The CompareValidator control compares a value in one control with a fixed value or a value in another control.

password&nbsp;&nbsp;

<asp:TextBox ID="TextBox2" runat="server" TextMode="Password"></asp:TextBox>

<asp:CompareValidator ID="CompareValidator1"

runat="server" ControlToCompare="TextBox3"

ControlToValidate="TextBox2"

ErrorMessage="pwd and confirm pwd should be same"></asp:CompareValidator>

<br />

<br />

<br />

confirm pwd&nbsp; <asp:TextBox ID="TextBox3" runat="server" TextMode="Password"></asp:TextBox>

## RegularExpressionValidator

The RegularExpressionValidator allows validating the input text by matching against a pattern of a regular expression.

<asp:RegularExpressionValidator ID="RegularExpressionValidator1"

runat="server" ControlToValidate="TextBox5"

ErrorMessage="RegularExpressionValidator"

ValidationExpression="[A-Z]{2}"></asp:RegularExpressionValidator>

|  |  |
| --- | --- |
| **Character Escapes** | **Description** |
| \b | Matches a backspace. |
| \t | Matches a tab. |
| \r | Matches a carriage return. |
| \v | Matches a vertical tab. |
| \f | Matches a form feed. |
| \n | Matches a new line. |
| \ | Escape character. |

Apart from single character match, a class of characters could be specified that can be matched, called the metacharacters.

|  |  |
| --- | --- |
| **Metacharacters** | **Description** |
| . | Matches any character except \n. |
| [abcd] | Matches any character in the set. |
| [^abcd] | Excludes any character in the set. |
| [2-7a-mA-M] | Matches any character specified in the range. |
| \w | Matches any alphanumeric character and underscore. |
| \W | Matches any non-word character. |
| \s | Matches whitespace characters like, space, tab, new line etc. |
| \S | Matches any non-whitespace character. |
| \d | Matches any decimal character. |
| \D | Matches any non-decimal character. |

Quantifiers could be added to specify number of times a character could appear.

|  |  |
| --- | --- |
| **Quantifier** | **Description** |
| \* | Zero or more matches. |
| + | One or more matches. |
| ? | Zero or one matches. |
| {N} | N matches. |
| {N,} | N or more matches. |
| {N,M} | Between N and M matches. |

## RangeValidator Control

The RangeValidator control verifies that the input value falls within a predetermined range.

Age <asp:TextBox ID="TextBox4" runat="server"></asp:TextBox>

<asp:RangeValidator ID="RangeValidator1" runat="server"

ErrorMessage="Enter 25 to 35"

ControlToValidate="TextBox4"

MinimumValue="25"

MaximumValue="35"

Type="Integer"

></asp:RangeValidator>

## ValidationSummary

The ValidationSummary control does not perform any validation but shows a summary of all errors in the page. The summary displays the values of the ErrorMessage property of all validation controls that failed validation.

## CustomValidator

Its User define validation

<asp:TextBox ID="TextBox4" runat="server"></asp:TextBox>

<asp:CustomValidator ID="CustomValidator1" runat="server" ControlToValidate="TextBox4"

ErrorMessage="CustomValidator" OnServerValidate="CustomValidator1\_ServerValidate1"></asp:CustomValidator>

<br />

<br />

<br />

</div>

<asp:Button ID="Button1" runat="server" Text="Button" />

**protected void CustomValidator1\_ServerValidate1(object source, ServerValidateEventArgs args**)

{

if(args.Value.Length ==5)

{

//args.IsValid=true;

Response.Write("Valid User");

}

else

{

Response.Write("InValid User");

}

}

## Navigation methods in ASP.NET

ASP.NET supports following ways to navigate between pages in your application.  
  
**Hyperlink control**  
This is server control use for navigation to another page specified in the NavigateURL property. Hyperlink control doesn’t expose any server side event.  
  
**Response.Redirect method**  
This method is used to navigate to another page from code. You can use this method to navigate from a Linkbutton or ImageButton control.  
  
**Example:**

Private Sub LinkButton1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles LinkButton1.Click  
        Response.Redirect("Page2.aspx")  
End Sub

**Server.Transfer method**  
This method can be used only with .aspx file. It allows to retain some information between the requests when its preserveForm argument is set to true.

## Explain the difference between Server.Transfer and response.Redirect.

Redirect and Transfer both cause a new page to be processed. The difference lies in the way the interaction between the client and the server occurs.  
  
Response.Redirect messages the client browser asking it to request for another page.  
e.g. if a browser is on page A which has a Response.Redirect, then it asked to request for another page B by the server. When the client browser requests for it, then it is provided with the requested page B.  
  
With Server.Transfer, the browser is not requested to ask for another page. Instead it is directly provided with the page B. In this scenario, the browser address bar continues to show the address of the previous URL.

**Validation**

Validation is a process of testing and ensuring that the user has entered required and properly formatted information through the web form.

**Client-side validation**

In client-side validation method, all the input validations and error recovery process is carried out on the client side i.e on the user’s browser. It can be done using JavaScript, AJAX, HTML5 etc.

**Server-side validation**

In server-side validation, all the input validations and error recovery process is carried out on the server side. It can be done using programming languages like C#.NET, VB.NET etc.

**Differences and comparison:**

Client-side validation is faster than server-side because, the validation takes place on client side (on browser) and the networking time from client to server is saved.

On the other hand, server-side validation is done on the web server. Then the server renders the data into html page and sends back to the client (browser).

Server-side validation is more secure than the client-side as the user cannot see the code even he does a view-source.

**Stored Procedure**

* Stored Procedure is precompiled set of one or more SQL statements which performs some specific task
* Stored Procedure is a group of sql statements which is stored in the database and executes when it calls
* Store Procedure should be executed standalone using EXEC
* We can pass parameters and return values from stored procedure
* Store Procedure can be used to implement transaction

**Advantages**

* It is execute fast
* We can reuse the queries
* We can avoid Sql Injection
* We can avoid network traffic

**A SQL injection is a technique that attackers use to gain unauthorized access to a web application database by adding a string of malicious code to a database query.**

**Sqldatabase**

--select all table from current db

select \* from sys.tables

select \* from emp

insert into emp values(15,'SElva',65000,'09-03-1998')

--stored procedure

create procedure procedurename(@parameter1 datatype,@parameter2 datatype,...)

as

begin

//stm

end

update emp set esal= esal+1000 where ename='priya'

create proc proc11

as

begin

select \* from emp where esal>25000

update emp set esal= esal+100 where esal>25000

select \* from emp

end

--call

--execute procedurename(arg value)

execute proc11

--2nd max salary for stored procedure

create proc secondmaxsal

as

begin

select \* from emp where esal=

(select min(esal) from emp where esal in

( select top 2 esal from emp order by esal desc))

end

execute secondmaxsal

--insert stored procedure

create proc insertporocedure

(@id int,

@name varchar(10),

@sal int,

@dob datetime)

as

begin

insert into emp values(@id,@name,@sal,@dob)

end

execute insertporocedure 18,'Kalai',78000,'12-12-1997'

--update stored procedure

create proc updateprocedure

(@id int)

as

begin

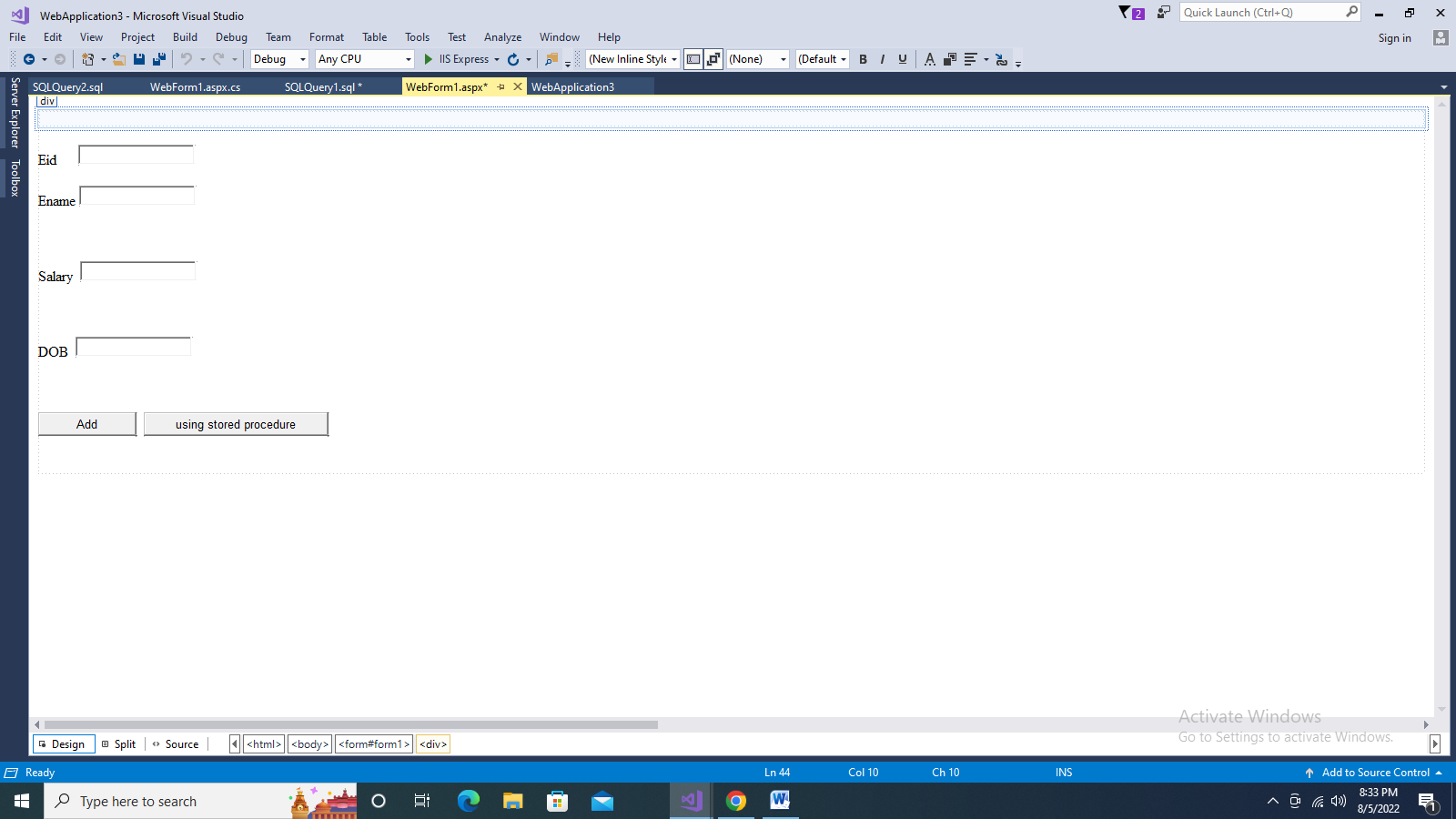
update emp set esal=esal+1000 where eid=@id

end

--call

execute updateprocedure 102

select \* from emp



protected void Button1\_Click(object sender, EventArgs e)

{

SqlConnection con = new SqlConnection(@"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=mydb;Integrated Security=True");

con.Open();

SqlCommand cmd = new SqlCommand("Insert into emp values(" + TextBox1.Text + ",'" + TextBox2.Text + "'," + TextBox3.Text + ",'" + TextBox4.Text + "')", con);

cmd.ExecuteNonQuery();

con.Close();

Response.Write("Data submit successfully");

}

//using stored procedure

protected void Button2\_Click(object sender, EventArgs e)

{

SqlConnection con = new SqlConnection(@"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=mydb;Integrated Security=True");

con.Open();

SqlCommand cmd = new SqlCommand("insertporocedure", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@id", SqlDbType .Int).Value = TextBox1.Text;

cmd.Parameters.AddWithValue("@name", SqlDbType.VarChar).Value = TextBox2.Text;

cmd.Parameters.AddWithValue("@sal", SqlDbType.Int).Value = TextBox3.Text;

cmd.Parameters.AddWithValue("@dob", SqlDbType.DateTime).Value = TextBox4.Text;

cmd.ExecuteNonQuery();

con.Close();

Response.Write("Data submit successfully");

}

**Master Page**

Master Page is Place Holder of the Content Page.

Its look and feel and behavior for all pages of your application.

Master Page Extension is Pagename.master.

Content Page

As you want Display the Content page (its normal page)

Page.aspx

It output will be the combination of master page and content page.

Master Page

<%@ Master Language="C#" AutoEventWireup="true" CodeFile="MasterPage.master.cs" Inherits="MasterPage" %>

<!DOCTYPE html>

<html>

<head runat="server">

<title></title>

<asp:ContentPlaceHolder id="head" runat="server">

</asp:ContentPlaceHolder>

</head>

<body>

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

<div>

<table>

<tr>

<td>

<asp:Panel ID="Panel1" runat="server" Height ="132px" Width ="1241px" BackColor="#FFCCCC">

<asp:ImageButton ID="ImageButton1" runat="server" Height="81px" ImageUrl="~/image1.jpg" Width="175px" />

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<asp:Label ID="Label1" runat="server" Font-Bold="True" Font-Size="X-Large" ForeColor="#FF3399" Text="Welcome to my website"></asp:Label>

<br />

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<br />

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<asp:LinkButton ID="LinkButton2" runat="server" Font-Bold="True" Font-Size="Larger" PostBackUrl="~/Home page.aspx">Home Page</asp:LinkButton>

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<asp:LinkButton ID="LinkButton1" runat="server" Font-Bold="True" Font-Size="Larger" PostBackUrl="~/Login Page.aspx">Login Pagte</asp:LinkButton>

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<asp:LinkButton ID="LinkButton3" runat="server" Font-Bold="True" Font-Size="Larger" PostBackUrl="~/Register page.aspx">Register Page</asp:LinkButton>

</asp:Panel>

</td>

</tr>

</table>

<table>

<tr>

<td>

<asp:Panel ID="Panel2" runat="server" Height ="1040px" Width ="200px" BackColor="#CCFFFF"></asp:Panel>

</td>

<td>

<asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">

</asp:ContentPlaceHolder>

</td>

</tr>

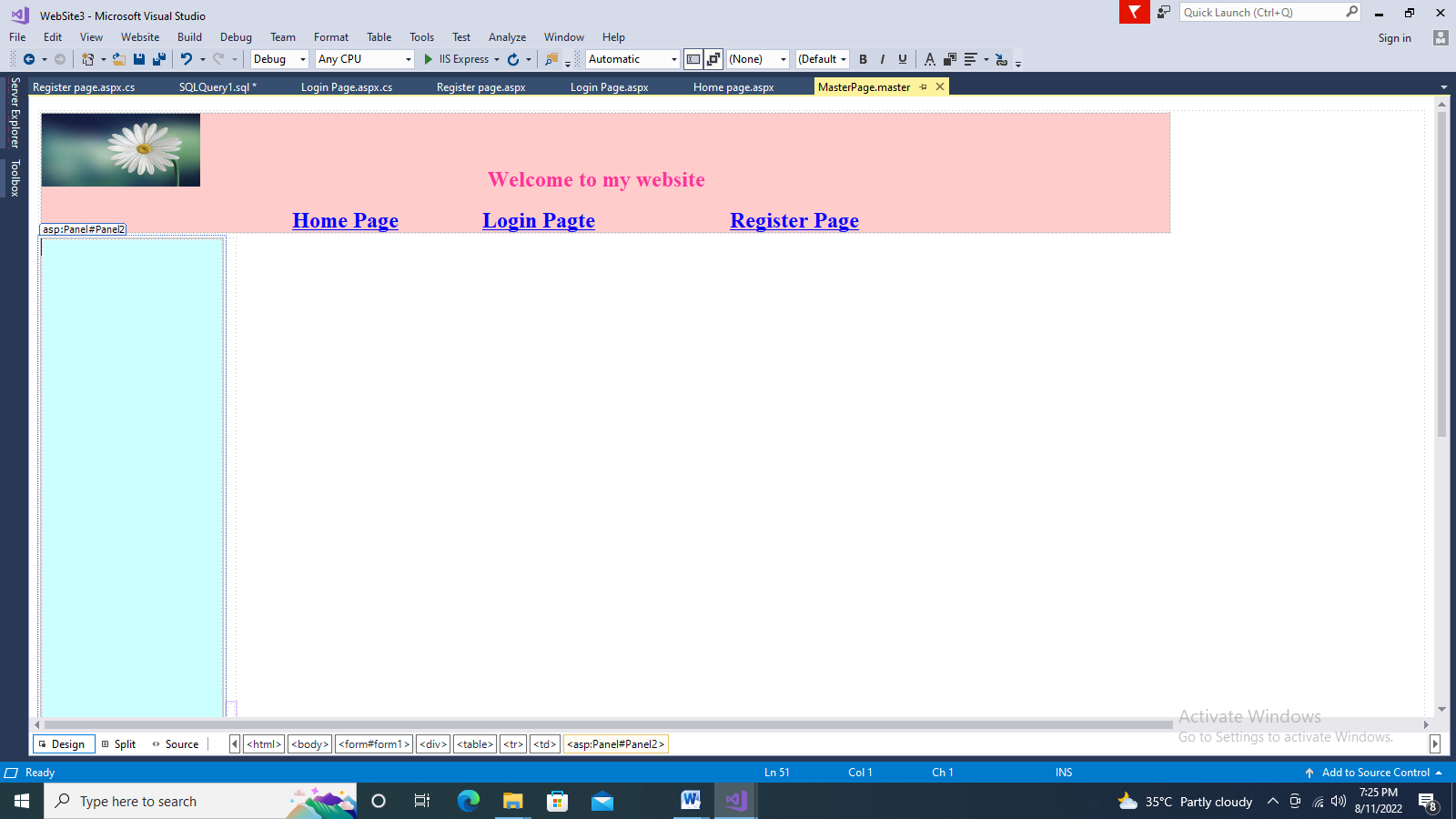
</table>

</div>

</form>

</body>

</html>



**Contenet Page for Login page**

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">

<table class="auto-style2" >

<tr>

<td class="auto-style3">

<asp:Label ID="Label3" runat="server" Text="Login PAge"></asp:Label>

</td>

</tr>

<tr>

<td class="auto-style3">

<asp:Label ID="Label1" runat="server" Text="UserId"></asp:Label>

</td>

<td>

<asp:TextBox ID="TextBox1" runat="server" Height="16px" Width="223px"></asp:TextBox>

</td>

</tr>

<tr>

<td class="auto-style3">

<asp:Label ID="Label2" runat="server" Text="Password"></asp:Label>

</td>

<td>

<asp:TextBox ID="TextBox2" runat="server" Height="16px" Width="223px"></asp:TextBox>

</td>

</tr>

<tr>

<td class="auto-style3">

<asp:Button ID="Button1" runat="server" Text="Submit" OnClick="Button1\_Click" />

</td>

<td>

<asp:Button ID="Button2" runat="server" Text="Cancel" />

</td>

<td>

<asp:Button ID="Button3" runat="server" Text="Register Here" Height="16px" OnClick="Button3\_Click" Width="88px" />

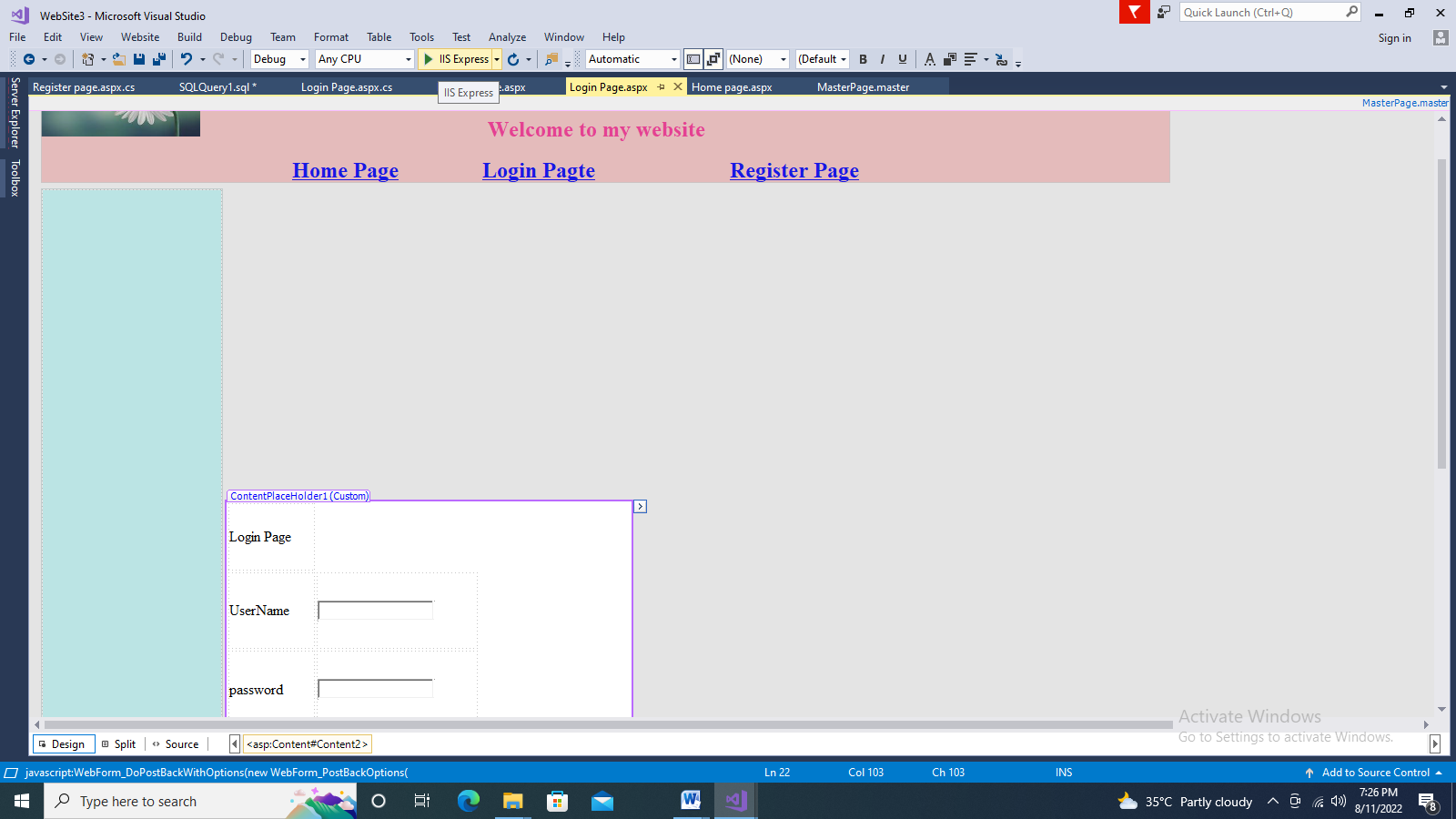
</td>

</tr>

</table>

</asp:Content>

**Design page**



**Contenet page for Register Page**

<table class="auto-style2">

<tr>

<th> Register Page</th>

</tr>

<tr>

<td>

<asp:label id="lbl1" runat="server" text="UserName"></asp:label>

</td>

<td>

<asp:textbox runat="server" ></asp:textbox>

</td>

</tr>

<tr>

<td>

<asp:label id="Label1" runat="server" text="Password"></asp:label>

</td>

<td>

<asp:textbox runat="server" ></asp:textbox>

</td>

</tr>

<tr>

<td>

<asp:label id="Label2" runat="server" text="Con Password"></asp:label>

</td>

<td>

<asp:textbox runat="server" ></asp:textbox>

</td>

</tr>

<tr>

<td>

<asp:label id="Label3" runat="server" text="MailId"></asp:label>

</td>

<td>

<asp:textbox runat="server" ></asp:textbox>

</td>

</tr>

<tr>

<td>

<asp:label id="Label4" runat="server" text="Adress"></asp:label>

</td>

<td>

<asp:textbox runat="server" ></asp:textbox>

</td>

</tr>

<tr>

<td>

<asp:label id="Label5" runat="server" text="Phone No"></asp:label>

</td>

<td>

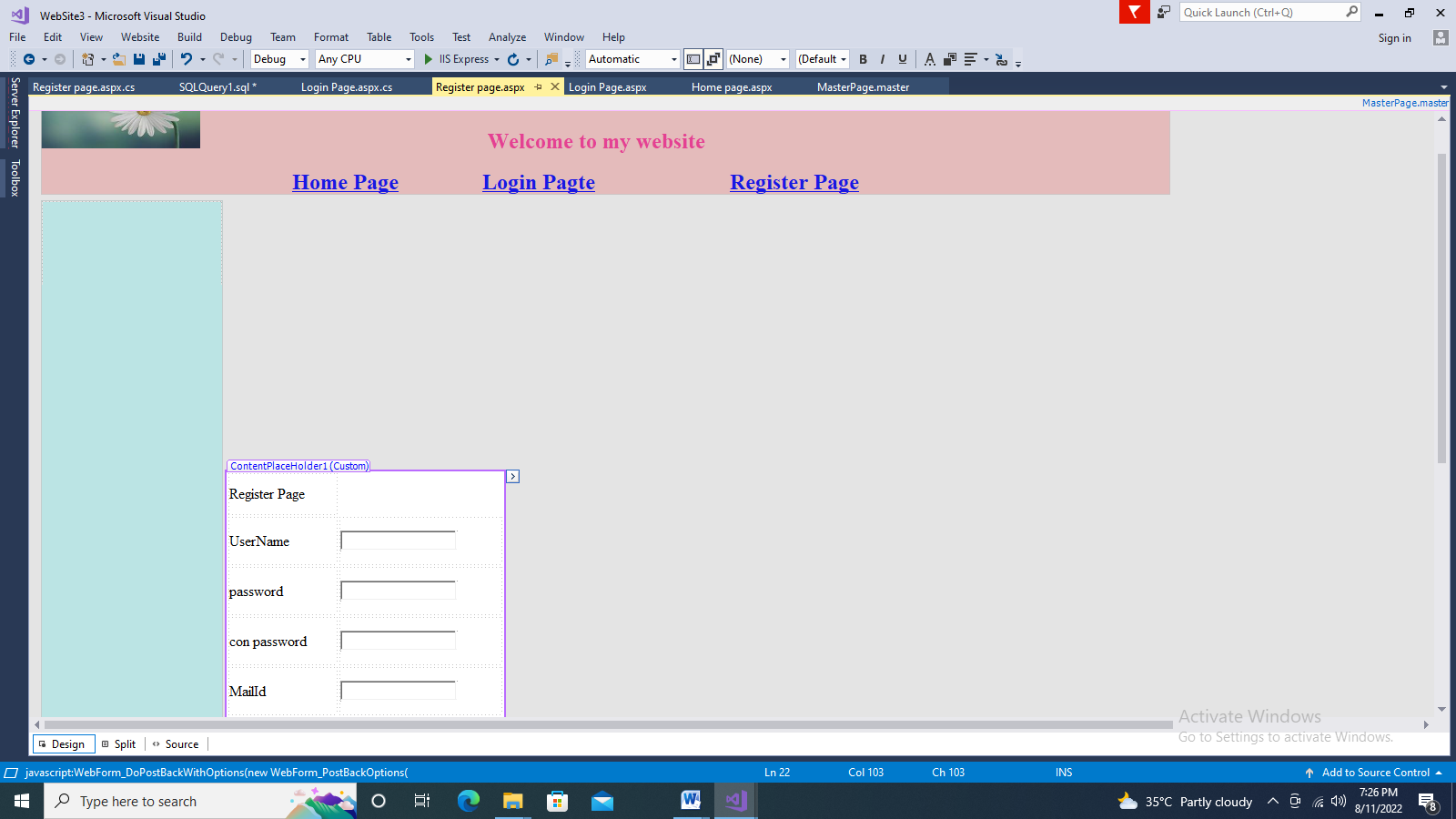
<asp:textbox runat="server" ></asp:textbox>

</td>

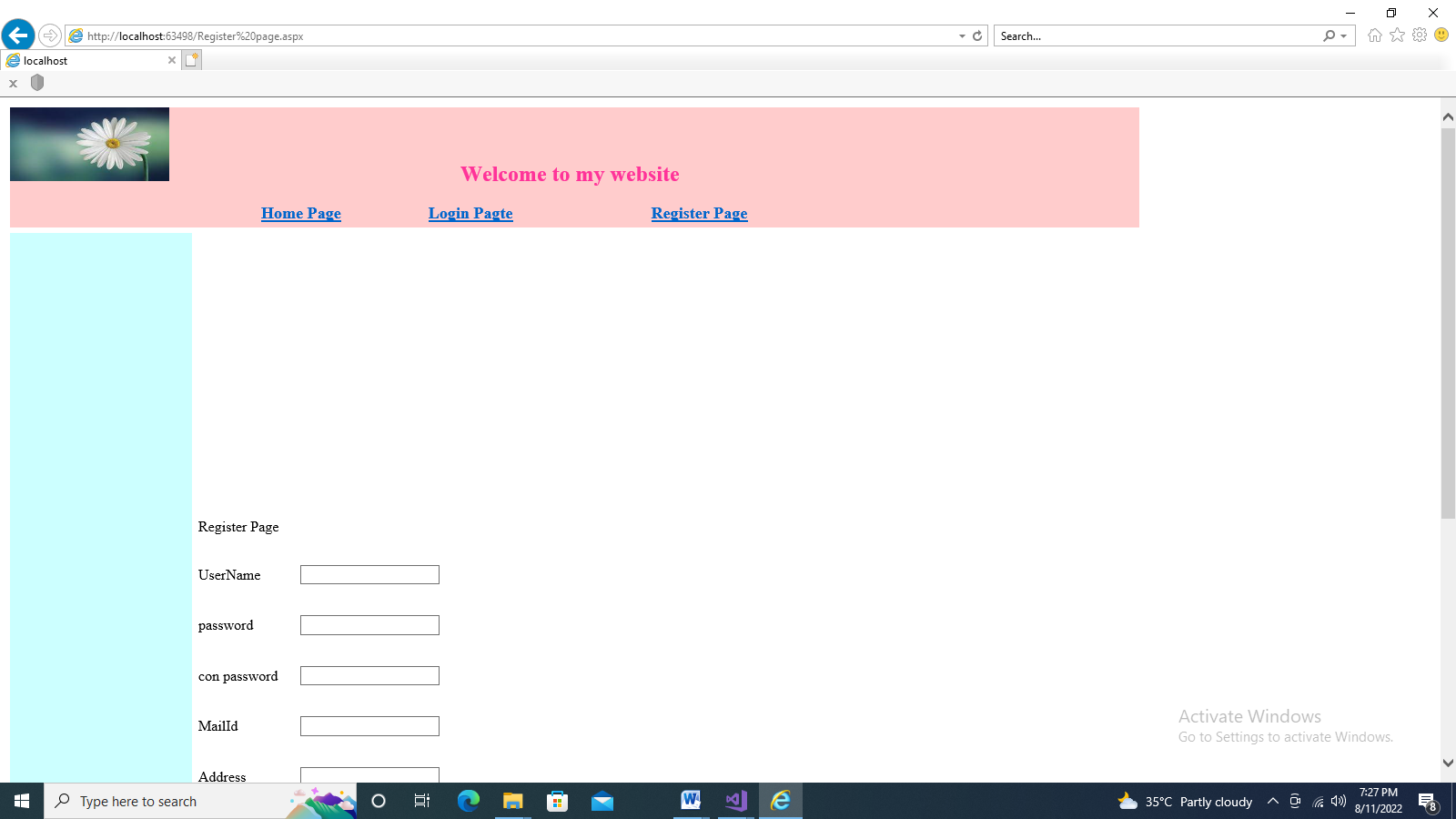
</tr>

</table>

**Design Page**



**Output window**



**XML (Extensible Markup Language**)

**Extensible Markup Language** (XML) stores and transports data. If we use a XML file to store the data then we can do operations with the XML file directly without using the database. The XML format is supported for all applications. It is independent of all software applications and it is accessible by all applications.

**Code window**

using System.Data;

using System.Data.SqlClient;

public partial class \_Default : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

//ReadXML (xml to Gridview)

protected void Button1\_Click(object sender, EventArgs e)

{

DataSet ds = new DataSet();

ds.ReadXml(MapPath("XMLFile2.xml"));

GridView1.DataSource = ds;

GridView1.DataBind();

}

//write xml

protected void Button2\_Click(object sender, EventArgs e)

{

DataSet ds = new DataSet();

ds.ReadXml(MapPath("XMLFile2.xml"));

ds.WriteXml(MapPath("emp1.xml"));

}

//DB to XML

protected void Button3\_Click(object sender, EventArgs e)

{

SqlConnection con = new SqlConnection(@"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=joelDB;Integrated Security=True");

SqlCommand cmd = new SqlCommand("select \* from emp", con);

SqlDataAdapter da = new SqlDataAdapter(cmd);

DataSet ds = new DataSet();

da.Fill(ds, "emp");

TextBox1.Text = ds.GetXml();

}

//webservice

A web service is a web-based functionality accessed using the protocols of the web to be used by the web applications

A web service is a web application which is basically a class consisting of methods that could be used by other applications. It also follows a code-behind architecture such as the ASP.NET web pages, although it does not have a user interface.

protected void Button4\_Click(object sender, EventArgs e)

{

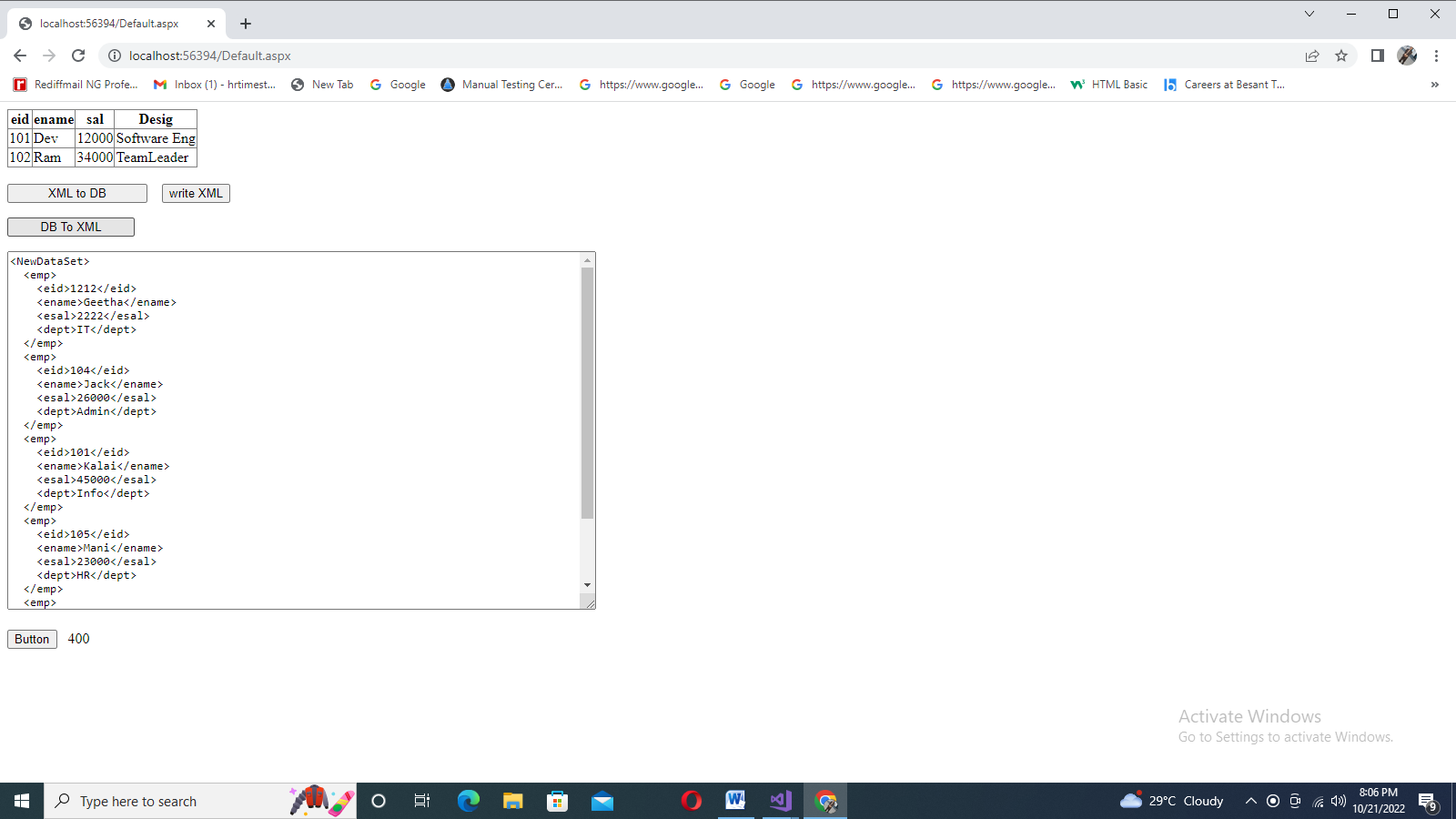
ServiceReference1.WebServiceSoapClient ww = new ServiceReference1.WebServiceSoapClient();

Label1.Text = ww.simpleinterest(10000, 2, 2).ToString();

}

}

Output for XML and webservice



## State Management

In an ASP NET application, state management in ASP NET is an object and preserves type state control. This is because ASP NET applications are basically stateless. In ASP NET, the information of users is stored and maintained till the user session ends. Each time the page is posted on the server, a new instance of the Web page class is created. Whenever the user enters information, this information might get lost in the round trip from the browser (MSDN), if they enter into the web application.

Now, let’s discuss the types of State Management.

## Types of State Management in ASP.NET

There are two types of State management in ASP net. They are :

* Server-side
* Client-side

These are further subdivided into the following -

### Server-Side

* Session
* Application

### Client-Side

* Cookies
* viewstate
* Query String

### Server-Side of State Management in ASP NET

#### Session

An important technique to maintain state. It is used to store identity and information; information is stored in the server using Sessionid.

To start the user session -

Example

protected void btnSubmit\_Click(object sender, EventArgs e)

{

   Session["UserName"] = txtName.Text;

   Response.Redirect("Home.aspx");

}

##### **Session Event**

Two types -

Session starts - Raised every time a new user requests without a session ID.

Example

void Session\_Start(object sender, EventArgs e)

{

   Session["Master"] = "~/Master.master";

}

Session end - Raised everytime the user ends the session or a time out occurs.

Example

void Session\_End(object sender, EventArgs e)

{

   Response.Write("Session\_End");

}

The session is stored in the following ways in ASP.NET:

InProcMode - Default session mode. When the server starts, the session value is stored, and when the server is restarted, it ends.

State Server Mode - Session date is made to store on a separate server in this mode.

SQL Server Mode - It’s a secure mode in which the session is made to store in the database.

Custom Mode - Session data is generally stored in InProcMode, SQL Server Mode, etc. In case we want to store using any other techniques, we use the custom mode.

#### Application

It is a server-side management state and is also known as the application level state management. This is mainly used to store user activity in server memory and application events.

This is further classified into three types :

Application start - The event begins with the start of the domain.

Example

Void ApplicationStart(object sender, EventArgs e)

{

   Application["ApplicationstartMessage"] = "Welcome to Simplilearn";

}

Application error - This is used to manage/handle an error or exception that had been previously unhandled.

Example

void ApplicationError(object sender, EventArgs e)

{

   // Unhandled exception code block

}

Application end - Whenever the domain ends, this ends as well.

Example

Void ApplicationEnd(object sender, EventArgs e)

{

   Application["ApplicationEndMessage"] = "Applications are Closed";

}

### Client-Side State Management

#### Cookie

One of the smallest but important parts of the ASP NET, it is used to store the session and application information of the user. It can be constant and temporary and works with browser requests. The server can read these cookies from the client-side and perform data abstraction.

There are two types of cookies that are available -

Persistence - The Persistence cookie works along with Time and Date.

Response.Cookies["CookiesName"].Expires = DateTime.Today.AddHours(2);

Non-Persistence - Temporary cookie created with application access and closed application is discarded.

#### Viewstate

It is used to manage page-level state and is used for storing, sending, and receiving information.

We can store small values in Viewstate, but it is pretty easy to apply since it does not require any server resources. The page using Viewstate becomes heavy if more data is stored.

##### **Example**

if (ViewState["User\_Name"] != null)

lblName.Text = ViewState["User\_Name"].ToString();

#### Query String

The query string is used to store the value in the URL.

##### **Example**

Response.Redirect("ShowStringValue.aspx?Username=" + txtUsername.Text);

##### **Output**

Query_String

**Viewstate: Example**

**Viewstate and Querystring**

public partial class WebForm1 : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

if(!IsPostBack)

{

ViewState["name"] = "Hello";

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

Label1.Text = ViewState["name"].ToString();

}

protected void Button2\_Click(object sender, EventArgs e)

{

Response.Redirect("webform2.aspx?id="+TextBox1.Text);

}

}

**Cookies**

protected void Page\_Load(object sender, EventArgs e)

{

// Label1.Text = Request.QueryString["id"].ToString();

}

//write cookies

protected void Button1\_Click(object sender, EventArgs e)

{

Response.Cookies["my"].Value = TextBox1.Text;

Response.Cookies["my"].Expires = DateTime.Now.AddSeconds(2);

}

//Read cookies

protected void Button2\_Click(object sender, EventArgs e)

{

if (Request.Cookies["my"] != null)

{

Label1.Text = Request.Cookies["my"].Value.ToString();

}

else

{

Label1.Text = "NoUser";

}

}

//delete cookies

protected void Button3\_Click(object sender, EventArgs e)

{

Response.Cookies["my"].Expires = DateTime.Now;

}

}

}

