SSBT's Art, Commerce & Science College Bambhori, Jalgaon

Bachelor of Computer Applications

LAB MANUAL

Class : SYBCA **Semester : IV**

Subject : Lab on Web Development Technology-II

`Academic Year : 2023-2024

Name Of Faculty : Ms.Vaishnavi . U. Suryavanshi

SSBT's Art's Commerce and Science College , JalgaonDepartment of Computer Applications

Practical: 01

DOP: **DOC:**

Title: Create an ASP .NET web application demonstrating Code behind and ASP.NET Page events.

Objective : To understand the concept of code-behind in ASP.NET web applications.
To learn about various ASP.NET page events and their significance in the application lifecycle.

Theory :

ASP.NET Code Behind:

- ASP.NET allows the separation of presentation markup (HTML) and server-side code using a model known as code-behind. In this model, the HTML markup resides in the .aspx file, while the server-side logic is placed in a separate class file with a .cs or .vb extension, known as the code-behind file.

ASP.NET Page Events:

- ASP.NET provides a set of page events that occur during the lifecycle of a web page. These events enable developers to respond to various stages of the page lifecycle, such as initialization, loading, rendering, and unloading.
- Code-Behind and Page Events in Action:
- In this lab, students will learn how to create an ASP.NET web application using code-behind and utilize page events effectively. They will create a simple web form with HTML markup in the .aspx file and implement server-side logic in the code-behind file.

Code-Behind and Page Events in Action:

- In this lab, students will learn how to create an ASP.NET web application using code-behind and utilize page events effectively. They will create a simple web form with HTML markup in the .aspx file and implement server-side logic in the code-behind file.

Code:

```
public partial class PageLifeCycle : System.Web.UI.Page
{
   protected void Page_PreInit(object sender, EventArgs e)
   {
      Response.Write("<br/>" + "PreInit");
   }

   protected void Page_Init(object sender, EventArgs e)
   {
      Response.Write("<br/>" + "Init");
   }

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

```
        Response.Write("<br/>" + "InitComplete");
    }

    protected override void OnPreLoad(EventArgs e)
    {
        Response.Write("<br/>" + "PreLoad");
    }

    protected void Page_Load(object sender, EventArgs e)
    {
        Response.Write("<br/>" + "Load");
    }

    protected void Page_LoadComplete(object sender, EventArgs e)
    {
        Response.Write("<br/>" + "LoadComplete");
    }

    protected override void OnPreRender(EventArgs e)
    {
        Response.Write("<br/>" + "PreRender");
    }

    protected override void OnSaveStateComplete(EventArgs e)
    {
        Response.Write("<br/>" + "SaveStateComplete");
    }

    protected void Page_UnLoad(object sender, EventArgs e)
    {
        // Runtime Error: Response is not available in this context.
        // Response.Write("<br/>" + "UnLoad"); // Error
    }
}
```

Output :

PreInit
Init
InitComplete
PreLoad
Load
LoadComplete
PreRender
SaveStateComplete

Conclusion:

Submitted By :

Sign :

Name :

Roll No :

Checked By :

Ms.Vaishnavi .U.Suryavanshi

SSBT's Art's, Commerce and Science College, JalgaonDepartment of Computer Applications

Practical: 02

DOP: **DOC:**

Title: Create an ASP .NET web application to demonstrate use of Web Server Controls.

Objective :

To demonstrate the usage and implementation of Web Server Controls in ASP.NET web applications, providing students with practical experience in designing interactive user interfaces, handling user input, and enhancing user experience through dynamic content presentation.

Theory :

- **Web Server Controls in ASP.NET:**
  Web Server Controls are components in ASP.NET that run on the server and render HTML markup to the client browser. They provide a rich set of interactive elements such as TextBoxes, Buttons, Labels, DropDownLists, GridViews, and more.

- **Key Concepts:**
  Web Server Controls vs. HTML Controls:
  Web Server Controls offer more functionality and flexibility compared to HTML controls. They provide server-side events and properties, making them easier to manage and manipulate dynamically.
- **Types of Web Server Controls:**
  ASP.NET offers a wide range of Web Server Controls catering to various requirements, including input controls for user input, data controls for displaying data, validation controls for user input validation, navigation controls for site navigation, and more.
- **Event Handling:**
  Web Server Controls support server-side event handling, allowing developers to respond to user actions such as button clicks, text changes, selection changes, etc. This enables the creation of interactive and responsive web applications.
- **Properties and Methods:**
  Web Server Controls expose properties and methods that can be accessed programmatically on the server-side code, enabling dynamic manipulation of control behavior, appearance, and content.
- **Control Styling and Customization:**
  ASP.NET provides ways to customize the appearance and behavior of Web Server Controls using CSS, themes, and control templates, allowing developers to create visually appealing and consistent user interfaces.

Code:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="web_server_control.WebForm1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <h1> Webform Controls</h1>
   <h2>STUDENT REGISTRATION</h2>
   <form id="form1" runat="server">
      <div>
      </div>
      first name:<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
      <p>
         last name:<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
      </p>
      Email id:<asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>

      <p>
         Gender:<asp:RadioButton ID="RadioButton1" runat="server" Text="male" />
           
         <asp:RadioButton ID="RadioButton3" runat="server" Text="female" />
      </p>
      <p>
         <asp:Button ID="Button1" runat="server" OnClick="Button1_Click1" Text="submit" />
      </p>
      <p>
         <asp:Label ID="Label1" runat="server"></asp:Label>
      </p>
      <p>
         <asp:Label ID="Label2" runat="server" Text="Label"></asp:Label>
      </p>
      <p>
         <asp:Label ID="Label3" runat="server" Text="Label"></asp:Label>
      </p>
      <p>
         <asp:Label ID="Label5" runat="server" Text="Label"></asp:Label>
      </p>
   </form>
</body>
</html>
```

- **C# code**

```
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace web_server_control
{
   public partial class WebForm1 : System.Web.UI.Page
   {
      protected void Page_Load(object sender, EventArgs e)
      {

      }

      protected void Button1_Click1(object sender, EventArgs e)
      {
         string fname = TextBox1.Text;
         string lname = TextBox2.Text;
         string email = TextBox3.Text;
         String gender = " ";
         if(RadioButton1.Checked)
         {
            gender = RadioButton1.Text;
         }
         else
         {
            gender = RadioButton3.Text;
         }
         Label1.Text = "first name=" + fname;
         Label2.Text = "last name=" + lname;
         Label3.Text = "email id=" + email;
         Label5.Text = "Gender=" + gender;
      }
   }
}
```

Output:

Webform Controls

STUDENT REGISTRATION

first name: vaishnavi

last name: suryavansh

Email id: abc123@g

Gender: male female

submit

first name=vaishnavi

last name=suryavanshi

email id=abc123@gmail.com

Gender=female

Conclusion: _____

Submitted By : **Checked By :**

Sign :

Name : **Ms. Vaishnavi . U. Suryavanshi**

Roll No:

SSBT's Art's Commerce and Science College , JalgaonDepartment of Computer Applications

Practical: 01

DOP: **DOC:**

Title: Create an ASP .NET web application to demonstrate use of Validation Controls.

Objective : To demonstrate the implementation and utilization of Validation Controls in ASP.NET web applications, enabling students to understand how to perform client-side and server-side validation of user input, enhance data integrity, and improve user experience by providing feedback on invalid data entries.

Theory :

- **Validation Controls in ASP.NET:**

Validation Controls are a set of pre-built components in ASP.NET that facilitate the validation of user input on both the client and server sides. They ensure that data entered by users meets specific criteria, such as format, range, or presence, before it is submitted to the server for processing.

Key Concepts:

- **Client-Side Validation:**

Validation controls in ASP.NET can perform validation logic on the client side using JavaScript, enhancing user experience by providing immediate feedback to users without requiring a round trip to the server.

- **Server-Side Validation:**

Validation controls can also perform validation logic on the server side to ensure data integrity and security, especially for critical operations or when client-side validation fails.

- **Validation Summary:**

ASP.NET includes the ValidationSummary control, which aggregates error messages from multiple validation controls and displays them in a centralized location, providing a concise summary of validation errors to the user.

- **Error Display:**

Validation controls can display error messages next to the validated input controls or in a centralized location, offering flexibility in error message presentation.

- **Custom Validation:**

ASP.NET allows developers to create custom validation logic using the CustomValidator control, enabling validation based on custom business rules or complex criteria.

Client-Side Scripting: Validation controls generate client-side JavaScript code to perform validation logic on the client browser, reducing server load and enhancing responsiveness.

Code:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication6.WebForm1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
      <div>
         <asp:Label ID="Label1" runat="server" Text="Name"></asp:Label>
         :   
         <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
           
         <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
ControlToValidate="TextBox1" ErrorMessage="Please Enter The Name"
ForeColor="Red"></asp:RequiredFieldValidator>
         <br />
         <br />
         <asp:Label ID="Label3" runat="server" Text="Email"></asp:Label>
 :   
         <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
    
         <asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server"
ControlToValidate="TextBox2" ErrorMessage="Enter The Email"
ForeColor="Blue"></asp:RequiredFieldValidator>
         <br />
         <asp:Label ID="Label2" runat="server" Text=" "></asp:Label>
      </div>
      <asp:Button ID="Button1" runat="server" Text="Submit  " />
   </form>
</body>
</html>
```

Output:

Name : Please Enter The Name

Email : Enter The Email

Submit

Conclusion:

Submitted By :

Sign :

Name :

Roll No :

Checked By :

Ms.Vaishnavi . U. Suryavanshi

SSBT's Art's Commerce and Science College , JalgaonDepartment of Computer Applications

Practical: 04

DOP: **DOC:**

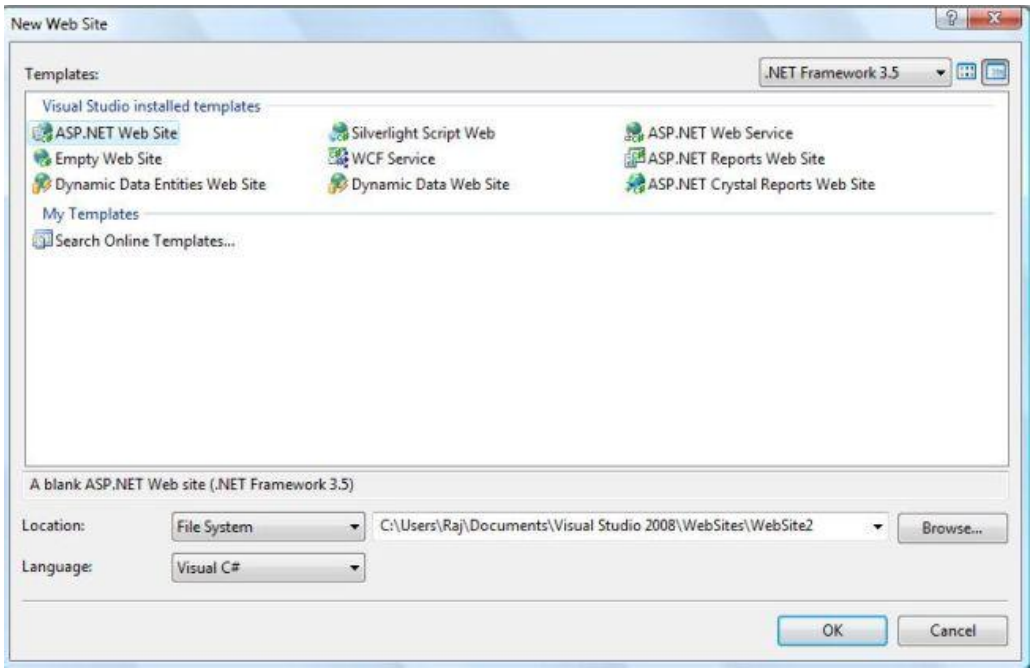
Title: Create an ASP .NET Web application to demonstrate AdRotator Control.

Objective : To introduce AdRotator Controls.

Theory :

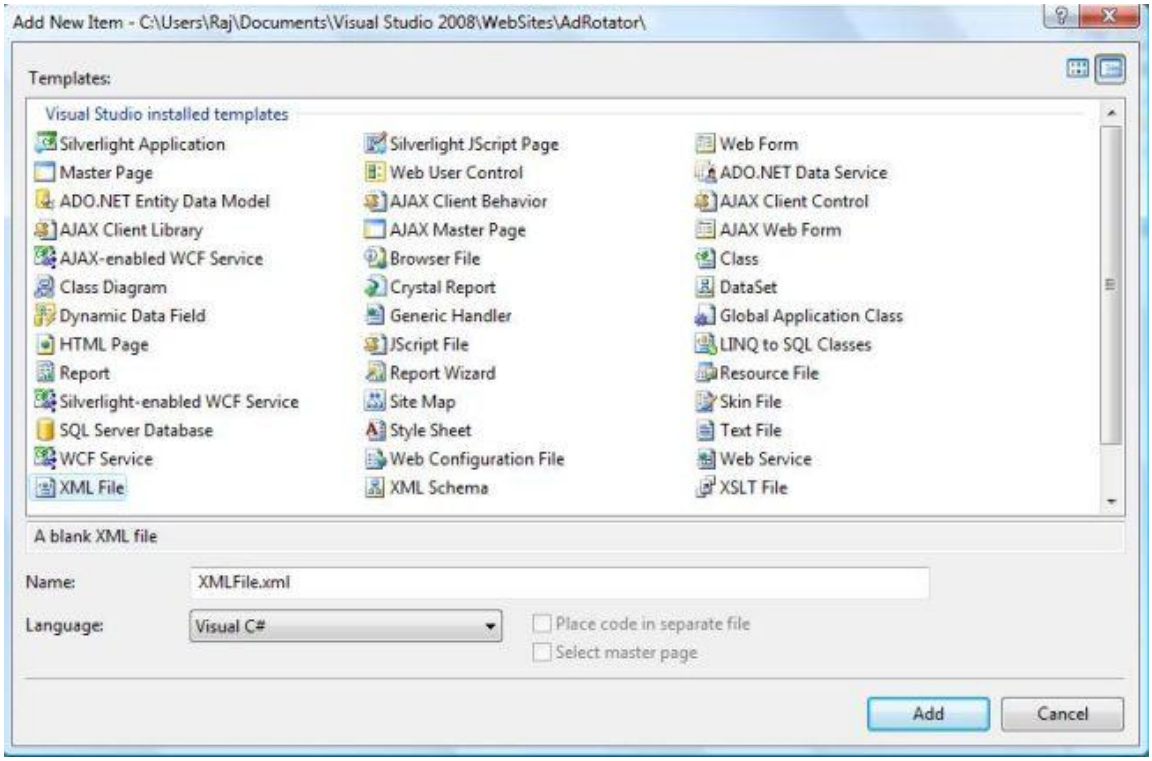
ASP.NET Website Project

First of all, start Visual Studio and create a new ASP.NET Web Site. I'm using Visual Studio 2008. If you're using a newer version of Visual Studio, just create an ASP.NET Website using a Web Form or MVC.



2) Create the Ad XML file

Add a new XML File using Add New Item in Visual Studio. Right click on Visual Studio project name in Solution Explorer, Add New Item, and select XML file.



Code:
Add the following code or similar code in the XML. The root node of the XML file is Advertisement and it has multiple Ad tags. Each Ad tag has an ImageURL, NavigateUrl, AlternateText, Impressions, Keyword, and Category.

```
<?xml version="1.0" encoding="utf-8" ?>
<Advertisements>
    <Ad>
        <ImageUrl>~/Banners/468X60_add.gif</ImageUrl>
        <NavigateUrl>http://www.Mindcracker.com/Registration/Register.aspx</NavigateUrl>
        <AlternateText>Advertisement</AlternateText>
        <Impressions>100</Impressions>
        <Keyword>Header</Keyword>
        <Category>Script</Category>
    </Ad>
    <Ad>
        <ImageUrl>~/Banners/468X60_Consultant.gif</ImageUrl>
        <NavigateUrl>http://www.mindcracker.com/Consultants/</NavigateUrl>
        <AlternateText>Advertisement</AlternateText>
        <Impressions>100</Impressions>
        <Keyword>Header</Keyword>
        <Category>Script</Category>
    </Ad>
    <Ad>
        <ImageUrl>~/Banners/468X60_Employer.gif</ImageUrl>
        <NavigateUrl>http://www.Mindcracker.com/Registration/Register.aspx</NavigateUrl>
        <AlternateText>Advertisement</AlternateText>
        <Impressions>100</Impressions>
        <Keyword>Header</Keyword>
```

```
        <Category>Script</Category>
    </Ad>
    <Ad>
        <ImageUrl>~/Banners/468X60_Jobseeker.gif</ImageUrl>
        <NavigateUrl>http://www.mindcracker.com/Jobs/</NavigateUrl>
        <AlternateText>Advertisement</AlternateText>
        <Impressions>100</Impressions>
        <Keyword>Header</Keyword>
        <Category>Script</Category>
    </Ad>
    <Ad>
        <ImageUrl>~/Banners/468X60_Recruiter.gif</ImageUrl>
        <NavigateUrl>http://www.Mindcracker.com/Registration/Register.aspx</NavigateUrl>
        <AlternateText>Advertisement</AlternateText>
        <Impressions>100</Impressions>
        <Keyword>Header</Keyword>
        <Category>Script</Category>
    </Ad>
</Advertisements>
```

- **Bind XML File**

There are two ways to bind the XML file on a page.

```
<asp:XmlDataSource ID="XmlDataSourceAd" runat="server"
DataFile="~/Advertise.xml">
</asp:XmlDataSource>
<asp:AdRotator ID="AdRotatorHeader" runat="server" KeywordFilter="Header"
OnAdCreated="AdRotatorHeader_AdCreated" Target="_blank" DataSourceID="XmlDataSourceAd"/>
<asp:Label ID="LabelHeaderScript" runat="server" Visible="False"></asp:Label>
protected void AdRotatorHeader_AdCreated(object sender, AdCreatedEventArgs e)
{
try
{
if (e.AdProperties["Category"].ToString() == "Script")
{
LabelHeaderScript.Text = e.AdProperties["ImageUrl"].ToString();
LabelHeaderScript.Visible = false;
AdRotatorHeader.Visible = true;
}
}
catch (Exception ex)
{
string str = ex.Message;
AdRotatorHeader.Visible = false;
}
}
```

Output :

Example

Conclusion:_____

_____

Submitted By : **Checked By :**

Sign :

Name : **Ms.Vaishnavi .U.Suryavanshi**

Roll No :

SSBT's Art's, Commerce and Science College, JalgaonDepartment of Computer Applications

Practical: 05

DOP: **DOC:**

Title: : Create an ASP .NET web application to demonstrate use of global.asax file.

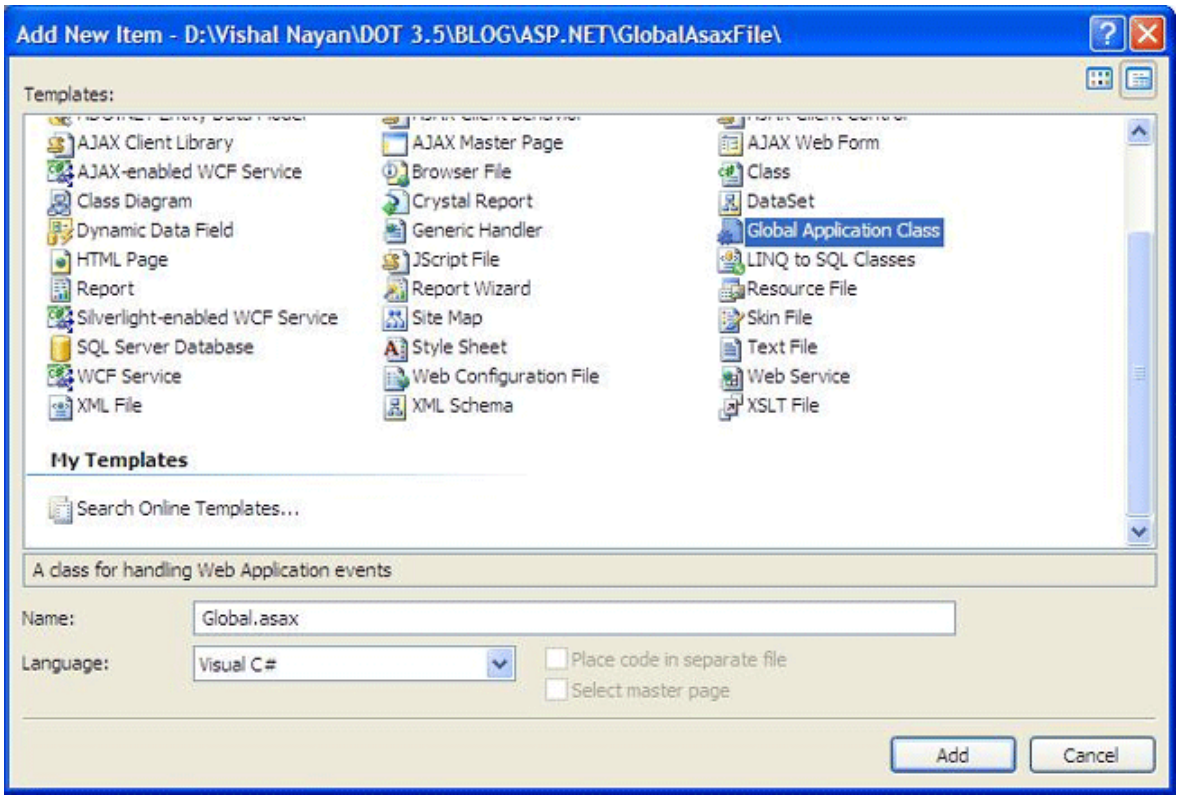
Objective : The objective of creating an ASP.NET web application with a global.asax file is to demonstrate the use of global application-level events and methods .

Theory :

The Global.asax file in an ASP.NET web application provides a way to handle application-level events and define application-wide logic. It acts as a global handler for events that occur throughout the application's lifecycle, such as application start, end, session start, session end, and more. This file allows developers to write custom code to respond to these events and perform tasks like initializing resources, managing sessions, logging, and error handling.

- **How to add global.asax file:**

Select Website >>Add New Item (or Project >> Add New Item if you're using the Visual Studio web project model) and choose the Global Application Class template.

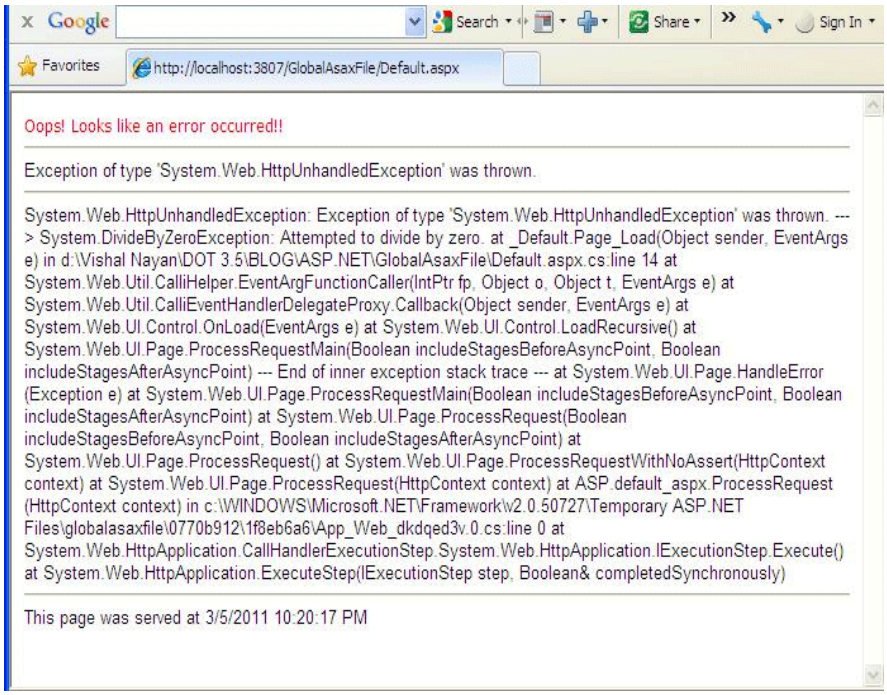


Code:

- **Application_Error**

```
protected void Application_Error(Object sender, EventArgs e)
{
   Response.Write("<font face=\"Tahoma\" size=\"2\" color=\"red\">");
   Response.Write("Oops! Looks like an error occurred!!<hr></font>");
   Response.Write("<font face=\"Arial\" size=\"2\">");
   Response.Write(Server.GetLastError().Message.ToString());
   Response.Write("<hr>" + Server.GetLastError().ToString());
   Server.ClearError();
}
```

Output:

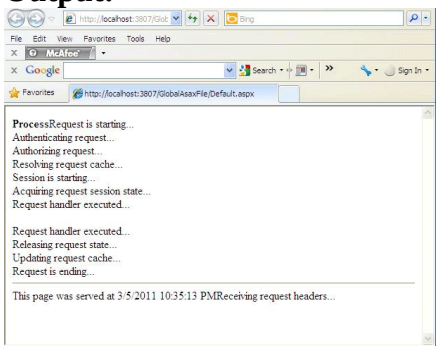


our own authentication codes, because this method is called just before authentication is performed.

```
protected void Application_AuthenticateRequest(Object sender, EventArgs e)
{
   Response.Write("Authenticating request...<br>");
   bool cookieFound = false;
   HttpCookie authCookie = null;
   HttpCookie cookie;
   for (int i = 0; i < Request.Cookies.Count; i++)
   {
      cookie = Request.Cookies[i];
      if (cookie.Name == FormsAuthentication.FormsCookieName)
      {
```

```
            cookieFound = true;
            authCookie = cookie;
            break;
        }
    }
    // If the cookie has been found, it means it has been issued from either
    // the windows authorisation site, is this forms auth site.
    if (cookieFound)
    {
        // Extract the roles from the cookie, and assign to our current principal, which is attached to the
        // HttpContext.
        FormsAuthenticationTicket winAuthTicket = FormsAuthentication.Decrypt(authCookie.Value);
        string[] roles = winAuthTicket.UserData.Split(';');
        FormsIdentity formsId = new FormsIdentity(winAuthTicket);
        System.Security.Principal.GenericPrincipal princ = new
System.Security.Principal.GenericPrincipal(formsId, roles);
        HttpContext.Current.User = princ;
    }
    else
    {
        // No cookie found, we can redirect to the Windows auth site if we want, or let it pass through so
        // that the forms auth system redirects to the logon page for us.
    }
}
```

Output:



Conclusion: _____

_____

Submitted By : **Checked By :**

Sign :

Name : **Ms. Vaishnavi.U.Suryavanshi**

Roll No:

SSBT's Art's Commerce and Science College , JalgaonDepartment of Computer Applications

Practical: 06

DOP: **DOC:**

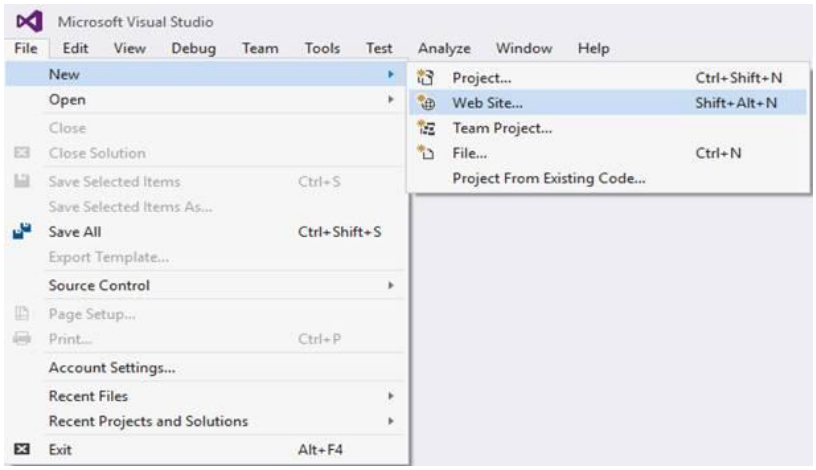
Title: Create a Master Page and ASP.NET Content pages using Master Page.

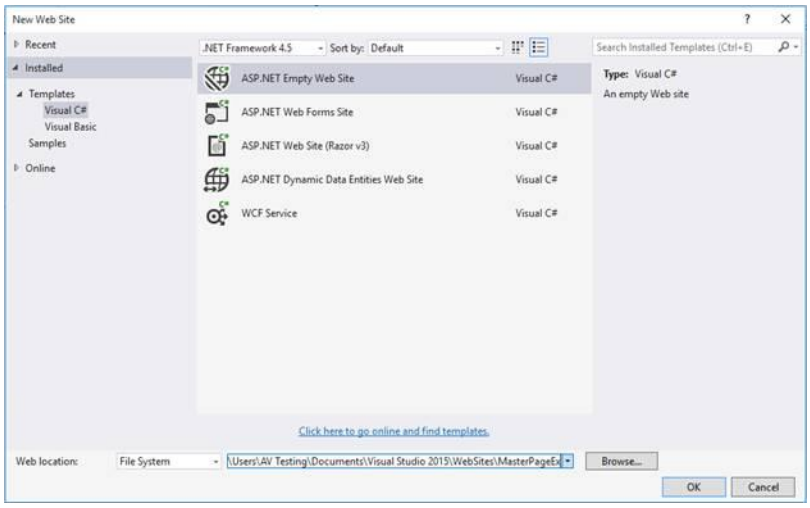
Objective : The objective is to create a Master Page and ASP.NET Content Pages using that Master Page in order to establish a consistent layout and structure across multiple pages within an ASP.NET web application.

Theory :

- **Step 1**

- **Create new project in Visual Studio**
  Go to File-> New-> Web Site-> Visual C#->ASP.NET Empty Website-> Entry Application Name-> OK.

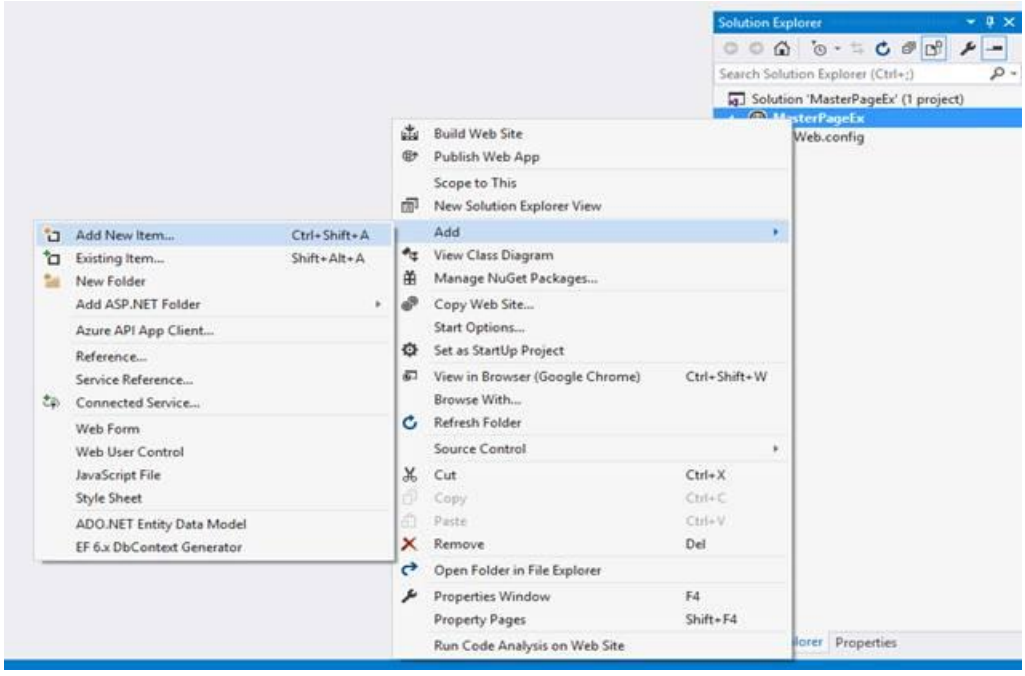


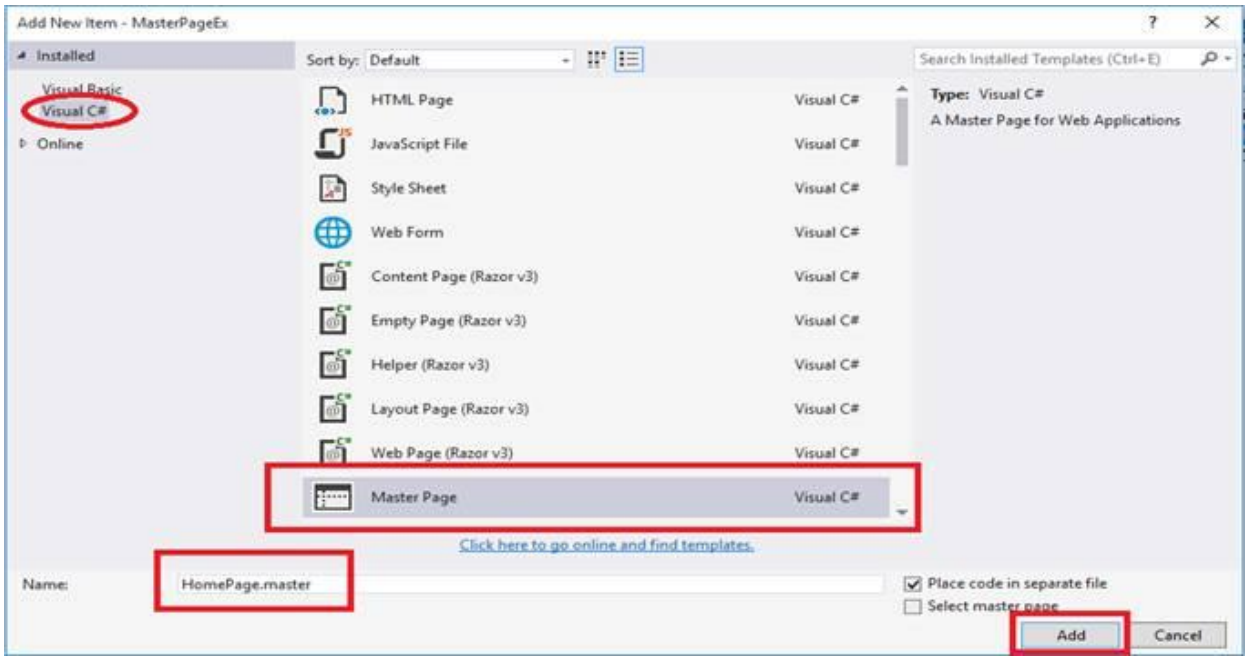


Step 2

Create Master Page

- Project name-> Add-> Add New Item->Visual C#-> Master Page->Write Master Page Name-> Add.





- HomePage.master master page is created.

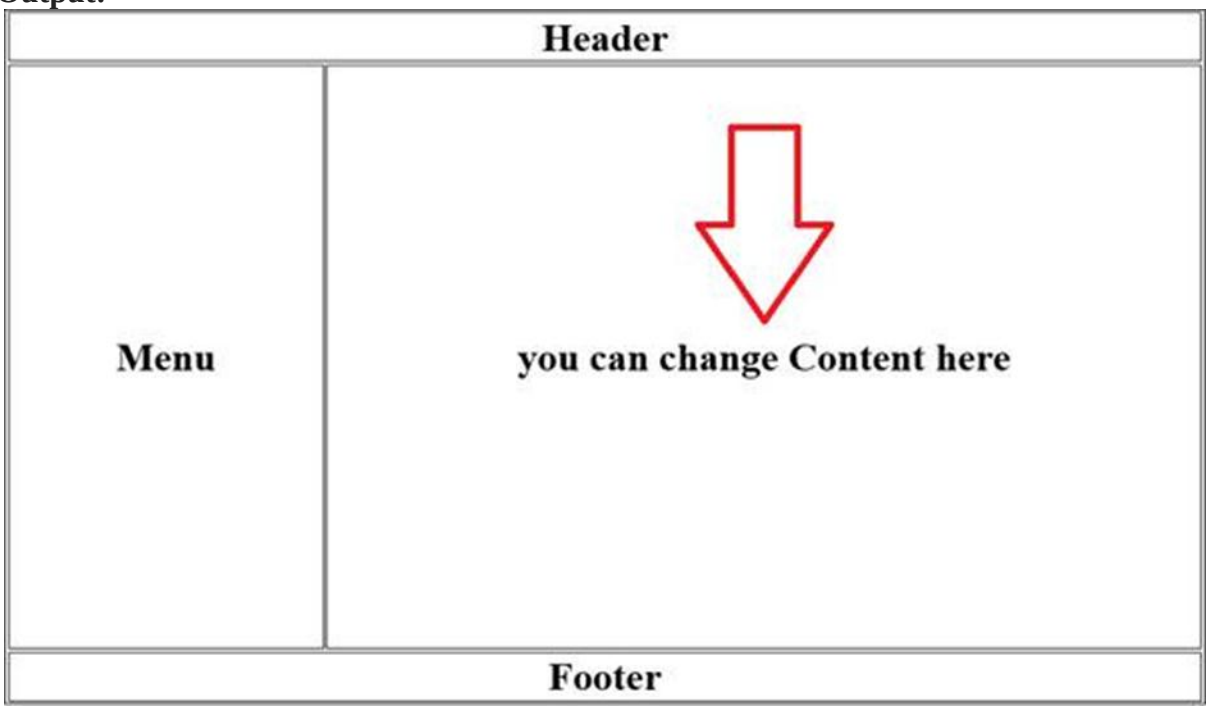
- **Code**:

<%@ Master Language="C#" AutoEventWireup="true" CodeFile="HomePage.master.cs" Inherits="HomePage" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"> head runat="server">
<title></title>
</head>

```
<body>
  <form id="form1" runat="server">
    <div>
      <table border="1">
        <tr>
          <td colspan="2" style="text-align: center">
            <h1>Header</h1>
          </td>
        </tr>
        <tr>
          <td style="text-align: center; height: 480px; width: 250px">
            <h1>Menu</h1>
          </td>
          <td style="text-align: center; height: 480px; width: 700px">
            <asp:ContentPlaceHolder ID="MainContentPlaceHolder1" runat="server">
              <h1>you can change Content here</h1>
            </asp:ContentPlaceHolder>
          </td>
        </tr>
        <tr>
          <td colspan="2" style="text-align: center">
            <h1>Footer</h1>
          </td>
        </tr>
      </table>
    </div>
  </form>
</body>
</html>
```

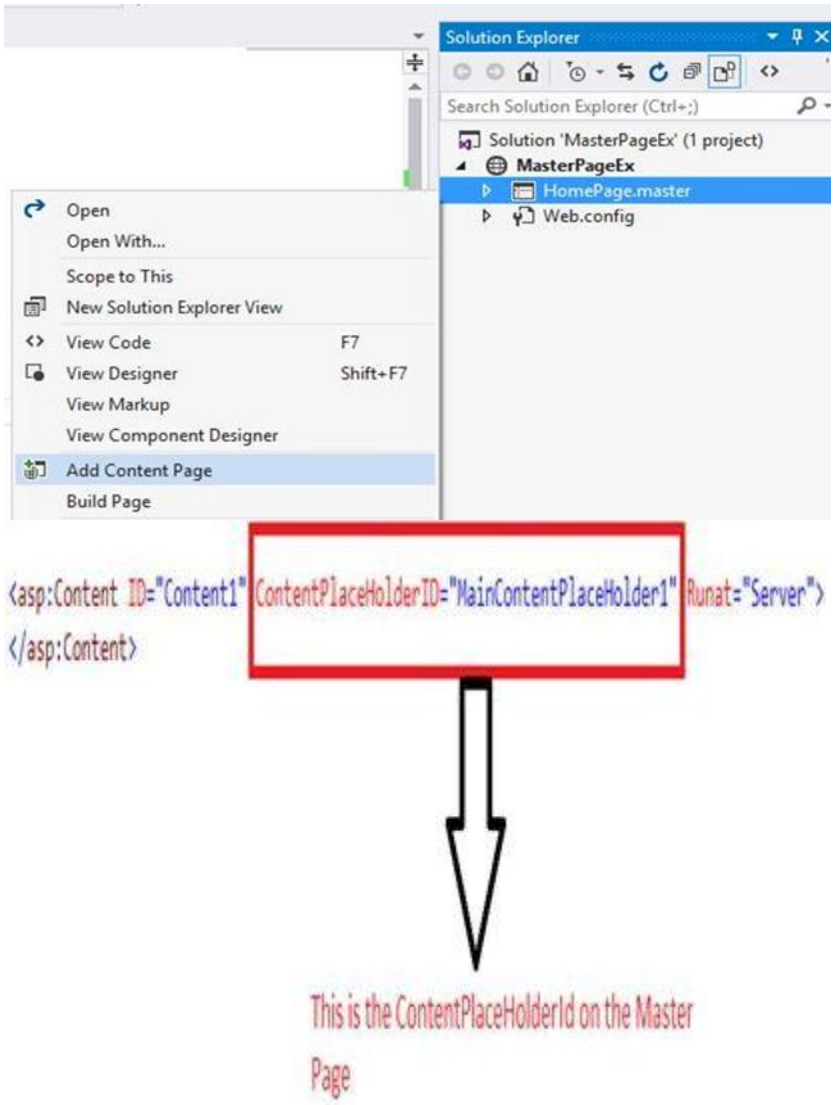
Go to the design mode and you will see the image, as shown below.

Output:



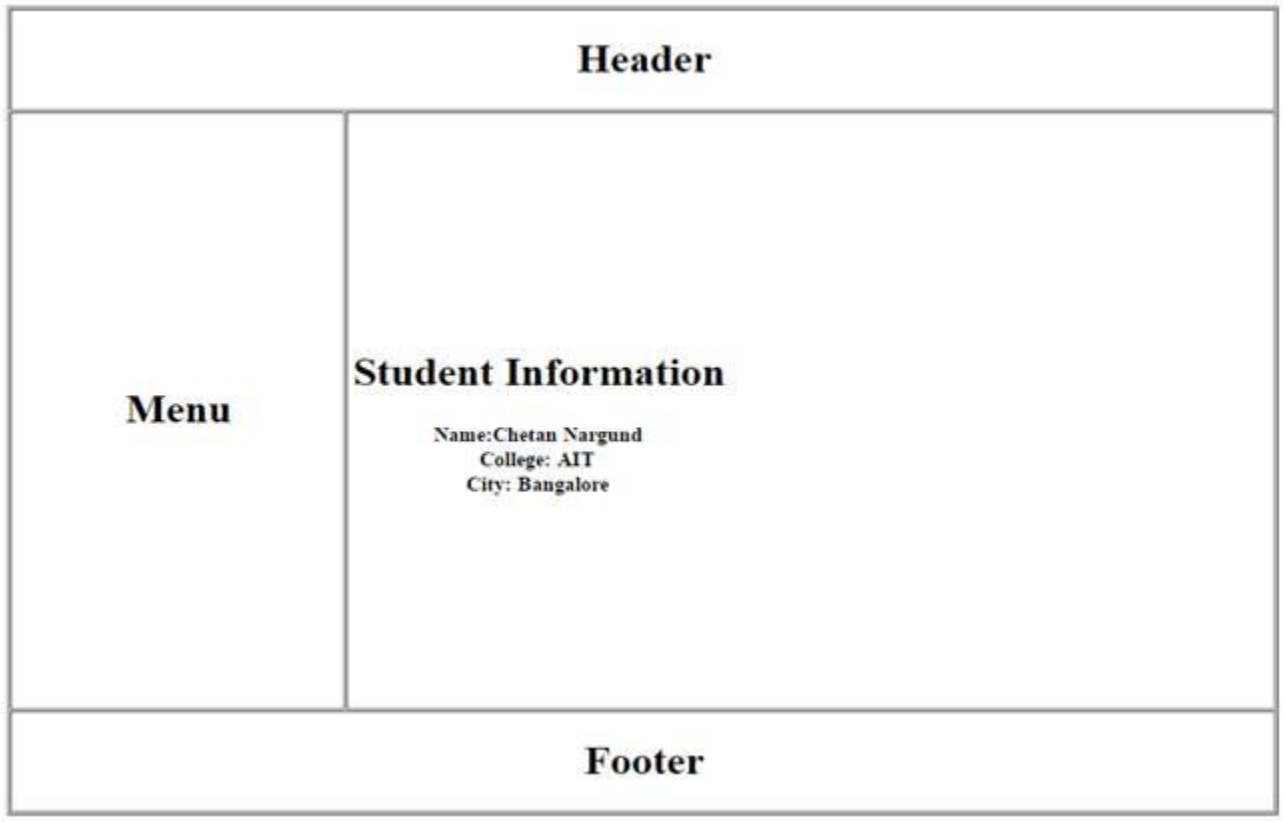
Adding the Content Pages to Master Page

Master Page -> Add Content Page.



write your code inside Content Placeholder.See the below code.
<%@ Page Title="" Language="C#" MasterPageFile="~/HomePage.master" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %> < asp: Content ID = "Content1" ContentPlaceHolderID = "MainContentPlaceHolder1"
runat = "Server" > < table > < tr > < th > < h1 > Student Information < /h1> < /th> < /tr> < tr > < td > < b > Name: Chetan Nargund < /b> </td > < /tr> < tr > < td > < b > College: AIT < /b> </td > < /tr> < tr > < td > < b > City: Bangalore < /b></td > < /tr> < /table> < /asp:Content>

Output:


Header
Menu
Student Information
Name:Chetan Nargund
College: AIT
City: Bangalore
Footer

Conclusion:

Submitted By : **Checked By :**

Sign :

Name : **Ms.Vaishnavi . U. Suryavanshi**

Roll No :

SSBT's Art's, Commerce and Science College, Jalgaon Department of Computer Applications

Practical: 07

DOP: **DOC:**

Title: Write an ASP .net program that demonstrates use of Intrinsic Objects.

Objective :

The objective is to create an ASP.NET program that demonstrates the use of Intrinsic Objects, including Request, Response, Session, and Server, to interact with various aspects of the web application environment.

Theory :

In ASP.NET, Intrinsic Objects are predefined objects that provide access to various aspects of the web application environment. These objects are readily available for use without the need for explicit instantiation or declaration. The key intrinsic objects include:

- **Request:** Represents the current HTTP request being processed by the web server. It provides access to request parameters, form data, cookies, headers, and other request-related information.
- **Response:** Represents the HTTP response that the web server sends back to the client. It allows developers to send content, headers, and status codes to the client browser.
- **Session:** Provides a way to store and retrieve user-specific data across multiple requests and pages during a user session. It enables state management in web applications and helps maintain user state between page visits.
- **Server:** Represents the web server itself and provides access to server-specific properties and methods. It allows developers to perform tasks such as executing server-side scripts, redirecting requests, and accessing application-level variables.

Code:

Aspx File

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="IntrinsicObjectsDemo.Default" %>

<!DOCTYPE html>

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

```
        <h1>User Information</h1>
        <p>IP Address: <% Response.Write(Server.HtmlEncode(Request.UserHostAddress)); %></p>
        <p>Browser: <% Response.Write(Server.HtmlEncode(Request.Browser.Browser)); %> <% Response.Write(Server.HtmlEncode(Request.Browser.Version)); %></p>
      </div>
    </form>
</body>
</html>
```

- **C# Code:**

```
using System;

namespace IntrinsicObjectsDemo
{
    public partial class Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            // You can perform additional logic here if needed
        }
    }
}
```

Output

User Information

IP Address: 127.0.0.1
Browser: Mozilla 5.0

Conclusion:_____

_____

Submitted By : **Checked By :**

Sign :

Name : **Ms. Vaishnavi .U .Suryavanshi**

Roll No:

SSBT's Art's, Commerce and Science College, Jalgaon Department of Computer Applications

.

Practical: 08

DOP: **DOC:**

Title: Create an ASP .NET web application to demonstrate use of session and cookies.

Objective : The objective is to create an ASP.NET web application to demonstrate the use of session and cookies for managing user state across multiple requests. Session and cookies are used to store and retrieve user-specific data, such as user preferences, shopping cart items, or login credentials, throughout a user's visit to the website..

Theory :

Session and cookies are essential features in web development for maintaining user state across multiple requests and pages within an application.

- **Session:**
  Sessions are used to store user-specific data on the server-side throughout a user's visit to a website.
  Each user is assigned a unique session ID, which is typically stored as a cookie on the client-side.
  Session data can be accessed and modified across different pages of the application during the user's session.
  Sessions are ideal for storing sensitive or temporary data, such as user authentication tokens, shopping cart contents, or user preferences.
- **Cookies:**
  Cookies are small pieces of data sent by the server to the client's browser and stored on the client's machine.
  Cookies can be used to store information such as user preferences, login credentials, or tracking data.
  Cookies are sent back to the server with every subsequent request, allowing the server to recognize and track the user.
  Cookies can have expiration dates, domain restrictions, and security settings controlled by the server.
  They are commonly used for personalization, tracking user behavior, and implementing features like "Remember Me" login functionality.
  Code:
- **Cookies Code:**

Code for Default.aspx design page

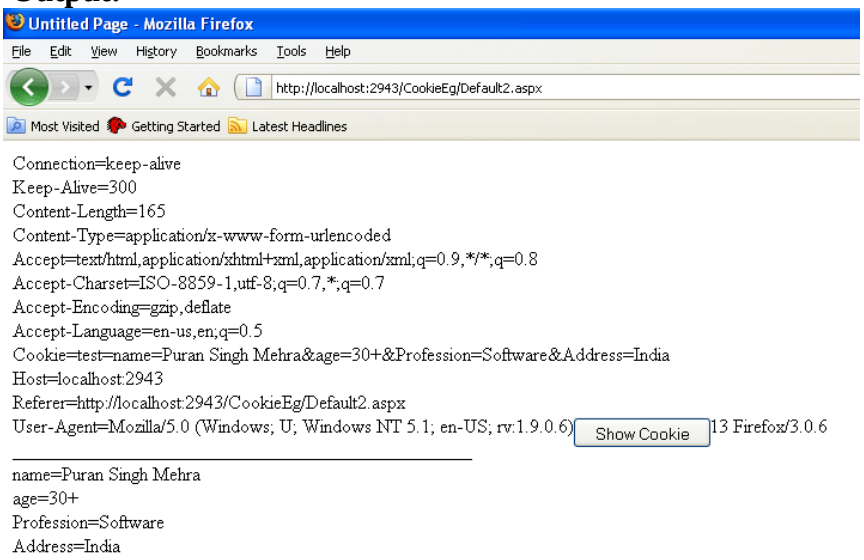
```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default2.aspx.cs"
Inherits="Default2" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title>Untitled Page</title>
```

```
</head>
<body>
   <form id="form1" runat="server">
   <div>
      <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Style="top: 231px;
         left: 476px; position: absolute; height: 26px; width: 116px" Text="Show Cookie" />
   </div>
   </form>
</body>
</html>
```

- **C# Code:**

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

public partial class Default2 : System.Web.UI.Page
{
   protected void Page_Load(object sender, EventArgs e)
   {
      foreach (string str in Request.Headers)
      {
         Response.Write(str + "=" + Request.Headers[str] + "<br>");
      }
      Response.Write("_____" + "<br>");
   }

   protected void Button1_Click(object sender, EventArgs e)
   {
      foreach (string str in Request.Cookies["test"].Values)
      {
         Response.Write(str + "=" + Request.Cookies["test"].Values[str] + "<br>");
      }
   }
}
```

Output:



- **Session Code:** we will add code on the button click.

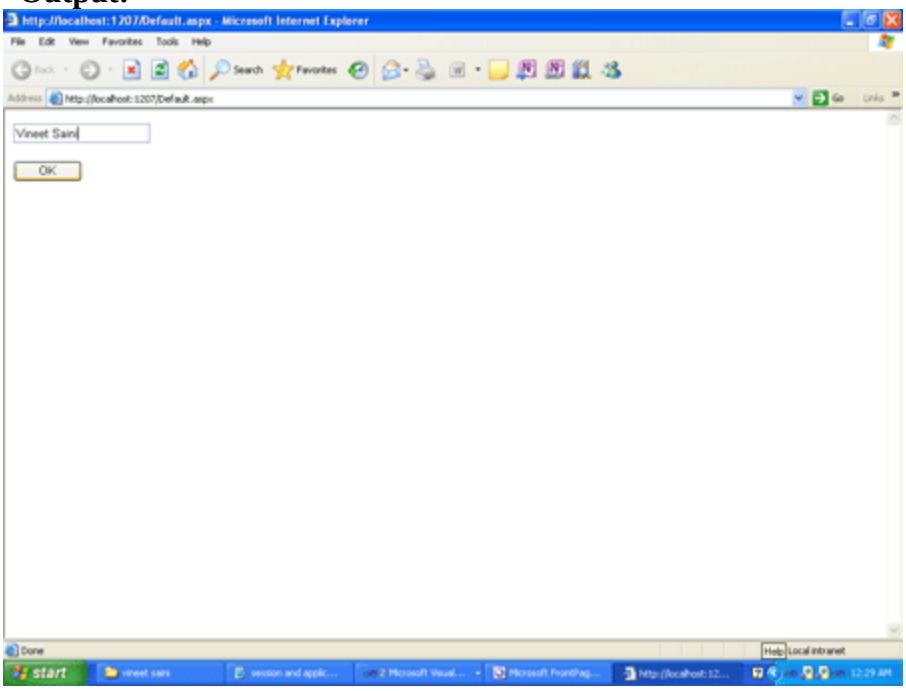
```
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace session
{
   public partial class _Default : System.Web.UI.Page
   {
      protected void Page_Load(object sender, EventArgs e)
      {
      }
      protected void Button1_Click(object sender, EventArgs e)
      {
         Session["username"] = TextBox1.Text;
         Response.Redirect("WebForm1.aspx");
      }
   }
}
```

Now create a label on this web form, then add the following code on web form loading.

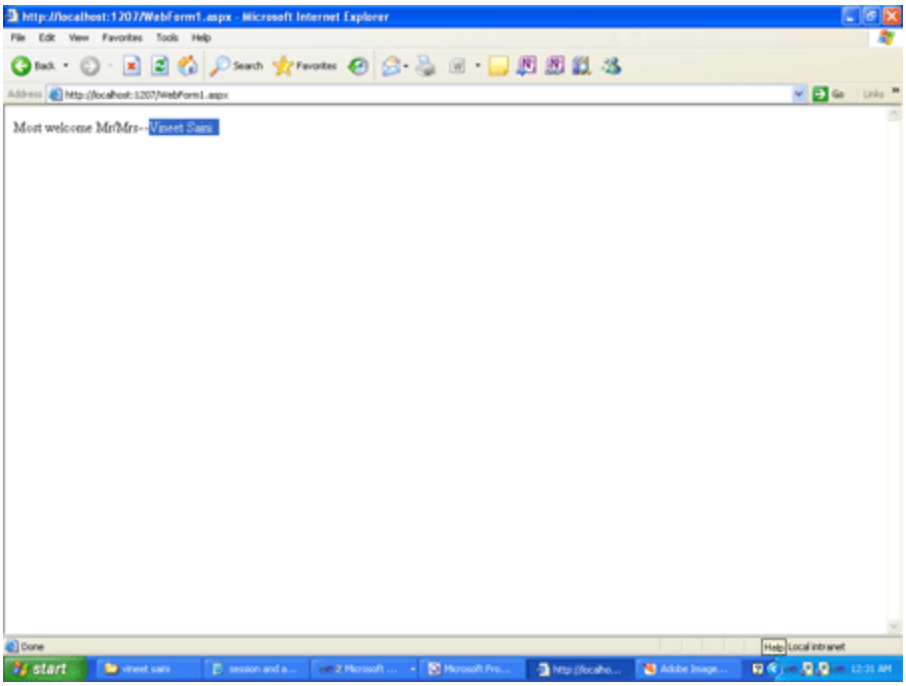
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace session
{
   public partial class WebForm1 : System.Web.UI.Page
   {
```

```
        protected void Page_Load(object sender, EventArgs e)
        {
            Label1.Text = "Most welcome Mr./Mrs--" + Session["username"].ToString();
        }
    }
}
```

Output:



Then click the ok button.



Conclusion:

Submitted By :

Sign :

Name :

Roll No :

Checked By :

Ms. Vaishnavi . U. Suryavanshi

SSBT's Art's, Commerce and Science College, JalgaonDepartment of Computer Applications

Practical: 09

DOP: **DOC:**

Title: Create an ASP .NET Web application to retrieve data from database using wizard.

Objective : The objective is to create an ASP.NET Web application that utilizes a wizard control to guide users through a multi-step process for retrieving data from a database.

Theory :
In ASP.NET, a wizard control facilitates the creation of multi-step processes or forms, guiding users through a sequence of steps to accomplish a task or gather information. When combined with data retrieval from a database, a wizard-based web application enables users to interactively navigate through different stages of data selection and presentation.

Key components and concepts involved:

- **Wizard Control**

The ASP.NET Wizard control provides a user interface for dividing complex tasks into sequential steps. Each step typically contains form elements, user input fields, or data displays relevant to the specific stage of the process.
Users can navigate between steps using navigation buttons such as "Next," "Previous," and "Finish."

- **Data Retrieval:**

Data retrieval involves fetching information from a database based on user input or predefined criteria. A data source control (e.g., SqlDataSource) is used to interact with the database and retrieve data according to specified queries or parameters.
Data retrieved from the database can be displayed in various controls such as GridView, DropDownList, or Repeater within the wizard steps.

Code:

```
<head runat="server">
   <title>Data Retrieval Wizard</title>
</head>
<body>
   <form id="form1" runat="server">
      <asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>
      <asp:Wizard ID="Wizard1" runat="server" DisplaySideBar="False"
OnFinishButtonClick="Wizard1_FinishButtonClick">
         <WizardSteps>
            <asp:WizardStep runat="server" Title="Step 1">
               <asp:Label ID="Label1" runat="server" Text="Please select a
category:"></asp:Label><br />
               <asp:DropDownList ID="DropDownList1" runat="server"
DataSourceID="SqlDataSource1" DataTextField="CategoryName"
DataValueField="CategoryID"></asp:DropDownList>
               <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$
ConnectionStrings:YourConnectionString %>"
                  SelectCommand="SELECT [CategoryID], [CategoryName] FROM
```

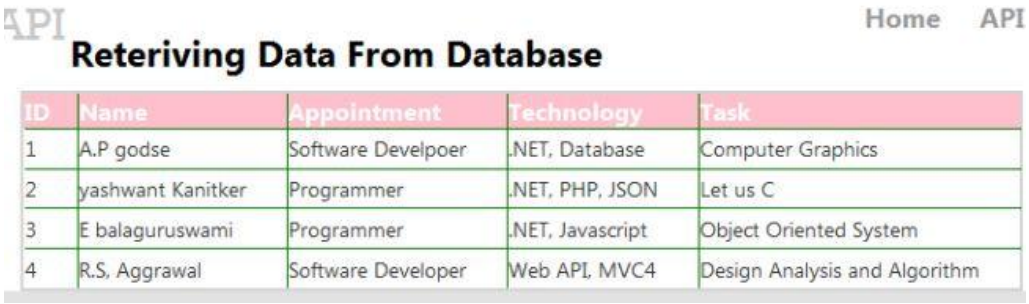
```
[Categories]"></asp:SqlDataSource>
            </asp:WizardStep>
            <asp:WizardStep runat="server" Title="Step 2">
               <asp:Label ID="Label2" runat="server" Text="Please select a
product:"></asp:Label><br />
               <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
DataSourceID="SqlDataSource2">
                  <Columns>
                     <asp:BoundField DataField="ProductName" HeaderText="Product Name" />
                     <asp:BoundField DataField="UnitPrice" HeaderText="Unit Price" />
                  </Columns>
               </asp:GridView>
               <asp:SqlDataSource ID="SqlDataSource2" runat="server" ConnectionString="<%$
ConnectionStrings:YourConnectionString %>"
                  SelectCommand="SELECT [ProductName], [UnitPrice] FROM [Products]
WHERE [CategoryID] = @CategoryID">
                  <SelectParameters>
                     <asp:ControlParameter ControlID="DropDownList1" Name="CategoryID"
PropertyName="SelectedValue" Type="Int32" />
                  </SelectParameters>
               </asp:SqlDataSource>
            </asp:WizardStep>
         </WizardSteps>
      </asp:Wizard>
   </form>
</body>
</html>
```

C# Code :

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="WizardDemo.Default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
using System;

namespace WizardDemo
{
   public partial class Default : System.Web.UI.Page
   {
      protected void Page_Load(object sender, EventArgs e)
      {
         if (!IsPostBack)
         {
            Wizard1.ActiveStepIndex = 0; // Set the active step to the first step
         }
      }

      protected void Wizard1_FinishButtonClick(object sender, WizardNavigationEventArgs e)
      {
```

```
        }
    }
}
```

Output:



Conclusion:_____

_____

Submitted By : **Checked By :**

Sign :

Name : **Ms.Vaishnavi . U . Suryavanshi**

Roll No :

SSBT's Art's, Commerce and Science College, JalgaonDepartment of Computer Applications

Practical: 10

DOP: **DOC:**

Title: Create an ASP .NET application to demonstrate ADO.NET objects (Connection, Command, DataReader, DataSet, DataAdaptor)

Objective : Develop an ASP.NET application to demonstrate the usage of ADO.NET objects (Connection, Command, DataReader, DataSet, DataAdapter) for interacting with a SQL Server database, displaying data retrieved from a database table in a GridView control on a web page

Theory :

ADO.NET (ActiveX Data Objects for .NET) is a set of classes in the .NET Framework that facilitates data access from relational databases, XML, and other data sources. It includes several key objects:

- **Connection:** Represents a connection to a data source like a SQL Server database. It manages the connection to the database.
- **Command:** Executes SQL queries or stored procedures against a database. It includes methods for executing queries, stored procedures, and retrieving data.
- **DataReader:** Provides a forward-only, read-only stream of data from a data source. It's typically used for retrieving large datasets efficiently in a read-only manner.
- **DataSet:** Represents an in-memory cache of data retrieved from a data source. It can hold multiple DataTable objects, each representing a table of data, along with relationships and constraints.
- **DataAdapter:** Acts as a bridge between a DataSet and a data source. It populates a DataSet with data from the data source and updates changes made in the DataSet back to the data source.

Code:

Step 1: add namespace using System.Data.Oledb; for access Database.

Step 2: Create connection object.

```
OleDbConnection con;
OleDbCommand cmd;
OleDbDataReader dr;
```

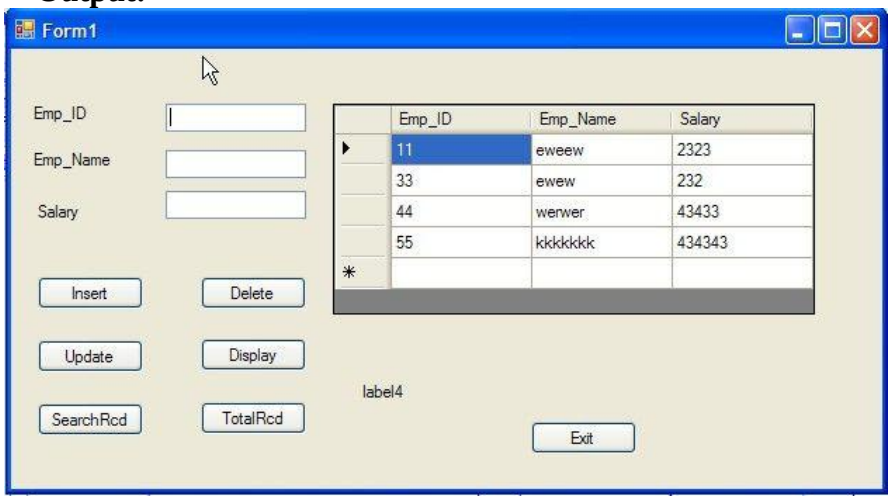
Step 3: Form1_Load

```
con = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\\Documents and Settings\\Admin\\Desktop\\Ado Connected Demo\\db1.mdb");
private void display()
con.Open();
```

```
cmd = new OleDbCommand("select * from Employee",con);
dr = cmd.ExecuteReader();
DataTable dt = new DataTable();
dt.Load(dr);
dataGridView1.DataSource = dt;
con.Close();
Display_Click
display();
Insert_Click
con.Open();
int aa = Convert.ToInt32(textBox1.Text);
string bb = textBox2.Text;
int cc = Convert.ToInt32(textBox3.Text);
cmd = new OleDbCommand("INSERT INTO Employee(Emp_ID, Emp_Name,Salary) VALUES ('" + aa + "','" + bb + "','" + cc + "')", con);
cmd.ExecuteNonQuery();
MessageBox.Show("one record inserted:");
con.Close();
display();
Delete_Click
con.Open();
int aa = Convert.ToInt32(textBox1.Text);
cmd = new OleDbCommand("DELETE FROM Employee where Emp_ID='" + aa + "'", con);
cmd.ExecuteNonQuery();
MessageBox.Show("one record Delete:");
con.Close();
display();
update_Click
con.Open();
int aa = Convert.ToInt32(textBox1.Text);
string bb = textBox2.Text;
int cc = Convert.ToInt32(textBox3.Text);
string abc = "UPDATE Employee SET Emp_ID ='" + aa + "', Emp_Name ='" + bb + "', Salary ='" + cc + "' WHERE Emp_ID = '" + aa + "'";
OleDbCommand cmd = new OleDbCommand(abc, con);
cmd.ExecuteNonQuery();
MessageBox.Show("one record updated:");
con.Close();
display();
```

```
Find_Click
con.Open();
int aa = Convert.ToInt32(textBox1.Text);
string abc = "SELECT Emp_ID,Emp_Name,Salary FROM Employee where Emp_ID='" + aa + "'";
cmd = new OleDbCommand(abc, con);
MessageBox.Show("one record search:");
dr = cmd.ExecuteReader();
DataTable dt = new DataTable();
dt.Load(dr);
dataGridView1.DataSource = dt;
con.Close();
Exit_Click

Application.Exit();
```

Output:



Conclusion: _____

_____

Submitted By : **Checked By :**

Sign :

Name : **Ms. Vaishnavi .U. Suryavanshi**

Roll No :

SSBT's Art's, Commerce and Science College, JalgaonDepartment of Computer Applications

Practical: 11

DOP: **DOC:**

Title: Create an ASP .NET application to demonstrate Data Bounds Controls.

Objective : Develop an ASP.NET application to demonstrate data binding using the GridView control, fetching data from a SQL Server database table, and displaying it in a tabular format on a web page.

Theory :
ADO.NET is a set of classes in the .NET Framework for accessing and manipulating data from diverse sources such as databases, XML, and more. Key components include:

- **Connection:** Establishes a connection to a data source.
- **Command:** Executes commands (SQL queries, stored procedures) against a data source.
- **DataReader:** Streams data from a data source in a read-only, forward-only manner.
- **DataSet:** Represents an in-memory cache of data with multiple DataTable objects, relationships, and constraints.
- **DataAdapter:** Bridges between a DataSet and a data source, populating and updating data.

ASP.NET provides data-bound controls like GridView, Repeater, etc., to display data from databases. These controls automatically handle data binding, allowing developers to display data in a tabular or list format on web pages with minimal code

ADO.NET's disconnected architecture allows data to be fetched from a database, manipulated in-memory, and changes propagated back to the database. This architecture enhances performance and scalability in web applications.

Code:
Employee.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="DataReapterDemo.aspx.cs"
Inherits="DataReapterDemo" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html
  xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
   <title></title>
     <style>
      tr {
           height:40px;
        }
    </style>
    </head>
    <body>
     <form id="form1" runat="server">
      <div>
       <center>
```

```
    <div style="  border: 2px solid red;text-align: left;border-radius: 2px;Padding-top: 3px;background-
color: Lime;width: 500px;border-radius: 8px;font-size: 20px;">
     <asp:Repeater ID="rp1" runat="server">
     <HeaderTemplate>
      <table  style="width:500px;padding-top:0px;Background-color:Gold" >
       <tr>
       <td style="font-size: 26px;
 text-align: center;
 height: 48px;">
     <asp:Label ID="lblhdr" runat="server" Text = "Student Profile"></asp:Label>
      </td>
      </tr>
 </table>
    </HeaderTemplate>
    <ItemTemplate>
     <table style="width:500px;">
      <tr>
       <td>
       <asp:Label ID="lblempid1" runat="server" Text="Employee ID:"></asp:Label>
        </td>
        <td>
         <asp:Label ID="lblempid2" runat="server" Text='<%# Eval("EmpId") %>'>
          </asp:Label>
          </td>
        <td rowspan="5">
        <asp:Image ID="img1" runat="server" Width="100px" ImageUrl= '
         <%#"~/images/" + Eval("EmpImage")   %>'/>
          </td>
          </tr>
          <tr>
          <td>
         <asp:Label ID="lblempname1" runat="server" Text="Employee Name"></asp:Label>
          </td>
          <td>
          <asp:Label ID="lblempname2" runat="server" Text='<%# Eval("EmpName") %>'>
          </asp:Label>
           </td>
          </tr>
           <tr>
           <td>
        <asp:Label  ID="lblempemailId1" runat="server" Text="Employee EmailId"></asp:Label>
           </td>
           <td>
         <asp:Label ID="lblempemailId2" runat="server" Text='<%# Eval("EmpEmailId") %>'>
         </asp:Label>
           </td>
           </tr>
            <tr>
            <td>
          <asp:Label ID="lblempmob1" runat="server" Text="Mobile Number"></asp:Label>
               </td>
```

```
                <td>
            <asp:Label ID="lblempmob2" runat="server" Text='<%# Eval("EmpMobileNum") %>'>
             </asp:Label>
                </td>
                </tr>
                <tr>
                <td>
             <asp:Label ID="lblempgen1" runat="server" Text="Gender"></asp:Label>
                </td>
                <td>
              <asp:Label ID="lblempgen2" runat="server" Text='<%# Eval("EmpGender") %>'>
               </asp:Label>
                </td>
                </tr>
                </table>
                </ItemTemplate>
                <FooterTemplate>
                 <table>
                  <tr>

                   </tr>
                   </table>
                     </FooterTemplate>
                   </asp:Repeater>
                </div>
              </center>
           </div>
       </form>
    </body>
</html>
```

- **Employee.aspx.cs (C# Code)**

```
using System;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

using System.Data.SqlClient;
using System.Data;
using System.Web.Configuration;

public partial class DataReapterDemo : System.Web.UI.Page
{
    SqlConnection con = new
SqlConnection(WebConfigurationManager.ConnectionStrings["myconnection"].ConnectionString);
    protected void Page_Load(object sender, EventArgs e)
        {
        if (!IsPostBack)
            {
            Bind();
     }    }
```

```
public void Bind()
   {
      SqlCommand cmd = new SqlCommand("select * from Employee where EmpId = 1200",con);
      SqlDataAdapter da = new SqlDataAdapter(cmd);
      DataSet ds = new DataSet();
      da.Fill(ds, "Employee");
      rp1.DataSource = ds.Tables[0];
      rp1.DataBind();

   }
}
```

Output:

Column Name	Data type
EmpId	Int
EmpName	Varchar(50)
EmpEmailId	Varchar(50)
EmpMobileNum	Bigint
EmpImage	Varchar(50)
EmpGender	Varchar(10)

Conclusion:_____

_____

Submitted By : **Checked By :**

Sign :

Name : **Ms.Vaishnavi .U. Suryavanshi**

Roll No :

Practical: 12

DOP: **DOC:**

Title: Create an ASP .NET application to demonstrate Navigation Controls.

Objective : To understand and implement website navigation controls in an ASP.NET web application using a SiteMap file. Participants will learn how to create and configure navigation controls, define a SiteMap hierarchy, and display site navigation menus.

Theory :

1. Website Navigation Controls

SiteMapPath Control: SiteMapPath is a navigation control that displays the path to the current page within a site hierarchy.

TreeView Control:

TreeView is a hierarchical navigation control that displays site structure in a tree-like format.

Menu Control:

Menu is a navigation control that provides a menu system for site navigation.

2. SiteMap

SiteMap File: A SiteMap file is an XML file that defines the hierarchical structure of a website, including the navigation links and their relationships

- **Code:**

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
<h1> Home Page </h1>
</div>
</form>
</body>
</html>
```

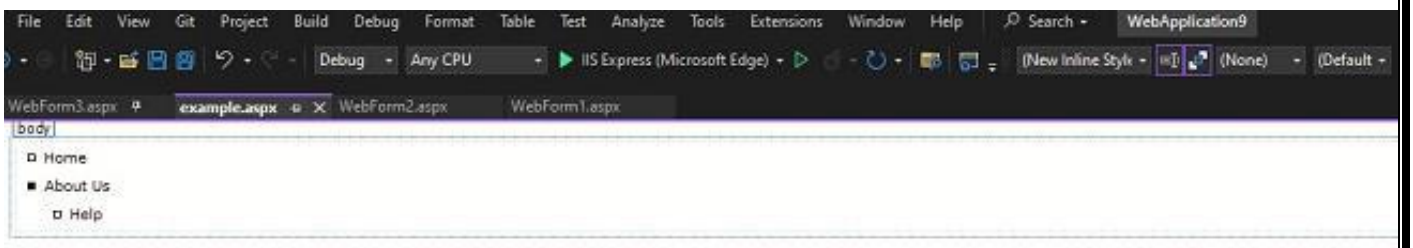
 Web form 2.aspx :

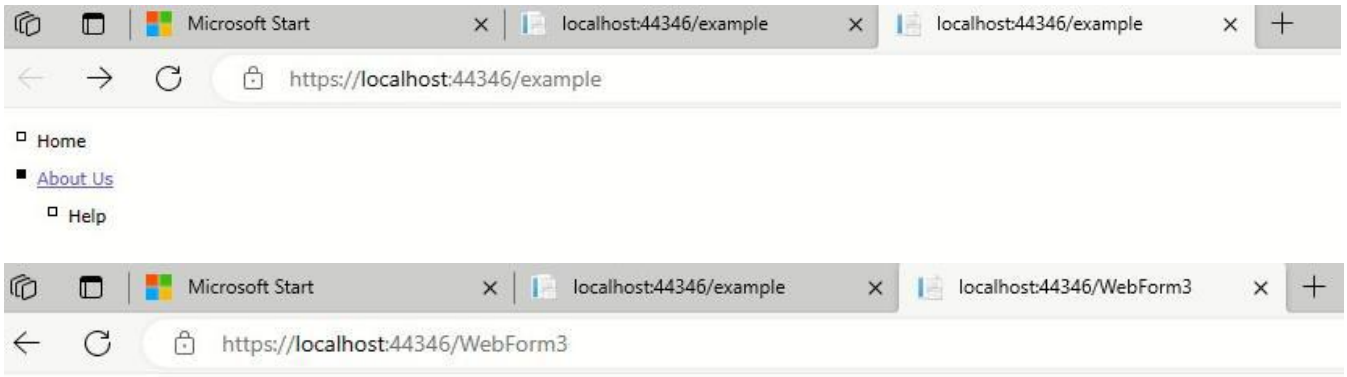
```
 <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm2.aspx.cs"
Inherits="WebApplication9.WebForm2" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
<h1>About us </h1>
</div>
</form>
</body>
</html>
```

 Web form 3 .aspx:

```
 <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm3.aspx.cs"
Inherits="WebApplication9.WebForm3" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
<h1>Help </h1>
</div>
</form>
</body>
</html>
```

Output:





Conclusion:_____

Submitted By : **Checked By :**

Sign :

Name : **Ms.Vaishnavi .U. Suryavanshi**

Roll No :