Writing Windows Form Application

Introduction to Win Forms:

Windows Forms (WinForms) is a graphical (GUI) class library included as a part of Microsoft .NET Framework, providing a platform to write rich client applications for desktop, laptop, and tablet PCs.

A Windows Forms application is an event-driven application supported by Microsoft's.NET Framework. Unlike a batch program, it spends most of its time simply waiting for the user to do something, such as fill in a text box or click a button.

All visual elements in the Windows Forms class library derive from the Control class. This provides a minimal functionality of a user interface element such as location, size, color, font, text, as well as common events like click and drag/drop.

Basic Controls:

The following table lists some of the commonly used controls:

S.No.	Widget	Description
1.	Forms	The container for all the controls that make up the user interface.
2.	TextBox	It represents a Windows text box control.
3.	Label	It represents a standard Windows label.
4.	Button	It represents a Windows button control.
5.	ListBox	It represents a Windows control to display a list of items.
6.	ComboBox	It represents a Windows combo box control.
7.	RadioButton	It enables the user to select a single option from a group of choices when paired with other RadioButton controls.
8.	CheckBox	It represents a Windows CheckBox.
9.	PictureBox	It represents a Windows picture box control for displaying an image.
10.	ProgressBar	It represents a Windows progress bar control.
11.	ScrollBar	It Implements the basic functionality of a scroll bar control.
12.	DateTimePicker	It represents a Windows control that allows the user to select a date and a time and to display the date and time with a specified format.

	TreeView	It displays a hierarchical collection of labeled items, each represented by a TreeNode.
13.	ListView	It represents a Windows list view control, which displays a collection of items that can be displayed using one of four different views.

For more information, refer to the practical exercises.

Web Applications using ASP.NET

A Visual Studio Web application is built around ASP.NET. ASP.NET is a platform — including design-time objects and controls and a run-time execution context — for developing and running applications on a Web server.

ASP.NET Web applications run on a Web server configured with Microsoft Internet Information Services (IIS). However, you do not need to work directly with IIS. You can program IIS facilities using ASP.NET classes, and Visual Studio handles file management tasks such as creating IIS applications when needed and providing ways for you to deploy your Web applications to IIS.

Elements of ASP.NET Web Applications

Creating ASP.NET Web applications involves working with many of the same elements you use in any desktop or client-server application. These include:

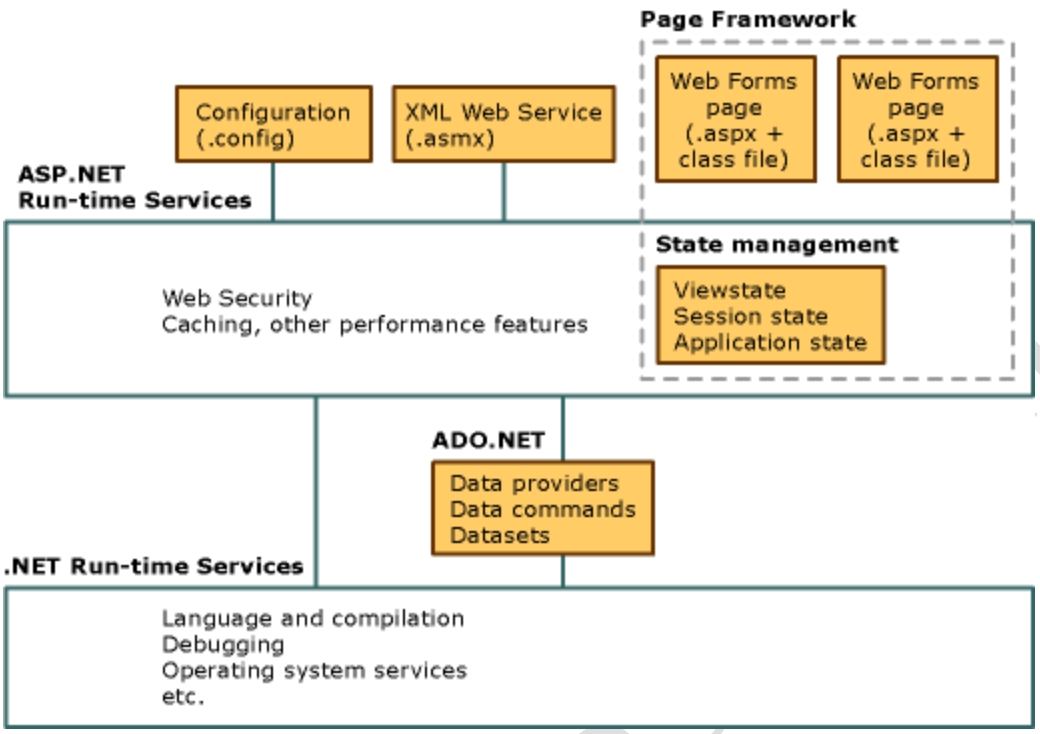
- **Project management features**: When creating an ASP.NET Web application, you need to keep track of the files you need, which ones need to be compiled, and which need to be deployed.

- **User interface:** Your application typically presents information to users; in an ASP.NET Web application, the user interface is presented in Web Forms pages, which send output to a browser. Optionally, you can create output tailored for mobile devices or other Web appliances.

- **Components:** Many applications include reusable elements containing code to perform specific tasks. In Web applications, you can create these components as XML Web services, which makes them callable across the Web from a Web application, another XML Web service, or a Windows Form, for example.

- **Data:** Most applications require some form of data access. In ASP.NET Web applications, you can use ADO.NET, the data services that are part of the .NET Framework.

- **Security, performance, and other infrastructure features:** As in any application, you must implement security to prevent unauthorized use, test and debug the application, tune its performance, and perform other tasks not directly related to the application's primary function.



Different Types of form controls in ASP.NET

Button Controls

ASP.NET provides three types of button control:

- **Button**: It displays text within a rectangular area.
- **Link Button**: It displays text that looks like a hyperlink.
- **Image Button**: It displays an image.

  <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="Click" / >

Text Boxes and Labels

Text box controls are typically used to accept input from the user. A text box control can accept one or more lines of text depending upon the settings of the TextMode attribute.

Label controls provide an easy way to display text which can be changed from one execution of a page to the next. If you want to display text that does not change, you use the literal text.

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

Check Boxes and Radio Buttons

A check box displays a single option that the user can either check or uncheck and radio buttons present a group of options from which the user can select just one option.

To create a group of radio buttons, you specify the same name for the **GroupName** attribute of each radio button in the group. If more than one group is required in a single form, then specify a different group name for each group.

If you want check box or radio button to be selected when the form is initially displayed, set its Checked attribute to true. If the Checked attribute is set to true for multiple radio buttons in a group, then only the last one is considered as true.

<asp:CheckBox ID= "chkoption" runat= "Server">
</asp:CheckBox>
<asp:RadioButton ID= "rdboption" runat= "Server">
</asp: RadioButton>

List box Control

These control let a user choose from one or more items from the list. List boxes and drop- down lists contain one or more list items. These lists can be loaded either by code or by the **ListItemCollection** editor.

<asp:ListBox ID="ListBox1" runat="server">
</asp:ListBox>

HyperLink Control

The HyperLink control is like the HTML <a> element.

<asp:HyperLink ID="HyperLink1" runat="server"> HyperLink </asp:HyperLink>

Image Control

The image control is used for displaying images on the web page, or some alternative text, if the image is not available.

<asp:Image ID="Image1" ImageUrl=”url” runat="server">

Drop down List Control

<asp:DropDownList ID="DropDownList1" runat="server"
</asp:DropDownList>

Launch another form on button click

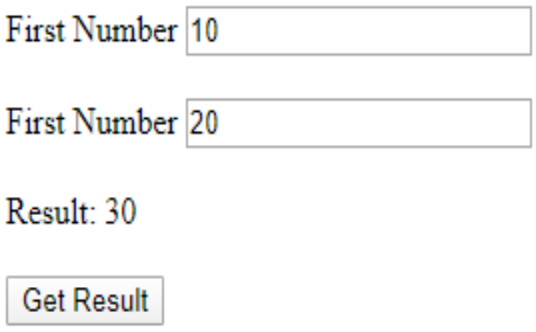
```
protected void btnSelect_Click(object sender, EventArgs e) {
        Response.Redirect("AnotherForm.aspx");
```

}

Example 1 – Creating a basic form in ASP.Net


```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="MyForm.aspx.cs" Inherits="WebApplication1.MyForm" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>This is my first application</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:Label ID="lblName" runat="server" Text="Name:"></asp:Label>
        <asp:TextBox ID="txtName" runat="server"></asp:TextBox><br/>
        <asp:Label ID="lblAddress" runat="server" Text="Address:"> </asp:Label>
        <asp:TextBox ID="txtAddress" runat="server"></asp:TextBox> <br/>
        <asp:Button ID="btnSubmit" runat="server" Text="Submit" />
    </form>
</body>
</html>
```

Example – 2 (Handling Events)



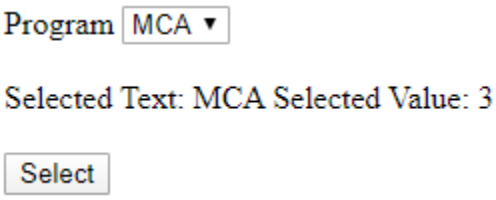
EventHandling.aspx

```
<form id="form1" runat="server">
        <div>
                <asp:Label ID="Label1" runat="server" Text="First Number"> </asp:Label>
                <asp:TextBox ID="txtFirst" runat="server"></asp:TextBox> <br/><br/>
                <asp:Label ID="Label2" runat="server" Text="First Number"> </asp:Label>
                <asp:TextBox ID="txtSecond" runat="server"></asp:TextBox> <br/><br/>
                <asp:Label ID="lblResult" runat="server" Text="Result:"> </asp:Label>
                <br/><br/>
                <asp:Button ID="btnSubmit" runat="server" Text="Get Result"
        onClick="btnSubmit_Click"/>
        </div>
</form>
```

EventHandling.cs

```
protected void btnSubmit_Click(object sender, EventArgs e) {
        int first = Convert.ToInt32(txtFirst.Text);
        int second = Convert.ToInt32(txtSecond.Text);
        int res = first + second; lblResult.Text = "Result: " + res;
}
```

Example 3 – Using Drop down list



Dropdown.aspx

```
<form id="form1" runat="server">
        <div>
                <asp:Label ID="Label1" runat="server" Text="Program"> </asp:Label>
                <asp:DropDownList ID="dropProgram" runat="server"> </asp:DropDownList>
                <br/><br/>
                <asp:Label ID="lblSelected" runat="server" Text="Selected:"> </asp:Label>
                <br/><br/>
```

```
        <asp:Button ID="btnSelect" runat="server" Text="Select"
    OnClick="btnSelect_Click" />
        </div>
    </form>
```

Dropdown.cs

```
private void LoadData() {
    List<ListItem> mylist=new List<ListItem>();
    mylist.Add(new ListItem("BCA","1"));
    mylist.Add(new ListItem("BBA","2"));
    mylist.Add(new ListItem("MCA","3"));
    mylist.Add(new ListItem("MBA","4"));
    dropProgram.Items.AddRange(mylist.ToArray());
}
protected void btnSelect_Click(object sender, EventArgs e) {
    string text = dropProgram.SelectedItem.ToString();
    string value = dropProgram.SelectedValue;
    lblSelected.Text = "Selected Text: " + text + " Selected Value: " + value;
}
```

Example – 4 Using Radio button

Gender ○ Male ◉ Female

Female Selected

Select

example.aspx

```
<form id="form1" runat="server">
    <div>
        <asp:Label ID="Label1" runat="server" Text="Gender"></asp:Label>
        <asp:RadioButton ID="radioMale" Text="Male" GroupName="gender"
    runat="server" />
        <asp:RadioButton ID="radioFemale" Text="Female" GroupName="gender"
    runat="server" /> <br/><br/>
        <asp:Label ID="lblSelected" runat="server" Text="Selected
```

```
            Radio:"> </asp:Label> <br/><br/>
            <asp:Button ID="btnSelect" runat="server" Text="Select"
        OnClick="btnSelect_Click" />
        </div>
</form>
```

example.cs

```
protected void btnSelect_Click(object sender, EventArgs e) {
        if (radioMale.Checked)
            lblSelected.Text = "Male Selected";
        else
            lblSelected.Text = "Female Selected";
}
```

Example – 5 Using Check box

example.aspx

```
<form id="form1" runat="server">
<div>
        <asp:Label ID="Label1" runat="server" Text="Gender"></asp:Label>
        <asp:CheckBox ID="chkMale" Text="Male" GroupName="gender" runat="server" />
        <asp:CheckBox ID="chkFemale" Text="Female" GroupName="gender" runat="server"
/> <br/><br/>
        <asp:Label ID="lblSelected" runat="server" Text="Selected Radio:"> </asp:Label>
<br/><br/>
        <asp:Button ID="btnSelect" runat="server" Text="Select" OnClick="btnSelect_Click" />
</div>
</form>
```

example.cs

```
protected void btnSelect_Click(object sender, EventArgs e) {
        if (chkMale.Checked)
                lblSelected.Text = "Male Selected";
        else
                lblSelected.Text = "Female Selected";
}
```

Validation Controls in ASP.NET

An important aspect of creating ASP.NET Web pages for user input is to be able to check that the information users enter is valid. ASP.NET provides a set of validation controls that provide an easy-to-use but powerful way to check for errors and, if necessary, display messages to the user.

There are six types of validation controls in ASP.NET

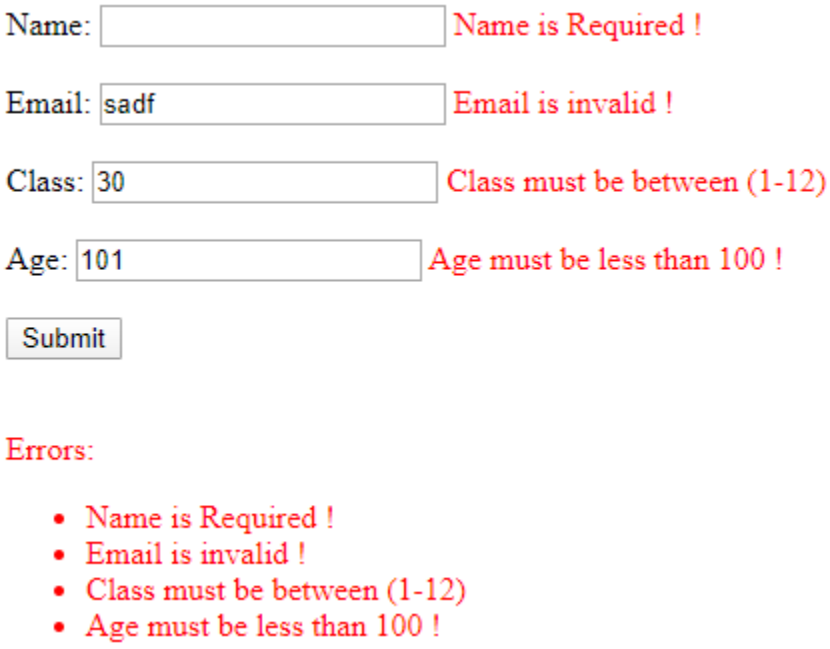
- RequiredFieldValidation Control
- CompareValidator Control
- RangeValidator Control
- RegularExpressionValidator Control
- CustomValidator Control
- ValidationSummary

The below table describes the controls and their work:

Validation Control	Description
RequiredFieldValidation	Makes an input control a required field
CompareValidator	Compares the value of one input control to the value of another input control or to a fixed value
RangeValidator	Checks that the user enters a value that falls between two values
RegularExpressionValidator	Ensures that the value of an input control matches a specified pattern
CustomValidator	Allows you to write a method to handle the validation of the value entered

ValidationSummary	Displays a report of all validation errors occurred in a Web page

Example of Validation Controls




```
<form id="form1" runat="server">
        <div>
                <asp:Label ID="Label1" runat="server" Text="Name:"></asp:Label>
                <asp:TextBox ID="txtName" runat="server"></asp:TextBox>
                <asp:RequiredFieldValidator ID="validator1" runat="server"
                ForeColor="Red" ErrorMessage="Name is Required !"
                ControlToValidate="txtName"> </asp:RequiredFieldValidator>
                <br/><br/>

                <asp:Label ID="Label2" runat="server" Text="Email:"></asp:Label>
                <asp:TextBox ID="txtEmail" runat="server"></asp:TextBox>
                <asp:RegularExpressionValidator ID="validator2" runat="server"
                ForeColor="Red" ControlToValidate="txtEmail"
                        ErrorMessage="Email is invalid !"
                        ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-
                .]\w+)*">
                </asp:RegularExpressionValidator>
```

```
            <br/><br/>

            <asp:Label ID="Label3" runat="server" Text="Class:"></asp:Label>
            <asp:TextBox ID="txtClass" runat="server"></asp:TextBox>
            <asp:RangeValidator ID="validator3"
                  runat="server" ControlToValidate="txtClass"
                  ForeColor="Red" ErrorMessage="Class must be between (1-12)"
                  MaximumValue="12" MinimumValue="1" Type="Integer">
            </asp:RangeValidator> <br/><br/>

            <asp:Label ID="Label4" runat="server" Text="Age:"></asp:Label>
            <asp:TextBox ID="txtAge" runat="server"></asp:TextBox>
            <asp:CompareValidator ID="validator4" runat="server" ValueToCompare="100"
            ControlToValidate="txtAge" ErrorMessage="Age must be less than 100 !"
            ForeColor="Red" Operator="LessThan" Type="Integer">
            </asp:CompareValidator>
            <br/><br/>

            <asp:Button ID="btnSubmit" runat="server" Text="Submit" />
      </div> <br/><br/>

      <asp:ValidationSummary ID="validator5" runat="server"
            ForeColor="Red" DisplayMode ="BulletList"
            ShowSummary ="true" HeaderText="Errors:" />
</form>
```