UNIT III: ASP.NET WITH C#

UNIVERSITY QUESTIONS WITH ANSWER

1. What is user-defined Exception? Explain with example.[Nov18]

.NET provides a hierarchy of exception classes ultimately derived from the base class Exception. However, if none of the predefined exceptions meets your needs, you can create your own exception classes by deriving from the Exception class.When creating your own exceptions, end the class name of the user-defined exception with the word "Exception", and implement the three common constructors, as shown in the following example. The example defines a new exception class named InvalidAgeExecption The class is derived from Exception class.

Syntax:-

```
public class ExecptionName:Exception{

}
```

Program:-

```
using System;
public class InvalidAgeException : Exception
{
public InvalidAgeException(String message)
base(message)
{

}
}
public class TestUserDefinedException
{
static void validate(int age)
{
if (age < 18)
{
throw new InvalidAgeException("Sorry, Age must be greater than 18");
}
}
public static void Main()
{
    Try
{
validate(12);
}
catch (InvalidAgeException e)
{
Console.WriteLine(e);
}
Console.WriteLine("Rest of the code");
}
}
```

Output:-

```
12
Sorry, Age must be greater than 18.
```

2. What is debugging. Explain the process of debugging in detail.[Nov18]

Debugging refers to the process of trying to track down errors in your programmes. It can also refer to handling potential errors that may occur. There are three types of errors that we'll take a look at:

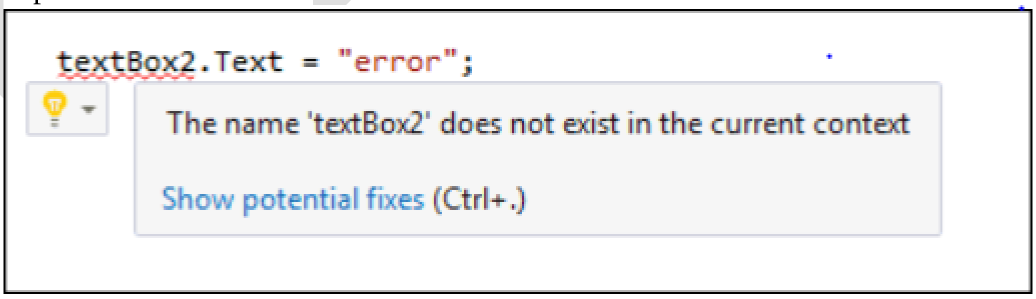
- Design-time errors
- Run-Time errors
- Logical errors

Design-time errors

Design-Time errors are ones that you make before the programme even runs. Design-Time errors are easy enough to spot because the C# software will underline them with a wavy coloured line. You'll see three different coloured lines: blue, red and green. The blue wavy lines are known as **Edit and Continue** issues, meaning that you can make change to your code without having to stop the programme altogether. Red wavy lines are **Syntax** errors, such as a missing semicolon at the end of a line, or a missing curly bracket in an IF Statement. Green wavy lines are **Compiler Warnings**.

A] Red Wavy Lines

This is an Edit and Continue error. It's been flagged because the form doesn't have a control called textBox2 - it's called textBox1. We can simply delete the 2 and replace it with a 1. The programme can then run successfully. Holding your mouse over the wavy underline gives an explanation of the error.



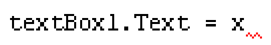
```
textBox2.Text = "error";
```

The name 'textBox2' does not exist in the current context

Show potential fixes (Ctrl+.)

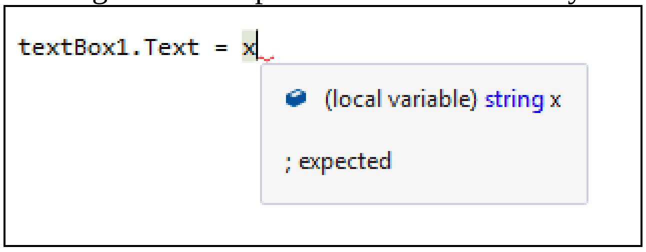
B.] Red Waves

These are Syntax errors. In the code below, we've missed out the semicolon at the end of the line:



```
textBox1.Text = x
```

Holding the mouse pointer over the red wavy line gives the following message:

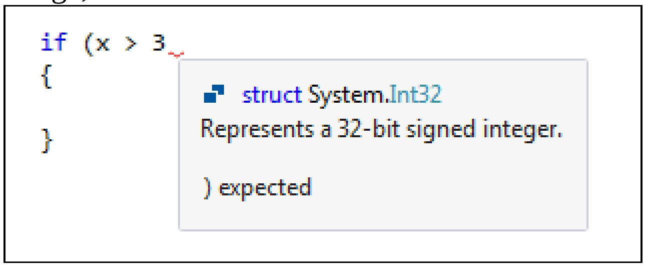


```
textBox1.Text = x
```

(local variable) string x

; expected

It's telling us that a semicolon (;) is expected where the red wavy underline is.In the next image, we've missed out a round bracket for the IF Statement:



```
if (x > 3
{

}
```

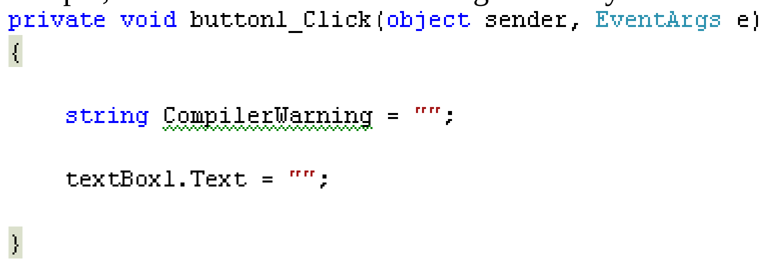
struct System.Int32
Represents a 32-bit signed integer.

) expected

Adding the round bracket will make the red wavy underline go away.

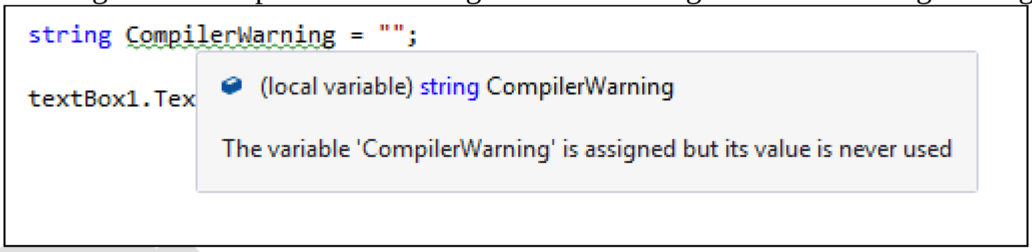
C.] Green Wavy Lines

These are Compiler Warnings, the C# way of alerting you to potential problems. As an example, here's some code that has a green wavy underline:



```
private void button1_Click(object sender, EventArgs e)
{

    string CompilerWarning = "";

    textBox1.Text = "";

}
```

Holding the mouse pointer over the green underlines gives the following message:



```
string CompilerWarning = "";

textBox1.Tex
```

(local variable) string CompilerWarning

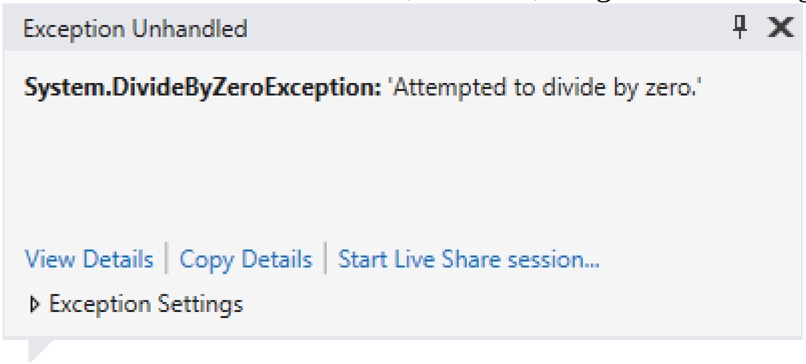
The variable 'CompilerWarning' is assigned but its value is never used

Run Time Error

Run-Time errors are ones that crash your programme. The programme itself generally starts up OK. It's when you try to do something that the error surfaces. A common Run-Time error is trying to divide by zero. In the code below, we're trying to do just that:

```
private void button1_Click(object sender, EventArgs e)
{

    int Num1 = 10;
    int Num2 = 0;
    int answer;

    answer = Num1 / Num2;

}
```

The program itself reports no problems when it is started up, and there are no coloured wavy lines. When we click the button, however, we get the following error message:


Exception Unhandled

System.DivideByZeroException: 'Attempted to divide by zero.'

Had we left this in a real programme, it would just crash altogether ("bug out"). But if you see any error message like this one, it's usually a Run-Time error.Look out for these type of error messages. It does take a bit of experience to work out what they mean; but some, like the one above, are quite straightforward.

Logic Error

Logic errors are ones where you don't get the result you were expecting. You won't see any coloured wavy lines, and the programme generally won't "bug out" on you. In other words, you've made an error in your programming logic. As an example, take a look at the following code, which is attempting to add up the numbers one to ten:

```
private void button1_Click(object sender, EventArgs e)
{

    int startLoop = 11;
    int endLoop = 1;
    int answer = 0;

    for (int i = startLoop; i < endLoop; i++)
    {
        answer = answer + i;
    }

    MessageBox.Show("answer =" + answer.ToString());
}
```

When the code is run, however, it gives an answer of zero. The programme runs OK, and didn't produce any error message or wavy lines. It's just not the correct answer!The problem is that we've made an error in our logic. The **startLoop** variable should be 1 and the **endLoop** variable 11. We've got it the other way round, in the code. So the loop never executes.Logic errors can be v.ery difficult to track down. To help you find where the problem is, C# has some very useful tools you can use. To demonstrate these tools, here's a new programming problem. We're trying to write a programme that counts how many times the letter "g" appears in the word "debugging".

```
private void button1_Click(object sender, EventArgs e)
{
    int LetterCount = 0;
    string strText = "Debugging";
    string letter;

    for (int i = 0; i < strText.Length; i++)
    {
        letter = strText.Substring(1, 1);

        if (letter == "g")
        {
            LetterCount++;
        }
    }

    textBox1.Text = "g appears " + LetterCount + " times";
}
```

The answer should, of course, be 3. Our programme insists, however, that the answer is zero. It's telling us that there aren't and g's in Debugging. So we have made a logic error, but where? C# .NET has some tools to help you track down errors like this. The first one we'll look at is called the BreakPoint.

3.Write a short note on cookies in ASP.NET? [Nov 18]

ASP.NET Cookie is a small bit of text that is used to store user-specific information. This information can be read by the web application whenever user visits the site.When a user requests for a web page, web server sends not just a page, but also a cookie containing the date and time. This cookie stores in a folder on the user's hard disk.When the user requests for the web page again, browser looks on the hard drive for the cookie associated with the web page. Browser stores separate cookie for each different sites user visited.

Types of cookies

- **Temporary Cookie or Session Cookie**
  - Stored as temporary files in client's RAM
  - Transferred with request and response
- **Persistent Cookie**
  - Stored as a file on hard disk

Advantages of Cookie:

- It's clear text so user can able to read it.
- We can store user preference information on the client machine.
- Cookies do not require any server resources since they are stored on the client.
- It's easy way to maintain.
- Cookies are easy to implement
- Fast accessing.

Disadvantages of Cookie:

- If user clear cookie information we can't get it back.
- No security.
- Each request will have cookie information with page.

- Cookies exist as plain text on the client machine and they may pose a possible security risk as anyone can open and tamper with cookies.

Example

```
protected void Button1_Click(object sender, EventArgs e)
   {
            String name = TextBox1.Text;
      HttpCookie cookiee;
      if (name != null && name.Length > 0)
      {
         cookiee = new HttpCookie("UserName", TextBox1.Text);

         cookiee.Expires = DateTime.Now.AddYears(1);
         Response.Cookies.Add(Username);

         Response.AppendCookie(cookiee);
      }

         Response.Redirect("Default2.aspx");
   }
public partial class Default2 : System.Web.UI.Page
{
   protected void Page_Load(object sender, EventArgs e)
   {
      if (Request.Cookies != null && Request.Cookies.Count > 0)
      {
         Label1.Text = "Welcome " + Request.Cookies["UserName"].Value + " to oursite";
      }
      else
      {
         Label1.Text = "Welcome guest to oursite";
      }
   }
}
```

4.What is ViewState in ASP.NET? State its advantages and disadvantages.

View State:

- View state is the state of the page and all its control. It is automatically maintained across posts by asp.net framework.
- A technique to persist data at client side through server round trips on a single page.
- Primary use is to retain the state of controls during Postback.
- "View State is used to maintain the state of controls during page postback and if we save any control values or anything in view state we can access those values throughout the page whenever it required."
- *View state* is an ASP.NET feature that provides for retaining the values of page and control properties that change from one execution of a page to another.

- Before ASP.NET sends a page back to the client, it determines what changes the program has made to the properties of the page and its controls. These changes are encoded into a string that's assigned to the value of a hidden input field named _VIEWSTATE.

Advantages of view state:

These are the main advantage of using View State:

1. Easy to implement
2. No server resources are required
3. Enhance security features, like it can be encoded and compressed.

Disadvantages of view state:

- **Security Risk:** The Information of View State can be seen in the page output source directly. You can manually encrypt and decrypt the contents of a Hidden Field, but It requires extra coding.
- **Performance:** Performance is not good if we use a large amount of data because View State is stored in the page itself and storing a large value can cause the page to be slow.
- **Device limitation :** Mobile Devices might not have the memory capacity to store a large amount of View State data.
- It can store values for the same page only.

When we should use view state?

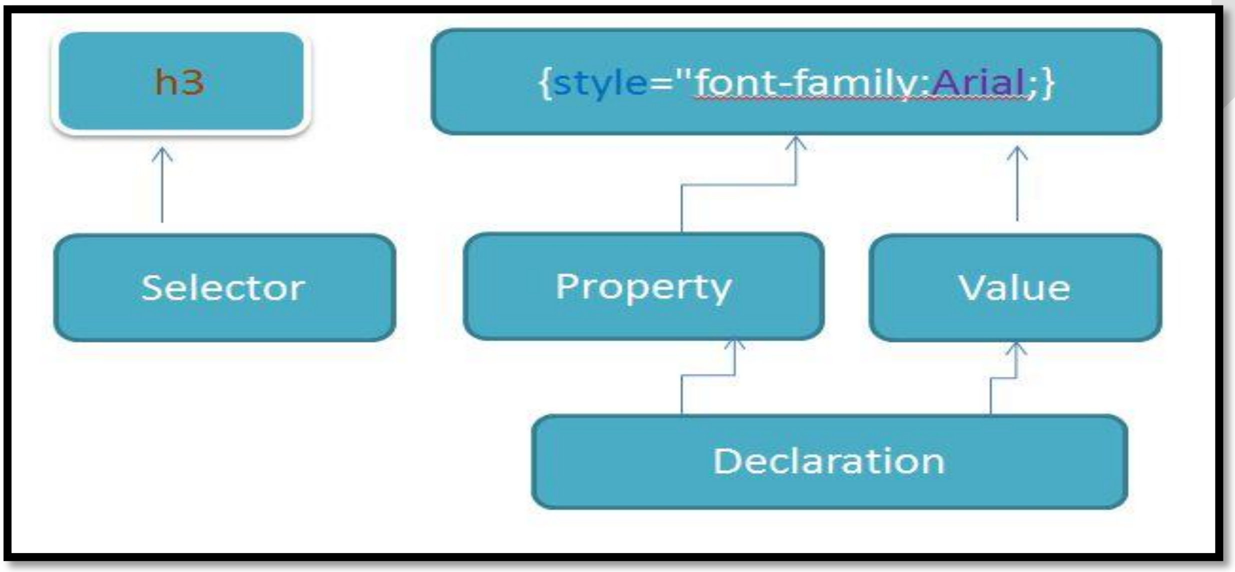
- Size of data should be small, because data are bind with page controls, so far large amount of data it can be cause of performance overhead.
- Try to avoid storing secure data in view state.

Example

```
protected void Button1_Click(object sender, EventArgs e)
  {
     //Value of Textbox1 and TectBox2 is assigin on the ViewState

     ViewState["name"] = TextBox1.Text;

     ViewState["password"] = TextBox2.Text;

     //after clicking on Button TextBox value Will be Cleared
     TextBox1.Text = TextBox2.Text = string.Empty;
    //  TextBox1.Text = "";
  }
  protected void Button2_Click(object sender, EventArgs e)
  {
     //If ViewState Value is not Null then Value of View State is Assign to TextBox

     if (ViewState["name"] != null)
     {
        TextBox1.Text = ViewState["name"].ToString();
     }

     if (ViewState["password"] != null)
     {
        TextBox2.Text = ViewState["password"].ToString();
        }
  }
```

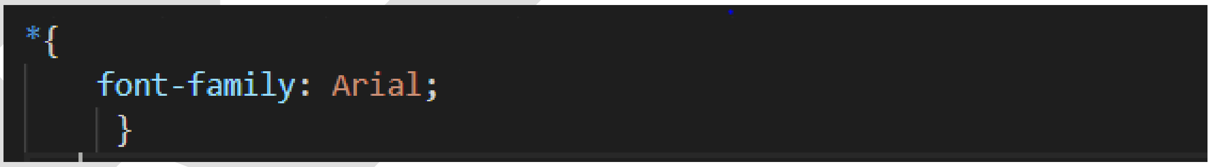
5. Explain the four most important selectors present in CSS. [Nov 18]

CSS stands for Cascading Style Sheets. It is generally used to display HTML in various views. It is used to design the view for HTML. CSS is a combination of a selector and a declaration.A selector is a HTML tag to for a style and a declaration is a combination of the property and a value.The Declaration's Property is predefined and the value is dependent on our requirements. If we have a number of properties then we can separate them by a colon if we want to design the font color, back color and font size. For this we have a number of CSS properties. The way to specify them in CSS is to separate them by a colon.



Universal Selector:-

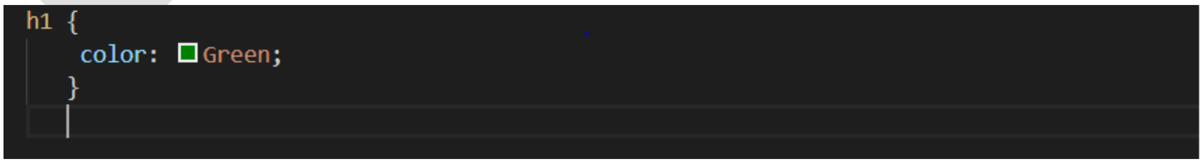
The Universal selector, indicated by an asterisk (*), applies to all elements in your page. The Universal selector can be used to set global settings like a font family. The following rule set changes the font for all elements in our page to Arial:



```
*{
    font-family: Arial;
  }
```

Type Selector :-

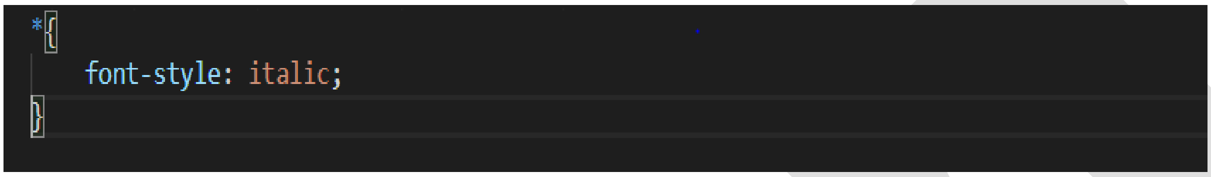
The Type selector enables us to point to an HTML element of a specific type. With a Type selector all HTML elements of that type will be styled accordingly.



```
h1 {
    color: Green;
  }
```

The ID Selector:

The ID selector is always prefixed by a hash symbol (#) and enables you to refer to a single element in the page. Within an HTML or ASPX page, you can give an element a unique ID using the id attribute. With the ID selector, you can change the behavior for that single element, like this:
#IntroText

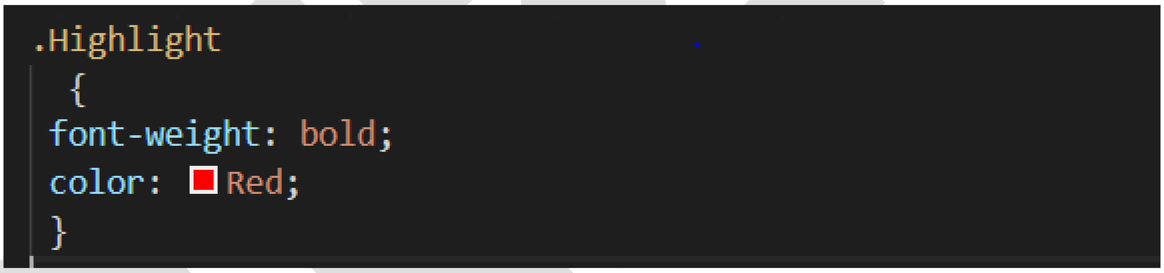


```
*{
    font-style: italic;
}
```

Because we can reuse this ID across multiple pages in our site (it only must be unique within a single page), you can use this rule to quickly change the appearance of an element that you use once per page, but more than once in our site, for example with the following HTML code: <p id=”IntroText”>I am italic because I have the right ID. </p>

Class Selector :-

The Class selector enables us to style multiple HTML elements through the class attribute. This is handy when we want to give the same type of formatting to several unrelated HTML elements. The following rule changes the text to red and bold for all HTML elements that have their class attributes set to highlight:



```
.Highlight
  {
 font-weight: bold;
 color: 🟥Red;
 }
```

The following code snippet uses the Highlight class to make the contents of a <span> element and a link (<a>) appear with a bold typeface: This is normal text but <span class=”Highlight”>this is Red and Bold.</span> This is also normal text but <a href=”CssDemo.aspx” class=”Highlight”>this link is Red and Bold as well.</a>

6. Create a web application to demonstrate use of Master Page with applying Styles and Themes for page beautification. Write necessary steps with code for the same. [Nov 18]

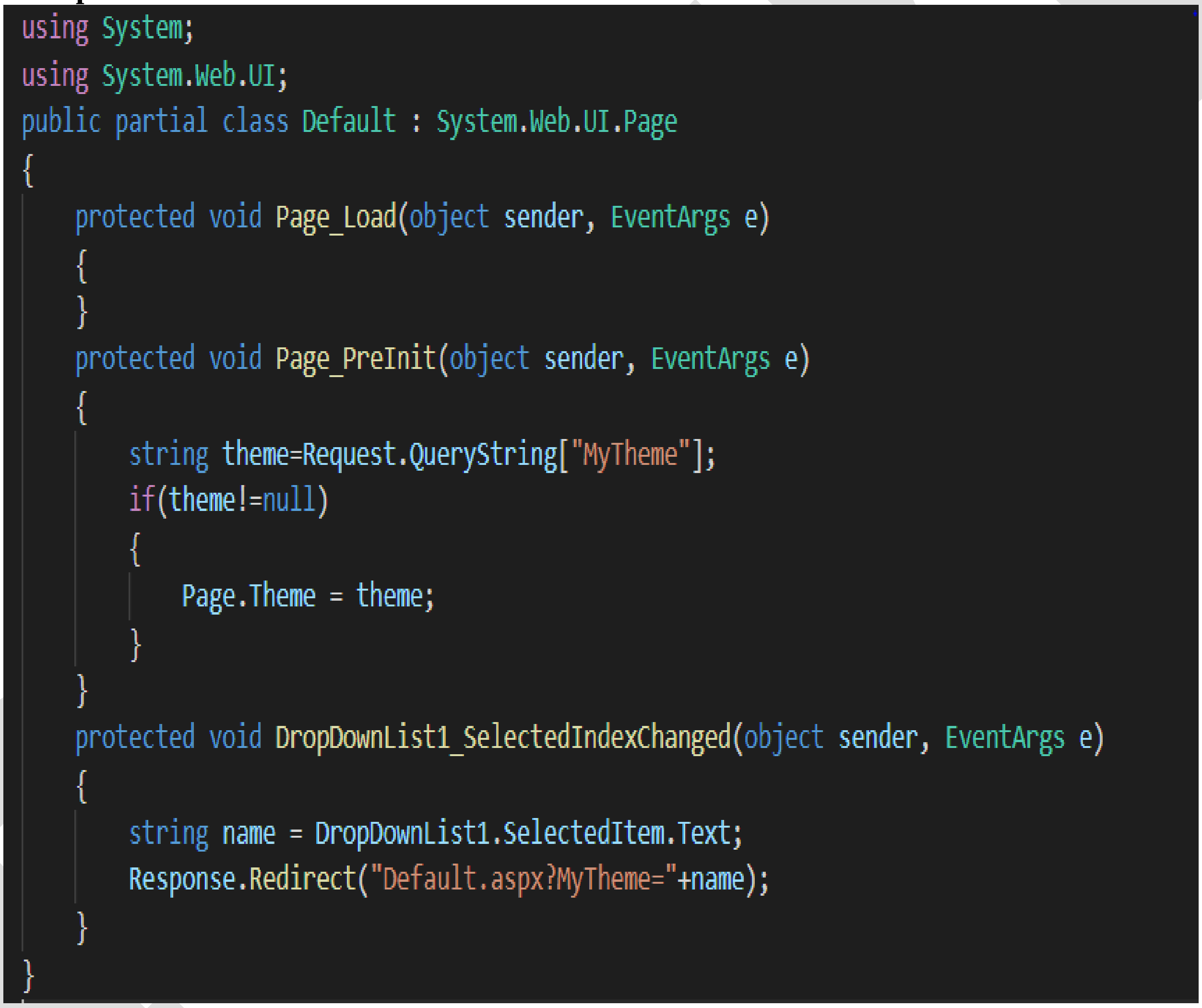
Introduction: -

Master Pages in ASP.NET allow you to control the layout, design, and common controls of all pages in a website/web application. With a single master page, you can define the look and feel of your application which includes multiple content place holders. Along with Master Pages, you can work with themes to provide the user with a great User Interface. In this article, we will see an overview of how to:

Steps: -

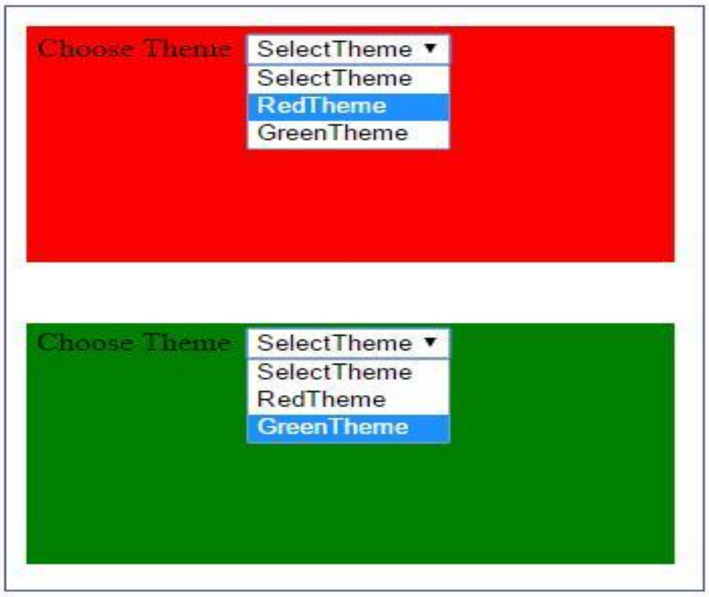
1.Create a Master Page with single and multiple content place holders
2.Add Content Pages
3.Nested Master Pages
4.Programming the Master Page
5.Creating themes for the application.
6.Finally we will see how we can work with multiple themes

Example: -



```
using System;
using System.Web.UI;
public partial class Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    protected void Page_PreInit(object sender, EventArgs e)
    {
        string theme=Request.QueryString["MyTheme"];
        if(theme!=null)
        {
            Page.Theme = theme;
        }
    }
    protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
    {
        string name = DropDownList1.SelectedItem.Text;
        Response.Redirect("Default.aspx?MyTheme="+name);
    }
}
```

Output:-

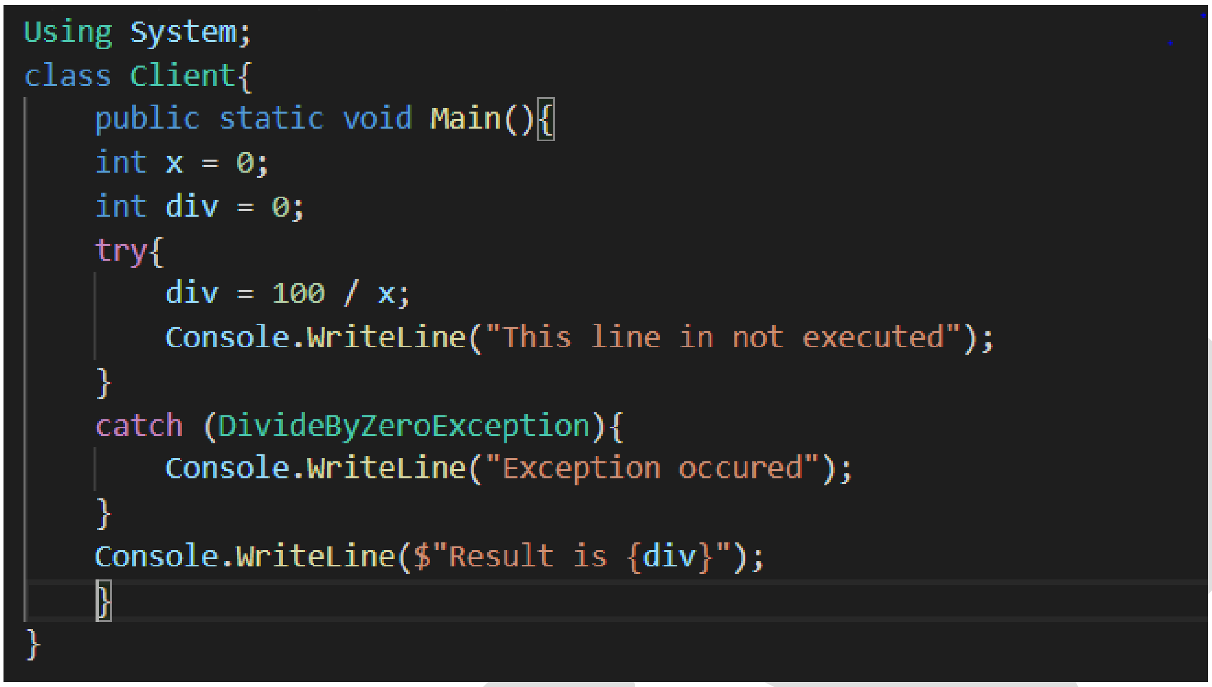


7. Explain exception handling mechanism in C# with its key features?[ATKT 2019]

An exception is a problem that arises during the exception of a program. A C# exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero. Exceptions provide a way to transfer control from one part of a program to another. C# exception handling is built upon four keywords: try, catch, finally and throw. –

- **try -** A try block identifies a block of code for which particular exceptions is activated. It is followed by one or more catch blocks.
- **catch -** A program catches an exception with an exception handler at the place in a program where you want to handle the problem. The catch keyword indicates the catching of an exception.
- **finally -** The finally block is used to execute a given set of statements, whether an exception is thrown or not thrown. For example, if you open a file, it must be closed whether an exception is raised or not.
- **throws -** A program throws an exception when a problem shows up. This is done using a throw keyword.

Program:



```
Using System;
class Client{
    public static void Main(){
    int x = 0;
    int div = 0;
    try{
        div = 100 / x;
        Console.WriteLine("This line in not executed");
    }
    catch (DivideByZeroException){
        Console.WriteLine("Exception occured");
    }
    Console.WriteLine($"Result is {div}");
    }
}
```

OUTPUT:

```
Exception occurred
Result is 0
```

Several key features are as follows:-

- Exceptions are object-based: Exception object is created, you can even create and throw your own exception objects.
- Exceptions are caught based on their type: This allows you to streamline error-handling code without needing to shift through obscure error codes.
- Exceptions are a generic part of the .NET Framework: This means they're completely cross language compatible.
- Exception handlers are multi-layered: You can easily layer exception handlers on top of other exception handlers, some of which may check only for a specialized set of errors.

8. What are the state management techniques in ASP.NET?[ATKT19]

State Management can be defined as the technique or the way by which we can maintain or store the state of the page or application until the User's Session ends. ASP.NET provides us with 2 ways to manage the state of an application. It is basically divided into the 2 categories Client Side State Management and Server Side State Management.

Client Side State Management:

It is a way in which the information which is being added by the user or the information about the interaction happened between the user and the server is stored on the client's machine or in the page itself. This management technique basically makes use of the following:

a. View State
b. Hidden Fields
c. Query String

d. Cookies

#View State: View State can be used to maintain the State at a page level.
Advantages:
-It is very simple to use.
Disadvantages:
-Information is not encrypted, so it can be easy for a Hacker to get its value.
-Cannot be used to store sensitive data (Passwords, Credit Card Pins).

#Hidden Fields: ASP.NET provides a server control called "Hidden Field" which can be used to store a value at a page level, which is similar to a View State.
Advantages:
-Hidden Fields store the value in the page itself, hence do not use server resources.
Disadvantages:
-Will make a page heavy, if too many Hidden Fields are used to store data.
-Cannot store sensitive data, as the value that is stored is neither hashed, nor encrypted.

#Query String: A Query String is a string which is appended to the end of the Page URL. It is very simple to use and can be used to send data across pages. It stores information in a key-value pair. A "?" signature is used to append the key and value to the page URL.

#Cookies: ASP.Net provides another way of state management, which is by using Cookies. Cookies are one of the best ways of storing information. It is nothing but a text file which is stored on the client's machine. There are 2 ways of assigning / storing values in cookies:
-Using the Request Object
-Using the HTTPCookies Object
Advantages:
-Very easy to use.
-Stored on the client's machine, hence no server resources are utilized.
Disadvantages:
-A user can disable cookies using browser settings.

Server Side State Management:
It is another way which ASP.NET provides to store the user's specific information or the state of the application on the server machine. It completely makes use of server resources (the server's memory) to store information. This management technique basically makes use of the following:
a. Application State
b. Session State

Application State: It stores information as a Dictionary Collection in key - value pairs. This value is accessible across the pages of the application / website. There are 3 events of the Application which are as follows:
Application_Start
Application_Error
Application_End

Session State: The Session basically stores the values as a dictionary collection in key/value pairs. It completely utilizes server resources to store the data. It is a secure way of storing data, since the data will never be passed to the client. There are various ways in which we can store a session and they are as follows:

a. OFF
b. InProc
c. State Server
d. SQL Server

OFF:If we do not want to use sessions in our application, then we can set the mode as "OFF".

InProc: This is the default mode which is used in ASP.NET to store session variables. InProc mode basically stores the session value in the process in which the ASP.NET application is running. Since it is stored in the same process, it provides high performance as compared to other modes.

State Server: If we use this mode, then a separate Windows Service which runs in a different process stores the Session. It basically isolates the Session from the ASP.NET running process. It provides less performance as compared to the Inproc mode.

SQL Server: This mode stores the Session in SQL Server instead of the server's memory. If our application is stored on multiple server machines, then it becomes difficult to use the "Inproc" mode since the request can go to any server for processing. So if we want to use sessions in such cases, then we can store the Session in SQL Server so that it becomes centralized and can be accessed by any server during the request process. It is the most secure way of storing Sessions but gives the lowest performance for an application.

9. Elaborate cookies with suitable code snippet.[ATKT19]

→ It is a method used to store information. These are small files which are created in the web browser's memory. It is transparent and can be implemented without the details.

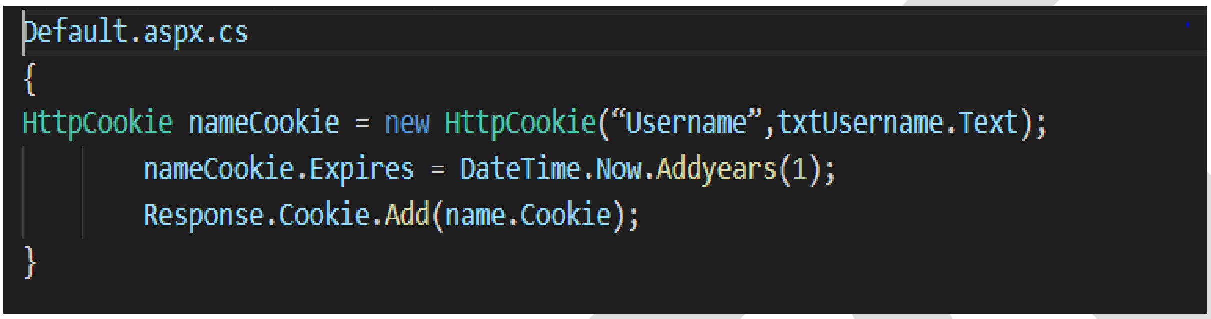
The drawbacks that affect query strings namely, they're limited to simple string information, and they're easily accessible and readable if the user finds and opens the corresponding file.

It is easy to use. The Request and Response objects (which are provided through Page properties) provide a Cookies collection.

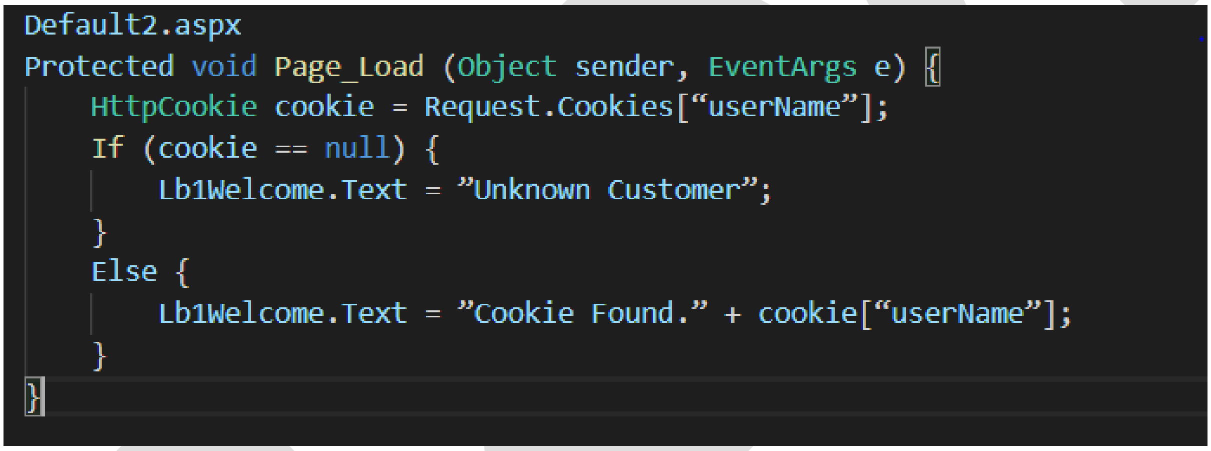
- A cookie is a name-value pair that is stored on the client's computer.
- A web application sends a cookie to a browser via an HTTP response.
- Each time the browser sends an HTTP request to the server, it attaches any cookies that are associated with that server.
- By default asp .net uses a cookie to store the session id for a session, but cookies can also be created and sent to a user's browser.
- A session cookie is kept in the browser's memory and exists only for the duration of the browser's session.
- A persistent cookie is kept in the user's disk and is retained until the cookie's expiry date.

- To create a cookie, you specify its name and value.
- To create a persistent cookie, the Expires property also has to be set to the time one wants the cookie to expire.

Example:



```
Default.aspx.cs
{
HttpCookie nameCookie = new HttpCookie("Username",txtUsername.Text);
        nameCookie.Expires = DateTime.Now.Addyears(1);
        Response.Cookie.Add(name.Cookie);
}
```



```
Default2.aspx
Protected void Page_Load (Object sender, EventArgs e) {
    HttpCookie cookie = Request.Cookies["userName"];
    If (cookie == null) {
        Lb1Welcome.Text = "Unknown Customer";
    }
    Else {
        Lb1Welcome.Text = "Cookie Found." + cookie["userName"];
    }
}
```

10. What is cross page posting? Explain with an example.[ATKT 19]

A cross page post back is a technique that extends the post back mechanism so information can be transferred from one page can send the user to another page. We can use the PostBackURLattribute to specify the page you want to post to.

Example:

Step1: Create a new ASP.NET website called CrossPagePosting. By default, the website is created with a single webpage, Default.aspx. Right click the project in the Solution Explorer > Add New Item >Web Form. Keep the original name Default2.aspx and click 'Add'. The website will now contain two pages, Default.aspx and Default2.aspx.

Step 2: On the source page, Default.aspx, drop a button on the form. Set the text of the button as 'TargetButton'. Set the 'PostBackUrl' property of a Button to the URL of the target page, Default2.aspx.

```
<asp:Button ID="Button1" runat="server" PostBackUrl="~/Default2.aspx" Text="TargetButton" /></div>
```

Step 3: In the target page Default2.aspx, drop a label on the page from the toolbox.

Step 4: In the Page_Load() of Default2.aspx, you can then access the 'PreviousPage' property to check if the page is being accessed as Cross Page postback.

```
protected void Page_Load(object sender, EventArgs e)
{
      if (Page.PreviousPage != null)
      {
      }
}
```

Step 5: To retrieve values from the source page, you must access controls using the 'FindControl()' method of the 'PreviousPage'. We will be accessing the Text property of the Button control placed in Default.aspx.

```
 protected void Page_Load(object sender, EventArgs e)
{
       if (Page.PreviousPage != null)
       {
          Button btn = (Button)(Page.PreviousPage.FindControl("button1"));
          Label1.Text = btn.Text;
       }

}
```

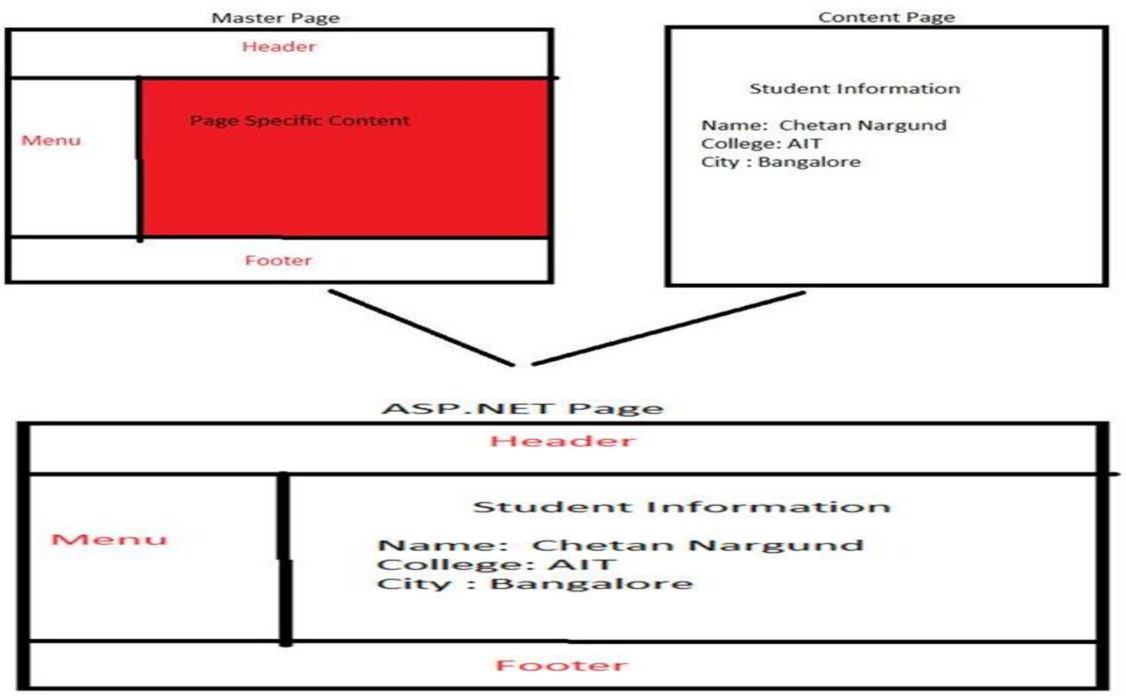
Step 6: In the Solution Explorer, right click Default.aspx > 'Set as Start Page'. Run the application and click on the button. As you can observe, the page is posted to Default2.aspx and the value containing the name of the button control gets displayed in the label.

11.Explain Master Page with its uses and Working.?[ATKT 19]

Master page allows you to create a consistent look and behaviour for all the pages in your web applications. Master Page Design is common for all the pages. Master page actually consists of two pieces, the master page itself and one or more content pages. A master page is an ASP.NET file with the extension. master. ContentPlaceholder control, which defines a region on the master page, is where the content pages can plug in page specific content.

Important points about master pages

- The @ Master directive defines it as a master page.
- ContentPlaceHolder control is only available for master pages.
- ASP.NET page that uses master pages is called content pages.

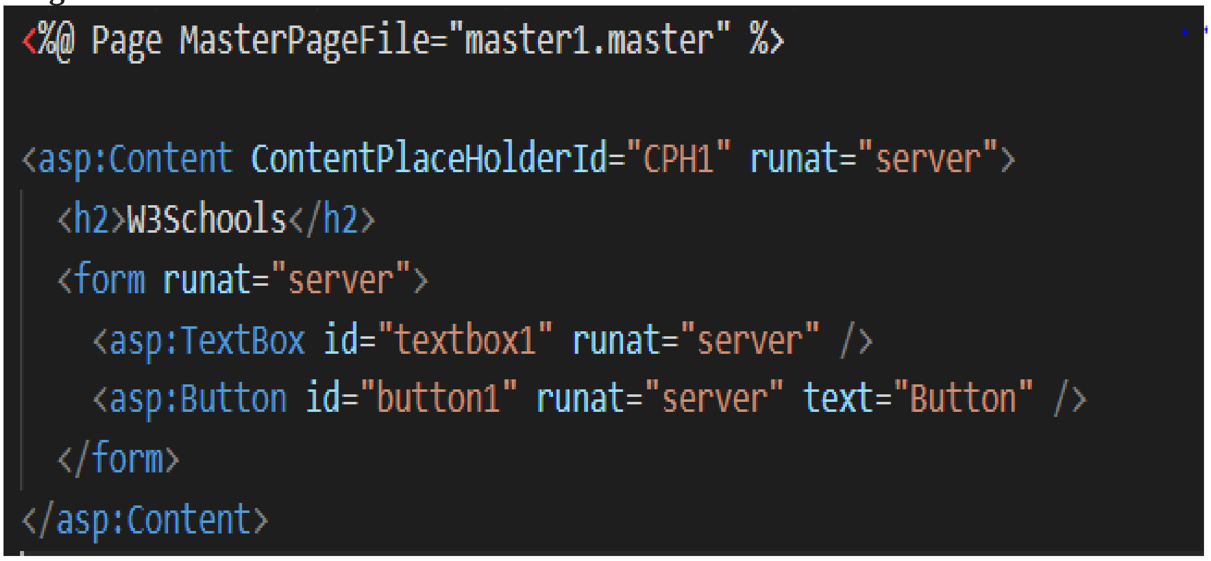


Uses

The master pages can be used to accomplish the following:

- Creating a set of controls that are common across all the web pages and attaching them to all the web pages.
- A centralized way to change the above created set of controls which will effectively change all the web pages.
- Dynamically changing the common UI elements on master page from content pages based on user preferences.

Program:-



```
<%@ Page MasterPageFile="master1.master" %>

<asp:Content ContentPlaceHolderId="CPH1" runat="server">
  <h2>W3Schools</h2>
  <form runat="server">
    <asp:TextBox id="textbox1" runat="server" />
    <asp:Button id="button1" runat="server" text="Button" />
  </form>
</asp:Content>
```

12.What is Theme? Explain Global Theme

An **ASP**.NET **theme** is a collection of properties that define the appearance of pages and controls in your Web site. A **theme** can include skin files, which define property settings for **ASP**.NET Web server controls, and can also include cascading style sheet files (.css files) and graphics.A theme decides the look and feel of the website. It is a collection of files that define the looks of a page. It can include skin files, CSS files & images. We define themes in a special AppThemes folder. Inside this folder is one or more subfolders named Theme1, Theme2 etc. that define the actual themes. The theme property is applied late in the page's life cycle, effectively overriding any customization you may have for individual controls on your page.

How to apply themes

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="
Default" Theme="Theme1"%>
```

Uses of Themes

- Since themes can contain CSS files, images and skins, you can change colors, fonts, positioning and images simply by applying the desired themes.
- You can have as many themes as you want and you can switch between them by setting a single attribute in the web.config file or an individual aspx page. Also you can switch between themes programmatically.
- Setting the themes programmatically, you are offering your users a quick and easy way to change the page to their likings.
- Themes allow you to improve the usability of your site by giving users with vision problems the option to select a high contrast theme with a large font size.

Global Theme:

A Global theme is a theme that is applied to all the web sites on a web server and includes property settings, and graphics. This theme allows us to maintain all the websites on the same web server and define the same style for all the web pages of the web sites.Let's create a theme application with the help of the following steps:

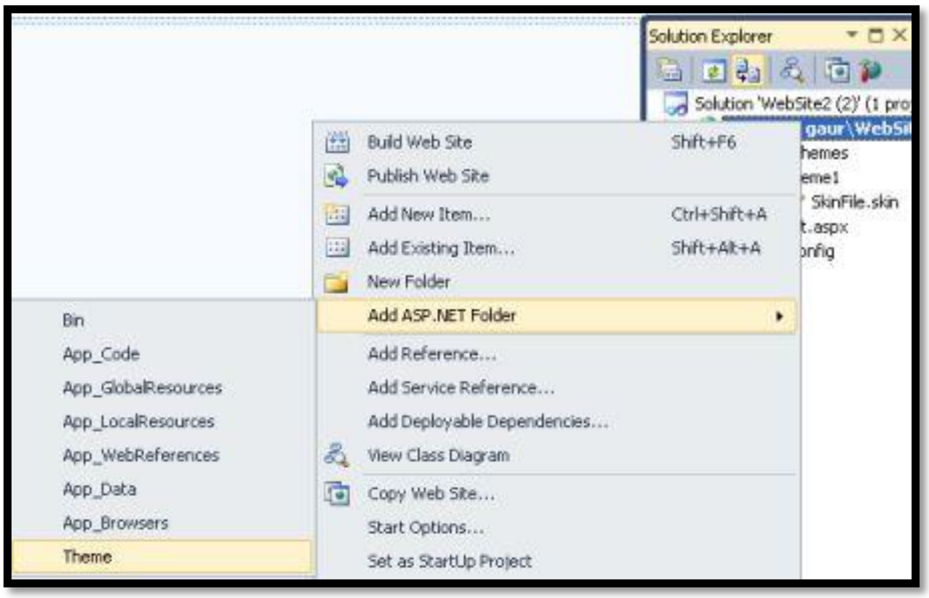
Step 1 : Open Microsoft Visual Studio 2010.

Step 2 : Select File->New->Web Site.

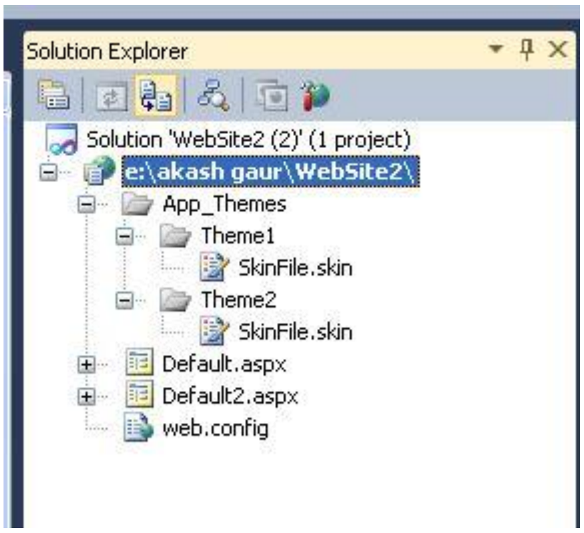
Step 3 : Select ASP.NET Web Site and then click Ok.

Step 4 : Now right-click in the application name in the Solution Explorer window and select Add ASP.NET Folder->Theme from the context menu.

menu.

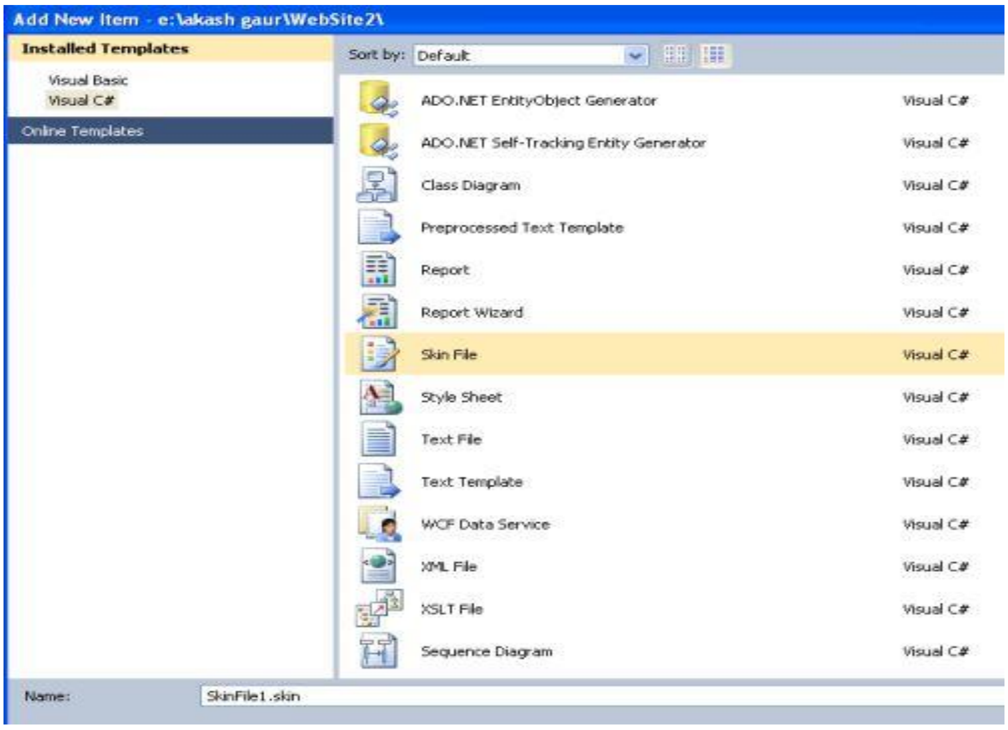


A sub folder named Theme1 is automatically created inside the **APP_Themes** folder in the Solution Explorer.

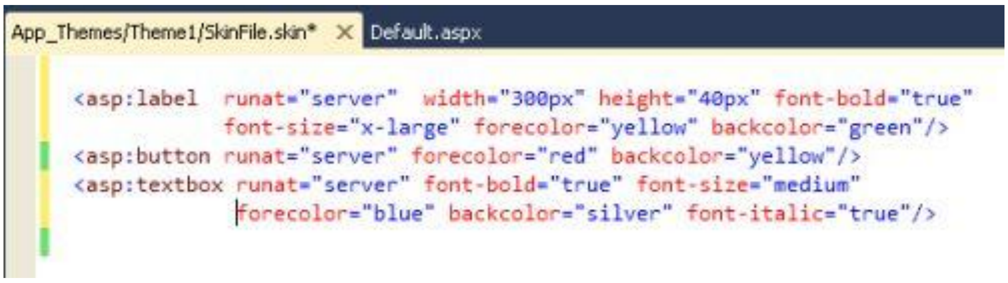


Step 5 : Right-click on the Theme1 folder and select the **Add New Item** option.

Step 6 : Select the Skin file and click the Add button.


Add New Item - e:\akash gaur\WebSite2\
Installed Templates
Visual Basic
Visual C#
Online Templates
Sort by: Default
ADO.NET EntityObject Generator — Visual C#
ADO.NET Self-Tracking Entity Generator — Visual C#
Class Diagram — Visual C#
Preprocessed Text Template — Visual C#
Report — Visual C#
Report Wizard — Visual C#
Skin File — Visual C#
Style Sheet — Visual C#
Text File — Visual C#
Text Template — Visual C#
WCF Data Service — Visual C#
XML File — Visual C#
XSLT File — Visual C#
Sequence Diagram — Visual C#
Name: SkinFile1.skin

Now write the following code in the skin file.



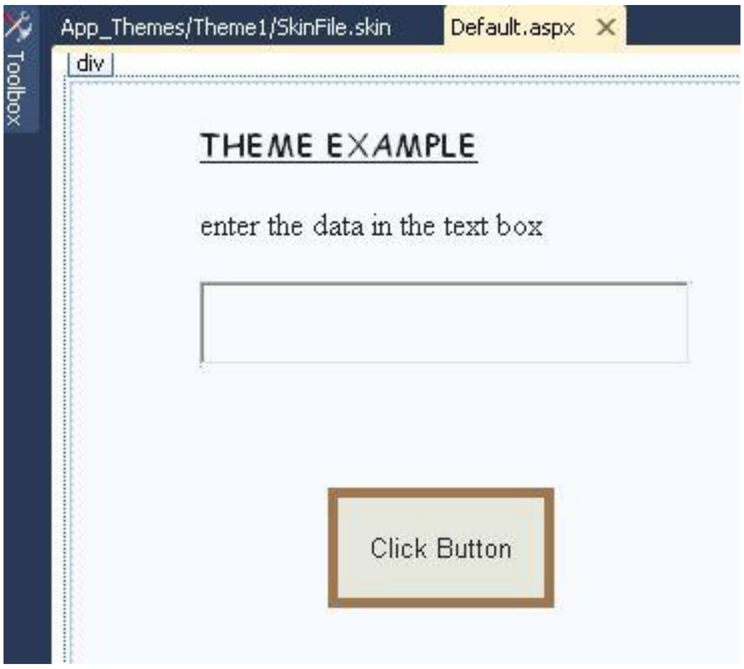
```
<asp:label  runat="server"  width="300px" height="40px" font-bold="true"
            font-size="x-large" forecolor="yellow" backcolor="green"/>
<asp:button runat="server" forecolor="red" backcolor="yellow"/>
<asp:textbox runat="server" font-bold="true" font-size="medium"
            forecolor="blue" backcolor="silver" font-italic="true"/>
```

Step 7 : Write the following code for Default.aspx page.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
      Inherits="_Default" Theme="Theme1" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
   <div>
      <br />

       <asp:Label ID="Label2" runat="server" Font-Bold="False"
              Font-Names="Comic Sans MS" Font-Underline="True"
              Text="THEME EXAMPLE"></asp:Label>
      <br />
      <br />
```

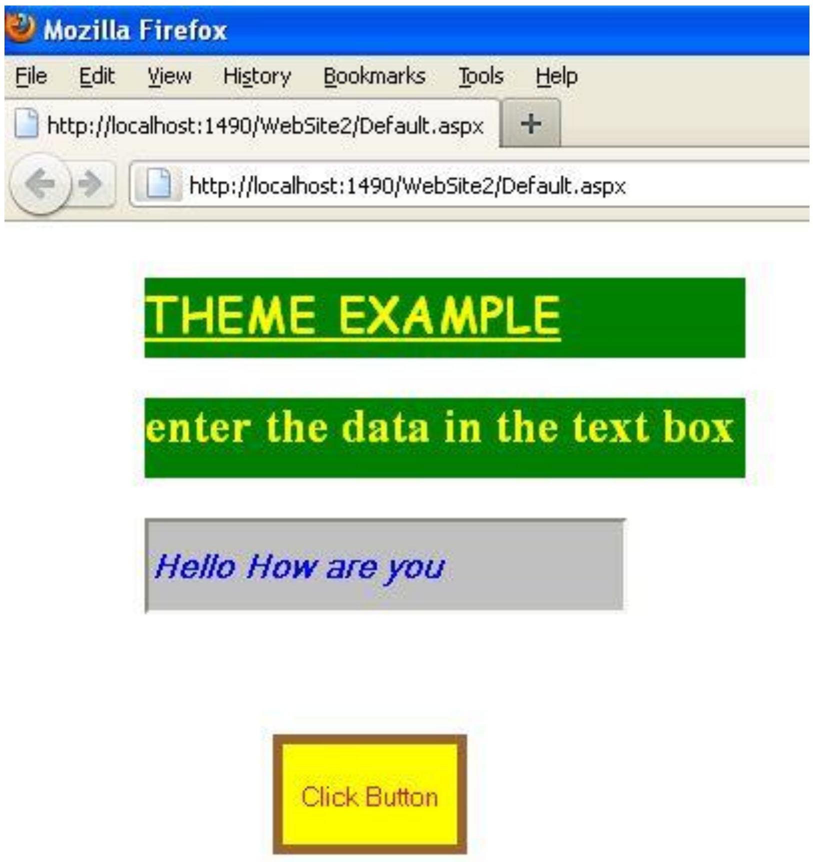
```
        <asp:Label ID="Label1" runat="server" Text="enter the data in the text box">
        </asp:Label>
        &nbsp;&nbsp;<br />
        <asp:TextBox ID="TextBox1" runat="server" Height="42px" Width="237px"></asp:Te
xtBox>
        <br />
        <asp:Button ID="Button1" runat="server" Text="Click Button"

BorderColor="#996633" BorderStyle="Solid" BorderWidth="5px" Height="60px" />
            <asp:Label ID="Label3" runat="server" Text="Select the Theme"></asp:Label>

    </div>
    </form>
</body>
</html>
```

The design window will look like:


App_Themes/Theme1/SkinFile.skin
Default.aspx
Toolbox
div
THEME EXAMPLE
enter the data in the text box
Click Button

Step 8 : Now Press F5 key to run the application.

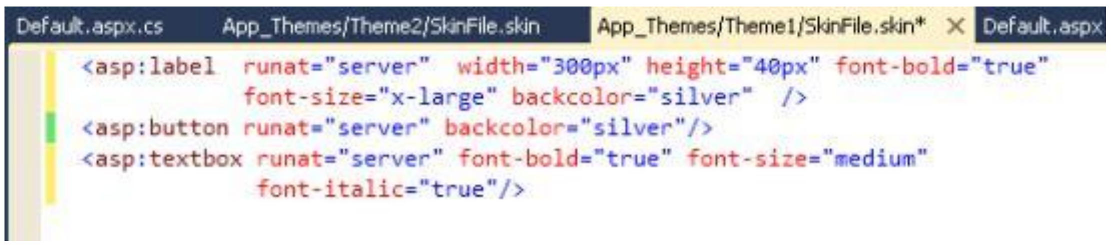
Output :


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:1490/WebSite2/Default.aspx
http://localhost:1490/WebSite2/Default.aspx
THEME EXAMPLE
enter the data in the text box
Hello How are you
Click Button

Now we will create an application for applying the theme at run time.

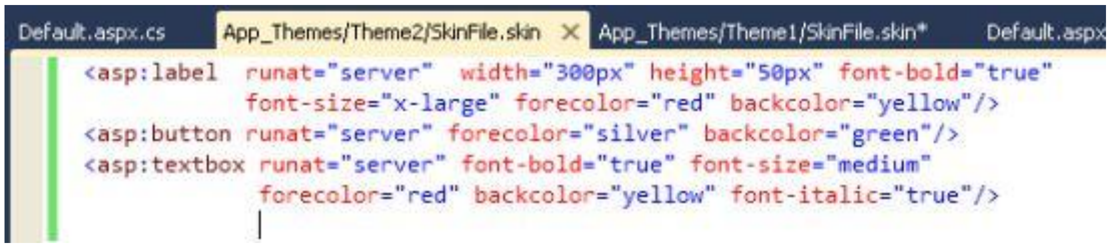
Step 9 : To do this add another skin file to the application and write the following code for both skin files.

Code for simple theme is as follows:


Default.aspx.cs
App_Themes/Theme2/SkinFile.skin
App_Themes/Theme1/SkinFile.skin*
Default.aspx

```
<asp:label  runat="server"  width="300px" height="40px" font-bold="true"
            font-size="x-large" backcolor="silver"  />
<asp:button runat="server" backcolor="silver"/>
<asp:textbox runat="server" font-bold="true" font-size="medium"
             font-italic="true"/>
```

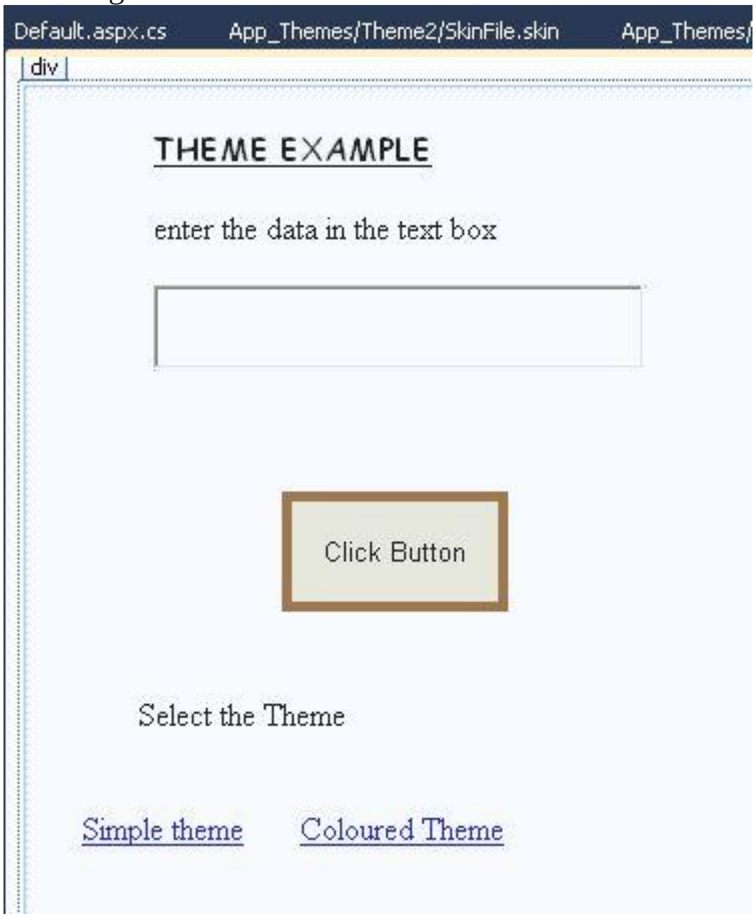
Code for coloured theme is as follows:


Default.aspx.cs
App_Themes/Theme2/SkinFile.skin
App_Themes/Theme1/SkinFile.skin*
Default.aspx

```
<asp:label  runat="server"  width="300px" height="50px" font-bold="true"
            font-size="x-large" forecolor="red" backcolor="yellow"/>
<asp:button runat="server" forecolor="silver" backcolor="green"/>
<asp:textbox runat="server" font-bold="true" font-size="medium"
             forecolor="red" backcolor="yellow" font-italic="true"/>
```

Step 10 : Add the following code to the Default.aspx file:

The design window looks like:



THEME EXAMPLE

enter the data in the text box

Select the Theme

Step 11 : Now write the following code for the default.aspx.cs file:

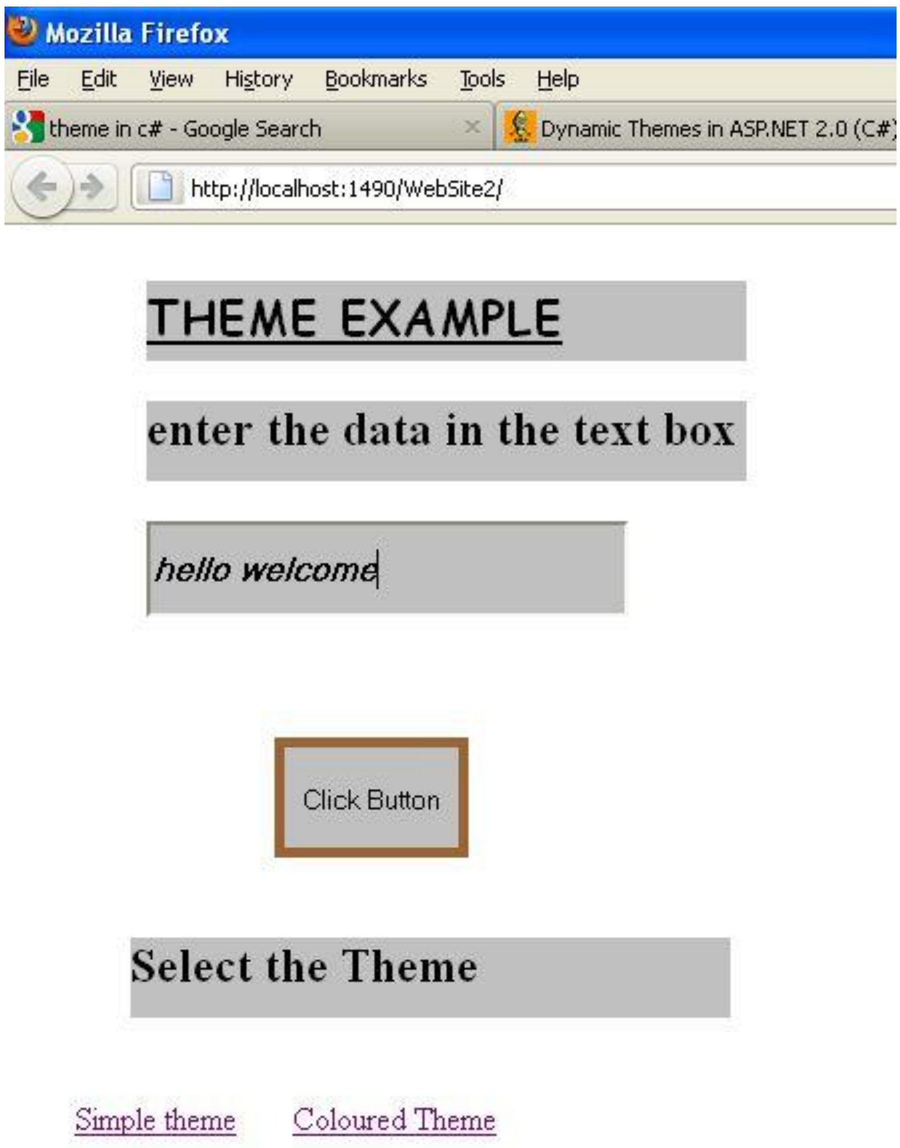
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
public partial class _Default : System.Web.UI.Page
{
    void Page_PreInit(object sender, EventArgs e)
    {
      // Get the theme name from a QueryString variable
       string ThemeName;
       ThemeName = Request.QueryString["Theme"];
       if (ThemeName != null)
       {
          Page.Theme = ThemeName;
```

```
        }
    }
}
```

Step 12 : Press F5 keys to run the application:

Output :

After selecting the Simple theme the output is as follows:


Mozilla Firefox
File Edit View History Bookmarks Tools Help
theme in c# - Google Search
Dynamic Themes in ASP.NET 2.0 (C#)
http://localhost:1490/WebSite2/
Click Button
Simple theme
Coloured Theme

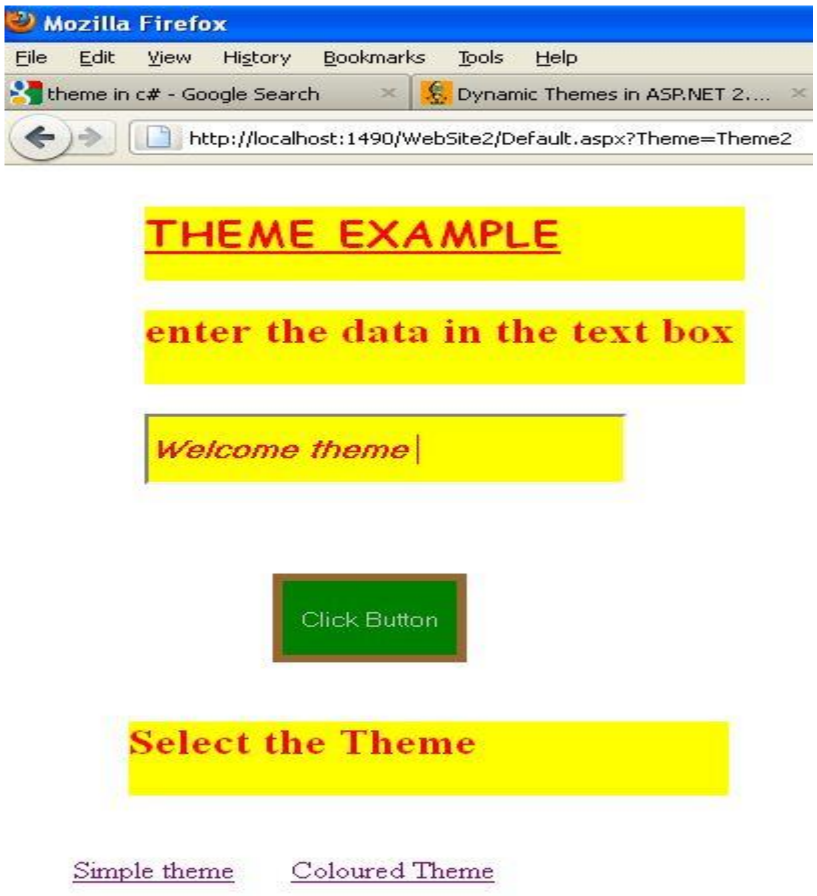
THEME EXAMPLE

enter the data in the text box

hello welcome

Select the Theme

After selecting the Coloured theme the output as follows:


Mozilla Firefox
File Edit View History Bookmarks Tools Help
theme in c# - Google Search
Dynamic Themes in ASP.NET 2....
Click Button
Simple theme
Coloured Theme

http://localhost:1490/WebSite2/Default.aspx?Theme=Theme2

THEME EXAMPLE

enter the data in the text box

Welcome theme

Select the Theme