**14CS505**

# III/IV B.Tech (Regular ) DEGREE EXAMINATION
## Scheme of Evaluation

**November, 2016** **Computer Science and Engineering**

**Fifth Semester** **Enterprise Programming-1**

*Answer Question No 1 compulsorily* (1x12=12Marks)

*Answer ONE question from each Unit* (4x12=48Marks)

**I. Answer the following.** (12x1M=12Marks)

**a) Define Web Server .List any two web servers.**

A web server, is a software,that is used to deliver web pages to clients using the Hypertext Transfer Protocol (HTTP). For example, IIS is a web server that can be used to run asp.net web applications. **Web servers: IIS SERVER,APACHE.**

**b) What is the difference between Server.Transfer and Response.Redirect?**

The Response.Redirect() method is used to send the user to any type of page,even send the user to another website by using an absolute URL.

The Server.Transfer() method allows you to jump only from one ASP.NET page to another, in the same web application.

**c) What is the purpose of Web.Config file?**

Web.Config file includes settings for customizing security, state management, memory management.

**d) List common properties of ASP.NET validation controls**

ControlToValidate

ErrorMessage

ForeColor

Display

**e) Compare Cookie and Session.**

| COOKIE | SESSION |
|---|---|
| **1.** Do not support all data types. | **1.**Supports all data types. |
| **2.** Insecure | **2.** Highly secure |
| **3.** preferences for a website | **3.** Storing items in a shopping basket |

**f) Write the syntax for Adrotator element.**

```
<asp:AdRotator ID="Ads"
AdvertisementFile="MainAds.xml"
runat="server"
KeywordFilter=”Computer” />
```

**g) What are the different types of Styles ?**

i) Inline style

ii) internal style

iii)external style

**h) Create an object for SqlConnection class.**

SqlConnection con=new SqlConnection("data source=.\\sqlexpress;database=master;user id=sa;password=sqlserver"); Here "con" is the object created.

**i) What are the different templates available in GridView.**

HeaderTemplate

FooterTemplate

ItemTemplate

EditItemTemplate

InsertItemTemplate

**j) Explain about LINQ to Objects.**

**LINQ to Objects:** This is the simplest form of LINQ. It allows you to query collections of in-memory objects. LINQ return an usual type of object, called an iterator object. Although the iterator object looks like an ordinary collection to your code, it doesn't hold any information.LINQ evaluates your expression and quickly grabs the information you need. This trick is called deferred execution.

**k) Define Virtual Directory.**

Virtual directory is the path provided to files in physical directory so that any remote user can access those files .

**l) What are the advantages of Web Services?**

Web Services also let developers use their preferred programming languages.

Web Services are virtually platform-independent.

Reusability.

## UNIT-I

**2a)What is .NET? Explain features of .Net in detail. (6M)**

**.NET:** .NET is a cluster of technologies that are designed to help developers build a variety of applications.

**The .NET languages**: These include Visual Basic, C#, F#, and C++, although third-party developers have created hundreds more.

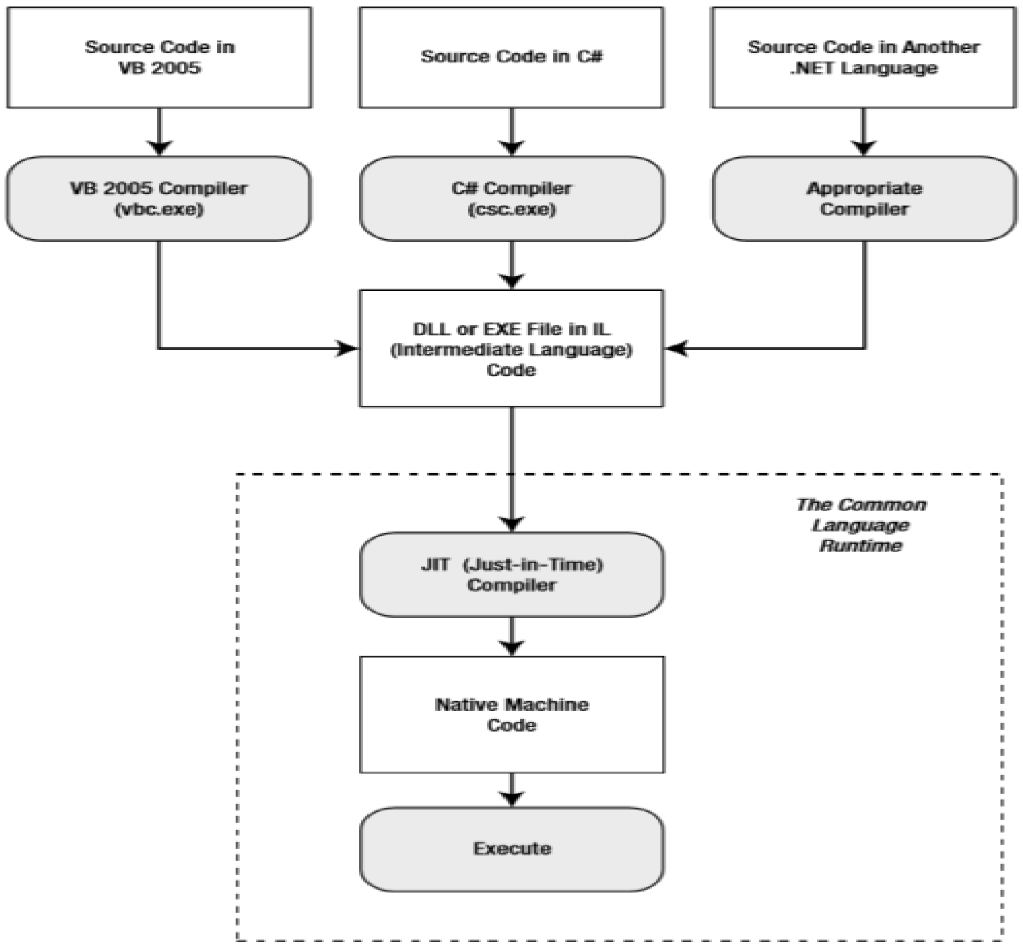
**The Common Language Runtime (CLR):** This is the engine that executes all .NET programs and provides automatic services for these applications, such as security checking, memory management, and optimization.

**The .NET Framework class library**: The class library collects thousands of pieces of prebuilt functionality that you can "snap in" to your applications. These features are sometimes organized into technology sets, such as ADO.NET (the technology for creating database applications) and Windows Presentation Foundation (WPF, the technology for creating desktop user interfaces).

**ASP.NET:** This is the engine that hosts the web applications you create with .NET, and supports almost any feature from the .NET Framework class library. ASP.NET also includes a set of web-specific services, such as secure authentication and data storage.

**Visual Studio**: This optional development tool contains a rich set of productivity and debugging features. Visual Studio includes the complete .NET Framework, so you won't need to download it separately.

**Language Compilation in .Net**


Source Code in VB 2005
Source Code in C#
Source Code in Another .NET Language
VB 2005 Compiler (vbc.exe)
C# Compiler (csc.exe)
Appropriate Compiler
DLL or EXE File in IL (Intermediate Language) Code
The Common Language Runtime
JIT (Just-in-Time) Compiler
Native Machine Code
Execute

**The Common Language Runtime:**

The CLR is the engine that supports all the .NET languages. All .NET code runs inside the CLR.

**Deep language integration:** VB and C#, like all .NET languages, compile to IL. In other words, the CLR makes no distinction between different languages—in fact, it has no way of knowing what language was used to create an executable. This is far more than mere language compatibility; it's language integration.

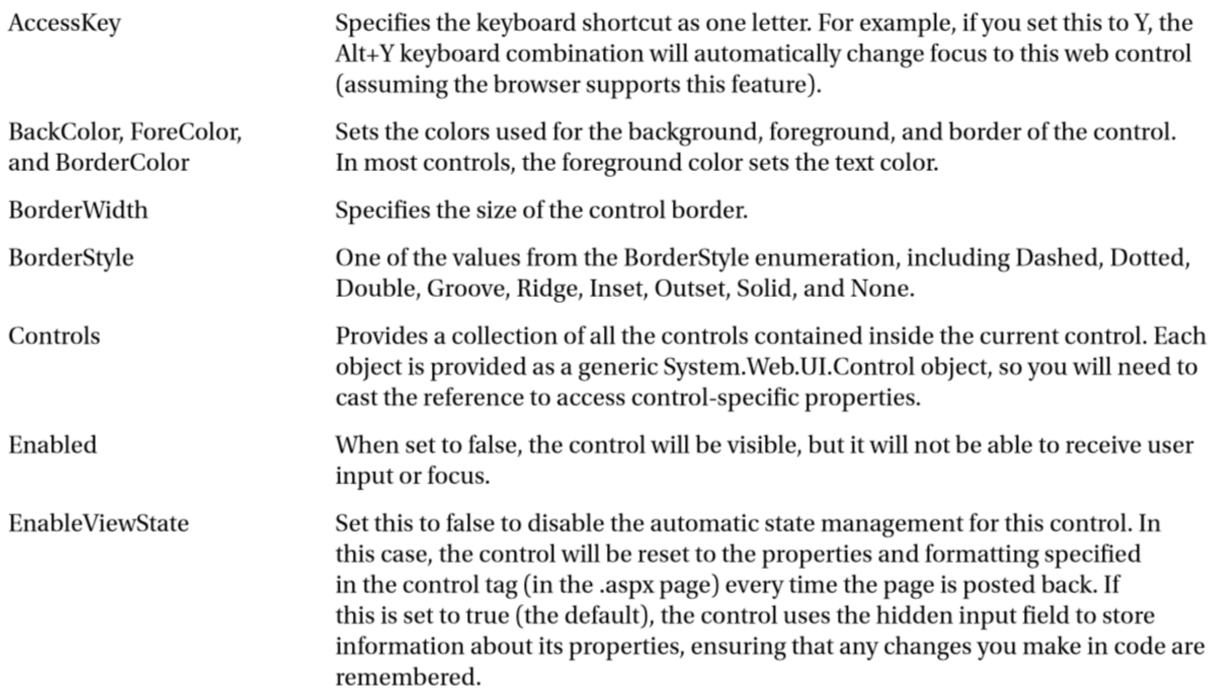
**Side-by-side execution**: The CLR also has the ability to load more than one version of a component at a time. In other words, you can update a component many times, and the correct version will be loaded and used for each application. As a side effect, multiple versions of the .NET Framework can be installed, meaning that you're able to upgrade to new versions of ASP.NET without replacing the current version or needing to rewrite your applications.

**Fewer errors:** Whole categories of errors are impossible with the CLR. For example, the CLR prevents many memory mistakes that are possible with lower-level languages such as C++.

**2b) Design user registration form using ASP.NET web controls. Explain the properties of each control (6M)**

**WEB CONTROLS:**

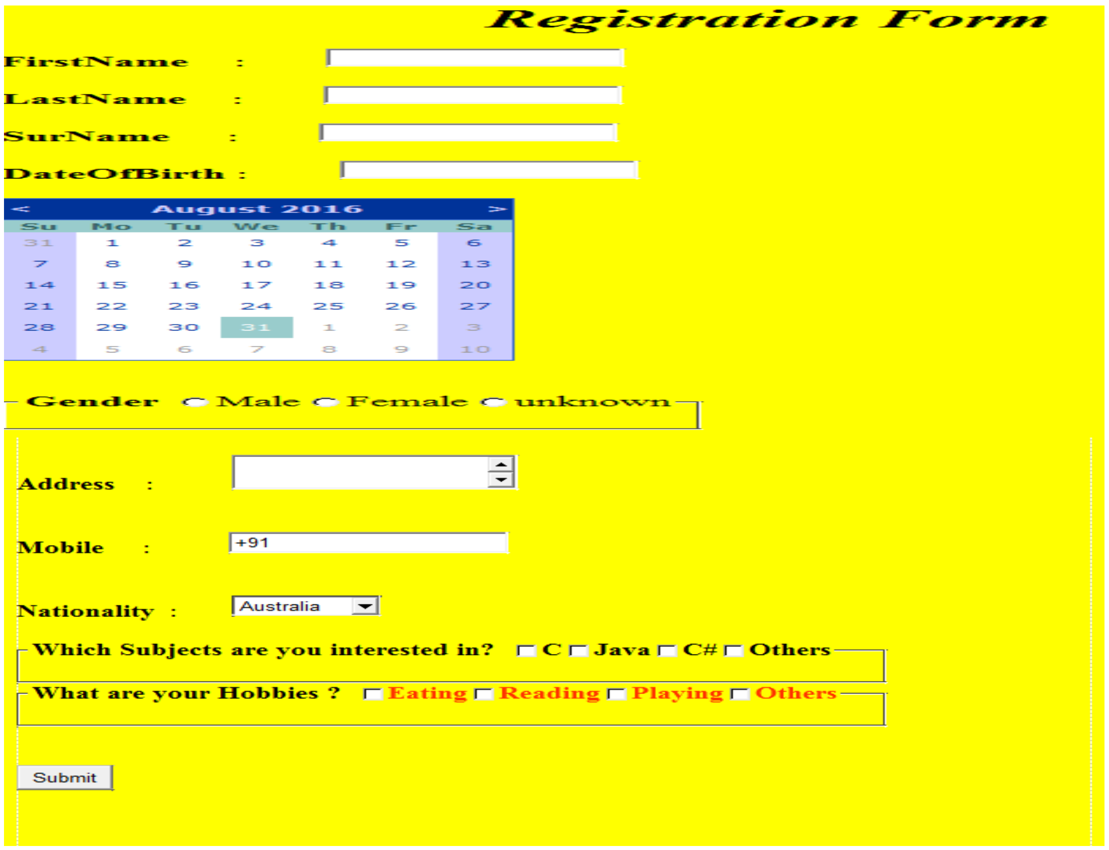
Label,TextBox,CheckBox,RadioButton,DropDownList,button,Calendar,HyperLink,LinkButton, DropdownList, BulletedList, CheckBoxList, RadioButtonList.



| | |
|---|---|
| AccessKey | Specifies the keyboard shortcut as one letter. For example, if you set this to Y, the Alt+Y keyboard combination will automatically change focus to this web control (assuming the browser supports this feature). |
| BackColor, ForeColor, and BorderColor | Sets the colors used for the background, foreground, and border of the control. In most controls, the foreground color sets the text color. |
| BorderWidth | Specifies the size of the control border. |
| BorderStyle | One of the values from the BorderStyle enumeration, including Dashed, Dotted, Double, Groove, Ridge, Inset, Outset, Solid, and None. |
| Controls | Provides a collection of all the controls contained inside the current control. Each object is provided as a generic System.Web.UI.Control object, so you will need to cast the reference to access control-specific properties. |
| Enabled | When set to false, the control will be visible, but it will not be able to receive user input or focus. |
| EnableViewState | Set this to false to disable the automatic state management for this control. In this case, the control will be reset to the properties and formatting specified in the control tag (in the .aspx page) every time the page is posted back. If this is set to true (the default), the control uses the hidden input field to store information about its properties, ensuring that any changes you make in code are remembered. |

| | |
|---|---|
| Font | Specifies the font used to render any text in the control as a System.Web.UI.WebControls.FontInfo object. |
| Height and Width | Specifies the width and height of the control. For some controls, these properties will be ignored when used with older browsers. |
| ID | Specifies the name that you use to interact with the control in your code (and also serves as the basis for the ID that's used to name the top-level element in the rendered HTML). |
| Page | Provides a reference to the web page that contains this control as a System.Web.UI.Page object. |
| Parent | Provides a reference to the control that contains this control. If the control is placed directly on the page (rather than inside another control), it will return a reference to the page object. |
| TabIndex | A number that allows you to control the tab order. The control with a TabIndex of 0 has the focus when the page first loads. Pressing Tab moves the user to the control with the next lowest TabIndex, provided it is enabled. This property is supported only in Internet Explorer. |
| ToolTip | Displays a text message when the user hovers the mouse above the control. Many older browsers don't support this property. |
| Visible | When set to false, the control will be hidden and will not be rendered to the final HTML page that is sent to the client. |

**DESIGN VIEW :**

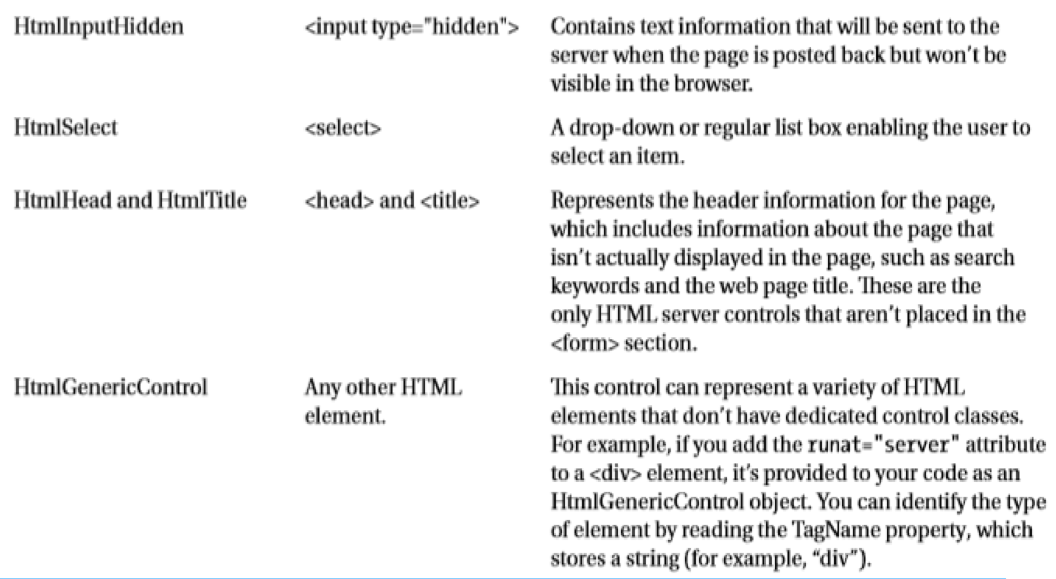

Registration Form
FirstName :
LastName :
SurName :
DateOfBirth :
< August 2016 >
Su Mo Tu We Th Fr Sa
Gender ○ Male ○ Female ○ unknown
Address :
Mobile : +91
Nationality : Australia
Which Subjects are you interested in? ☐ C ☐ Java ☐ C# ☐ Others
What are your Hobbies ? ☐ Eating ☐ Reading ☐ Playing ☐ Others
Submit

**DEFAULT.ASPX.CS :**

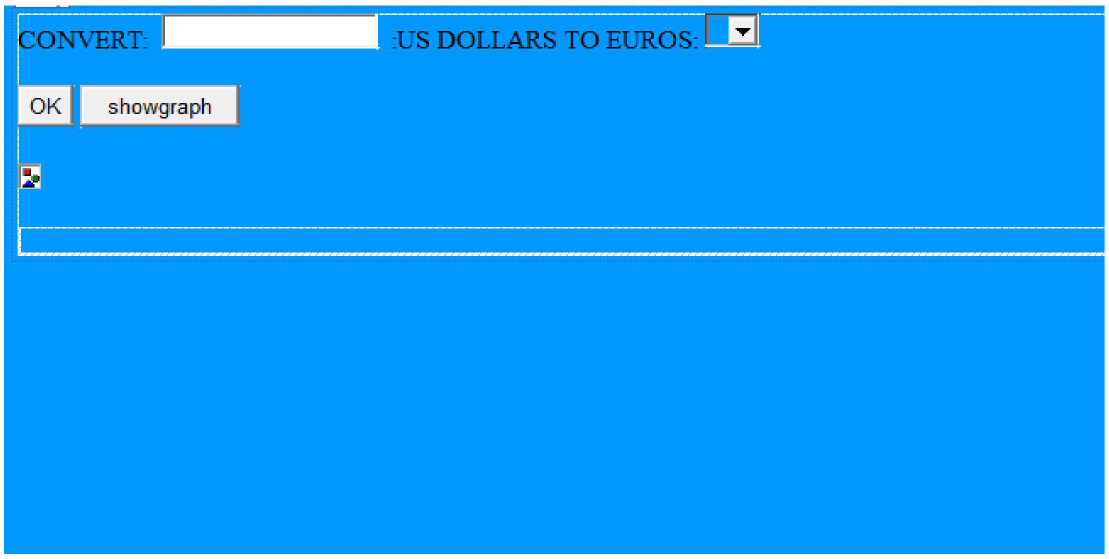
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

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

    }

    protected void Calendar1_SelectionChanged(object sender, EventArgs e)
    {
        DateTime k = Calendar1.SelectedDate;
        TextBox4.Text= k.ToString();
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
       if(RadioButton1.Checked)
        {
            Response.Write("you are"+ RadioButton1.Text+ "<br/>");
        }
        else if (RadioButton2.Checked)
        {
            Response.Write("you are" + RadioButton2.Text + "<br/>");
        }
        else if (RadioButton3.Checked)
        {
            Response.Write("you are" + RadioButton3.Text + "<br/>");
        }



        if (CheckBox1.Checked)
        {
            Response.Write("your interested subject1:" + CheckBox1.Text + "<br />");
        }
        if (CheckBox2.Checked)
        {
            Response.Write("your interested subject2:" + CheckBox2.Text + "<br />");
        }
         if (CheckBox3.Checked)
        {
           Response.Write("your interested subject1:" + CheckBox3.Text + "<br />");
        }
         if(CheckBox4.Checked)
        {
           Response.Write("You did not mention your favourite subject");
        }
        if (CheckBox5.Checked)
        {
           Response.Write("your hobby1 is:" + CheckBox5.Text + "<br />");
        }
        if (CheckBox6.Checked)
        {
           Response.Write("your hobby2 is:" + CheckBox6.Text + "<br />");
        }
        if (CheckBox7.Checked)
        {
           Response.Write("Your hobbie3 is:" + CheckBox6.Text + "<br />");
        }
        if (CheckBox8.Checked)
        {
           Response.Write("You did not mention your hobby");
        }
        string a = TextBox1.Text;
        string b = TextBox2.Text;
        string c = TextBox3.Text;
        string d = TextBox4.Text;
        string f = TextBox5.Text;
        string g = TextBox6.Text;
        string h = DropDownList1.Text;
        para.InnerHtml ="firstname"+ a + "<br />" +"second name"+ b + "<br />" +"surname"+ c +
"<br />"+
           "DOB"  + d + "<br />" +"address"+ f + "<br />"+"contact" + g + "<br />" +"nationality"+ h;
    }

}
```

**3a) Explain HTML Server Control classes with an example .** **(6M)**

| Class Name | HTML Element | Description |
|---|---|---|
| HtmlForm | &lt;form&gt; | Wraps all the controls on a web page. All ASP.NET server controls must be placed inside an HtmlForm control so that they can send their data to the server when the page is submitted. Visual Studio adds the &lt;form&gt; element to all new pages. When designing a web page, you need to ensure that every other control you add is placed inside the &lt;form&gt; section. |
| HtmlAnchor | &lt;a&gt; | A hyperlink that the user clicks to jump to another page. |
| HtmlImage | &lt;img&gt; | A link that points to an image, which will be inserted into the web page at the current location. |
| HtmlTable, HtmlTableRow, and HtmlTableCell | &lt;table&gt;, &lt;tr&gt;, &lt;th&gt;, and &lt;td&gt; | A table that displays multiple rows and columns of static text. |
| HtmlInputButton, HtmlInputSubmit, and HtmlInputReset | &lt;input type="button"&gt;, &lt;input type="submit"&gt;, and &lt;input type="reset"&gt; | A button that the user clicks to perform an action (HtmlInputButton), submit the page (HtmlInputSubmit), or clear all the user-supplied values in all the controls (HtmlInputReset). |
| HtmlButton | &lt;button&gt; | A button that the user clicks to perform an action. This is not supported by all browsers, so HtmlInputButton is usually used instead. The key difference is that the HtmlButton is a container element. As a result, you can insert just about anything inside it, including text and pictures. The HtmlInputButton, on the other hand, is strictly text-only. |
| HtmlInputCheckBox | &lt;input type="checkbox"&gt; | A check box that the user can select or clear. Doesn't include any text of its own. |
| HtmlInputRadioButton | &lt;input type="radio"&gt; | A radio button that can be selected in a group. Doesn't include any text of its own. |
| HtmlInputText and HtmlInputPassword | &lt;input type="text"&gt; and &lt;input type="password"&gt; | A single-line text box enabling the user to enter information. Can also be displayed as a password field (which displays bullets instead of characters to hide the user input). |
| HtmlTextArea | &lt;textarea&gt; | A large text box for typing multiple lines of text. |



| | | |
|---|---|---|
| HtmlInputHidden | <input type="hidden"> | Contains text information that will be sent to the server when the page is posted back but won't be visible in the browser. |
| HtmlSelect | <select> | A drop-down or regular list box enabling the user to select an item. |
| HtmlHead and HtmlTitle | <head> and <title> | Represents the header information for the page, which includes information about the page that isn't actually displayed in the page, such as search keywords and the web page title. These are the only HTML server controls that aren't placed in the <form> section. |
| HtmlGenericControl | Any other HTML element. | This control can represent a variety of HTML elements that don't have dedicated control classes. For example, if you add the runat="server" attribute to a <div> element, it's provided to your code as an HtmlGenericControl object. You can identify the type of element by reading the TagName property, which stores a string (for example, "div"). |



**DEFAULT.ASPX.CS :**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class CC : System.Web.UI.Page
{
   protected void Page_Load(object sender, EventArgs e)
   {
      Currency.Items.Add(new ListItem("euros", "45"));
      Currency.Items.Add(new ListItem("japanese_yen", "65"));
      Currency.Items.Add(new ListItem("canadian_dollars", "85"));
```

```
        graph.Visible = false;
    }
    protected void convert_serverclick(object sender, EventArgs e)
    {
        Decimal oldamount;
        bool success = Decimal.TryParse(US.Value, out oldamount);
        if ((oldamount <= 0) || (success == false))
        {
            result.Style["color"] = "red";
            result.InnerText = "specify correct format";
        }
        else
        {
            result.Style["color"] = "YELLOW";
        }
        ListItem li = Currency.Items[Currency.SelectedIndex];
        Decimal newamount = oldamount * Decimal.Parse(li.Value);
        result.InnerText = oldamount.ToString() + "us dollars to";
        result.InnerText += newamount.ToString() + li.Text;
    }
```

**3b) Explain in detail about Tracing. (6M)**

ASP. NET provides a feature called tracing that gives you a far more convenient and flexible way to report diagnostic information.

To use tracing, you need to explicitly enable it. There are several ways to switch on tracing. One of the easiest ways is by adding an attribute to the Page directive in the .aspx file**:**

**<%@ Page Trace="true" … %>**

Enabling tracing by using the built-in Trace object:

```
protected void Page_Load(Object sender, EventArgs e)
{
  Trace.IsEnabled = true;
 }
```

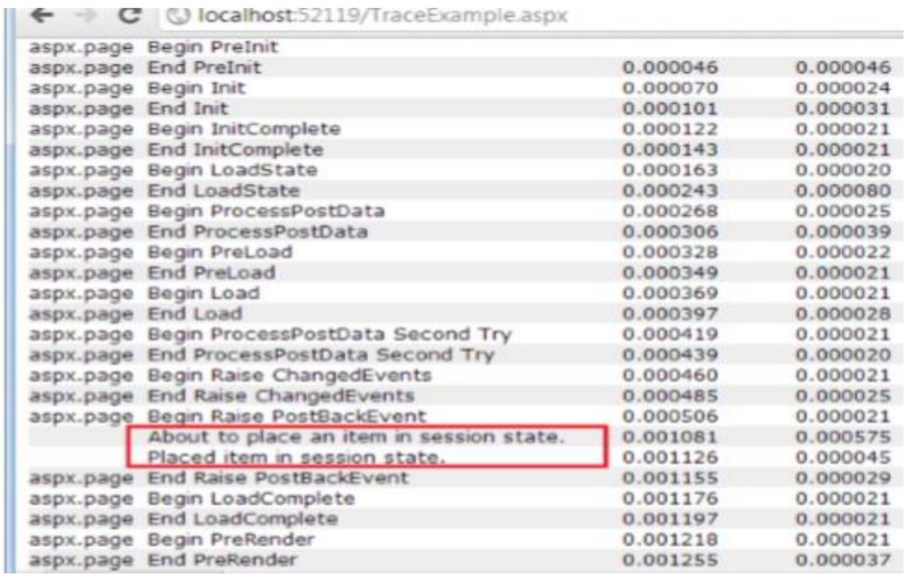
**Performing Application-Level Tracing:**

To enable application-level tracing, you need to modify settings in the web.config file, as shown here:

```
<configuration>
<system.web>
<trace enabled="true" requestLimit="10" pageOutput="false"  local only="true" most recent="true"/>
</system.web>
 </configuration>
```

**Writing Trace Information:**

```
protected void cmdWrite_Click(Object sender, EventArgs e)
{    Trace.Write("About to place an item in session state.");
    Session["Test"] = "Contents";
    Trace.Write("Placed item in session state.");
 }
```



localhost:52119/TraceExample.aspx

```
aspx.page  Begin PreInit
aspx.page  End PreInit                                   0.000046    0.000046
aspx.page  Begin Init                                    0.000070    0.000024
aspx.page  End Init                                      0.000101    0.000031
aspx.page  Begin InitComplete                            0.000122    0.000021
aspx.page  End InitComplete                              0.000143    0.000021
aspx.page  Begin LoadState                               0.000163    0.000020
aspx.page  End LoadState                                 0.000243    0.000080
aspx.page  Begin ProcessPostData                         0.000268    0.000025
aspx.page  End ProcessPostData                           0.000306    0.000039
aspx.page  Begin PreLoad                                 0.000328    0.000022
aspx.page  End PreLoad                                   0.000349    0.000021
aspx.page  Begin Load                                    0.000369    0.000021
aspx.page  End Load                                      0.000397    0.000028
aspx.page  Begin ProcessPostData Second Try              0.000419    0.000021
aspx.page  End ProcessPostData Second Try                0.000439    0.000020
aspx.page  Begin Raise ChangedEvents                     0.000460    0.000021
aspx.page  End Raise ChangedEvents                       0.000485    0.000025
aspx.page  Begin Raise PostBackEvent                     0.000506    0.000021
           About to place an item in session state.      0.001081    0.000575
           Placed item in session state.                 0.001126    0.000045
aspx.page  End Raise PostBackEvent                       0.001155    0.000029
aspx.page  Begin LoadComplete                            0.001176    0.000021
aspx.page  End LoadComplete                              0.001197    0.000021
aspx.page  Begin PreRender                               0.001218    0.000021
aspx.page  End PreRender                                 0.001255    0.000037
```

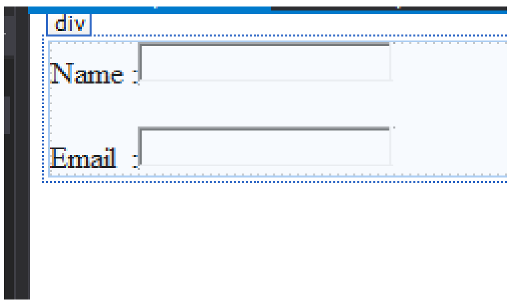
**UNIT-II**

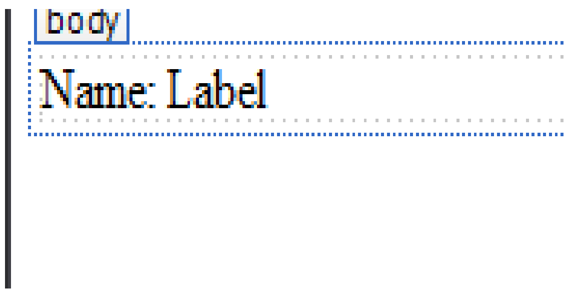
**4a) Discuss about Session State in detail . (6M)**

Session-state is one of ASP.NET's premiere features. It allows you to store any type of data in memory on the server. The information is protected, because it is never transmitted to the client, and it's uniquely bound to a specific session

ASP.NET tracks each session by using a unique 120-bit identifiers. ASP.NET uses a proprietary algorithm to generate this value, thereby guaranteeing (statistically speaking) that the number is unique and it's random enough that a malicious user can't reverse-engineer or "guess" what session ID a given client will be using. This ID is the only piece of session-related information that is transmitted between the web server and the client.

**Default.aspx** **Default2.aspx**





**DEFAULT.ASPX.CS**

using System;
using System.Collections.Generic;

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

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

   }

   protected void Button1_Click(object sender, EventArgs e)
   {
      Session[" Name"] = TextBox1.Text;
      Session["Emai"]=TextBox2.Text;l
      Response.Redirect("default2.aspx");
   }
}
```

**DEFAULT2.ASPX.CS**

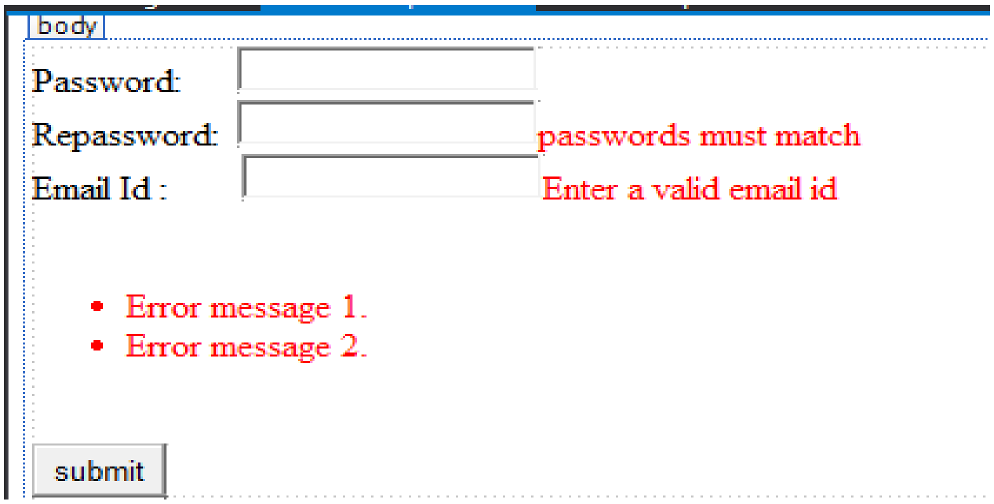
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class Default2 : System.Web.UI.Page
{
   protected void Page_Load(object sender, EventArgs e)
   {

         Label1.Text = Session[" Name"].ToString();
         Label2.Text=Session["Email"].ToString();
   }
}
```

**4b) Explain the following validation controls with an example. (6M)**

**i) Regular Expression ii) Compare validator iii) Validation Summary**

**RegularExpressionValidator:** Validation succeeds if the value in an input control matches a specified regular expression.

**CompareValidator:** Validation succeeds if the input control contains a value that matches the value in another input control, or a fixed value that you specify.
**ValidationSummary:** It displays the list of errors, it automatically retrieves the value of the ErrorMessage property from each validator.



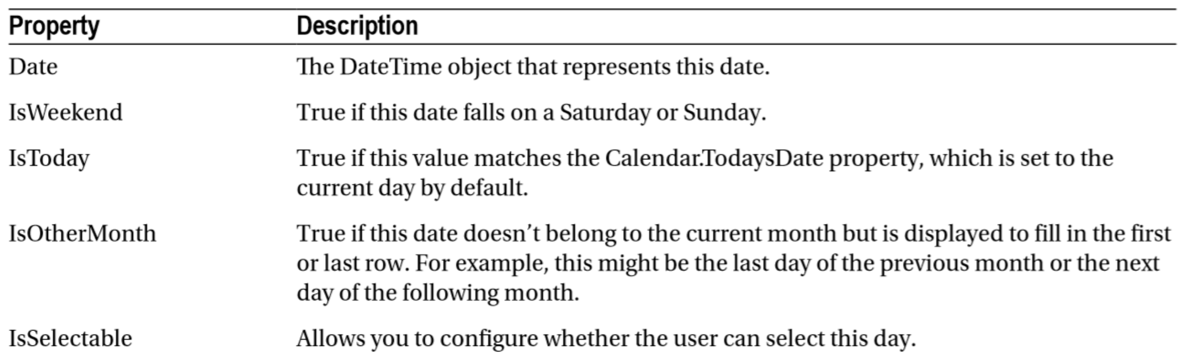
passwords must match
Enter a valid email id

- Error message 1.
- Error message 2.

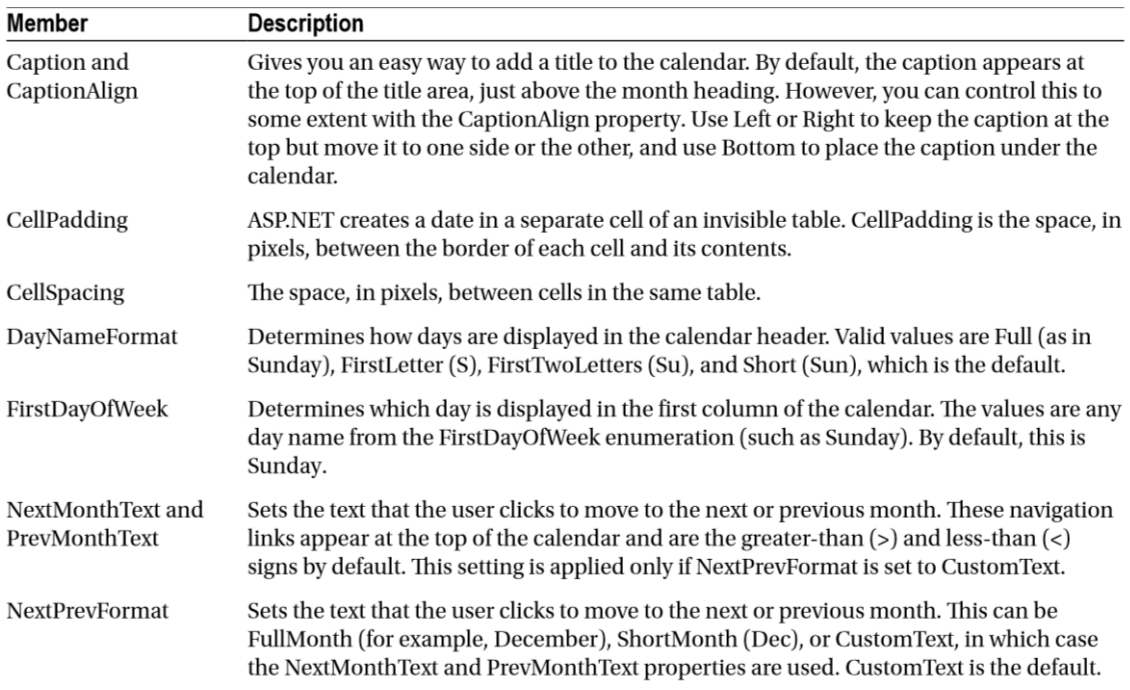
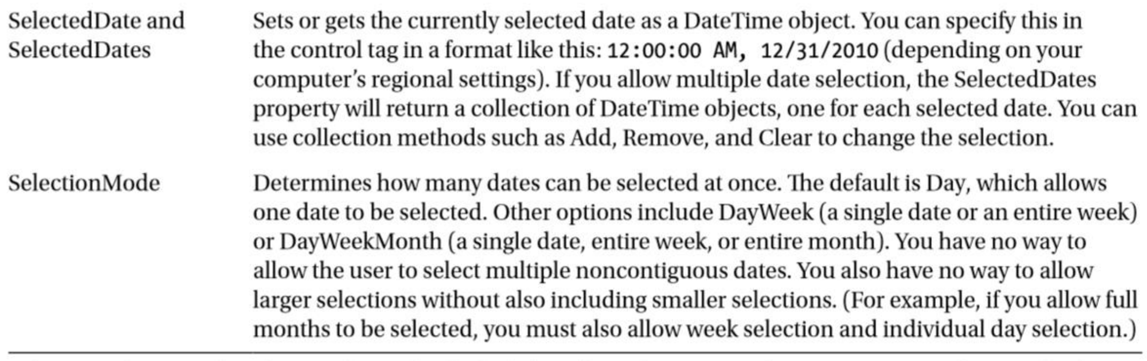
```
Password:&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <br />
        Repassword:&nbsp;
        <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
        <asp:CompareValidator ID="CompareValidator1" runat="server"
ControlToCompare="TextBox1" ControlToValidate="TextBox2" ErrorMessage="passwords must
match" ForeColor="Red"></asp:CompareValidator>
        <br />
        Email Id :&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
        <asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
        <asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"
ControlToValidate="TextBox3" ErrorMessage="Enter a valid email id"
ForeColor="Red"></asp:RegularExpressionValidator>
        <br />
        <asp:ValidationSummary ID="ValidationSummary1" runat="server" ForeColor="Red" />
        <br />
        <asp:Button ID="Button1" runat="server" Text="submit" />
```

**UNIT-III**

**5a) What are properties of Calendar control ? Write a program to compare two dates. (6M)**

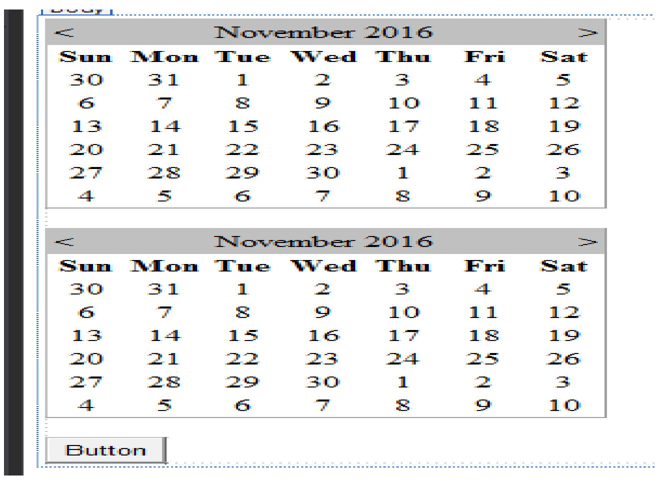


| Property | Description |
|---|---|
| Date | The DateTime object that represents this date. |
| IsWeekend | True if this date falls on a Saturday or Sunday. |
| IsToday | True if this value matches the Calendar.TodaysDate property, which is set to the current day by default. |
| IsOtherMonth | True if this date doesn't belong to the current month but is displayed to fill in the first or last row. For example, this might be the last day of the previous month or the next day of the following month. |
| IsSelectable | Allows you to configure whether the user can select this day. |

| Member | Description |
|---|---|
| Caption and CaptionAlign | Gives you an easy way to add a title to the calendar. By default, the caption appears at the top of the title area, just above the month heading. However, you can control this to some extent with the CaptionAlign property. Use Left or Right to keep the caption at the top but move it to one side or the other, and use Bottom to place the caption under the calendar. |
| CellPadding | ASP.NET creates a date in a separate cell of an invisible table. CellPadding is the space, in pixels, between the border of each cell and its contents. |
| CellSpacing | The space, in pixels, between cells in the same table. |
| DayNameFormat | Determines how days are displayed in the calendar header. Valid values are Full (as in Sunday), FirstLetter (S), FirstTwoLetters (Su), and Short (Sun), which is the default. |
| FirstDayOfWeek | Determines which day is displayed in the first column of the calendar. The values are any day name from the FirstDayOfWeek enumeration (such as Sunday). By default, this is Sunday. |
| NextMonthText and PrevMonthText | Sets the text that the user clicks to move to the next or previous month. These navigation links appear at the top of the calendar and are the greater-than (>) and less-than (<) signs by default. This setting is applied only if NextPrevFormat is set to CustomText. |
| NextPrevFormat | Sets the text that the user clicks to move to the next or previous month. This can be FullMonth (for example, December), ShortMonth (Dec), or CustomText, in which case the NextMonthText and PrevMonthText properties are used. CustomText is the default. |
| SelectedDate and SelectedDates | Sets or gets the currently selected date as a DateTime object. You can specify this in the control tag in a format like this: `12:00:00 AM, 12/31/2010` (depending on your computer's regional settings). If you allow multiple date selection, the SelectedDates property will return a collection of DateTime objects, one for each selected date. You can use collection methods such as Add, Remove, and Clear to change the selection. |
| SelectionMode | Determines how many dates can be selected at once. The default is Day, which allows one date to be selected. Other options include DayWeek (a single date or an entire week) or DayWeekMonth (a single date, entire week, or entire month). You have no way to allow the user to select multiple noncontiguous dates. You also have no way to allow larger selections without also including smaller selections. (For example, if you allow full months to be selected, you must also allow week selection and individual day selection.) |

**Design View:**



| < | | | November 2016 | | | > |
|---|---|---|---|---|---|---|
| Sun | Mon | Tue | Wed | Thu | Fri | Sat |
| 30 | 31 | 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| 27 | 28 | 29 | 30 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 |

| < | | | November 2016 | | | > |
|---|---|---|---|---|---|---|
| Sun | Mon | Tue | Wed | Thu | Fri | Sat |
| 30 | 31 | 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| 27 | 28 | 29 | 30 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 |

**Default.aspx.cs:**

```
protected void Button1_Click(object sender, EventArgs e)
  {
     DateTime d = Calendar1.SelectedDate;
     DateTime f = Calendar2.SelectedDate;
     if (d < f)
        Response.Write("Less Than");
     else if(d > f)
        Response.Write("greater Than");
     else
         Response.Write("equal");

  }
```

**5b)What are the advantages of master pages ?How master page and content pages are connected to each other . (6M)**

i. Master pages enable consistent and standardized layout of the website.

ii. You can make layout changes of the site in master page instead of making changes in the pages.

iii. It provide an object model which allows you to customize the master page from individual content pages.

iv. It allows you to centralize the common functionality of your pages so that you can make updates in just one place.

**Master Pages and Content Pages Are Connected:**

The ContentPlaceHolder is the portion of the master page that a content page can change

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

When you create a content page, ASP.NET links your page to the master page by adding an attribute to the Page directive. This attribute, named MasterPageFile, indicates the associated master page.

```
<%@ Page Language = "C#" MasterPageFile = "~/SiteTemplate.master"    AutoEventWireup =
"true" CodeFile = "SimpleContentPage.aspx.cs"    Inherits = "SimpleContentPage" Title =
"Untitled Page" %>
<asp:Content ID = "Content1" ContentPlaceHolderID = "MainContent" runat = "Server">
```

**UNIT-III**

**6a) Explain about data binding with dictionary collection with an example application**. **(6M)**

A dictionary collection is a special kind of collection in which every item (or definition, to use the dictionary analogy) is indexed with a specific key (or dictionary word).

```
protected void Page_Load(Object sender, EventArgs e)
{
   if (!this.IsPostBack)
   {
          Dictionary<int, string> fruit = new Dictionary<int, string>();
          fruit.Add(1, "Kiwi");
          fruit.Add(2, "Pear");
          fruit.Add(3, "Mango");
          fruit.Add(4, "Blueberry");
          fruit.Add(5, "Apricot");
          fruit.Add(6, "Banana");
          fruit.Add(7, "Peach");
          fruit.Add(8, "Plum");
          MyListBox.DataSource = fruit;
          MyListBox.DataTextField = "Value";
          this.DataBind();
  }
}
```

Each item in a dictionary-style collection has both a key and a value associated with it. If you don’t specify which property you want to display, ASP.NET simply calls the ToString() method on each collection item.

**6b) Design a webform to display student details based on department name. (6M)**

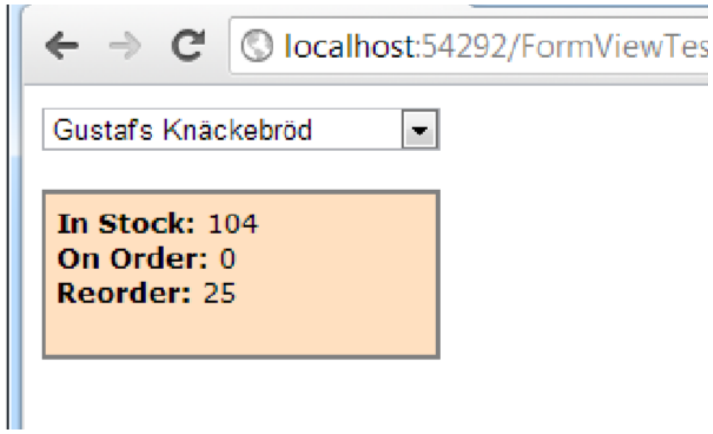
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.SqlClient;

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

```
{
   SqlConnection con;
   SqlCommand cmd;
   SqlDataReader sdr;
   protected void Page_Load(object sender, EventArgs e)
   {
            con = new SqlConnection("data source=.\\sqlexpress;database=master;user
           id=sa;password=sqlserver");
           cmd = new SqlCommand();
           cmd.Connection = con;
           cmd.CommandText = "select * from STUDENTS  where DNAME="'CSE"";
           con.Open();
           sdr = cmd.ExecuteReader();
           GridView1.DataSource = sdr;
           GridView1.DataBind()
           con.Close();
   }
```

**7a) Design an application that retrieves data from a datasource using Formview. (6M)**

**The FormView**

FormView: The FormView shows a single record at a time and supports editing. The difference is that the FormView is based on templates, which allow you to combine fields in a flexible layout that doesn't need to be table based.



**In Stock:** 104
**On Order:** 0
**Reorder:** 25

<asp:SqlDataSource ID = "sourceProducts" runat = "server"  ConnectionString = " < %$ ConnectionStrings:Northwind % > "  SelectCommand = "SELECT ProductID, ProductName FROM Products"> </asp:SqlDataSource>
<asp:DropDownList ID = "lstProducts" runat = "server"  AutoPostBack = "True" DataSourceID = "sourceProducts"  DataTextField = "ProductName" DataValueField = "ProductID" Width = "184px"> </asp:DropDownList>
<asp:FormView ID = "FormView1" runat = "server" DataSourceID = "sourceProducts">
<ItemTemplate>

```
<b > In Stock:</b>
<%# Eval("UnitsInStock") %>
<br />
<b > On Order:</b>
 <%# Eval("UnitsOnOrder") %>
 <br />
 <b > Reorder:</b>
  <%# Eval("ReorderLevel") %>
  <br />  </ItemTemplate>
</asp:FormView>
```

**7b)Discuss about LINQ to Objects with an application. (6M)**

**LINQ to Objects**: This is the simplest form of LINQ. It allows you to query collections of in-memory objects. LINQ  return an usual type of object, called an iterator object. Although the iterator object looks like an ordinary collection to your code, it doesn’t hold any information.LINQ evaluates your expression and quickly grabs the information you need. This trick is called deferred execution

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class LINQTOOBJECTS : System.Web.UI.Page
{
   protected void Page_Load(object sender, EventArgs e)
   {
      List<Employee> employees = new List<Employee>();
      employees.Add(new Employee(1, "Nancy", "Davolio", "Ms."));
      employees.Add(new Employee(2, "Andrew", "Fuller", "Dr."));
      employees.Add(new Employee(3, "Janet", "Leverling", "Ms."));

      var matches = from employee in employees
                 where employee.LastName.StartsWith("D")
                 select employee;
      gridEmployees.DataSource = matches;
      gridEmployees.DataBind();


   }
   public class Employee
```

```
{
   public int EmployeeID { get; set; }
   public string FirstName { get; set; }
   public string LastName { get; set; }
   public string TitleOfCourtesy { get; set; }
   public Employee(int employeeID, string firstName, string lastName,
   string titleOfCourtesy)
   {
      EmployeeID = employeeID;
      FirstName = firstName;
      LastName = lastName;
      TitleOfCourtesy = titleOfCourtesy;
   }
}
```

## UNIT-IV

**8a) Explain in detail about MVC. (6M)**

In high –level terms ,the MVC pattern means that an MVC application will be split into atleast three pieces:

***Models,*** which contain or represent data that user works with. These can be simple view models, which just represent data being transferred between views and controllers; or they can be domain models, which contain the data in a business domain as well as the operations, transformations and rules from manipulating that data.

***Views,*** which are used to render some part of the model as a user interface.

***Controllers,*** which process incoming requests, perform operations on the model, and select views to render to the user .

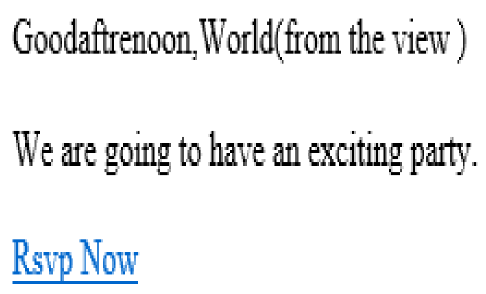
Models are the definition of the universe your application works in. controllers.

Views contain the logic required to display elements of the models to the user and nothing more. They have no direct awareness of the model and do not directly communicate with the model in any way.

Controllers, are the bridge between views and the model requests come in from the client and are serviced by the controller, which selects an appropriate view to show the user and ,if required ,an appropriate operation to perform on the model.

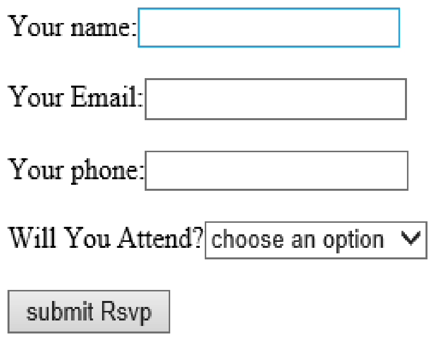
Each piece of the MVC architecture is well defined and self contained—this is referred to as the *separation of concerns.* The logic that manipulates the data in the model is contained only in the model; the logic that displays the data is only in the view, and the code that handles user requests and input is contained only in controller. With a clear division between each of the pieces , your application will be easier to maintain and extend over its lifetime, no matter how large it becomes.

**8b) Create a simple application using MVC architecture. (6M)**


Goodaftrenoon,World(from the view )

We are going to have an exciting party.

Rsvp Now


Your name:

Your Email:

Your phone:

Will You Attend? choose an option

**HOMECONTROLLER.CS**

```
using partyinvites.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace partyinvites.Controllers
{
    public class HomeController : Controller
    {
       public ViewResult Index()
        {
            int hour = DateTime.Now.Hour;
            ViewBag.Greeting = hour < 12 ? "Goodmorning" : "Goodaftrenoon";
            return View();
        }
        [HttpGet]
        public ViewResult Rsvpform()
        {
            return View();
        }
        [HttpPost]
        public ViewResult Rsvpform(GuestResponse guestResponse)
        {
            if(ModelState.IsValid)
            {
                return View("thanks", guestResponse);
            }
            else
            {
                return View();
            }
        }
    }
}
```

**MODELS**

**GUESTRESPONSE.CS**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace PartyInvites.Models
{
    public class GuestResponse
    {
        [Required(ErrorMessage="please enter your naame")]
        public String name { get; set;}
        [Required(ErrorMessage="please enter mail id")]
        [Regular Expression("+||.+||..+") ErrorMessage="please enter valid email id"]
        public String email { get; set; }
        [Required(ErrorMessage = "please enter your number")]
        public String phone { get; set; }
        [Required(ErrorMessage="please specify whether you will attend")]
        public bool willattend { get; set; }
    }
}
```

## INDEX:

```
@{
    Layout = null;
}
<!DOCTYPE html>
<html>
    <head>
        <meta name="viewport" content="width=device-width"/>
        <title>Index</title>
```

```
  </head>
  <body>
    <div>
      @ViewBag.Greeting,World(from the view )
      <p>We are going to have an exciting party.<br/>
      </p>
      @Html.ActionLink("Rsvp Now","RsvpForm")
    </div>
  </body>
</html>
```

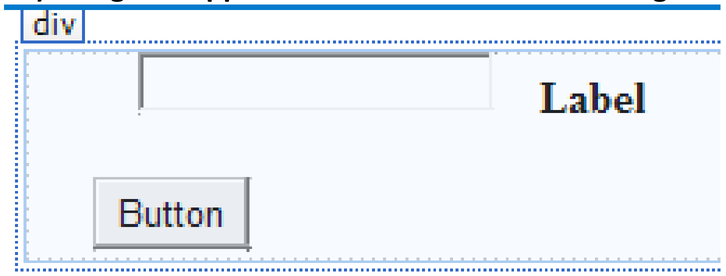
**<u>RSVPFORM</u>**

```
@model partyinvites.Models.GuestResponse
@{
  Layout = null;
}
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width" />
  <title>RsvpForm</title>
</head>
<body>
  @using (Html.BeginForm())
  {
    @Html.ValidationSummary()
    <p>Your name:@Html.TextBoxFor(x => x.name)</p>
    <p>Your Email:@Html.TextBoxFor(x => x.Email)</p>
    <p>Your phone:@Html.TextBoxFor(x => x.phone)</p>
    <p>
      Will You Attend?@Html.DropDownListFor(x => x.WillAttend, new[]
            {
              new SelectListItem(){Text="Yes i'll be there",Value=bool.TrueString},
              new SelectListItem(){Text="no,i can not come",Value=bool.FalseString}
            }, "choose an option")
    </p>
    <input type="submit" value="submit Rsvp" />
    }
</body>
</html>
```

**THANKS:**

```
@model partyinvites.Models.GuestResponse
@{
    Layout = null;

}
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Thanks</title>
</head>
<body>
    <div>
        <h1>Thankyou,@Model.name!</h1>
        @if (Model.WillAttend == true)
        {
            @:Its great that you are coming.The drinks are already in the fridge.
        }
        else
        {
            @:sorry to hear that you can not make,but thanks for letting us know.
        }
    </div>
</body>
</html>
```

**9a) Design an application to demonstrate creating and consuming a WCF web service. (6M)**



**HELLOSERVICE.CS**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace HelloServices
{
   // NOTE: You can use the "Rename" command on the "Refactor" menu to change the class
name "HelloService" in both code and config file together.
   public class HelloService : IHelloService
   {
      public String GetMessage(String name)
      {
         return "hello" + name;
      }
   }
}
```

**<u>APP.CONFIG FILE:</u>**

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
   <startup>
      <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5.2" />
   </startup>
  <system.serviceModel>
   <behaviors>
     <serviceBehaviors>
      <behavior name="mexBehaviour">
        <serviceMetadata httpGetEnabled="true"/>
      </behavior>
     </serviceBehaviors>
   </behaviors>
   <services>
     <service name="HelloService.HelloService" behaviorConfiguration="mexBehaviour">
      <endpoint address="HelloService" binding="basicHttpBinding"
contract="HelloService.IHelloService">
      </endpoint>
      <endpoint address="HelloService" binding="netTcpBinding"
contract="HelloService.IHelloService">
      </endpoint>
      <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataxchange"/>
      <host>
```

```
        <baseAddresses>
          <add baseAddress="http://localhost:8080/"/>
          <add baseAddress="http://localhost:8090/"/>
        </baseAddresses>
      </host>
    </service>
  </services>
 </system.serviceModel>
</configuration>
```

**PROGRAM.CS**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Security;

namespace HelloServiceHost
{
    class Program
    {
        static void Main(string[] args)
        {
            using (ServiceHost host = new ServiceHost(typeof(HelloServiceHost.HelloService))
            {
                host.Open();
            Console.WriteLine("host started @" + DateTime.Now.ToString());
            Console.ReadLine();
            }
        }
}
```

**CLIENT SIDE PROGRAMMMING**

**HELLOSERVICECLIENT.ASPX**

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

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
```

```
</head>
<body>
   <form id="form1" runat="server">
   <div style="font-weight: 700">

       &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
       <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
&nbsp;&nbsp;&nbsp;
       <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
       <br />
       <br />
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
       <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Button" />

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

**<u>HELLOSERVICEHOST.ASPX.CS</u>**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

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

   }
   protected void Button1_Click(object sender, EventArgs e)
   {
          HelloService.HelloServiceClient client = new
          HelloService.HelloServiceClient("BasicHttpBinding_IHelloService");
          Label1.Text = client.GetMessage(TextBox1.Text);
   }
}
```

**9b) Explain deployment of an ASP.NET application with simple example .** **(6M)**

**Deploying with Visual Studio:**

i) You can create a virtual directory when you create a new project
ii) You can use the Copy Web Site feature to transfer an existing website to a virtual directory
iii) You can use the Publish Web Site feature to compile your website and transfer it to another location.
iv) If you use web projects (not projectless websites), you can use the web package feature to bundle IIS settings, security certificates, and SQL scripts, with the actual files of your application, into a single convenient package.

**Prepared By:**

| | |
|---|---|
| **(K. Arun Babu)** | **(V.Chakradhar)** |
| **Assistant Professor** | **HOD** |
| **Dept. of CSE** | **Dept. of CSE** |