The .NET Framework

As you've already learned, the .NET Framework is really a cluster of several technologies:

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.

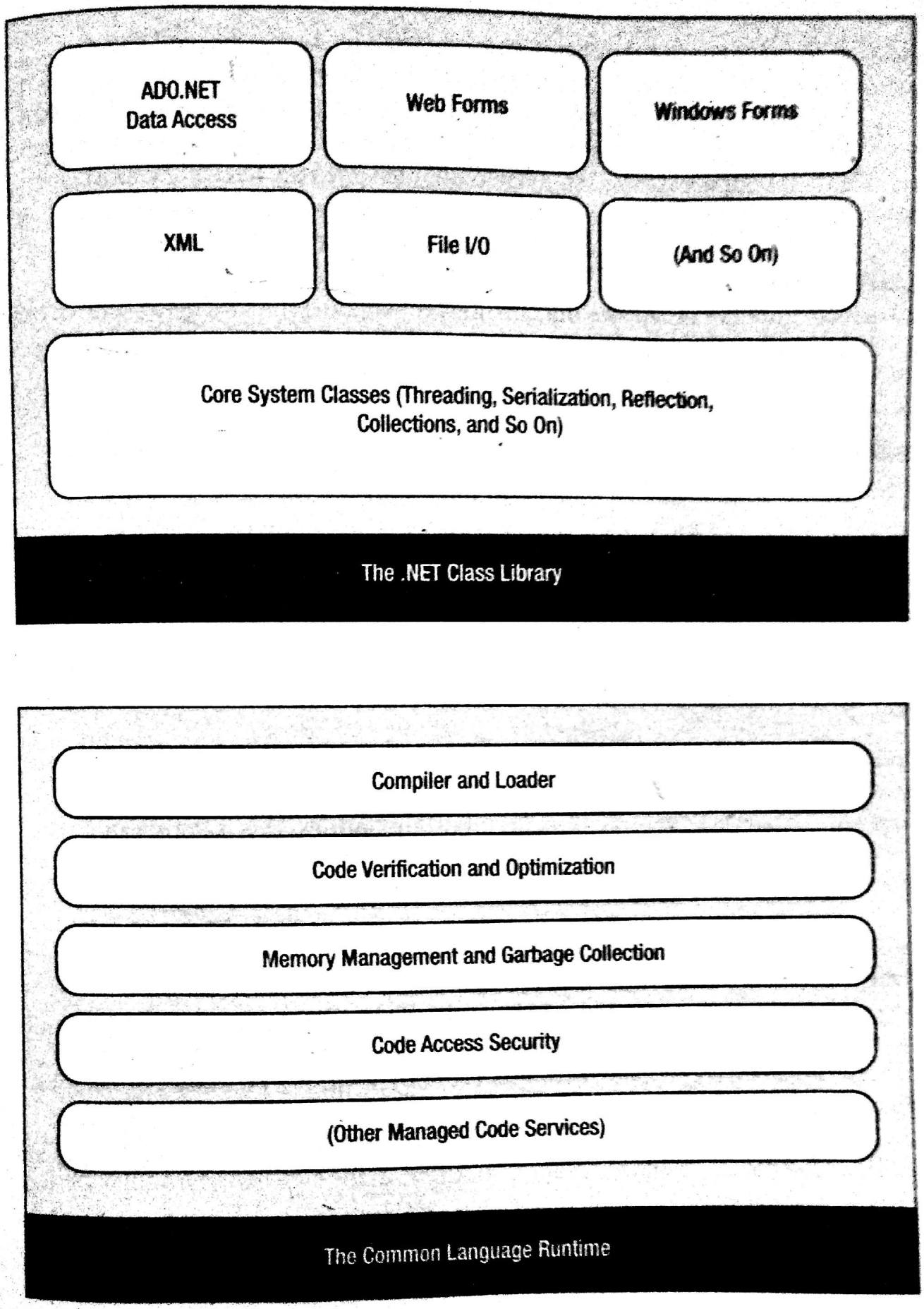


Figure 1-4. The .NET Framework

Lab1

Table Controls

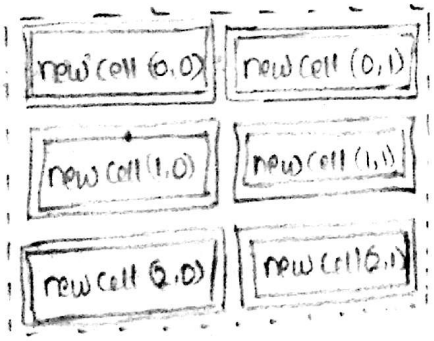
Enter no. of Rows: 3 Enter no. of columns: 2

☑ click to apply Border

click

###

Output :



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

namespace WebApplications
{
    public partial class webform1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }
        public void Button1_click(object sender, EventArgs e)
        {
            int r = int.Parse(TextBox1.Text);
            int c = int.Parse(TextBox2.Text);
            for(int row=0; row<r; row++)
            {
                TableRow tr = new TableRow();
                Table1.BorderStyle = BorderStyle.Dashed;
                Table1.Controls.Add(tr);
                for(int col=0; col<c; col++)
                {
                    TableCell tc = new TableCell();
                    tc.Text = "new Cell(" + row.ToString() + "," +
                              col.ToString() + ")";
                    tr.Controls.Add(tc);
                    if(CheckBox1.Checked)
                    {
                        tc.BorderStyle = BorderStyle.Double;
                    }
                }
            }
        }
    }
}
```

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
                                    Inherits="WebApplication5.WebForm1" %>

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        Table Controlls <br/> <br/>
        Enter No.of rows : <asp:TextBox id="TextBox1" runat="server">
            </asp:TextBox>
                     
   <br/> <br/>
    <asp:CheckBox id="CheckBox1" runat="server" text="Click to apply
                                                    Border" />

    <br/>
    <asp:Button id="Button1" runat="server" onclick="Button1_click"
                                        Text="Click" />

    <br/> <br/>
    <asp:Table id="Table1" runat="server">
    </asp:Table>
  </div>
  </form>
</body>
</html>
```

(2)

Table 6-2. *WebControl Properties* Any 5

Property	Description
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.

write an asp.net application program to demonstrate Registration form

webform1.aspx Design

ENTER NAME : Tejaswi

ENTER REG.NO : 14

ENTER GENDER :

⊙ FEMALE

○ MALE

ENTER BRANCH : CSE / ECE / EEE / CIVIL

ENTER DATE OF BIRTH :

< 2/1/1998 >

SU	MO	TU	WE	TH	FR	SA
						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	31					

ENTER HOBBIES :

☐ Playing

☑ Reading

☑ singing

Submit Reset

Name : Tejaswi

reg.no : 14

Gender : FEMALE

Branch : CSE

Date of Birth : 22/1/1998

hobbies : Reading, singing

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace WebApplication4
{
    Public partial class WebForm1 : System.Web.UI.Page
    {
        Protected void Page_Load(object sender, EventArgs e)
        {
        }
        Protected void Button1_Click(object sender, EventArgs e)
        {
            String a = TextBox1.Text;
            String r = TextBox2.Text;
            String c = RadioButtonList1.selectedvalue;
            String d = listBox1.SelectedValue;
            String f = Calendar1.SelectedDate.ToString();
            String g = " ";
            foreach(ListItem li in CheckBoxList1.Items)
            {
                if(li.selected)
                {
                    g+ = li.Text + ",";
                }
            }
            Res.InnerText + = "name:" + a + "<br/> roll no:" + r +
        "<br/> branch :" + d + "<br/> Date of birth :" + f + "<br/> hobbies" +
                                                                    g;
        }
        Protected void Button2_Click(object sender, EventArgs e)
        {
            TextBox1.Text = " ";
            TextBox2.Text = " ";
            RadioButtonList1.selectedvalue = " ";
            ListBox1.Selectedvalue = " ";
        }
    }
}
```

Any 5 (or) 6

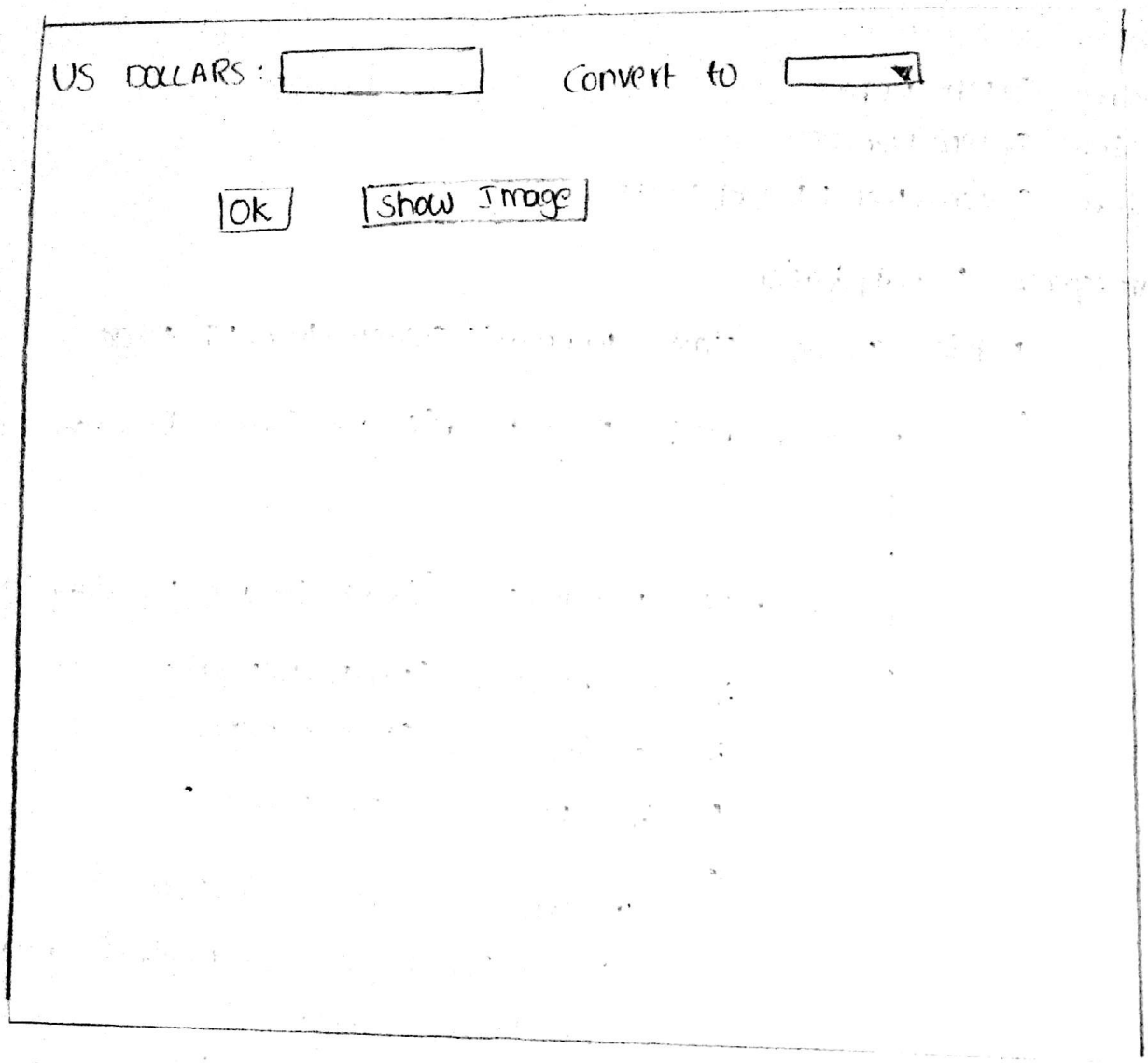
(3)

Table 5-3. The HTML Server Control Classes

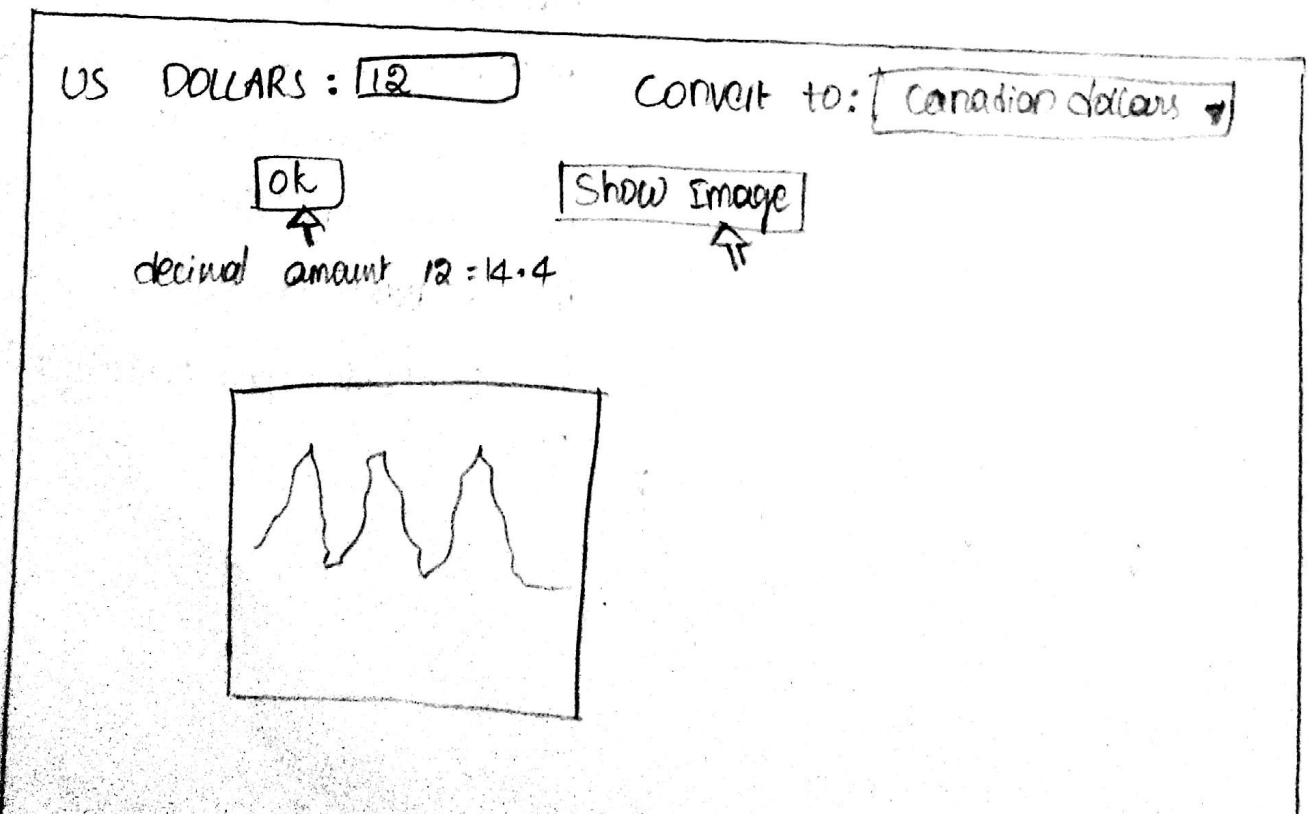
Class Name	HTML Element	Description
HtmlForm	<form>	Wraps all the controls on a web page. All ASP.NET server controls must be placed inside an HtmlForm control so that they can send their data to the server when the page is submitted. Visual Studio adds the <form> element to all new pages. When designing a web page, you need to ensure that every other control you add is placed inside the <form> section.
HtmlAnchor	<a>	A hyperlink that the user clicks to jump to another page.
HtmlImage	<img>	A link that points to an image, which will be inserted into the web page at the current location.
HtmlTable, HtmlTableRow, and HtmlTableCell	<table>, <tr>, <th>, and <td>	A table that displays multiple rows and columns of static text.
HtmlInputButton, HtmlInputSubmit, and HtmlInputReset	<input type="button">, <input type="submit">, and <input type="reset">	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	<button>	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	<input type="checkbox">	A check box that the user can select or clear. Doesn't include any text of its own.
HtmlInputRadioButton	<input type="radio">	A radio button that can be selected in a group. Doesn't include any text of its own.
HtmlInputText and HtmlInputPassword	<input type="text"> and <input type="password">	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	<textarea>	A large text box for typing multiple lines of text.
HtmlInputImage	<input type="image">	Similar to the <img> tag, but inserts a "clickable" image that submits the page. Using server-side code, you can determine exactly where the user clicked in the image—a technique you'll consider later in this chapter.
HtmlInputFile	<input type="file">	A Browse button and text box that can be used to upload a file to your web server, as described in Chapter 17.

(continued)

write an asp.net application program to demonstrate Currency Converter



output



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

namespace WebApplications
{
    Public partial class Webform1 : System.Web.UI.Page
    {
        Protected void Page_Load(object Sender, EventArgs e)
        {
            if(this.IsPostBack == false)
            {
                C.Items.Add(new ListItem("Euros", "0.85"));
                C.Items.Add(new ListItem("Japanese Yen", "110.33"));
                C.Items.Add(new ListItem("Canadian Dollar", "1.27"));
            }
        }
        Protected void Button1_Click(Object Sender, EventArgs e)
        {
            decimal A;
            bool success = Decimal.TryParse(TextBox1.Text, out A);
            if (success)
            {
                ListItem item = C.Items[C.SelectedIndex];
                decimal newi = A * Decimal.Parse(item.Value);
                P.InnerHtml = "decimal Amount" + A.ToString() +
                              newi.ToString();
            }
            else
            {
                P.InnerHtml = "You have entered invalid output";
            }
        }
        Protected void Button2_Click(Object Sender, EventArgs e)
        {
            Graph.Src = "Pic" + C.SelectedIndex.ToString() + ".jpg";
            Graph.Visible = true;
        }
    }
}
```

List Controls

The list controls include the ListBox, DropDownList, CheckBoxList, RadioButtonList, and BulletedList. They all work in essentially the same way but are rendered differently in the browser. The ListBox, for example, is a rectangular list that displays several entries, while the DropDownList shows only the selected item. The CheckBoxList and RadioButtonList are similar to the ListBox, but every item is rendered as a check box or option button, respectively. Finally, the BulletedList is the odd one out—it's the only list control that isn't selectable. Instead, it renders itself as a sequence of numbered or bulleted items.

All the selectable list controls provide a SelectedIndex property that indicates the selected row as a zero-based index (just like the HtmlSelect control you used in the previous chapter). For example, if the first item in the list is selected, the SelectedIndex will be 0. Selectable list controls also provide an additional SelectedItem property, which allows your code to retrieve the ListItem object that represents the selected item. The ListItem object provides three important properties: Text (the displayed content), Value (the hidden value from the HTML markup), and Selected (true or false depending on whether the item is selected).

In the previous chapter, you used code like this to retrieve the selected ListItem object from an HtmlSelect control called Currency, as follows:

```
ListItem item;
item = Currency.Items[Currency.SelectedIndex];
```

If you used the ListBox web control, you can simplify this code with a clearer syntax:

```
ListItem item;
item = Currency.SelectedItem;
```

Design Write an asp.net application program to demonstrate List Controls.

LIST CONTROLS

Gender:

o female

o male

Branch: [cse ▼]

Section: [A / B / C]

hobbies:

☐ reading

☐ playing

☐ singing

Date of Birth: [ ]

<	JULY 2018					>
SU	MO	TU	WE	TH	FR	SAT
						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	31					

• unbound

[submit]

list.aspx.cs

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

namespace WebApplications
{
    public partial class list : System.Web.UI.Page
    {
        Protected void Page_Load(Object Sender, EventArgs e)
        {
        }
        Protected void Button1_Click(Object Sender, EventArgs e)
        {
            String gen = RadioButtonList1.Text;
            String br = DropDownList1.SelectedValue;
            String sec = ListBox1.SelectedValue;
            String hob = " ";
            foreach(ListItem li in CheckBoxList1.Items)
            {
                if(li.selected)
                {
                    hob += li.Text + ",";
                }
            }
            TextBox1.Text = Calendar1.SelectedDate.ToString();
            BulletedList1.Items.Add("Gender:"+gen);
            BulletedList1.Items.Add("Branch:"+br);
            BulletedList1.Items.Add("Section:"+sec);
            BulletedList1.Items.Add("Hobbies:"+hob);
            BulletedList1.Items.Add("DOB"+TextBox1.Text);
        }
    }
}
```

(6)

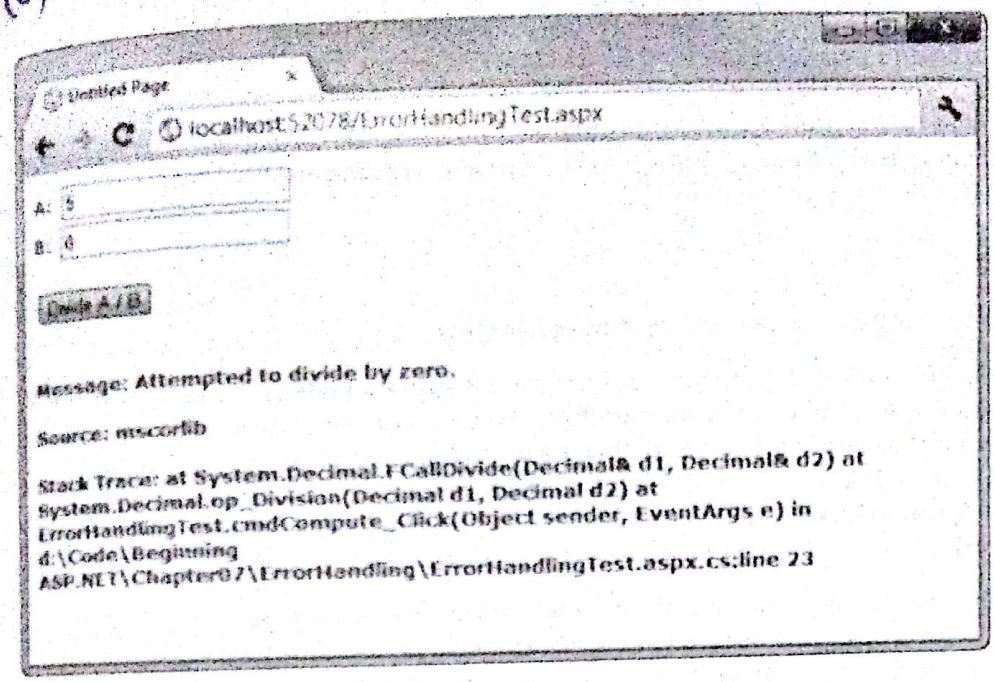


Figure 7-4. Catching and displaying exception information

Obviously, you can easily prevent this exception from occurring by using extra code-safety checks, or you can elegantly resolve it by using the validation controls. However, this code provides a good example of how you can deal with the properties of an exception object. It also gives you a good idea about the sort of information that will be returned.

Here's the page class code for this example:

```
public partial class ErrorHandlingTest : System.Web.UI.Page
{
    protected void cmdCompute_Click(Object sender, EventArgs e)
    {
        try
        {
            decimal a, b, result;
            a = Decimal.Parse(txtA.Text);
            b = Decimal.Parse(txtB.Text);
            result = a / b;
            lblResult.Text = result.ToString();
            lblResult.ForeColor = System.Drawing.Color.Black;
        }
        catch (Exception err)
        {
            lblResult.Text = "<b>Message:</b> " + err.Message;
            lblResult.Text += "<br /><br />";
            lblResult.Text += "<b>Source:</b> " + err.Source;
            lblResult.Text += "<br /><br />";
            lblResult.Text += "<b>Stack Trace:</b> " + err.StackTrace;
            lblResult.ForeColor = System.Drawing.Color.Red;
        }
    }
}
```

Write an asp.net application prgm to demonstrate Greeting card Generator

Design:

webform1.aspx.

Greeting card Generator

Choose a background color: Pink

Choose a font color: Black

Choose a font size: 19

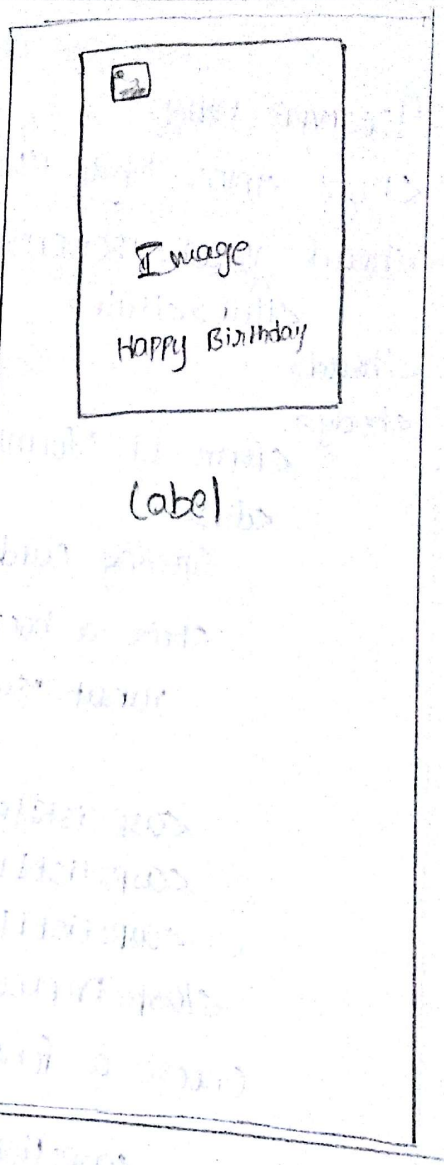
Choose Border Style:

- Unbound
- None
- Solid
- Double

☐ click to display image

Write the greeting text: Happy Birthday

Button



webform1.aspx.cs

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

namespace WebApplications
{
    Public partial class webform1: System.Web.UI.Page
    {
        Protected void Page_Load (object Sender, EventArgs e)
        {
            if(IsPostBack)
            {
                ListItem li = new ListItem();
                li.Text = Borderstyle.None.ToString();
                li.Value =((int)Borderstyle.None).ToString();
                RadioButtonList1.Items.Add(li);

                ListItem lil = new ListItem();
                lil.Text = Borderstyle.Solid.ToString();
                lil.Value = ((int) Borderstyle.solid).ToString();
                RadioButtonList1.Items.Add(lil);

                ListItem li2 = new ListItem();
                li2.Text = Borderstyle.Double.ToString();
                li2.Value = ((int) Borderstyle.Double).ToString();
                RadioButtonList1.Items.Add(li2);

                RadioButtonList1.SelectedIndex = 0;
            }
        }
        Proted void Button1_click (Object sender, EventArgs e)
        {
            Panel1.Backcolor = Color.FromName (DropDownList1.
                                SelectedItem.Text);
            Label2.Font.Name = DropDownList2.selectedItem.Text;
```

```
if( Int32.Parse(TextBox1.Text)>0)
{ Label2.Font.Size = FontUnit.Point(Int32.Parse(TextBox1.
                                    Text));
}
int b = Int32.Parse(RadioButtonList1.SelectedValue);
Panel1.BorderStyle = (BorderStyle)b;
image1.ImageUrl = "1.jpg";
if(checkBox1.Checked)
{     image1.Visible = true;
}
else
{     image1.Visible = false;
}
Label2.Text = TextBox2.Text;
}
}
}
```