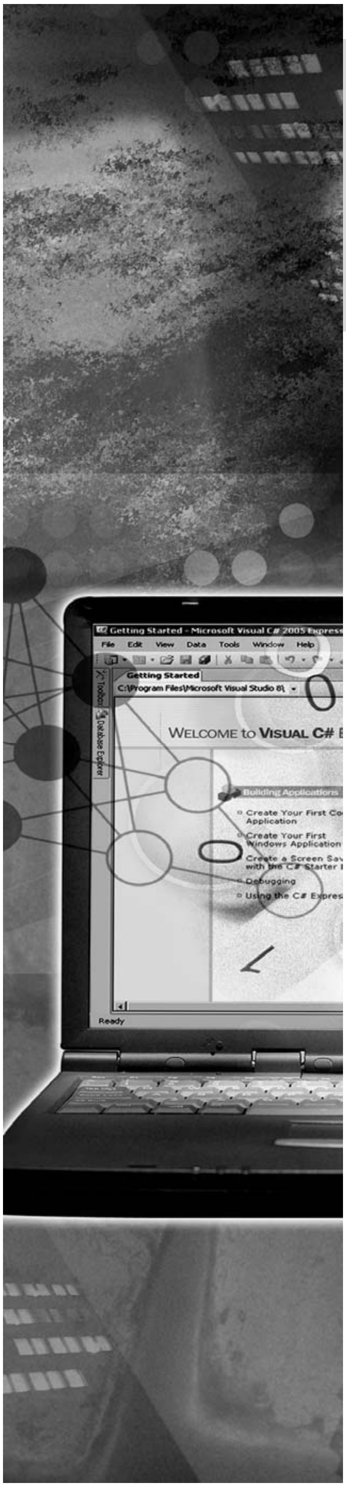


Chapter 4

Performing Validation

Data entered by a user, in any application, should be valid and properly formatted. Otherwise, a lot of resources could be wasted in fixing problems that could arise due to incorrect data formats. Data should be checked for pre determined formats or other defined input criteria as soon as a user provides it.

This creates the need for validation, which is used to check whether data entered by a user is according to the predetermined format and input criteria.

This chapter explains how to validate user input by using various controls and validating events and also displaying error messages to the user.

Objectives

In this chapter, you will learn to:

- Validate user input using controls and events

Introducing Validation

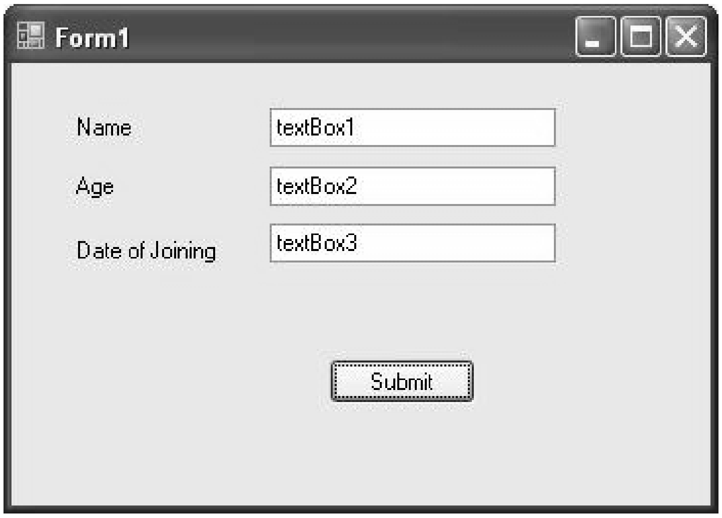
An interactive application needs to accept data entered by a user. To enable an application to accept valid data, the data entered by the user should be validated.

Validation is a means to check the accuracy and correctness of the data entered by a user in an application. Validation can be performed for various checks such as checking the date entry for the correct format, checking a number field for proper range, and checking a string field for appropriate length.

By performing validation, you can prompt the user to enter correct data and therefore, reduce errors. Otherwise a lot of resources would be wasted in fixing errors.

Validation of User Inputs

To validate data in a control, the Validating event of that control can be used. The Validating event enables you to prevent the user from shifting focus from the control being validated to another control on the form until validation has been completed. For example in the following Data Entry Form, the user cannot move to any other control within the form until the validation rules for the current control are satisfied.



The Data Entry Form

For the preceding requirement, you need to perform validations, as shown in the following table.

Control	***Validation***	***How to perform***
textBox1	*Name text box cannot be left blank*	*To check whether the Name text box is blank or not, we will check the length of the typed string. If this is zero the text box is blank otherwise it is not.* *To check the length of the text entered in the Name text box use the following code:* `private void textBox1_Validating(object sender, CancelEventArgs e) { if (textBox1.Text.Length == 0) { // Prompt the user to enter his name }`
textBox2	*Age cannot be less than 35*	*To check the age entered in the Age text box use the following code:* `private void textBox2_Validating(object sender, CancelEventArgs e) { if (Convert.ToInt32(textBox2.Text) < 35) { // Prompt the user to enter age greater than 35 }`
textBox3	*Date of Joining cannot be later than the current date*	*To check the date entered in the Date of Joining text box use the following code:* `private void textBox3_Validating(object sender, CancelEventArgs e) { if (Convert.ToDateTime(textBox3.Text) > DateTime.Now) { // Prompt`

Control	*Validation*	*How to perform*
		`the user to enter date prior to the current date }`

Code in the Validating Event of the Controls

Note

`Convert.ToInt32` function is used to convert `char` to `int` data type. Similarly, `Convert.ToDateTime` function is used to convert `char` to `DateTime` data type.

Notice that the validating event requires two arguments. The arguments are:

- **`sender`**: Provides a reference of the object that has raised the event.
- **`e`**: Is used to cancel the validating event.

After you have performed validation, the user needs to be prompted by an appropriate message. This message displays that the user has to enter the requisite data. This message can be displayed by using the `MessageBox` class, ErrorProvider control, or StatusStrip control.

Just a minute:

Name the event that enables the user to prevent shifting of focus from a control until all the validation rules have been met.

Answer:

The Validating event

The MessageBox Class

The `MessageBox` class is used to display informative messages to users. To display a message box, you need to call the `Show()` method of the `MessageBox` class. The `Show()` method is a static method that accepts the title name, message, buttons, and icons as arguments.

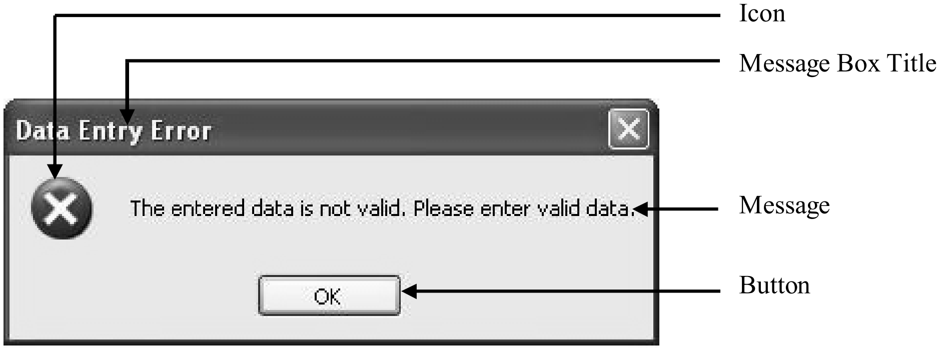
Note

You cannot create an instance of the MessageBox class by using the new keyword.

The following code snippet invokes the message box by using the `show()` method of the `MessageBox` class:

```
private void button1_Click(object sender, EventArgs e)
{
MessageBox.Show("The entered data is not valid. Please enter valid
data.", "Data Entry Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
}
```

The following figure shows the message box that will appear after you execute the code snippet.



The Data Entry Error Message Box

In the preceding code snippet the first parameter, "`The entered data is not valid. Please enter valid data.`", is displayed in the message box. The second parameter, "`Data Entry Error`", is the title of the message box. The third parameter, `MessageBoxButtons.OK,` specifies the number and type of buttons to be displayed in the message box.

There are various options that can be used for the third parameter. The following table describes these options.

Option	***Description***
AbortRetryIgnore	*Contains Abort, Retry, and Ignore buttons*
OK	*Contains an OK button.*
OKCancel	*Contains OK and Cancel buttons.*
RetryCancel	*Contains Retry and Cancel buttons.*
YesNo	*Contains Yes and No buttons.*
YesNoCancel	*Contains Yes, No, and Cancel buttons.*

The Button Options in a Message Box

The fourth parameter, `MessageBoxIcon.Error`, specifies the icon to be displayed in the message box. There are various options that can be used for the fourth parameter. The following table describes these options.

Option	***Description***
Asterisk	*Contains a symbol consisting of the letter 'I' within a circle.*
Error	*Contains a symbol consisting of 'X' written in white in a circle with a red background.*
Exclamation	*Contains a symbol consisting of an exclamation mark point in a triangle with a yellow background.*
Hand	*Contains a symbol consisting of a white' X' in a circle with a red background.*
Information	*Contains a symbol consisting of a lowercase letter 'i' in a circle.*
None	*Contains no symbols.*
Question	*Contains a symbol consisting of a question mark in a circle.*
Stop	*Contains a symbol consisting of white X in a circle with a red background.*
Warning	*Contains a symbol consisting of an exclamation point in a triangle with a yellow background.*

The Icon Options in a Message Box

Just a minute:

How many parameters are there in the show method of the MessageBox class?

1. *One*
2. *Two*
3. *Three*
4. *Four*

Answer:

4. Four

Just a minute:

Which method of the MessageBox class is used to display a message in the message box?

Answer:

The `Show()` method

The StatusStrip Control

The StatusStrip control is derived from `System.Windows.Forms.ToolStrip` class. This is a container control that can hold other controls called child controls. The StatusStrip control is used to display information about the objects that are on the current form. This control is also used to provide feedback on the progress of any operation being performed by the form. By default, a StatusStrip control docks at the bottom of the form.

Some of the properties of the StatusStrip control are described in the following table.

Properties	***Description***
Items	*Is a collection of items to be displayed on the StatusStrip control.*
LayoutStyle	*Specifies the layout orientation of the StatusStrip control.*
Dock	*Defines which borders of the control are bound to the container.*

Some Properties of the StatusStrip Control

A StatusStrip control contains four controls known as child controls. They can be placed on the StatusStrip control as and when required. They cannot be used independently. They have properties, methods, and events that are similar to the parent controls.

The child controls of the StatusStrip control are:

- StatusLabel
- ProgressBar
- DropDownButton
- SplitButton

The StatusLabel Control

This control is used to display the status information. It is also used to prompt the user for a valid entry. This control has various properties, such as the Text property and the Spring property. The Text property is used to change the information displayed on the StatusLabel control. The Spring property is used to fill any remaining space on the StatusStrip control with the StatusLabel control.

The following example shows the use of the Text and Spring properties of the StatusLabel control:

```
StatusLabel1.Spring = true;
StatusLabel1.Text = “Successfully Completed”;
```

Some of the properties of the StatusLabel control are described in the following table.

Properties	***Functions***
Text	*Gets and sets the text to be displayed on the control.*
Spring	*Specifies whether the item fills up the remaining space.*
TextAlign	*Specifies the alignment of the text that will be displayed on the item*

Some Properties of the StatusLabel Control

The ProgressBar Control

This control represents the Windows progress bar control. It is used to show the completion status of any task. It is used when the progress of the task performed by the application needs to be shown.

Some of the properties of the ProgressBar control are described in the following table.

Properties	***Functions***
Minimum	*Specifies the lower bound of the range the ProgressBar is working with.*
Maximum	*Specifies the upper bound of the range the ProgressBar is working with.*
Value	*Gets or sets the current value of the ProgressBar, in the range specified by the minimum and maximum properties.*
Step	*Specifies the amount to increment the current value of the control by when the PerformStep() method is called.*

Some Properties of the ProgressBar Control

The DropDownButton Control

This control displays the ToolStripDropDown from which a user can select a single item from many items. It is generally used when the items to be displayed on the StatusStrip control cannot be accommodated on it.

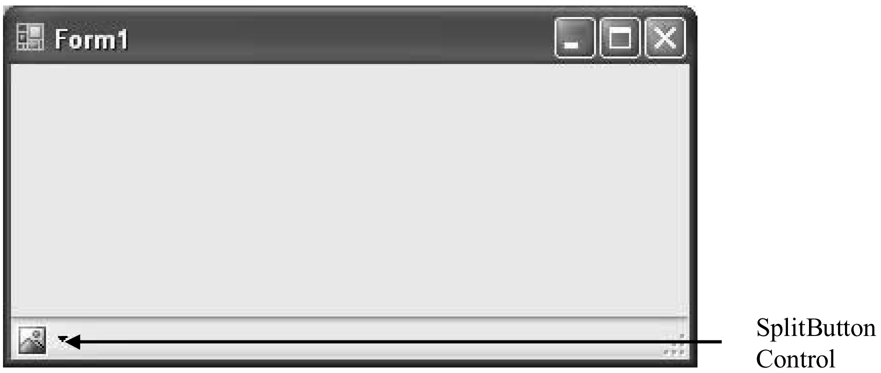
Some of the properties of the DropDownButton control are described in the following table.

Properties	***Functions***
DisplayStyle	*Specifies whether the image and text are rendered.*
DoubleClikEnabled	*Specifies whether the double click event will occur.*
DropDownItems	*Specifies a ToolStripItem to display when the item is clicked.*

Some Properties of the DropDownBotton Control

The SplitButton Control

This control is a combination of a standard button on the left and a drop-down button on the right, as shown in the following figure.



SplitButton Control on a StatusStrip Control

It is used when the functionalities of both, the standard button and the drop-down button, is required.

Some of the properties of the SplitButton control are described in the following table.

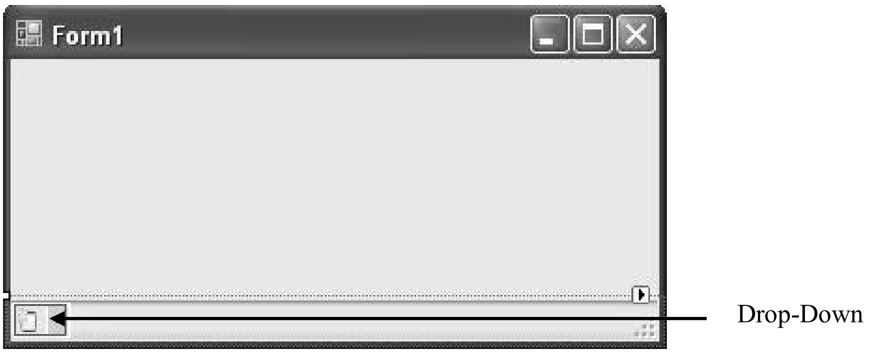
Properties	***Functions***
DisplayStyle	*Specifies whether the image and text are rendered.*
DoubleClikEnabled	*Specifies whether the double click event will occur.*
Padding	*Specifies the internal spacing within this item.*

Some Properties of the SplitButton Control

Add and Configure Controls on a StatusStrip Control

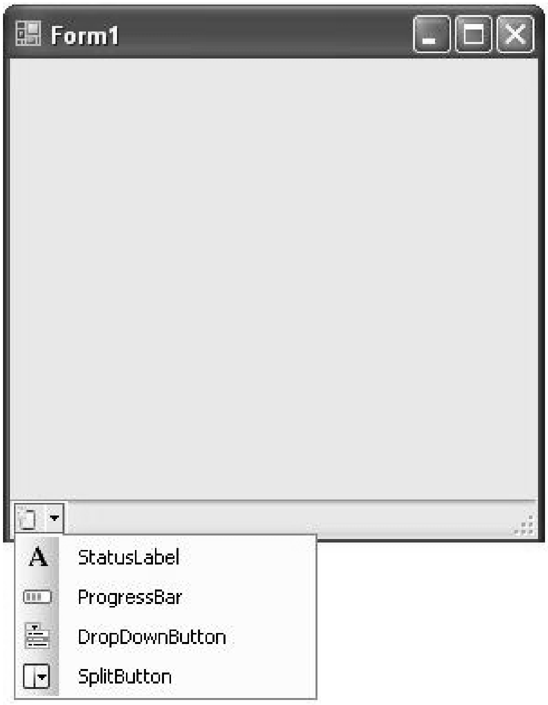
To add a child control to a **StatusStrip** control, you need to perform the following steps:

1. Select the **StatusStrip** control on the form, as shown in the following figure.



StatusStrip Control after Selection

2. Click the drop-down and select the required child control.



StatusStrip Control with Drop-Down List

To configure a control in a **StatusStrip** control you need to perform the following steps:

1. Select the control on the **StatusStrip** control.
2. Use the **Properties** window to set the properties such as **Text** property.

Just a minute:

Which of the following position is the default docking position of the StatusStrip control on the form?

1. *Left*
2. *Right*
3. *Top*
4. *Bottom*

Answer:

4. Bottom

Just a minute:

Which property of the ProgressBar control specifies the amount to increment the current value of the control?

1. *Minimum*
2. *Maximum*
3. *Value*
4. *Step*

Answer:

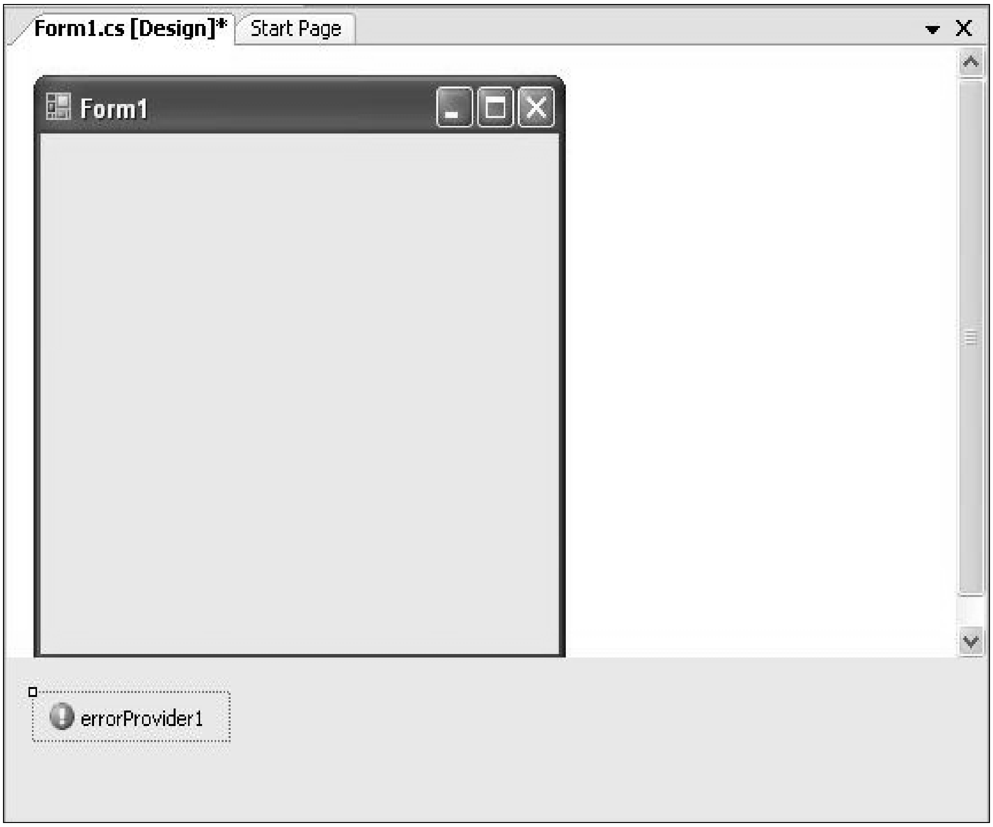
4. Step

The ErrorProvider Control

The ErrorProvider control can be used to display an error message, whenever the user positions the mouse pointer over the error icon. This error icon is displayed next to the control in which the user has entered invalid data.

Adding an ErrorProvider Control

You can add the ErrorProvider control to the form. To add the ErrorProvider control on the form, you can drag the ErrorProvider control onto a form, say Form1. This control appears as a red exclamation mark displayed in the component tray, as displayed in the following figure.



The Form with ErrorProvider Control in the Component Tray

Note

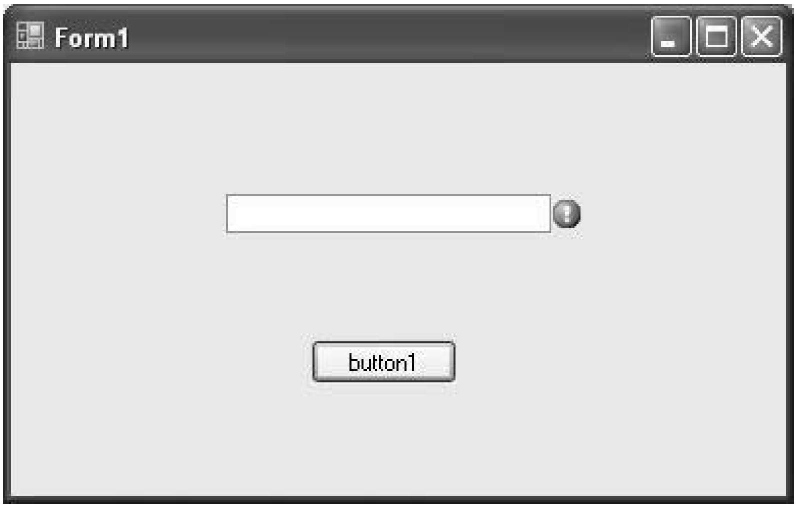
The ContainerConrol property of the ErrorProvider control automatically sets to the form to which it is added. If the ErrorProvider control is added dynamically then this property needs to be set manually.

Configuring an ErrorProvider Control

When an ErrorProvider control is added to a form, a new property, `Error on errorProviderControlName` gets added to all the controls on the form. Here, errorProviderControlName is the name of the ErrorProvider control.

Set the `text` in this property to the error message that you want to display. When this property is set, a red coloured icon is displayed next to the control and a text message is displayed when the user places the mouse pointer over the icon.

An ErrorProvider control is shown in the following figure.



The Error Icon and the Error Message

Generally, the error message should be displayed when an error is identified in code in an application. To display the error message, you can use the `SetError()` method of the ErrorProvider control. The `SetError()` method requires two parameters, the control and the message to be displayed, as shown in the following example:

```
errorProvider1.SetError(NameTextBox, "Please enter your Phone
number.");
```

To cancel the display of the error icon the `clear()` method of the ErrorProvider control is used, as shown in the following code snippet:

```
errorProvider1.Clear();
```

You can also change the blinking of the icon by setting its `BlinkStyle` property, as shown in the following code snippet:

```
errorProvider1.BlinkStyle = ErrorBlinkStyle.NeverBlink;
```

The following table describes the functions of some of the properties of the ErrorProvider control.

Properties	***Functions***
BlinkStyle	*Controls whether the error icon blinks when an error is set.*
BlinkRate	*Indicates the rate in milliseconds at which the error icon blinks.*
ContainerControl	*Specifies the parent control that contains the controls on which the Errorprovider can display error icons.*

Some Properties of the ErrorProvider Control

The following table describes the functions of some of the methods of the ErrorProvider control.

Methods	***Functions***
Clear	*Clears all errors associated with the ErrorProvider component.*
Dispose	*Releases all resources used by the ErrorProvider component.*
GetError	*Returns the current error description string for the specified control.*
SetError	*Sets the error description string for the specified control.*

Some Methods of the ErrorProvider Control

Just a minute:

Which property of the ErrorProvider control automatically sets to the form to which it is added?

1. *SetError*
2. *ContainerControl*
3. *Error on errorProviderControlName*
4. *BlinkStyle*

Answer:

2. ContainerControl

Activity: Validating Data

Problem Statement

SigmaCom is a telecommunication company which is developing software that would gather information about its employees. The company requires that the information entered should be correct and all the fields should be filled with required data and that no fields are left blank.

Help the company to develop the required application.

Solution

To meet the preceding requirement, you need to perform the following tasks:

1. Create a new VC# application.
2. Design the Employee Information form.
3. Add code to perform the desired tasks.
4. Execute the application and verify the output.

Task 1: Creating a New VC# Application

To create a new VC# application, you need to perform the following steps:

1. Select **Start→All Programs→Microsoft Visual Studio 2005→Microsoft Visual Studio 2005**. The **Start Page - Microsoft Visual Studio** window is displayed.
2. Select **File→New→Project**. The **New Project** dialog box is displayed.
3. Select the project type as **Visual C#** from the **Project types** pane and **Windows Application** from the **Templates** pane.
4. Type the name of the new project as **DataValidation** in the **Name** text box.
5. Specify the location where the new project is to be created as **c:\chapter4\Activity1** in the **Location** combo box.
6. Click the **OK** button.

Task 2: Designing the Employee Information Form

When you create a VC# project, by default, the **Form1** form, is added to the project. To add controls to the **Form1** form, you need to perform the following steps:

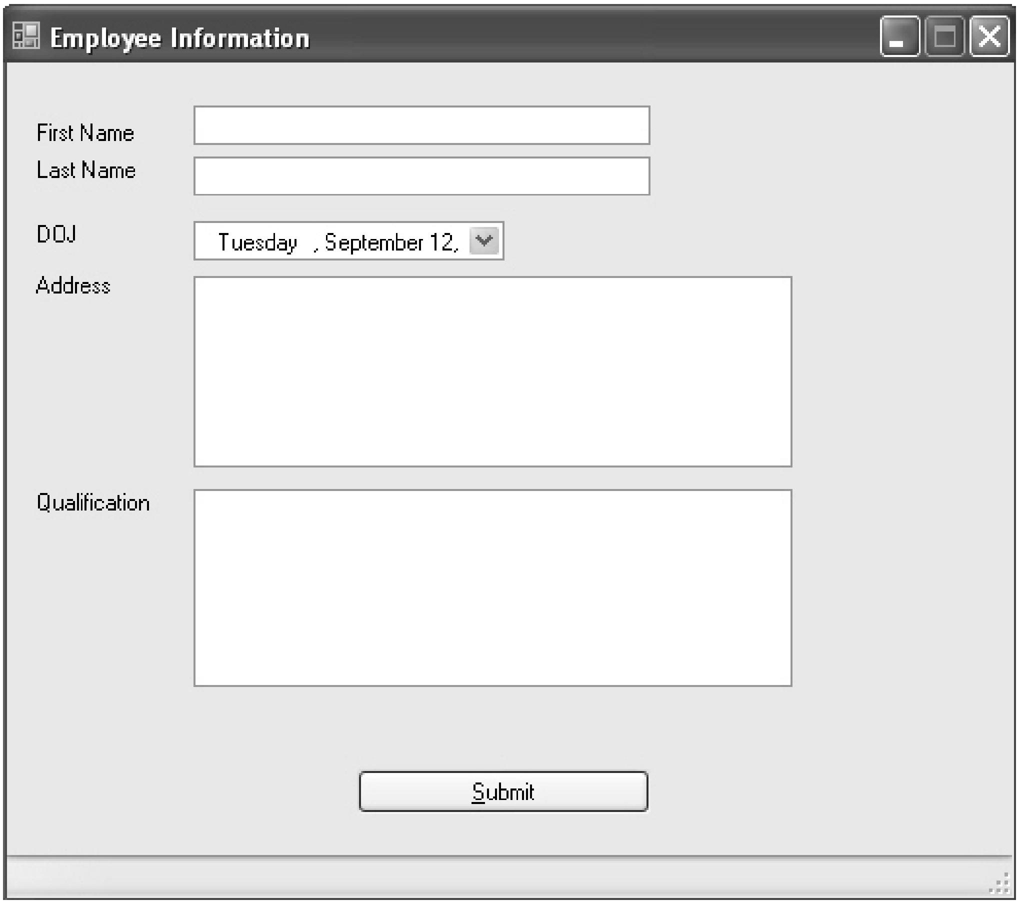
1. Drag five **Label** controls, four **TextBox** controls, a **DateTimePicker** control, a **Button** control, a **StatusStrip** control, a **StatusLabel** control on the **StatusStrip** control, and an **ErrorProvider** control to the form from the **Toolbox** window.
2. Right-click the control and then select **Properties** from the shortcut menu to open the **Properties** window for each control added in Step 1.
3. Set the properties of the form and controls as listed in the following table.

Controls	***Properties***	***Values***
Form1	*Name*	*frmEmployee*
	MaximizeBox	*False*
	Text	*Employee Information*
Label1	*Name*	*lblFName*
	Text	*First Name*
Label2	*Name*	*lblLName*
	Text	*Last Name*
Label3	*Name*	*lblDOJ*
	Text	*DOJ*
Label4	*Name*	*lblAddress*
	Text	*Address*
Label5	*Name*	*lblQuali*
	Text	*Qualification*
TextBox1	*Name*	*txtFName*
TextBox2	*Name*	*txtLName*
TextBox3	*Name*	*txtAddress*
	Multiline	*True*
TextBox4	*Name*	*txtQuali*
	Multiline	*True*
DateTimePicker1	*Name*	*dtpDOJ*
Button1	*Name*	*btnSubmit*
	Text	*&Submit*
StatusStrip1	*Name*	*statusStrip1*
ErrorProvider1	*Name*	*errorProvider1*
StatusLabel1	*Name*	*tssWarning*

Controls	***Properties***	***Values***
	Text	

Properties and Values of the Form's Controls

The Employee Information form should resemble the following figure.



The Employee Information Form

The **ErrorProvider** control is also added to the form but it is not visible because it is used by setting its **SetError** property at run time.

Task 3: Adding Code to Perform the Desired Tasks

To write code in the **Enter** event of the textbox controls and the **DateTimePicker** control, you need to perform the following steps:

1. Double-click the **Form1.cs** file in the **Solution Explorer** window.
2. Select the textbox controls and the **DateTimePicker** control.
3. Press the **F4** key or right-click the control and select **Properties**. This will open the **Properties** window.
4. Click the ⚡ button in the **Properties** window.
5. Locate the **Enter** event and double-click it.

   Code in the **Enter** event of the textbox controls and the **DateTimePicker** control is as follows:

```
//The following code is written to display a message in the
StatusStrip //control when the focus enters into the control.

        private void txtFName_Enter(object sender, EventArgs e)
        {
            tssWarning.Text = "Please Enter Your First Name";
        }

        private void txtLName_Enter(object sender, EventArgs e)
        {
            tssWarning.Text = "Please Enter Your Last Name";
        }

        private void txtQuali_Enter(object sender, EventArgs e)
        {
            tssWarning.Text = "Please Enter Your Qualification";
        }

        private void dtpDOJ_Enter(object sender, EventArgs e)
        {
            tssWarning.Text = "Please Enter Your DOJ. It Must Be
            Earlier Than The Present Date";
        }

        private void txtAddress_Enter(object sender, EventArgs e)
        {
            tssWarning.Text = "Please Enter Your Address";
        }
```

Note

The Enter event of the textbox control occurs when the focus is on the text box. In this event, code for providing the user with an appropriate prompt using the StatusStrip control is written.

6. Declare the **ValidateData** function in the main class.

   Code in the **ValidateData** function is as follows:

```
//This function checks whether data entry in the controls is correct
private int ValidateData()
        {
            int flag=0;
            if (txtFName.Text == "")
            {
                txtFName.Focus();
                errorProvider1.SetError(txtFName, "Please Fill In The
                First Name");
                flag = 1;
            }
            if (txtLName.Text == "")
            {
                txtLName.Focus();
                errorProvider1.SetError(txtLName, "Please Fill In The
                Last Name");
                flag = 1;
            }
            if (dtpDOJ.Value > DateTime.Now)
            {
                dtpDOJ.Focus();
                errorProvider1.SetError(dtpDOJ, "Please Fill The
                Correct Date of Joining");
                flag = 1;
            }
            if (txtAddress.Text == "")
            {
                txtAddress.Focus();
                errorProvider1.SetError(txtAddress, "Please Fill In
                The Address");
                flag = 1;
            }
            if (txtQuali.Text == "")
            {
                txtQuali.Focus();
                errorProvider1.SetError(txtQuali, "Please Fill In The
                Qualification");
                flag = 1;
            }
            return flag;
        }
```

7. Double-click the **Form1.cs** file in the **Solution Explorer** window.
8. Double-click the **btnSubmit** button control in the design window of the **Form1.cs** file.

   Code in the **btnSubmit** button control's **click** event is as follows:

```
private void btnSubmit_Click(object sender, EventArgs e)
```

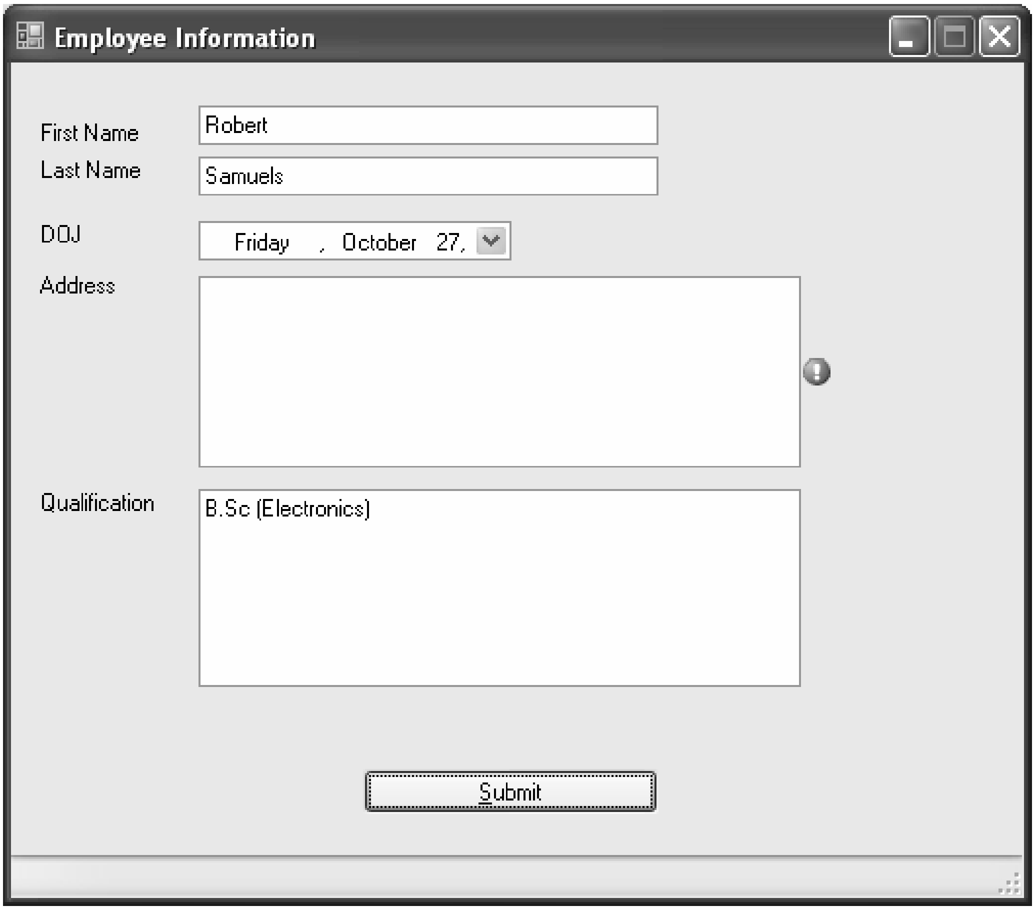
```
{
  if (ValidateData() == 0)
  {
        errorProvider1.Clear();
        MessageBox.Show("Submitted");
        tssWarning.Text = "";
  }
  else
  {
         MessageBox.Show("Correct The Error(s)");
  }
}
```

Task 4: Executing the Application and Verifying the Output

To execute the application and verify the output, you need to perform the following steps:

1. Select **Debug→Start Debugging** or press the **F5** key to execute the application.
2. Check the output of the application by providing the correct input and by entering the incorrect input. You will be prompted with a message in the **StatusStrip** control when the focus enters in the text boxes.

When you enter invalid information or leave the text box blank an appropriate message will be shown in the **ErrorProvider** control, as displayed in the following figure.



First Name	Robert
Last Name	Samuels
DOJ	Friday , October 27,
Address	
Qualification	B.Sc (Electronics)

The Form at Run Time

3. Close the Visual Studio application.

Practice Questions

1. Which of the following property is used to fill any remaining space on the StatusStrip control with the StatusLabel control?
   a. Text property
   b. Spring property
   c. Size property
   d. Padding property

2. Name the property, which specifies the layout orientation of the StatusStrip control.

3. Name the property of the ErrorProvider control, which indicates the rate in milliseconds at which the error icon blinks.

4. List the child controls that can be added to the StatusStrip control.

5. Name the property, which is added to the property list of a form when an ErrorProvider control is added to the form.

Summary

In this chapter, you learned that:

- Validation is a means to check the accuracy and the correctness of the data entered by the user.
- The Validating event enables you to prevent the user from shifting the focus from the control being validated to some other control on the form until all validation rules have been met.
- The `MessageBox` class is used to display prompts and messages on the screen. After closing the message box the user cannot see the message.
- An instance of the `MessageBox` class cannot be created by using the new keyword.
- The StatusStrip control is derived from `System.Windows.Forms.ToolStrip` class. It is a container control that can hold other controls.
- The StatusStrip control contains four types of controls. The controls are:
  - StatusLabel
  - ProgressBar
  - DropDownButton
  - SplitButton
- The ErrorProvider control is used to display an error message, whenever the user positions the mouse pointer over the error icon.

Exercises

Exercise 1

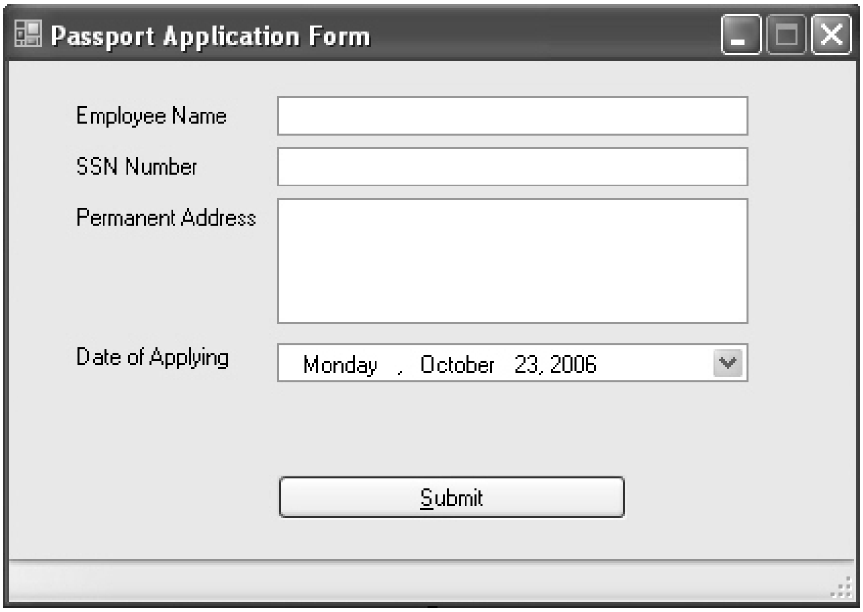
To get a passport from the passport office, you need to fill up the Passport application form. Create a form that requires information such as name, SSN number, permanent address, and the date of applying.

You will also need to give appropriate messages, using the StatusStrip control, when a user enters data in the text boxes. In addition, you need to perform checks to ensure the following:

- The text boxes cannot be left empty.
- The SSN number text box should allow the user to enter numeric values only. In addition, the Delete and the Backspace key should also work in the text box.
- The date of applying should be later than the current date.

If the above conditions are not fulfilled by the user, then an appropriate message should be displayed using the ErrorProvider control.

The created form should be similar to the following window of the Passport Application Form.



Monday , October 23, 2006