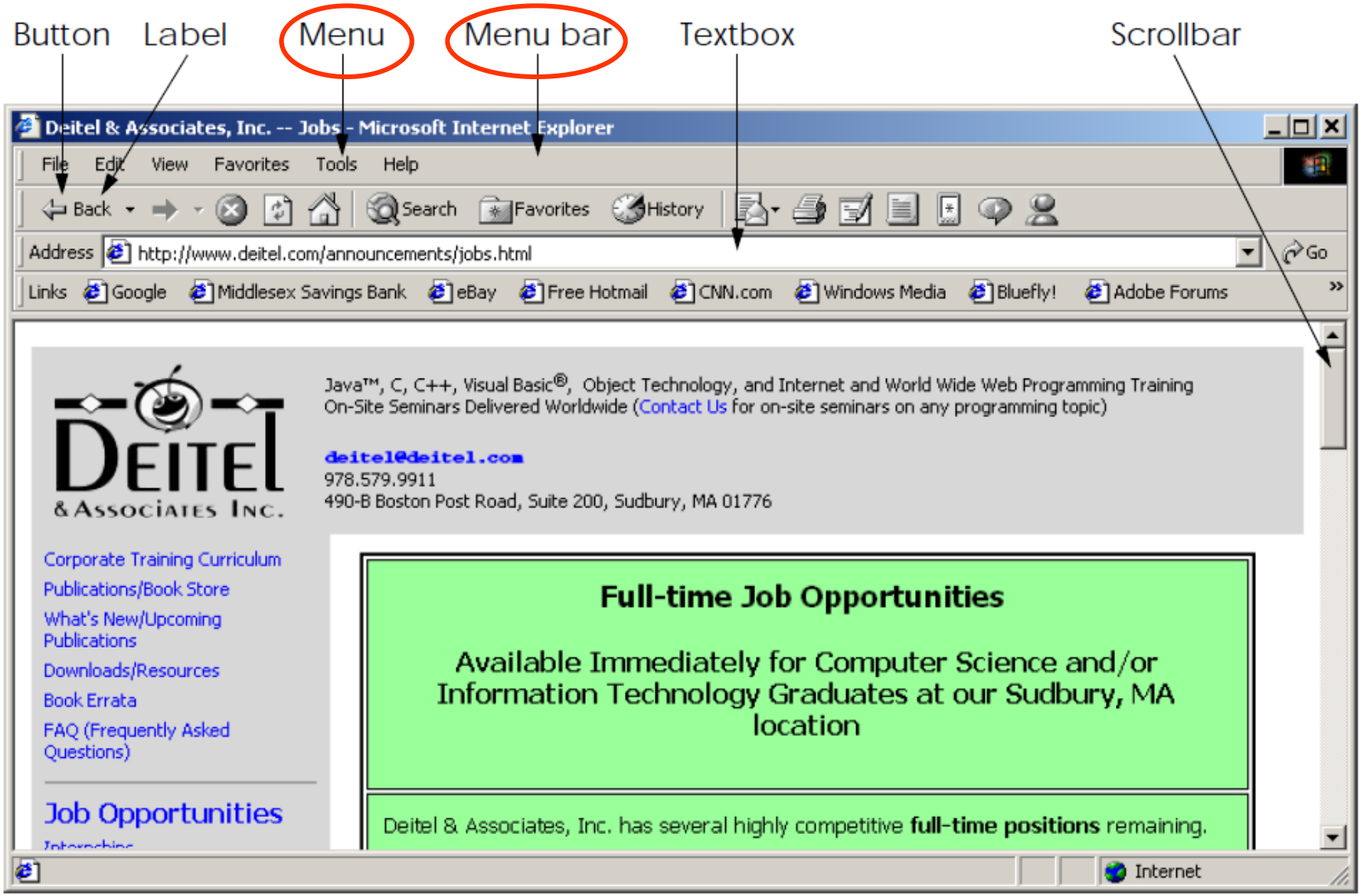
Chapter 12 - Graphical User Interface Concepts: Part 1

Introduction



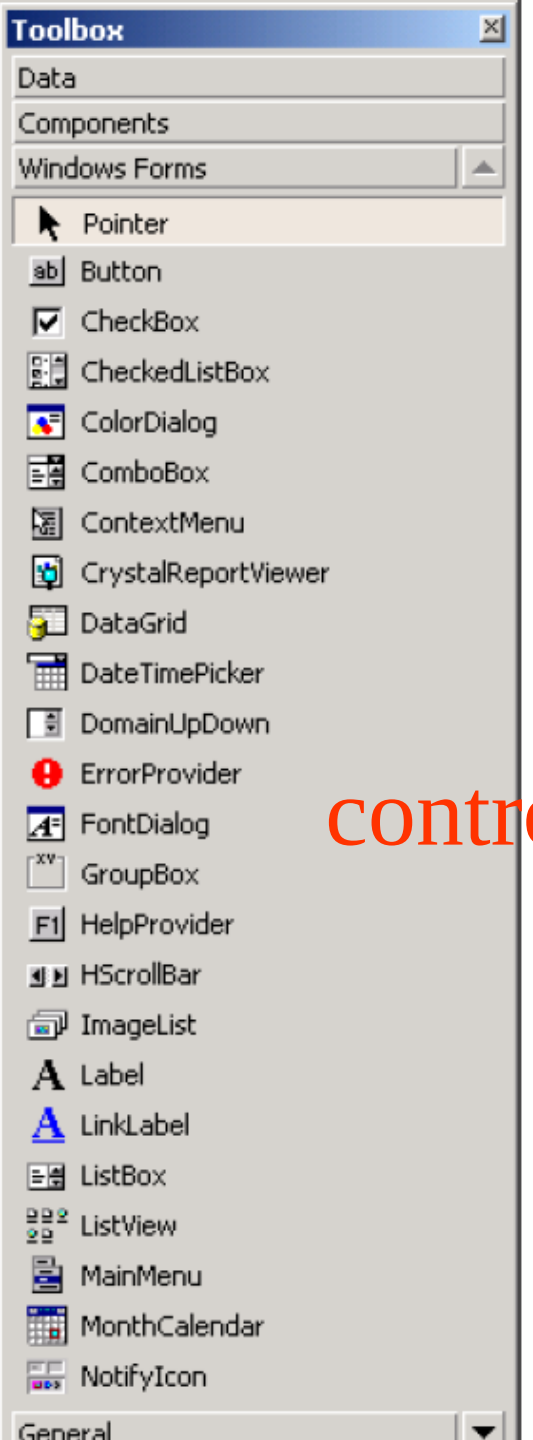
Introduction

Control	Description
`Label`	An area in which icons or uneditable text can be displayed.
`TextBox`	An area in which the user inputs data from the keyboard. The area also can display information.
`Button`	An area that triggers an event when clicked.
`CheckBox`	A GUI control that is either selected or not selected.
`ComboBox`	A drop-down list of items from which the user can make a selection, by clicking an item in the list or by typing into the box, if permitted.
`ListBox`	An area in which a list of items is displayed from which the user can make a selection by clicking once on any element. Multiple elements can be selected.
`Panel`	A container in which components can be placed.
`ScrollBar`	Allows the user to access a range of values that cannot normally fit in its container.

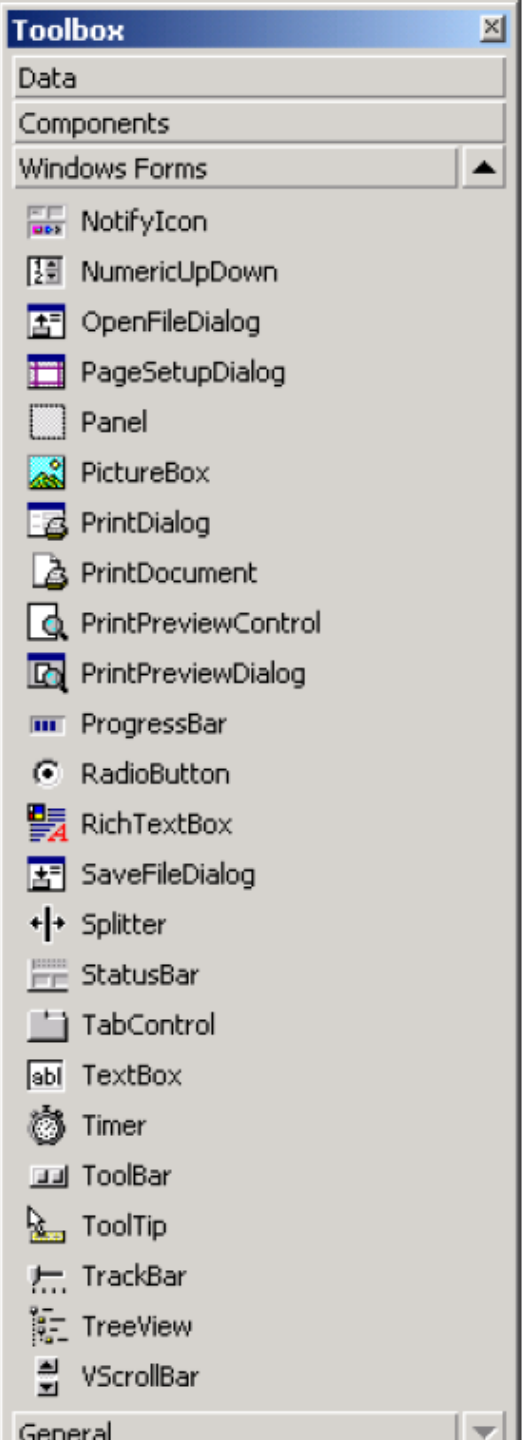
Fig. 12.2 Some basic GUI components .

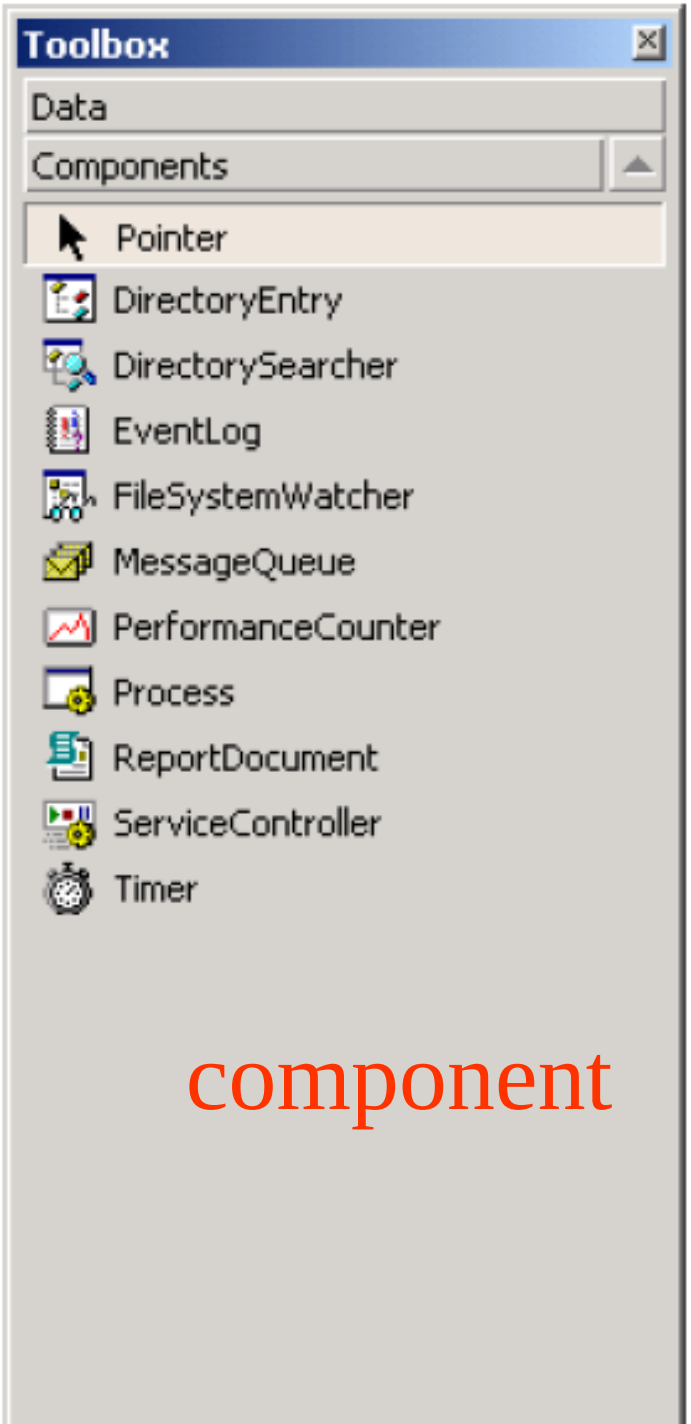
12.2 Windows Forms

- Component
  - the term "component" is generally used for an object that is reusable
  - It is a class with the property that it implements IComponent interface
    - Such as printReporter, Socket(networking) component,
  - Generally, it lacks visual parts
- Control
  - (It is also a class) Component with graphical part
    - Such as button, textbox or label
  - A control is a component that provides user-interface (UI) capabilities.



control





component

Component & control from MSDN

- Component:
  - In the .NET Framework, a component is a class that
    - implements System.ComponentModel.Icomponent interface
    - Or derives directly or indirectly from a class that implements IComponent.
- Control:
  - .NET Framework provides two base classes for controls:
    - System.Windows.Forms.Control for client-side Windows Forms controls
      - It derives from Component and itself provides UI capabilities.
    - System.Web.UI.Control for ASP.NET server controls
      - it implements IComponent and provides the infrastructure on which it is easy to add UI functionality.
  - All controls derive directly or indirectly from these two classes.

檔案(F) 編輯(E) 檢視(V) 專案(P) 建置(B) 偵錯(D)

開始 Debug

Form1.cs [設計] Program.cs

Form1

方案總管

搜尋 方案總管 (Ctrl+;)

- 方案 'WindowsFormsApplication1' (1 專案)
  - WindowsFormsApplication1
    - Properties
    - 參考
    - App.config
    - Form1.cs
      - Form1.Designer.cs
      - Form1.resx
      - Form1
        - Form1()
        - components : IContainer
        - Dispose(bool) : void
        - InitializeComponent() : void
    - Program.cs

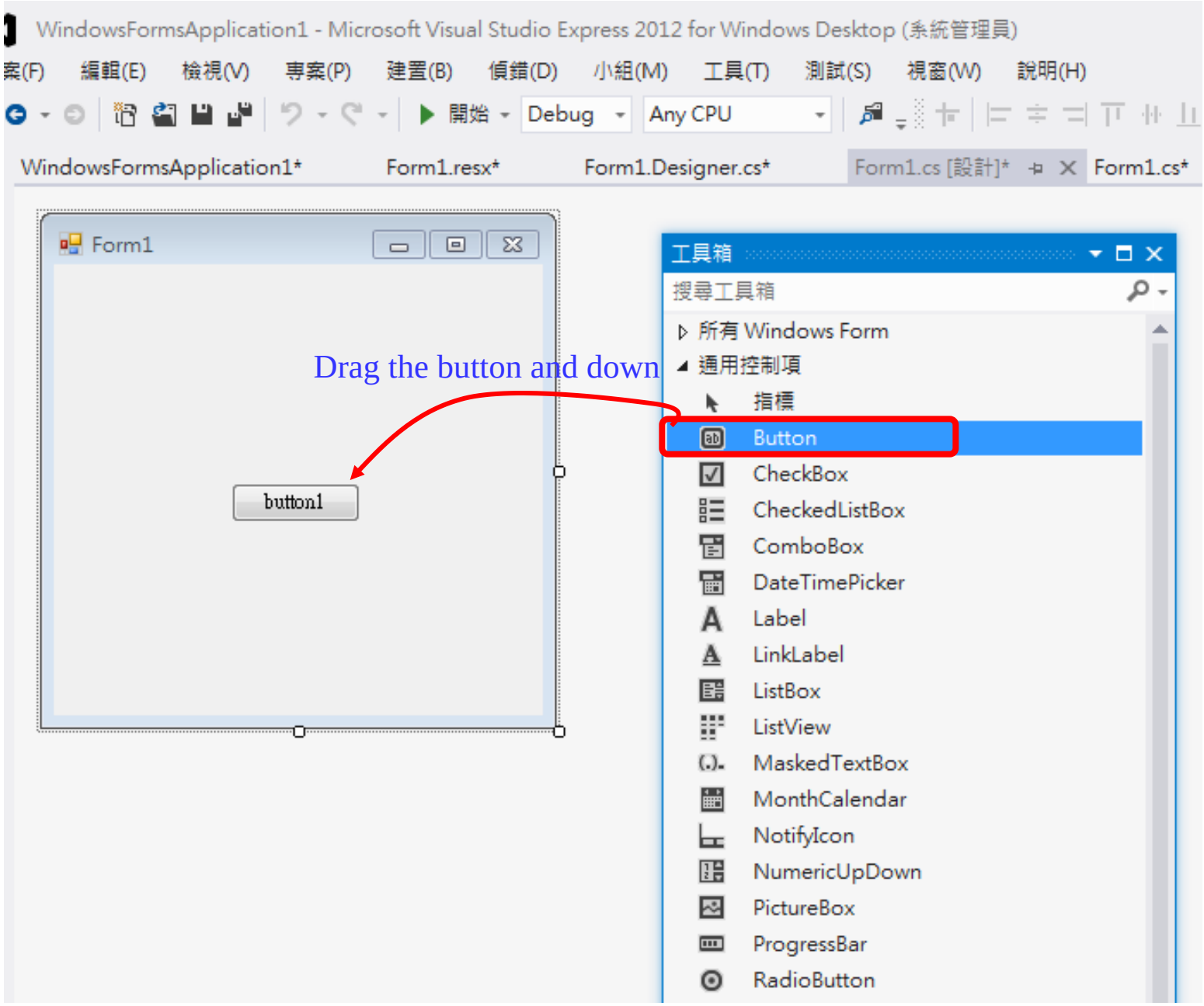
屬性

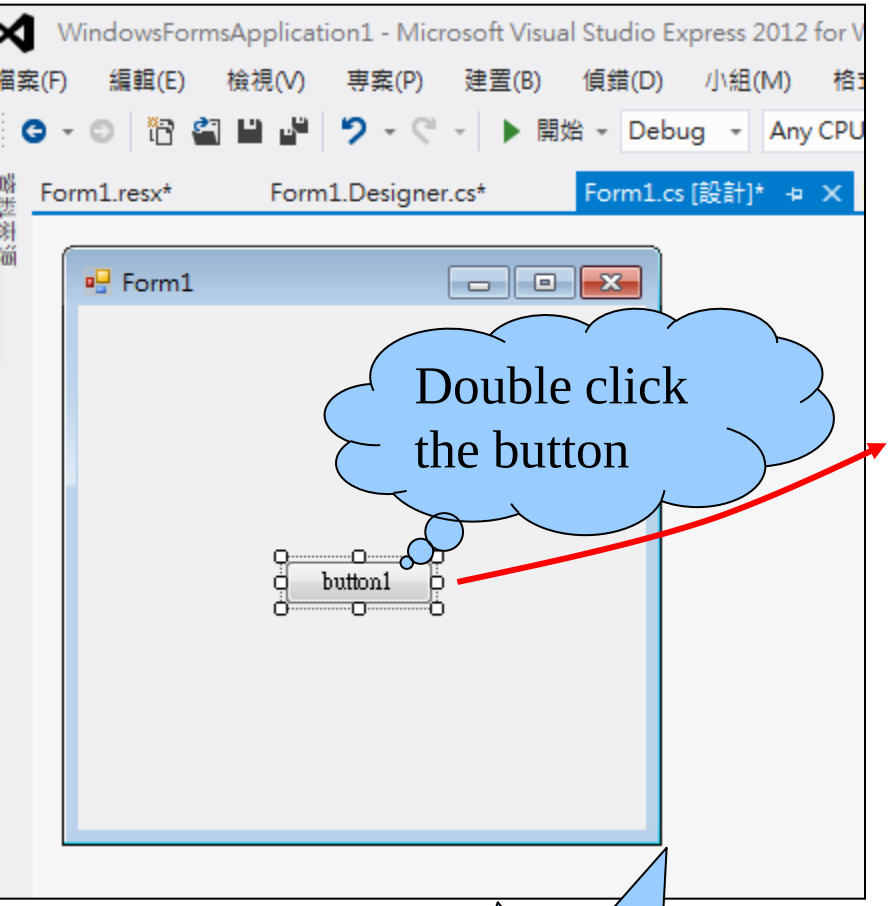
Form1 System.Windows.Forms.Form

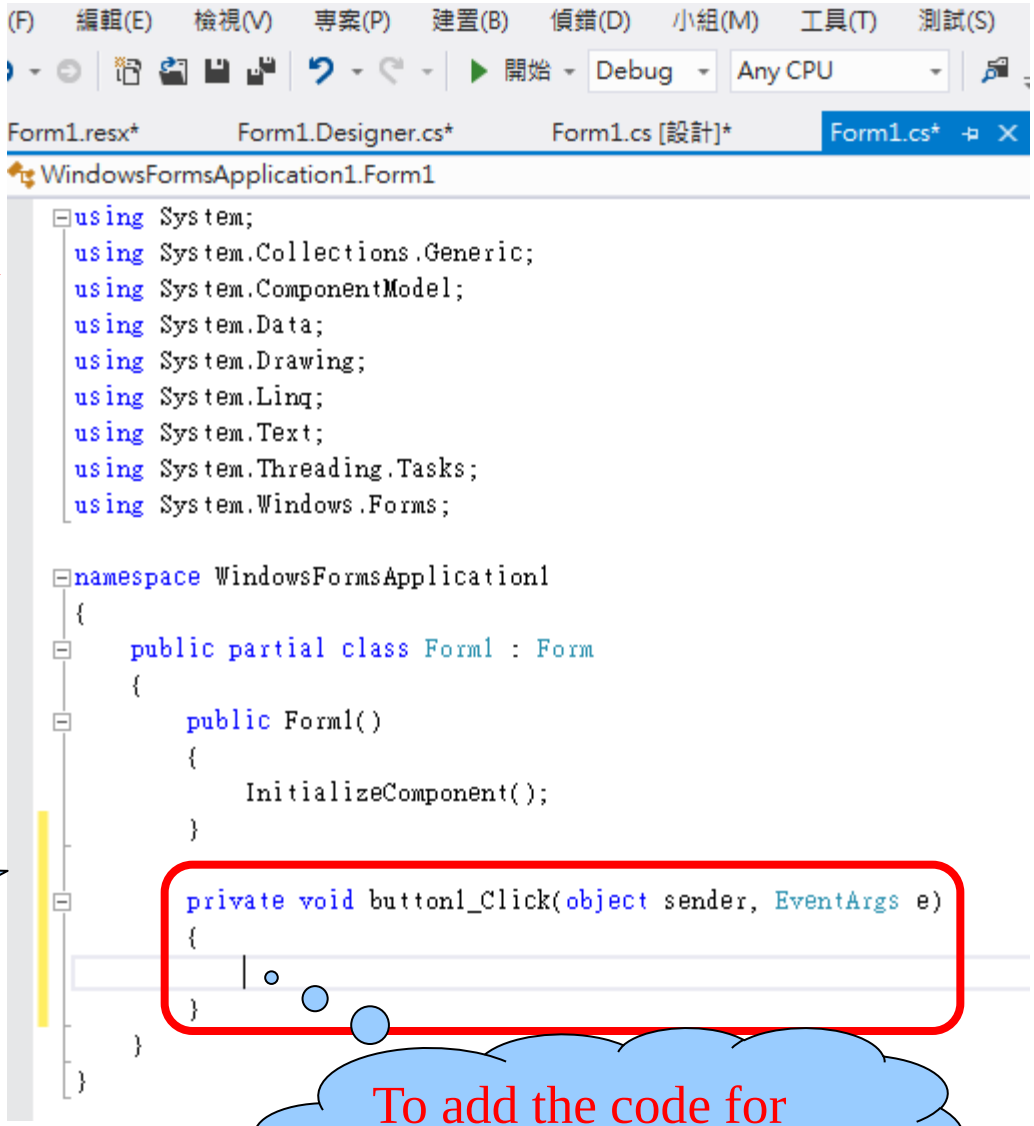
外觀

BackColor	Control
BackgroundImage	(無)
BackgroundImageLayout	Tile
Cursor	Default
Font	新細明體, 9pt
ForeColor	ControlText
FormBorderStyle	Sizable
RightToLeft	No

Form Properties and Events	Description / Delegate and Event Arguments
Common Properties	
AcceptButton	Which button will be clicked when *Enter* is pressed.
AutoScroll	Whether scrollbars appear when needed (if data fill more than one screen).
CancelButton	Button that is clicked when the *Escape* key is pressed.
FormBorderStyle	Border of the form (e.g., **none**, **single**, **3D**, **sizable**).
Font	Font of text displayed on the form, as well as the default font of controls added to the form.
Text	Text in the form's title bar.
Common Methods	
Close	Closes form and releases all resources. A closed form cannot be reopened.
Hide	Hides form (does not release resources).
Show	Displays a hidden form.
Common Events	*(Delegate* ***EventHandler****, event arguments* ***EventArgs****)*
Load	Occurs before a form is shown. Visual Studio .NET generates a default event handler when the programmer double clicks on the form in the designer.







Codes for button click add to Form1. cs

Basic Event Handling

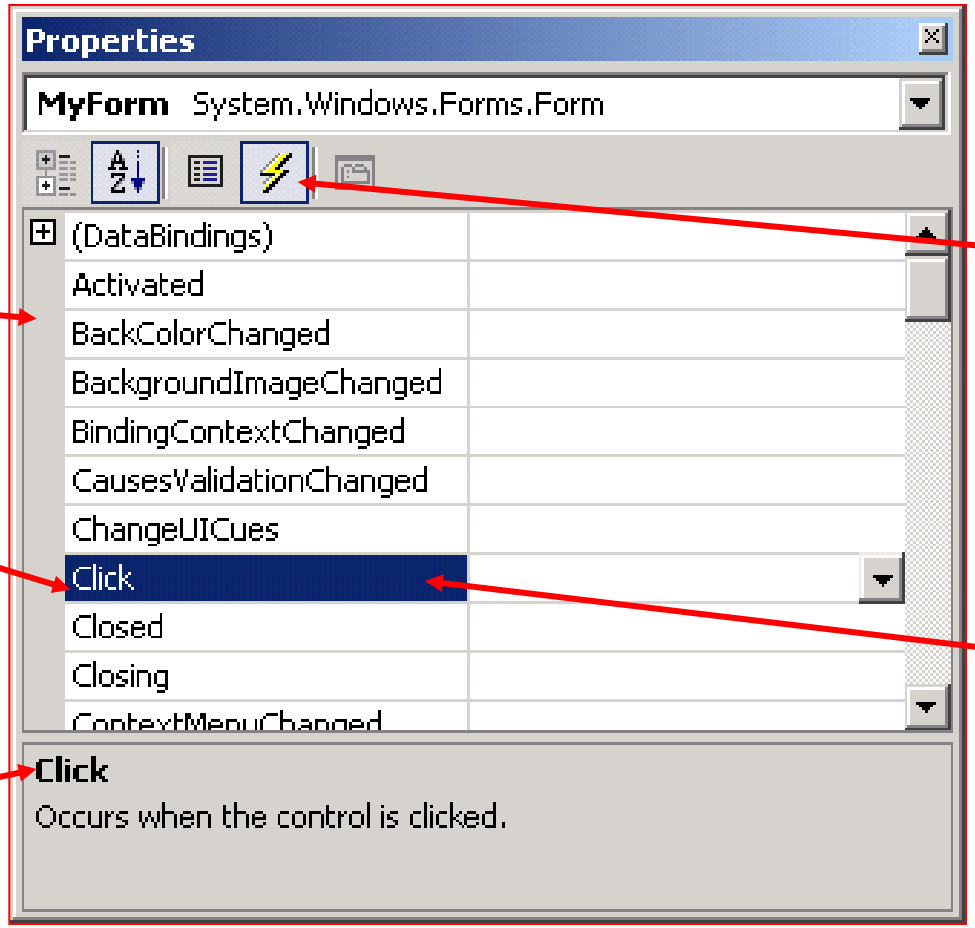


Fig. 12.6 Events section of the **Properties** window.

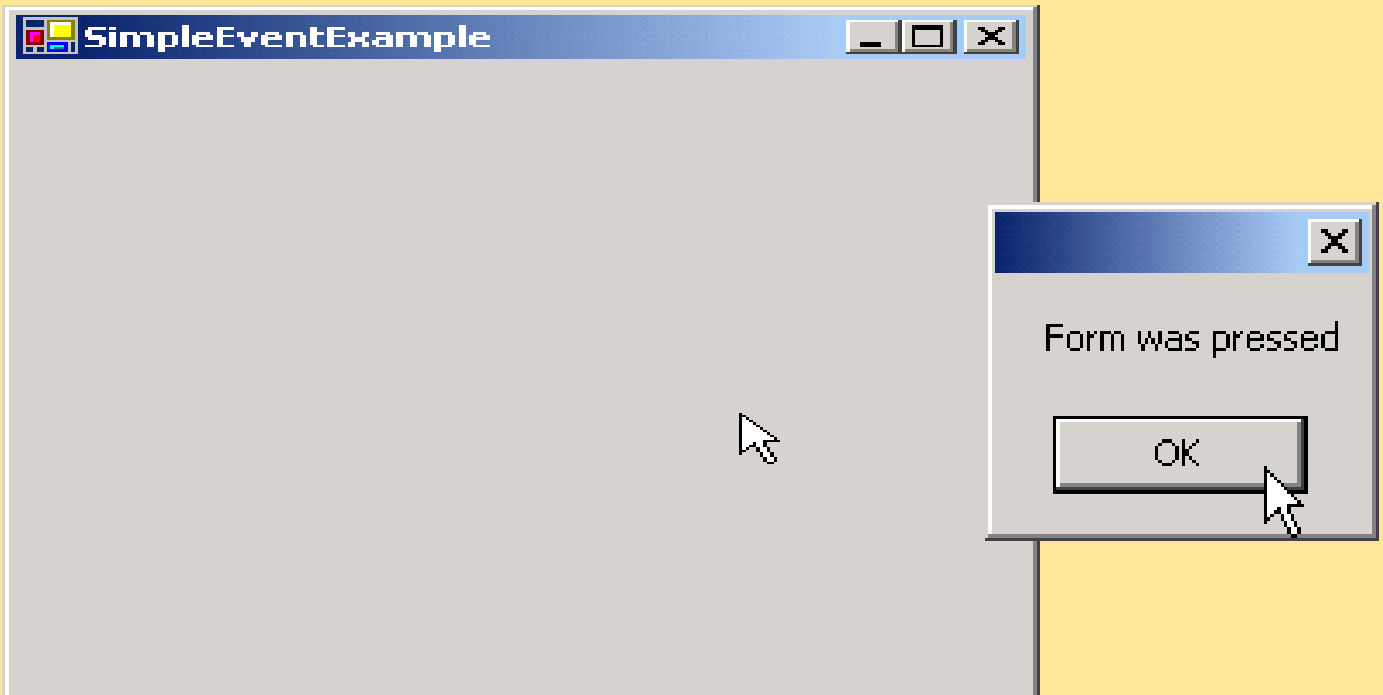
13.4 Event Handling

- Event
  - Generally, it is generated by users or outer system,
    - like movement from mouse or keyboard, package arriving from some web site that you are querying
  - Event handlers performs action (codes) response to the events
    - codes written by programmer or system default

13.4 Event Handling

- Event-handling model
  - Three parts
    1. Event source (like button)
       - is a GUI component with which user can interact
    2. Event object (like mouse_Move, button_Click)
       - Encapsulates information about event that occurred
    3. Event listener
       - Receives event object when notified by OS, then the listener responds to the event
  - Programmer must perform two tasks
    1. Register event listener for event source (C# can be done automatically)
    2. Implement event-handling method (event handler)

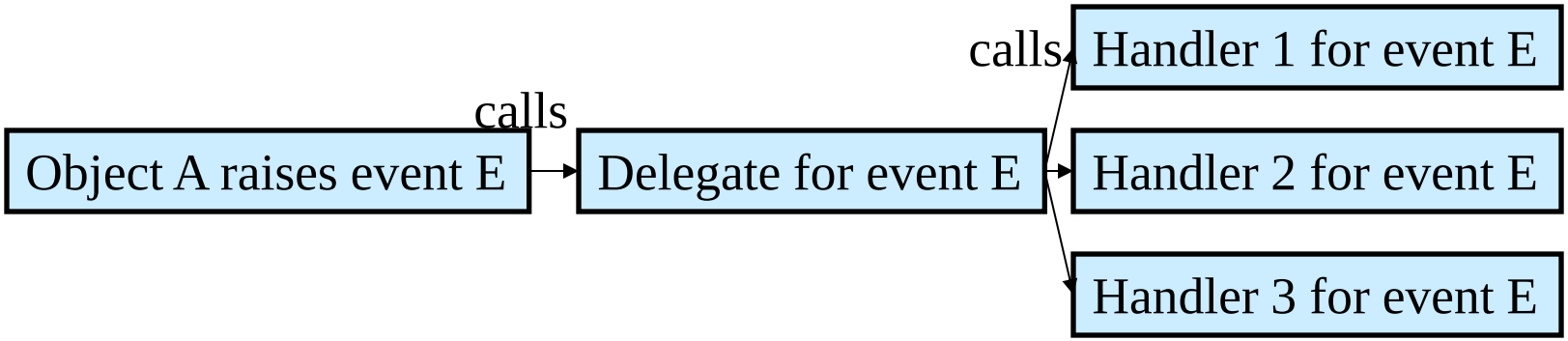
```
3   Using System;
4   using System.Drawing;
5   using System.Collections;
6   using System.ComponentModel;
7   using System.Windows.Forms;
8   using System.Data;
10  public class MyForm : System.Windows.Forms.Form {
12     private System.ComponentModel.Container components = null;
14     [STAThread]
15     static void Main() {
17        Application.Run( new MyForm() );
18     }
21     private void MyForm_Click( object sender, System.EventArgs e ) {
23        MessageBox.Show( "Form was pressed" );
24     }
25  }
```



12.3 Event-Handling Model

- (Associated with event) delegate
  - Contain lists of method references
    - Method and delegatge's parameter must have same signature
  - Delegate is intermediaries for objects and methods
  - Two object reference (sender and event) are passed into, through
    - ControlName_EventName
- Steps for delegate using in event handling (automatically in C#)
  1. Declare a delegate
  2. Create a delegate
  3. Add event handler to delegate

12.3 Event-Handling Model



Event multicasting

Have multiple handlers for one event

Fig. 12.5 Event-handling model using delegates.

12.4 Control Properties and Layout

- Common properties of control
  - Text property
    - Specifies the text appearing on a control
  - Focus method
    - Transfers the focus to a control, becoming active control
  - TabIndex property
    - Order in which controls are given focus when pressed tab
    - Automatically set by Visual Studio .NET if not specified
  - Enable property
    - Indicate a control's accessibility

12.4 Control Properties and Layout

- Common properties of control
  - Visibility control
    - Hide control from user, using method Hide
  - Anchor property
    - Anchoring control to specific location, like top margin
      - Constant distance from specified location
      - Unanchored control will move
    - Docking allows control to spread itself along and entire side
    - Both options refer to the parent container
  - Size structure
    - Allow for specifying size range
      - Can use MinimumSize and MaximumSize property

12.4 Control Properties and Layout

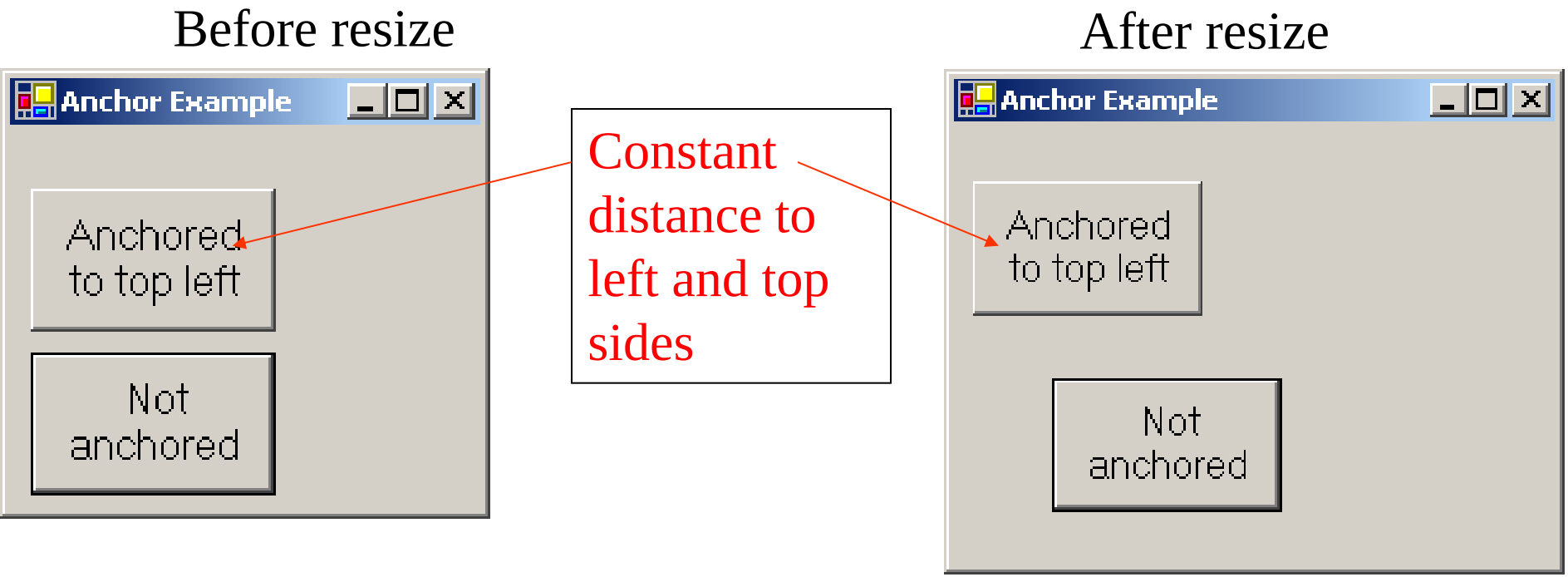
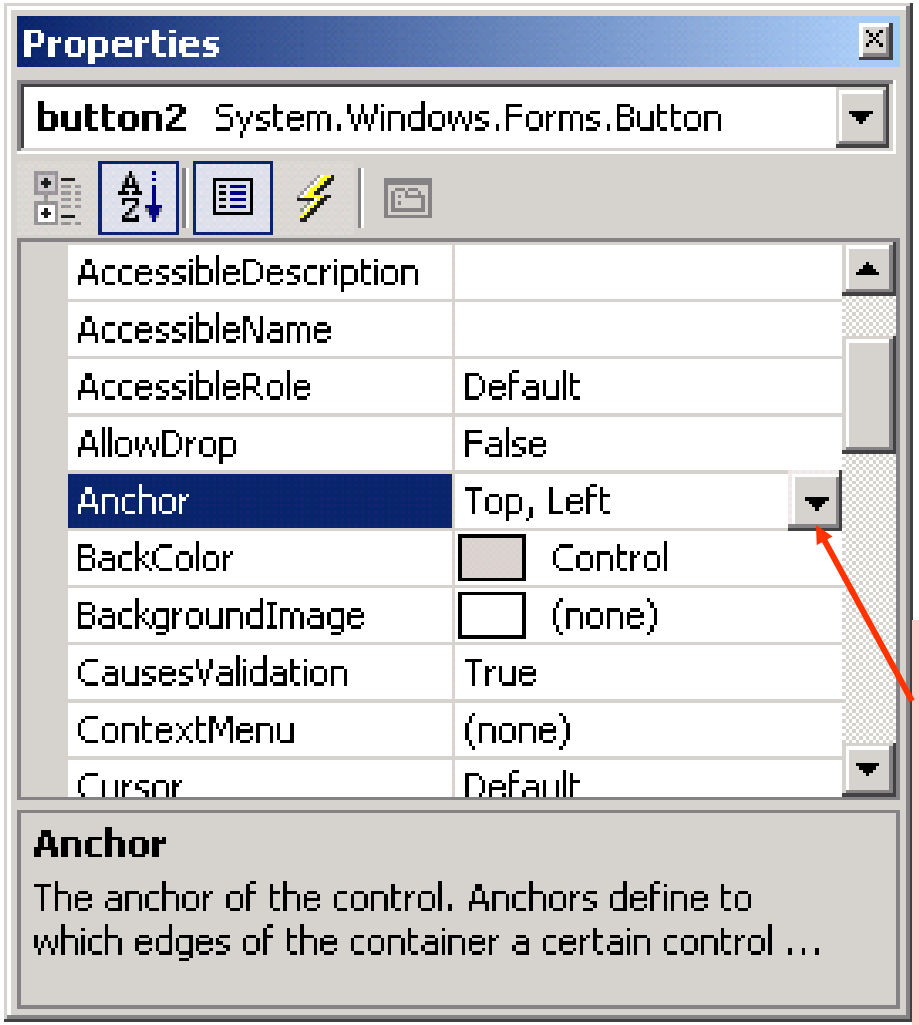
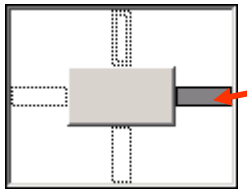


Fig. 12.11 Anchoring demonstration.

12.4 Control Properties and Layout





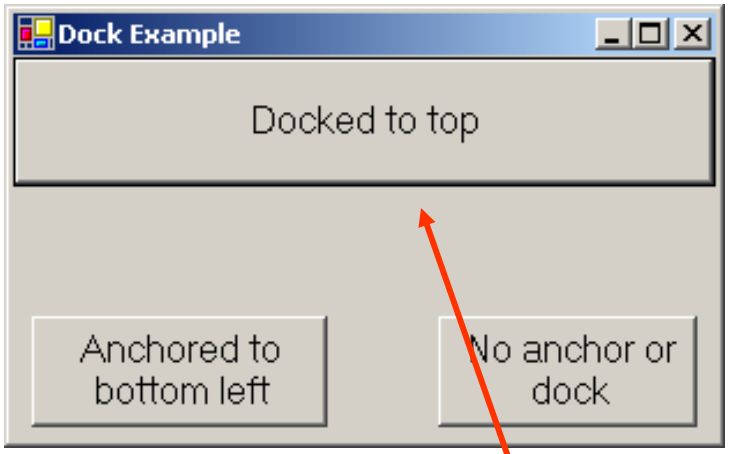
Darkened bar indicates to which wall control is anchored

Click down-arrow in Anchor property to display anchoring window

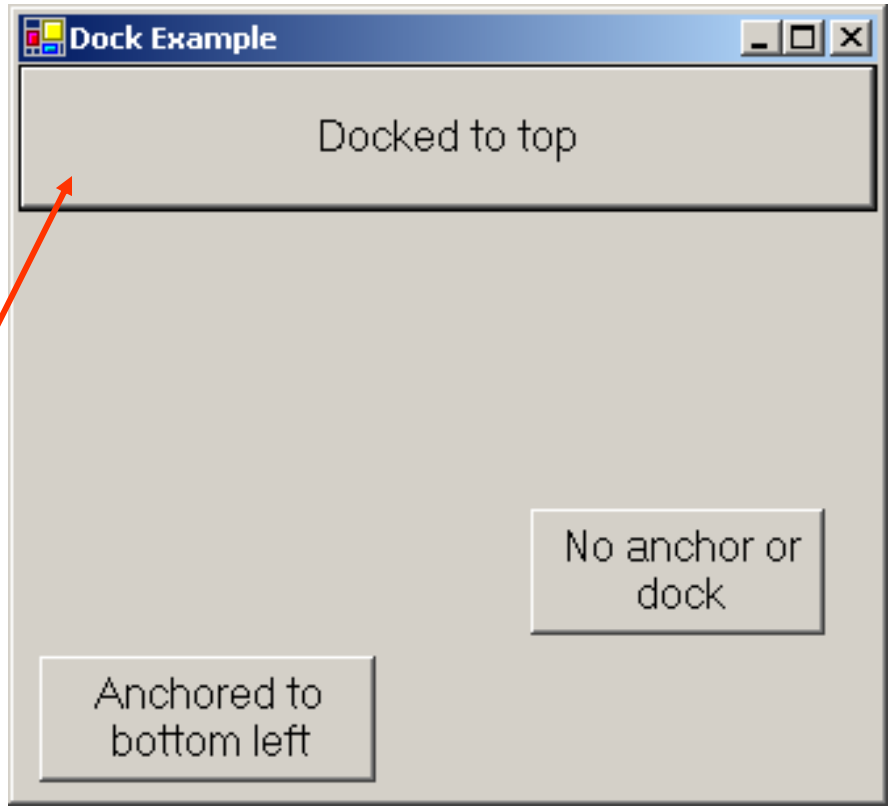
Fig. 12.12 Manipulating the **Anchor** property of a control.

12.4 Control Properties and Layout

Before resize



After resize



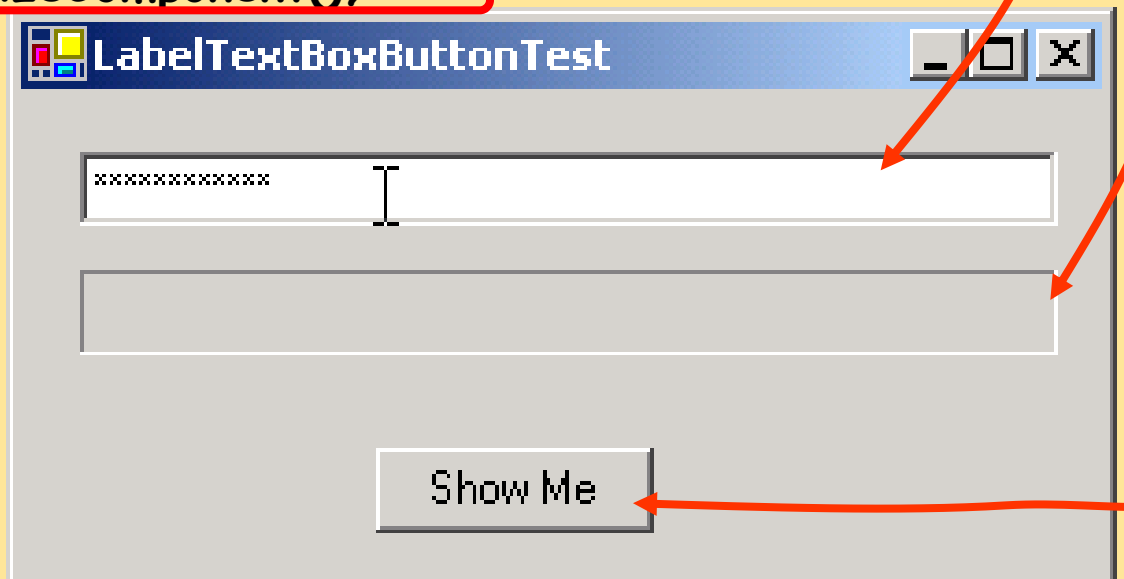
Control expands along top portion of the form

Class `Control` Properties and Methods	Description
Common Properties	
`BackColor`	Background color of the control.
`BackgroundImage`	Background image of the control.
`Enabled`	Whether the control is enabled (i.e., if the user can interact with it). A disabled control will still be displayed, but "grayed-out"—portions of the control will become gray.
`Focused`	Whether a control has focus. (The control that is currently being used in some way.)
`Font`	**`Font`** used to display control's **`Text`**.
`ForeColor`	Foreground color of the control. This is usually the color used to display the control's **`Text`** property.
`TabIndex`	Tab order of the control. When the *Tab* key is pressed, the focus is moved to controls in increasing tab order. This order can be set by the programmer.
`TabStop`	If **`true`**, user can use the *Tab* key to select the control.
`Text`	Text associated with the control. The location and appearance varies with the type of control.
`TextAlign`	The alignment of the text on the control. One of three horizontal positions (left, center or right) and one of three vertical positions (top, middle or bottom).
`Visible`	Whether the control is visible.

Label Properties	Description / Delegate and Event Arguments
Common Properties	
Font	The font used by the text on the **Label**.
Text	The text to appear on the **Label**.
TextAlign	The alignment of the **Label**'s text on the control. One of three horizontal positions (**left**, **center** or **right**) and one of three vertical positions (**top**, **middle** or **bottom**).

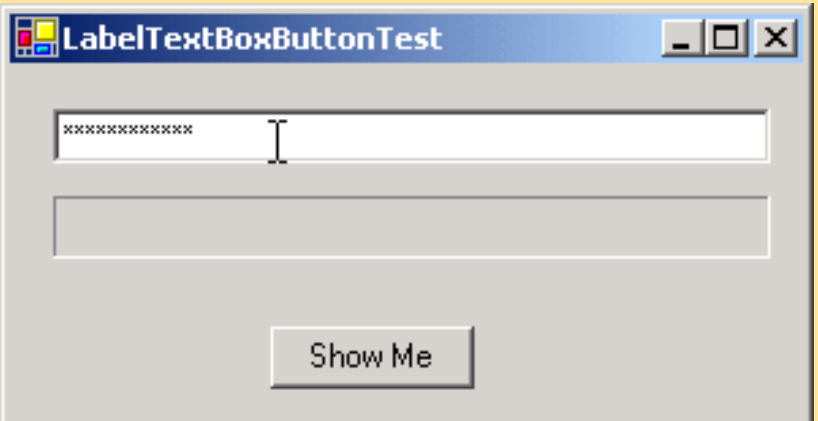
TextBox Properties and Events	Description / Delegate and Event Arguments
Common Properties	
AcceptsReturn	If **true**, pressing *Enter* creates a new line if textbox spans multiple lines. If **false**, pressing *Enter* clicks the default button of the form.
Multiline	If **true**, textbox can span multiple lines. Default is **false**.
PasswordChar	Single character to display instead of typed text, making the **TextBox** a password box. If no character is specified, **Textbox** displays the typed text.
ReadOnly	If **true**, **TextBox** has a gray background and its text cannot be edited. Default is **false**.
ScrollBars	For multiline textboxes, indicates which scrollbars appear (**none**, **horizontal**, **vertical** or **both**).
Text	The text to be displayed in the text box.

```
4  using System;
5  using System.Drawing;
6  using System.Collections;
7  using System.ComponentModel;
8  using System.Windows.Forms;
9  using System.Data;
11 namespace LabelTextBoxButtonTest {
17  public class LabelTextBoxButtonTest :
      System.Windows.Forms.Form{
20     private System.Windows.Forms.Button displayPasswordButton;
21     private System.Windows.Forms.Label displayPasswordLabel;
22     private System.Windows.Forms.TextBox inputPasswordTextBox;
26     private System.ComponentModel.Container components = null;
28         public LabelTextBoxButtonTest() {
30             InitializeComponent();
31         }
```



```
35      protected override void Dispose( bool disposing ) {
37         if ( disposing ) {
39            if ( components != null ) {
41               components.Dispose();
42            }
43         }
44         base.Dispose( disposing );
45      }
46   #region Windows Form Designer generated code
51      private void InitializeComponent() {
53         this.displayPasswordButton =
54            new System.Windows.Forms.Button();
55         this.inputPasswordTextBox =
56            new System.Windows.Forms.TextBox();
57         this.displayPasswordLabel =
58            new System.Windows.Forms.Label();
59         this.SuspendLayout();
```

Code in region can be expand or collapse by vs.net editor



Call form class's method

```
63          this.displayPasswordButton.Location =
64              new System.Drawing.Point( 96, 96 );
65          this.displayPasswordButton.Name =
66              "displayPasswordButton";
67          this.displayPasswordButton.TabIndex = 1;
68          this.displayPasswordButton.Text = "Show Me";
69          this.displayPasswordButton.Click +=
70              new System.EventHandler(
71              this.displayPasswordButton_Click );
75          this.inputPasswordTextBox.Location =
76              new System.Drawing.Point( 16, 16 );
77          this.inputPasswordTextBox.Name =
78              "inputPasswordTextBox";
79          this.inputPasswordTextBox.PasswordChar = '*';
80          this.inputPasswordTextBox.Size =
81              new System.Drawing.Size( 264, 20 );
82          this.inputPasswordTextBox.TabIndex = 0;
83          this.inputPasswordTextBox.Text = "";
87          this.displayPasswordLabel.BorderStyle =
88              System.Windows.Forms.BorderStyle.Fixed3D;
89          this.displayPasswordLabel.Location =
90              new System.Drawing.Point( 16, 48 );
91          this.displayPasswordLabel.Name =
92              "displayPasswordLabel";
```

EventHandler is a delegate

displayxx_.Click displayxx__Click are methods

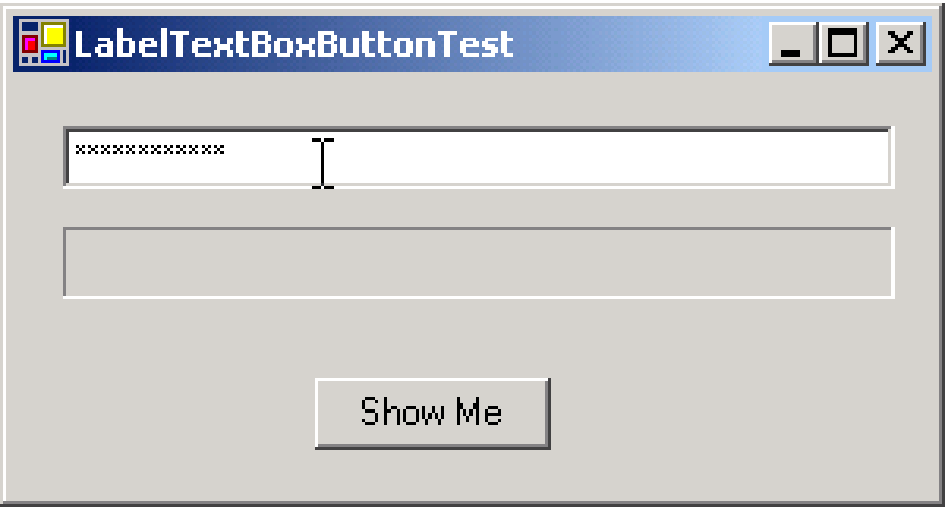
```
93          this.displayPasswordLabel.Size =
94             new System.Drawing.Size( 264, 23 );
95          this.displayPasswordLabel.TabIndex = 2;
99          this.AutoScaleBaseSize =
100            new System.Drawing.Size( 5, 13 );
101         this.ClientSize =
102            new System.Drawing.Size( 292, 133 );
103         this.Controls.AddRange(
104            new System.Windows.Forms.Control[] {
105               this.displayPasswordLabel,
106               this.inputPasswordTextBox,
107               this.displayPasswordButton  } );
108         this.Name = "LabelTextBoxButtonTest";
109         this.Text = "LabelTextBoxButtonTest";
110         this.ResumeLayout( false );
111      }
113      #endregion
```

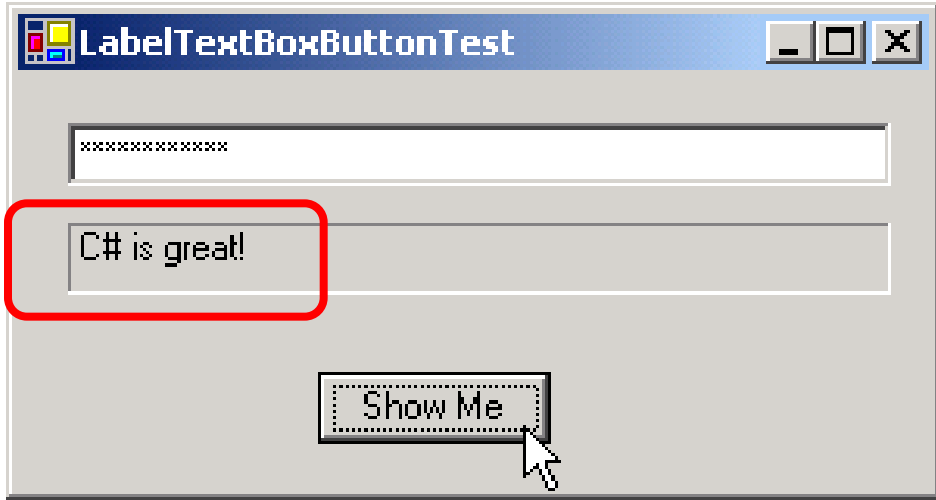
Control.Name is for identified (ID) in codes;

Control.Text is for shown in screen

```
117      [STAThread]
118      static void Main() {
120         Application.Run( new LabelTextBoxButtonTest() );
121      }
```

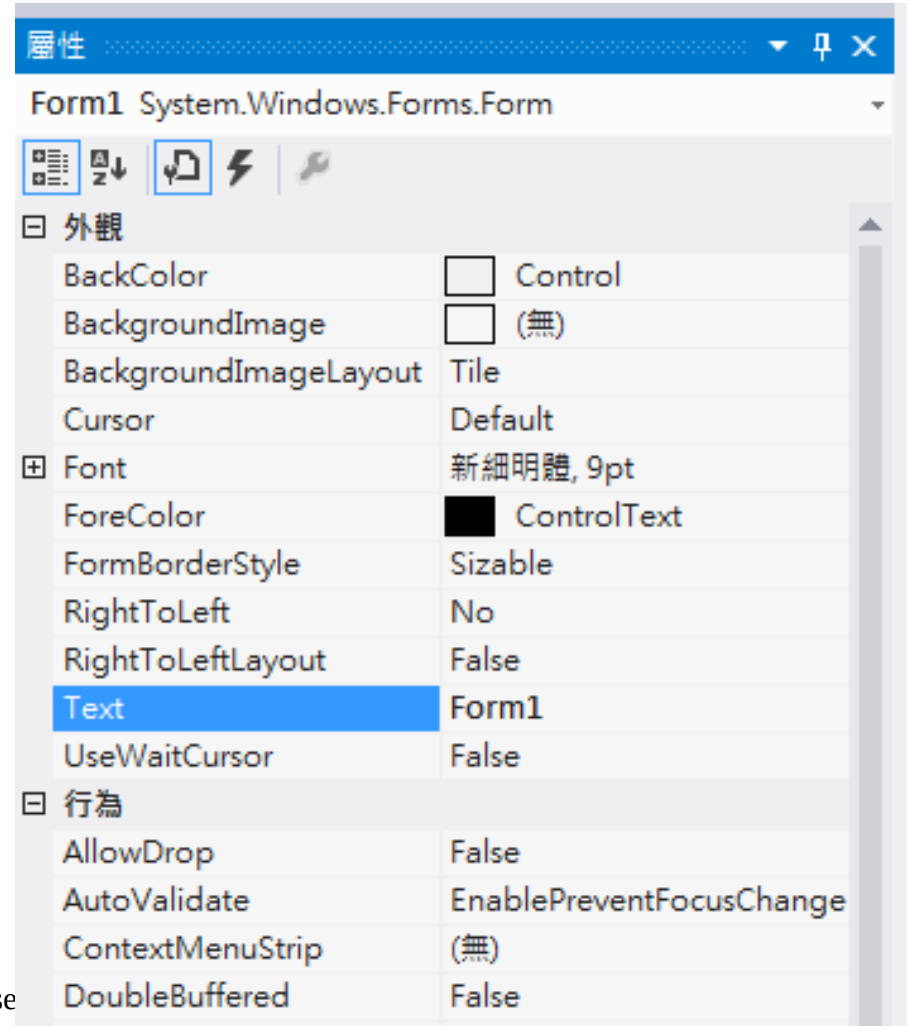
```
123         protected void displayPasswordButton_Click(
124             object sender, System.EventArgs e ) {
127             displayPasswordLabel.Text =
                    inputPasswordTextBox.Text;
129         }
130     }
131 }
```





Setting a property on your form

- The property window is a powerful tool that you can use to change all visual and functional properties for the form and the control in the form



Form1.Designer.cs | Form1.cs [設計] | Form1.cs

WindowsFormsApplication1.Form1

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

方案總管

搜尋 方案總管 (Ctrl+;)

- 方案 'WindowsFormsApplication1' (1 專案)
  - WindowsFormsApplication1
    - Properties
    - 參考
    - App.config
    - Form1.cs
      - Form1.Designer.cs
      - Form1.resx
      - Form1
        - Form1()
        - components : IContainer
        - Dispose(bool) : void
        - InitializeComponent() : void
    - Program.cs

Partial class means a class can be defined in two files (but with the same class name)

Form1.Designer.cs | Form1.cs [設計] | Form1.cs | Program.cs

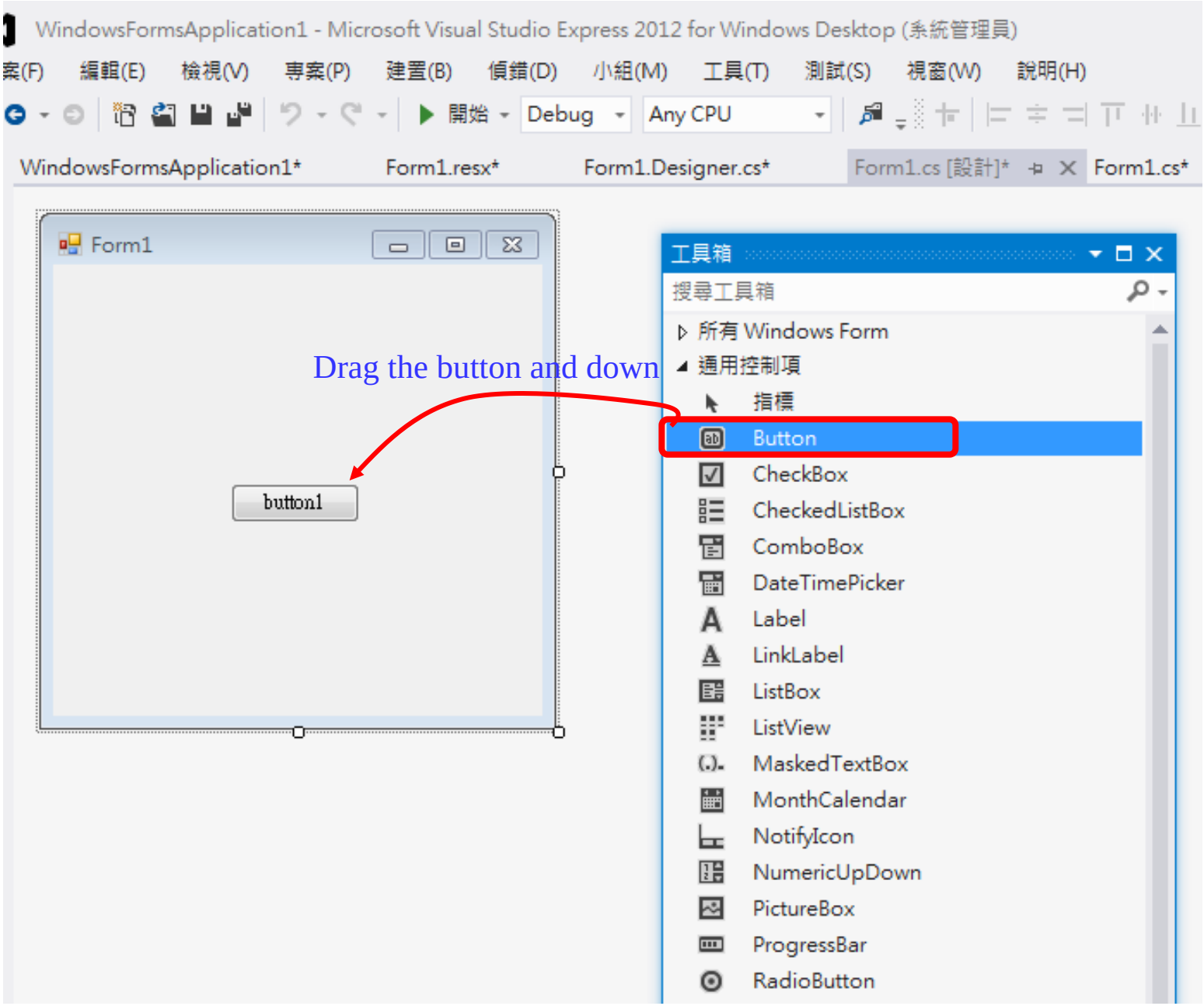
WindowsFormsApplication1.Form1

```
namespace WindowsFormsApplication1
{
    partial class Form1
    {
        /// <summary>
        /// 設計工具所需的變數。
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// 清除任何使用中的資源。
        /// </summary>
        /// <param name="disposing">如果應該處置 Managed 資源則為 true，否則為 false。</
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form 設計工具產生的程式碼

        /// <summary>
        /// 此為設計工具支援所需的方法 - 請勿使用程式碼編輯器
        /// 修改這個方法的內容。
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.Text = "Form1";
        }
```

方案總管

搜尋 方案總管 (Ctrl+;)

- 方案 'WindowsFormsApplication1' (1 專案)
  - WindowsFormsApplication1
    - Properties
    - 參考
    - App.config
    - Form1.cs
      - Form1.Designer.cs
      - Form1.resx
      - Form1
        - Form1()
        - components : IContainer
        - Dispose(bool) : void
        - InitializeComponent() : void
    - Program.cs



WindowsFormsApplication1* Form1.resx* Form1.Designer.cs* Form1.cs [設計]

WindowsFormsApplication1.Form1

```
namespace WindowsFormsApplication1
{
    partial class Form1
    {
        /// <summary> ...
        private System.ComponentModel.IContainer components = null;

        /// <summary> ...
        protected override void Dispose(bool disposing)...

        #region Windows Form 設計工具產生的程式碼

        /// <summary>
        /// 此為設計工具支援所需的方法 - 請勿使用程式碼編輯器
        /// 修改這個方法的內容。
        /// </summary>
        private void InitializeComponent()
        {
            this.button1 = new System.Windows.Forms.Button();
            this.SuspendLayout();
            //
            // button1
            //
            this.button1.Location = new System.Drawing.Point(103, 127);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(75, 23);
            this.button1.TabIndex = 0;
            this.button1.Text = "button1";
            this.button1.UseVisualStyleBackColor = true;
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 12F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(284, 262);
            this.Controls.Add(this.button1);
            this.Name = "Form1";
```

C# IDE automatically reflect the adding component in Form1.Designer.cs

MultiDelegate in JAVA (for reference)

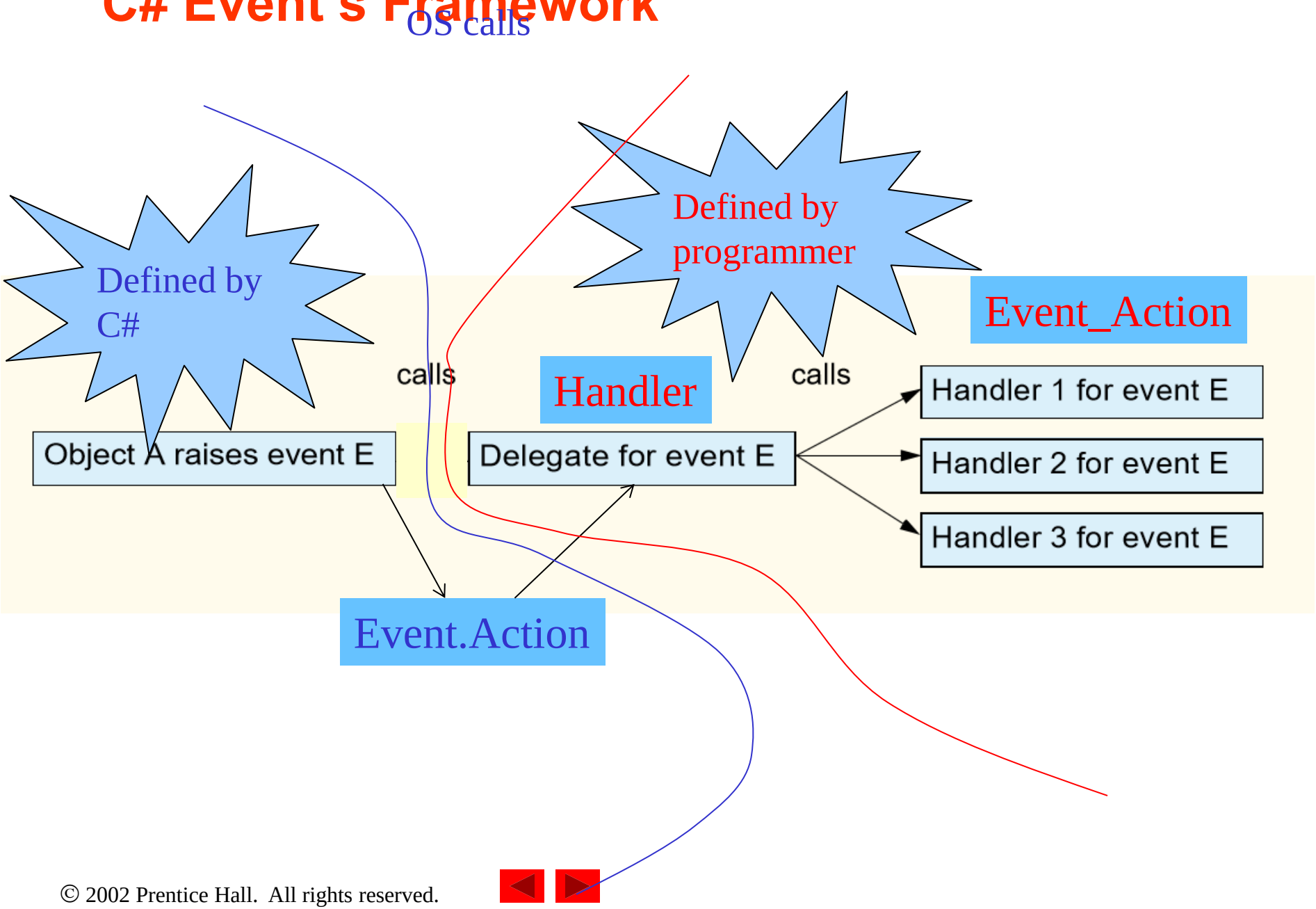
```
3   import java.awt.*;
4   import java.awt.event.*;
5   import javax.swing.*;
7   public class TextFieldTest extends JFrame {
8   private JTextField textField1, textField2, textField3;
9   private JPasswordField passwordField;
12   public TextFieldTest()  {
14        super( "Testing JTextField and JPasswordField" );
16        Container container = getContentPane();
17        container.setLayout( new FlowLayout() );
20        textField1 = new JTextField( 10 );
21        container.add( textField1 );
24        textField2 = new JTextField( "Enter text here" );
25        container.add( textField2 );
```

```
29  textField3 = new JTextField( "Uneditable text
    field",20);
30  textField3.setEditable( false );
31  container.add( textField3 );
34  passwordField = new JPasswordField( "Hidden text" );
35  container.add( passwordField );
38  TextFieldHandler handler = new TextFieldHandler();
39  textField1.addActionListener( handler );
40  textField2.addActionListener( handler );
41  textField3.addActionListener( handler );
42  passwordField.addActionListener( handler );
44  setSize( 325, 100 );
45  setVisible( true );
47  }
```


```
49  public static void main( String args[] ) {
51  TextFieldTest application = new TextFieldTest();
52  application.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
53  }
```

```
56  private class TextFieldHandler implements ActionListener {
59     public void actionPerformed( ActionEvent event ) {
61       String string = "";
64       if ( event.getSource() == textField1 )
65          string = "textField1: " +event.getActionCommand();
68       else if ( event.getSource() == textField2 )
69          string = "textField2: " + event.getActionCommand();
72       else if ( event.getSource() == textField3 )
73          string = "textField3: " + event.getActionCommand();
76       else if ( event.getSource() == passwordField ) {
77          string = "passwordField: " +
78                 new String( passwordField.getPassword() );
79          }
81       JOptionPane.showMessageDialog( null, string );
83     }
85   }
87 }
```

C# Event's Framework



Java Event's Framework