LABELS, BUTTONS AND PICTUREBOXES

When programmers code a Visual Basic program they will often simply edit the properties that are associated with the controls on the form. Each object, however, has properties that can be easily changed in code using dot notation.

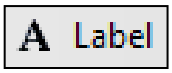
The generalized code for any assignment statement is:

```
Object.Property = PropertyValue
```

This is a simple statement that can be adapted to any object, as we will see with our first two controls: labels and buttons.

LABELS

Labels are probably the most commonly used controls in Visual Basic because they are used to display text to the screen. Although the programmer can write code that could change the content of a label during run-time, the content of a label cannot be directly changed by the user.



The following are just a few of the common label properties:

Name

- Identifies the name of the label.

EXAMPLE: `lblTitle`

Text

- Determines the text that will appear in the label.

EXAMPLE: `lblTitle.Text = "Visual Basic rules!"`

AutoSize

- If set to true, the label is automatically resized based on the font size.
- If set to false, the programmer can manually set the size of the label.

EXAMPLE: `lblTitle.AutoSize = False`

Font

- Sets the font of the text.

EXAMPLE:
```
lblTitle.Font = New Font ("Times New Roman", 14,
                FontStyle.Bold)
```

TextAlign

- Determines the position of the text within the label.
- Values include: **BottomCenter, BottomLeft, BottomRight, MiddleCenter, MiddleLeft, MiddleRight, TopCenter, TopLeft, TopRight**

EXAMPLE: `lblTitle.TextAlign = ContentAlignment.MiddleCenter`

Visible

- Determines whether the label is visible or hidden.

EXAMPLE: `lblTitle.Visible = True`

USING THE *WITH* STATEMENT

Oftentimes programmers will have to set multiple properties on a control. The **With** statement allows programmers to make multiple references to the same object in a more concise manner. For example, if I wanted to set four different properties of a label, I could use a **With statement block** as follows:

```
With lblTitle
   .Text = "Visual Basic Rocks!"
   .AutoSize = False
   .TextAlign = ContentAlignment.TopCenter
   .Visible = False
End With
```

BUTTONS

A **button** is the main control that is used to create **events**. Due to the fact that Visual Basic is an event driven language, an event must take place before any code can be executed. An example of an event that would typically happen in VB is the click event of a button. This event happens when the user clicks the mouse on a button.

The following are just some of the common button properties:

Name

- Identifies the name of the button.

EXAMPLE: `btnOK`

Text

- Determines the text that will appear in the button.

EXAMPLE: `btnOK.Text = "OK"`

AutoSize

- If set to true, the button is automatically resized based on the font size.
- If set to false, the programmer can manually set the size of the button.

EXAMPLE: `btnOK.AutoSize = False`

Font

- Sets the font of the text

EXAMPLE:
```
btnOK.Font = New Font (“Times New Roman”, 14,
     FontStyle.Bold)
```

Enabled

- Indicates whether the control is enabled
- If set to **false**, the button does not respond to mouse clicks.

EXAMPLE: `btnOK.Enabled = False`

FlatStyle

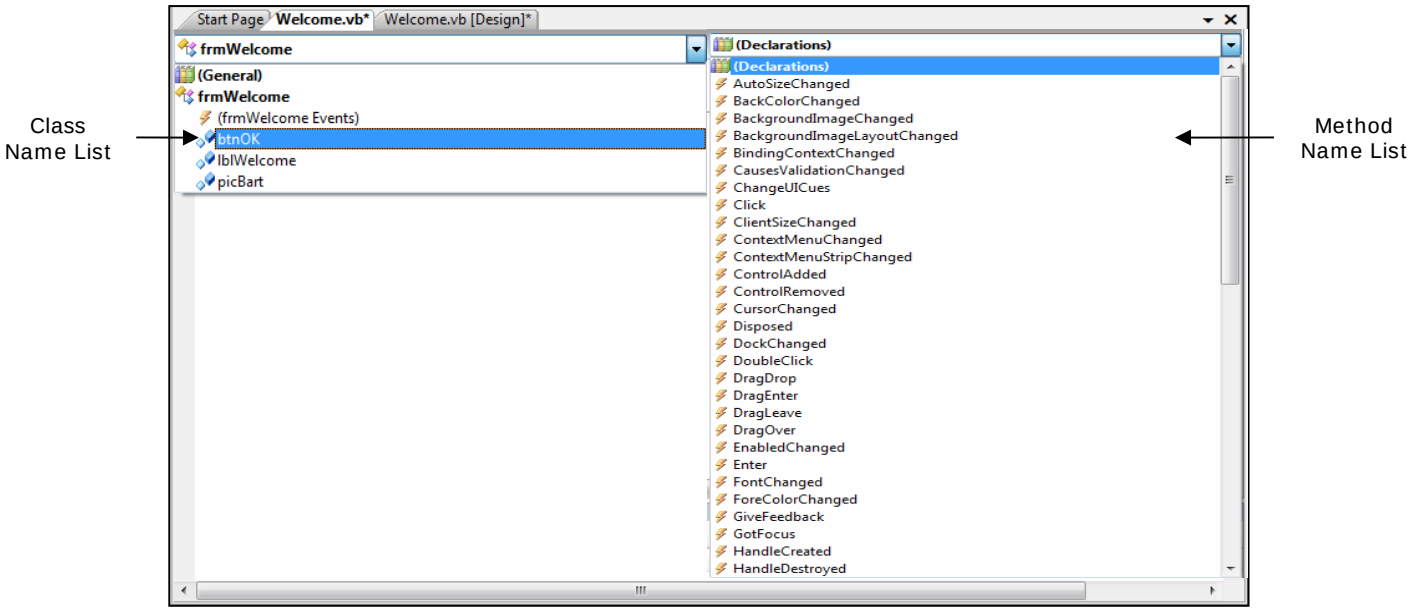
- Determines the appearance of the button when the user moves the mouse over it and clicks it.
- Values include: **Flat, Popup, Standard, System**

EXAMPLE: `lblTitle.Visible = True`

THE EVENT PROCEDURE

A **procedure** is a block of code written to perform specific tasks. An **event procedure**, also called an **event handler**, is a type of procedure that performs tasks in response to user interaction with an object. For example, when the user clicks a button, specific actions should occur.

To add a **Click** event procedure to a button, you could either double-click the button or you could click on View Code and select the control from the **Class Name** list and the event from the **Method Name** list.



The following code appears in the Code Editor that indicates the click event of a button:

```
Private Sub btnOK_Click(ByVal sender As System.Object, ByVal e As
   System.EventArgs) Handles btnOK.Click

End Sub
```

If, for example, I wanted the color of a label to change when the user clicks the button, the code would look something like this:

```
Private Sub btnOK_Click(ByVal sender As System.Object, ByVal e As
   System.EventArgs) Handles btnOK.Click

   'Change the background and foreground colour of the label
   lblTitle.BackColor = Color.FromArbg (100, 0, 0)
   lblTitle.ForeColor = Color.BurlyWood

End Sub
```

ACCEPT AND CANCEL BUTTONS

An **accept button** is a button on a form that is clicked when the user presses the ENTER key. A **cancel button** is a button on a form that is clicked when the user presses the ESC key. Forms have two properties, **AcceptButton** and **CancelButton**, which allow you to designate an accept button and a cancel button. When you selected these properties in the **Properties** window of the form, a drop-down list appears which contains the names of all the buttons on the form. You can then select the button that you want to designate as the accept button or cancel button.

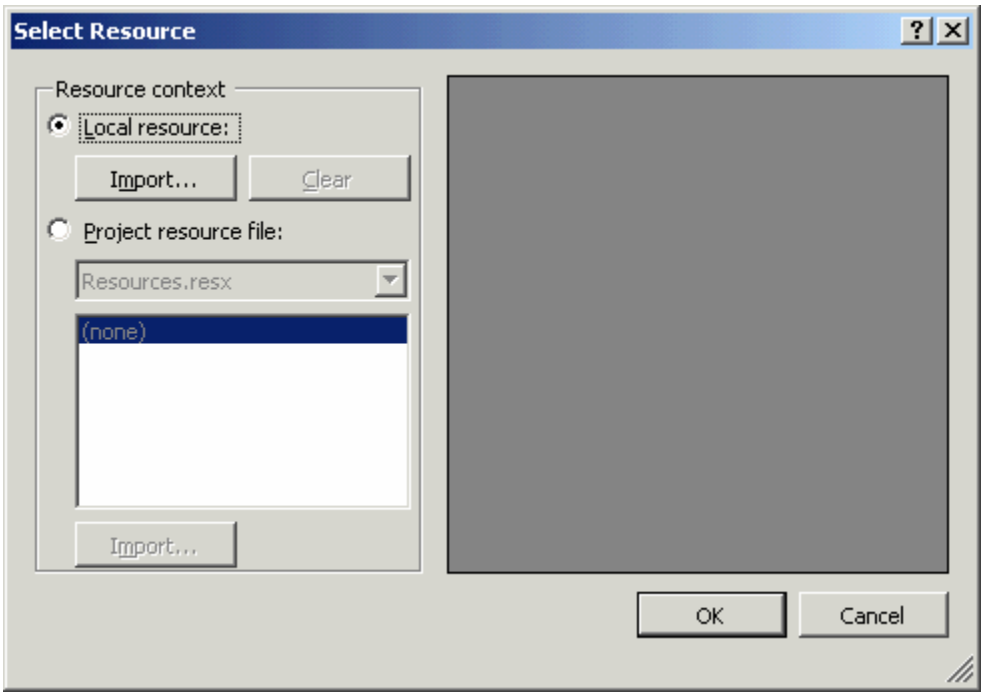
Any button that is frequently clicked should probably be selected as the accept button. This will allow keyboard users to access the button quickly and easily. EXIT or CANCEL buttons are likely candidates to become cancel buttons.

PICTUREBOXES

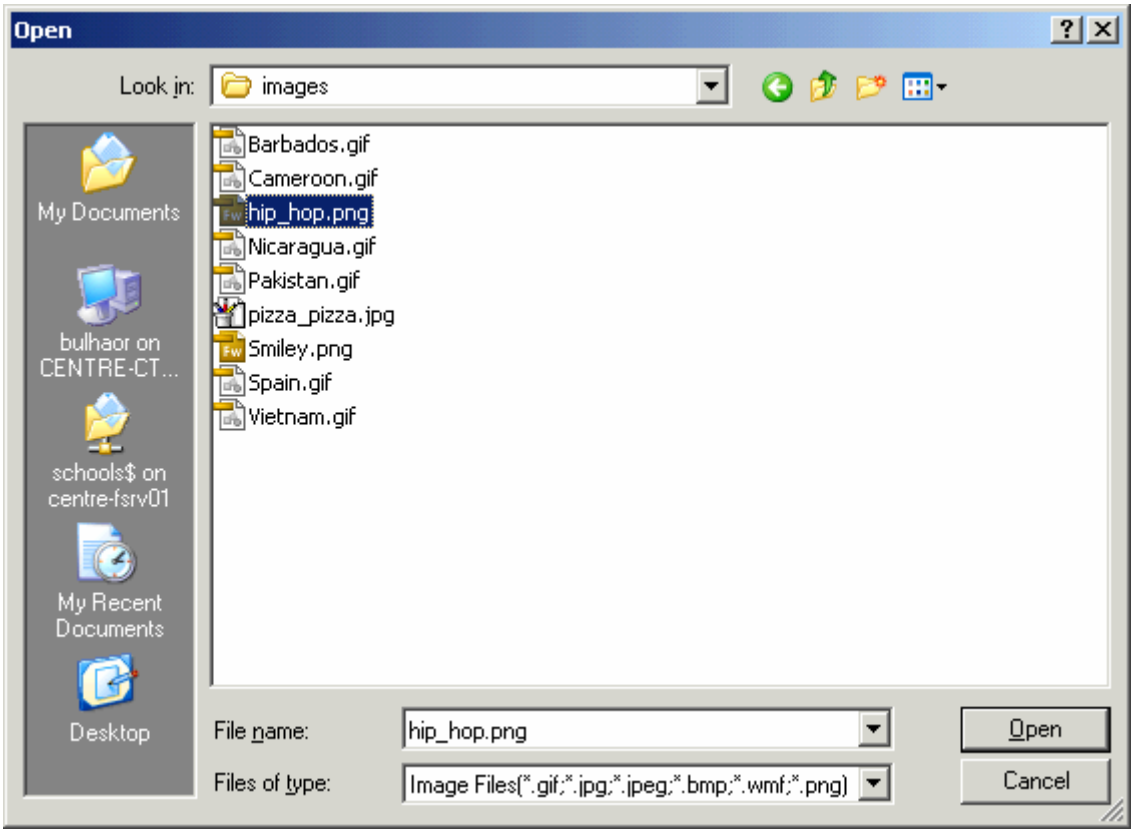
A **PictureBox** object is used when programmers want to display an image on an interface. To insert a picture in a **PictureBox**, you need to first double-click the **PictureBox** icon from the toolbox.

Once you create the **PictureBox**, you need to select the **Image** property from the **Properties Window**:

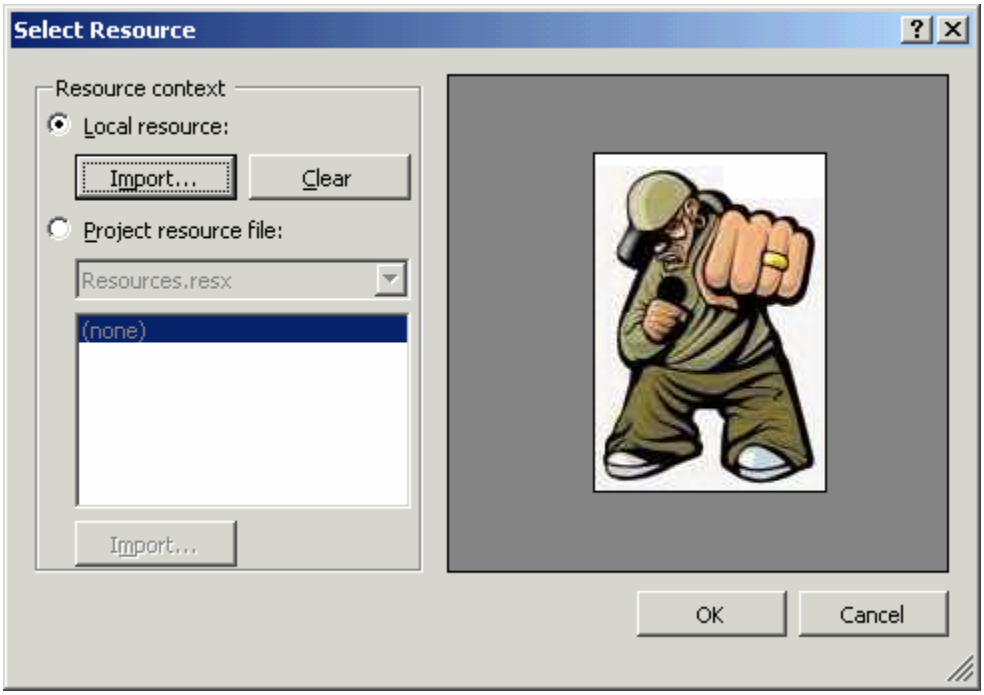
From the **Select Resource** dialog box, you need to select **Local resource** and then the **Import** button:



Next you need to select a picture file and then click **Open**:



Once the picture file has been selected, you need to click **OK**:



In order to control how the PictureBox will handle the placement of the image within the PictureBox and to control the sizing of the image, you will need to become familiar with the **SizeMode** properties of the **PictureBox** object. The properties are as follows:

Normal:	The image is positioned in the upper-left corner of the **PictureBox**, and any part of the image too big for the **PictureBox** is clipped.
StretchImage:	The image stretches to fit the **PictureBox**.
AutoSize:	The **PictureBox** object is resized to always fit the image.
CenterImage:	The image is centered in the **PictureBox**.
Zoom:	The object is resized to fill the **PictureBox**.