Introduction to Visual Basic .NET

Outline

- Elements of a Visual Basic Application
- Getting started in Visual Basic
- Adding an event procedure
- Adding controls
- Adding additional event procedures
- Focus on program design and implementation: Creating a main menu
- Knowing about: The help facility
- Common programming errors and problems

Elements of a VB Application

The Visual Basic language is an object-oriented language that consists of two fundamental parts :

- The Visual part -- consists of a set of objects.
- The Language (code) part -- consists of a high-level procedural programming language.
- To create an application -- which is a VB application or program that can be run under the Windows operating system both elements of the language, objects and code, must be used together.

The Visual Element

- The visual part of an application consists of the graphical user interface (GUI) of the application.
- A GUI is constructed by placing a set of visual objects on a form.
- The standard object Toolbox contains the objects that can be used in constructing a GUI.
- Each object contains two basic characteristics:
  - Properties -- define particular characteristics of the object and
  - Methods -- are the predefined procedures that are supplied with the object for performing specific tasks.
- Each object from the Toolbox recognizes certain actions, which are referred to as *events*.

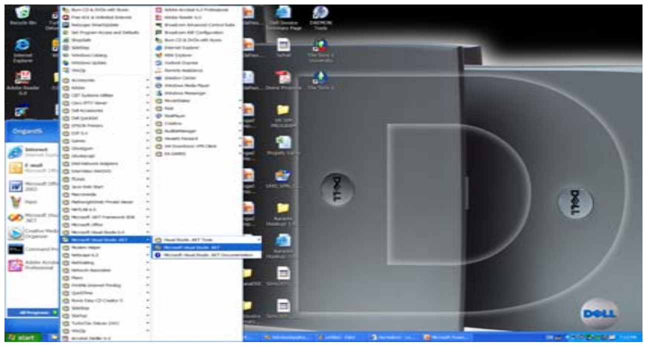
The Language Element

- Visual Basic is a high-level programming language that supports all of the procedural programming features found in other modern languages.
- In GUIs and event-driven applications, the code that is executed depends on what events occur, which in turn depends on what the user does.

Getting Started in VB

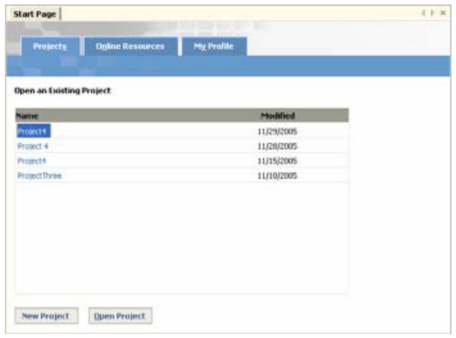
- Visual Studio is the integrated development environment (IDE) that is used to create, test, and debug projects.
- Launching Visual Basic .NET displays the Start Page.

Start Visual Studio .NET from the Windows Desktop



Visual Basic .NET Start Page

Start page allows the programmer to
- open recent projects,
- open any previously saved project, and
- create a new project.

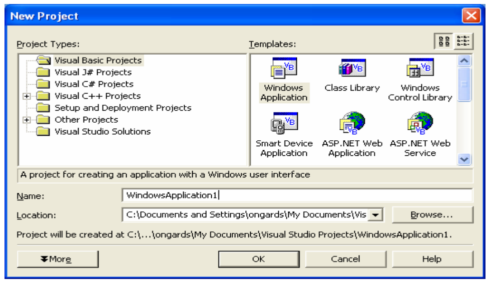


Open an Existing Project

Name	Modified
Project4	11/29/2005
Project 4	11/28/2005
Project4	11/15/2005
ProjectThree	11/10/2005

New Project Dialog Window

Clicking the New project button on the Start Page to open the New Project Dialog box


New Project

Project Types:
- Visual Basic Projects
- Visual J# Projects
- Visual C# Projects
- Visual C++ Projects
- Setup and Deployment Projects
- Other Projects
- Visual Studio Solutions

Templates:
- Windows Application
- Class Library
- Windows Control Library
- Smart Device Application
- ASP.NET Web Application
- ASP.NET Web Service

A project for creating an application with a Windows user interface

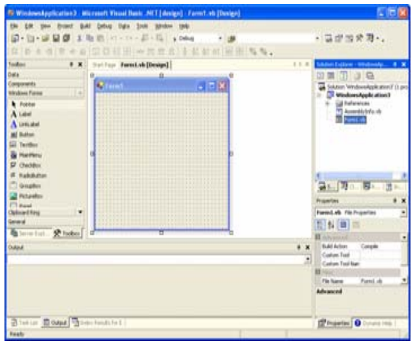
Name: WindowsApplication1

Location: C:\Documents and Settings\ongards\My Documents\Vis

Project will be created at C:\...\ongards\My Documents\Visual Studio Projects\WindowsApplication1.

VB .NET IDE Windows Workspace

- When a new project is created, the GUI designer component of the IDE is displayed.
- The IDE also has two other components: a code editor and a debugger.
- The IDE offers all of the capabilities of a Windows-based application, such as the ability to resize and close any of the child windows, as well as the overall parent window.

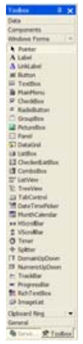


VB .NET Workspace

VB .NET Applications

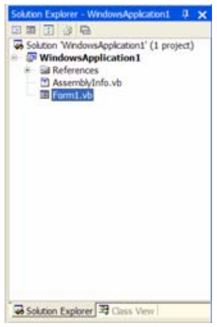
- The steps that are required for creating a Visual Basic application are:
  1. Create the graphical user interface (GUI).
  2. Set the properties of each object on the interface.
  3. Write the code or add events.
  4. Debug -- test the application.
- The first step, creating the GUI, consists of adding objects from the Toolbox to the design form.

Toolbox



- The Toolbox window contains a set of controls that can be placed on a Form window to produce a graphical user interface (GUI – “goo-ey”).
- The toolbox can be opened by choosing the command Toolbox in the View menu.

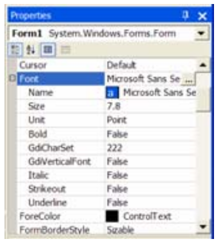
Solution Explorer Window



Solution Explorer - WindowsApplication1
Solution 'WindowsApplication1' (1 project)
WindowsApplication1
References
AssemblyInfo.vb
Form1.vb
Solution Explorer
Class View

- Solution Explorer Window provides an easy access to different application files including files contains forms and codes.

Properties Window -- Set Properties

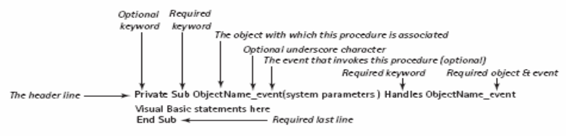


Cursor	Default
Font	Microsoft Sans Se
Name	Microsoft Sans Se
Size	7.8
Unit	Point
Bold	False
GdiCharSet	222
GdiVerticalFont	False
Italic	False
Strikeout	False
Underline	False
ForeColor	ControlText
FormBorderStyle	Sizable

- Once objects have been added to the form, the next step is setting the properties of the objects.
- The properties of objects are set through the Properties window or code (inside the program).
- Two important properties of objects are the Name property and the Text property.
- The Name property allows the programmer to assign a descriptive name to an object, rather than using the default name provided by Visual Basic for that object.
- The value of the Text property of an object is displayed to the user when the application is running.

Add Event Procedure

- An event procedure is a procedure or event handler executed when that event occurs.
- The first line of a procedure is a header line.
- A header line begins with the optional keyword Private and must contain the keyword Sub, the name of the procedure, and a set of parentheses.
- The last line of each procedure consists of the keywords End Sub.
- All statements from the header line to and including the End Sub statement are collectively referred to as the procedure’s body.
- The first and last lines of a procedure, consisting of the header line and the End Sub statement, are referred to as the procedure’s template.



Optional keyword
Required keyword
The object with which this procedure is associated
Optional underscore character
The event that invokes this procedure (optional)
Required keyword
Required object & event
The header line
Required last line

```
Private Sub ObjectName_event(system parameters) Handles ObjectName_event
Visual Basic statements here
End Sub
```

Figure 2–19 *The Structure of an Event Procedure*

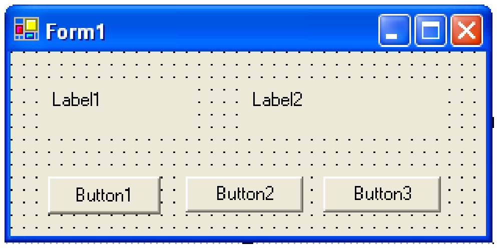
Test or Run Application

- You can run your program at any time during program development:
  - Select the Debug Menu and click Start or
  - Press the F5 function key or
  - Use the hot key sequence Alt+D, then press the S key
- Design time: when an application is being developed
- Run time: when a program is executing

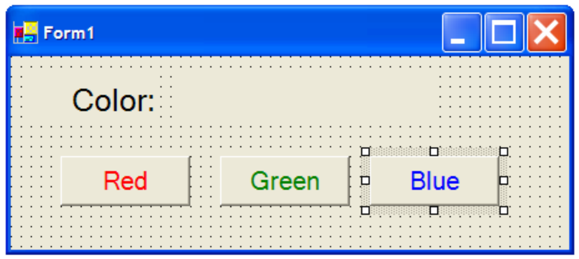
Saving and Recalling a Project

- To save an application
  - Click the File menu and then click Save All or
  - Click the SaveAll icon in the Standard Toolbar
- To retrieve a project:
  - Select Open Solution from the File menu

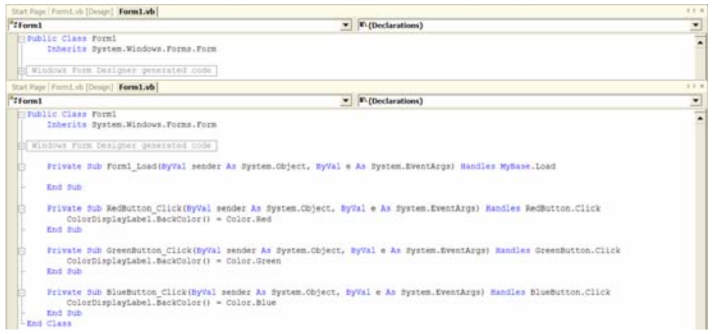
Example – Step 1: Adding Controls



Example -- Step 2: Adding Properties



Example -- Step 3: Adding Events



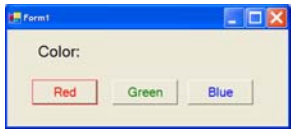
```
Start Page | Form1.vb [Design] | Form1.vb
Form1                                  (Declarations)
Public Class Form1
    Inherits System.Windows.Forms.Form

 Windows Form Designer generated code
```

```
Start Page | Form1.vb [Design] | Form1.vb
Form1                                  (Declarations)
Public Class Form1
    Inherits System.Windows.Forms.Form

 Windows Form Designer generated code

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

    End Sub

    Private Sub RedButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles RedButton.Click
        ColorDisplayLabel.BackColor() = Color.Red
    End Sub

    Private Sub GreenButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles GreenButton.Click
        ColorDisplayLabel.BackColor() = Color.Green
    End Sub

    Private Sub BlueButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BlueButton.Click
        ColorDisplayLabel.BackColor() = Color.Blue
    End Sub
End Class
```

Running an Application

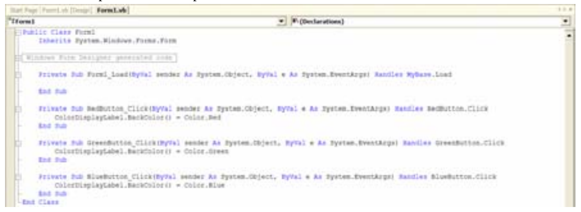
A project being developed can be executed by using any one of the following three methods:

- Select the Debug Menu and click Start.
- Press the F5 function key.
- Use the hot key sequence Alt+D, then press the S key.



Adding an Event Procedure

- A procedure that is executed when an event occurs is referred to as an event procedure or event handler.
- The first line of a procedure is always a header line. A header line begins with the optional keyword Private and must contain the keyword Sub, the name of the procedure, and a set of parentheses.
- The last line of each procedure consists of the keywords End Sub.
- All statements from the header line to and including the End Sub statement are collectively referred to as the procedure's body.
- The first and last lines of a procedure, consisting of the header line and the End Sub statement, are referred to as the procedure's template.



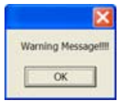
```
Start Page | Form1.vb [Design] | Form1.vb
Form1                                  (Declarations)
Public Class Form1
    Inherits System.Windows.Forms.Form

 Windows Form Designer generated code

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

    End Sub

    Private Sub RedButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles RedButton.Click
        ColorDisplayLabel.BackColor() = Color.Red
    End Sub

    Private Sub GreenButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles GreenButton.Click
        ColorDisplayLabel.BackColor() = Color.Green
    End Sub

    Private Sub BlueButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BlueButton.Click
        ColorDisplayLabel.BackColor() = Color.Blue
    End Sub
End Class
```

The `MessageBox.Show` Method

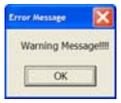
- The **`MessageBox.Show`** method can be used in a procedure to display a message to the user through a message box.
- The message box also contains a title and an icon.
- The general forms of the **`MessageBox.Show`** method are:

```
MessageBox.Show(text)
MessageBox.Show(text, caption)
MessageBox.Show(text, caption, buttons)
MessageBox.Show(text, caption, buttons, icon)
MessageBox.Show(text, caption, buttons, icon,
  defaultbutton)
```

Message Box with Caption

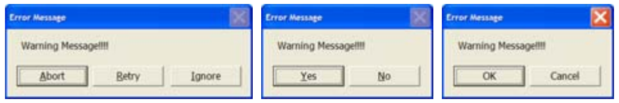


```
MessageBox.Show("Warning Message!!!!")
```



```
MessageBox.Show("Warning Message!!!!", "Error Message")
```

Message Box with Buttons

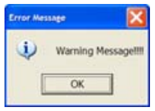


```
MessageBox.Show("Warning Message!!!!", "Error Message",
   MessageBoxButtons.AbortRetryIgnore)
MessageBox.Show("Warning Message!!!!", "Error Message",
   MessageBoxButtons.YesNo)
MessageBox.Show("Warning Message!!!!", "Error Message",
   MessageBoxButtons.OKCancel)
```

Other values

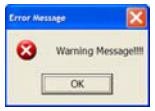
- `MessageBoxButtons.OK`
- `MessageBoxButtons.RetryCancel`
- `MessageBoxButtons.YesNoCancel`

Message Box with Icons



The **information** (`Asterisk` and `Information`) icon should be used when a message displayed with an OK button.

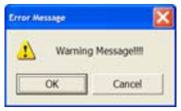
```
MessageBox.Show("Warning Message!!!!", "Error
   Message", MessageBoxButtons.OK,
   MessageBoxIcon.Asterisk)
MessageBox.Show("Warning Message!!!!", "Error
   Message", MessageBoxButtons.OK,
   MessageBoxIcon.Information)
```



The **stop** (`Error, Hand` and `Stop`) icon should be used with an OK button when an error or problem occurs and needs to be resolved.

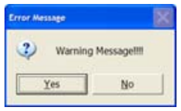
```
MessageBox.Show("Warning Message!!!!", "Error
   Message", MessageBoxButtons.OK,
   MessageBoxIcon.Error)
MessageBox.Show("Warning Message!!!!", "Error
   Message", MessageBoxButtons.OK,
   MessageBoxIcon.Hand)
MessageBox.Show("Warning Message!!!!", "Error
   Message", MessageBoxButtons.YesNo,
   MessageBoxIcon.Stop)
```

Message Box with Icons



The **exclamation** (`Exclamation` and `Warning`) icon should be used when the user must make a decision before the program can continue.

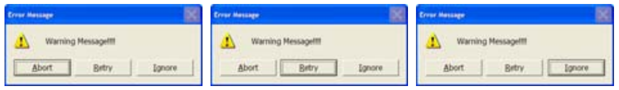
```
MessageBox.Show("Warning Message!!!!", "Error
   Message", MessageBoxButtons.OKCancel,
   MessageBoxIcon.Exclamation)
MessageBox.Show("Warning Message!!!!", "Error
   Message", MessageBoxButtons.OKCancel,
   MessageBoxIcon.Warning)
```



The **question** (**Question**) icon should be used when a question needs to be answered.

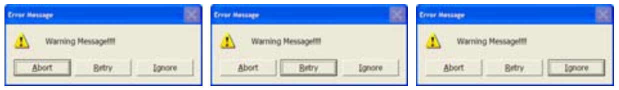
```
MessageBox.Show("Warning Message!!!!", "Error
   Message", MessageBoxButtons.YesNo,
   MessageBoxIcon.Question)
```

Message Box with Default Button



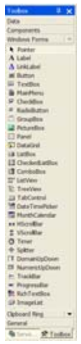
```
MessageBox.Show("Warning Message!!!!", "Error Message",
    MessageBoxButtons.AbortRetryIgnore, MessageBoxIcon.Exclamation,
    MessageBoxDefaultButton.Button1)
MessageBox.Show("Warning Message!!!!", "Error Message",
    MessageBoxButtons.AbortRetryIgnore, MessageBoxIcon.Exclamation,
    MessageBoxDefaultButton.Button2)
MessageBox.Show("Warning Message!!!!", "Error Message",
    MessageBoxButtons.AbortRetryIgnore, MessageBoxIcon.Exclamation,
    MessageBoxDefaultButton.Button3)
```

Message Box with Default Button



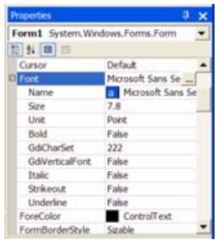
- **`MessageBoxDefaultButton.Button1`**- - specifies the leftmost button and is the default.
- **`MessageBoxDefaultButton.Button2`**- - specifies the second button from the left.
- **`MessageBoxDefaultButton.Button3`**- - specifies the third button from the left.
- When the button is clicked, the value returned is one of the following:
  - **`DialogResult.Abort`**
  - **`DialogResult.Cancel`**
  - **`DialogResult.Ignore`**
  - **`DialogResult.No`**
  - **`DialogResult.OK`**
  - **`DialogResult.Retry`**
  - **`DialogResult.Yes`**

Adding Controls



- Objects placed on a form are called controls.
- These objects can be added from the Toolbox.
- The simplest method of adding a control to a form is to double-click the desired object in the Toolbox.

Setting the Initial Object Properties

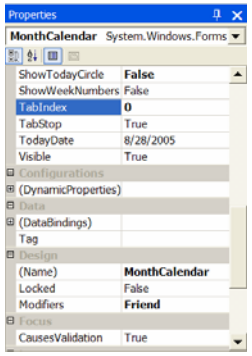


Cursor	Default
Font	Microsoft Sans Se
Name	Microsoft Sans Se
Size	7.8
Unit	Point
Bold	False
GdiCharSet	222
GdiVerticalFont	False
Italic	False
Strikeout	False
Underline	False
ForeColor	ControlText
FormBorderStyle	Sizable

Once controls have been added to a form, we can use the Properties window to change one or more of the properties of the controls.

Looking at the Focus and Tab Sequence

- When an application is run and a user is looking at the form, only one of the form's controls will have *input focus*, or *focus*.
- Only objects which are capable of responding to user input through either the keyboard or mouse can receive focus.
- In order to receive the focus, a control must have its *Enabled*, *Visible*, and TabStop properties set to True.
- The sequence in which the focus shifts from control to control as the tab key is pressed by the user is called the *tab sequence*.
- The programmer can alter the default tab order – which is obtained from the sequence in which controls are placed on the form – by modifying an object's TabIndex value.



MonthCalendar System.Windows.Forms

ShowTodayCircle	**False**
ShowWeekNumbers	False
TabIndex	**0**
TabStop	True
TodayDate	8/28/2005
Visible	True
Configurations	
(DynamicProperties)	
Data	
(DataBindings)	
Tag	
Design	
(Name)	**MonthCalendar**
Locked	False
Modifiers	**Friend**
Focus	
CausesValidation	True

Adding Additional Event Procedures

- Once we have added objects to a form and set the properties of these controls, the next step in creating a Visual Basic application is to supply these objects with event code.
- Each object can have many events associated with it.
- To enter the event code for an object, we can double-click the object to open its Code window.

Comments

- *Comments* are explanatory remarks made within a program.
- Comments are written using an apostrophe or the keyword **`Rem`**.
- Comments are ignored by Visual Basic and do not impact the execution of a program.

Statement

- All *statements* belong to one of two categories of statements:
  – executable statements
  – nonexecutable statements.
- An *executable* statement causes some specific action to be performed by the compiler or interpreter.
- A *nonexecutable* statement describes some feature of either the program or its data but does not cause the computer to perform any action.

Help Facility

- Visual Basic's Help Facility can be accessed by selecting either the Contents, Search, or Index options from the Help menu
- The Contents tab displays a Table of Contents for the documentation
- The Index tab provides both a general index of topics and a text box for user entry of a specific topic
- The Search tab provides a means of entering a search word or phrase

Help Facility

- Dynamic Help
  - The Dynamic Help window displays a list of help topics that changes as you perform operations
  - To open the Dynamic Help window, click Help on the menu bar and then click Dynamic Help
- Context-sensitive Help
  - Context-sensitive Help immediately displays a relevant article for a topic
  - To use this facility, select an object and press F1