

Module 2: Development Environment Features

Overview

- Describing the Integrated Development Environment
- Creating Visual Basic .NET Projects
- Using Development Environment Features
- Debugging Applications
- Compiling in Visual Basic .NET

Describing the Integrated Development Environment

- There is one IDE for all .NET projects
- Solutions can contain multiple programming languages
 - Example: Visual Basic .NET and C# in the same solution
- The IDE is customizable through “My Profile”
- The IDE has a built-in Internet browser

◆ Creating Visual Basic .NET Projects

- Choosing a Project Template
- Analyzing Project Structures
- What Are Assemblies?
- Setting Project References
- What Are Namespaces?
- Creating Namespaces
- Importing Namespaces
- Setting Project Properties

Choosing a Project Template

- Windows Application
- Class Library
- Windows Control Library
- ASP .NET Web Application / Service / Control Library
- Console Application
- Windows Service
- Others

Analyzing Project Structures

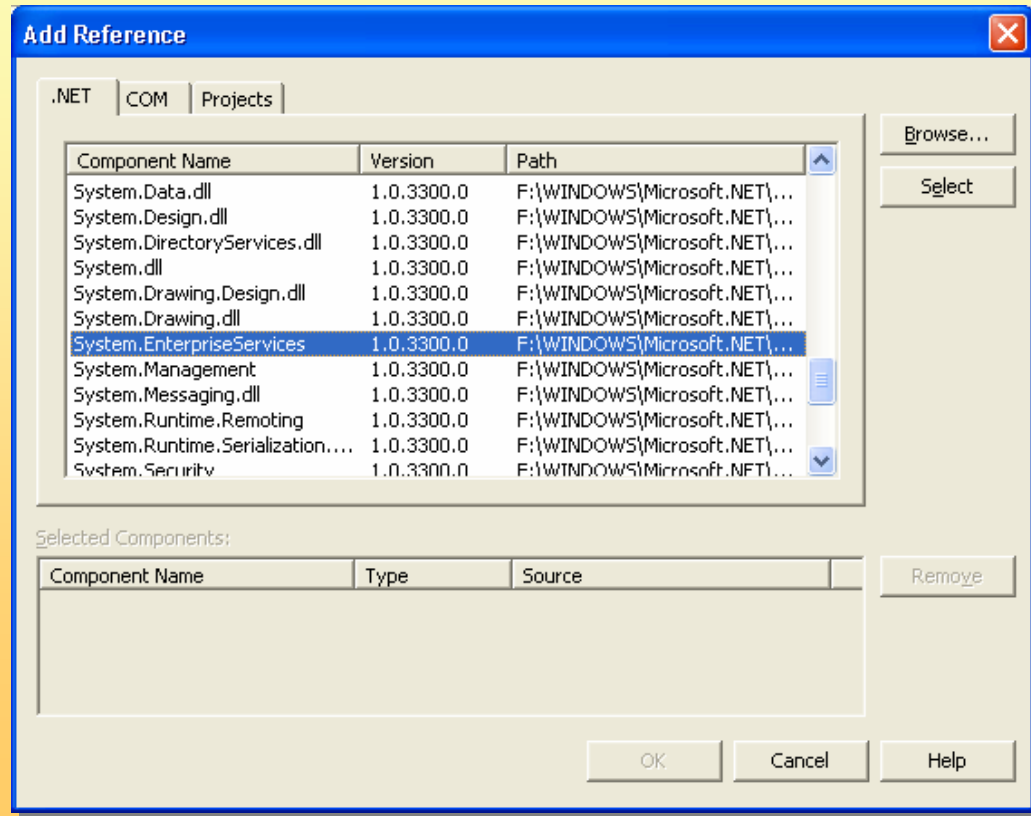
- Solution files (.sln, .suo)
- Project files (.vbproj)
- Local project items
 - Classes, forms, modules, etc. (.vb)
- Web project items
 - XML Web services (.asmx)
 - Web forms (.aspx)
 - Global application classes (.asax)

What Are Assemblies?

- An assembly is an .exe or .dll file with other supporting files that make up a Visual Studio .NET application
- The .NET Framework provides predefined assemblies
- Assemblies are created automatically when you compile source files
 - Click **Build** on the Build menu
 - Use the command-line command `vbc.exe`

Setting Project References

- .NET assemblies
- COM components
- Projects



What Are Namespaces?

- Namespaces organize objects defined in an assembly
 - Group logically related objects together
- Namespaces create fully qualified names for objects
 - Prevent ambiguity
 - Prevent naming conflicts in classes

Creating Namespaces

- Use Namespace ... End Namespace syntax
- Use the root namespace defined in Assembly Properties

```
Namespace Top           'Fully qualified as MyAssembly.Top
  Public Class Inside   'Fully qualified as MyAssembly.Top.Inside
    ...
  End Class
Namespace InsideTop     'Fully qualified as MyAssembly.Top.InsideTop
  Public Class Inside   'Fully qualified as MyAssembly.Top.InsideTop.Inside
    ...
  End Class
End Namespace
End Namespace
```

Importing Namespaces

- Fully qualified names can make code hard to read

```
Dim x as MyAssembly.Top.InsideTop.Inside
```

- Using the Imports statement results in simpler code by providing scope

```
Imports MyAssembly.Top.InsideTop  
...  
Dim x as Inside
```

- Import aliases create aliases for a namespace or type

```
Imports IT = MyAssembly.Top.InsideTop  
...  
Dim x as IT.Inside
```

Setting Project Properties

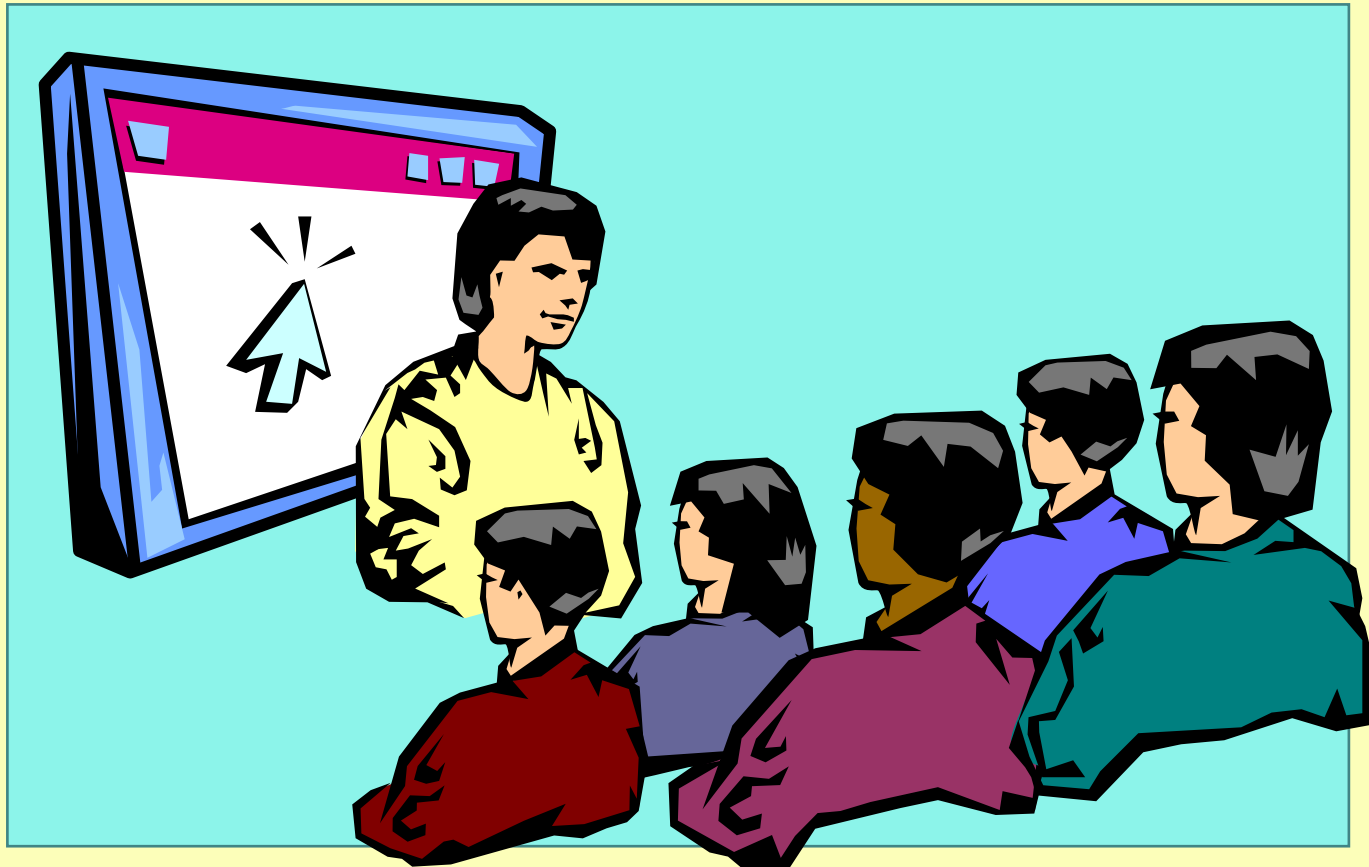
■ Common Property Settings

- Defining assembly name
- Root namespace
- Project output type
- Startup object
- Importing project-level namespaces

■ Configuration Property Settings

- Debugging settings
- Build options

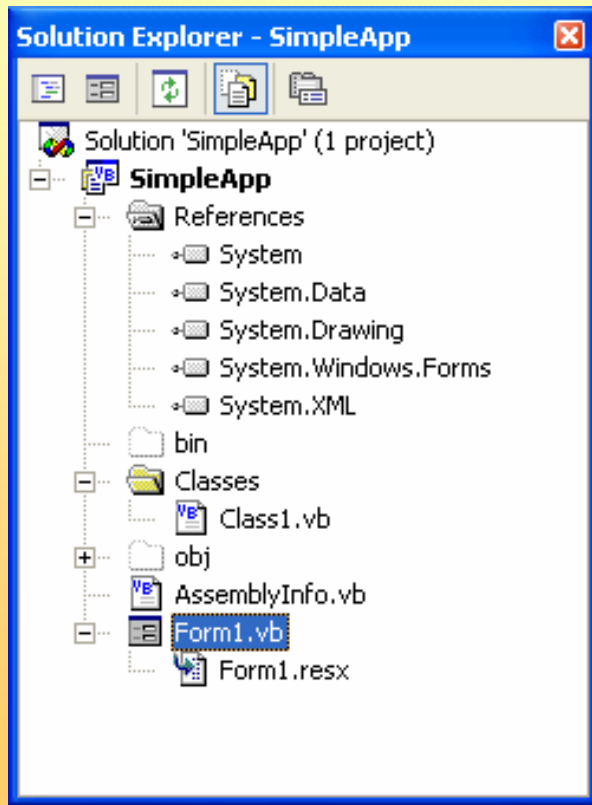
Demonstration: Creating a Visual Basic .NET Project



◆ Using Development Environment Features

- Using Solution Explorer
- Using Server Explorer
- Using the Object Browser
- Using the Task List
- Using Dynamic Help
- Using XML Features
- Recording and Using Macros

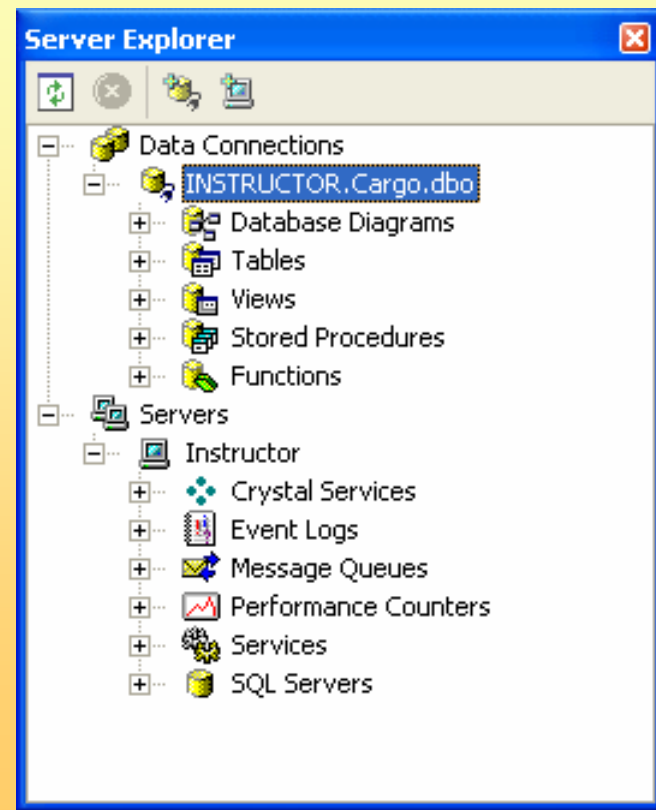
Using Solution Explorer



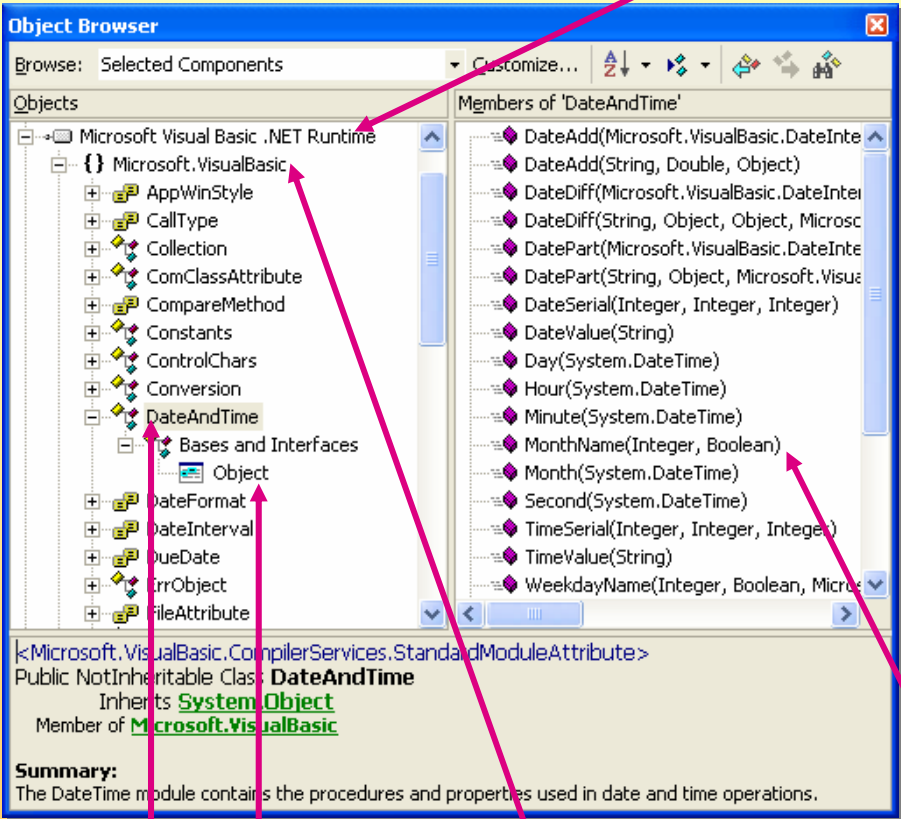
- Displays project hierarchy
 - Project references
 - Forms, classes, modules
 - Folders with subitems
- “Show All Files” mode
- Manipulating projects
 - Drag-and-drop editing
 - Context menus

Using Server Explorer

- Managing Data Connections
- Viewing and Managing Servers
- Using Drag-and-Drop Techniques



Using the Object Browser



The screenshot shows the Object Browser window with the following structure:

- Objects:** Microsoft Visual Basic .NET Runtime > Microsoft.VisualBasic > DateAndTime
- Members of 'DateAndTime':** DateAdd, DateDiff, DatePart, DateSerial, DateValue, Day, Hour, Minute, MonthName, Month, Second, TimeSerial, TimeValue, WeekdayName
- Summary:** Public NotInheritable Class **DateAndTime**
Inherits **System.Object**
Member of **Microsoft.VisualBasic**

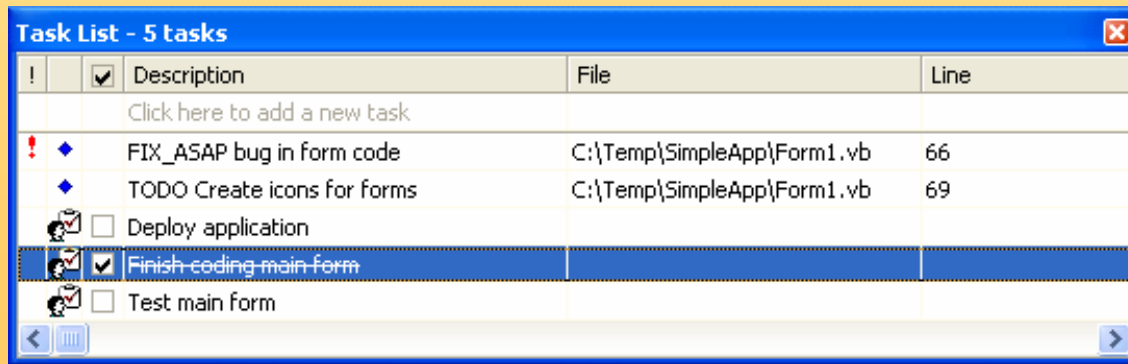
Annotations in the image:

- Library:** Points to the Microsoft.VisualBasic namespace.
- Class:** Points to the DateAndTime class.
- Inheritance:** Points to the Inherits System.Object line.
- Namespace:** Points to the Microsoft.VisualBasic namespace.
- Method:** Points to a specific member in the DateAndTime class.

- Examine objects and their members
- Access lower-level items
 - Shows inheritance and interfaces
- Examine how the .NET Framework class libraries use Inheritance

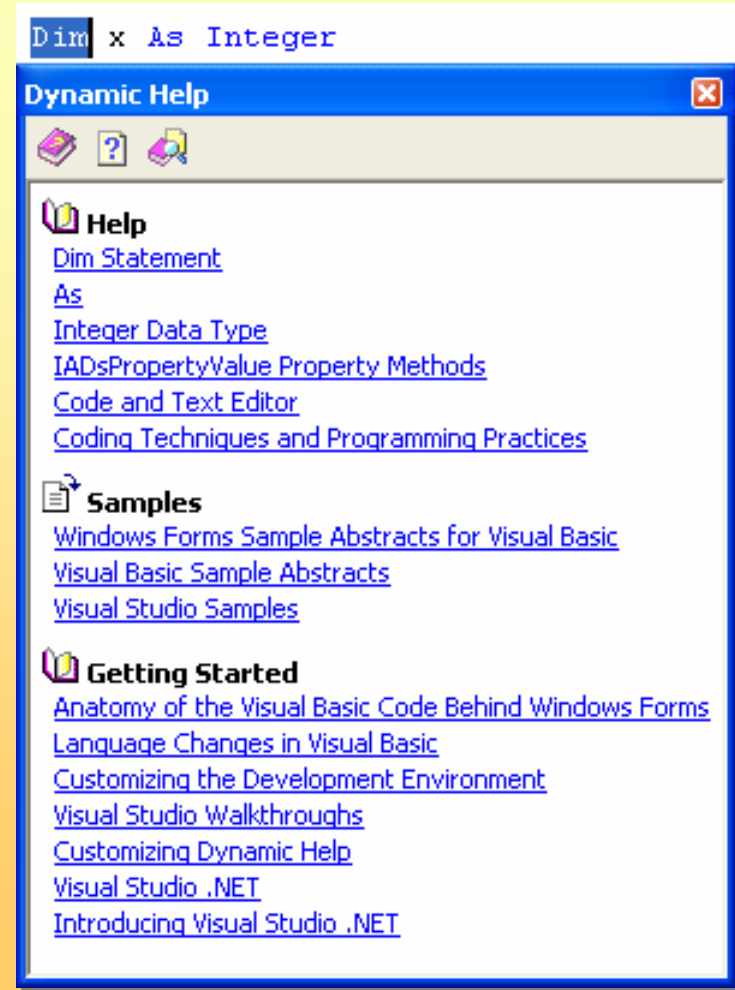
Using the Task List

- Similar to the Tasks feature in Microsoft Outlook
- Stored with the solution in the .suo file
- Adding to the Task List
 - You can add tasks manually by typing in appropriate field
 - Visual Basic .NET adds build errors, upgrade comments, etc.
 - You can use token strings to add comments in code



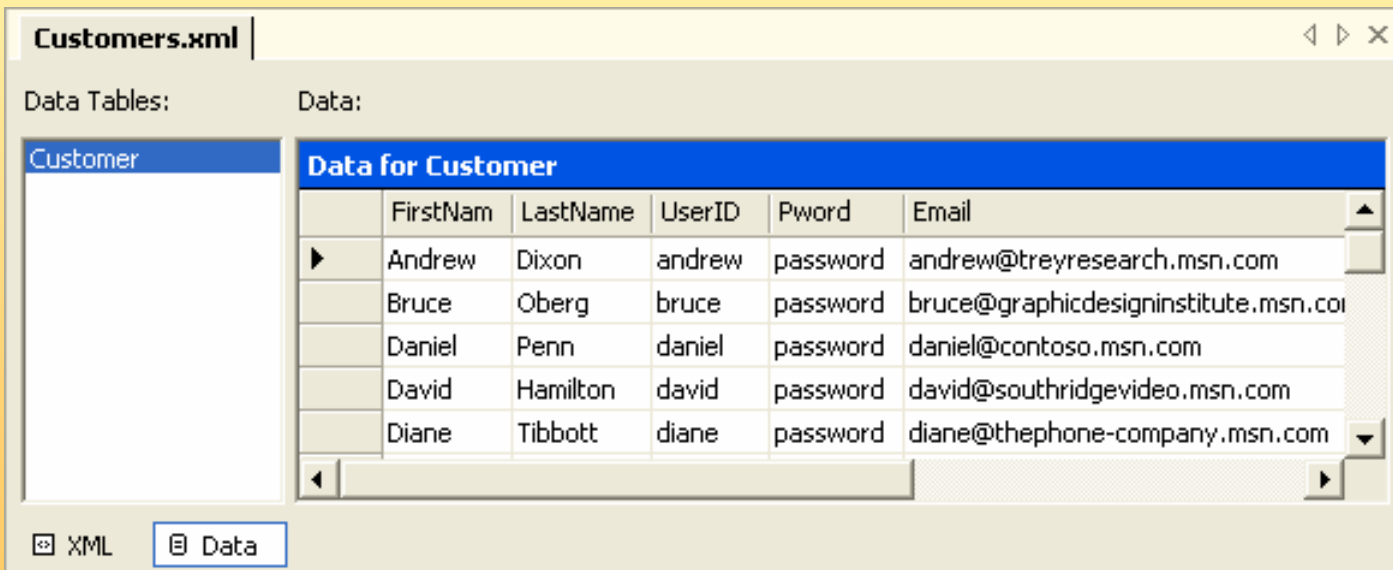
Using Dynamic Help

- Automatically displays relevant Help topics based on focus and cursor placement
- Use the Options dialog box to configure the Dynamic Help window

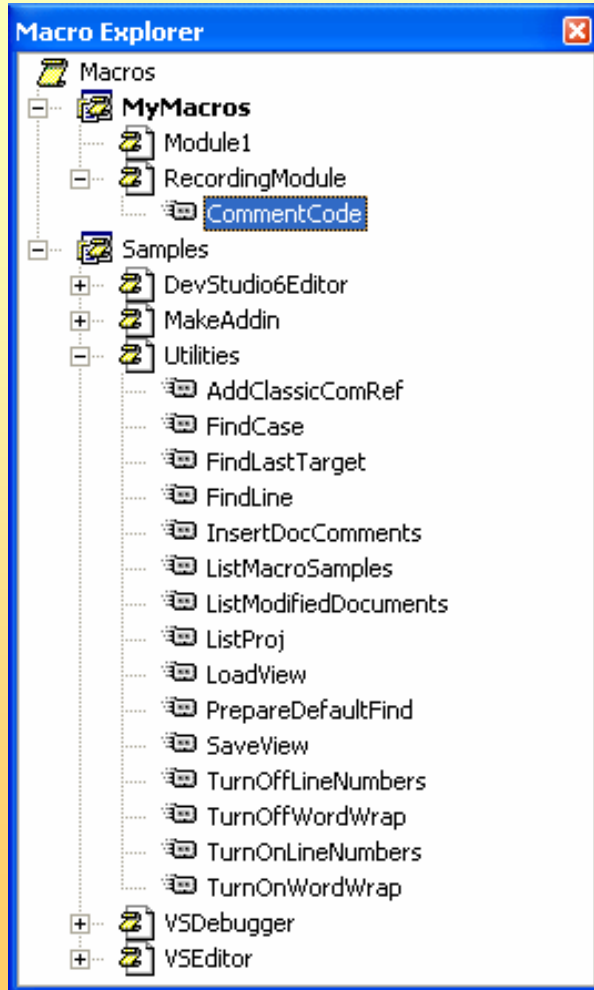


Using XML Features

- HTML and XML Document Outline window
- AutoComplete
- Color-coding
- Data View for manipulating data

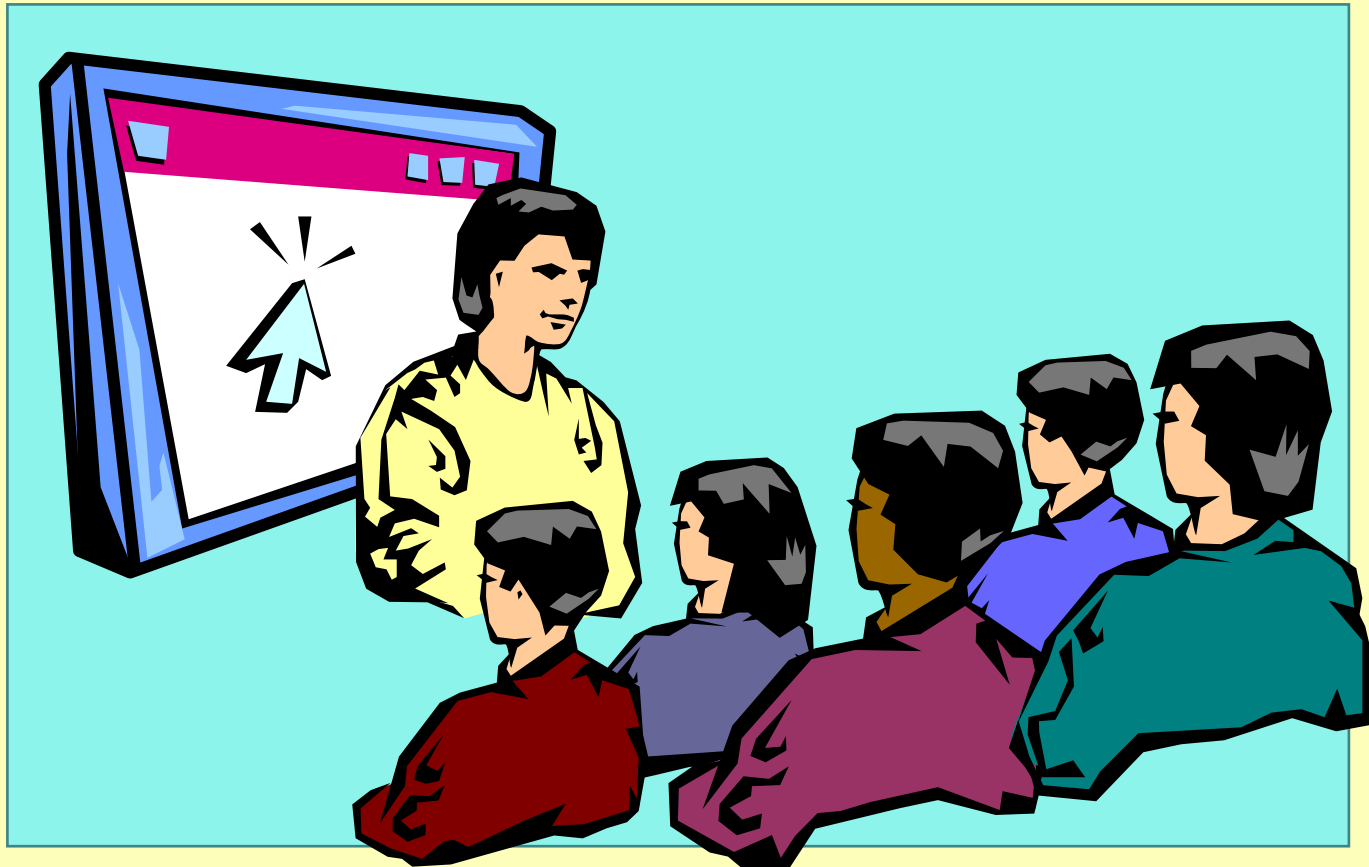


Recording and Using Macros



- You can use macros for repetitive tasks such as inserting comments
- Macro Explorer provides macro navigation
- The IDE provides samples:
 - Toggle line numbering
 - Saving/loading Window Views
 - Debugging macros
- To record new macros, go to the Tools/Macros menu

Demonstration: Using the Visual Studio .NET IDE

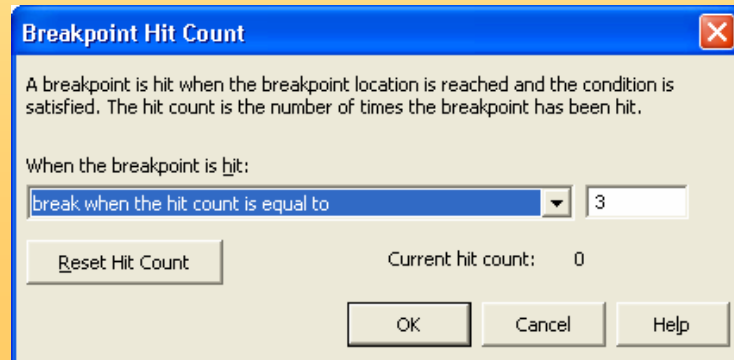
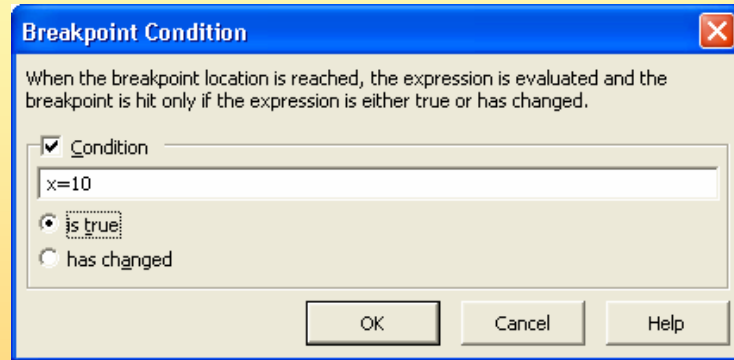
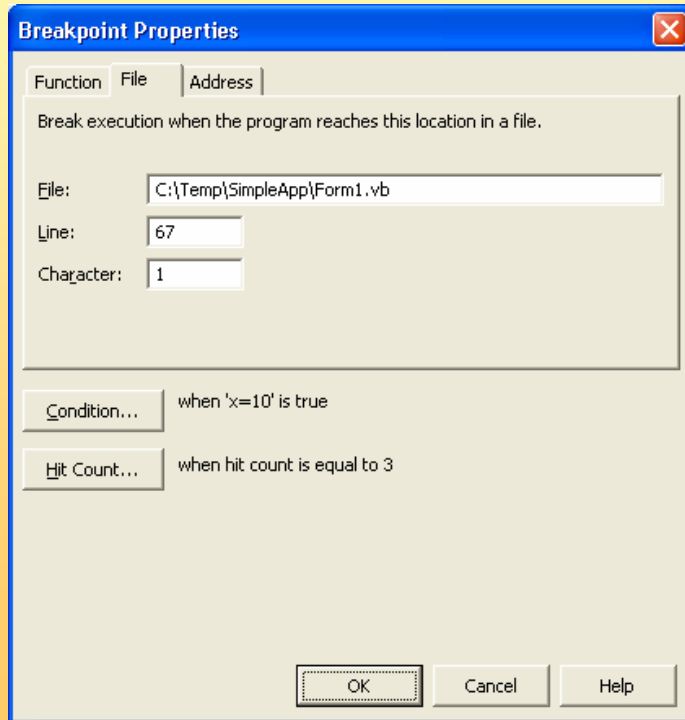


◆ Debugging Applications

- Setting Breakpoints
- Debugging Code
- Using the Command Window

Setting Breakpoints

- Set breakpoints to halt code execution at a specific line
- Use the Breakpoint Properties dialog box to set conditions

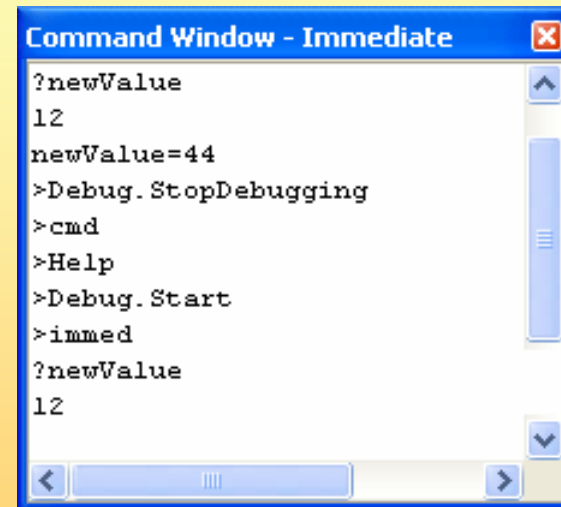


Debugging Code

- Use the Debug menu or toolbar to step through code
- Use the debugging windows:
 - Locals: to view and modify local variables
 - Output: to view output from the compiler
 - Watch: to view watch expressions
 - Call Stack: to view call history, including parameter information
 - Breakpoints: to view, add, or temporarily disable breakpoints

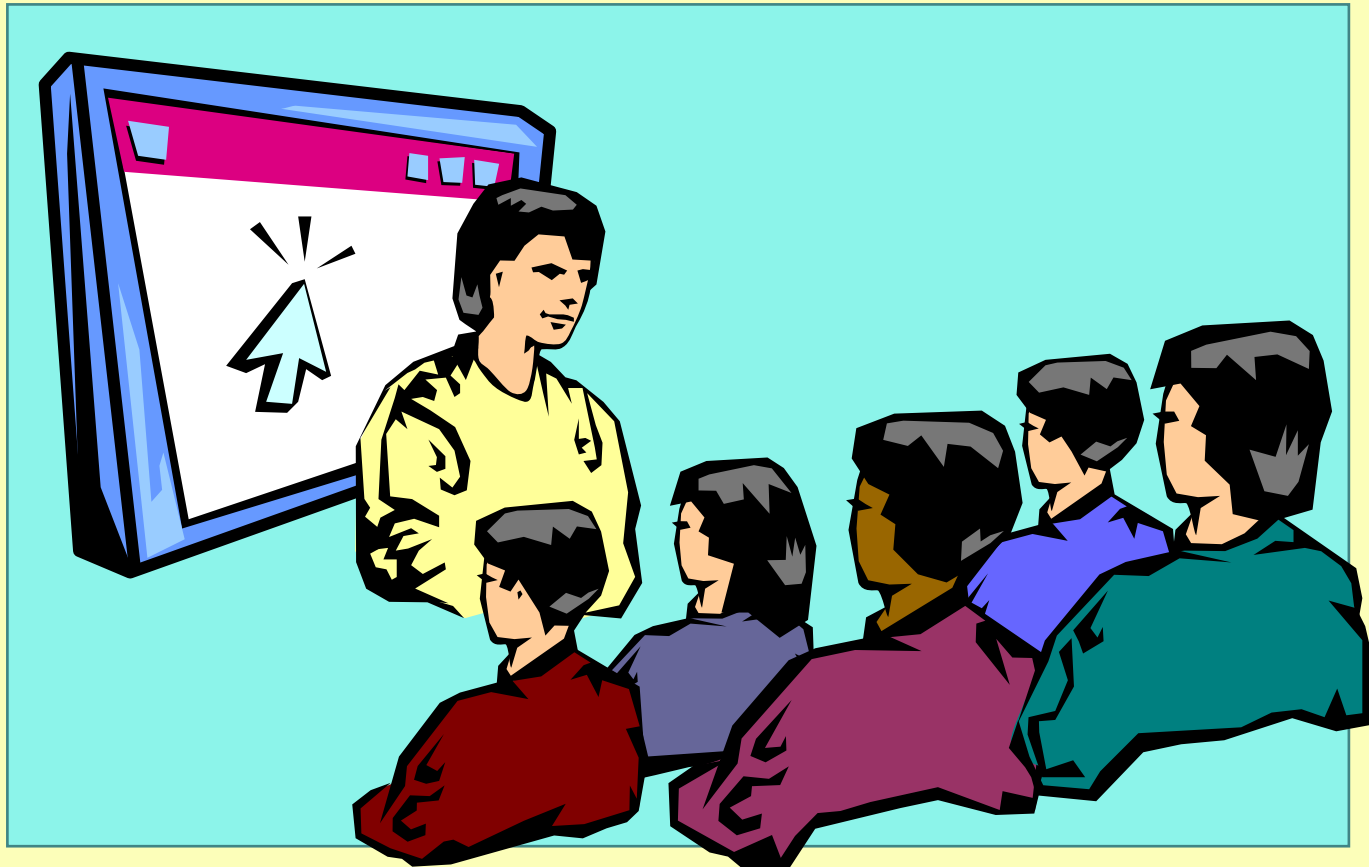
Using the Command Window

- Immediate mode
 - Similar to the Immediate window
- Command mode
 - Use Visual Studio IDE features
- Switching modes
 - Use **>cmd** to change to Command mode
 - Use **immed** to return to Immediate mode



```
Command Window - Immediate
?newValue
12
newValue=44
>Debug.StopDebugging
>cmd
>Help
>Debug.Start
>immed
?newValue
12
```

Demonstration: Debugging a Project



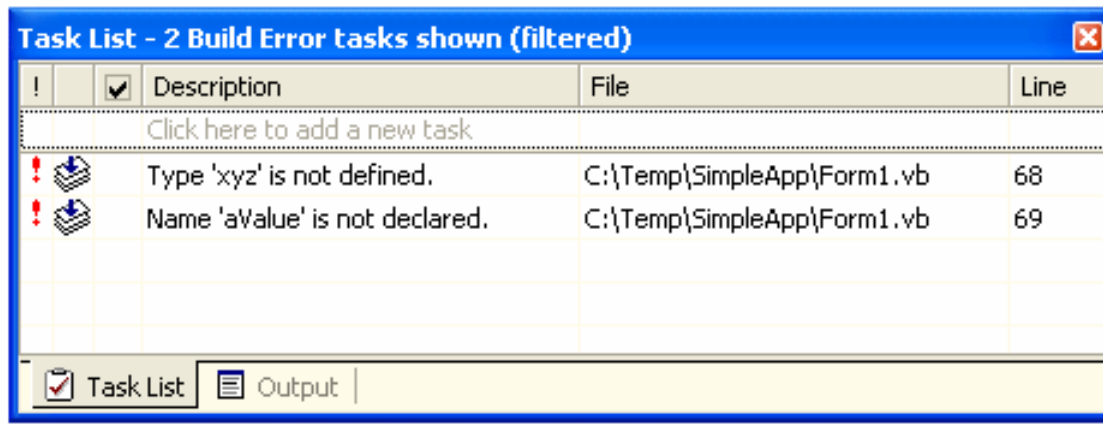
◆ Compiling in Visual Basic .NET

- Locating Syntax Errors
- Compilation Options

Locating Syntax Errors

- The Task List displays compilation errors
 - Displays error description, file, and line number
- Double-click the entry to view the error

```
Dim newValue As xyz  
aValue.DoIt ()
```



The screenshot shows a 'Task List' window with the title 'Task List - 2 Build Error tasks shown (filtered)'. The window contains a table with the following data:

!	Description	File	Line
	Click here to add a new task		
! [icon]	Type 'xyz' is not defined.	C:\Temp\SimpleApp\Form1.vb	68
! [icon]	Name 'aValue' is not declared.	C:\Temp\SimpleApp\Form1.vb	69

At the bottom of the window, there are two tabs: 'Task List' (selected) and 'Output'.

Compilation Options

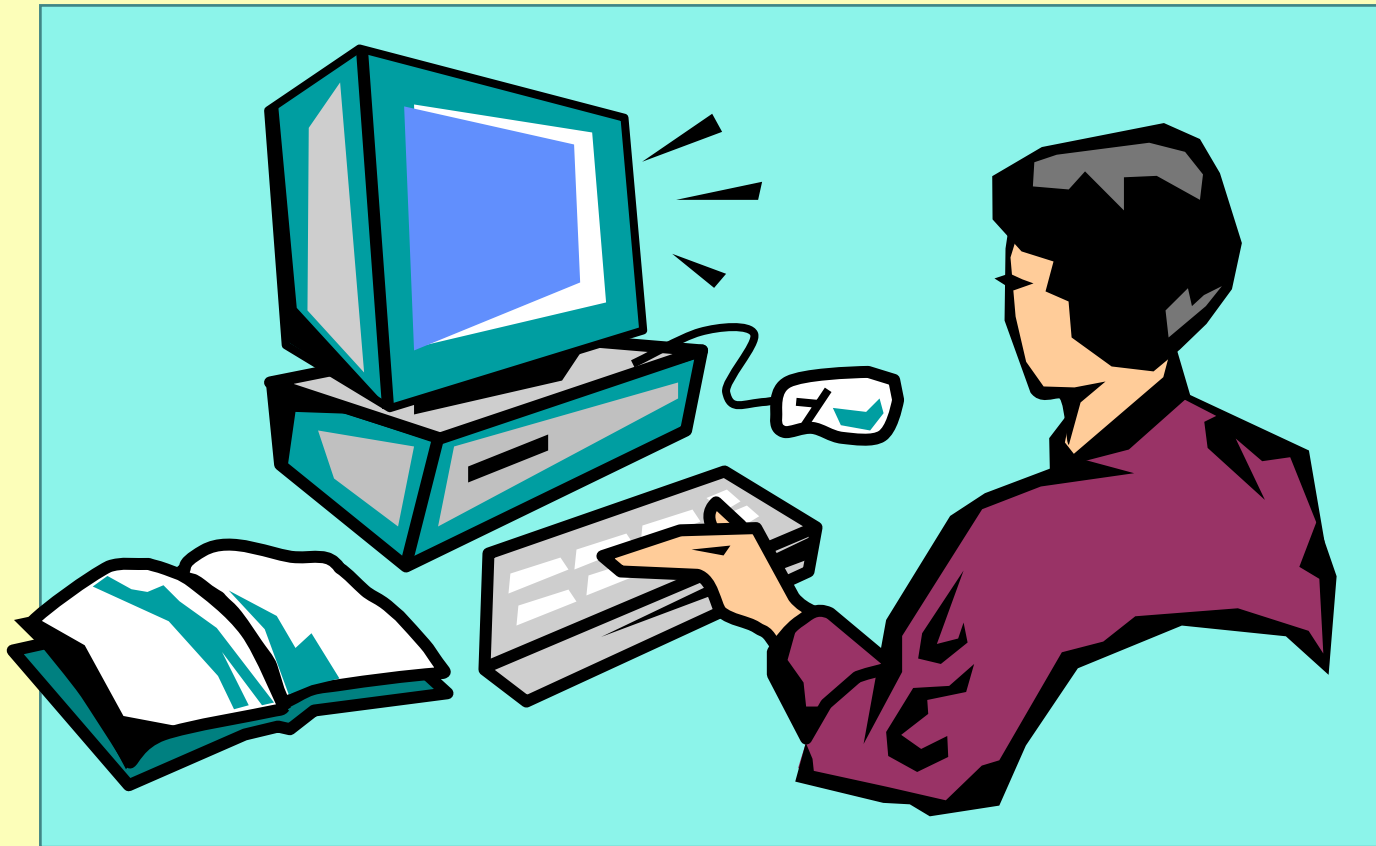
■ Build configurations

- Debug – provides debug information
- Release – optimizes code and executable size

■ Build options

- Build – only builds changed projects
- Rebuild – rebuilds project regardless of changes

Lab 2.1: Exploring the Development Environment



Review

- Describing the Integrated Development Environment
- Creating Visual Basic .NET Projects
- Using Development Environment Features
- Debugging Applications
- Compiling in Visual Basic .NET