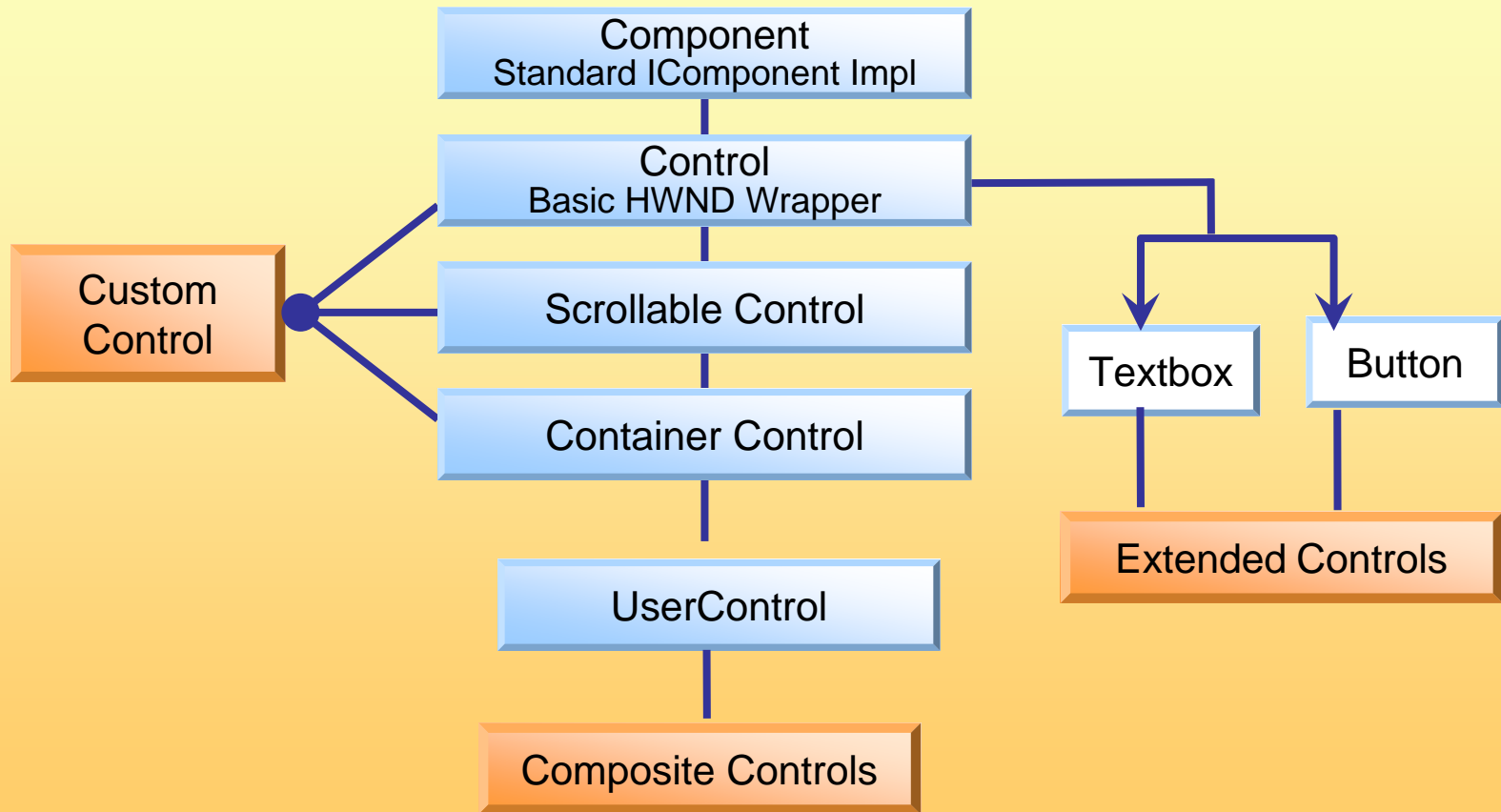


Module 3: Building Controls

Overview

- Extending and Creating Controls
- Adding Design-Time Support for Controls
- Licensing a Control

Lesson: Extending and Creating Controls



Options for Building Controls

- Extended controls

```
Public Class NumericTextBox
    Inherits System.Windows.Forms.TextBox

    End Sub
End Class
```

- Composite controls

- Controls that are composed of other controls
- Designed to act as containers for other controls

- Custom controls

[CodeExample](#)

How to Expose and Override Properties for Controls

- Exposing properties of a control within a container

```
Public Property QuantityTextBoxContextMenu() As  
    ContextMenu  
    Get  
        Return txtQuantity.ContextMenu  
    End Get  
    Set(ByVal Value As ContextMenu)  
        txtQuantity.ContextMenu = Value  
    End Set  
End Property
```

- Overriding properties

[CodeExample](#)

How to Raise and Override Events for Controls

- To raise events for a composite control

```
Public Event AddUserControl()  
...  
If e.KeyCode = Keys.Enter Then  
    RaiseEvent AddUserControl()  
End If
```

- To override events

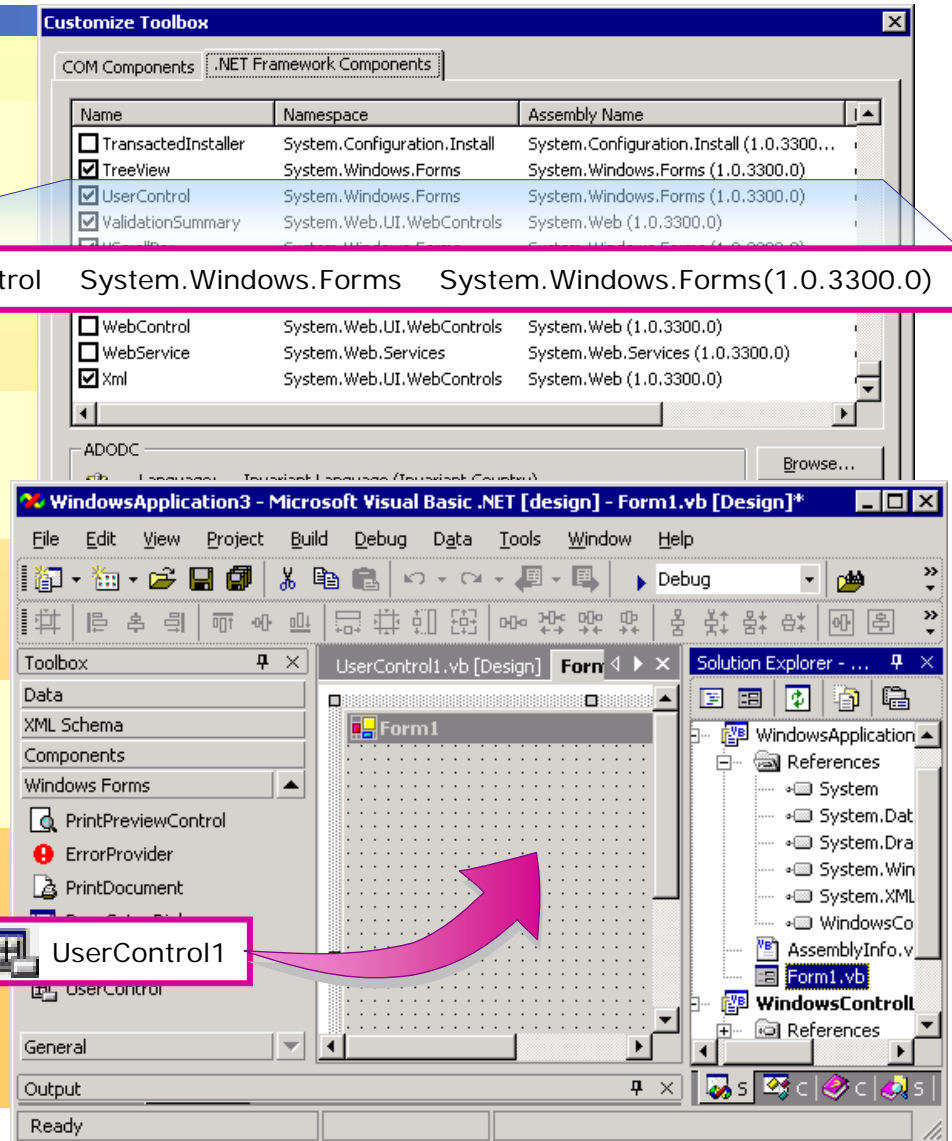
```
Public Class FirstControl  
    Inherits Control  
...  
Protected Overrides Sub OnPaint(e As  
    PaintEventArgs)
```

How to Test a Control

1 Create a new project within the same solution

2 Add UserControl to the Toolbox

3 Add UserControl to the form from the Toolbox



Code Walkthrough: Creating Controls



In this walkthrough, you will see the code that is involved in creating different controls

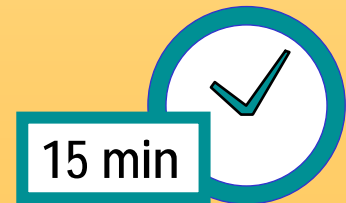
Practice: Creating a Composite Control



In this practice, you will

- Create a Windows Control Library Project and add a UserControl
- Add controls to the UserControl
- Expose properties
- Raise events
- Test the design-time instance
- Test the run-time instance

Begin reviewing the objectives
for this practice activity



Lesson: Adding Design-Time Support for Controls

- Property Attributes
- How to Add Attributes That Provide Information to the Visual Designer
- Design-Time Support Options Built into the .NET Framework
- Practice: Adding Design-Time Support for Controls

Property Attributes

■ Property Attributes

- Allow you to specify the behavior of properties at design time

■ Property attributes allow you to specify

- Grouping options for custom properties in the Properties window of the Visual Studio .NET environment
- Default values
- Custom editors for custom properties

■ Examples of Property Attributes

- Browsable
- Category
- TypeConverter
- Description
- DefaultProperty
- Editor

How to Add Attributes That Provide Information to the Visual Designer

To add attributes to your code

- 1 Define a new attribute or use an existing attribute by importing its namespace from the .NET Framework class library

```
Imports System.ComponentModel
```

- 2 Initialize the attribute directly preceding the element to be described

```
<Category("Appearance")>Public  
    BorderColor As Color
```

Design-Time Support Options Built into the .NET Framework

- **Property Browser**
 - Associates property editors for different data types
- **Type Converter**
 - Converts the value entered into the Properties window to the correct data type within a control
- **Custom UI Editors**
 - Chooses the correct UI editor for the Properties window based on the type of property
- **Custom Designers**
 - Allows you to modify the design-time appearance and behavior of controls and objects
- **Extenders**
 - Provide the ability to add or filter properties for controls

Lesson: Licensing a Control

- Files in Licensing
- How to Enable Licensing for a Control
- Demonstration: Creating and Validating a License for a Control
- How LicFileLicenseProvider Works in .NET

Files in Licensing

■ LIC file

- Design-time license file that exists anywhere that the LicenseProvider class specifies

```
"Namespace.ClassName is a licensed component"
```

■ LICX file

- Design-time license that resides with the assembly that consumes a licensed component

```
"Namespace.ClassName, ClassName, Version,  
Culture, PublicKeyToken"
```

■ .Licenses file

- Binary run-time license file used at run time

How to Enable Licensing for a Control

To enable licensing for a control

- 1** Import the **System.ComponentModel** namespace
- 2** Append the **LicenseProvider** attribute to the declaration of the class you want to license
- 3** Declare a **License** object
- 4** Call the **Validate** or **IsValid** methods of the **LicenseManager** to fill the License object with a valid license
- 5** Override the **Dispose** method of the licensed class and explicitly call the **Dispose** method on the **License** object
- 6** Create a text file with a **.LIC** extension and the correct format and save it to the assembly folder

Demonstration: Creating and Validating a License for a Control



In this demonstration, you will see how to create and validate a license for a control

How LicFileLicenseProvider Works in .NET

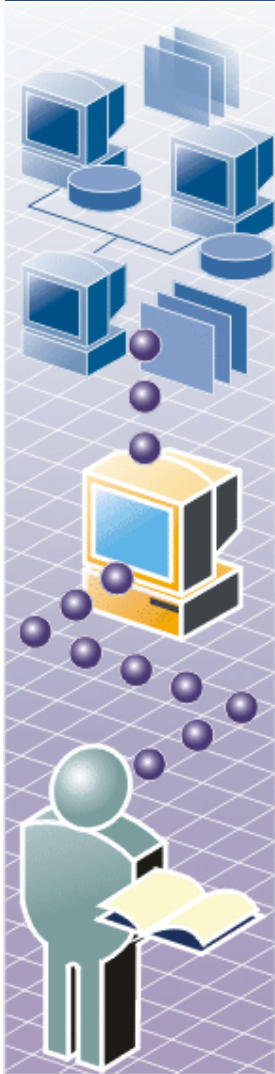
When you attempt to add a licensed component in your application

- 1** The `LicenseManager` attempts to locate a valid `.LIC` file
- 2** If the `LicenseManager` finds a suitable `.LIC` file, it fills the `License` object with a valid license
- 3** Visual Studio .NET generates a `.LICX` file in the host application
- 4** When you build the application and run it, the `LC.exe` utility (license compiler) looks at the `.LICX` file for a list of licensed classes, instantiates these classes, extracts the runtime license key, and embeds the collection of runtime keys in a binary `.Licenses` file

Review

- Extending and Creating Controls
- Adding Design-Time Support for Controls
- Licensing a Control

Lab 3.1: Building Controls



- Exercise 1: Defining an Event and Raising It from an Extended Control
- Exercise 2: Creating a Composite Control
- Exercise 3: Adding Design-Time Support