

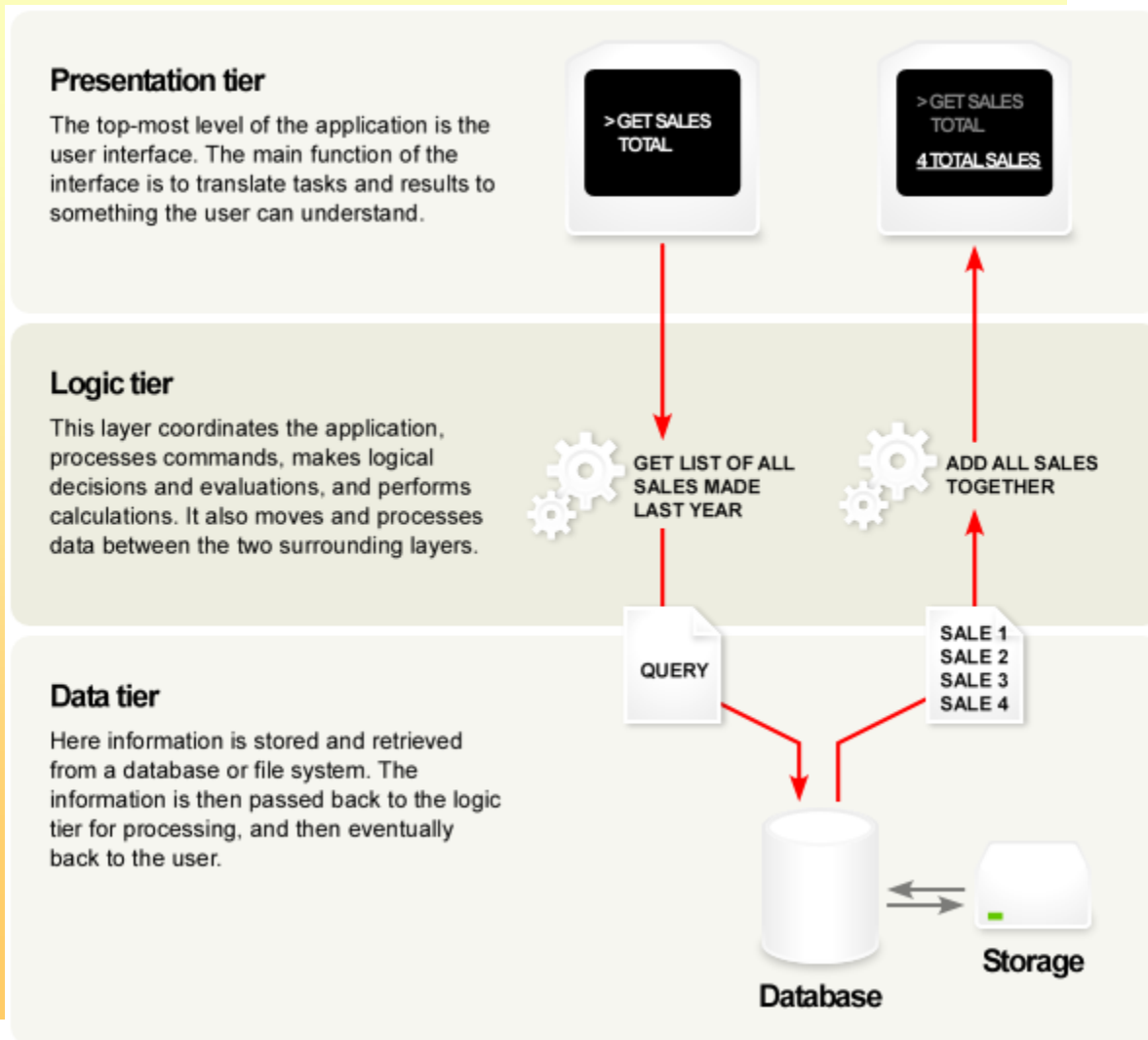
Windows Forms(1)

Programiranje korisničkih interfejsa

Bojan Furlan

System.Windows.Forms

- podržava 3-slojnu arhitekturu
- OO programski model

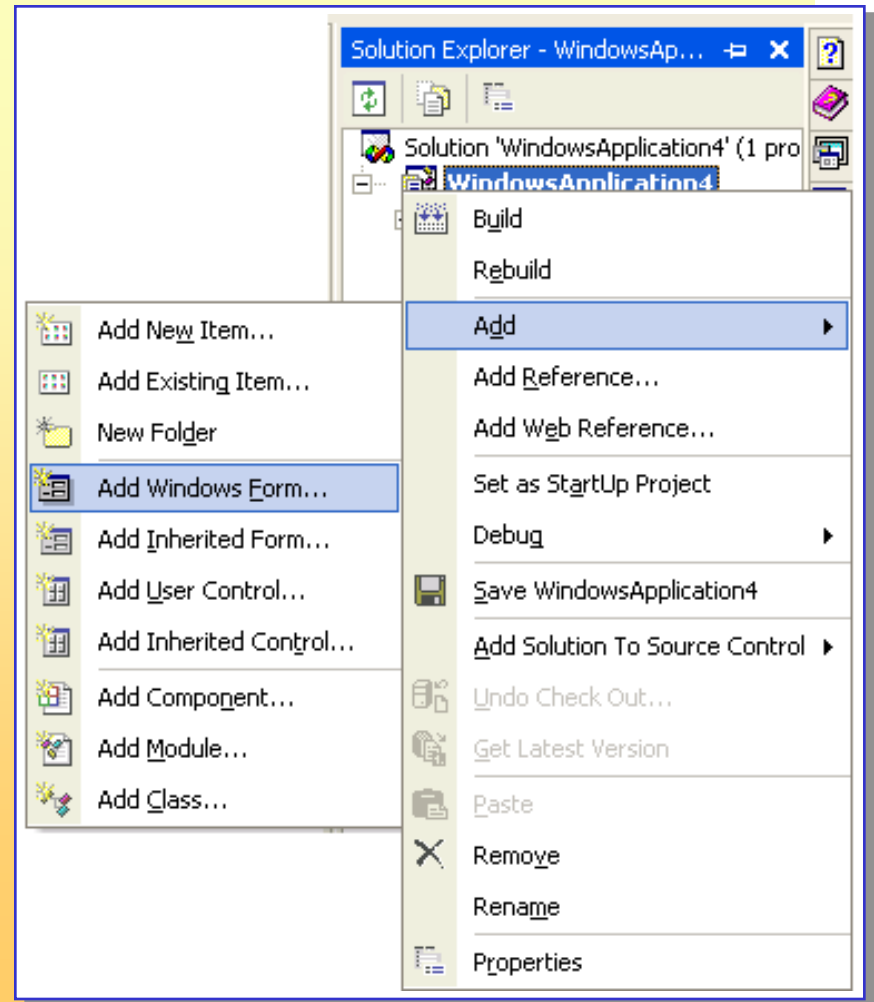


Windows Forms vs. Web Forms

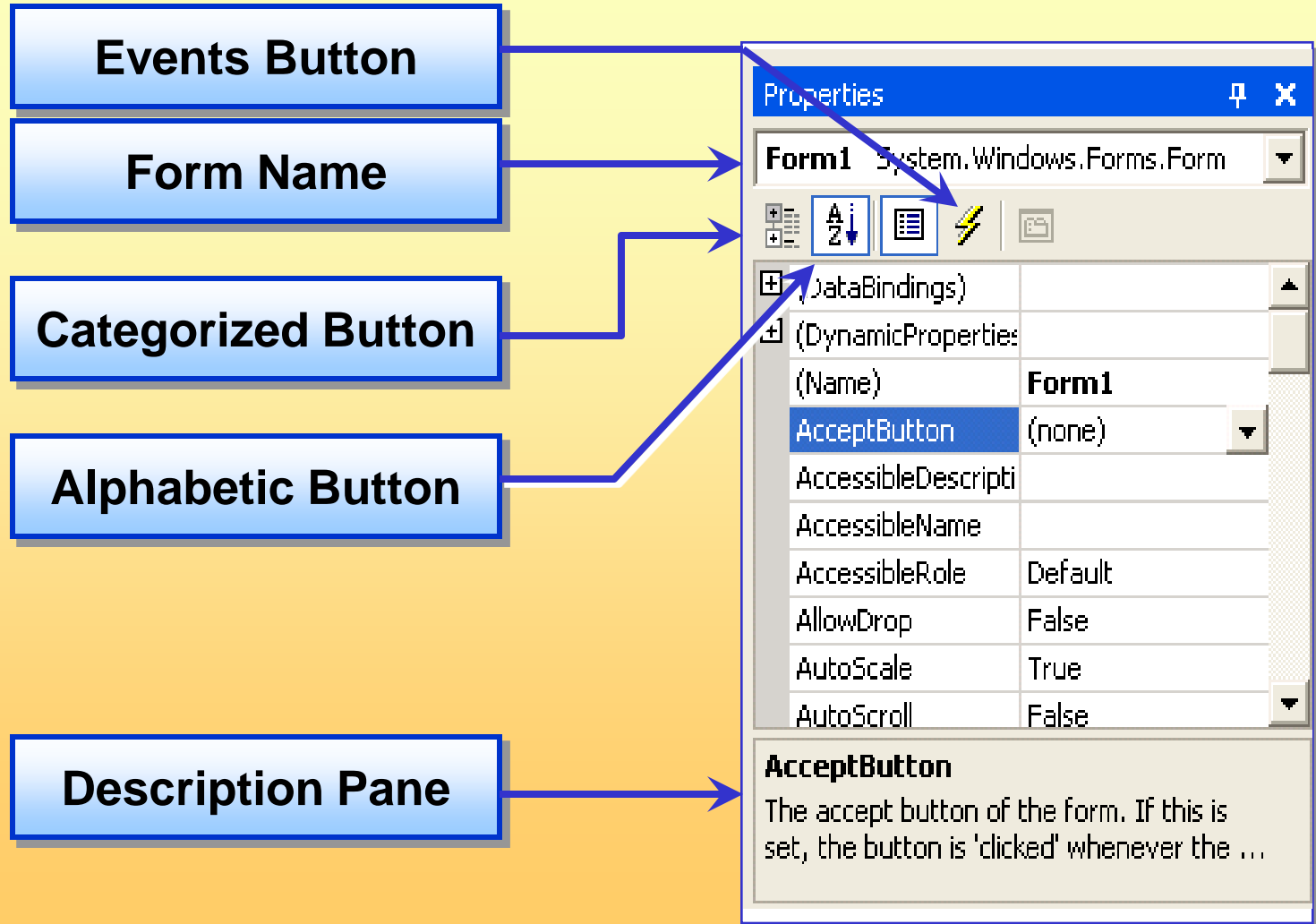
Feature	Windows Forms	Web Forms
Deployment (Updates)	Can be run without altering the registry	No download required
Graphics	Includes GDI+	Interactive or dynamic graphics require round trips to the server for updates
Responsiveness	Provide the quickest response speed for interactive applications	Requires a round trip to the Web server
Platform	Requires .NET Framework running on the client computer	Require only a browser
Programming model	Based on a client-side, Win32-based message-pump mode	Asynchronous, disconnected model (via HTTP)
Security and computer resources access	Code-based and role-based security(granularity)	Role-based security

How to Create a Form

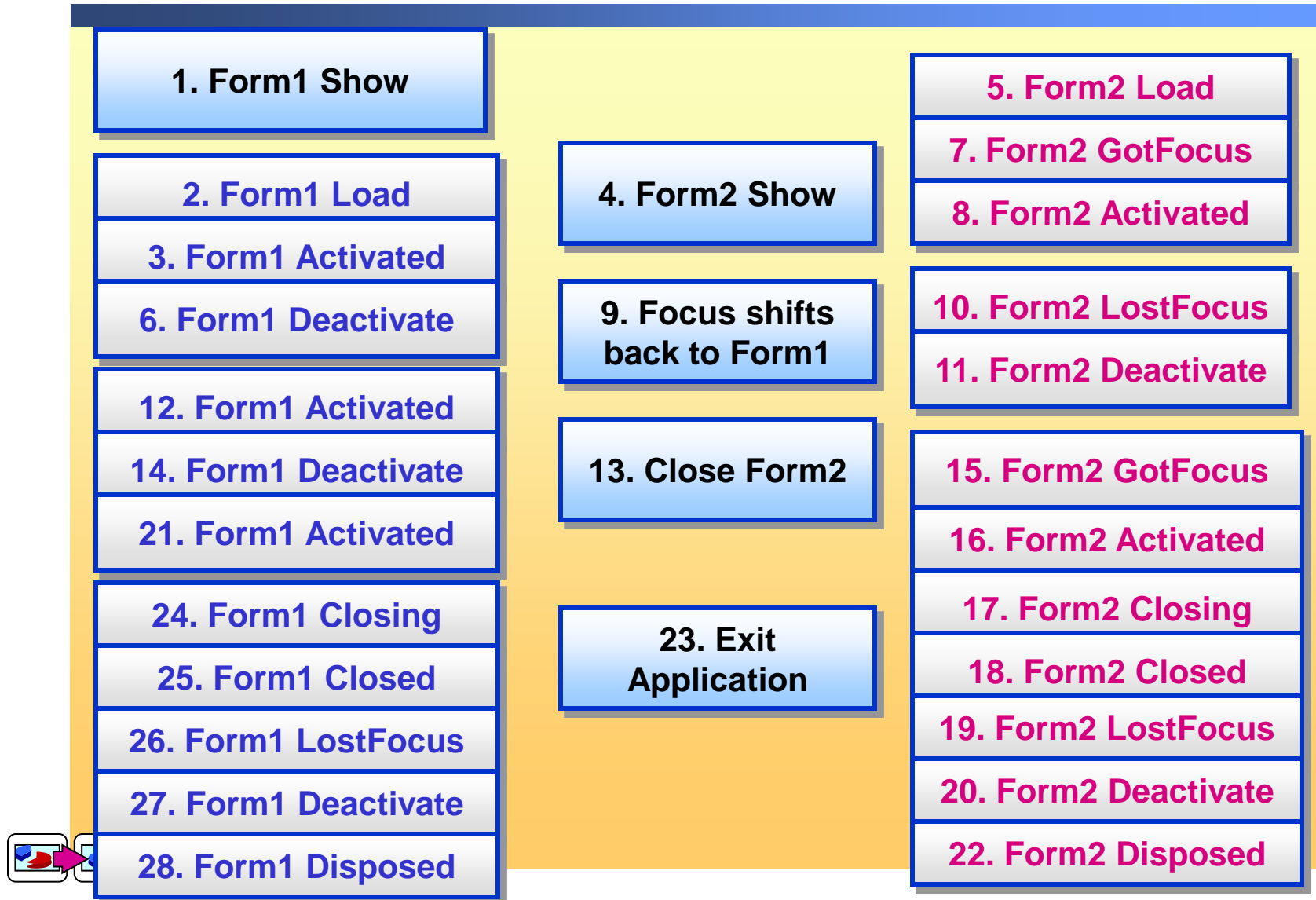
- A base form is created when you create a new project
- To create a new form
 1. Right-click the project in Solution Explorer
 2. Click Add
 3. Click Add Windows Forms



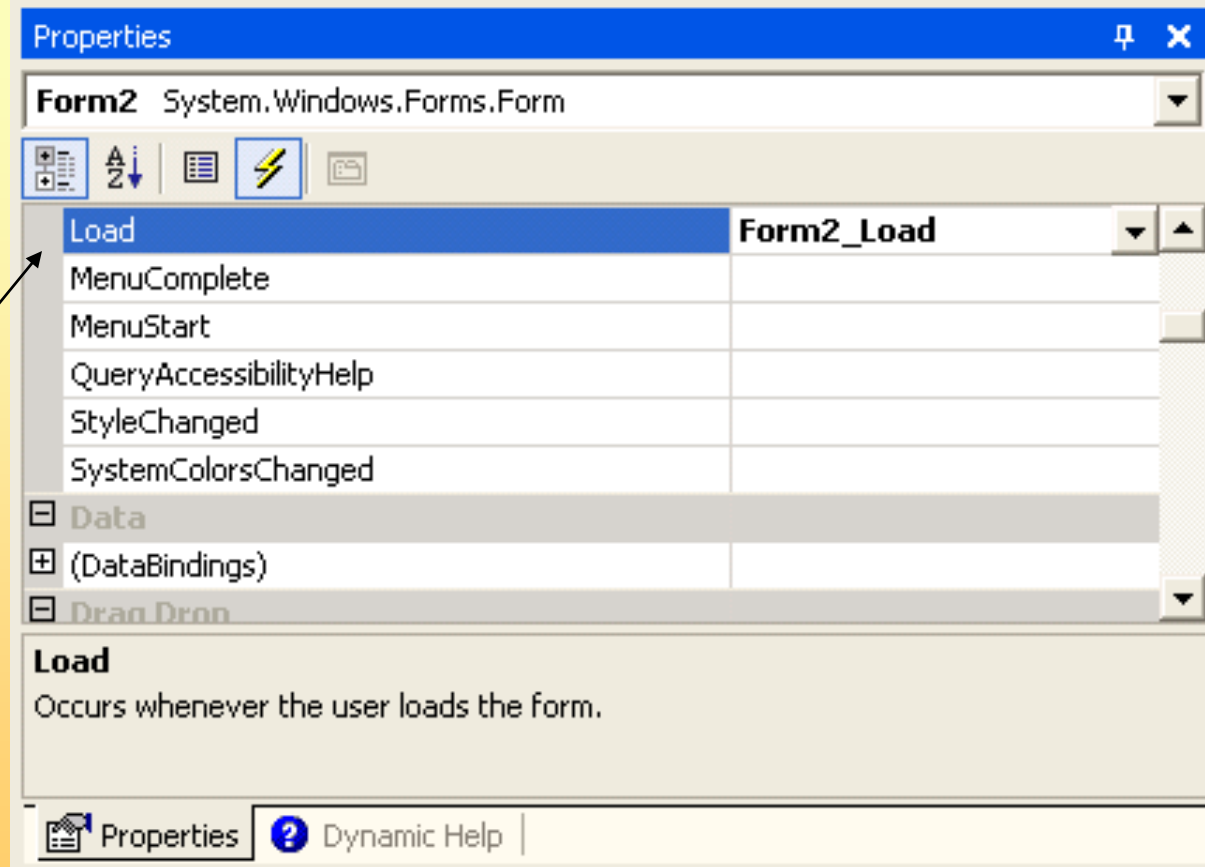
How to Set Form Properties



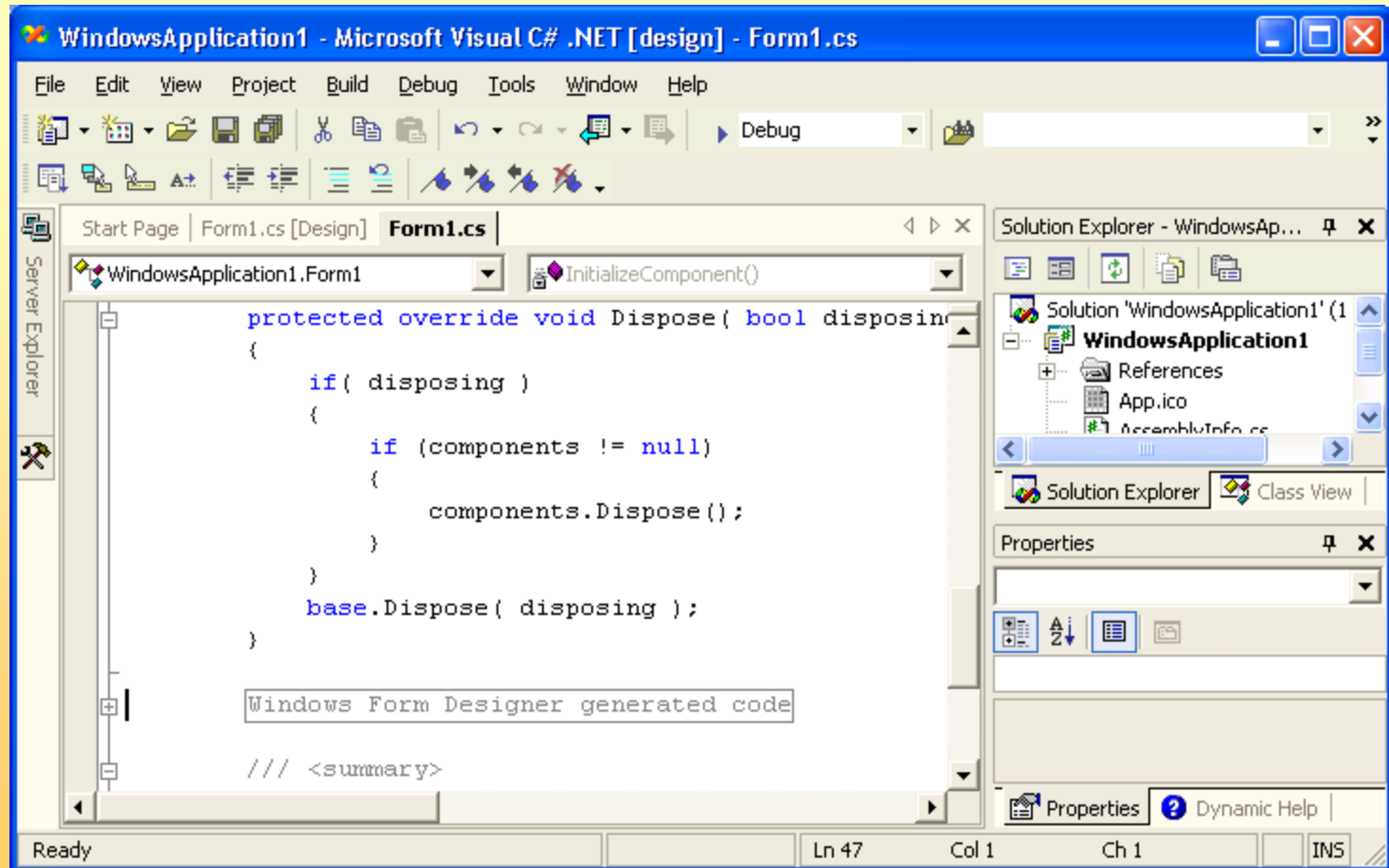
Form Life Cycle



How to Handle Form Events



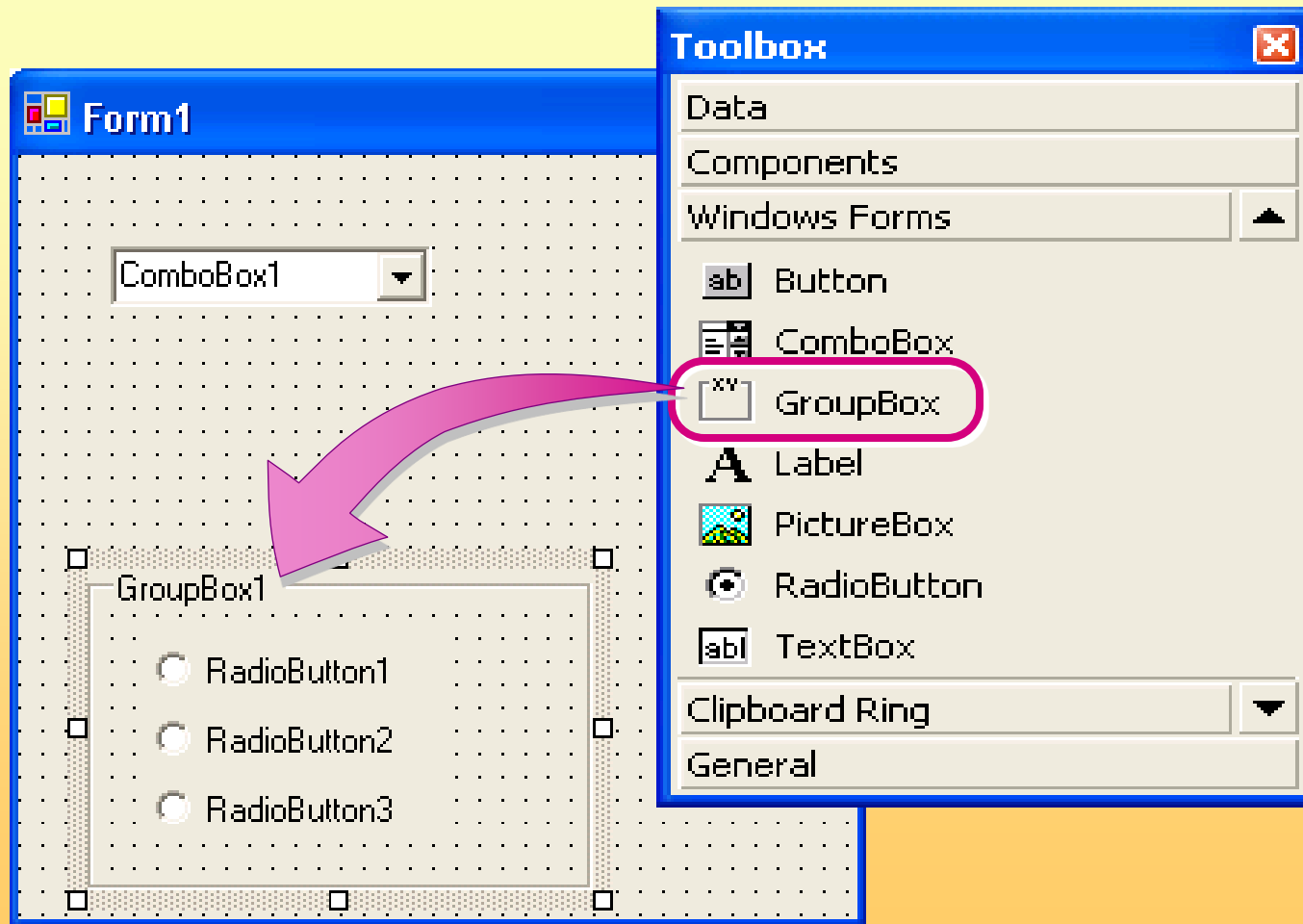
Windows Forms Designer-Generated Code



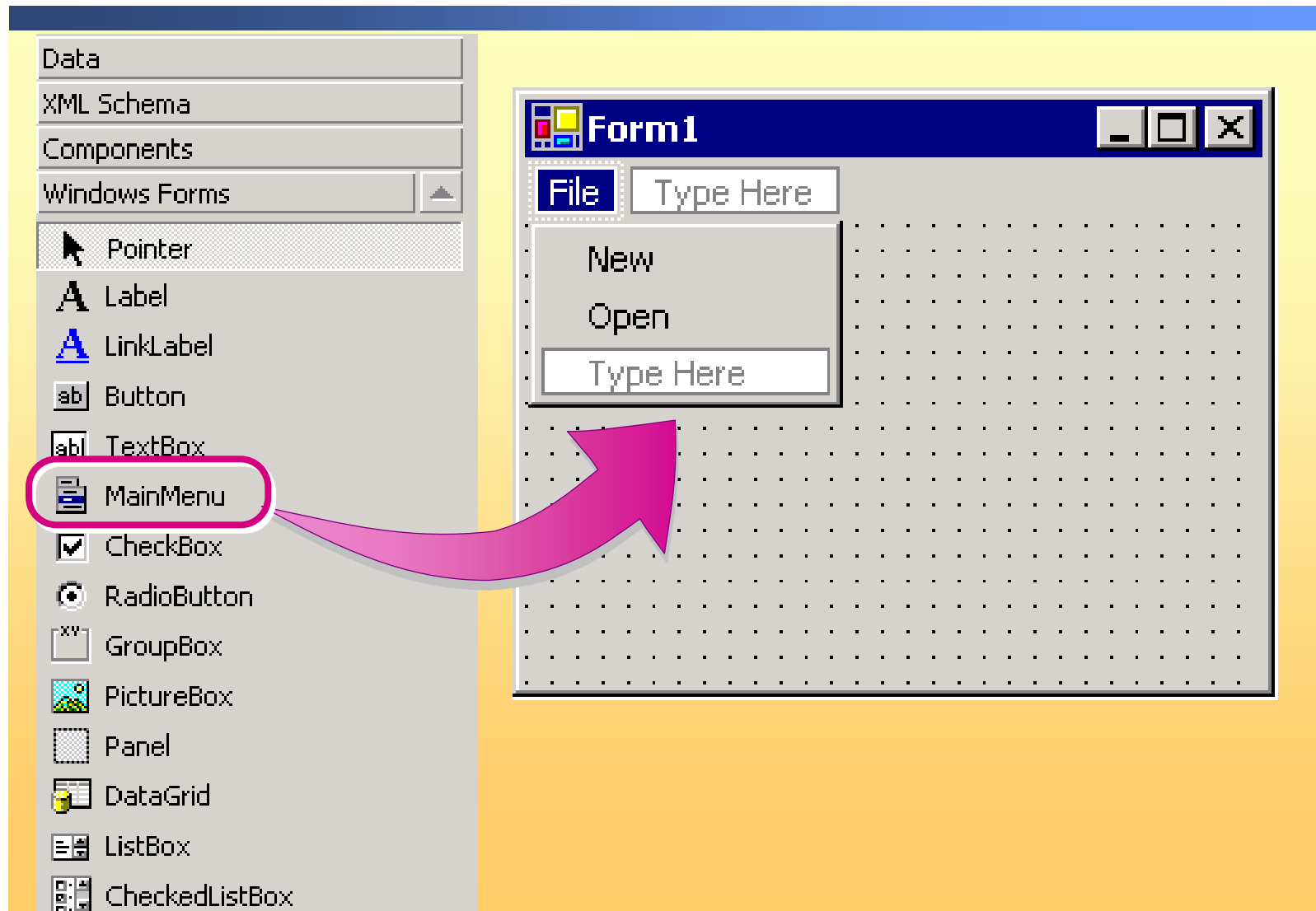
Lesson: Adding Controls to a Form

- **How to Add Controls to a Form**
- **How to Add Menus to a Form**
- **How to Customize the Controls Toolbox**
- **Practice: Creating a Form and Adding Controls**

How to Add Controls to a Form



How to Add Menus to a Form

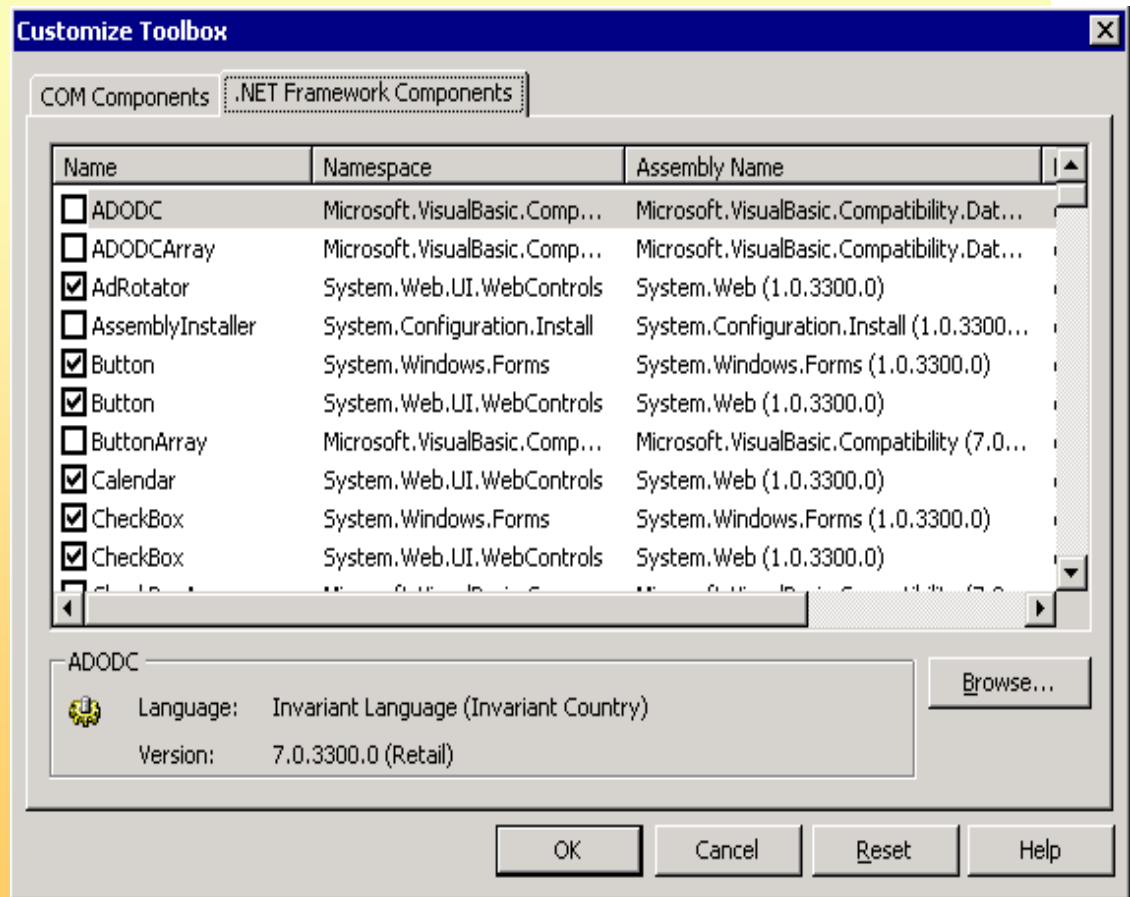


How to Customize the Controls Toolbox

1 Right-click the
Toolbox

2 Click Customize
Toolbox

3 Select the required
control on the .NET
Framework
Components page



Lesson: Creating an Inherited Form

- **Access Modifiers**
- **How to Create an Inherited Form**
- **Practice: Creating an Inherited Form**

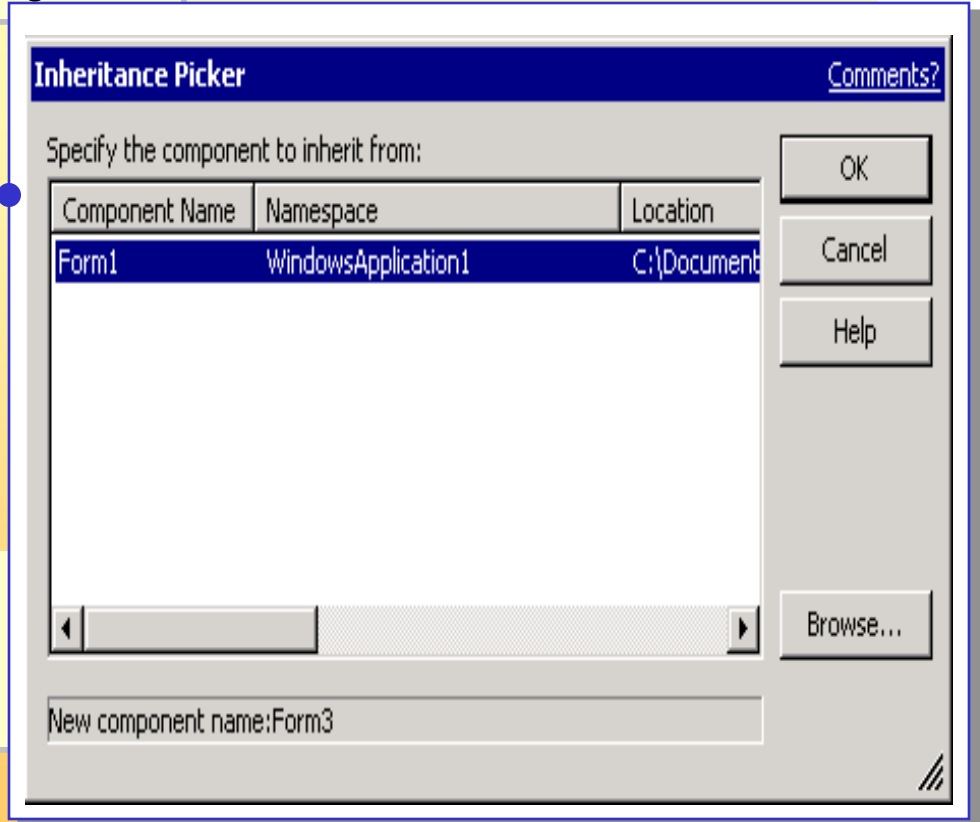
Access Modifiers (Visual Inheritance)

Access Modifier	Description
Private	Read-only to a child form, all of its property values in the property browser are disabled (shaded)
Protected	Accessible within the class and from any class that inherits from the class that declared this member
Public	Most permissive level. Public controls have full accessibility

How to Create an Inherited Form

Create an inherited form by using the Inheritance Picker dialog box

Create an inherited form programmatically



```
public class Form2 : Namespace1.Form1
```

In this practice, you will

- Set the properties of the controls on the base form to prepare them for inheritance
- Add a new form to the project inheriting it from the base form
- Set the properties on the inherited form and the controls

Begin reviewing the objectives for this practice activity

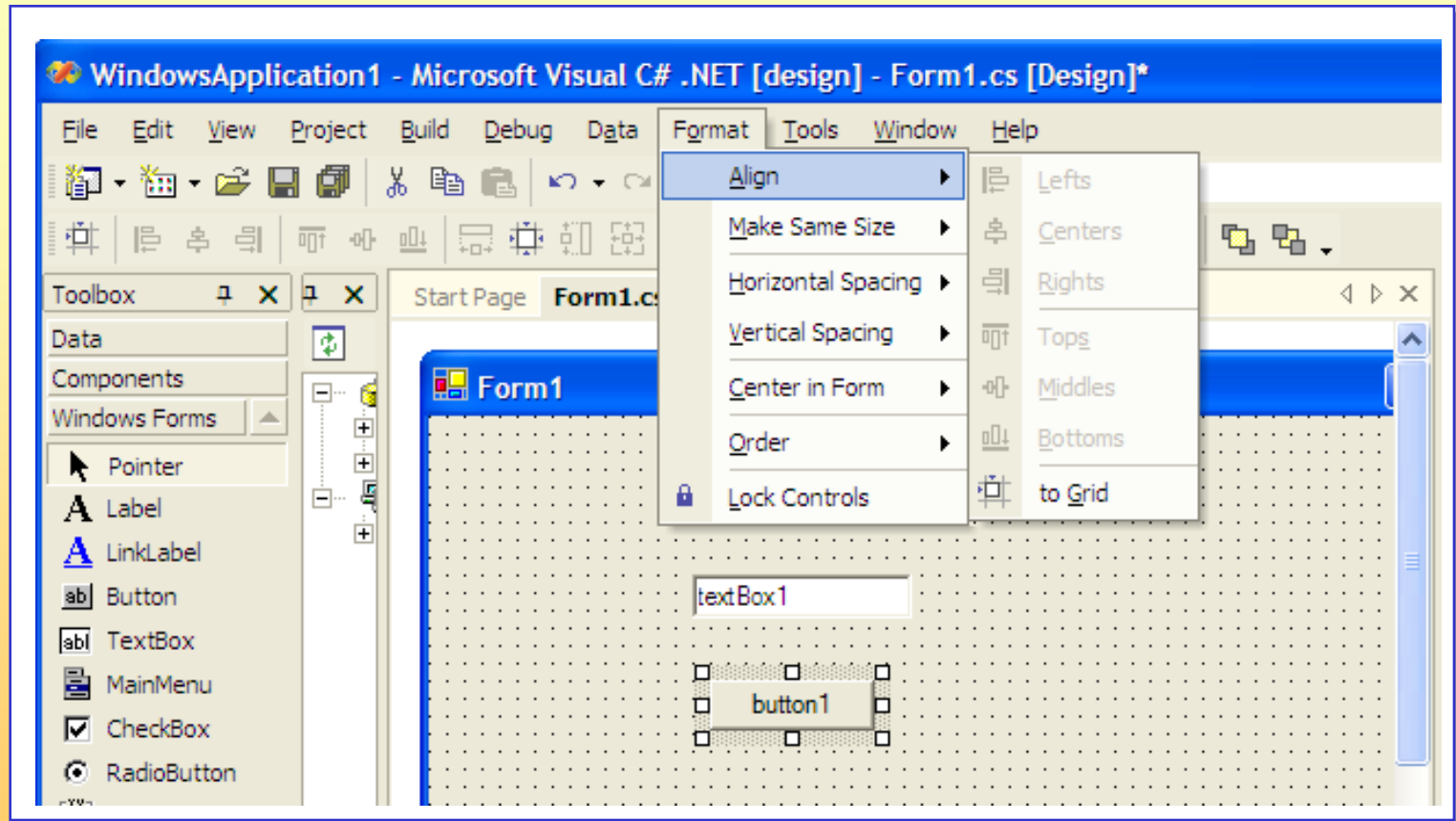
10 min



Lesson: Organizing Controls on a Form

- **How to Arrange Controls on a Form by Using the Format Menu**
- **How to Set the Tab Order for Controls**
- **How to Anchor a Control in Windows Forms**
- **How to Dock a Control in Windows Forms**
- **Demonstration: Organizing Controls on a Form**

How to Arrange Controls on a Form by Using the Format Menu



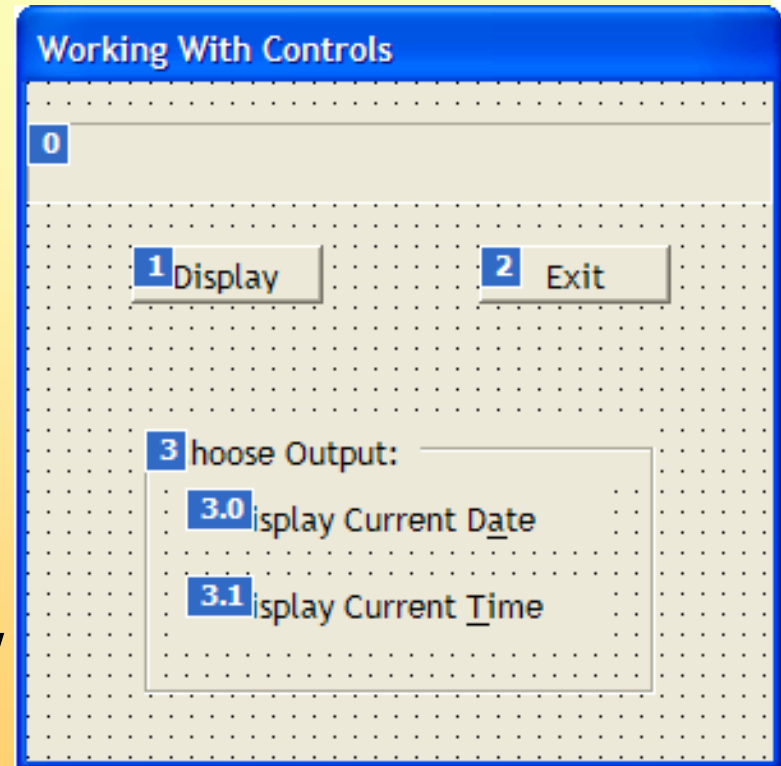
How to Set the Tab Order for Controls

■ To set the tab order for controls

- On the **View** menu, select **Tab Order**
- Click a control to change its tab order

-- OR --

- Set the **TabIndex** property
- Set the **TabStop** property to **True**



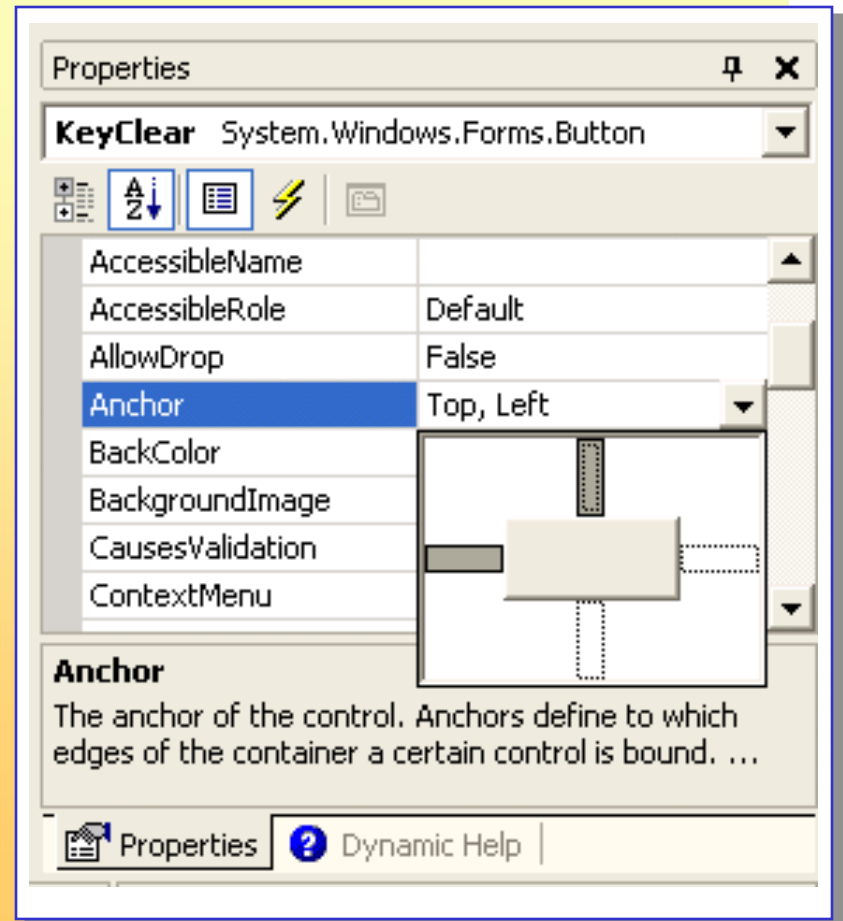
How to Anchor a Control in Windows Forms

■ Anchoring

- Ensures that the edges of the control remain in the same position with respect to the parent container

■ To anchor a control to the form

- Set its **Anchor** property
- Default value: **Top, Left**
- Other Styles: **Bottom, Right**



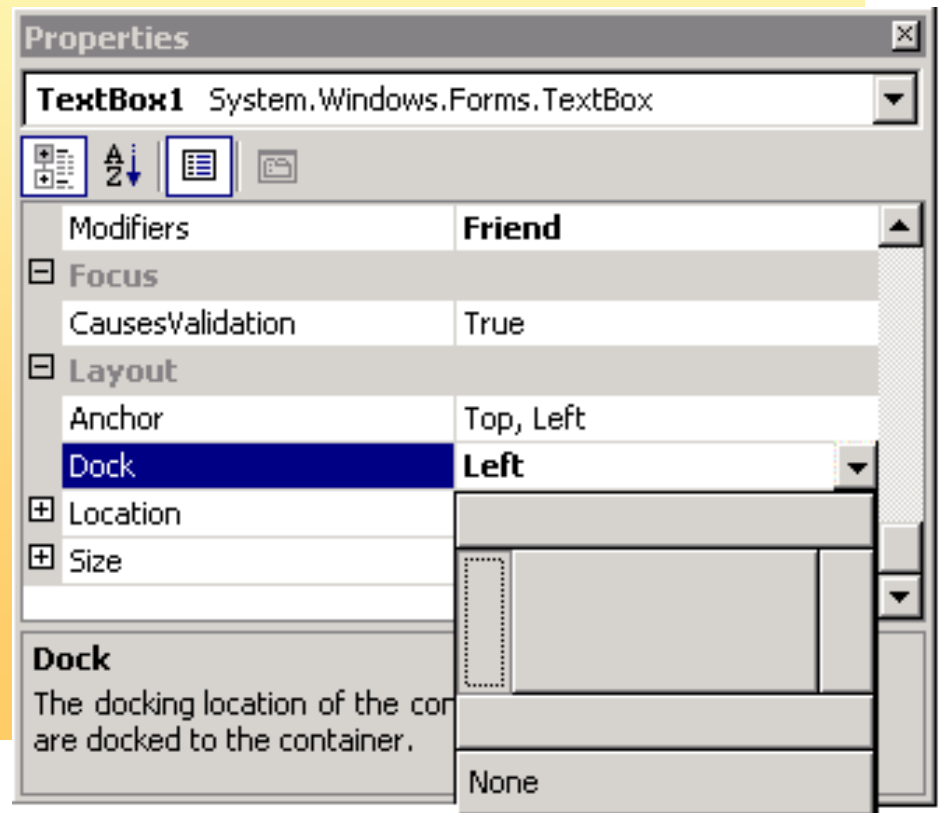
How to Dock a Control in Windows Forms

■ Docking

- Enables you to glue the edges of a control to the edges of its parent control

■ To dock a control

- Set the Dock property

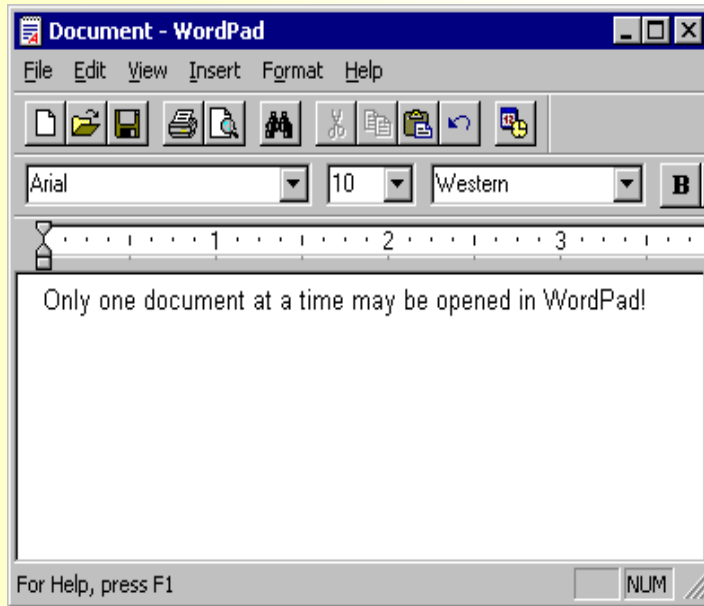


Lesson: Creating MDI Applications

- **SDI vs. MDI Applications**
- **How to Create MDI Applications**
- **How Parent and Child Forms Interact**
- **Practice: Creating an MDI Application**

SDI vs. MDI Applications

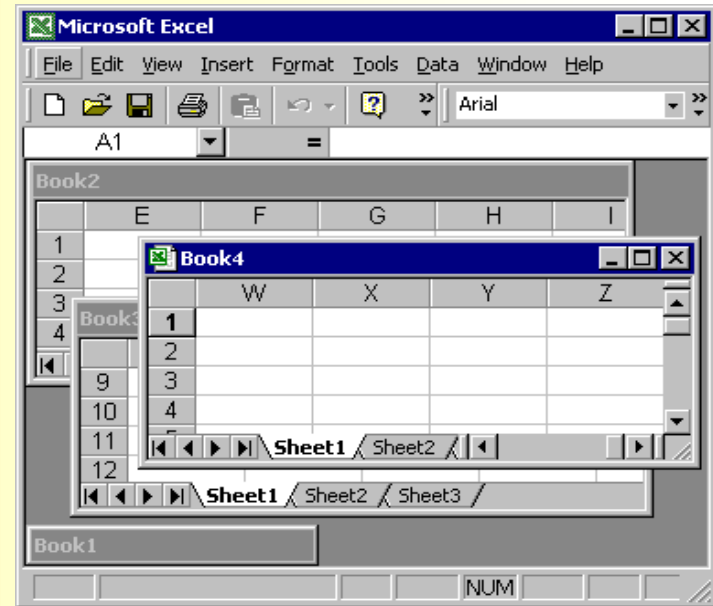
SDI



Only one document is visible

You must close one document before you open another

MDI



Displays multiple documents at the same time

Each document is displayed in its own window

How to Create MDI Applications

- **To create a parent form**
 - Create a new project
 - Set the **IsMdiContainer** property to **True**
 - Add a menu item to invoke the child form
- **To create a child form**
 - Add a new form to the project
- **To call a child form from a parent form**

```
protected void MenuItem2_OnClick(object sender, System.EventArgs e)
{
    Form2 NewMdiChild = new Form2();
    // Set the Parent Form of the Child window.
    NewMdiChild.MdiParent = this;
    // Display the new form.
    NewMdiChild.Show();
}
```


How Parent and Child Forms Interact


- To list the available child windows that are owned by the parent
 - Create a menu item (Windows) and set its MdiListproperty to True
- To determine the active MDI child
 - Use the **ActiveMdiChild** property

```
Form activeChild = this.ActiveMdiChild;
```

How Parent and Child Forms Interact

- To arrange child windows on the parent form
 - Call the **LayoutMdi** method

```
protected void CascadeWindows_Click(object  
    sender,  
    System.EventArgs e)  
{  
    Form1.LayoutMdi(MdiLayout.Cascade);  
}
```

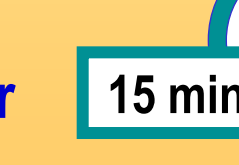


In this practice, you will

- Create the parent form
- Create the child form
- Display the child form from the parent form

Begin reviewing the objectives for this practice activity

15 min



- Create the parent form
- Create the child form
- Display the child form from the parent form

Begin reviewing the objectives for this practice activity

15 min

