

# Windows Forms(3)

Programiranje korisničkih interfejsa

Bojan Furlan

# Overview

- **Adding Accessibility Features**
- **Adding Help to an Application**
- **Localizing an Application**

# Lesson: Adding Accessibility Features

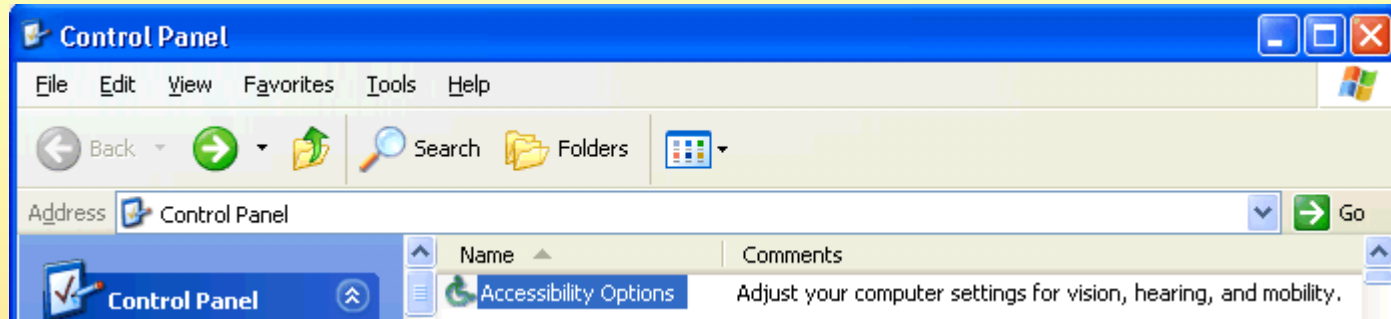
The image shows a Windows XP desktop environment. In the center, the "Accessibility Options" dialog box is open, with the "Display" tab selected. The "High Contrast" section is visible, with the "Use High Contrast" checkbox unchecked. Below it, the "Cursor Options" section contains two sliders: "Blink Rate" (set between "None" and "Fast") and "Width" (set between "Narrow" and "Wide"). At the bottom of the dialog are "OK", "Cancel", and "Apply" buttons.

In the bottom-left corner, the Start menu is open, showing "All Programs" and "Accessories". The "Accessories" menu is expanded, showing a sub-menu for "Accessibility" which is also expanded to show the following items:

- Accessibility Wizard
- Magnifier
- Narrator
- On-Screen Keyboard
- Utility Manager

# Accessibility Support in the .NET Framework

## ■ Accessibility options



## ■ Microsoft accessibility aids

- Narrator
- Magnifier
- On-Screen Keyboard

## ■ Developers can provide accessibility support by setting properties on forms and controls in their applications

# How to Make Forms and Controls Accessible

- 1 Set standard properties to values that support accessibility**  
Text, Font Size, Forecolor, Backcolor, BackgroundImage

- 2 Set accessibility properties**  
At design time or programmatically

Control Property	Description
AccessibleName	Briefly describes and identifies the object. Examples: button or menu item text
AccessibleDescription	Provides greater context for low-vision or blind users
AccessibleRole	Describes the use of the element in the user interface

```
this.AppExitButton = new System.Windows.Forms.PushButton();  
this.AppExitButton.Text = "E&xit";  
AppExitButton.AccessibleRole =  
    System.Windows.Forms.AccessibleRole.PushButton;  
AppExitButton.AccessibleName = "Exit";  
AppExitButton.AccessibleDescription = "Use this button to  
    exit the application";  
this.Controls.Add(this.AppExitButton);
```

# How to Test Accessibility

**1** Build the application

**2** Turn on an accessibility aid, such as Narrator

**3** Run the application

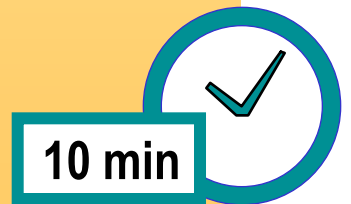
# Practice: Adding Accessibility Support to an Application



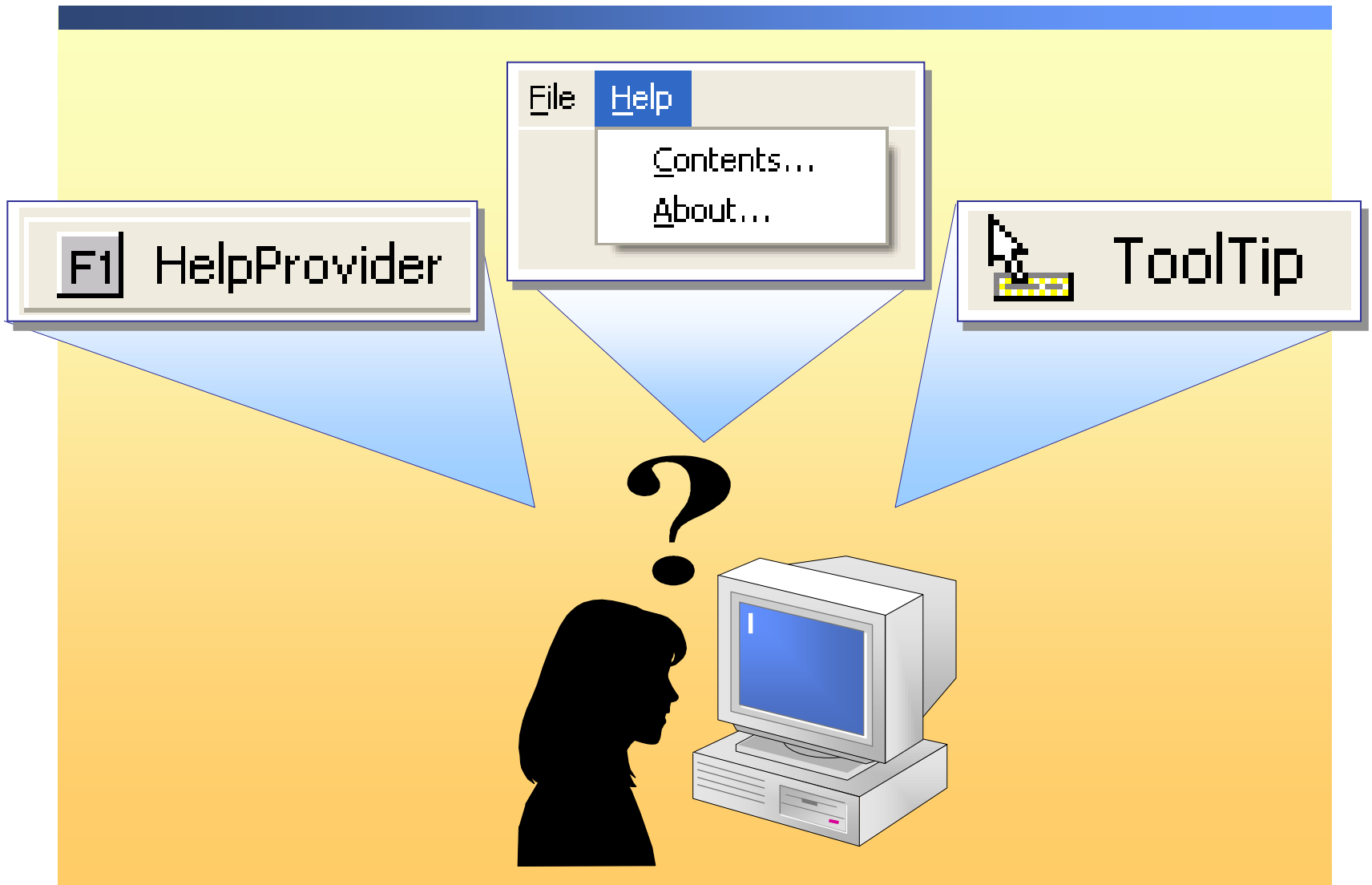
## In this practice, you will

- Set the AccessibleName property for a control
- Enable Narrator
- Run the application to see the results

**Begin reviewing the objectives for  
this practice activity**



# Lesson: Adding Help to an Application





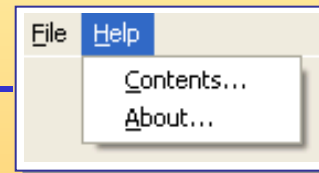
# Help in the .NET Framework

## ■ Context-sensitive Help

- **HelpProvider** control
- **HelpButton** property



## ■ Help menu support



## ■ The ToolTip control



# How to Add Context-Sensitive Help for Forms and Controls

**1** Add the HelpProvider control  
Set the **HelpNamespace** property

**2** Add a HelpButton to the form

**3** For each control that you want to add Help information set the following properties

- HelpKeyword
- HelpNavigator
- HelpString

**4** Build and test the application  
Give a control focus and press F1

# How to Link Help Topics to a Menu

**1**

Set the **HelpNamespace** property to point to a file or URL, such as `http://localhost/InternalBusinessAppHelp.htm`

**2**

Add a **MainMenu** control to the form

- Add **Help** menu item and subitems
- Implement **Help** menu subitem click event procedures to open the **HelpNamespace**

Parent of the Help dialog box

Path and name of Help file

```
Help.ShowHelp (this, HelpProvider.HelpNamespace);
```

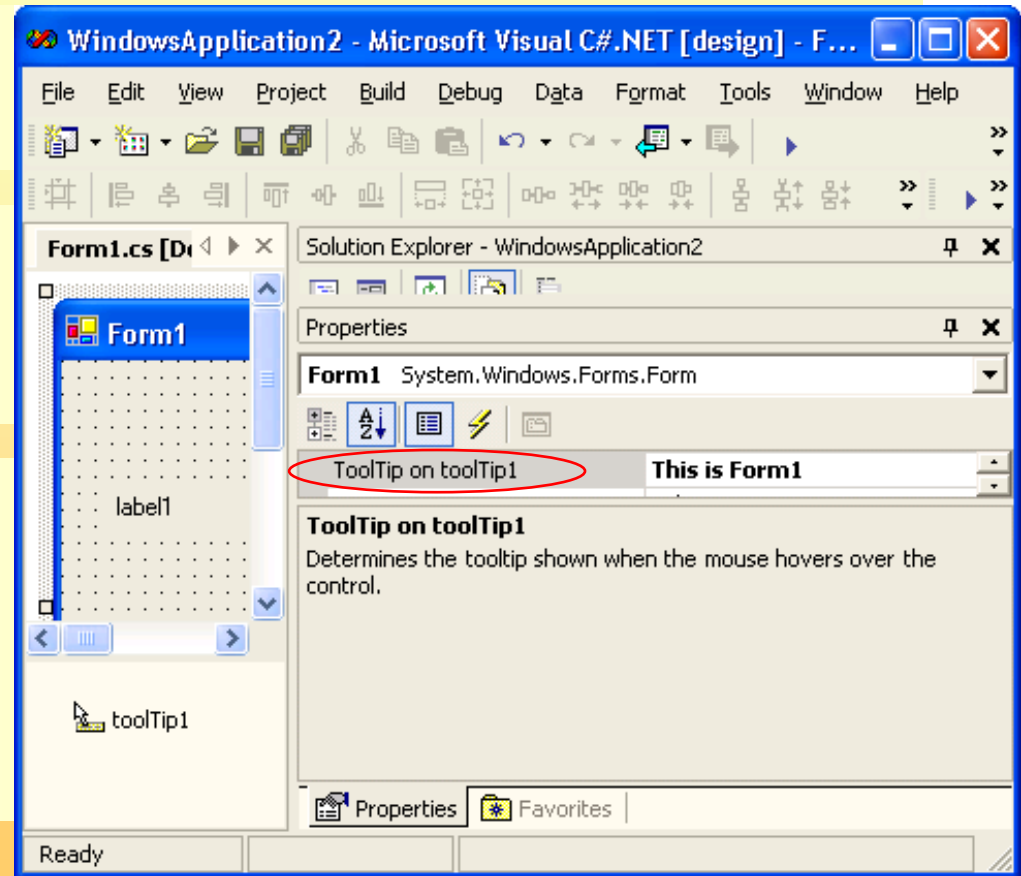
# How to Display Help with the ToolTip Control

**1** Add the ToolTip control

**2** Set the value for the ToolTip on ToolTip... property

**3** Build and test the application

Point to a control that has an associated ToolTip



# Practice: Adding Help to an Application



## In this practice, you will

- Add context-sensitive Help to an application
- Link a Help file to context-sensitive Help
- Link a Help file to a **Help** menu item

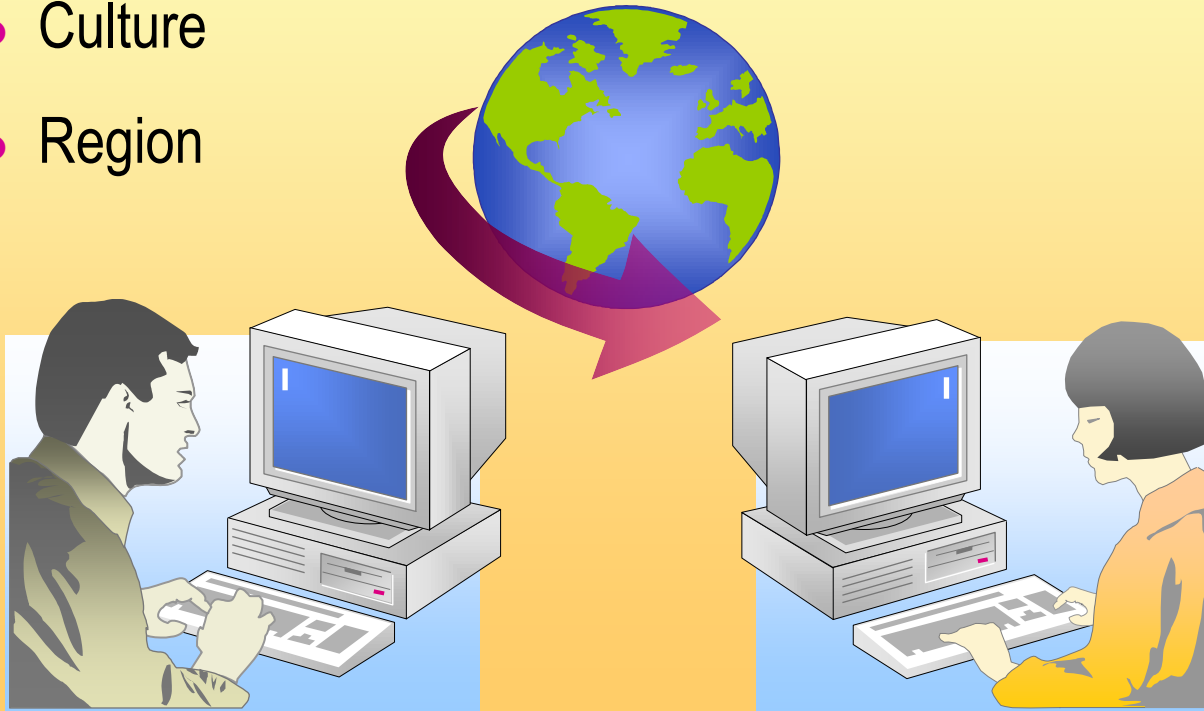
Begin reviewing the objectives for  
this practice activity

15 min



# Lesson: Localizing an Application

- Globalization
- Localization
  - Culture
  - Region



# Localization in the .NET Framework

- **Localizing the user interface elements**
- **Localizing other resources**
  - Strings
  - Bitmaps
  - Other objects, such as audio files

# How to Set Localization Properties

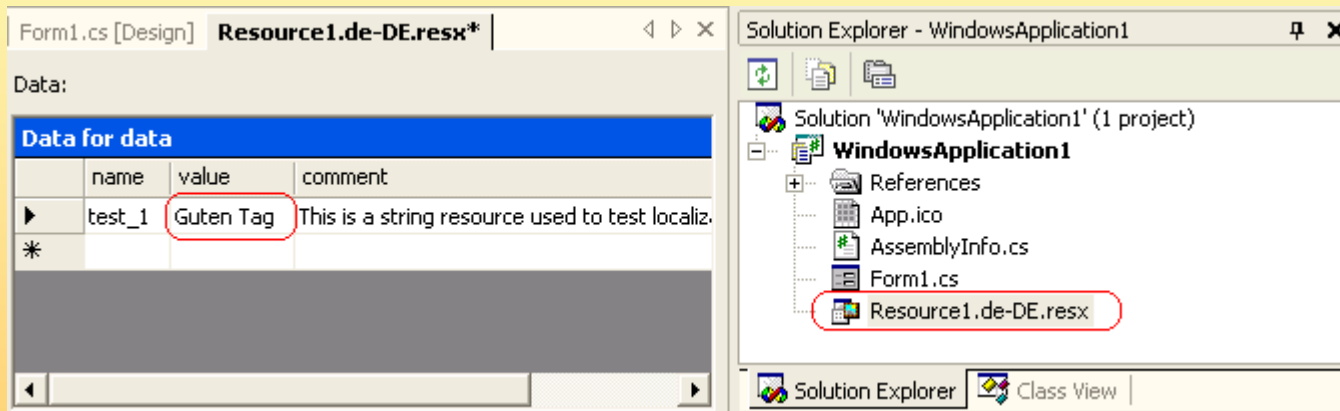
- 1** Create the default culture version of the form
- 2** Set the Localizable property of the form to True
- 3** Set the appropriate value for the Language property of the form
- 4** Modify the Text property values for the form and controls into the appropriate language
- 5** Resize and/or reposition each control as needed
- 6** Build the application



# How to Create Localized Resource Files

**1** Open an existing project and add an assembly resource file for the appropriate culture

**2** Add entries to the resource file with values in the appropriate language for the culture



The screenshot shows the Visual Studio IDE. On the left, the 'Resource1.de-DE.resx\*' file is open in the 'Data' view. It displays a table with the following content:

Data for data			
	name	value	comment
▶	test_1	Guten Tag	This is a string resource used to test localiz.
*			

On the right, the 'Solution Explorer' shows the project structure for 'WindowsApplication1'. The file 'Resource1.de-DE.resx' is highlighted with a red circle, indicating its location within the project.

**3** Save the file

**4** Build the application

# How to Change the Locale

The user can change the regional and language options from Control Panel

- 1 Add code to an application to programmatically set the culture and UI Culture for an application to the new value

```
using System.Globalization;
using System.Resources;
using System.Threading;

...
Thread.CurrentThread.CurrentUICulture =
    Thread.CurrentThread.CurrentCulture;
```

- 2 Add code to the application to use a resource manager to extract the elements from the resource file

```
ResourceManager rm = new
    ResourceManager("MyNamespace.Resource1",
        Assembly.GetExecutingAssembly());

MessageBox.Show(rm.GetString("test_1"));
```

Root name of the resource file

Main assembly for the resources

# Practice: Localizing an Application



## In this practice, you will

- Localize the user interface of an application
- Add localized string resources to an application

**Begin reviewing the objectives for  
this practice activity**

15 min

