CIS 365 Lab 11: Control interactions

Adam Lewis

<2015-05-18 Mon>

Contents

1 Objectives

1. Demonstrate how the different controls communicate with each other within WPF.

2. Introduce the use of pre-defined dialog boxes.

2 Background

Event-driven programming requires different controls cooperate with each other in order to get work done. We will look at a simple application that shows how we can manipulate the contents of a TextBox in the code-behind for other controls in a user interface.

3 Instructions

3.1 Setup

Start Visual Studio and create a new WPF application named *lab11*.

3.2 User Interfacebp

Add the following to the starting form in your application:

- Replace the **Grid** layout in your application window with a **DockPanel** layout. Make the size of the panel match the size of the application main window.
- Add a **GroupBox** control to the **DockPanel**. Set the **DockPanel.Dock** property to "Right". Set the name of this control to be **gbxFontsAndStuff**.
  - Add a **Grid** layout control to the **GroupBox**. Configure the grid to have 1 column with five rows.
  - Add two check boxes to the grid layout: Italic and Bold: Name the controls as **cbItalic** and **cbBold**. Set values appropriately.
  - Add a **Slider** control to the grid layout. Name this control **sldFontSize**. Set the minimum property's value to 8 and the maximum's property to "32".
  - Add a button named **btnSetFS** with label "Set Font Size".
  - Add a **FontDialog** component named **fontDialog1**.
  - Add a button named **btnFontChange** with label "Set Font"
- Add a **TextBox** control to your application. Set the name of the control to **tbxSampleText**. Set the **Text** property of this text box to some reasonable string of gibberish.

3.3 Code behind

Now let's do some things that should how changes in state within one control can result in the changes in state in other controls on a form.

3.3.1 Checkboxes and rich text controls

Select the **cbItalic** control. In the *Properties* window, switch to the *Events* tab and double-click on the **CheckedChanged** event. A new method **cbItalic**$_{\textbf{CheckedChnaged}}$ is added to the Code for the Form. Add the following to the this method:

```
private void cbItalic_CheckedChanged(object sender, EventArgs e)
{
  tbxSampleTextBox.FontWeight = FontWeights.Italic;
}
```

Now do the same thing for the **cbItalicUnchecced** event, setting the weight to "Normal":

```
private void cbItalic_UnChecked(object sender, EventArgs e)
{
  tbxSampleTextBox.FontWeight = FontWeights.Normal;
}
```

1. Your turn Implement the code-behind for the **CheckedChanged** event for the other two check boxes on the user interface. Note that you will need to set the **FontWeight** property of the textbox rather than **FontStyle** property. See the MSDN page for **FontWeight**.

3.3.2 Changing font sizes

Select the **setFontSize** button and add the code-behind for the **Click** event. In this case, we need to get the new font size from the **sldFontSize** control's **Value** property. The text box has properties that allow us to adjust the characteristics of the fonts in content.

```
private void btnSetFS_Click(object sender, EventArgs e)
{
  tbx.SampleText.FontSize = sldFontSize.Value;
}
```

4 Submission instructions

Put your final code and a screen shot of your application into a PDF file. Attach this PDF file to your submission on Blackboard.