**MenuStrips Using Multiple Document Interface (MDI) Part 4**

**Creating Dialog Boxes**

Applications typically include menus with commands that perform actions. However, many commands require information from the user before executing. For example, an Alignment command must be told the type of paragraph alignment before it can execute. An application uses dialog boxes to get the information needed to execute a command.

Dialog boxes have certain features. They cannot be sized, minimized or maximized. They include at least an OK button and a Cancel button, and they require the user to select one of these buttons before control is given back to the calling form.

A dialog box is created for an application by first adding a new form to the project. The Form control has properties for designating the form as a dialog box:

* **ControBox** is set to False because a dialog box should not offer access to the System menu.
* **MaximizeBox** is set to False because a dialog box should not be sized.
* **MinimizeBox** is set to False because a dialog box should not be sized.
* **FormBorderStyle** is set to FixedDialog to prevent the dialog box from being sized.

After setting properties, the dialog box can then be customized with objects that get the needed user input. For example, an Alignment dialog box might look similar to:

Left

Center

Right

OK

Cancel

Alignment

The Button control has a property that must be set for the OK and Cancel buttons in a dialog box:

* **DialogResult** can be set to None, OK, Cancel, Abort, Retry, Ignore, Yes, or No. None is the default. The option should be set to correspond to the Text property of a dialog box button. For example, a Yes button has Text set to Yes and DialogResult set to Yes.

The Form control has a method for displaying a dialog box as a modal form:

* **ShowDialog()**  displays a form nodally.

A modal form does not allow other forms to receive input until it has been removed from the screen. For example, dialog box displayed as a modal form requires the user to select the OK or Cancel button, which removes the dialog box from the screen, before other forms can receive input.

The code below instantiates a dialog box object and then displays the object as a modal form. No other statements are executed until the user selects either the OK or Cancel buttons in the dialog box. If the user selected OK, the If…Then statement is executed to change the alignment of the text box in the active form.:

* Dim alignmentDB As New AlignmnetDialogBox()
* Dim activeDoc As NewDocument = Me.ActiveMdiChild
* ‘Show dialog box
* alignmentDB.ShowDialog()
* ‘change alignment of text in active document
* If alignmentDB.DialogResult= Windows.Form.DialogResult.OK Then If alignmentDB.radLeft.Check Then activeDoc.txtDocument.TextAlign= HorizontalAlignment.Left ElseIf alignmentDB.radCenter.Checked Then activeDoc.txtDocument.TextAlign=HorizontalAlignment.Center ElseIf alignmentDB.radRight.Checked Then activeDoc.txtDocument.TextAlign=HorizontalAlignment.Right End If

**TextEditor Assignment Part 4**

1. **Add a Dialog Box to the Project**
   1. Select Project 🡪 Add Windows Form. A dialog box is displayed.
      1. In the Templates list, click Windows Form, if it is not already selected.
      2. In the Nam box, type AlignmentDialogBox.vb
      3. Select Add. A form named AlignmentDialogBox.vb is displayed in a Design window.
   2. Set the Text property of the AlignmentDialogBox form to Alignment.
   3. Set the ControBox, MaximizeBox and MinimizeBox properties to False.
   4. Set the FormBorderStyle to FixedDialog.
   5. Use the table below for setting object Properties.

|  |  |  |  |
| --- | --- | --- | --- |
| **Object** | **(Name)** | **Text** | **DialogResult** |
| GroupBox1 | grpAlignmentOptions | ChooseAlignment |  |
| RadioButton1 | radLeft | Left |  |
| RadioButton2 | radCenter | Center |  |
| RadioButton3 | radRight | Right |  |
| Button1 | btnOK | OK | OK |
| Button2 | btnCancel | Cancel | Cancel |

1. **Add A Format Menu To Form1**
   1. Display the Form1 Design window.
   2. Modify the parent form to include a Format menu. Use the table below for setting object properties.

|  |  |
| --- | --- |
| Object | Text |
| MenuItem1 | Format |
| MenuItem2 | Alignment |

1. **Modify The Application Code**
   1. Display the form1Code window
   2. Create an AlignmentToolStripMenuItem\_Click event…
      1. Dim alignmentDB As New AlignmentDialogBox()
      2. Dim activeDoc As NewDocument = Me.ActiveMdiChild
      3. ‘Show dialog box
      4. alignmentDB.ShowDialog()
      5. ‘change alignment of text in active document
      6. If alignmentDB.DialogResult=Windows.FormsDialogResult.OK Then
      7. If alignmentDB.radLeft.Checked Then activeDoc.txtDocument.TextAlign=HorizontaAlignment.Left
      8. ElseIf alignmentDB.radCenter.Checked Then
      9. activeDoc.txtDocument.TextAlign = HorizontalAlignment.Center
      10. Else
      11. activeDoc.txtDocument.TextAlign=HorizontalAlignment.Right
      12. End If
2. **Run the Application**
   1. Save the modified TexEditor project and then run the application.
   2. Display a new document window and then type several paragraphs of text into the New Document 1 window.
   3. Select Format 🡪 Alignment. A dialog box is displayed
      1. Click Center
      2. Click OK. The text is center aligned.
   4. Test the other alignment options.
   5. Close the document window and quit the TextEditor application.