Dave's Development Blog
Software Development using Borland / Codegear / Embarcadero RAD Studio

Handling Folding and Unfolding code in the IDE

By David | July 21, 2011 0 Comment

First of all I need to apologise for a mistake in the original article on setting up a expert/wizard where by a package based wizard would not load correctly in the IDE. What is missing from the code is the exporting of the Register procedure in the interface section of the wizard module as shown below. Once this is added the wizard menu will appear under the **Help** menu of the IDE.

```
Unit IDEFoldedBitsWizard;

Interface

Uses
  ToolsAPI;

Type
  TIDEFoldedBitsWizard = Class(TInterfacedObject, IOTAWizard, IOTAMenuWizard)
  public
    ...
  End;

  Function InitWizard(Const BorlandIDEServices : IBorlandIDEServices;
    RegisterProc : TWizardRegisterProc;
    var Terminate: TWizardTerminateProc) : Boolean; StdCall;

  Procedure Register; // MISSING LINE

Exports
  InitWizard Name WizardEntryPoint;

Implementation
...
```

Now for code folding.

Handling code folding is a little more difficult that other OTA things as the **BorlandIDEServices** does not export an interface from any of its sub-interfaces. By this I mean you can not get an **IOTAElideActions**

interface from any property or method directly. So to get the interface we have to ask another interface whether it supports it. Luckily for us the clue is in the **ToolsAPI.pas** file where it says this interface is present in **IOTAEditView** interface.

What we need to do (as shown below) is ask the **IOTAEditView** interface if it has the **IOTAElideActions** interface and give us a reference to it.

```
procedure TIDEFoldedBitsWizard.Execute;

var
  TopView: IOTAEditView;
  I : IOTAElideActions;

begin
  TopView := (BorlandIDEServices As IOTAEditorServices).TopView;
  If TopView.QueryInterface(IOTAElideActions, I) = S_OK Then
    Begin
      I.ToggleElisions;
      TopView.Paint;
    End;
end;
```

What this piece of code does it toggle the folding and unfolding of the code at the current cursor position in the active editor. We use the QueryInterface method to obtain the new interface (so long as it returns S_OK) and then we can use it as any other interface. One thing I've found with folding / unfolding code in the editor is that it DOES NOT update the interface until you move something, therefore you need to force this by calling the **IOTAEditView**'s **Paint** method.

Now for a quick explanation of the different methods available.

The **IOTAElideActions120** interface implements the following (available from Delphi 2006 onwards):

- **ElideNearestBlock**: This folds the block in which the cursor currently sits;
- **UnElideNearestBlock**: This unfolds the block in which the cursor currently sits;
- **UnElideAllBlocks**: This folds ALL blocks in the editor;
- **EnableElisions**: This enables code folding.

The **IOTAElideActions** interface implements the following (available from Delphi 2010 onwards):

- **ToggleElisions**: This folds and unfolds the current block in which the cursor resides;
- **ElideNamespaces**: This folds all Namespaces in the currently active module;
- **ElideRegions**: This folds all Regions in the currently active module;
- **ElideTypes**: This folds all Types in the currently active module;
- **ElideNestedProcs**: This folds all Nested Procedures and Functions in the currently active module;
- **ElideGlobals**: This folds all Globally defined variables in the currently active module;
- **ElideMethods**: This folds all Methods in the currently active module;

Unfortunately there is no way to test whether you are in a nested block, however if you call the **IOTAEditView**'s **Paint** method when moving and displaying a new cursor position this should automatically open any folded code and draw the screen correctly.

regards
Dave.

Category: Open Tools API Tags: Borland, BorlandIDEServices, CodeGear, Delphi, Embarcadero, Experts, IOTAEditorServices, IOTAEditView, IOTAElideActions, IOTAElideActions120, OTA, RAD Studio