Dave's Development Blog

Software Development using Borland / Codegear / Embarcadero RAD Studio

Chapter 7: IDE Compilation Events

By David | August 5, 2011 1 Comment

The interface(s) were are going to look at are the IOTAIDENotifierXxx interfaces. Now when implementing these interfaces its important to implement ALL the inherited interfaces, else the methods signatures of these versions will NOT be called. I've recently made this mistake again and wondered for an hour why my Delphi 2010 project manager menu didn't appear. It was simply because I didn't place the IOTALocalMenu interface in the class heritage list ☹ The above is especially important if you want the expert to work in earlier version of Delphi.

I'm using Delphi 2010 at the moment, so the interfaces available to me in ToolAPI.pas are as follows:

- **IOTAIDENotifier** (I think this has always been available)
- **IOTAIDENotifier50** (Available from Delphi 5 onwards)
- **IOTAIDENotifier80** (Available from Delphi 2005 onwards)

So we now need to define our wizard interface with these interfaces as follows:

```
TTestingHelperWizard = class(TNotifierObject, IOTANotifier, IOTAIDENotifier,
  IOTAIDENotifier50 {$IFDEF D2005}, IOTAIDENotifier80 {$ENDIF}, IOTAWizard)
{$IFDEF D2005} Strict {$ENDIF} Private
{$IFDEF D2005} Strict {$ENDIF} Protected
Public
  Constructor Create;
  Destructor Destroy; Override;
  // IOTAWizard methods
  procedure Execute;
  function GetIDString: string;
  function GetName: string;
  function GetState: TWizardState;
  // IOTANotifier methods
  Procedure FileNotification(NotifyCode : TOTAFileNotification;
    Const FileName : String; var cancel : Boolean);
  procedure BeforeCompile(const Project: IOTAProject; var Cancel: Boolean); overload;
  procedure AfterCompile(Succeeded: Boolean); overload;
  // IOTANotifier50 methods
  procedure BeforeCompile(const Project: IOTAProject; IsCodeInsight: Boolean;
    var Cancel: Boolean); Overload;
  procedure AfterCompile(Succeeded: Boolean; IsCodeInsight: Boolean); overload;
  // IOTANotifier80 methods
  {$IFDEF D2005}
  procedure AfterCompile(const Project: IOTAProject; Succeeded:
    Boolean; IsCodeInsight: Boolean); Overload;
  {$ENDIF}
Published
end;
```

You will notice that the IOTANotifier interface is not specifically implemented. This is handled automatically by implementing the class with the TNotifierObject parent class which handles all those method of IOTANotifier.

You will note that the newer version of Delphi provide different overloaded version of the same methods for **BeforeCompile** and **AfterCompile**. These newer version expose additional information that these IDE provide, for instance, later IDEs trigger these method for CodeInsight, therefore if you only want to handle these methods for proper compilation then you need to ignore the calls IF CodeInsight is true.

I'm not going to go into the whole implementation of the wizard, you can refer to Chapter 1: Starting an Open Tools API Project.

Now for an explanation of the interface methods (I will not go though the IOTAWizard interface methods here as they are already covered in the above article):

- **BeforeCompile**: These methods are called before each project is compiled. Place in this method any checks or processing you want to do before the project is compiled;
- **AfterCompile**: These methods are called after each compilation. Place in this method any checks or processing you want to do after the project is compiled
- **FileNotification**: This method is called for various file operations with the IDE. The NotifyCode parameter is an enumerate that tells you what type of operation has occurred as follows:
  - **ofnFileOpening**: the file passed in FileName is opening where FileName is the name of the file being opened;
  - **ofnFileOpened**: the file passed in FileName has opened where FileName is the name of the file that was opened;
  - **ofnFileClosing**: the file passed in FileName is closing where FileName is the name of the file being closed;
  - **ofnDefaultDesktopLoad**: I haven't found when this is triggered in my test but I assume it is when the IDE loads the Default Desktop settings;
  - **ofnDefaultDesktopSave**: I haven't found when this is triggered in my test but I assume it is when the IDE saves the Default Desktop settings;
  - **ofnProjectDesktopLoad**: this is triggered when the IDE loads a project's desktop settings where FileName is the name of the desktop settings file;
  - **ofnProjectDesktopSave**: this is triggered when the IDE saves a project's desktop settings where FileName is the name of the desktop settings file;
  - **ofnPackageInstalled**: this is triggered when a package (BPL) list loaded by the IDE where

FileName is the name of the BPL file;

- **ofnPackageUninstalled**: this is triggered when a package (BPL) list unloaded by the IDE where FileName is the name of the BPL file;
- **ofnActiveProjectChanged**: this is triggered when a project is made active in the IDE's Project Manager where the FileName is the project file (.dproj, bdsproj or .dpr for older version of Delphi);

You can stop the IDE from doing anything with these files by changing the Cancel parameter of the method to TRUE.

For a real life implementation of these interfaces (from which these code was extracted) please download and look at my Integrated Testing Helper IDE expert.

regards
Dave.

Category: Integrated Testing helper Open Tools API RAD Studio Tags: Borland, BorlandIDEServices, CodeGear, Delphi, Experts, IOTAIDENotifier, IOTAIDENotifier50, IOTAIDENotifier80, IOTANotifier, IOTAWizard, OTA, RAD Studio

One thought on "Chapter 7: IDE Compilation Events"

David Post author
August 23, 2011

In this post I made a mistake and didn't highlight how to create and destroy the notifier.

In you wizard initialisation code (Register, InitWizard, etc) call

iIndex := (BorlandIDEServices As IOTAServices).AddNotifier(TTestingHelperWizard .Create);

In the finalization section of the module call

(BorlandIDEServices As IOTAServices).RemoveNotifier(iIndex);

Sorry 🙂

Comments are closed.