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

Chapter 7.1: IDE Compilation Events – Revisited…

By David | August 10, 2011 0 Comment

As a continuation of the previous post on compiler events, in Delphi 2010 onwards there are 2 new interfaces named `IOTACompileServices` and `IOTACompileNotifier` and I thought I would describe how to use it and what it does.

We'll deal with the services interface first as we'll need this to understand this to implement the other interface. The `IOTACompileServices` interface is exposed by the `BorlandIDEServices` global variable, so we can get a reference to this compile services by simply casting this variable as we've done for various other interfaces as shown below:

```
Function InitialiseWizard : TTestingHelperWizard;

Begin
  ...
  {$IFDEF D2010}
  iCompiler := (BorlandIDEServices As IOTACompileServices).AddNotifier(
    TCompilerNotifier.Create);
  {$ENDIF}
  ...
End;
```

This interface exposes the following methods:

- `AddNotifier`: This adds a `IOTACompilerNotifier` to the IDE so that it can handle compile events;
- `CancelBackgroundCompile`: This method cancels a background compilation. It waits for background thread to terminate before returning and can optionally prompt the user as to whether the background compilation should be terminated and returns True if the thread was cancelled and False if not;
- `CompileProjects`: This method compiles a list of projects defined by an array of IOTAProject interfaces supplied. The method returns `crOTABackground` if background compiling is enabled. You need to implement a IOTACompileNotifier to be informed of the result of background compilation;
- `DisableBackgroundCompilation`: This method prevent any subsequent compilation requests to the IDE

from taking place in the background thread, regardless of settings IDE settings;

- **EnableBackgroundCompilation**: This method re-enables the compilation of projects in the background thread;
- **IsBackgroundCompileActive**: This method returns True if background compilation is active else returns False;
- **RemoveNotifier**: This method removes a **IOTACompileNotifier** object from the IDE using the index number returned by **AddNotifer**.

Now for the notifier. First we need to define a class which implements the notifier interface **IOTACompileNotifier** and then get the IDE to use it as follows:

```
  TCompilerNotifier = Class(TNotifierObject, IOTACompileNotifier)
  {$IFDEF D2005} Strict {$ENDIF} Private
  {$IFDEF D2005} Strict {$ENDIF} Protected
    Procedure ProjectCompileStarted(const Project: IOTAProject; Mode: TOTACompileMode);
    Procedure ProjectCompileFinished(const Project: IOTAProject; Result:
TOTACompileResult);
    Procedure ProjectGroupCompileStarted(Mode: TOTACompileMode);
    Procedure ProjectGroupCompileFinished(Result: TOTACompileResult);
  Public
  End;
  {$ENDIF}
```

Below are some simple implementations that output messages through a custom function in ITHelper called **DebugMsg**. You could substitute another message handler as described in previous posts.

```
Const
  strCompileMode : Array[Low(TOTACompileMode)..High(TOTACompileMode)] Of String = (
    'Make', 'Build', 'Check', 'Make Unit');
  strCompileResult : Array[Low(TOTACompileResult)..High(TOTACompileResult)] of String = (
    'Failed', 'Succeeded', 'Background');

Procedure TCompilerNotifier.ProjectCompileStarted(const Project: IOTAProject; Mode:
TOTACompileMode);

Begin
  DebugMsg(Format('Compile Started (%s)...', [strCompileMode[Mode]]), Project);
End;

Procedure TCompilerNotifier.ProjectCompileFinished(const Project: IOTAProject; Result:
TOTACompileResult);

Begin
  DebugMsg(Format('Compile Finished (%s)...', [strCompileResult[Result]]), Project);
End;
```

```
Procedure TCompilerNotifier.ProjectGroupCompileStarted(Mode: TOTACompileMode);

Begin
  DebugMsg(Format('Group Compile Finished (%s)...', [strCompileMode[Mode]]));
End;

Procedure TCompilerNotifier.ProjectGroupCompileFinished(Result: TOTACompileResult);

Begin
  DebugMsg(Format('Group Compile Finished (%s)...', [strCompileResult[Result]]));
End;
```

The first bit of code in the article shows how to add the notifier to the IDE but we must remove it from the IDE on close down as follows:

```
  ...
Initialization
Finalization
  (BorlandIDEServices As IOTACompileServices).RemoveNotifier(iCompiler);
End;
```

So the question now is how do these events behave?

The order of these events are straight forward and I'll describe them for a project that has multiple projects where there are dependencies and thus multiple projects compiled at a time:

1. `ProjectGroupCompileStarted` is called first where the `Mode` parameter returning Make for a straight forward compile, Build for a Build and Check for a syntax check. I'm not sure how in the Delphi 2010 you can make a single unit, but I presume that MakeUnit would be returned for this;
2. `ProjectCompileStarted` is called before each project is compiled with the `Mode` parameter returning the same values as above;
3. `ProjectCompileFinished` is called after each project is compiled with the `Result` parameter returning whether the compilation was successful, failed or is in the background;
4. `ProjectGroupCompileFinished` is called after all the projects are compiled (regardless of whether they have succeeded or failed) and returns the overall result of the compilation, thus if a project has failed then this will return a Failure. Consequently if all projects are compiled successfully then this will return Success.

I hope this helps 🙂

Dave.

Category: Browse and Doc It Integrated Testing helper Open Tools API RAD Studio Tags: BorlandIDEServices, CodeGear, Delphi, Embarcadero, Experts, IOTACompileNotifier,

IOTACompileServices , OTA , RAD Studio