Dave's Development Blog

Software Development using Borland / Codegear / Embarcadero RAD Studio

FastMM and Unit Testing

By David | December 7, 2018 0 Comment

Overview

I've recently migrated most of my development projects to RAD Studio 10.3 Rio and one the the things I'm interested in using in my code are the in-line variable and constant declarations but before I can do that I needed to update my Browse and Doc It plug-in to be able to parse in-line declarations.

Unit Testing

So like a good developer I created the test cases before trying to implement the code and TestInsight dutifully logged the cases as missing any test. So I start to code some of the implementation and I'm looking at my code and thinking this is fairly ugly and I'm sure this is going to leak memory – ah yes I need to enable memory checking. The default `ReportMemoryLeaksOnShutDown := True` gave me bucket loads. Ah $%£&^*%! But the output didn't really help me know where. Were these existing memory leaks I've overlooked or just in the new code.

I've recently read Delphi High Performance by Primoz Gabrijlecic (and highly recommend it) and it recommends using the full version of FastMM so I downloaded it (cloned it) from GitHub. It took me a while to get it configured how I wanted it and the following is what you need to do if you want the same or similar behaviour.

- Once you're downloaded **FastMM**, you need to add its path to your project (or your library path);
- Next you MUST put **FastMM4** as the first unit in your project **.DPR** file;
- Next you need to add a conditional define of **FASTMM** to enable the the use of FastMM in DUnit or your project;
- Because I wanted to see memory leaks and a call stack to the leak I also added conditional defines of **FULLDEBUGMODE** and **LOGMEMORYLEAKTOFILE**.

With these settings, if you have a memory leak in your (test) application you will be notified when the application shuts down in a similar message to that reported by `ReportMemoryLeakOnShutDown := True`. However, if you look in the directory where the **EXE** is you will find a text file called `ApplicationName_MemoryManager_EventLog.txt`.

Now I use Stefan Glienke's TestInsight and I wanted to know if the test leaked when they are run in the background however with the above settings the memory leaks will ONLY be shown if you debug the application. The only way I found to do this is to unfortunately alter the include file in the FastMM directory as you need to undefine a conditional, something I don't believe can be configured in the IDE at project level.

The condition to be disabled is `{$define RequireDebuggerPresenceForLeakReporting}`. You can disabled it by just putting a period (.) between the opening brace and the word **define**.

After all of the above my test application churned out a bucket full of memory leaks and I thought that I might have cocked up previous code changes but as it happened, commenting out the code for the new method I was working on got rid of all the memory leaks and yes it was the piece of code I thought was suspicious.

There’s More…

The unit tests now threw 3 access violations which they didn’t before using the default FastMM that comes with the IDE. I debugged the (even more ugly) code and found that I was using a reference to an object after it had been freed by another method. A simple workaround applied and the test passed.

So for me I think that you should use the full version of FastMM in your unit test. It makes them slower with all the settings above but I think you can be assured that you code will be free of memory leaks and errors like the above. Will I use FastMM in my projects? For OTA project, no as I don’t know if it will cause issues with 2 different memory managers. In my applications? Not at the moment as I already use Eurekalog which checks for memory leaks but I think I will enabled FastMM for all my unit tests.

All I need to do now with the project is finish the parser updates with a few UI tweaks and then I need to reimplement the framework using interfaces instead of an object hierarchy (that’s not going to be quick) to clean up the ugly code.

Hope this helps.

regards
Dave.

Category: Delphi RAD Studio Unit Testing