One Last Compile...

Onwards and upwards

Sometimes I think learning Delphi is like climbing a hill. When you first start off you're rather daunted by the size of the task ahead of you. The bottom of the hill is rather slippery and difficult, and you wonder deep down whether you've got the strength of character to get all the way to the top. Then, almost before you know it, you're a reasonable way up and it's not as bad as you thought. Maybe the slope isn't quite so steep any more. And hey, the view's pretty good. Lots of things seem a lot clearer now than when you began. Sure, there's a lot of hill left, but by golly you're going to crack on to that summit. And then usually somebody comes along and asks me why I'm staring out of the window.

I think I've made a few strides further towards the summit in the last month. I've done a few things, used a few more features, and suddenly I'm aware of how much more I know than I did three years ago. It's not just the ability to be able to do things in Delphi, it's also this awareness that I've acquired about whether I'm doing things the right way or not. It's getting harder for me to do things the wrong way... no, hold on, dangerous statement, it's always *easy* to do things the wrong way. But now at least I have a little voice in the back of my head scolding me, telling me that I should *really* be putting all of this code into its own class, and gosh, isn't this code *terribly* tightly coupled to all these other units?

Example: I've just discovered the joys of the repository and, in particular, being able to store forms as templates. Which is a bit silly really, as it's been in Delphi for ages and it's certainly old hat to most of you, but I've never played with it before. (Going back to my hill metaphor, there are always masses of people miles ahead of you, and there never seems to be anybody behind.) The facility to build up a library of forms and then inherit or copy their functionality is just wonderful. One of my boss's pet peeves is having lots of forms which are slightly inconsistent in their layout, I always like to have the `Close` button somewhere different each time, and now I can just copy a standard template and all those things will be taken care of for me.

Anyway, me discovering the repository several decades after everybody else is not the point of my story. No, the thing is that having discovered the repository I then spent lots of time playing with it and, of course, soon accumulated huge numbers of forms. I discovered that components and controls that I'd placed on templates couldn't be deleted from descendant forms. The forms had quite a lot in common, but a few needed to be slightly different. Not a big problem, I could always change the visible properties on the components I didn't need and hide them that way. A bit messy, but feasible. But my little voice said what I should be doing is building a hierarchy of templates, with the stuff that was common to all forms at the very top, then introducing different variations as I went down the list. Then I could just inherit from the appropriate point in the hierarchy.

See? Very OO, very Delphi. I didn't do it, of course, because I was in a hurry for an executable (I always am) and it was quicker to shovel everything to the side of the form and make it all invisible, but nobody's perfect.

It gets better. Having acquired all of these forms, I then needed a master form to control the creation of them. It gets a bit tedious, writing all of these `MyForm := TMyForm.Create(Application)` statements. If only, I mused, there was some bit of wizardry to automate the creation of these forms. I could just pass it the name of the kind of form I wanted, it could create it and pass me back a pointer. Gosh, said my little voice, that sounds just like a factory pattern, you know, one of those patterny things that you keep reading about but haven't got around to understanding yet.

So I went and hunted around and found the copy of the Gang of Four's *Patterns* book and read about factories. I read the section twice, in fact. It's an important book, and I urge you to buy it. Sadly, I didn't understand a word, so I went back to producing endless lines of very similar but slightly different bits of code.

You may think this a tad pathetic, but the important thing for me is that I now know about OO and Patterns. Understanding them properly will take longer. And one day, I'm sure, I'll be using them. In coding, as in climbing hills, it's important not to rush these things.