One Last Compile...

If I had a hammer

As a wise but slightly irritating man once said, 'Give a man a hammer, and the world is full of nails.' I say this not merely because of a recent unfortunate incident involving my flatmate and a power drill, but because of some thoughts which struck me during the recent DCon 99 conference. *[Aha! A good place for a plug for DCon 2000, bound to be the best Delphi conference you've ever been to! Keep an eye out for the ads... Ed.]*

Conferences are usually good for introspection, for examining the way you work, the way you write code and how you can improve as a programmer and as a person. They're also an excellent opportunity to eat a lot of free biscuits, which is why I usually go home from these events a wiser but slightly more overweight person. But I digress. What follows is the result of my musings following Phil Brown's talk on Design Patterns. In passing, I should say that this was an excellent talk, as were all I attended at the conference, and for the first time I thought there might be actually something in these design pattern thingummyjigs. [*Editor, please check spelling of this.]* One of Mr Brown's more controversial statements, made as an aside to his main talk, was that he didn't use data-aware controls, and kind of wished they'd been left out of Delphi.

I blanched, as did many. But I'm beginning to think he's probably right. If we can return to my hammer and nails metaphor for a second, a Delphi equivalent of the phrase might be 'Give a man Delphi and the world is full of `DBGrids`.'

I've drawn attention to the 'problem' of data-aware controls before: previously I've expressed my horror at the thought of having to live without them. They are in many ways one of the most impressive aspects of Delphi. It was probably while the Borland folks were congratulating themselves on how neat they were that they made that rather bizarre decision to have the letters 'S-Q-L' whizzing around Athena's head in the logo. And yet, there's something about them which is kind of, I don't know, unsettling. Sure, they work, and pretty well most of the time. I love `DBGrids`. My customers love `DBGrids`, or they would if they knew that's what they're called. Certainly they would loudly lament their absence. But they also pull you down a particular path, and it may be that the path is not the best one to follow.

At this point the expectant reader is probably, err, expecting an example to prove my point. Well, I wish I could oblige. Fundamentally, I'm just talking about a hunch, y'see, not something I'm fully convinced about. But I'll do my best.

It means, for one thing, that your data is always in the shape of the underlying database. A lot of the time that will be appropriate. But sometimes it won't be, and yet it is staggeringly difficult to persuade yourself to take the trouble to move the data out into a different structure, work with it, then put it back. If it was a normal application, you'd hardly turn a hair. But in database applications with data-aware controls you get hypnotised by those little `TTable` and `TQuery` components and their datasources, and cutting off your access to them is as unthinkable as giving up oxygen. You worry, far more than is healthy or necessary, about what might be changing in the database while you're away from it.

The downside, of course, is time. Data-aware controls are quick. Writing applications without them is slow. Which is why I will probably continue to use them, even though I will hang my head in shame the next time I go to a talk by Phil Brown. If I was bold and innovative and daring, I would proclaim loudly: 'Delphi Coders of the World Unite! You have nothing to lose but your `DBGrids`!' But I'm not, so I won't, and I must leave you now to go and take my flatmate some grapes and a get-well-soon card.