ADTmag.com

Review: Visible Developer

By Mike Gunderloy

Visible Developer 4.0
$2495
Visible Systems
(800) 6-VISIBLE
www.visible.com

There are a lot of .NET code generation products out there - I've even reviewed a few of them. Of course, not every code generation tool is in the same niche. Visible Developer is designed to be at the heart of a 3-tier application development process, and it's very good at what it does.

In Visible Developer, you work at a high level of abstraction. Within the Visible Developer user interface, you've got one or more schemas, which map back to databases, and include (among other things) tables and their relationships. Built on top of these schemas are models, which include business objects, their methods, and properties. You create and manipulate these objects using a variety of tabbed dialogs. For example, when you have a rule for a business object, you specify when it's evaluated, what it's looking at, where to find the help in case of violations, and so on. (One nice touch is that you can set rules at the schema level and have them inherited to models, or set them individually at the model level).

Once you understand how the pieces fit together, this part of the product is extremely easy to use; I could see training smart users to prototype their own applications. There's also an integrated source code control system that requires you to check out objects to make changes, which is a nice safety net. The designers are fast, and although you can't extend them to include new types of information, they include pretty much everything I can think of for a typical line of business application.

After you're done modeling, you can generate the application. This is a matter of choosing an application type from the menu: choices include ASP.NET or .NET Windows Forms (either VB or C#), classic ASP, or VB6. The generation is quite fast, and based on templates (for an additional license fee you can modify or hook up your own templates). It's also possible to make it very granular. For example, you can choose to regenerate only the user interface layer for selected business objects, instead of rebuilding everything.

The code that's generated works well and has the most extensive comments that I've seen from any such product. The code is also very well designed to customized while still maintaining the ability to regenerate. When it generates code, Visible Developer inserts "edit points" where you're free to hook up your own code. For example, every class gets an edit point in its initializer. As long as you keep your own code between these comments, it won't get tromped on if you regenerate the class. Visible Developer even manages to let you customize the user interface for your forms, and open them in the Visual Studio .NET designer, without losing the ability to regenerate the code behind them (thanks to a clever use of inheritance in form files). User interface generation is flexible; among other things, you can choose to have lists and details as separate forms, or to build integrated master-detail forms that are nested several levels deep.

I looked at Version 4.0 of Visible Developer, which is in late beta; it should be released in the next week or two. This version adds some nice new features including the ability to support multiple levels of child property collections and more user interface flexibility. The most significant addition, though, may be that a single business object can integrate data from multiple schemas. If you're struggling with data scattered across disparate databases, this feature can be a lifesaver - imagine all the code that you won't have to write yourself to store everything to the right place. If you'd like to see for yourself, you can visit the Visible Web site for an online demo or to download a trial version.

Mike Gunderloy, MCSE, MCSD .NET, MCDBA is an independent software consultant and author working in eastern Washington. He’s the editor of ADT's Developer Central newsletter and author of numerous books and articles. You can reach him at MikeG1@larkfarm.com.

This article originally appeared in the
February 2005 issue of Application Development Trends.