Evaluation of Visual Fabrique (VF)

Dr Peter Lappo
www.smr.co.uk

Scope and Method

This is a review of Visual Fabrique (VF) V1.0.371 EAP Release.

In order to conduct this evaluation I followed the tutorial supplied by Jetbrains. In addition I examined the documentation provided and the sample applications to discover additional features. This not a review of the tutorial itself nor the application developed as a consequence.

What is Visual Fabrique

VF is a 4GL for Java web application development allowing the user to create database driven web applications consisting of web pages with client side scripting and server side application code.

Installation

The installation was painless on Windows XP with Java 1.4 installed. All I had to do to make my life a little easier was to create a shortcut on my desktop.

The Development Process

Developing applications in Visual Fabrique (VF) is similar to Visual Basic development as I remember it from several years ago. Pages (forms in VB) are created and laid out with controls, properties are defined, events implemented and business objects written. There is more to VF than this of course but in essence that is it.

Following the tutorial proved to be a good idea because it gave me a lot of experience with using the actual tool and getting to know its idiosyncrasies. The tutorial develops a librarian application with a set of pages to catalogue and view library books. It has user login and security and a discussion forum. Unfortunately I didn't keep an account of how long I spent going through the tutorial but I estimate that at least a days work is required as there are 14 steps resulting in 8 pages, 1 visual component, 2 database entities and 2 business services. All the steps were done manually, which is a good thing, but I did “cheat” by copying and pasting fragments of code from the tutorial. This copy and paste saved some time but constantly switching between web browser and VF was not productive and is not typical of how an application would be developed in practice. Furthermore the instructions in the tutorial weren't always obvious and I made mistakes, which again slowed me down. Fortunately there was a complete application to refer to when I got stuck.

Inevitably there were some teething problems with the tool itself because I used a beta version. For instance, the compiler error messages are hopeless as are the runtime error messages. On one occasion I'd managed to get two controls with the same name in the same page which caused a compiler error. I had to delete quite a lot of code before I found out what was wrong and of course put most of it back later! Jetbrains say they have improved the error messages. There was no undo/redo in the version I used but I expect Jetbrains will add this in a production version of this tool.

As mentioned earlier a VF application consists of pages, controls, business objects, business services and Active library components which in my opinion could be VF's strength, especially if

the user can add to this library themselves. All the usual controls are provided that you'd expect for a desktop application let alone a web environment including tabbed pane, spin control, date chooser, calendar, menu bar, table and tree controls. I thought the table paginator was particularly neat. It allowed me to handle large data sets on multiple pages with very little work. In fact VF has quite a rich set of controls which all seemed to function very effectively in Internet Explorer version 6 (I didn't try any other browsers or versions).

Interestingly custom controls and business objects were added to the palette when an Active library component was added, such as security or discussion forums. This is potentially a great saving in time as visual and data components can be quickly and easily added to an application. The user management component for instance came with a set of business objects and visual components including web pages to manage user login and registration. Some of the visual components are available at runtime within VF itself. The VF forms allow you to configure the component's data prior to starting the application. For instance, the VF forms allow you to manage groups and web page permissions. But the problem with Active visual components and indeed your own custom visual controls is that they don't display in a WYSIWYG manner making it difficult to see what the page will look like at runtime. The Active visual components also suffer from the fact that the layout of the components or the structure of their business objects couldn't be changed.

User Interface

The user interface consists of the following main panels:-

1) Web pages.
2) Application properties and events.
3) Business objects.
4) Business services.
5) Active library specific panels.

As you can see from the descriptions a model view control architecture is encouraged but of course it still depends on the developer to have a good sense of design to put things in their proper places. I particularly liked the business object screen which allowed you to design a model using UML diagrams. Attributes, methods, event handlers, queries and relationships could all be defined very easily, but despite the fact that VF uses Hibernate I couldn't find a way of mapping to my own database structure and didn't find a SQL script to generate a database. I'd expect this to change in a production release.

The business services are a good idea as it allows reusable application global services to be developed, such as the email function in the tutorial.

The web page panel is the interface you will use the most as you design your web pages here. The panel is subdivided into a Design, Xhtml and Inspector view.

The Design and Xhtml views are used to enter all the html required for the page and includes elements that hook in VF components. The Design view is supposed to provide similar functionality to a tool like Dreamweaver, except in the version of VF I used I would say it is several versions behind. Its useful but not very good by today's standards. The Xhtml view is really a code view of the Design view, and works very well and includes code completion.

The Inspector is used define the components entered in the Design or Xhtml view or to create new components. In essence it is a property editor. The Inspector has to be the poorest part of the tool and in my opinion lets down the rest of the application. The reason is that its just a very thin front end on the XML files that VF uses to store its page data. I liked the fact that the pages are designed

as components just like Tapestry. (Aside: Why couldn't Jetbrains use the Tapestry framework instead of inventing yet another one?) But defining deeply nested properties using a process of add property, define property, add property, define property with a mouse is tedious and unproductive. All the time you get the feeling that you are editing an XML file. Which of course you are! Things were made worse by the popup menus misbehaving and sometimes creating the wrong property which I expect is just a bug. Part of the problem is there are a lot of properties and events in VF and some of them can be created in different ways depending on where you want to run the code on the client or server or whether they are collections of things like controls. In fact this large number of possibilities, while providing a lot of power also produces a lot of confusion. Fortunately for me the tutorial told me what to do but I don't expect this tool to be simply picked up and used to develop production quality applications without a reasonable amount of study.

When you define properties in the Inspector code fragments are needed sometimes. For instance there were fragments of code to enable / disable visual components depending on the current user's security rights. I can understand why Jetbrains developed a special language called Fabscript to define the code fragments in properties or events as it allows for code generation to Java, Javascript or indeed C# but I didn't really like it. This was probably due to lack of documentation so it wasn't easy to figure out whether your code is correct. Although it does have code checking and code completion which made life a bit easier. I use code completion a lot when developing in Java but I also browse to the source code or examine Java Docs to get a better understanding of how to use a class. This feature is not available in VF. At first I thought Fabscript was a good idea, now I think I'd prefer to choose my own language as it would reduce the learning time and I would know exactly what to use either Javascript or Java.

VF supports the idea of web page inheritance allowing you to create a page that contains the common portions of an application like search or logon. This is essential in any larger project as it can be used to provide a level of consistency between pages.

I can't say a lot about the Active panels, they are satisfactory and served their purpose very well. The Active library is quite comprehensive and includes components for security, user management, discussion forums, blogs, email, validation, voting and localisation. Each library comes with a set of components and / or database entities. The components are either visual components that can be used on a page or complete pages that can be used as is or inherited from. However, it is clear that these libraries are a very powerful feature as significant functionality can be added very quickly to the application.

Some of the Jetbrain's refactoring technology has been incorporated into the tools so that renaming something is propagated throughout the application. This is an unusual but very useful feature for this kind of tool.

Finally I didn't try any of the “Enterprise” level features like the ability to produce EJBs, deploy to your favoured application server, or transaction management.

Computing Resources

The version of the tool I used was very memory hungry and the deployment tools were quite slow. Once an application had started there didn't seem to be a way to close it and free up its resources. I did most of my work on a 1GHz 512MB laptop and found that quite soon swapping was occurring and causing everything to grind to a halt for several seconds. At least 1GB RAM will be essential if the memory usage is not improved.

Missing Things

I felt there were several things missing in the tool or perhaps I didn't find them. There didn't appear

to be anyway of logging messages other than System.out.println. An interface to log4j would have been nice.

Neither was it possible to write unit tests. This is a big handicap if like me you write your code in a test first manner. I can see that code level unit testing may be difficult to achieve for simple one or two line code fragments, as were used in the tutorial, but the lack of automated testing will make this tool less productive when your application evolves as more time will be consumed by testing. You can of course use third party tools to conduct system level testing.

The lack of undo was also irritating but I expect this will be fixed in later releases as Jetbrain's other tools support very comprehensive undo.

Perhaps the biggest omission that I found was the lack of data mapping tools to map to alternative database structures. Although I would expect this to change with a production release as Hibernate is used under the covers which is capable of very complex data mappings.

The other nice to have would be an active library development facility as I can see that for many people the ability to create reusable web components is a particularly powerful way of assembling web applications.

Conclusion

By the time I had finished the tutorial I had had enough of the Inspector. I think I'd prefer to use an intelligent xml text editor instead. The whole process of adding and changing properties was painful. Although to be fair it is difficult to see what an alternative interface would look like given the complexity of the properties. Perhaps more custom property pages as seen in the Design view. That said the tools comes with a rich set of visual components and a useful set of library components which make up for this deficiency.

Unfortunately Visual Fabrique is no silver bullet, but nevertheless still fairly powerful and capable of saving you time. You still have to do the hard part which is business requirement analysis and writing the application code but it does take some of the grunt work out of database web application development. I personally wouldn't risk using the took on a large project or for a customer facing web site at this stage as its difficult to say whether it will scale successfully and there didn't seem to be a way of integrating it with existing databases or version control tools. Perhaps the biggest reason is that I prefer to have more control over the code in case I hit a brick wall with the application and can't code round the problem without a great deal of difficulty. However, the current version could be very successful for low risk, small scale or internal web applications until VF has been field tested more extensively.

Key Points

Strengths

1. Comprehensive set of visual controls for web application development.
2. Good set of tools for “visual” development.
3. Good set of reusable components and libraries.
4. Significant decrease in web application code.
5. Page and component inheritance.

Weaknesses

1. Web page Inspector user interface is too “mousey”.

2. Lack of mapping tools to existing database designs.
3. No unit testing.
4. Web page Design view is primitive by today's standards.
5. Lack of version control and undo.
6. Large complex tool that will require significant knowledge to get the best out of it.