JOURNAL OF OBJECT TECHNOLOGY

Online at www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2003**

Vol. 2, No. 2, March-April 2003

Product Review

JBuilder 8

Reviewed by David Neuendorf and Richard Wiener

JBuilder 8, like its predecessors, is a large and powerful product. Because of time constraints, it is not our intention to review and discuss every aspect of this product but focus only on the features that interested us the most.

JBuilder 8 now uses JDK 1.4.1 as its default version of Java but allows one to choose JDK 1.3 or some other virtual machine in its place.

A most welcome new feature of JBuilder 8 is a wizard that automates the process of constructing a JBuilder project from existing Java source files. If deployment descriptors are provided in the web-inf directory or if EJB deployment descriptors are provided, JBuilder will quickly construct a new project from this raw input. This was tested using a suite of Java source files. Using the File/New/Project/Project for Existing Code icon, a project was assembled around the existing Java source files in a matter of seconds. For previous users of WebGain's VisualCafe, JBuilder now allows a VisualCafe project to be imported and translated into a JBuilder project.

On the J2EE development front, JBuilder 8 allows a developer to select a framework to create a web application. The choices include Borland Enterprise Server, BEA WebLogic Server, IBM WebSphere, Sun ONE and Sybase EAServer. Borland indicates that other server integrations are available from the Borland Developer Network (http://bdn.borland.com). EJB development has been streamlined and made more efficient for large EJB projects. Unit tests using EJB clients based on JUnit are now possible and supported by user-friendly and useful wizards. Web service wizards now support runtimes based on SOAP, Axis and BEA WebLogic Native Web Services.

There is a UDDI (Universal Description, Discovery and Integration) explorer that allows a user to search the internet for services by company, service type, and other criteria. The explorer also makes it possible to publish a web service.

Once a desired service has been identified, there is a wizard to create Java source files for an application to consume the service, or even to create a similar service. There is also a wizard for exporting any existing Java class as a service.

JBuilder 8 provides useful support for version management using a CVS repository. Once a project is checked into a version management system, a History tab allows us to view different versions of an application, revert to an older version or examine

Cite this review as follows: David Neuendorf, Richard Wiener: Review of JBuilder 8, in *Journal of Object Technology*, vol. 2, no. 2, March-April 2003, pp. 205-208.
http://www.jot.fm/products/review4

differences between versions, show the revision information and work with merge conflicts.

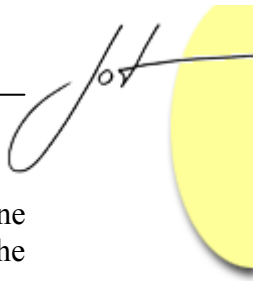
One of the most important features of JBuilder is (and has been) its powerful Java editor. Support for more formatting options has been added to the current editor. Parenthesis highlighting is a most welcome feature. The new line numbering capability might be a feature that is useful to some developers. The new drag-and-drop editing allows code to be moved around while preserving local formatting and indentation. Since the IDE uses JDK 1.4, the mouse wheel now functions properly in the editor. On the down side, the powerful code formatting features of the JBuilder editor do not apply to html or jsp code. There was no intelligent indentation of tags, and even the tab key did nothing, because it had been mapped by default to the code format function of the editor, while the standard tab functionality had been mapped to the F2 key.

JBuilder 8 adds several re-factoring tools to those available in earlier versions. The re-factorings seem to have been selected both for the amount of work to be saved by automating them, and for how error-prone the manual change would be.

- To extract a method from a block of code, select the lines of code and invoke the tool. It will prompt for a method name, then automatically determine the necessary signature based on the code selected.
- The "introduce a variable" tool can replace a complex expression wherever it occurs in a method with a temporary variable.
- Method parameters can be added, moved or re-ordered. All references to the method are changed as well.
- A block of code can be automatically surrounded with a try/catch block. The refactoring tool determines the types of exceptions that can be thrown by the code, and adds a catch block for each one. If a variable that is used outside of the try block is declared in the selected code, the declaration is moved above the try block.
- The editor can manage import statements automatically. It can create either class or package import statements, and remove obsolete imports.
- We have been on projects where it was necessary to change the package structure of an application. This was a painful and error prone process. With JBuilder 8, this job is a simple matter of changing the package specification in a dialog box. In a few seconds, JBuilder creates any necessary directories, changes package and import statements, and moves source files around as needed to complete the structure change.

The Enterprise Performance Pack edition of JBuilder 8 comes with Borland's Optimizeit 5.0 Suite product integrated into the IDE. A program can be profiled either in the IDE or externally in the standalone version of Optimizeit. Included are a code coverage tool, a thread debugger and memory and CPU profilers.

Testing the CPU profiler both inside and separately from the IDE, we found that things were a bit cramped in the IDE, so we preferred running Optimizeit standalone. The test program was a large application that generates technical drawings of products in



several different product lines. The code took much longer to generate a drawing for one product line than for any of the others. We used Optimizeit to investigate the cause of the slowdown.

The Optimizeit CPU profiler's display made it easy to find the problem. There was a display of all threads that had been active during the session, color coded to indicate time periods in which each thread had been active. Our problem occurred in the event thread. When we selected this thread, we were shown a call tree, in which we selectively opened branches based on the CPU time percentage shown on each branch. This allowed us to quickly drill down to find the lowest level code consuming large CPU time. In just a few minutes, we found that a clipping algorithm was performing poorly when arcs were being clipped by other arcs. The problem products had a circular cross section, with many circles clipping other circles. The exact methods that needed work were readily identified.

One of the regrets we expressed in our review of JBuilder 6 was the lack of any support for the Struts framework. JBuilder 8 has extensive support for Struts. It has wizards to convert existing JSP Model 1 (non-MVC) web applications to use the Model 2 Struts MVC architecture, and to generate all of the necessary types of files: JSP, Actions, ActionForms, and the configuration files. There are wizards to extract the fields from a form in an existing JSP file to create the getter and setter methods in an ActionForm, and *vice versa*. The struts-config.xml file can be easily edited using the provided Struts Config Editor. Once developed, a Struts application can be tested and debugged within the IDE with the integrated Tomcat server.

The provided tools do a pretty good job of automating the mindless work involved in developing a Struts application. There are still some things the tools should have been able to figure out from information that the user has entered once. For example, when creating a JSP file named Lookup.jsp based on an ActionForm named LookupForm, JBuilder ought to be able to figure out that the form tag's action attribute should be "Lookup" and generate a form tag like `<html:form action = "Lookup">`. Instead, what it generates is `<form method = "post">`. In spite of this minor gripe, JBuilder will make it much easier to develop and maintain Struts applications.

The JBuilder help has no Struts development tutorial, but there are detailed instructions about the steps used to create an application. We were able to duplicate an example from *Mastering Jakarta Struts*, by James Goodwill (Wiley Books, 2002) with no problems.

So as before and in summary, we greatly appreciate the many new and outstanding software development features introduced in JBuilder 8. This product has been evolving and improving over many years and has become a class act.

About the authors

Dave Neuendorf is a Java consultant and President of NeuSys, Inc. (www.neusysinc.com) of Aurora, Indiana. With degrees in chemistry and materials science, he converted his programming hobby into a second career. A Sun Certified Java Developer, Dave has been working with Java since 1996 and JDK 1.02. He specializes in user interface development using Swing. You may contact him at dwn@neusysinc.com.

Richard Wiener is Associate Professor of Computer Science at the University of Colorado at Colorado Springs. He is also the Editor-in-Chief of JOT and former Editor-in-Chief of the Journal of Object Oriented Programming. In addition to University work, Dr. Wiener has authored or co-authored 21 books and works actively as a consultant and software contractor whenever the possibility arises.