2005 JAVAPro EDITORS' CHOICE WINNER
Editors' Choice Awards Selected
FTP's Tech Trivia Contest
Over $20,000 in Prizes
www.ftponline.com/15th

JAVAPro™

www.javapro.com A Fawcette Technical Publication 2005 Vol. 9 No. 5

FTP FAWCETTE TECHNICAL PUBLICATIONS 1990-2005 15TH ANNIVERSARY

Attracting Platform Integration

SOA, Web Services Bind J2EE and .Net for Composite Applications

Employ SOA Frameworks to Build Cool Interfaces

Handle J2EE's Complexity With Deployment Descriptors

How Modeling Impacts Your Development Team

Discover the JFace Toolkit

Lead the Way to Professional Programming Success

One Source for All Your

CHANNELS
To better serve your needs, FTPOnline has been restructured around seven channels: Architecture, Java, .NET Development, Windows IT, ASP.NET, Database and Security. More channels to come!

1

SPECIAL REPORTS
Get comprehensive information on subjects critical to all IT professionals, such as Security, Service-Oriented Architecture, and Operations Management.

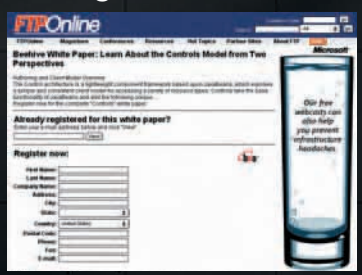
2

WEBCASTS
Watch and listen to industry experts discuss hot IT topics.

3

WHITE PAPERS
Download white papers that examine evolving technologies.

4



NEWSLETTERS
Free e-mail newsletters in your area of interest, delivered right to your inbox.

5

FTPOnline

2005 Fawcette Technical publications, Inc. All product names herein are the properties of their respective owners.

Technical Information

Newly Expanded, Easily Accessible

6 **RSS FEED**
Get quick updates on the latest blogs and articles published at FTPOnline.

7 **MAGAZINES**
Filled with down-loadable code, interviews with industry visionaries, in-depth tutorials, overviews of implementation and management strategies, article archives, and more!

Go there today:

www.ftponline.com

JAVAPro™

COVER STORY

FEATURES

COLUMNS

DEPARTMENTS

Java Pro (ISSN: 1096-4495) is published bimonthly with two additional special issues in June and October for the basic rate of $34.97 per year by Fawcette Technical Publications Inc., 2600 South El Camino Real, Suite 300 San Mateo, CA USA, 94403. Tel. (650) 378-7100; fax (650) 570-6307. Send address changes to *Java Pro*, P.O. Box 3485, Northbrook, IL 60065-9819. Periodicals Postage Paid at San Mateo, CA and at additional mailing offices. ©2005 Fawcette Technical Publications Inc., all rights reserved. *Java Pro* is an independent publication not affiliated with Sun Microsystems. Sun Microsystems is not responsible in any way for the editorial policy or other contents of the publication. All contents of ***Java Pro*** are copyright ©2005 by Fawcette Technical Publications Inc., unless otherwise noted. "VBITS" and "Interactive Developer" are trademarks of Fawcette Technical Publications Inc., a California Corporation, James E. Fawcette, President. Java is a trademark of Sun Microsystems. Rather that put a trademark symbol in every occurrence of other trademarked names, we state that we are using the names only in an editorial fashion with no intention of infringement of the trademark. Although all reasonable attempts are more to ensure accuracy, the publisher does not assume any liability for errors or omissions anywhere in the publication. **Canadian CPM: 40013204; Customer: 2418150; Return Address: 4960-2 Walker Road, Windsor, ON 49A 6J3, Canada. ISSN: 1096-4495.**

2005 VOL. 9, NO. 5
PUBLISHER, Jeff Hadfield

EDITORIAL
EDITOR, Kay Keppler
EDITOR, Terrence O'Donnell
TECHNICAL EDITOR, Daniel F. Savarese

CONTRIBUTORS
Kevin Jones, Daniel F. Savarese, and Peter Varhol

ART & PRODUCTION
VICE PRESIDENT, ART & PRODUCTION, Michael Hollister
SENIOR ART DIRECTOR, Bruce Gardner
PRODUCTION MANAGER, Kathleen Sweeney Cygnarowicz
ART DIRECTOR, Brian Rogers
SENIOR PRODUCER, Richard Kwong
ASSOCIATE WEB PRODUCER, Shane Lee
WEB AND PRINT GRAPHIC DESIGN INTERN, Liz Dreessen

ADVERTISING SALES
EXECUTIVE ASSISTANT TO THE VICE PRESIDENT, PUBLISHING, Susan LaCroix

CIRCULATION
SENIOR CIRCULATION DIRECTOR, Karen Koenen
CIRCULATION MANAGER, Fred Perry
CONFERENCES ASSOCIATE, Gerry Guzman

MARKETING
MARKETING MANAGER, Susan Ogren
SENIOR DESIGNER, Margaret Horoszko

CONFERENCES
VICE PRESIDENT, Tim Smith
CONFERENCE MARKETING MANAGER, Michael Coleman
SALES OPERATIONS MANAGER, David Seymour
CONFERENCE OPERATIONS PLANNER, Will Hansen
MARKETING AND EDITORIAL PLANNER, Sean Cassidy
MARKETING INTERN, Maureen Silva

CUSTOMER SERVICE
CUSTOMER SERVICE REPRESENTATIVE, Jose Porcell

OPERATIONS
EXECUTIVE VICE PRESIDENT/CHIEF FINANCIAL OFFICER, John Sutton
SYSTEM ADMINISTRATOR, Tin Cao
DIRECTOR OF FINANCE, Darlyn Phillips
ACCOUNTS PAYABLE ACCOUNTANT, Elena Ostrovsky
STAFF ACCOUNTANT, Betty Tsang-Hwah Wu
ACCOUNTS RECEIVABLE, Iain Niellands
HUMAN RESOURCES MANAGER, Pam Davis

FTPONLINE
MANAGING EDITOR/BUSINESS UNIT MANAGER, Nina Goldschlager
E-ADVERTISING MANAGER, David Seymour
EDITORIAL PROJECT MANAGER, Nick Fuentes
REGIONAL SALES MANAGER, Les Heizler
EAST COAST SALES MANAGER, David Lyman

FAWCETTE TECHNICAL PUBLICATIONS
PRESIDENT, James E. Fawcette
VICE PRESIDENT, CHIEF INFORMATION OFFICER, Aaron Weule
VICE PRESIDENT, PUBLISHING, Jeff Hadfield
CORPORATE COUNSEL, Wilson, Sonsini, Goodrich & Rosati

JAVAPro online

The ONLY place on the Web for *Java Pro* content, code, and community is www.javapro.com

Visit *Java Pro* online for this extended content and more.

FTPOnline Blogs

Check out the FTPOnline blog page to get insights from Jeff Hadfield, vice president, publishing, and other FTP editors sounding off on IT and development issues. *www.ftponline.com/weblogger/*

WebSphere Family Grows

Locator+ Code: JF_050826_1

HTTP://WWW.FTPONLINE.COM/WEBLOGGER/FORUM.ASPX?ID=1&DATE=08/26/2005&BLOG=#429

Does everything at IBM and IBM Rational get the WebSphere name? As the platform's family of products continues to grow, Jim Fawcette provides some clarification about what is being integrated along with a comprehensive list of WebSphere and IBM Rational offerings.

Eclipse Conference Brings Together People, Processes

Locator+ code: PV_050925

HTTP://WWW.FTPONLINE.COM/WEBLOGGER/FORUM.ASPX?ID=11&DATE=09/25/2005&BLOG=#440

by Peter Varhol

Java Pro's Object Enterprise columnist Peter Varhol attended the recent Eclipse Members Conference in Chicago, and he reflects on the unique perspective this group has on the Eclipse Foundation and the rest of its membership. Read what Peter has to say about the foundation's focus from his perspective and what it means for the Java developer community.

More Online Exclusives

Who Needs Struts?

Locator+ code: JP050928LB_T

by Lew Bloch

Like other important frameworks, Struts has its place in the Java development community. However, sometimes for smaller projects its better to write a few lines of code by hand rather than to incur a major framework's overhead. Take a look at the simple techniques you can use to take advantage of the full power of the MVC pattern, without the extra XML, third-party libraries, opaque servlets, or the other downsides frameworks can pose for smaller-scale projects.

What Are Locator+ Codes?

Locator+ codes give you instant access to a feature on *Java Pro* Online. Simply type the Locator+ code into the field in the upper-right corner of the page, and click on the "go" button.

Locator+ Code: jf_040510 go

JAVA INSIGHT NEWSLETTER SIGN-UP

www.javapro.com

Every week, the *Java Insight* e-mail newsletter brings you up-to-date news, technical information, opinions, interviews, and analysis on topics and technologies such as J2EE, middleware, and application servers; J2ME, devices, and embedded development; data access; servlets and JSP; Web services; and/or XML. Sign up for *free* at www.javapro.com.

Full Speed Ahead

by Terrence **O'DONNELL**

The editors at Fawcette Technical Publications (FTP) have been busy hosting the Java Pro Live! (JPL) Conference in San Diego last September, hosting the 2005 Java Technology Roundtable at JPL, selecting our Editors' Choice Awards, and of course keeping these pages filled with valuable content.

This year's JPL followed 2004's premiere event in Boston with a Left Coast version that provided a good mix of activities covering the cutting edge of Java-related technologies in the areas of development, architecture, and management. Lee Nackman, IBM Rational's vice president for product development at Rational, kicked off the event with a keynote on the aligning, or coming together, of Java, Eclipse, and service-oriented architecture (SOA) by telling attendees that, "the IT world is moving from a technology-driven world to a business-driven world." Nackman provided a discerning view into the increasing role of business processes in the development of enterprise applications. The alignment of these key technologies, he said, marks a new trend in which business people are driving more decisions about the kinds of IT projects organizations are investing in.

This trend—promised for many years—is creating new challenges and opportunities for organizations. SOA is prominent in changing the way we think about delivering business processes in the design, building, implementation, and deployment of enterprise applications, and Ted Farrell, chief architect, application development for Oracle, addressed these challenges in his JPL keynote on development, WSIF, and metadata frameworks. Farrell explained that you can employ these frameworks to build SOA applications that meet the goals of the applications you want to build. In a preconference interview (visit www.ftponline.com/channels/java/reports/javaprolive/2005/farrell/), Farrell noted the mettle of the J2EE platform for SOAs: "It's the thing you're building your business off of, and that's where J2EE really shines. It's an incredibly mature, robust platform. It has been for years, and there's tens of thousands of enterprise applications running on this architecture today. It really gives you the foundation for building SOA applications."

The fifth annual 2005 Java Technology Roundtable proved to be another JPL highlight. The distinguished panel—David Chappell of Sonic Software, Farrell, Tyler Jewell of Quest Software, Nackman, and Tim Stapleton of Beyond Browsing—met in a general session to discuss the Java issues of the day. In past years, FTP hosted this event offsite during Sun Microsystems' JavaOne conferences. This year we presented the roundtable before a live audience, giving conference attendees a real-time opportunity to hear what these experts had to say about the current status of Java technologies and what to expect in 2006. Read a recap of the event at the FTPOnline.com Web site (www.ftponline.com/channels/java/reports/javaprolive/2005/roundtable/).

We also announced the 2005 *Java Pro* Editors' Choice Awards at JPL. When tasked with choosing products for these awards, the editors, including selected authors and columnists, pondered the numerous tools, IDEs, platforms, suites, and comprehensive integrated solutions designed to work with the complexities of enterprise-scale processes. Then there were the myriad products for more granular application life-cycle processes: reporting and testing, graphics, JVMs, optimizers, and many others. We recognized immediately that the task was daunting, and better left to those—you, the readers—who use the products daily to judge them for specific functionality for the upcoming Readers' Choice Awards.

The editors decided instead to acknowledge broader offerings that correspond to the primary coverage areas we focus on in these pages: development, architecture, and platform. We also chose one for innovation in the enterprise. We considered products from vendors who we identified as being perennial industry leaders providing noteworthy benefits to Java professionals and exhibiting market leadership. The winners are the Eclipse SDK for best overall development product, BEA WebLogic Workshop for best overall architecture product, SAP NetWeaver for best overall integration product for the platform, and IBM WebSphere for best product exhibiting innovation in the enterprise stack. Congratulations to these vendors. See the In Brief column in this issue, and visit the FTPOnline.com Web site (www.ftponline.com/channels/java/reports/javaprolive/2005/awards/), for detailed summaries of the selection process and winning products.

Finally, Kay Keppler, who served this publication well for many years, has moved on to other editorial endeavors. It's now my turn to take the helm at *Java Pro*. The names may have changed, but the mission is still intact: to provide IT professionals that use Java-based platforms and technologies pragmatic, technical information and implementation guidance for developing and deploying strategic solutions. I encourage you to e-mail me with your thoughts and suggestions to help us continue to fulfill this objective. *JP*

Terrence O'Donnell, Editor
todonnell@fawcette.com

Editors' Choice Awards

Award-winning products announced at Java Pro Live! reflect Java technology innovation

Choosing Editors' Choice Award-winning products wasn't easy for the editorial staff of *Java Pro*. The Java marketplace is rich with many tools, IDEs, platforms, suites, and comprehensive integrated solutions, let alone the many products for more granular aspects of the application life cycle, such as reporting and testing, JVMs, and others. Scrutinizing so many across a range of categories can translate into substantial overlap, and selecting products for specific functionality is best left to readers for our Readers' Choice Awards.

For the Editors' Choice Awards we acknowledged products that correspond with our coverage areas in the magazine, narrowing our selection to categories for development, architecture, platform, and then one for a product that best reflects innovation across the enterprise stack. We considered products from vendors we identified as being perennial industry leaders and then cast votes to determine those that best provide noteworthy benefits to Java professionals, exhibit market leadership, and demonstrate innovation. We announced the winners in September at our Java Pro Live! Conference.

Best Overall Development Product, Eclipse SDK. The Eclipse platform continues to demonstrate widespread acceptance for building IDEs that can be used to develop embedded Java programs, Web sites, and other applications. It delivers mechanisms exposed through APIs, classes, and methods; useful frameworks; and the rules for building seamlessly integrated tools. Its phenomenal growth and the many benefits developers derive from this open source platform make the Eclipse SDK a solid development product.

Best Overall Architecture Product, BEA WebLogic Workshop. Organizations are increasingly looking to take business processes to the next level by migrating to SOAs and building enterprise-scale applications. BEA's WebLogic Workshop offers a solution for reducing the complexity of SOA migration in addition to the overall lifetime costs of IT infrastructures. Developers and architects can use its Java Controls feature to connect to and use IT assets, while hiding the complexity inherent in the J2EE platform when making such connections. WebLogic Workshop provides for customizing controls and adding to them the business logic necessary for reusable, composite components.

Best Overall Integration Product for the Platform, SAP NetWeaver. When developing or considering a business strategy for optimizing both the supply-side and the demand-side of a business, organizations require a comprehensive integration and application platform capable of enabling and managing change. SAP NetWeaver offers a platform that can work with existing IT infrastructures to allow IT professionals to design, build, and execute business strategies and processes. With its enhanced Java development infrastructure and J2EE support, SAP NetWeaver provides a standards-based middleware solution that drives innovation and combines existing systems.

Best Product Exhibiting Innovation in the Enterprise Stack, IBM WebSpere. Organizations looking for a comprehensive set of software tools that support the entire application development life cycle can achieve that goal with the IBM WebSphere platform. It comprises a vast collection of integration and application infrastructure software that enables architects and developers to design, build, test, deploy, and manage enterprise-scale applications for traditional mainframe, Java, or composite applications. The platform covers all aspects of the J2EE application life cycle, and there is arguably no broader or deeper stack for managing the application life cycle and its associated processes.

HOW TO REACH US

Editorial Offices: Fawcette Technical Publications, 2600 South El Camino Real, San Mateo, CA 94403; Phone: (650) 378-7100; Fax: (650) 570-6307; Web: www.javapro.com; e-mail: javapro@fawcette.com.

Article Submission: To submit articles for publication please contact Terry O'Donnell, Editor, todonnell@fawcette.com. Download the Author Guidelines at www.ftponline.com/javapro/code/12dec01/author_guide.zip.

Product Announcements: To submit announcements for new products or updates to existing products, please e-mail press releases to Terry O'Donnell, Editor, todonnell@fawcette.com.

Reprints and Permissions: For all *Java Pro* editorial and advertising reprints contact Serv U Reprints, 101 Rhoades Way, Folsom, CA 95630; Phone: (916) 983-6562; Fax: (916) 983-6762.

To Quote from an Article: Please contact Susan LaCroix, 2600 South El Camino Real, San Mateo, CA 94403; Phone: (650) 378-7118; or e-mail: slacroix@fawcette.com. Specify the issue date and title of the article, the portion you would like to quote, and the purpose.

Photocopy Rights: Permission to photocopy for internal or personal use may be granted by Fawcette Technical Publications. Please contact Susan LaCroix at slacroix@fawcette.com for more information.

Customer Service and Subscription Information: For subscription orders, inquiries, or address changes please contact Customer Service, *Java Pro*, P.O. Box 3485, Northbrook, IL 60065-9819; Phone: (866) 387-5776; Fax: (847) 291-4816; for international inquiries call (847) 559-7309; or e-mail jva@omeda.com. Foreign and Canadian orders must be payable in U.S. dollars, plus postage. The surface rate to Canada and Mexico is $52.97 per year. For all other countries the air mail rate is $78.97 per year.

Back Issues: To order *Java Pro* back issues, call (650) 378-7100 or (800) 848-5523 and ask for Customer Service. Back issues cost $10. Additional postage will be charged to deliveries outside the USA.

Formula One e.Spreadsheet Engine

API-driven, embedded, 100% pure-Java, scalable spreadsheet toolset

Spreadsheets play an essential role in the business world. And now, they can also perform a vital function in your Java applications. How? With the Formula One e.Spreadsheet Engine, an API-driven, 100% pure Java toolset for embedding Excel spreadsheet functionality in your applications.

Excel-enable your Java applications

Use a state-of-the-art graphical development environment to build fully-formatted Excel report templates that include formulas, functions, colors, merged cells, even charts. It's fast, easy and effective.

Embed live, Excel-compatible data grids

Include interactive Excel forms and reports in your Java applications. Let users manipulate them using their spreadsheet skills and then commit the changes to databases or applications – or save them as an XLS file on their desktops.

Automate complex calculations and rules

Embed Excel spreadsheets that automate complex calculations and business rules on J2EE servers. Stop translating spreadsheet logic into Java code today, and start leveraging the development skills of your organization's spreadsheet experts.

Read and write server-based Excel spreadsheets

Give your users server-based spreadsheets populated with up-to-the-minute information directly from data-bases, XML files and enterprise applications. Get control of runaway data warehouses and spreadmarts now.

Make spreadsheets a part of your strategies

Visit us today at www.reportingengines.com to request a free trial of the e.Spreadsheet Engine. We'll show you how to make Excel spreadsheets a vital and productive part of your enterprise computing strategies.

ACTUATE
ReportingEngines

www.reportingengines.com
sales@reportingengines.com
888-884-8665 +1-913-851-2200

FREE TRIALS, DEMOS AND SAMPLE CODE

Adversaries and Partners

Do development and deployment technologies on the J2EE and Microsoft .Net platforms compete, or is there plenty to go around?

by Chris **SCHALK**

We hear a lot of discussion these days about service-oriented architecture (SOA), seamless integration, interoperability, Web services, and business logic and processes and what they mean for enterprises. Many resources document the comparisons and distinctions of the development and deployment technologies on both Java 2 Platform, Enterprise Edition (J2EE) and Microsoft .Net. Are these two platforms really competing against each other? Or, are they both integral to achieving interoperability in the enterprise? Let's take a look at both.

Go Online

Visit www.javapro.com for related resources. Simply type the Locator+ code into the field in the upper-right corner of the page.

Download

JP0509 Download all the code for this issue.

Discuss

JPTALK Discuss this article in the "Talk to the Editors of *Java Pro* Magazine" discussion forum.

Read More

JP0509CS_T Read this article online.

VS0506AN_T Read the related article "Customize Sites with Web Parts" by Peter Vogel.

EA0501AK_T Read the related article "Enterprise Integration with Web Services and BPEL" by Adam Kolawa, Ph.D.

JPEP030311BK_T Read the related article "Did .Net Really Beat J2EE Flat? Can It?" by Budi Kurniawan.

One long-held stereotype for Java and J2EE developers, in fact, is that they tend to be more technically inclined and have considerably more core programming expertise. This sentiment is bolstered by a common belief like: "they have to be more skilled in computer science to work with the complexities of J2EE!" Is J2EE a technology meant only for computer scientists?

While it is true that there are complexities involved in many facets of J2EE technology, the same could be said about some of the advanced aspects of .Net. Another reason why J2EE sometimes gets a bad rap for complexity is that it is largely seen as a server-oriented technology, which is definitely a more difficult area of computer science in general. Typically, this technology involves developing nonvisual, back-end code to process large amounts of data such as back-end business logic for large-scale, industrial applications.

However, to J2EE's credit, it has recently made significant strides into more client-related software development. One key, recent change in J2EE has been a rise of increasingly powerful Java integrated development environments (IDEs) as well as some key new Web client development technologies that are geared more toward *corporate developers* as opposed to the traditional *systems* or *component developers*.

The Rise of J2EE IDEs

In the past few years Java and J2EE developers have had an ever-improving set of tools to help them develop a vast array of J2EE technologies. More sophisticated and powerful Java IDEs have lowered the technical bar for Java developers and allowed a larger population of business developers into the Java arena. While Java IDEs aren't necessarily a new phenomenon, recently there has been a marked increase in developer productivity provided in the latest Java IDEs. For example, some of the new tools such as the open source Eclipse and NetBeans along with Oracle JDeveloper and Intellij Idea now provide a long list of extremely powerful Java productivity development features (see Figure 1). These features include code refactoring, code completion, automatic indenting and formatting, collapsible edit regions (or code folding), and code profiling. The new, more powerful IDEs also have features that provide integration with many external open source development technologies such as Apache Ant, CVS, and Jakarta Struts.

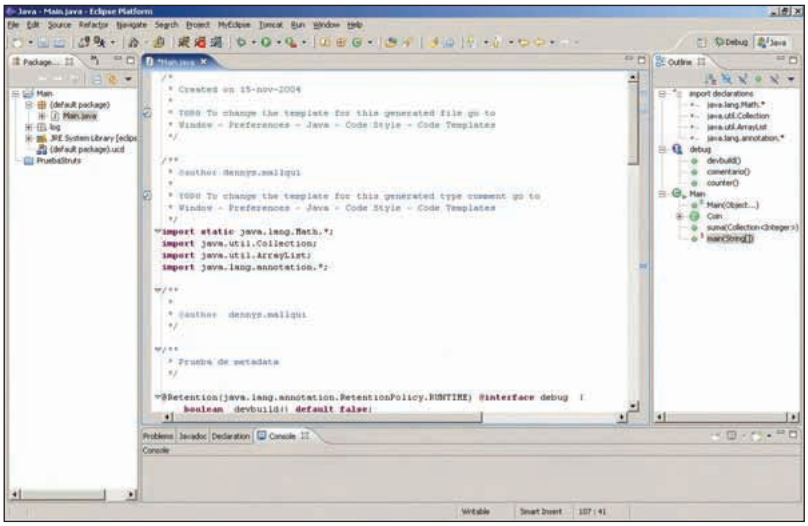


Figure 1 | Open Source IDE Eclipse 3.1 provides an open source IDE that is jam-packed with powerful Java productivity development features.

In addition to offering core Java coding, Java IDEs have branched out and begun offering more visual development tools, including support technologies like Unified Modeling Language (UML), Struts Page Flow, and JavaServer Pages (JSP) visual design (see Figure 2).

The other important event that has increased awareness of J2EE IDEs is that they are either free or are very cheap. Both the core development platforms Eclipse and NetBeans are open source and are free. However, more feature-complete IDEs that are built on top of the Eclipse platform are for sale, though usually at modest prices. In a recent announcement at JavaOne 2005, Oracle announced that its Java IDE, JDeveloper, is now completely free of charge.

One of the newest areas of J2EE technologies to help break the mold of J2EE being a nonvisual, server-centric technology is JavaServer Faces (JSF). JSF is a new Web user interface development framework that is part of the standard J2EE technology. Its goal is to provide a single, standard, and component-centric way to build Web user interfaces productively for J2EE applications. This J2EE technology has also led to great innovations with J2EE IDEs, with new visual development support now being offered in several Java IDEs. We'll discuss a more thorough comparison shortly, but first let's have a look at what's currently hot with the .Net technology.

Is .Net Always Easier?

Looking to stay in the lead in the traditional client-oriented software development arena, Microsoft has been busy working on their entire .Net development infrastructure with its Windows Vista project (formally known as LongHorn). Windows Vista has been in development for a long time (since 2003), but it's nearing completion and is expected to be released in 2006. Some of the new technology currently being beta tested and previewed is Microsoft's next-generation IDE: Visual Studio 2005.

Visual Studio 2005 is Microsoft's slickest development tool yet. It is armed with many developer-productivity features. The new ASP 2.0 visual controls provided in Visual Studio 2005 simplify many common Web development tasks by reducing substantially the amount of manual coding required by the application developer. Another big feature in the new .Net release is its new team development features where the entire life cycle of development, from initial design, prototyping, coding, and unit testing to completion, is handled in a distributed environment.

Microsoft has always strived for lowering the technical bar for the larger development community of corporate developers; however, when it comes to more advanced development, people still contend that Microsoft enterprise development can be just as complex as J2EE. Another important factor is that although there are some very compelling development features provided with

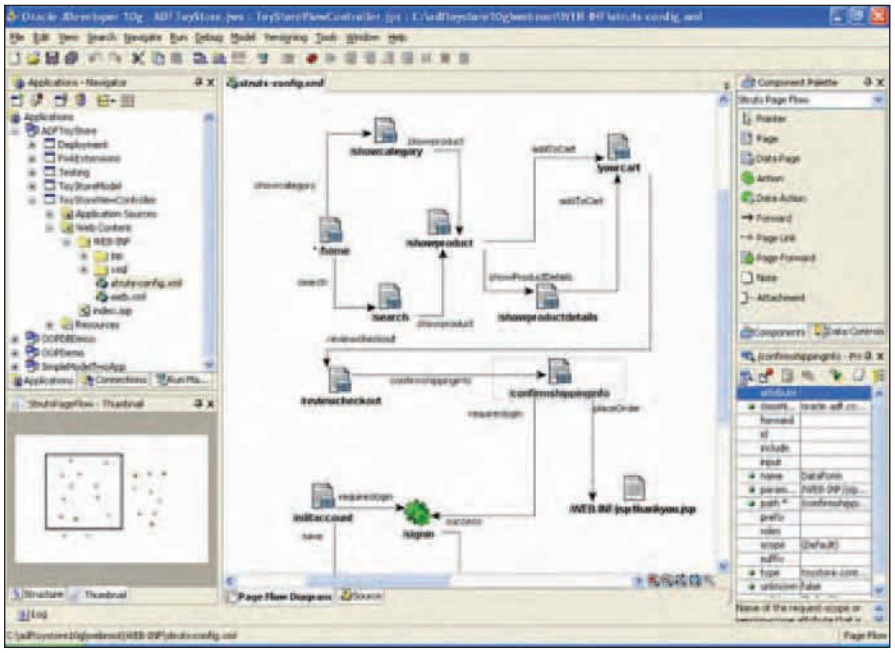


Figure 2 | Visual Paging Oracle JDeveloper's Struts visual page flow editor is one example of how Java IDEs are now offering more visual development tools.

Microsoft's latest offerings, developers have to be comfortable with the fact that their applications will reside entirely on a Microsoft technology stack. The only exception is when Web services are used to communicate out of the stack to other services that could be running on any platform. We'll discuss more about this topic shortly.

To sum up, both the J2EE and Microsoft camps have been extremely busy retooling their architectures to enable larger portions of development communities. J2EE has a new host of J2EE IDEs that offer tremendous features for little or no cost, while Microsoft is banking on delivering a "best of quality" technology suite that all works together in a cohesive manner—providing its Microsoft technology.

Now that we've discussed some of the key aspects of both J2EE and .Net, let's take a look at some of the technologies that are very similar on both platforms and see exactly how each stacks up against the other. To begin this in-depth comparison, let's look at the component technologies JSF and Microsoft ASP.Net by comparing them directly and their respective development environments.

Since ASP has been available in the public for a longer period of time, there has been a large external development community providing an increasing amount of Web components, or to use Microsoft's terminology: *controls*. With the Vista release, Microsoft is also revamping their component/control technology with ASP 2.0. It is intended to be even easier to develop with than previous versions of ASP. The overriding theme for ASP 2.0 is "less coding," and less coding is achieved by making the controls themselves more intelligent, thus reducing the amount of manual coding required to customize.

This direction is logical: can you imagine if we still had to start our car engines with a hand crank? An example of ASP 2.0's improvement is evident with their new GridView control. It is more powerful than its predecessor, the DataGrid, and requires less manual coding to build a grid of data, which could be from a variety of sources, that can be scrolled, paged, sorted by column, and edited. In general, GridView, along with the other 2.0 controls, is enhanced with the vision of better productivity and less manual coding. The controls also use Visual Studio's Smart Tasks popup editors, which make them easily customizable.

In addition to ASP 2.0 enhancements, ASP has always benefited from a thriving development community that has provided many libraries of useful complementary controls in many different areas, including charting, imaging, XML, forums, and many others. These ASP libraries are easily found on Microsoft's official ASP.Net Web site (see Resources online at www.javapro.com), which serves as a vast registry of numerous rich and diverse ASP controls and affiliated ASP community sites. In short, the ASP development community is thriving.

For the Business Developer

As mentioned previously, JSF is J2EE's new, standard Web user interface technology. Its goal is to simplify the often complicated task of building a robust Web user interface to the back-end business logic by offering a component-centric development approach. Working with intelligent Web components, as opposed to writing low-level code to construct Web applications, opens J2EE Web development to new audiences who may be slightly less technical but are more focused on the business aspects of application development.

The introduction of the JSF API in early 2004 offered both a specification and a reference implementation. The reference implementation included a set of base Web components (UI components) that covered the basics of Web development, in that they provided a set of core components to assemble Web applications. This set included components for building basic Web UIs with input fields, buttons, links, and a basic table or grid component to render tabular data.

Although the base components provided in the initial specification offered the basics, what was more important was that the specification itself was very flexible, and the intent was that the larger Java community would begin complementing the initial set of base components with much more powerful and specialized components, thus leading to more developer productivity. This intention has largely been achieved based on the growing community of JSF component developers. Some examples of external JSF component development providers include:

- Apache MyFaces – An open source JSF component library and a JSF implementation that provides a more advanced set of useful Web components, including scrollable data tables, trees, tab bars, and so on.
- Oracle ADF Faces – Oracle has rewritten its UIX technology, which was used primarily by Oracle applications, into a 100 percent, JSF-compliant compo-

nent library. ADF Faces components also complement what is provided in the base components with over 100 new components geared toward Web developer productivity and power. These components include tree components, menus, date pickers, color pickers, and a multiselect shuttle. ADF Faces also comes with a performance-enhancing, partial-page rendering technology that allows only a portion of a page to reload, such as when scrolling through tabular data, as opposed to loading the entire page. ADF Faces also provides a handy dialog framework that can generate the necessary client-side JavaScript to open a popup window, accept input, and close itself. The newest feature of ADF Faces is its *skinning* technology, which allows developers to develop an entire look and feel for their Web application.

- WebGalileo Faces – JScape has recently introduced its latest set of very powerful JSF components that include tabbed panels, menus, trees, tables, an HTML editor, and even a calculator.
- ESRI – For the more specialized task of displaying graphical or map data, ESRI has introduced their own JSF component library.

These component libraries are just a small sample of a growing community. A relatively new JSF community Web site, JSFCentral.com, now tracks the progress of JSF technology in general and has a growing list of JSF component technology providers (see Resources online at www.javapro.com). JSF component development is definitely heating up.

When comparing ASP controls and JSF components, we can conclude that because of the recent growth of new JSF component libraries, JSF technology is beginning to challenge ASP's dominance in the Web component-based development area. Now let's turn to a comparison of the platform's respective development environments.

ASP Visual Development

Microsoft's development environment, Visual Studio 2005, offers tightly integrated ASP.Net control interaction that improves the usability of the controls themselves. The aforementioned GridView component is one example. Having quickly mentioned the ability to easily pull up advanced property editors that are customized for each control, let's review some of the most impressive features of Visual Studio 2005:

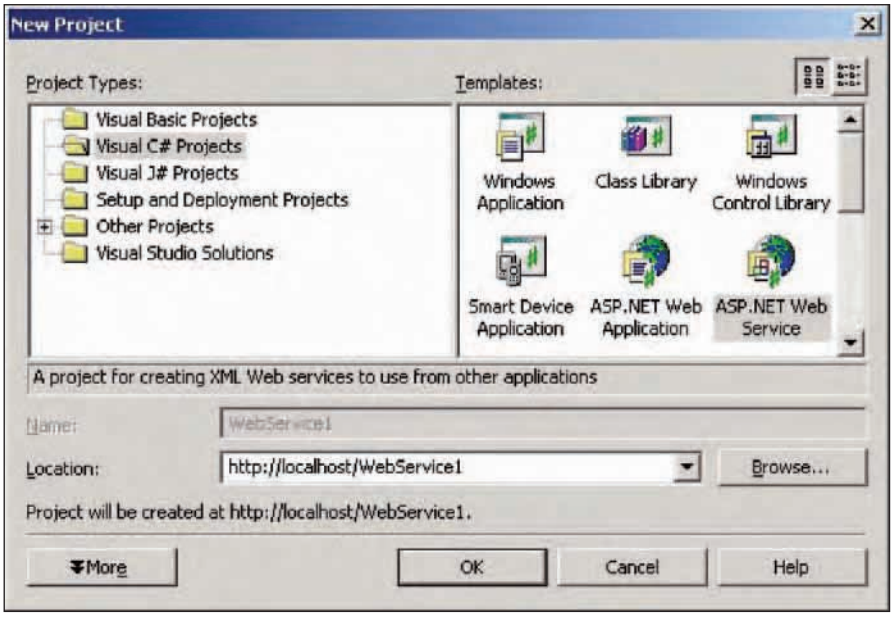


Figure 4 | Web Services Wizard Similar to J2EE, Visual Studio's Web service project wizard makes it easy to consume Web services from external providers and plug their functionality into .Net applications.

- *Web page visual development* – Visual Studio provides intuitive visual rendering of ASP controls in the visual editor. If you drop an ASP control onto the canvas, it renders as (as closely as possible) the run-time rendering, occasionally with dummy data to give the correct visual impression of the control's run-time rendering. As you select the control, you can either edit its properties using a generic property editor window or a nifty new Smart Tasks popup editor that is customized per control.
- *Databinding* – You can use a Smart Tasks popup property editor to bind data and assign a datasource, simply by dragging a data object directly from the Servers Explorer to the canvas. This action provides a predatabound GridView and datasource control.
- *Other productivity features* – Components such as property editors and page templates are handled through Smart Tasks editors, which are simple-to-use popup property editors that are customized for each ASP.Net control. It allows for customized configuration of complex controls. For example, you can easily define the properties of a GridView control such as the datasource; the columns of data to appear; and whether or not editing, sorting, or paging is possible. It is even possible to define the appearance of the control itself, such as alternating row colors and basic color themes.

J2EE obviously doesn't have a single, all-encompassing development environment; instead, the different vendors and open source providers have been providing a set of competing J2EE IDEs that for the most part offer similar development experiences. For this comparison, we'll examine the development experience of the JSF technology in a few different J2EE IDEs and contrast them with the development experience offered by Microsoft.

JSF Visual Development

Let's look at several leading Java development tools providers including Oracle, Sun Microsystems, and IBM to compare JSF/J2EE visual development to ASP visual development. As a whole, you'll find that the JSF-enabled J2EE IDEs compare very well with Microsoft's Visual Studio in that they all support to certain degrees the aforementioned key features such as Web page visual development, DataBinding, and numerous similar productivity features.

Let's first consider the Oracle JDeveloper 10.1.3 development tool, which is Oracle's latest version of its J2EE development environment that now offers JSF visual development support. This support is very comparable to Microsoft Visual Studio's visual ASP development support. It is also similar to Visual Studio in that it is a multipurpose IDE, which means that in addition to supporting JSF development it also supports many other types of Java-related development, ranging from coding of simple Java classes to back-end business logic development with its integrated Enterprise JavaBeans (2.1 and now 3.0) along with Toplink development features.

With regard to Visual Studio's Web page visual development feature, JDeveloper 10.1.3 offers HTML, JSP, and even JSF-enabled JSP visual Web page development. JDeveloper's visual editor provides a runtime rendering of the JSF components while they are being edited. JDeveloper also compares well with Microsoft's strong integration with ASP controls by providing tight integration with the Oracle ADF Faces component library, with productive databinding and smart property editors.

In addition to being integrated with ADF Faces, it is also very easy to load external JSF component libraries into JDeveloper's environment. Apache MyFaces, WebGalileo Faces, and other Faces-component libraries easily integrate into the JDeveloper development environment. Databinding is also provided in JDeveloper by using its DataControl palette in which data items such as database tables are represented by server-side objects—entity beans or Toplink POJOs—that you can drag onto a page, generating a comparable JSF component on the fly and binding it to the server-side object. UI-first databinding is also possible where a JSF component can be dropped onto the page and then databound to a server-side data object.

For the other productive property features, JDeveloper provides re-entrant property editor wizards, which are invoked when a component is dropped onto a page or when a component's properties are accessed through a context menu from the visual editor. These re-entrant wizards offer the ability for advanced customization of the more complex components such as the ADF Faces table component that provides the similar features seen in Microsoft's GridView control: scrolling, paging, editing, and so on. JDeveloper even provides Web template support for HTML, JSP, and JSF.

Sun Microsystems' approach to J2EE development is provided in two products: open source NetBeans Java/J2EE IDE and, for JSF-specific development, Java Studio Creator. For general J2EE development ranging from basic Java class development to enterprise JavaBeans, Sun's NetBeans development provides an all-encompassing environment.

For more traditional Java development NetBeans packs a powerful punch with a long list of coder-friendly features including refactoring, profiling, code folding, and so on. NetBeans also provides the ability to productively build the more server-oriented objects such as Enterprise JavaBeans, servlets, JSPs, and JSP tag libraries.

For developers strictly using JSF, Sun provides Java Studio Creator, which is a relatively unique product in that it focuses primarily on the corporate developer by providing a complete visual development experience for JSF. Similar to JDeveloper, Studio Creator also provides JSF Web page visual development where JSF components render themselves in the editor. *Databinding* is also supported in Studio Creator, where it's possible to drag a datasource object such as a table onto a JSF component like a drop-down menu that results in the component being databound to the data object (see Figure 3 online at www.javapro.com).

Again, similar to Oracle, Sun is also providing a next-generation set of JSF components, currently code named BraveHeart, which provides similar functionalities to the value-add component libraries such as ADF Faces and MyFaces.

Like Sun, IBM also offers more than one development environment for building different types of J2EE applications—WebSphere Application Developer and Rational Application Developer for WebSphere—although there is some overlap. For example, WebSphere Application Developer provides visual development support for JSF applications. It has a visual, JSF-enabled JSP editor as well as their own JSF component library. Productive property editors are available for editing complex components such as their datagrid (table) component. A similar databinding palette also does databinding easily.

More Tools

There are several other JSF development tools worth mentioning. These tools include Exadel's Studio Pro and M7's NitroX Java IDE. Exadel formerly had products called JSF Studio and Struts Studio, but the features for both were merged into Studio Pro. Studio Pro provides visual JSF development in addition to visual Struts development. Exadel's standout feature is probably its navigation diagrammer. Similar to Oracle JDeveloper and Sun's Studio Creator, it is possible to use Studio Pro to define the entire JSF navigation model, visually.

M7's NitroX development tool also provides powerful JSF development with similar features including Visual JSF page development, visual navigation view (navigation editing is still accomplished through a console or to the source directly.)

Both Studio Pro and NitroX are built on the Eclipse platform and clearly demonstrate that although they are less structured, the J2EE development tools environment is thriving with innovation because the core plumbing required for a Java IDE is offered for free to any company who wishes to turn it into a product.

Time will tell how profitable these smaller companies will remain, but they definitely provide a healthy dose of competition in the very Darwinian environment of J2EE. In short, JSF can compete with ASP. Although Microsoft provides a consistent development environment that covers the full development life cycle along with rich ASP.Net Web control libraries, J2EE also has a formidable arsenal on its side with its many JSF component libraries and new, productive JSF development environments that compete with each other as much as they do with Microsoft.

Now that we've see how J2EE can compete with .Net, especially in the newer areas of Web client development, let's examine how J2EE and .Net can *integrate* together as well.

How can J2EE and .Net integrate? Web services and SOA can bring these two technologies together. Web services form the glue that binds J2EE and .Net. Web services evolved out of some basic community-

driven standards based on transmitting pure data in XML between different nodes on the Internet. What transpired was an explosion in a whole host of interconnectivity technologies on both the J2EE and .Net platforms. Both technologies saw a common interest that led to the joint development and maturation of standards. These standards have resulted in the ability today to build and deploy services on the Internet (or any TCP/IP network) and have them communicate with each other, transcending the operating system or core technology from which they came.

On the J2EE side, using the J2EE Web technologies provided in the Servlet API made it possible to build publicly accessible Web services. They provide accessible method calls on any Java class. J2EE development tools such as JDeveloper also provide simple ways to generate the necessary code to consume the output from a Web service.

At the same time as the J2EE camp was getting its Web services act together, Microsoft was also working to provide a set of Web services-friendly features in its .Net architecture. What resulted was a set of easy development features where it is possible to expose any portion of .Net technology built in Visual Basic, C++, or C# and expose it to the Web through the same Web services protocols. Similar to J2EE, it is also very easy to consume Web services from external providers and plug their functionality into .Net applications (see Figure 4).

Services Solution

As Web services continue to standardize, a new standard built on top of the Web services standard protocols has emerged: SOA. SOA is a solution for architecting composite .Net and J2EE applications. With an SOA, developers no longer need to remain bound to any technology camp (J2EE, .Net, or even other legacy computer technologies). Instead, they could construct heterogeneous applications that are built up from a set of Web services as opposed to building a traditional client-server or single datasource application.

What has emerged with SOA is a new standard known as Business Process Execution Language (BPEL) or BPEL4WS. BPEL allows you to link different (Web) services together to achieve a business purpose in a single XML document (BPEL).

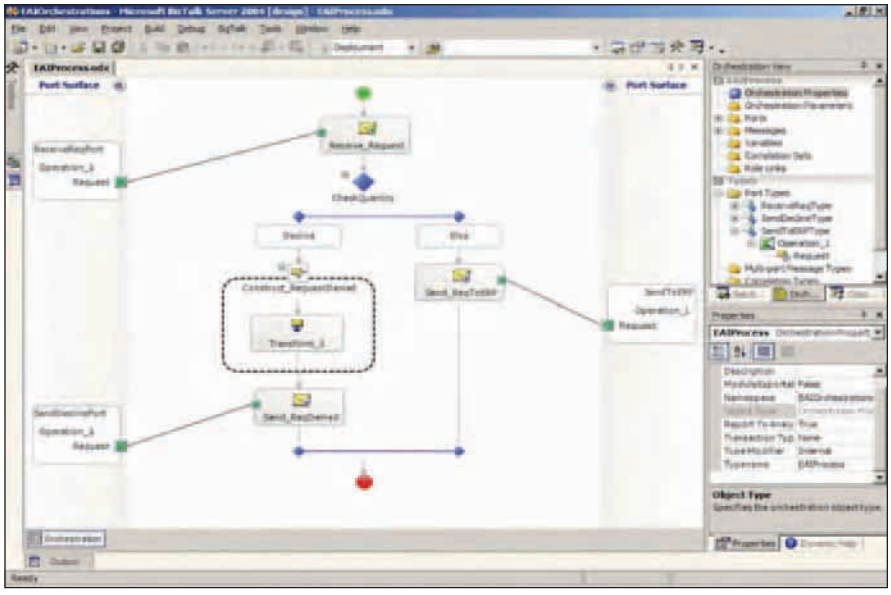


Figure 6 | Create a Business Process The Microsoft BizTalk Server environment provides a rich BPEL design tool that lets you productively string together different services into a business process flow.

For example, you can define a BPEL process to shop for the lowest rate for home loans from different providers on the Web and have only the most inexpensive loan provider's rate return as a final result. This process is possible in BPEL because coding logic into the BPEL process flow—such as comparing between the different rates from multiple providers and returning only the lowest rate—is feasible. BPEL can also account for multiple, long-running processes in parallel, which is known as an asynchronous process.

Since the final output of a BPEL process is also a Web service, it is possible to chain multiple BPEL process flows together to achieve more complicated business flows, which could involve numerous external services. With BPEL as a standard it has become possible to construct synchronous and asynchronous composite applications by linking together a series of Web services in an XML document. For BPEL to work it also requires a server that can interpret the XML BPEL document, which lists the different services and logic that connects them. In this regard, both J2EE and .Net have technology to build SOA applications.

Oracle's new BPEL design-and-deployment environment provides a complete solution for building composite, service-based applications. With the designer (which works in either JDeveloper or Eclipse), you can design visually a BPEL process flow and then deploy it to a BPEL-enabled server that handles the processes (see Figure 5 online at www.javapro.com).

On the .Net side, Microsoft provides an equivalent environment to build SOA applications with its BizTalk Server 2006. Don't let the "Server" part of the name fool you; the BizTalk environment also provides a rich BPEL design tool from which to productively string together different services into business process flows (see Figure 6).

We've examined how J2EE has made great progress on transforming itself from a server-oriented technology reserved only for computer scientists into a thriving, community-driven Web client powerhouse with JSF and the Java IDEs that compete to support them. We've also discussed how Microsoft has strengthened its offering in the company's traditional areas of expertise in client development with the new Visual Studio 2005 and ASP 2.0 components. In the end of these technology battles the true winner is logically the consumer of these technologies, for as we can see with the adoption of SOA and evermore parallel component-centric development paradigm, both technologies will be usable and will even be fully interoperable. JP

Chris Schalk is a principal product manager and Java evangelist for application server and development tools at Oracle Corporation. Chris also maintains a blog on J2EE Web development. Contact Chris through his blog at: **http://jroller.com/page/cschalk**.

Software Architecture Summit

January 31 - February 1, 2006

Deliver on the Promise of the Agile Enterprise

Learn the latest insights on SOAs, middle-tier architecture, data layer access, and a host of other SA topics. Designed by your software architect peers, SAS will help you and your organization deliver on the promise of the agile enterprise.

- Key Issues in Software Architecture
- Application Architecture
- Data Layer Architecture
- Patterns for Service-Oriented Architecture
- Best Practices for Faster Application Development
- Object-Oriented Concepts in SOA
- Deployment and Operations

Software Architecture Summit returns to Moscone Center West in downtown San Francisco

SAS delivers information from every angle, giving you a complete agenda that will arm you with best practices and inspiration for your next projects. The event includes:

- Technology break out sessions
- Issue-oriented panel discussions
- Industry theories and higher-level discussions, led by SA luminaries
- Dynamic keynotes from SAs at the forefront of the field

Register by November 16 and save up to $350 • Visit www.ftponline.com/sas or call 1-800-848-5523

SAS 2005 featured Gordon van Huizen, Sonic Software, Yaacov Cohen, Mainsoft, and Steven Davis, Walt Disney Studios

Maximize your time in San Francisco: Get the Gold Passport

For your convenience, SAS is once again co-located with VSLive! San Francisco. Join us January 31-February 1 for two full days of breakout sessions and keynotes on the latest trends in software architecture. Plus, when you sign-up for the all-inclusive Gold Passport you'll gain access to sessions, content and take-home material from Smart Client Live!, ASP Live!, SQL Live! and more. See www.vslive.com/sf for complete details.

Meet the Experts

SAS brings the experts to you. Conference speakers will share insights on best approaches to patterns and practices as well as case studies from industry leaders.

SAS speakers and editorial board include:

Steven P. Davis
Vice President of IT
Walt Disney Studios

Simon Guest
Program Manager on
Microsoft's .NET
Enterprise Architecture Team

Billy Hollis
Independent Consultant and
President of DotNetMasters

Peter Varhol
Engineering Management at
Progress Software

Sign up by November 16 | Save $300

Save hundreds when you register by the Super Early Bird discount deadline, November 16, 2005. Take off a chunk of the registration cost and maximize tremendous hotel discounts at top-rate sites within walking distance of Moscone West in downtown San Francisco. Visit www.ftponline.com/sas for details.

Register by November 16 and save up to $350 • Visit www.ftponline.com/sas or call 1-800-848-5523

Dynamic Service-Oriented Architecture

An SOA framework provides the innovation for building loosely coupled interfaces and cool composite applications

by Ted **FARRELL**
and Raghu **KODALI**

In today's technology environment, enterprises implement and manage increasingly complex business needs. These requirements are changing continually as markets shift because of global competitive pressures. To address the changing needs of the enterprise in a cost-effective and timely manner, a flexible and dynamic IT environment is needed. Let's look at an approach that allows you to manage complex enterprise business processes.

Creating a flexible IT environment can be achieved by using a service-oriented architecture (SOA). SOA is a technology architecture approach that enables the creation and assembly of services. These services are functionality exposed through an abstract interface that can be shared and reused across any number of applications. The shared and reusable nature of these services has given rise to the term *loosely coupled*. A collection of these loosely coupled services can be assembled into what are known as *composite applications*.

Developing a composite application can address the flexibility needs of an enterprise by enabling developers to assemble loosely coupled services to create a wide variety of applications, or extend existing applications. If used properly, these services offer substantial benefits to the enterprise. Loosely coupled applications promise to be easier to modify and update while still retaining the scalability

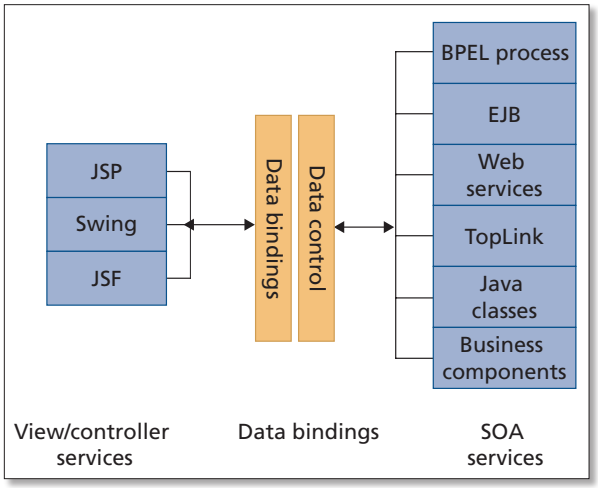


Figure 1 | Bind an Interface to Services Two artifacts—DataControls and DataBindings—provide abstraction to back-end services such as BPEL processes, EJB applications, Web services, Java object model, standard Java classes, or POJOs.

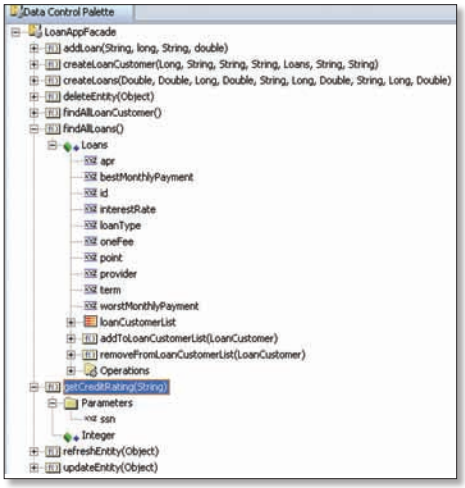


Figure 2 | Invoke Back-End Methods Java standard IDEs can be used to expose metadata for service interfaces in different formats that provide productive ways of developing user interfaces.

Go Online

Visit www.javapro.com for related resources. Simply type the Locator+ code into the field in the upper-right corner of the page.

Download
JP0509 Download all the code for this issue.

Discuss
JPTALK Discuss this article in the "Talk to the Editors of *Java Pro* Magazine" discussion forum.

Read More
JP0509TF_T Read this article online.

EA0404AB_T Read the related article "Build Your Business Apps with BPEL" by Avi Borthakur and David Shaffer.

EA0402JM_T Read the related article "The New Economics of Software as a Service" by John McDowall.

JPEP030109J2_T Read the related article "Develop and Deploy a Simple EJB" by Budi Kurniawan.

Listing 1 View Technology Access

```
<?xml version="1.0" encoding="UTF-8" ?>
<DataControlConfigs   xmlns=
  "http://xmlns.oracle.com/adfm/configuration"
  version="10.1.3.32.85" Package="buslogic.mypackage"
  id="DataControls">
<AdapterDataControl id="LoanAppFacade" FactoryClass=
  "oracle.adf.model.adapter.DataControlFactoryImpl"
  ImplDef=
  "oracle.adfinternal.model.adapter.ejb.EjbDefinition"
  SupportsTransactions="true" SupportsSortCollection=
  "false" SupportsResetState="false"
  SupportsRangesize="false" SupportsFindMode="false"
  Definition="buslogic.LoanAppFacade" BeanClass=
  "buslogic.LoanAppFacade" xmlns=
  "http://xmlns.oracle.com/adfm/datacontrol">

<Source>
  <ejb-definition ejb-version="3.0"
    ejb-name="LoanAppFacade"
    ejb-type="Session"
    ejb-business-interface="buslogic.LoanAppFacade"
    xmlns="http://xmlns.oracle.com/adfm/adapter/ejb"/>
</Source>
</AdapterDataControl>
</DataControlConfigs>
```

You can access a EJB 3.0 application through the standard service interface from any type of view technology like JSP or JSF after a DataControl is created for a session façade.

Listing 2 Metadata Exposure

```
<?xml version="1.0" encoding="UTF-8" ?>
<JavaBean xmlns=
  "http://xmlns.oracle.com/adfm/beanmodel" version=
  "10.1.3.32.51" id="LoanAppFacade" BeanClass=
  "buslogic.LoanAppFacade" Package="buslogic"
  isJavaBased="false">
<AccessorAttribute id="loans" IsCollection="true"
  BeanClass="buslogic.persistence.Loans"
  CollectionBeanClass="ReadOnlyCollection">
</AccessorAttribute>
<MethodAccessor IsCollection="false" Type="void"
  id="addLoan"  ReturnNodeName="Return">
<ParameterInfo id="provider" Type="java.lang.String"
  isStructured="false"/>
<ParameterInfo id="term" Type="long" isStructured=
  "false"/>
<ParameterInfo id="loan_type" Type="java.lang.String"
  isStructured="false"/>
<ParameterInfo id="interest_rate" Type="double"
  isStructured="false"/>
</MethodAccessor>
<MethodAccessor IsCollection="false" Type=
  "java.lang.Integer" id="getCreditRating"
  ReturnNodeName="Return">
<ParameterInfo id="ssn" Type="java.lang.String"
  isStructured="false"/>
</MethodAccessor>
</JavaBean>
```

You can expose metadata for service interfaces through standard Java IDEs in different formats and provide productive ways of developing user interfaces.

and robustness of traditional applications. However, deploying loosely coupled applications requires consideration beyond the services of an SOA.

One of the most commonly overlooked aspects of composite application development is how to expose those services to the end user. While each of the individual services of an SOA is important in its own right, the true value is realized in the front end of the application. As with any application, the difference between a good composite application and an ungainly one often resides in the user interfaces. Therefore, a simplified SOA requires not only a standard way to create and expose loosely coupled services, but also a means to bind the data from these services into the user interface.

Dynamic Delivery

In typical service-oriented applications, heterogeneous back-end applications, systems, or services are exposed through standard service-oriented interfaces using Web Services Description Language (WSDL) with different types of bindings. While WSDL abstractions work nicely for Web services, few of them employ user interfaces that consume Web services directly through a WSDL interface. Most user interfaces are typically interacting with back-end business systems that directly contain business logic or communicate through specific adapters that bind to the data coming from the back end and invoke the business methods. As a result, most composite applications have user interfaces that are tightly coupled to the back-end business logic, thus limiting the flexibility that is expected from SOAs.

The correct approach to building user interfaces for composite applications would be accessing the back-end business logic not exposed as Web services. There have been attempts to address this approach by using templates or frameworks that would provide some loose coupling to back-end systems. However, most of these templates and frameworks are not scalable enough to meet the requirements of heterogeneous, back-end systems. On the other hand, standard enterprise development platforms, including J2EE 1.4, do not offer a solution either.

> Loosely coupled applications promise to be easier to modify and update while still retaining the scalability and robustness of traditional applications

Historically, software frameworks have been viewed as proprietary solutions

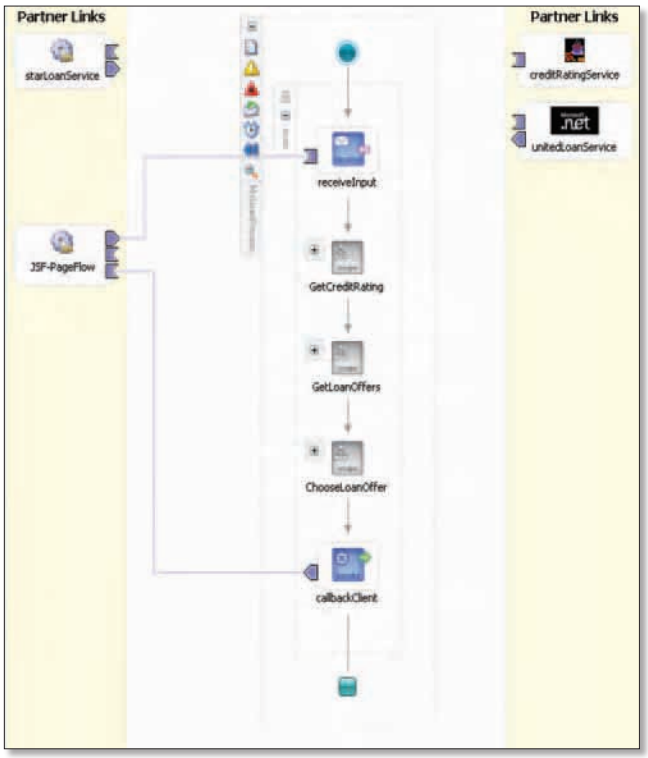


Figure 3 | Integrate Flows This framework can be used to integrate page and process flows that have been constructed with JSF and BPEL, or other Java technologies.

that bypass the complexity of sophisticated enterprise platforms such as J2EE. However, these frameworks were often based on proprietary technologies implemented by a single vendor, and concerns about vendor lock-in limited their success. Some open source projects such as Apache Struts have enjoyed substantial adoption for simplifying Web development but still do not fully address the needs of building user interfaces for composite applications in a large enterprise.

One of the key properties of an SOA is that composite application development is much more dynamic than the traditional paradigm. In an SOA the user interface must be as loosely coupled as the services themselves. Building on recent advances with both proprietary and open source technologies, Oracle is working with the Java community to deliver a new specification within the Java Community Process (JCP) to deliver a standard dynamic framework optimized for an SOA. With a standard composite application framework, much of the effort required to build user interfaces is eliminated. This benefit enables developers to improve their productivity by focusing their skills on application logic and business process.

Java Specification Request (JSR) 227 specifies a declarative framework that greatly improves developer productivity when binding a user interface to services. It defines a way to abstract bindings between the business tier and view/controller layers. Abstraction to back-end services such as Business Process Execution Language (BPEL) processes, Enterprise JavaBeans (EJB) applications, Web services, Java object model, standard Java classes, or plain old Java objects (POJOs) are done using two artifacts: DataControls and DataBindings (see Figure 1).

Binding the Back End

DataBindings serve as lightweight objects that can be used to bind the data coming from back-end systems onto the UI components of the view layer. This binding formalizes the interactions between a UI component and values or methods available as part of a service. By using the declarative bindings, any Java UI-rendering technology can be used to declaratively bind to any service. Declarative bindings can be integrated with Java IDEs to provide features such as drag-and-drop functionality on top of standard service interfaces that expose heterogeneous back-end systems. To facilitate a common mechanism for accessing diverse services, declarative bindings access the services through a lightweight abstraction layer called DataControls.

DataControls serve as a service-oriented interface that has information about data collections and service methods. They provide supplemental metadata about a service's capabilities and constraints and also support a simple interface for normalizing typical interactions with the service. DataControls comprehensively describe the attributes and methods so that UI components can make intelligent assumptions about how to render the data.

In the examples provided here we can see how the metadata for service interfaces and data bindings look for a J2EE application that processes credit requests and is developed using EJB 3.0 technology, and a LoanFlow application that is serving as a business process developed using BPEL.

The EJB 3.0-based application is composed of entities that are mapped to database tables using metadata annotations and wrapped by a session façade (LoanAppFacade) that is exposing collections of data retrieved using EJB3 entities and has business logic methods such as getCreditRating().

Once a DataControl is created for a session façade, the EJB 3.0 application can be accessed through the standard service interface from any type of view technology such as Swing, JavaServer Pages (JSP), JavaServer Faces (JSF), or others (see Listing 1).

Metadata for service interfaces can be exposed through standard Java IDEs in different formats and provide productive ways of developing user interfaces (see Listing 2

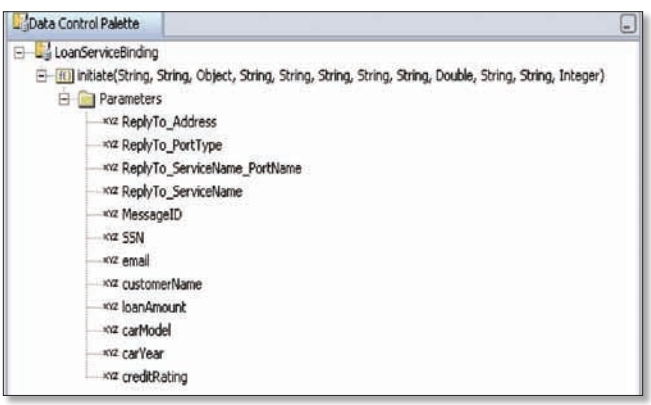


Figure 4 | Develop Page Flow You can use standard technologies like JSF to develop page flows once a DataControl is created.

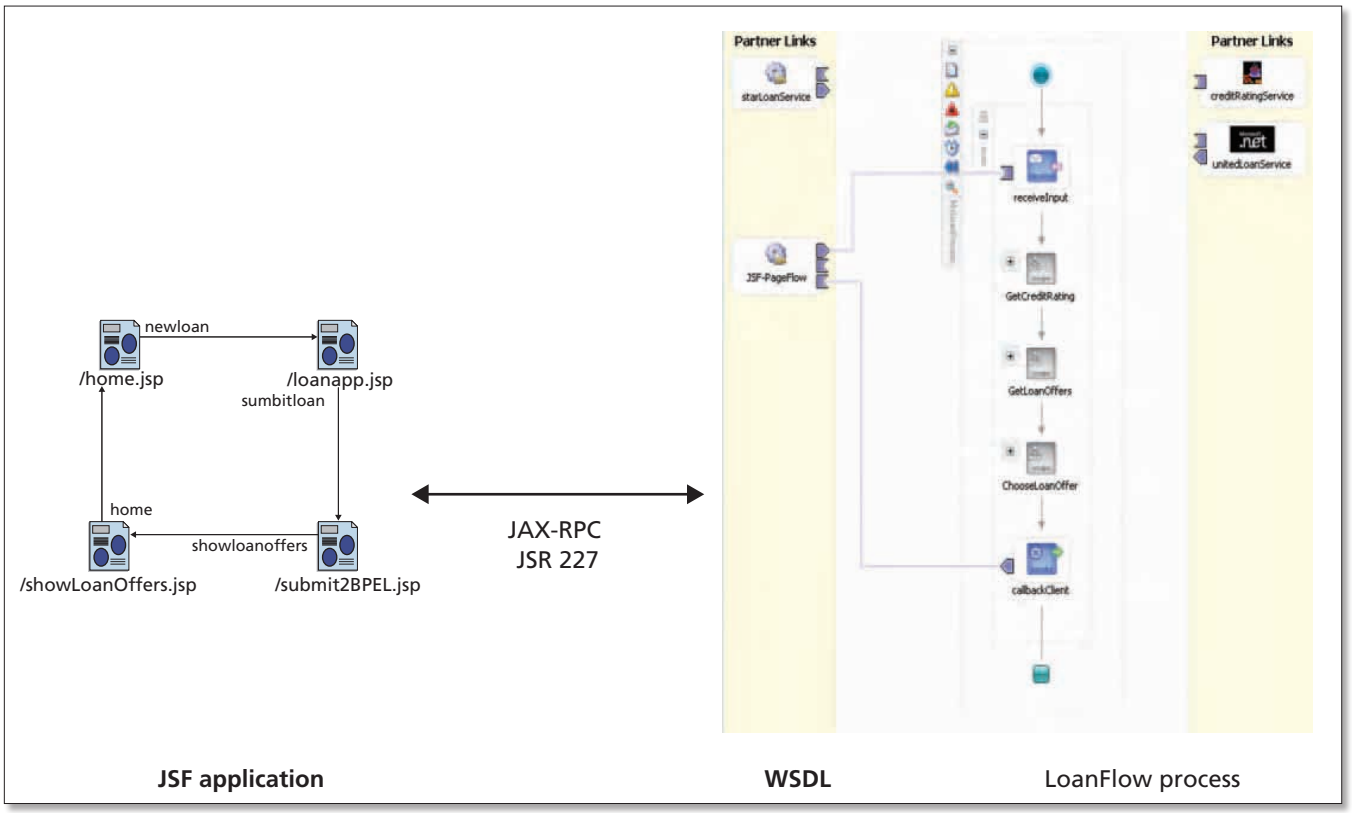


Figure 5 | Capture Information Input forms use standard JSF UI components to capture information.

and Figure 2). Examples include drag-and-drop collections of data that can be bound to a standard UI component or dragging and dropping a service method that can invoke back-end business methods based on user interaction.

It is important to remember that the JSR 227 specification is not limited to one particular back-end technology. The example illustrated in Figure 3 demonstrates how this framework can be used to integrate page and process flows that have been constructed with JSF and BPEL, but it also can be used with EJB 2.x/3.0, Web services, JCA, or any POJO.

LoanFlow application is a business process (see Figure 3) developed using BPEL. LoanFlow process is consuming services from heterogeneous systems and interacting with loan providers through Web services. The LoanFlow process takes a loan application/document as a request message and returns the loan offer as a response in asynchronous fashion.

Ensure Loose Coupling

Once a DataControl is created for the LoanFlow WSDL (see Figure 4), you can develop page flows by using standard technologies like JSF. Input forms use standard JSF UI components to capture loan application information, and then invoke the LoanFlow business process to show the loan offers that have been received by the LoanFlow process from loan (see Figure 5).

SOAs are designed to allow users to consume services from anywhere and use them to build composite applications. To obtain the full benefits of an SOA, proper composite application development requires both architects and developers to ensure all aspects of the application are loosely coupled. Using a dynamic SOA framework and declarative bindings and DataControls enables composite applications to be loosely coupled beyond the services themselves. They provide an innovative way of building loosely coupled application interfaces, freeing tool developers from worries about the back end of the system and letting them focus on delivering better ways to build applications.

As we've seen in the examples shown here and this discussion, a dynamic SOA framework can make the difference between an ordinary composite application, and a great one. JP

> A key property of an SOA is that composite application development is much more dynamic than the traditional paradigm

Ted Farrell is vice president and chief architect for application development tools at Oracle. Ted is responsible for the technical and strategic direction of Oracle's development tools products, including Oracle JDeveloper 10g and Oracle's Java and XML integrated development environment (IDE) for J2EE application and service-oriented development. **Raghu Kodali** is consulting product manager and SOA evangelist for Oracle Application Server at Oracle. Raghu leads next-generation SOA initiatives and J2EE feature sets for Oracle Application Server, with particular expertise in EJB, J2EE deployment, Web services, and BPEL. Visit Raghu's blog at Loosely Coupled Corner (**www.jroller.com/page/raghukodali**).

About the IBM Rational Software Development Platform

Highlights

- **A complete, proven, open and modular solution for building software**
- **Enables business-driven development by unifying business, development and operations teams**
- **Helps teams rapidly discover, develop and deploy software assets**

ON DEMAND BUSINESS™

Software: Key to an on demand world

Look within any company that sustains competitive advantage and you will find software in action. Whether applied to an existing system, a new project or a packaged application, software development allows organizations to codify their unique knowledge and differentiate their business performance.

The high rate of failure

Successful software development requires the close collaboration of business, development and operations teams, all working toward a common goal. But all too often, these extended teams fail to produce meaningful results. When projects go astray, it is typically not because any one team is dysfunctional, but because the cross-functional organization is fundamentally misaligned or disconnected.

Business-driven development

For business and technology leaders who seek greater returns from their technology investments, IBM offers a proven alternative: business-driven development with the IBM Rational Software Development Platform. This integrated approach aligns line-of-business, development and operations teams with a shared process, toolset and vocabulary for developing software. The result is an organization that can deliver software with the power to transform, create new revenue sources and achieve breakthrough results.

A proven platform

The IBM Rational Software Development Platform is not a single product offering, but an integrated set of products that can be adapted to the unique needs of your team and technology environment. IBM tools span Microsoft® Windows®, UNIX®, Linux® and mainframe platforms and support a wide spectrum of programming languages, integrated development environments (IDEs) and cross-development environments for real-time and embedded developers. Core tools within the IBM Rational Software Development Platform are based on Eclipse, an award-winning open source platform for the construction of powerful software development tools and rich desktop applications.

The IBM Rational Software Development Platform is:

- ***Complete.*** *It unifies business, technology and operations teams with a comprehensive and integrated solution*
- ***Open and extensible.*** *You can leverage existing assets and choose from a wide array of partner technologies, including Microsoft .NET.*
- ***Modular.*** *You choose the exact tools and adoption path that best fit your needs.*
- ***Proven.*** *The IBM Rational Software Development Platform is based on tools and best practices that are the choice of thousands of high-performance software teams.*

Requirements and Analysis	
IBM Rational® RequisitePro®	Document, communicate and control changing requirements to maintain project focus
IBM Rational Software Modeler	Visually model systems and applications using Unified Modeling Language (UML) notation
IBM Rational Rose® Data Modeler	Design and model database schema to support your enterprise architecture
IBM WebSphere® Business Integration Modeler	Define, model, analyze, simulate and report business processes, and extend IBM WebSphere MQ Workflow with business tools to visualize process impact
IBM WebSphere Business Integration Monitor	Monitor business processes—in real time and using visual dashboards—for improved business decision-making
IBM WebSphere Studio Asset Analyzer	Maintain and extend existing assets through impact analysis and enhanced application understanding

Design and Construction	
IBM DB2® Information Integrator	Develop integrated, real-time access to diverse distributed information
IBM Lotus® Domino® Designer	Rapidly create multiplatform applications to automate business processes and increase workflow efficiency
IBM Lotus Enterprise Integrator for Domino	Provide real-time data access, bi-directional data movement, exchange, transformation and synchronization between applications and data sources
IBM Rational Application Developer for WebSphere Software	Quickly design, develop, analyze, test, profile and deploy Web, Web services, Java™, J2EE™ and Portal applications with a comprehensive Eclipse-based IDE that is optimized for IBM WebSphere software and provides capabilities for development on other technology platforms
IBM Rational Rose Technical Developer	Develop complex Java, C/C++ and embedded software using model-driven development techniques
IBM Rational Software Architect	Unify architecture, design and development with one tool
IBM Rational Suite® Development Studio for UNIX	Implement a full life-cycle solution for building software on UNIX platforms
IBM Rational Suite for Technical Developers	Implement a full life-cycle solution for building complex software for real-time and pervasive devices
IBM Rational Web Developer for WebSphere Software	Build, test and deploy Web, Web services and Java applications with an easy-to-use IDE that is optimized for IBM WebSphere software and provides capabilities for development on other technology platforms
IBM WebSphere Development Studio for iSeries	Create Web and J2EE applications with an advanced IDE for IBM ^ iSeries™
IBM WebSphere Host Access Transformation Services	Quickly transform 3270 and 5250 applications for the Web
IBM WebSphere Studio Application Developer Integration Edition	Build service-oriented applications that extend and integrate your existing information technology (IT) assets with a composite application development environment
IBM WebSphere Studio Device Developer	Develop embedded software with a Java 2 Micro Edition (J2ME™) application development environment
IBM WebSphere Studio Enterprise Developer	Support a mixed workload environment for IBM z/OS® development, including J2EE, COBOL and PL/1
IBM Workplace Designer	Use visual design tools to build and extend business applications for the IBM Workplace environment
Toolkits for IBM WebSphere Studio	
• **Embedded Voice Toolkit**	• Build speech recognition and text-to-speech capabilities in embedded applications
• **Everyplace Toolkit**	• Extend enterprise applications to mobile devices
• **Micro Environment Toolkit**	• Build applications that run on pervasive devices such as Pocket PC, Palm and Symbian
• **Multimodal Toolkit**	• Create applications that work in voice and data modes in the same user interaction
• **Voice Toolkit**	• Develop voice-enabled applications for call centers and telephony-integrated applications

Software Quality	
IBM DB2 Test Database Generator	Rapidly produce test data that reflects real-world conditions
IBM Rational Functional Tester Extension for Terminal-based Applications	Add support for testing 3270 and 5250 terminal-based applications to Rational Functional Tester
IBM Rational Functional Tester	Perform functional test automation to increase defect discovery within complex Java, Microsoft Visual Studio® .NET WinForm and Web-based applications
IBM Rational Manual Tester	Author and execute manual tests using novel design and reuse capabilities
IBM Rational Performance Tester	Verify Web application performance, scalability and reliability under high multi-user loads
IBM Rational PurifyPlus™	Automate runtime error detection, bottleneck discovery and code coverage analysis for increased unit testing and debugging effectiveness
IBM Rational Robot	Verify quality of client/server applications with a general-purpose test automation tool for quality assurance teams
IBM Rational Rose XDE Developer family	Perform model-driven development and runtime analysis within your Eclipse, IBM WebSphere Studio or Visual Studio development environment
IBM Rational Test RealTime	Perform component testing and runtime analysis for applications targeting embedded and other real-time environments
Software Configuration Management	
IBM Rational ClearCase® family	Manage software assets with comprehensive version control, workspace management, and build and release management
IBM Rational ClearQuest® family	Manage workflow and change requests, including defects and enhancements, across the development life cycle
IBM z/OS SCLM Suite	Enable software configuration management for mainframe applications development
Deployment Management	
IBM DB2 UDB Administration Tools	Manage tasks that keep DB2 performing at peak levels
IBM Fault Analyzer for z/OS and OS/390®	Assist in the identification and correction of enterprise application failure
IBM Tivoli® Access Manager	Provide end-to-end security for e-business, including Web single sign-on, Web-based administration and policy-based security
IBM Tivoli Configuration Manager	Automatically provision and update your test and deployment environment
IBM Tivoli Enterprise Console	Provides automated problem diagnosis and resolution to improve system performance and reduce support costs
IBM Tivoli Monitoring (Databases, Messaging and Collaboration, Microsoft .NET, Network Performance, Transaction Performance, Web Infrastructure)	Monitor essential system resources to detect bottlenecks and potential problems, and to recover automatically from critical situations
IBM Tivoli Identity Manager	Centrally coordinate the creation, approval and provisioning of user accounts
IBM Tivoli Privacy Manager for e-business	Protect consumer trust by implementing privacy policies which guard consumer information
IBM Tivoli Provisioning Manager	Automatically provision and update your test and deployment environment
IBM WebSphere Studio Application Monitor	Provide runtime performance monitoring and analysis for J2EE applications spanning mainframe and distributed platforms

Process and Portfolio Management	
IBM Lotus Domino	Rapidly build collaborative applications and messaging solutions with a security-rich, scalable and easy-to-use platform
IBM Lotus Notes®	Provide a robust platform for collaborative applications with enterprise-class messaging, calendaring and scheduling, and instant messaging capabilities
IBM Rational Portfolio Manager	Automate IT governance processes and align priorities, projects and people with business goals
IBM Rational Professional Bundle	Get all the desktop tools your enterprise needs to design, construct and test J2EE/portal/service-oriented applications
IBM Rational SUMMIT® Ascendant	Plan and manage enterprise IT projects and programs with a library of methods and Web-based tools for planning, estimating and monitoring
IBM Rational Team Unifying Platform™	Provide core infrastructure including life-cycle process guidance, requirements management, version control, defect and change tracking, test management, project measurement and reporting
IBM Rational Unified Process®	Adopt a flexible development process that can be configured to your team and technology needs

Accelerate success with IBM Software Services

IBM Software Services teams are dedicated to helping organizations improve their software development capability and accelerate project success. Our services encompass worldwide consulting, deployment and QuickStart packages along with a wealth of knowledge and training resources.

For more information

To learn more, contact your local IBM representative or visit our Web site at:

Developers:
ibm.com/developerWorks/platform

IT managers/executives:
ibm.com/software/developmentplatform

© Copyright IBM Corporation 2004
All Rights Reserved

IBM Corporation
Software Group
Route 100
Somers, NY 10589
U.S.A.

Produced in the United States of America
08-05
All Rights Reserved

IBM, the IBM logo, the on demand business logo, ClearCase, ClearQuest, DB2, Domino, eServer, iSeries, Lotus, Notes, OS/390, PurifyPlus, Rational, Rational Rose, Rational Suite, Rational Summit, Rational Unified Process, RequisitePro, SoDA, Team Unifying Platform, Tivoli, WebSphere and z/OS are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries or both.

Microsoft, Visual Studio and Windows are trademarks of Microsoft Corporation in the United States, other countries or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product and service names may be trademarks or service marks of others.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates.

G507-1653-03

Manage Deployment Descriptors

Ease the pain of managing J2EE's complex deployment and its run-time values in your enterprise environment

by Sean **BLANTON**

IT professionals have learned that managing the deployment of J2EE applications in large enterprise environments is a complex and expensive process. The expense comes from the technical complexity of deploying applications to multiple-server environments where run-time parameters contained within the Enterprise Application Archive (EAR) files must be uniquely configured for each server.

The problem can be understood easily by comparing an EAR file to a ZIP file. Imagine that you have a ZIP file containing 1,000 files. You want to copy that ZIP file to five different locations, but first you must change some of the text in ten of the zipped-up files, and the changes are different for each location where you need a copy. This task requires recreating the ZIP file by uniquely configuring the ten text files and re-zipping them five times. Then you copy each ZIP file to the correct location according to the changes made.

When working with J2EE applications, this same process is referred to as *assembling* J2EE applications. The files that contain the uniquely configured run-time parameters are called *deployment descriptors*. There are usually several other non-J2EE files that also contain run-time environment information. Managing these deployment descriptors for multiple-server configurations is greatly underserved by software tool vendors, forcing IT professionals to develop their own solutions. The tasks they are faced with are creating a reproducible process to update the run-time values in the deployment descriptors after development for each server in the enterprise with its unique configuration, constructing an EAR file for each environment, and delivering each to the correct environment.

Unlike other application run-time environments where separate configuration files can be deployed along with an executable, J2EE complicates the issue by requiring all of the deployment descriptor and other files to be archived into the EAR file, just like our ZIP file example. This situation transfers the problem from being a "deployment" issue squarely into being a "build" issue because an EAR file must be *built* for each uniquely configured server. In other environments such as C or C++, the application team could build one executable and deploy it to multiple servers along with an initialization file (INI). With J2EE, a separate executable (the EAR file) must be built for each server so that the unique configuration contained in the deployment descriptors are archived into the EAR file.

The Build Problem

In recent years, companies have invested in sophisticated software delivery tools to handle the complex deployment of configuration files that are deployed with the executables. However, now the problem is no longer to coordinate "delivery" because everything is packaged into a single EAR file, but instead to coordinate "build" because each server requires a different EAR file. Software delivery tools can deliver the deployable EAR file, but creating the deployable EAR file must be handled before the software delivery stage.

The problem is not entirely the fault of the J2EE specifications, but largely the result of increased use of third-party modules and interconnectivity

Go Online

Visit www.javapro.com for related resources. Simply type the Locator+ code into the field in the upper-right corner of the page.

Download

JP0509 Download all the code for this issue.

Discuss

JPTALK Discuss this article in the "Talk to the Editors of *Java Pro* Magazine" discussion forum.

Read More

JP0509SB_T Read this article online.

JPEP040128BK_T Read the related article "Using Data Sources the Right Way" by Kulvir Singh Bhogal and Kawan Sik Kang.

JP0403KJ_T Read the related article "Go Beyond Tag Libraries" by Kevin Jones.

JPEP040218BI_T Read the related article "Capitalize on J2EE Debugging Tools" by Brian Irwin.

Spreadsheet Column	Cell Content
Environment	Identifies the type of environment such as development, testing, or production
Instance	Indicates a particular server when more than one may be in the same environment
Archive	The name of the JAR, WAR, or EAR file to be changed
File	The name of the file that contains values that change between environments and instances
Name/XPath	Locates and identifies the value to change
Value	Sets to the parameter value that is to be used for the environment and instance indicated in that row

Table 1 | Spreadsheet Columns for Managing Run-time Values The spreadsheet columns are useful for managing the application and run-time information that may change.

of enterprise applications and services. A single application running on a production server may require parameters that specify multiple database connections, log file locations, message queues, security information, CICS regions, and other server-specific information. Put that same application on a second production server for a different business unit and all of these values could be different. It is not unusual to see applications where as many as 100 values might change. Take an application using parallel development that has five production servers and a few test and development servers and the problem is compounded.

To address this complexity, developers, architects, and change management and production engineers have had to develop their own internal solutions to build and deploy the EAR files that contain an application and the unique run-time specific configuration parameters. These customized approaches have resulted in a lot of planning, script writing, documentation, training, and trial and error to come up with a solution that works for the entire organization. The *correct* solution may vary because of the size of an organization: the number of application teams, the number of run-time environments, and the types of change and security controls in place.

Trends are emerging from companies that have automated their J2EE assembly process successfully. Effective approaches include software tools that are designed to reduce or eliminate scripting such as the Apache Maven Project and Openmake by Catalyst Systems Corporation. Other solutions to construct the different EAR files involve manually writing a set of scripts that use Apache's Ant; Perl; and less-commonly, but no less effectively, Unix shell scripts and GNU make. There are also a number of products like AntHill and Cruise Control that help you manage your scripts and control how and when they are run once they've been written.

The goal of these approaches is to minimize the maintenance requirements for developers, manage the sometimes large number of run-time parameters, modify the files with parameters, and to build the EAR files—all while allowing an organization to maintain their processes and controls. Sharing the approaches presented here should help you eliminate some of the trial and error involved when defining a solution. Regardless of the solution—manually scripted or handled through commercial build software—the tips presented here will help clarify the complexities of managing J2EE deployment descriptors.

The Developer's EAR File

Developers should build an EAR file that suits their environment for coding and testing. Ideally, developers should be able to apply their entire focus and spend their entire time on engineering the required software changes rather than worrying about managing parameters and building different EAR files for upstream environments. Only one member of a development team really needs to coordinate application parameters for the different environments.

Env.	Inst.	Archive	File	Token/XPath	Value
Int	1	EComm.ear/ ECommEJB.jar	META-INF/ ejb-jar.xml	/ejb-jar/ enterprise-beans/ session[@ id='ApplicationData']/ env-entry[@ id='EnvEntry_2']/ env-entry-value	http://devlb:8040
QA	1	EComm.ear/ ECommEJB.jar	META-INF/ ejb-jar.xml	/ejb-jar/ enterprise-beans/ session[@ id='ApplicationData']/ env-entry[@ id='EnvEntry_2']/ env-entry-value	http://testlb:8040
…	…	…	…	…	…
Prod	2	EComm.ear/ Ecomm.war	WEB-INF/ server-config.wsdd	//service[@name='Ecomm']/ parameter[@ name='beanJndiName']/ @value	PREPEND="$STAGE$/" IFDEF="$STAGE$"
Prod	2	EComm.ear/	log4j.properties	@LOG_LOC@	E:\applogs\prod\ecomm\ log4j\log4j.log

Table 2 | Spreadsheet for Managing Run-time Values In the spreadsheet for managing run-time values, columns are identified as Env. for environment, Inst. for instance, Archive for the name of the JAR, WAR, or EAR file to operate on, and File for the file that contains a value that changes, which is embedded in the Archive file. The Token/XPath column provides the way to locate or identify the value to change, according to the type of file, and Value is the appropriate value to be used for the Environment and Instance indicated in that row.

One thing that is not as ideal is to ask a developer to take a working application and parameterize properties and values in files. Doing so means that if a developer has coded a particular property in a file for a log file location with a value of C:\appsdev\efoobar\comm.log, then the developer will create a parameterized version of the same file where the value has a parameter like @COMM_LOG_LOC@. The strategy then is to replace the parameter with values meaningful to the different target run-time environments—for example, C:\appsqa\efoobar\comm.log for a test server and C:\appsprod\efoobar\comm.log for a production server.

The reason that parameterizing lots of files is popular is that Ant copy-and-replace tasks have a nifty seach-and-replace capability for tokens of this type, and it is tempting to make use of this tool in hand. However, additional labor is required to do the parameterization, and it is a risky manual process to eyeball and parameterize every relevant run-time value in a set of files. If one of these values is missed, the error would not be caught until run time after the EAR files have been built and deployed. We have found at larger customers' sites that values that should be parameterized inevitably will get overlooked now and then, causing costly redeployments to test environments.

In this case the burden for developers is not trivial: they must now keep track not only of which properties need to be changed for each environment, but also which properties in which files have been parameterized. While there are situations where it may be necessary to go this route, parameterizing property values should be minimized and used for as small a subset of files and parameters as possible. Starting out parameterizing everything in sight is a common road to follow.

Use a spreadsheet! A spreadsheet may sound like an odd tool for building J2EE applications, but think again. Managing parameters is really about documenting and tracking the configuration of each server and the infrastructure. Providing this information in a spreadsheet does two things. First, it provides a central location for all of the configuration information that can be managed by people. Second, the information from most spreadsheet programs can be exported in a format that can be used as part of an automated process. Microsoft Excel spreadsheets can be saved as either an XML file or a comma-separated-variable (CSV) file. The XML file can be processed using XML Path Language (XPath) using the Ant XSLT task, and the CSV format lends itself more to processing by Perl. See Table 1 for an example of spreadsheet column names, and see Table 2 for an example of the spreadsheet used to manage run-time values.

Manage Run-time Values

Using two spreadsheets in larger companies will usually be prudent. One spreadsheet is used for application-specific values determined by developers, and the other spreadsheet contains run-time infrastructure values. Security, change control procedures, or simply the size of an organization may require that a production control team separate from developers maintain the infrastructure spreadsheet. A Microsoft Access-type database can also be used, but you lose the advantage of file-based version control. If you have properties files all over the place, run—don't walk—to your favorite spreadsheet tool.

At this point we are saying it is best to take an EAR file from development and modify it so that it will work in another environment. We get the files and values to modify from a spreadsheet or two that has been exported to a usable format. Now what we want to do is to take the files we need to modify out of the archives inside the EAR file, modify them, and put them back and reassemble the EAR file.

You can extract individual files from archive files such as JAR, WAR, and EAR files with a command-line JAR program or the Ant Unjar task, but you can't extract a file inside of an archive that is inside another archive. You can't directly extract a web.xml file inside a WAR file that is archived in an EAR file. First, you will extract all of the contents of the EAR file and then operate on each archive in turn. Our ZIP file analogy was over-simplified; we are actually operating on ZIP files within ZIP files.

These are the different types of files that need to be updated for each environment:

- XML files – Many tools are available for manipulating XML. XPath lets you target elements and attributes to change without the need for parameterizing the XML by adding tokens. There are excellent Ant tasks (XMLTask) and Perl modules (XML:XPath) that provide this functionality.
- Properties files – These common files are defined by the java.util.Properties class and are basically simple lists of name-value pairs in a file. Ant is great for reading properties files, but has limited facility for regenerating them. The standard Ant tasks prefer the files to be parameterized ahead of time. You can also write a simple Java program for more sophistication, or if you are comfortable with Perl, it can be as easy as processing a simple hash, or sophistication can be added using Perl's powerful regular expressions.
- Other text files – As much as we try for standardization, some radical open

Quick Reference: Managing Deployment Descriptors

Use this process to help you with the complexities of managing J2EE deployment descriptors:

1. Manage run-time values in one or two spreadsheets.
2. Start with a copy of a development EAR file and the two spreadsheets exported into your favorite usable format.
3. Get a list of files and values to change from the spreadsheet data for each target environment.
4. Use Ant's Unjar to extract files to change in the development EAR file.
5. Bring in any parameterized versions of developer files that you had to use.
6. Apply XPath, Ant filters, and regular expressions to modify the files.
7. Put the modified files back into the EAR file.
8. Repeat these steps for every instance of each environment.

source group or upstart company is bound to throw in some oddly formatted files you can't do without. You may have to parameterize these files so that Ant can filter them or use Perl regular expressions.

- Other – There may be files that simply can't be processed and must be used as is. These can be binary or license files required by third-party components. For Ant, the typical way to handle them is to set a property value indicating the location of the file to use for the current instance that you are building. For build tools like Openmake you simply set the search path to point to the file to use.

Because the files are of different types and formats, different tools are useful for making the different types of updates. We have seen at customer sites that these different classes of files and the corresponding approaches have proven effective for making controlled updates.

Rebuild the EAR File

Now that you've extracted the files and modified them or replaced them, you need to reinsert the files back into the archives they came from and re-archive the EAR file to complete the change that allows it to be deployed to another server. Be super-organized about which files are source files and which are the modified files, and you should be fine. It is important to clearly separate files that are used as source and files that are modified for re-archiving. After re-archiving the new EAR file, you are ready to start on the next one.

We trust that these tips that come from real-life success stories will help eliminate some of the trial and error that is inevitable in developing a customized solution. In larger shops the resources required to manage all of the run-time infrastructure parameters is significant, and the spreadsheet is the right tool to use. A lot of coordination, documentation, and training is still required to pull this off, and if you are writing a customized solution, you are in for a lot of script writing and testing. (See the sidebar, "Quick Reference: Managing Deployment Descriptors," for a summary of our suggested best practices.)

Managing multiple deployment descriptor run-time values is not an easy task. Many organizations have not recognized the problems and instead have an ad hoc approach to manipulating EAR, WAR, and JAR files just before they are released to production. As the Java development process grows along with increasing complexity of business applications, the need for managing this data in an organized, well-documented method increases. These tips will help in beginning a process to streamline your own internal solution. JP

Sean Blanton develops, supports, sells, and evangelizes Openmake build management software for Catalyst Systems Corporation. In the course of his work with the services side of Catalyst he has helped over 100 companies manage software change processes and builds (particularly). Sean has a Ph.D. in physics from the University of Chicago and a B.S. from Columbia University. Contact Sean at **sean@openmake.com**.

It's All Here:

code, content from the magazines, unique content, free archives, discussion groups, and more.

All the technical content you've come to know and trust from the industry leader in providing hands-on, how-to information—in one place.

Visual Studio and Windows Server System are trademarks of Microsoft Corp.
Java is a trademark of Sun Microsytems.

FTPOnline
www.ftponline.com

Visual Studio Magazine
www.visualstudiomagazine.com

JAVAPro
www.javapro.com

Windows Server System Magazine
www.winservermag.com

WebLogicPro
www.WebLogicPro.com

Enterprise Architect
www.enterprise-architect.net

XML & WEB SERVICES MAGAZINE
www.xml-mag.com

Troubleshoot High CPU Issues

Got high CPU Issues? Explore operating system commands specific to JVMs to detect the cause of the problem

by Steve **POZARYCKI**

High CPU issues can occur for many reasons: your application server (or stand-alone Java application), user-created threads, bad coding practices, or third-party software. Unfortunately, determining where this happens can sometimes be difficult. If your Java applications ever encounter high CPU issues, you will find the tips for debugging presented here to be useful.

You can attempt to help troubleshoot this problem by using operating system-specific commands and gathering data. All of the information for these operating systems is based on the Sun JVM. However, the same information is provided from other JVMs (such as the BEA JRockit JVM) so the same concepts apply.

Here we'll look at examples of troubleshooting this problem on Solaris and Microsoft Windows systems. Information for Hewlett-Packard, AIX, and Linux can be found on the BEA support Web site (see Resources online at www.javapro.com).

Note that because HP does not have a prstat-like command available there is a binary executable that I developed that is available on the BEA Support Web site that you can download to gather the necessary information on HP. The information presented here was run on the WebLogic Platform, but the troubleshooting methodology will apply to any Java application.

Solaris Troubleshoot

Run the prstat command on the Java process, and do this for several iterations so you can see a pattern. For example:

```
prstat -L -p <PID> 1 1
```

If you do not have prstat available on your Solaris installation, do this instead:

```
ps -Le -o pid,user,s,lwp,pcpu,
  args | awk '$3 != "S" { print
  }'
```

where the output will show the Process id (PID), user, state, and the lightweight thread number. An example is something like this:

```
0 root T 1 0.0 sched
2769 root O 1 0.5 ps -Le -o
  pid,user,s,lwp,args
```

Listing 1 Output Redirect

```
$ prstat -L -p 9499 1 1
PID USERNAME SIZE RSS STATE PRI NICE TIME CPU
  PROCESS/LWPID
9499 spoz 153M 100M sleep 58 0 0:00.22 0.6% java/8
9499 spoz 153M 100M sleep 58 0 0:00.10 0.2% java/10
9499 spoz 153M 100M sleep 58 0 0:00.11 0.1% java/9
9499 spoz 153M 100M sleep 58 0 0:00.03 0.0% java/5
9499 spoz 153M 100M sleep 58 0 0:01.01 0.0% java/1
9499 spoz 153M 100M sleep 58 0 0:00.00 0.0% java/12
9499 spoz 153M 100M sleep 58 0 0:00.00 0.0% java/11
9499 spoz 153M 100M sleep 58 0 0:00.00 0.0% java/14
9499 spoz 153M 100M sleep 58 0 0:00.00 0.0% java/13
9499 spoz 153M 100M sleep 59 0 0:00.07 0.0% java/7
9499 spoz 153M 100M sleep 59 0 0:00.00 0.0% java/6
9499 spoz 153M 100M sleep 59 0 0:00.00 0.0% java/4
9499 spoz 153M 100M sleep 58 0 0:00.11 0.0% java/3
9499 spoz 153M 100M sleep 58 0 0:00.00 0.0% java/2
```

In the pstack command's output file redirect the java/8 process is at the top of prstat. The pstack output for lwp# 8 maps to thread# 76 from the pstack output (see Listing 2).

Go Online

Visit www.javapro.com for related resources. Simply type the Locator+ code into the field in the upper-right corner of the page.

Download

JP0509 Download all the code for this issue.

Discuss

JPTALK Discuss this article in the "Talk to the Editors of *Java Pro* Magazine" discussion forum.

Read More

JP0509SP_T Read this article online.

JP0407WL_T Read the related article "Troubleshoot Multicast Issues on BEA WebLogic Server" by Steve Pozarycki.

```
3383 spoz R 27 0.1
  /wwsl/sharedInstalls/solaris/j2
  sdk1.4.2_04/bin/java -client
  -Xms32m -Xmx200m –X
3383 spoz R 31 0.1
  /wwsl/sharedInstalls/solaris/j2
  sdk1.4.2_04/bin/java -client
  -Xms32m -Xmx200m –X
2770 spoz R 1 0.3 awk $3 != “S”
  { print }
```

You can then use the information from the fourth column of the output for the Light Weight Process (LWP) number in these steps. Run the pstack command on the Java process to get a mapping of the LWP to the PID. For example:

```
pstack 9499
```

Redirect the output to a file. If you use regular thread libraries (that is, you do not have /usr/lib/lwp in the LD_LIBRARY_PATH) in Solaris, a LWP is not mapped to the OS thread directly, so you have to take pstack from the process (check to see if you are using the alternate thread library).

Next, get several thread dumps of the java application over time to make sure you get the right threads executing. You can do this by doing a kill -3 <PID> on the Java process. Then map the LWP ID to the Java Thread ID. For example, if the LWP in question is 8, it could map to the Java thread 76. We would then calculate the hex value of 76 to be 0x4c.

Look in the thread dump to find the thread that matched nid=0x4c, and you will have the thread that is consuming the CPU. You will then have to figure out why this is happening in your code or if the top of the stacktrace is from WebLogic. Then you can contact support.

Let's look at an example of this process. Run the pstack command (pstack 9499) and redirect the output to a file (see Listing 1). If you use regular thread libraries (you do not have /usr/lib/lwp in the LD_LIBRARY_PATH) in Solaris, a LWP isn't mapped to the OS thread directly, so you have to take pstack from process (check to see if you are using the alternate thread library). Listing 1 shows the java/8 process at the top of prstat. Let's look at the pstack output for lwp# 8. It maps to thread# 76 from the pstack output (see Listing 2). Next get a thread dump of the java application by doing a kill -3 <PID> on the Java process. Since lwp# 8 is mapped to thread# 76, we calculate the hex value of 76 to be 4c, which maps to nid=0x4c in the JVM thread dump:

Listing 2 Thread Dump

```
---------------- lwp# 8 / thread# 76 ----------------
ff29d190 poll (e2e81548, 0, bb8)
ff24d154 select (0, 0, 0, e2e81548, ff2bf1b4,
  e2e81548) + 348
ff36b134 select (0, bb8, 7fffffff, fe4c8000, 0, bb8) +
  34
fe0f62e4 __1cCosFsleep6FpnGThread_xl_i_ (0, bb8,
  fe4c8000, 1, 0, 1e2fd8) + 234
fe23f050 JVM_Sleep (2, 0, bb8, fe4de978, fe4c8000,
  1e2fd8) + 22c
0008f7ac ???????? (e2e818d4, bb8, 1e2fd8, 984a4, 0,
  109a0)
0008c914 ???????? (e2e8194c, 1, fe4d6a80, 98564, 8,
  e2e81868)
fe5324e8 __1cMStubRoutinesG_code1_ (e2e819d8, e2e81c10,
  a, f6cb5000, 4, e2e818f0) + 3ec
fe0cbe94 __1cJJavaCallsLcall_helper6FpnJJavaValue
  _pnMmethodHandle_pnRJavaCallArguments_pnGThread__v
  _ (e2e81c08,fe4c8000, e2e81b54, 1e2fd8, 8e764,
  e2e81c10) +308
fe1f6dbc __1cJJavaCallsMcall_virtual6FpnJJavaValue
  _nLKlassHandle_nMsymbolHandlee81c08, e2e81b54) +
  150pnGThread__v_(f6cb64b8, e2e81b40, e2e81b44,
  fe4c8000, e2d8) + 60e_5pnGThread__v_ (e2e81c08,
  e2e81c04, e2e81c00,e2e81bf4, e2e81bec, 1e2f8000,
  e2e81d10, 1e, e) + 120FpnKJavaThread_pnGThread__v_
  (f6817ff8, 1e2fd8, fe4c7fd70) +
  3d8cKJavaThreadDrun6M_v_ (e2e02000, fe4d3e34,
  fe4c8000, 7fd70, 1e2fd8,fe213ec8 _start (fe4c8000,
  fe625d10, 0, 5, 1, fe401000) + 20
ff36b728 _thread_start (1e2fd8, 0, 0, 0, 0, 0) + 40
```

Do a kill -3 <PID> on the Java process to get a thread dump of the java application. Since lwp# 8 is mapped to thread #76, we calculate the hex value of 76 to be 4c, which maps to nid=0x4c in the JVM thread dump.

```
"Thread-6" prio=5 tid=0x1e2fd8
  nid=0x4c waiting on monitor
  [0xe2e81000..0xe2e819d8]
  at java.lang.Thread.sleep(
  Native Method)
  at weblogic.management.deploy.
  GenericAppPoller.run(
  GenericAppPoller.java:139)
```

In this example the thread using the most CPU is sleeping. This thread is specific to WebLogic, and it is an application poller that runs when WebLogic is started in Development mode. It runs every 30 seconds, so it is apparent that the thread dump was not captured in time to see activity in this thread. Ideally all three steps should be done quickly and in a row to capture the data as close together in time as possible. You can do so through a simple shell script like this:

```
#
# Takes an argument (PID of the
  Java process) and loops three
  times. This will append the
  prstat information to a file
  called dump_high_cpu.txt. The
  thread dump information will
  either be in file where stdout
  was redirected or printed on
  the screen.
#
for loopnum in 1 2 3
do
  prstat -L -p $1 1 1 >>
    dump_high_cpu.txt
  pstack $1 >> dump_high_cpu.txt
    kill -3 $1
  echo “prstat, pstack, and
    thread dump done. #” $loopnum
  sleep 1
  echo “Done sleeping.”
done
```

Windows Troubleshoot

You can use pslist and get thread details of the java process on Microsoft Windows. The pslist command is freely available from the Sysinternals freeware site (see Resources online at www.javapro.com). Run pslist -d <Java PID> and redirect the output to a file, and do this for several iterations so you can see a pattern. You should see the user time and kernel time increasing. Take thread dumps of the application over several iterations by pressing Ctrl-Break when the application is running in a command window. If the application is WebLogic and you have it running as a service, then you can execute a separate command to get a thread dump on Windows (see Resources online at www.javapro.com).

Take the Thread ID # that seems to be increasing, change the decimal value to a hex value (you can use the Calc function in Windows), and look in the thread dump for the nid=0x<Your Hex Value> until you find the offending thread. You will then have to figure out why this is happening in your code or if the top of the stacktrace is from WebLogic (for example), and then you can contact BEA support.

Or, you can also use the Process Explorer GUI from Sysinternals (see Resources online at www.javapro.com). This tool shows the CPU usage dynamically. Unfortunately, there is no logging or history in Process Explorer, so you will have to monitor it and write down the Thread id from the Java process that is taking up the most CPU. To do this with Process Explorer:

1. Find the java process, then right click and select properties.
2. Click the Threads tab to bring up all the threads associated with this java process. You can click on one of the threads that are listed as MSVCRT.dll+<Some hex offset>. You can see the Thread ID listed in the pane below it.
3. Observe which thread is using up the most CPU.
4. Take thread dumps of the application (as described previously for Windows).
5. Take the Thread ID # from step 2, and change the decimal value to a hex value (you can use the Calc function in Windows).
6. Look in the thread dump for the nid=0x<Your Hex Value> until you find the offending thread.

You will have to figure out why this is happening in your code or if the top of the stacktrace is from WebLogic (for example), and then you can contact BEA support.

Here's an example of this procedure using using pslist and thread dumps. Run pslist -d 1720 (the output shows only a couple of threads):

```
java 1720:
Tid Pri Cswtch State User Time
  Kernel Time Elapsed
2000 8 2 Wait:UserReq
  0:00:00.000 0:00:00.000
  0:04:41.965
588 9 4744 Wait:UserReq
  0:00:02.814 0:00:00.110
  0:04:41.965
1784 9 132 Wait:UserReq
  0:00:00.080 0:00:00.000
  0:04:41.955
1756 9 196 Wait:UserReq
  0:00:00.931 0:00:00.000
  0:04:41.955
1716 8 6 Wait:Queue
  0:00:00.000 0:00:00.000
  0:04:41.945
```

We take the same output again some time later to get another snapshot of the threads to see which have increased significantly, and doing so should point us at the Thread id (TID) to look at. We can then run pslist again (pslist -d 1720):

```
java 1720:
Tid Pri Cswtch State User Time
  Kernel Time Elapsed Time
2000 8 657 Wait:UserReq
  0:00:00.090 0:00:00.010
  0:08:01.211
588 10 59123 Wait:UserReq
  0:00:48.830 0:00:02.633
  0:08:01.211
1784 8 150 Wait:UserReq
  0:00:00.090 0:00:00.000
  0:08:01.201
1756 8 251 Wait:UserReq
  0:00:00.941 0:00:00.000
  0:08:01.201
1716 8 6 Wait:Queue
  0:00:00.000 0:00:00.000
  0:08:01.191
```

We now see that Thread id 588 is using up most of the user/kernel time and hence the CPU. Obviously this thread has a problem. We can then take the Thread id number of 588 and convert that to hexadecimal (0x24). We can then look at the thread dumps we took while the problem was occurring and look for nid=0x24. We see that this corresponds to ExecuteThread 10 from the thread dumps (see Listing 3 online at www.javapro.com).

We can see that it is on a socketWrite() native method, but it looks like the HelloWorld2.service() is doing something wrong. We can then look at that line number (94 of HelloWorld2.java) to see what is going on. From this code snippet of HelloWorld2.java from the service() method:

```
89 out.println(
  ExampleUtils.returnHtmlHeader(
  "Hello World 2"));
90 out.println("<h4>");
91 for (int i=0;i<100000000;i++)
  {
92 int j = 0;
93 j = j +i;
94 out.println(defaultGreeting +
  " " + defaultName + "!");
95 }
96
97 out.println("</h4>");
98 out.println(
  ExampleUtils.returnHtmlFooter()
  );
```

we see that for some reason the output stream is being written in a very long for loop, which is the error and cause of the high CPU in this case. If this code is corrected, then the CPU will not be maxed out.

Once you have the list of steps needed on your operating system to figure out high CPU issues, you should be able to more easily determine the cause of the issue. You can even put those simple steps into a script or cron job that can be run to monitor your application in a quick and easy manner. JP

Steven Pozarycki is a senior backline developer relations engineer with BEA Systems, Customer Support Group. He specializes in troubleshooting and solving complex customer issues with their mission-critical applications on BEA products. Contact Steve at **spoz@bea.com**.

FTPOnline Special Reports:

✓ Code Quality
✓ SOA
✓ Extensible ESB

Presenting in-depth special reports on critical topics important to all IT professionals. Check out these and our other must-read technical articles, tips, and market trends.

Go to: www.ftponline.com/special

✓ Code Quality

- Performance analysis for Web Services
- Stress-test Web forms and services
- Write unit tests

✓ SOA

- Competitive advantages
- Planning the transition

✓ Extensible ESB

- Cure integration headaches
- Extensible integration

✓ And Don't Miss Our Reports On:

- J2EE
- Security
- Storage & Disaster Recovery
- Exchange
- Testing & Performance
- Lifecycle Management
- Operations Management

FTPOnline

© 2005 Fawcette Technical Publications, Inc.

Moving to Modeling

by Peter VARHOL

Don't expect your development team to look the same when building applications through models

The rapid adoption of Java and object-oriented programming techniques, coupled with advances in software technology, has brought about application development techniques that were the stuff of imagination only a decade ago. Today it is at least theoretically possible to draw a structured diagram of an application solution, and generate that solution as source code, or even a full executable application. That application could run on multiple systems, as a distributed application, with the software components appropriately partitioned.

Much of this innovation was accomplished under the support of the Unified Markup Language (UML), and its prodigy the Model-Driven Architecture (MDA). UML enables developers to use diagrams to precisely specify the characteristics and activities of a software system. MDA moves past that to enable business analysts to specify a solution to a business problem using UML, and then apply transformations to automatically produce the software implementing the solution. Both of these technologies are standardized under the auspices of the Object Management Group (see Resources online at www.javapro.com).

There are other modeling approaches, including some that are based on a formal, well-defined syntax and semantics. Whichever one you may use, it is likely that you adopted it because of the promise of bringing application development closer to the business problems it tries to solve.

The advantages of such a move are readily apparent. By tying the business problem closer to the technology solution, it becomes more likely that you successfully address that problem. There is less a chance that the problem and solution veer off in different directions because of poor communication. Modeling should also help speed up the development process, which gets solutions into the business more quickly, and modeling should also improve application quality—a problem that continues to plague enterprise applications. Because models can be tested for accuracy, and because the presumption is that code generation can produce higher-quality (though not perfect) code, applications should be more reliable.

A Rocky Road

If you are typical, the move from a code-based development process to a model-based one has been rocky. In all likelihood, you started with a pilot project that used either MDA or a similar UML-based approach to defining the application, coupled with code generation to implement the application. The project you chose was important, but not particularly demanding or time critical.

In many cases, the problems started almost immediately. If you look across different development teams, projects, and organizations, the problems can seem random in nature. In some cases, the modeling and code generation tools don't seem to work properly. In others, the pilot project selected didn't seem to fit well into the tools they acquired for the project. Sometimes the learning curve was too long, and didn't result in the promised improvements in productivity.

The ability to define applications using models and generating code is not a panacea. First, the syntax and semantics of the diagramming language of choice is complex and can be difficult to learn and use. Second, the types of applications you can write will be limited by the expressiveness of the solutions available from modeling languages. While these will likely expand over time, there will always be application architectures that can't be replicated easily through a modeling approach.

There can be many possible causes of problems in adopting modeling tools and techniques, such as insufficient training or poor project definition. But many such problems can be traced back to a team that was organized for the last project, which was done in the traditional way, by writing code to implement requirements described in text. Developing applications using models rather than code is a significant break from past practices, and many development teams don't do enough to prepare for those changes.

Last, and possibly most significant, enterprise IT groups aren't well-structured to take advantage of the efficiencies of modeling, which isn't a knock on enterprise IT, but rather a recognition of the reality that different skills and development processes are needed in an age of application mod-

Go Online

Visit www.javapro.com for related resources. Simply type the Locator+ code into the field in the upper-right corner of the page.

Download

JP0509 Download all the code for this issue.

Discuss

JPTALK Discuss this article in the "Talk to the Editors of *Java Pro* Magazine" discussion forum.

Read More

JP0509OE_T Read this article online.

JP0406PV_T Read the related article "Building a Better Application Life Cycle" by Peter Varhol.

EAEP040205MB_T Read the related article "Build Software Like Building Bridges" by Mike Burba.

EA0401RM_T Read the related article "Model Strategy and Mission in UML Profiles" by Rick Murphy.

eling. This kind of organizational change can be difficult for many to grasp.

How would you structure an IT team to use a modeling approach to develop applications? Most keep the existing team structure, which might look something like this:

- An application architect whose duties are likely divided among several applications
- A development manager responsible for assignments, scheduling, and communication within and outside of the group
- One or more technical leads who had a broad and deep understanding of the technologies being used in the project
- Several to many developers, ideally including a mix of both senior- and junior-level skills and experience

This type of structure can create some inefficiencies when applied to a model-based development approach, because the development process is fundamentally different. Take, for example, MDA, which provides for the specification of a solution to a business problem in a set of UML diagrams and the application of transformations to those diagrams to produce successively more exacting code implementations.

Pick a Pattern

MDA requires some of these same skills, but they are often performing different roles. Because the transformations within the modeling environment largely determine the architecture, the architect may not be directly involved in the specific application. However, he or she has likely written at least some of those transformations, based on application development and deployment characteristics used in the organization.

The architect will also create design and implementation patterns for common activities that tend to be governed by formal or ad hoc practices within the IT groups. Patterns provide for consistent design and implementation across applications and development teams, which benefit both end users and those charged with maintaining applications over their useful lives.

The traditional development manager role is likely to remain similar in the modeling team because plans and communication remain a high priority. Schedules, resource allocation, direction, and status requirements are unlikely to change significantly, no matter what project technologies are employed.

However, there is less a need for a technical lead in the modeling team because there is less a need for deep expertise in both the language and the platform. Most of the design and implementation details are encompassed in the modeling tools, which in effect act as the technical advisor.

Perhaps most significant, the individual contributors are likely to be very different. Because coding skills won't matter as much in a successful modeling strategy, business or domain expertise is likely to be a higher priority.

In fact, business or domain expertise is one of the key tenets behind the promotion of modeling as a valuable development technique. As older mainframe-based enterprise applications are either retired or transformed into Web services, there are many veteran developers with outdated technical skills but savvy domain knowledge. Training these professionals in Java has been no more than modestly successful, so combining a deep business expertise and some Java skills with MDA-type modeling seems to be the path many are taking.

So what would a modeling development team look like? First, the architect would be removed from application design, and from any day-to-day involvement in design and implementation. There could be some initial feedback on the viability of the design model, but this will be in more of an advisory than a directive capacity.

The other big difference is in the technical team. Pure technology and language skills won't dominate this team; instead, individual contributors will likely have more knowledge of the business processes in that organization. Coding won't disappear, but it will largely be relegated to the implementation of defined patterns, building services wrappers around legacy code, and customizing existing applications, especially off-the-shelf applications. However, being able to comprehend the movement of data through the business, and the information needs of decision-makers, will be the key set of skills.

What does this type of project team mean for someone who has or is in the process of developing deep technical skills on the Java platform? From an individual's standpoint, it means that you have to rethink your career path. If your interests today are more toward the language and platform, however, doing so doesn't necessarily mean that the emerging emphasis on modeling will leave you out in the cold. Architects will still require sharp coding skills, as well as extensive experience in alternative technology platforms.

Transferable Skills

And enterprise application development isn't the only path to a software career. Technology and language expertise are still essential for those who design and build modeling solutions and other software development tools, as well as commercial applications in general. Technology skills will always be transferable between commercial and enterprise projects; whereas, the same can't necessarily be said about business domain knowledge.

All of this information means that many of the skill sets traditionally used in enterprise application development must change to be successful in a modeling project. Specifically, any development team embarking on a project employing modeling and code generation today has to shift the focus of training and execution to the business rather than the technology.

> Understanding the business process makes it even more possible to take a given problem and model the solution

In many cases, the requirements are incomplete or ambiguous, even if they have also been modeled. This ambiguity has been the source of many failed or poorly implemented projects in the past, which is where domain knowledge can be essential. Understanding the business process makes it more possible to take a given problem and model the solution because you have a better context with which to make decisions. To be fully successful in making application development more of a modeling than a coding activity, developers will out of necessity move closer to the business at hand. JP

Peter Varhol is a senior member of the technical staff for Progress Software. Contact Peter at **peterv@mv.mv.com.**

Take a JFace Detour

by Kevin **JONES**

You've become acquainted with creating an Eclipse plug-in. Now discover a valuable toolkit for building a viewer

Previously in this space I discussed Eclipse plug-ins, and we got our feet wet creating, archiving, and installing an Eclipse plug-in (see "Get Acquainted with Eclipse Plug-Ins" *Java Pro*, Vol. 9, No. 4). I also mentioned that as we develop the plug-in example, we might need to take some diversions along the way. Here is the first of those diversions. Let's take a look at JFace.

While SWT provides a set of raw widgets that we can use in our applications, JFace is a higher-level toolkit. JFace layers on top of SWT and allows for model-driven widgets. That is, in a JFace application a developer can keep the separation between the model (the data) and the user interface (the widget). JFace also offers editors, dialogs, and wizards that SWT does not. In general it is easier to work with JFace than it is to work with SWT.

One of the primary areas that JFace improves SWT is with the use of such things as tables and trees. JFace has the concept of a viewer. A *viewer* is a class that brings together content and the means to display that content. For example, it associates an SWT tree with its data, or an SWT table with its content. Building Eclipse plug-ins is often about displaying data to the user; therefore, taking advantage of JFace's support for the model-driven viewers will make application development easier.

Before thinking about using JFace to display data, and before looking at using JFace in our viewer, let's first build a simple JFace application to show which parts are needed. You might remember from a previous column that the initial code when building an SWT application looks something like this:

```
display = new Display();
shell = new Shell(display);

shell.open();

while (!shell.isDisposed())
{
  if (!display.readAndDispatch())
    display.sleep();
}

display.dispose();
```

In SWT you create a display, create a shell, and then enter the message loop. When the message loop terminates you dispose of the display and exit the application. A simple JFace application on the other hand looks like you see in Listing 1. Notice that there is no direct access to the shell, and the only time the display is accessed is to dispose of it. Notice also that there is no message loop. So what's going on?

First, our class extends the org.eclipse.jface.window.ApplicationWindow class, which in turn extends org.eclipse.jface.window.Window, and it is these two classes that do most of the work. The call to open() inside the run() method causes JFace to create the main window of the application. During this creation process the Window class calls the (overridden) createContents() method. It is inside createContents() that we create the widgets to be displayed within the main window. After createContents() returns, the open method checks the block flag; if this flag is set then the open method runs the SWT message loop. The block flag is set in the call to setBlockOnOpen(boolean). We are passing true to this method, so the event loop is indeed processed.

Tables and Viewers

JFace runs on top of SWT. As with the SWT JARs and linked/shared library, you will also need (at least) org.eclipse.core.runtime_3.1.0.jar, org.eclipse.jface.text_3.1.0.jar, org.eclipse.jface_3.1.0.jar, and org.eclipse.swt.win32.win32.x86_3.1.0.jar on your classpath.

The prior example shows one of the strengths of JFace: code reduction. In this example we no longer have to write an event-processing loop; less code means fewer bugs. Another major benefit of using JFace is that it enables us to use a separation of concerns when developing components. The concerns in this

Go Online

Visit www.javapro.com for related resources. Simply type the Locator+ code into the field in the upper-right corner of the page.

Download

JP0509 Download all the code for this issue.

JP0509PI Download the code for this article separately. The code demonstrates building a JDBC Viewer plug-in for the Eclipse platform.

Discuss

JPTALK Discuss this article in the "Talk to the Editors of *Java Pro* Magazine" discussion forum.

Read More

JP0509PI_T Read this article online.

JP0506PI_T Read the related article "Get Acquainted with Eclipse Plug-Ins" by Kevin Jones.

JP0505PI_T Read the related article "Eclipse SWT 101" by Kevin Jones.

JP0212PS_T Read the related article "Eclipse vs. Swing" by Daniel F. Savarese.

Listing 1 Display Disposal

```
package com.develop.kevinj.jface;

import org.eclipse.jface.window.ApplicationWindow;
import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Control;
import org.eclipse.swt.widgets.Display;
import org.eclipse.swt.widgets.Label;

public class Simple extends ApplicationWindow
{
   public Simple()
   {
   super(null);
   }

   public static void main(String[] args)
   {
   Simple s = new Simple().run();
   }

   private void run()
   {
   setBlockOnOpen(true);
   open();
   Display.getCurrent().dispose();
   }

   @Override
   protected Control createContents(Composite parent)
   {
   Label label = new Label(parent, SWT.CENTER);
   label.setText("Hello World!");
   return label;
   }

}
```

In this simple JFace application there is no direct access to the shell—and there is no message loop. The only time the display is accessed is to dispose of it.

case are the data and the means to display that data.

While we can use SWT widgets directly and display data using those widgets, JFace provides viewers, and it is their job to manage the relationship between the application's data and how that data is displayed. idea is that the data model holds, or knows how to get, all of the data that is needed to be displayed. The content provider knows how to get a single row of data from the model, and the label provider knows how to display that row on the table.

> JFace provides Viewers, and it is their job to manage the relationship between the application's data and how that data is displayed

When we get back to our plug-in we will display database data, which by its very nature is tabular, and we will use JFace's table viewers. However, before we do that we need to see how this viewer works.

We are going to build this application in steps. Rather than build the JDBC code into the plug-in we built in my previous column, we will write a stand-alone application, and then—next issue—we will integrate the code and add some bells and whistles.

To build a viewer we need four components: the viewer class, which in our case will be the TableViewer; the model that holds the data for the viewer; and two providers, one to provide the content for the viewer and one to provide the display of that content. The viewer orchestrates the other three classes. The

One extra issue in our case is that the data we want to display is dynamic. We don't know anything about the tables to be displayed until we execute the query. We don't even know the column names or even how many columns there are. The work to create and display our data is done in the CreateContents() code, which looks something like this:

```
protected Control createContents(
  Composite parent)
{
  Composite composite = new
    Composite(parent, SWT.NONE);
  composite.setLayout(
    new GridLayout(1, false));

  TableViewer tv =
    createTableViewer(composite);
  Table table = initializeTable(
    tv);

  JDBCTableModel model = new
    JDBCTableModel(driverClass,
    connectionString, query);

  applyModel(tv, table, model);

  return composite;
}
```

Note that the real code has exception handling, but it is omitted here to make the printed code more readable. Remember the method returns type is Control, so we cannot return a TableViewer directly. Instead, we create a Composite (which is simply a container for other controls), make the Composite the parent of the TableViewer, and return the Composite. We create the viewer in the createTableViewer() method, which looks something like this:

```
private TableViewer
  createTableViewer(
  Composite composite)
{
  TableViewer tv = new
    TableViewer(composite,
    SWT.SINGLE | SWT.H_SCROLL
                | SWT.V_SCROLL);
    tv.setLabelProvider(
      new JDBCLabelProvider());
    tv.setContentProvider(
```

Listing 2 Call Turnaround

```
public class JDBCContentProvider implements
   IStructuredContentProvider
{
   public Object[] getElements(Object input)
   {
   return ((JDBCTableModel)input).getElements();
   }

   public void dispose()
   {
   }

   public void inputChanged(
   Viewer arg0, Object arg1, Object arg2)
   {
   }
}

public class JDBCTableModel
{
   List<List> table = new ArrayList<List>();
   …
   public Object[] getElements()
   {
   return table.toArray();
   }
   …
}
```

A reference to the model provided in the setInput() call is passed to the getElements(Object input) method that turns that data into a form the viewer understands, namely, an array of objects. Simply turn around and call the model's getElements() method to return its data as an array of List objects.

```
      new JDBCContentProvider());
    return tv;
}
```

This code creates the viewer and specifies the label and content providers (more on these in a moment). Finally, we create the model and call the helper applyModel() method. The applyModel() method looks like this:

```
private void applyModel(
  TableViewer tv, Table table,
  JDBCTableModel model)
{
  int colCount =
    model.getColumnCount();
  for (int i = 0; i < colCount;
    i++)
  {
    TableColumn col = new
      TableColumn(table, SWT.LEFT);
    col.setText(
      model.getColumnText(i));
  }

  tv.setInput(model);

  for (int i = 0; i < colCount;
    i++)
  {
    table.getColumn(i).pack();
  }
}
```

Models and Providers

The applyModel() method does three things. First, it gets the underlying table and creates the columns to be added to the table. The column names are retrieved from the model. Next, it calls the viewer's setInput() method that applies the data to the viewer and triggers the viewer to call the providers to display the data. Finally, it calls pack on each column. You have to either call pack or set the column widths individually, otherwise the columns do not display.

The createContents() code shown previously creates the model, which is of type JDBCTableModel and will hold the data associated with the table viewer. This model has to do a couple of things. It has to access the specified database and run the specified query. The data from the query has to be made available to the content provider to be displayed in the viewer. The model also has to get the ResultSet's metadata, which will tell us how many columns there are and what those columns are called. This metadata is used in the applyModel() method to initialize the table.

The data presented by the model has to be generic, as we don't know ahead of time the sort of data that a given query will execute. To manage this data the model holds each row in the returned result set as a List of strings (one per column), and the table is then modeled as a list of Lists (one per row). The code looks like this:

```
ArrayList<List> table =
  new ArrayList<List>();
```

au_id	au_lname	au_fname	phone	address	city	state	zip	contract
172-32-1176	White	Johnson	408 496-7223	10932 Bigge Rd.	Menlo Park	CA	94025	1
213-46-8915	Green	Marjorie	415 986-7020	309 63rd St. #411	Oakland	CA	94618	1
238-95-7766	Carson	Cheryl	415 548-7723	589 Darwin Ln.	Berkeley	CA	94705	1
267-41-2394	O'Leary	Michael	408 286-2428	22 Cleveland Av. #14	San Jose	CA	95128	1
274-80-9391	Straight	Dean	415 834-2919	5420 College Av.	Oakland	CA	94609	1
341-22-1782	Smith	Meander	913 843-0462	10 Mississippi Dr.	Lawrence	KS	66044	0
409-56-7008	Bennet	Abraham	415 658-9932	6223 Bateman St.	Berkeley	CA	94705	1
427-17-2319	Dull	Ann	415 836-7128	3410 Blonde St.	Palo Alto	CA	94301	1
472-27-2349	Gringlesby	Burt	707 938-6445	PO Box 792	Covelo	CA	95428	1
486-29-1786	Locksley	Charlene	415 585-4620	18 Broadway Av.	San Francisco	CA	94130	1
527-72-3246	Greene	Morningstar	615 297-2723	22 Graybar House Rd.	Nashville	TN	37215	0
648-92-1872	Blotchet-Halls	Reginald	503 745-6402	55 Hillsdale Bl.	Corvallis	OR	97330	1
672-71-3249	Yokomoto	Akiko	415 935-4228	3 Silver Ct.	Walnut Creek	CA	94595	1
712-45-1867	del Castillo	Innes	615 996-8275	2286 Cram Pl. #86	Ann Arbor	MI	48105	1
722-51-5454	DeFrance	Michel	219 547-9982	3 Balding Pl.	Gary	IN	46403	1
724-08-9931	Stringer	Dirk	415 843-2991	5420 Telegraph Av.	Oakland	CA	94609	0
724-80-9391	MacFeather	Stearns	415 354-7128	44 Upland Hts.	Oakland	CA	94612	1
756-30-7391	Karsen	Livia	415 534-9219	5720 McAuley St.	Oakland	CA	94609	1
807-91-6654	Panteley	Sylvia	301 946-8853	1956 Arlington Pl.	Rockville	MD	20853	1
846-92-7186	Hunter	Sheryl	415 836-7128	3410 Blonde St.	Palo Alto	CA	94301	1
893-72-1158	McBadden	Heather	707 448-4982	301 Putnam	Vacaville	CA	95688	0
899-46-2035	Ringer	Anne	801 826-0752	67 Seventh Av.	Salt Lake City	UT	84152	1
998-72-3567	Ringer	Albert	801 826-0752	67 Seventh Av.	Salt Lake City	UT	84152	1

Figure 1 | Data View The code was tested against Microsoft SQL Server and must pass three parameters: the name of the JDBC driver class, the JDBC connection string, and the SQL query to execute. You must also have the JDBC driver's JAR file on your classpath.

Another late night writing scripts to solve J2EE Build headaches?

Stop scripting. Use Openmake® and go home.

Openmake provides immediate solutions for executing J2EE builds. From the creation of multiple deployment descriptors to building inside or outside of the Eclipse IDE, Openmake resolves the often complex J2EE build environment. Openmake builds are consistent, efficient and 100% reusable. Openmake integrates with all SCM tools for easy management of source objects. The Openmake SCM integration uses the SCM Repository to locate source code and track item history during the build. We don't stop there — Openmake provides real-time build monitoring, Audit (Bill of Material) Reports, Impact Analysis Reports, and Footprints, no scripting required. So go home and let Openmake run your J2EE builds.

openmake
a product of catalyst SYSTEMS CORPORATION

eclipse READY. Ready for IBM Rational software

www.openmake.com Call 800.359.8049 to schedule a web-based demo.

*The Ready for IBM Rational software Mark and the trademarks contained therein are trademarks of IBM Corp. IBM is not the licensor of this Business Partners™ product and does not make any warranties regarding this Business Partners™ product.

```
while(rs.next())
{
  ArrayList<String> row =
    new ArrayList<String>();
  for(int i = 0; i < columnCount;
    i++)
    row.add(rs.getString(i+1));
  table.add(row);
}
```

The table model also extracts the metadata for the table and stores it in data members. Once the model has been applied to the table viewer through the call to setInput(), the viewer starts calling the providers to get the data for a row and to specify how that data should be displayed.

The content provider implements the IStructuredContentProvider interface, and in our case we only care about one method: getElements(Object input). This method is passed a reference to the model provided in the setInput() call and turns that data into a form the viewer understands—notably an array of objects. In our code we simply turn around and call the model's getElements() method, which returns its data as an array of List objects (see Listing 2).

Each entry in the returned array is a row in the table. The viewer does not yet know how to display that row to know that it calls the table label provider. Our provider looks like this:

```
public class JDBCLabelProvider
  extends LabelProvider implements
    ITableLabelProvider
{
  public Image getColumnImage(
    Object element, int
    columnIndex)
  {
    return null;
  }

  public String getColumnText(
    Object element, int
    columnIndex)
  {
    List<String> data =
      (List<String>) element;
    return getText(data.get(
      columnIndex));
  }
}
```

Notice that we have no images to display, which is why the getImage() method returns null. The getColumnText() method is passed a reference to an element of the array provided by the content provider. It is called once per row, therefore, once per element. Our array contains a List<String>, and we simply cast the object to that type and then extract the column data specified by the index.

Run the Code

On my machine I have Microsoft SQL Server installed, and I tested this code against it. When you run the code you must pass three parameters (the code does zero checking that you pass three parameters, so be warned), the name of the JDBC driver class, the JDBC connection string, and the SQL query to execute. You must also have the JDBC driver's JAR file on your classpath. Using this code with SQL Server, the command line looks like this:

```
java –cp [classpath here]
  com.develop.kevinj.jface.
  jdbcviewer.JDBCViewer \
  com.microsoft.jdbc.sqlserver.
  SQLServerDriver \
  jdbc:microsoft:sqlserver:
  //localhost:1433;DatabaseName=
  pubs;user=sa;pwd= \
  "select * from authors"
```

(The \ indicates that these parts should all be on the same line.) The output is shown in Figure 1.

We've taken a beginning look at JFace, another core component of Eclipse, and we've seen two key parts of JFace: The fact that it hides a lot of the details of SWT and that it also layers good design practices over SWT. We've also covered viewers and in particular the table viewer. In an upcoming column we will integrate this knowledge into the plug-in we started previously. JP

Kevin Jones is a developer who researches and teaches Java programming and explores HTTP and XML. He lives in the U.K. and works for Developmentor, a training company based in the United States and Europe that specializes in technical training on Java and Microsoft platforms. Contact Kevin at **kevinj@develop.com**.

Advertiser Index

Catalyst Systems www.openmake.com	**5**
Enterprise Architect Online www.enterprise-architect.net	**27**
FTPOnline Resources www.ftponline.com	**C2, 1**
FTPOnline Special Reports www.ftponline.com/special/storage www.ftponline.com/special/testing www.ftponline.com/special/lifecycle	**31**
IBM www.ibm.com/developerworks/platform	**20-23**
InterSystems www.InterSystems.com	**C4**
Java Insight Newsletter www.javapro.com	**37**
Reporting Engines www.reportingengines.com	**7**
Software Architecture Summit www.ftponline.com/sas/	**14, 15**
Sybase www.sybase.com/workspace	**C3**

Advertising Sales Contact

➤ Robyn Johnson, Account Manager
650-378-7152
rjohnson@fawcette.com

This listing is provided as a courtesy to our readers and advertisers. The publisher assumes no responsibility for errors or omissions.
Sales Department: 2600 South El Camino Real, Suite 300 • San Mateo, CA, USA 94403 • 650-378-7100

Of Software and Sherman Tanks

by Daniel F. **SAVARESE**

In any team endeavor, organization, discipline, and method are the order of the day—programming is no exception

Early in my career, I believed that software development was fundamentally different from other disciplines. That's what the conventional wisdom was saying at the time, both inside and outside of academic circles. Today, I have no qualms asserting that I was too inexperienced to see those claims for the hogwash they were. Insiders and outsiders alike would flagellate the software industry for its bug-ridden products, inability to meet deadlines, and high project failure rates. They would claim we were doing everything wrong; yet, somehow, the critics could never spell out a provably right way to do it.

This preoccupation with perfection created a subindustry of its own. Innumerable software development processes were invented and marketed. Companies practicing these processes would sell software for implementing the processes, yet their products suffered just as many bugs and release schedule delays as those of their prospective customers. Book publishers created entire series dedicated to one development methdology or another. Ironically, a number of these books were written by programmers who used the processes they described on projects that were cancelled after failing to deliver according to schedule. The search for the holy grail of software engineering never bore fruit.

The More Things Change…

It was true in 1975 and it's true in 2005; there is no silver bullet that can subdue the sorrows of software development. Frederick Brooks explains the reason for this phenomenon in his well-regarded collection of essays in *The Mythical Man Month* (Addison Wesley, 1995). I would hope that most programmers who have been working for more than a few years have come to that realization on their own. Some things you only truly appreciate after learning them from the school of hard knocks.

If you believe software construction projects are more effective today than in the past, just study the FBI's recently cancelled virtual case file software project. It was started in 2002 and cancelled in 2005 after spending 170 million dollars. Or study Chrysler's famous Chrysler Comprehensive Compensation (C3) payroll software project that was cancelled after four years for not meeting its objectives, despite the acclaimed progenitors of extreme programming (XP) having formed the development team. The fact of the matter is that most software projects still fail, and even the best software team in the world can lay a rotten egg.

A secret about computer programming that many continue to deny is that it is just like everything else. I don't care if you're building a bridge, an airplane, or a house. The same factors that produce successful hardware projects produce successful software projects. The same factors that produce failed hardware projects produce failed software projects. I'm not referring to a specific process that you can follow like a recipe to guarantee success or failure. Instead, I'm referring to fundamental practices that are common to all effective development methodologies.

Outside of purely artistic or research endeavors, you can't build something if you don't know what it's supposed to do. That's why we gather requirements. You can't verify something does what it is supposed to do unless you can test it. That's why we develop test plans, and an entire taxonomy of testing has emerged to help us apply the appropriate kinds of tests to our particular software project. Again, outside of the arts and research, you can't build something if you don't know what it's going to look like. That's why we design what we're going to build before we build it, even though we may iteratively refine the design during construction.

If you've ever bought a house before it's been built completely, you know there's a point during the construction process where the homebuilder gives you a date after which you can no longer make changes to the options built into the house—"If you want hardwood floors in the kitchen, speak now or forever hold your peace." I think you get my point. I could continue spelling out these fundamental principles common to all construction projects, be they hardware or software, but then I'd never get to speak about Sherman tanks.

The belief that software construction is fundamentally different from other crafts is rooted in the misguided assumption that the *soft* is inherently different from the *hard*. However, a bridge is as much an artifact of

Go Online

Visit www.javapro.com for related resources. Simply type the Locator+ code into the field in the upper-right corner of the page.

Download

JP0509 Download all the code for this issue.

Discuss

JPTALK Discuss this article in the "Talk to the Editors of *Java Pro* Magazine" discussion forum.

Read More

JP0509PS_T Read this article online.

JP0501NC_T Read the related article "Why Coding Standards?" by Nigel Cheshire.

JP0403CL_T Read the related article "What UML Is and Isn't" by Craig Larman.

JP0305PS_T Read the related article "Write Tests to Refine Your Code" by Daniel F. Savarese.

the imagination as a Web service before it is constructed. The misconception that hardware projects rarely have bugs added to the belief that software is special.

The truth is that bridges collapse, airplanes need constant maintenance, and houses are full of imperfections. The Hubble Space Telescope needed eyeglasses (perhaps monocle is the more accurate metaphor). Hardware has bugs. Just watch a house as it is being built, or try your hand at making some home improvements. You'll be shocked by the primitive materials and methods employed, never mind how all of the bugs are patched over to hide them from view. It may well cross your mind that you'd never write code like that. It is only by applying an inconsistent standard for classifying hardware defects and software defects that you produce the result that hardware rarely has defects.

> A secret about computer programming that many continue to deny is that it is just like everything else

Build a Sherman Tank

My thesis—one that will surely elicit much disapproval—declares that perfection is an inappropriate goal for most software development projects. Certainly, there are classes of projects that demand much more exacting standards than I propose. For example, any software upon which lives depend must strive to avoid causing harm in the case of a malfunction. Large projects require different project management techniques than do small projects. Still, the majority of software and its operators can tolerate imperfection. We manage to live in imperfect houses and drive to work in imperfect cars on imperfect roads (terminally defective by virtue of requiring constant repair, I'll add). We've been doing just fine (barring the occasional rocket explosion or errant space probe—examples of projects that require more exacting standards) using imperfect software for decades.

Many commercially successful software products have been hatched from some very scary source code. A sad reality I've come to terms with over the years is that sound designs and maintainable implementations are not as important as producing what customers want when they want it. Ideally, we can strike a balance between these goals, but in the end, customers don't care how pretty your code is or how acclaimed the methodology you use is. Customers care only if the software does what they need when they need it. You can refactor your code until the cows come home, but that just means you'll never ship the software.

I believe that in the swath of the software industry focused on business and consumer applications, we ought to build Sherman tanks. During the Second World War, German Tiger and Panther tanks had stronger armor and more firepower than American Shermans, but were difficult to produce and maintain. They had complicated parts and, when they broke down, often could not be fixed on the front lines. In contrast, the Sherman had thin armor, little firepower, and a high profile (making it an easier target to hit). It took about four Shermans working together to take down a Tiger tank. However, when they broke, they were easy to fix. They used fewer components than German tanks, and the components were easy to obtain and reuse.

The Sherman could be adapted easily to different tasks, and scores of variants of the tanks were produced—often modified on the front lines (Sherman would be a fine name for a scripting language)—to clear hedgerows, clear minefields, act as bulldozers, fire flamethrowers, travel amphibiously, and lay bridges. Sherman tank software development aims to produce resilient, reusable, adaptable, and easy-to-maintain software. The software isn't perfect, but it's easy to fix when it breaks and to extend to meet unanticipated needs.

The first step toward achieving these seemingly unattainable goals is to accept imperfection. Imperfect software makes it to market and gets used widely (Java is a case in point). Perfect software—or, more accurately, almost perfect software—rarely sees the light of day or ever gets used (does anyone remember Taligent?). It's not that (almost) perfect software isn't developed. It simply doesn't ship before less-perfect software captures the market.

When to Say "When"

Accepting imperfection is all about knowing when to stop. You have to know when good enough is good enough and ship the product. Software development process is important, but it is a means to an end and not the end itself. When necessary, you have to be able to compromise on design, features, or defects to produce a release. You can balance the desire for high-quality software and the reality of imperfect software by designing your software to be a Sherman tank.

Generic programming, component reuse, and extensibility all play into the process. Generic programming allows you to reuse code across multiple parts of a project. Component reuse saves time, especially when you acquire the components from a third party. Extensibility allows you to plug in new functionality not anticipated by the original design when requirements change. The plug-in may be ugly, but at least you don't have to rearchitect a monolithic design.

All of the concepts I've mentioned are decades-old practices familiar to most programmers. The difficulty in producing software according to schedule that works is the same as the difficulty in achieving desired results in any team-based endeavor. You have to be organized, disciplined, and methodical. Programming is work. It can be boring. So can design and every other component of the software development life cycle. Overcoming the inertia of doing boring work depends on the professionalism of your team and the leadership ability of its managers. Knowing how to get work done right isn't enough. You actually have to do it. The Sherman tank didn't cross the Rhine without tank crews willing to get the job done. JP

Daniel F. Savarese is an independent software developer and technology advisor. He has been the founder of ORO Inc., a senior scientist at Caltech's Center for Advanced Computing Research, and vice president of software development at WebOS. Daniel is the original author of the Jakarta ORO text processing packages and the Jakarta Commons Net network protocol library. He is also a coauthor of *How to Build a Beowulf* (MIT Press, 1999). Contact Daniel at **www.savarese.org/contact.html**.

Sybase® WorkSpace: Bridging the Gap Between SOA and Traditional Tooling

Sybase WorkSpace, a unified application development environment, is the first designed to bridge the gap between the vision of a service-oriented architecture (SOA) and the reality of what traditional development tools can deliver. Sybase WorkSpace combines modeling, data management, services assembly and orchestration, Java™ development and mobilization in a single tool. Designed to support the principles of service-oriented development of applications (SODA), Sybase WorkSpace enables developers to quickly build and deliver many kinds of applications: from event- and data-driven applications to web-based, composite, and mobile applications. Sybase WorkSpace is built upon the Eclipse open source framework, making it easier and faster for developers to build complex applications that leverage heterogeneous infrastructures.

Get Sybase WorkSpace and start building tomorrow's applications today.

For more information and to download white papers, visit www.sybase.com/workspace

Copyright ©2005 Sybase, Inc. All rights reserved. All product and company names are trademarks of their respective owners.

SYBASE®

Innovations by InterSystems

Real-time data analytics with a real-fast database.

Imagine being able to query a lightning-fast operational database in real time.

Now you can, with our multidimensional database for transaction processing and real-time analytics.

Only Caché combines robust objects and robust SQL, thus eliminating object-relational mapping. It requires little administration, delivers speed and scalability on minimal hardware, and comes with a rapid application development environment.

These innovations mean faster time-to-market, lower cost of operations, and higher application performance. We back these claims with this money-back guarantee: *Buy Caché for new application development, and for up to one year you can return your license for a full refund if you are unhappy for any reason.**

Innovative database. Guaranteed performance.

InterSystems CACHÉ™

Rapid integration platform makes applications perform together.

Imagine being able to get your applications to perform together as an ensemble. Easily.

Now you can, with our universal integration platform.

Ensemble is the first fusion of an integration server, data server, application server, and portal development software – in a single, seamless product. This is the complete ensemble of technologies needed for rapid integration, fast development, and easy management.

These innovations mean all of your integration projects will be completed on time and on budget, with a simplified learning curve for your IT staff. We back these claims with this money-back guarantee: *For up to one year after you purchase Ensemble, if you are unhappy for any reason, we'll refund 100% of your license fee.**

Innovative integration. Guaranteed performance.

InterSystems ENSEMBLE™

For a free copy of CACHÉ, or to request a free ENSEMBLE proof-of-concept project, visit **www.InterSystems.com/Free8Q**

*Read about our money-back guarantees at the web page shown above.

© 2005 InterSystems Corporation. All rights reserved. InterSystems Caché and InterSystems Ensemble are trademarks of InterSystems Corporation. 5-05 ComboInno8JaPr