A BZ Media Publication

ECLIPSE review

Volume 1 | Number 1
Winter 2006

www.eclipsereview.com

Create Web Apps Faster With WTP

8 Can't Miss Eclipse Tips & Tricks

Better Coding With Eclipse Modeling Framework

Unit Testing: More Than A Fantasy

Sybase® WorkSpace =

Model-Driven Design + Services-Oriented Development + Multi-Channel Deployment on Eclipse

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

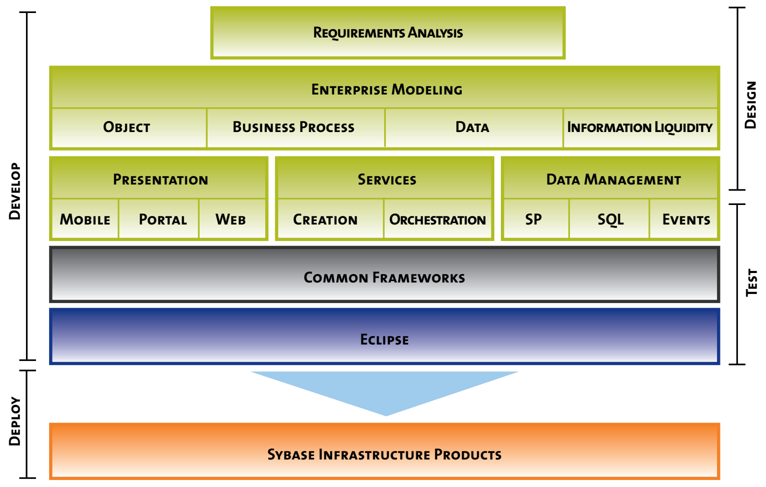
Model. Design. Develop. Deliver. All in one environment. Sybase WorkSpace.

DATABASE DEVELOPMENT:
- Visually create, edit and debug database objects
- Explore data servers throughout the enterprise
- Expose database procedures as services quickly and easily

ENTERPRISE MODELING:
- Manage and analyze requirements and changes
- Reverse engineer models from existing assets
- Model database schemas, EJBs, services and processes

SERVICE-ORIENTED DEVELOPMENT:
- Discover components throughout the enterprise
- Create services from Java components, rules, automated process and messages
- Visually assemble composite applications

MOBILE DEVELOPMENT:
- Instantly mobilize database procedures, applications and services
- Manage page flow and applications
- Support offline and online access



KEY BENEFITS:

Reduces the complexity of combining existing and services-oriented applications. Designed to support the principles of SODA, which simplifies the challenge of turning silos of data and business logic into responsive flows of information.

Improves productivity with mixed-mode development in a single tool. Supports a real-world mix of development styles including Rapid Application Development (RAD), Architected Rapid Application Development (ARAD), and Model-Driven Development (MDD).

Improves business agility through faster, more flexible development. Enables developers to focus on higher-level business process flows that leverage existing data and components in responding more quickly to business requirements.

Reduces errors and time spent on mundane tasks. Combines powerful productivity features present in graphical development styles while automating more of the development process.

Reduces the cost of software maintenance through better design discipline. Model-driven development ensures consistency, fewer one-off projects, and better alignment with business requirements.

Quickly mobilizes information; extends Java to mobile development. Enables codeless mobilization of all kinds of enterprise applications and services, allowing use of Java and J2EE as the programming platform as well as occasionally connected service invocation.

SYBASE®

www.sybase.com/workspace

CodePro AnalytiX™ is developed by the experts who brought you the popular book *Eclipse: Building Commercial Quality Plugins* — Eric Clayberg & Dan Rubel

tools that make every developer a quality expert

CodePro AnalytiX™

for Eclipse, Rational® and WebSphere®

Download a risk-free trial copy:
www.instantiations.com/codepro

Drive out quality problems earlier in the development process.
Powerful code audits and metrics make every developer a Java expert.

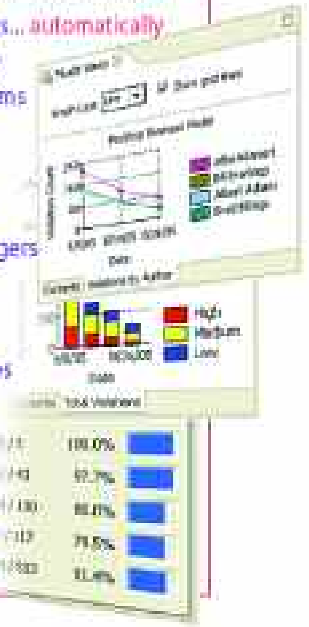
Create tight, maintainable systems.
Wizards generate unit and regression tests, and analyze testing code coverage.

Enforce quality measures across teams.
Define, distribute, and enforce corporate coding standards and quality measures, across individuals and teams… no matter where they're located.

Spend less time and money to develop high-performance Java systems.
Dramatically improve your application development process and code quality… leading to more reliable, maintainable systems.

Key Features of CodePro AnalytiX™

- Detect & correct code quality issues… automatically
- Define, distribute & enforce quality standards across development teams
- 800+ audit rules and metrics, 350+ Quick Fixes
- Powerful management reporting
- Code metrics with drilldown & triggers
- Audit Java, JSP and XML files
- JUnit test case generation
- Code coverage analysis of test cases
- Dependency analysis & reporting
- Integrated team collaboration
- Javadoc analysis & repair
- Seamless integration with Eclipse, Rational® and WebSphere® Studio



instantiations
the development tools experts

www.instantiations.com 1-800-808-3737

No one outside of IBM has more experience creating Eclipse-based Java development tools

ECLIPSE review

V1 | N1 | WINTER 2006

Cover Illustration: *"Touching Cyberspace," by Mark Frost*

Editorial Director
Alan Zeichick
alan@bzmedia.com
+1-650-359-4763

Executive Editor
George Walsh
+1-415-731-3843
gwalsh@bzmedia.com

Art Director
LuAnn T. Palazzo
lpalazzo@bzmedia.com

Publisher
Ted Bahr
+1-631-421-4158 x101
ted@bzmedia.com

Associate Publisher
Charlie Shively
+1-508-893-0736
cshively@bzmedia.com

Northeast Sales Manager
David Lyman
+1-978-465-2351
dlyman@bzmedia.com

Southeast Sales Manager
Jonathan Sawyer
+1-603-924-4489
jsawyer@bzmedia.com

Northwest Sales Manager
Paula Miller
+1-925-831-3803
pmiller@bzmedia.com

Southwest Sales Manager
Robin Nakamura
+1-408-445-8154
rnakamura@bzmedia.com

Ad Traffic Manager
Phyllis Oakes
+1-631-421-4158 x115
poakes@bzmedia.com

Director of Circulation
Agnes Vanek
+1-631-421-4158 x111
avanek@bzmedia.com

Director of Events
Donna Esposito
+1-415-785-3419
desposito@bzmedia.com

Director of Editorial Operations
David Rubinstein
drubinstein@bzmedia.com

Office Manager/Marketing
Cathy Zimmermann
czimmermann@bzmedia.com

Article Reprints
Lisa Abelson & Co.
+1-516-379-7097
labelson@optonline.net

Customer Service/Subscriptions
service@bzmedia.com

BZ Media is an Associate Member of the Eclipse Foundation, www.eclipse.org

FEATURES

"Eclipse makes the programming experience enjoyable without being cumbersome."

—Isaac Sacolick, COO, TripConnect

DEPARTMENTS

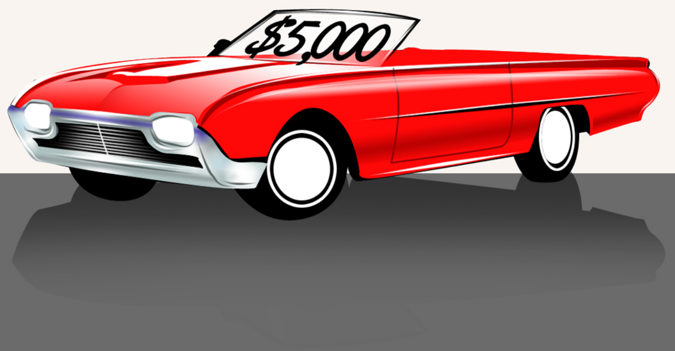
BZ Media

BZ Media LLC | 7 High Street, Suite 407 | Huntington, NY 11743 | +1-631-421-4158 | fax +1-631-421-4045 | www.bzmedia.com | info@bzmedia.com
President | Ted Bahr **Executive Vice President** | Alan Zeichick

Eclipse Review is published 4 times a year by BZ Media LLC, 7 High Street, Suite 407, Huntington, NY 11743. Periodicals privileges pending at Huntington, NY and additional offices. POSTMASTER: Send address changes to BZ Media, 7 High Street, Suite 407, Huntington, NY 11743. Ride along is included.

MYECLIPSE ENTERPRISE WORKBENCH

WHY PAY MORE?



LEGACY IDEs
PAY MORE

ERD MODELING	EXTRA
UML DEVELOPMENT	EXTRA
OBJECT RELATIONSHIP MAPPING	PROPRIETARY
SPRING AND TAPESTRY INTEGRATION	N/A
HIBERNATE MAPPING EDITOR	N/A
RICH CLIENT IMAGE EDITOR	N/A
ECLIPSE 3.1 SUPPORT	N/A
JAVA 5 SUPPORT	N/A
AJAX SUPPORT	N/A
WEB 2.0 SUPPORT	N/A
APPLICATION SERVER INTEGRATION	PROPRIETARY
MAINTENANCE AND SUPPORT	EXTRA
TOTAL	$3,000 – $10,000

MYECLIPSE
EXPECT MORE

STANDARD SUBSCRIPTION
AJAX SUPPORT - FULL FEATURED JAVASCRIPT EDITOR
VISUAL WEB DESIGNER (HTML/JSP/STRUTS & JSF TAGS)
VISUAL STRUTS DESIGNER
VISUAL JAVA SERVER FACES DESIGNER
FULL FEATURED SOURCE EDITORS
HTML/JSP/XML/XSL/XSD/SQL/CSS/JAVASCRIPT
DATABASE EXPLORE - 20 DATABASE CONNECTORS
ERD MODELING
SPRING AND TAPESTRY INTEGRATION
HIBERNATE MAPPING EDITOR
RICH CLIENT IMAGE EDITOR
COMPREHENSIVE APPLICATION SERVER INTEGRATION
ECLIPSE 3.1 SUPPORT
JAVA 5 SUPPORT
INCLUDED WORLD CLASS MAINTENANCE AND SUPPORT

TOTAL $30

PROFESSIONAL SUBSCRIPTION

ALL STANDARD FEATURES +++
UML MODELING AND ROUND TRIP ENGINEERING
AJAX SUPPORT - JAVASCRIPT DEBUGGER
SPECIALIZED ORACLE DATABASE DEVELOPMENT

TOTAL $50

Powered By
genuitec
The Eclipse Authority

Contact:
(888) 267-4176
www.myeclipseide.com
Download Free Trial

Welcome to the First Issue of Eclipse Review

The tools and techniques of software development have come a long way since I wrote source code using XEDIT on an IBM S/370 mainframe in the late 1970s. If your lineage as a programmer goes back a few decades or longer, you remember the era of discrete tools: you used an editor to create source files, and then ran them through a compiler and linker. You studied the log files or viewed command-line warnings.

After the code compiled, you tested it—not with a test tool, but by running it and seeing where it broke.

If you used tools beyond the text editor, compiler and linker, they were simple ones, like lint, a Unix tool that checked for syntax errors, or make, which automated the build process.

The universe began changing with the introduction of the first integrated development environment for the general developer audience. Borland's Turbo Pascal inspired tool makers, initially for DOS and Windows, but later across the board. We can see the heirs of Turbo Pascal today, in Borland's Delphi and JBuilder, Microsoft's Visual Studio, Apple's Xcode, Sun's NetBeans—and of course, Eclipse.

What makes Eclipse special, what makes it unique, isn't just the excellent and solid technology. Eclipse succeeds because it's more than bits: It's a vibrant community of strategic developers, plug-in providers and committers who are devoting time and resources to support and enhance the ecosystem.

Unlike many other open-source collaborations, Eclipse is driven by for-profit businesses, for the most part, who see a vital need for this software, not just for themselves, but for their customers. These companies are motivated to work together, in a vendor-neutral environment, creating a toolchain that they can use internally as well as leverage in their own product offerings.

Enlightened self-interest is a beautiful thing. It ensures that Eclipse will continue to advance and evolve, not only improving its core functionality, but also expanding into new areas. It also means that integration issues, which so often plague developer toolchains, will be addressed, because the companies building the ecosystem simply can't afford to have problems in that area.

That's why we've seen good results in bringing together such a huge array of Eclipse top-level projects, from BIRT and RPC, to newer efforts like the Data Tools Platform and AJAX Technology Framework.

Bottom line: We all win.

THANKS, MIKE AND IAN!

Eclipse Review is brought to you by BZ Media, which you may know as the publisher of SD Times (www.sdtimes.com) and Software Test & Performance Magazine (www.stpmag.com).

Our business, and our passion, is serving the developer community, not only with our publications, newsletters and Web sites, but also through technical conferences like the Software Test & Performance Conference (www.stpcon.com), Software Security Summit (www.s-3con.com) and EclipseWorld 2006 (www.eclipseworld.net), coming up Sept. 6-8 in Cambridge, Mass., right across the river from Boston. I hope to see you there; we're assembling a great technical program for this year's event.

Eclipse Review will be published quarterly this year, both in a printed magazine and as a downloadable digital edition (in PDF format) on www.eclipsereview.com.

At that same Web site, wou can sign up for a free subscription, either to the print edition (U.S. only) or to have the digital edition e-mailed to you (U.S. and worldwide).

I would like to thank the Eclipse Foundation for supporting the launch of Eclipse Review. Mike Milinkovich, the executive director, and Ian Skerrett, the director of marketing, have offered incredibly valuable advice and guidance. It's always a pleasure working with them, and with the members of the Eclipse Foundation, as we continue to grow Eclipse Review and EclipseWorld to better serve the needs of the community.

Finally: Eclipse Review is for you. Our goal is to help you develop better software using Eclipse-based tools and technologies. Tell us how you use Eclipse, the benefits you've seen, the improvements you've made, the challenges you face. Tell us how we can help you be more successful. Write me at alan@bzmedia.com.

—*Alan Zeichick, Editorial Director*

Save the Date!

Boston, MA
September 6-8

Attend EclipseWorld 2006,
the enterprise development conference!

EclipseWorld is for enterprise developers, architects and development managers who want to take their company's applications to a higher level!

For sponsorship or exhibiting information, contact Donna Esposito at 415-785-3419 or desposito@bzmedia.com

eclipse
WORLD
The Enterprise Development Conference

Cambridge Hyatt Regency

Boston, MA
September 6-8

EclipseWorld is brought to you by:

BZ Media SD Times Software Test & Performance Eclipse

BZ Media is a member of the Eclipse Foundation.

www.eclipseworld.net

MyEclipse Takes On AJAX, Adds Web Tools

Genuitec's tool suite for enterprise got smarter with its 4.1.1 release in early March. This maintenance upgrade of MyEclipse Enterprise Workbench builds on the late January introduction of version 4.1's new Web 2.0 Workbench for Asynchronous JavaScript and XML (AJAX), improved modeling using the UML Sequence Diagram, improved visual Web development, a new image editor and Spring/Hibernate integration. The maintenance release adds more new features, including source viewing capability for the Web 2.0 Browser; a new JavaScript Console view added to Web 2.0 Workbench and new customizations available in the Hibernate Reverse Engineering Mapping wizard. MyEclipse Enterprise Workbench, which is delivered as an Eclipse 3.1 plug-in, costs US$31.75 per year for the Standard Edition, and $52.95 for the Professional Edition (which adds UML modeling and an Oracle database connector).

URL www.myeclipseide.com

IBM Multimodal Tools For XHTML + Voice Applications

X + V, or XHTML + Voice, is a new XML-based markup language, submitted to the W3C, that is being used to build so-called multimodal applications—that is, those which have both visual and speech/voice user interfaces. These multimodal apps allow for user interaction from more than just a Web browser; they can also be used with devices such as smartphones and allow voice recognition over a standard wireline phone. If you're working with those technologies, download Multimodal Tools Project from IBM's alphaWorks Web site. MTP is a set of plug-ins that works with Eclipse 3.1, Web Tools Platform and the Eclipse Voice Tools Project. The plug-ins include an X + V editor and a multimodal browser launching configuration. The editor supports color highlighting, content assistance and content validation for the new language, and the application grammar can be created using a Voice Tool Project's wizard. Once a developer has finished developing the application, the multimodal browser can be launched directly within the tool. The plug-ins also include the XHTML + Voice Programmer's Guide, which contains examples of the X + V elements.

URL alphaworks.ibm.com/tech/mmtp

Asynchronous JavaScript and XML Through Exadel

Exadel has updated Exadel Studio, its Eclipse-based enterprise Java development suite, to support AJAX. Exadel Studio 3.5, which costs US$199, adds what the company describes as extensive support for the AJAX JavaServer Faces components in the Apache MyFaces Sandbox component library. The AJAX components are accessible through the Exadel Palette used with the Exadel Visual Page Editor. Dragging and dropping an AJAX user interface component onto a page using the Visual Page Editor launches a wizard for setting up the component for use in the Web page. Based on the Web Tools Project 1.0, Exadel Studio supports StrutsShale, which is a new JSF Web application framework, and the upgraded Facelets 1.0 extension to JSF.

URL www.exadel.com

Sorcerer from DHI Takes the Mystery Out of JavaScript

JS-Sorcerer 2006 is an intelligent JavaScript plug-in for Eclipse that helps build correct, interactive, and cross-browser compatible Web sites and applications. The plug-in performs syntax checking and type and flow analysis on standalone JavaScript files, and provides type-safe linking for applications and projects that consist of multiple JavaScript files. It detects errors at compile time, eliminating the need to invoke a browser simply to catch syntax and typographical errors. JS-Sorcerer lets developers write cross-browser code, according to the company, by making use of a standard interface to ECMAScript, W3C DOM, and the XMLHttpRequest (AJAX) object. As an Eclipse 3.1 plug-in, the US$199 JS-Sorcerer supports MyEclipse and the Eclipse Web Tools Platform, and according to the company, can also be used to help build AJAX and other rich Internet applications, in addition to enhancing standard JavaScript.

URL www.dhitechnologies.com

Canoo Paddles Updated UltraLightClient Visual Editor

The Swiss firm Canoo Engineering has updated the graphical editor used for building applications for its UltraLightClient, a Java library for rich Internet application development. The ULC client is based on Swing, and is a pure Java implementation that doesn't use AJAX, JavaScript or other languages. New to version 5.0 of the Visual Editor is improved startup performance when opening a visual class, a new class wizard, and improved copy & paste functionality. It also supports ULCFiller, a new class introduced with version 6 of the ULC client that explicitly fills in extra space in the user interface. A developer license using the ULC Visual Editor costs US$499; the editor requires the ULC client, available separately. Developer licenses for the ULC client cost $1,499 per seat.

URL www.canoo.com

IBM RATIONAL PRESENTS

YOU ★VS★ THE INCREDIBLE SHRINKING DEADLINE

MAIN ATTRACTIONS

KNOCKOUT INNOVATION

INTEGRATED DEVELOPMENT TOOLS SUPPORTING ASSET-BASED DEVELOPMENT ★ BASED ON ECLIPSE™ ★ RUNS ACROSS MULTIPLE PLATFORMS INCLUDING LINUX®

POWER TO CREATE BETTER SOFTWARE FASTER

IBM MIDDLEWARE. POWERFUL. PROVEN. FIGHT BACK AT WWW.IBM.COM/MIDDLEWARE/TOOLS

AND DOWNLOAD TRIAL VERSIONS OF RATIONAL SOFTWARE MODELER & RATIONAL SOFTWARE ARCHITECT

TripConnect Links Network Of Travelers Via Eclipse

In the not-so-distant past, the modus operandus of financially solvent development shops was to buy a huge IDE that could do everything they'd ever want it to (and a million things they didn't), pay for a lengthy support contract, and get hacking. Smaller shops without those resources had an extremely difficult time competing. They simply couldn't afford the tools.

Nowadays, many development teams—large and small—are seeing the error of IDE overkill and financial waste, seeking out technology that's appropriate for their needs. This not only lowers the bottom line for the big players, but reduces the cost of entry into the competitive world of software development. In any field, competition leads to innovation and better products. TripConnect (www.tripconnect.com) is an example of both an entrepreneurial spirit and the ability to create and maintain a software product while keeping costs to a minimum. As the company discovered, few software tools go further in keeping costs to a minimum than an open-source toolchain based on the Eclipse IDE.

CONNECTING PEOPLE, PROFITS

The idea behind TripConnect, launched in November 2005, is to help people get advice from a network of other travelers. The premise is that people value advice from acquaintances very highly, but they have no way of knowing where these people have traveled.

The New York City-based company provides a personal trip-planning experience to pleasure travelers by helping them find information based on where they want to go, what they want to do, and when they want to do it.

What differentiates the site is that it isn't limited to individual, unsolicited online reviews. On the TripConnect site, users sign up, create a profile, indicate places they've been and link to others by invitation. The idea is to help users create a travel "network." This network can be a group of friends, people interested in the same type of activity, or those who like to frequent a certain area of the world. In addition to seeing reviews of places that have been visited by those in a given network, users can communicate with one another to ask specific questions or visit the profile pages of users whose tastes run parallel to their own.

TripConnect's primary development team. From left to right: William Bagby and Isaac Sacolick.

For many companies, the idea that the Internet could enable an effective business model turned dollars into quarters not so long ago. However, a few markets have survived and flourished online and one of them is travel.

"Our business is based on helping people find travel information and helping travel suppliers serve their needs," says Isaac Sacolick, TripConnect's COO. "Travel is one of the more favorable advertising categories on the Internet primarily because the booking sites are all trying to differentiate themselves. As we build up traffic, our site will be more appealing to travel suppli-

For many companies, the idea that the Internet could enable an effective business model turned dollars into quarters. However, the Web travel market has survived.

"Everything I do needs to be set up so that somebody else can easily take it over when I move further into handling business tasks," Sacolick says.

ers who want to reach specific customer segments."

THREE FOR THE ROAD

TripConnect's human resources include Sacolick, CEO Carter Nicholas and a full-time programmer named William Bagby, Sacolick's respected coworker from another company. As with most small enterprises, job titles can be deceiving. Sacolick is in charge of operations, including everything from marketing to development. Right now, he estimates that 80 to 90 percent of his time is spent developing software.

While some of the site's initial development was outsourced, its functional specifications were written by Nicholas and Sacolick.

Sacolick wrote the technology specs, such as the database schema, and worked with an outside development team to complete the initial coding.

TripConnect continues to work with outside software development contractors and designers to supplement its own small programming team.

Sacolick know that he'll have to move out of development as the company grows. "Everything I do has to be set up so that somebody else can easily take it over when I move further into handling business tasks," he says.

Preparing for that stage of the company's success requires the team to have appropriate development procedures and practices in place. At the same time, limited financial and human resources tend to keep TripConnect in the realm of open source, with Eclipse as a key piece of the puzzle.

DISCOUNT DEVELOPMENT

The TripConnect site is based on Tomcat, Struts and MySQL. It's hosted on Linux servers and makes use of a number of FireFox plug-ins. At the heart of the development effort is the Eclipse IDE.

"I don't think we're doing anything extraordinary but I think we're a good example of how using a disciplined software development process and leveraging common tools and practices within Eclipse can be successful," Sacolick says.

The first step in building an application is deciding which programming language to use. Here, the TripConnect team looked at Perl, PHP, and Java primarily based on its experience with the languages. In Sacolick's experience, much can be accomplished using Perl and PHP very quickly and efficiently. Unfortunately, the code developed using those languages can become very hard to work with during a two or three-year life cycle due to its structure.

"Thinking down the road, Java was the only real answer for us. Once you choose Java, you need to decide whether you want to use a simple editor, a commercial IDE, NetBeans or Eclipse. Eclipse and NetBeans are free, which put them at the top of our list," Sacolick admits. But saving money wasn't the only reason: he had hands-on experience with Eclipse, which he used in some of his previous jobs. "It was free and we knew it worked."

Figure 1 | TripConnect reaches out to travelers and their community

TripConnect found a number of other features that complimented the Eclipse environment's price and their experience. These tools included code completion and the ability to find declarations of functions and function hierarchies.

At a higher level, TripConnect uses CVS integration as a dashboard for development. Eclipse allows them to flip between CVS and Java perspectives in addition to making use of Ant views. The company also uses a plug-in that provides access to the Apache Velocity template engine.

"For a small development shop like ours, large IDEs are too costly to purchase and implement," Sacolick says. "Since there are only two of us working on development, we don't need to go crazy standardizing our environment. We really only stick with two major tools that we consider standards: Eclipse as a development environment and Firefox as a browser. Everything we do is in one of those environments or 100 percent Web based. If we keep it simple, we can spend most of our time coding and testing and not playing around with software toys and tinkering with configurations."

REVIEWING THE TRIP

In many organizations, code reviews are seen as a dreadful process that developers are dragged into by force. The code goes up on the whiteboard and people start poking holes in it. After four or five weeks, everybody feels wiped out.

That's a shame. Code reviews are essential in discovering mistakes, both in simple coding and in the broader application design. It's important to get everyone on the team to buy into the idea that it's an important activity. TripConnect takes this process very seriously, whether it involves its own small development staff or includes contractors. To make it less of a chore, the team has come up with a review process that addresses their needs without becoming too complicated.

"When we check things into CVS, we use change sets and have a nomenclature to log comments in a check-in," Sacolick says. "It informs you of all the change sets and all the code that's changed when you are synchronizing your environment. We're using the DIP editor to actually drill out and look at the differences in the code and comment on it."

Sacolick believes the CVS repository gives his team a first line of defense and QA. If somebody sees a null pointer or an object that is not being checked for a null state, it's much easier to find at code check-in than after the application is live. "Using this method, we manage to find little defects that often happen in small groups that are trying to move fast." In addition, the development team checks to be sure the logging levels are done correctly, whether they make sense, and whether the logged details are too generalize or too specific.

Due to the size of TripConnect's development team, Sacolick and Bagby are in the habit of critiquing each other's work. Bagby's strengths lean more toward web application development, so he'll work with Sacolick's JSP files and offer suggestions on how to use CSS or JavaScript. In contrast, Sacolick tends to have a stronger understanding of database, IT and QA issues. This review process tends to help them educate each other, improving the small team's overall skills.

"When Will started here, he wasn't really good at doing database join queries," Sacolick says. "That's not the case anymore. When he started, I wasn't very strong at doing CSS-style Web pages. I don't think I'm an expert at it but now I know what I'm doing. In a small company where you have to wear many hats as a developer, a flexible tool like Eclipse really helps you get everybody on a level playing field and doing things in a similar manner."

TripConnect also makes use of what it calls a "refactoring barometer," which uses code check-ins to see if the developers are heading in the right direction or toward code that they're going to have to refactor later on. "From my perspective, it's OK to say that we're going to refactor something later as long we make note of it," Sacolick says. "Sometimes the comment is not in the code but we make every effort to make a note in the project list when we see something that needs to be refactored."

Sacolick says that TripConnect's method is intended to make code reviews a positive experience for everyone involved. "We may not use expensive tools, but we keep each team member informed of what the others are doing. We think we've turned it into a positive educational process that avoids pointing fingers and placing blame."

OFF THE GROUND RUNNING

Eclipse offers many benefits to TripConnect. It saves them money and keeps the small development group working as efficiently as possible, and lets the company stay lithe and put more features out faster—vital in a competitive industry like online travel.

> Eclipse offers many benefits to TripConnect. It saves them money and allows them to keep their small development team working as efficiently as possible.

"Eclipse makes the programming experience enjoyable without being cumbersome. It's easy to work with, learn and install, but you don't have to learn everything on Day One," he says.

"When we started with Eclipse, we used it as a project explorer and a schema manager for Java code. It wasn't until we became more experienced with the environment that we discovered we could plug in processes, use Ant with it, tweak our warnings in the Java compiler and synchronize our settings. These are all capabilities that we learned as the need arose. For us, its biggest strength is that you don't have to be an expert with the Eclipse before you start developing with it."

—George Walsh, Executive Editor

The Graphical Editing Framework

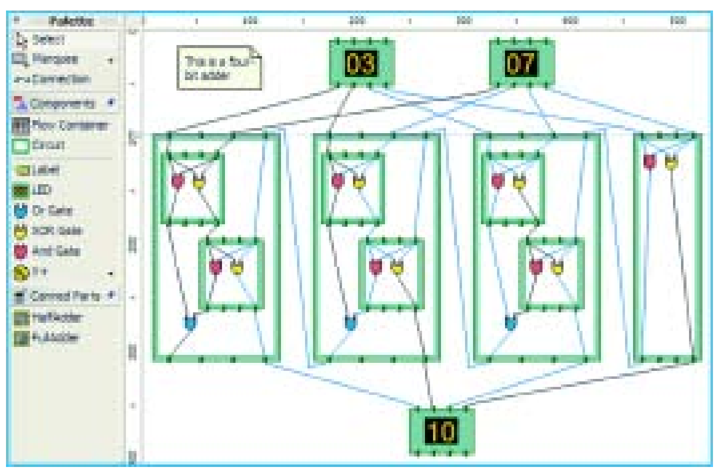
The Graphical Editing Framework is an Eclipse tools project that provides developers with an infrastructure that can be used to create graphical applications using Eclipse. To make use of GEF. you should be familiar with the Eclipse platform, SWT, workbench and related components. Let's explore GEF's dependencies, features and benefits.

START WITH DRAW2D

To address graphical applications, the Graphical Editing Framework provides two toolsets: Draw2d and GEF. Draw2d is a lightweight GUI toolkit built on top of SWT. GEF furnishes the infrastructure that allows for the editing of some models represented by Draw2d widgets (called figures). You use Draw2d when you only need image rendering functionality. Use the bigger GEF toolset when you also need to implement editing.

Draw2d provides rendering and layout support for figures that can be easily customized. Figures can be non-rectangular, nested, and adorned with borders, fonts, colors, and other graphical elements. The Draw2d library is not nearly as extensive or as functionally comprehensive as Swing, but it is safe to say that they are similar. Draw2d borrows some concepts and terminology from Swing, making it easy to learn and understand. The tool's figures are hosted in an SWT canvas, as shown in Figure 1, so the relationship between Draw2d and SWT is similar to the relationship between Swing and AWT, as Figure 2 describes.

Figure 1 | A GEF-based circuit diagram editor



***Pratik Shah** works at IBM on the Graphical Editing Framework. He graduated with a B.Sc. degree in Software Engineering from Rochester Institute of Technology.*

GRAPHICAL FLEXIBILITY

GEF can be thought of as an interaction layer built on top of Draw2d. Its most salient feature is its employment of an MVC-based design. While the view is derived from Draw2d, GEF acts as the controller that provides a model-to-view mapping. In other words, it displays a model graphically

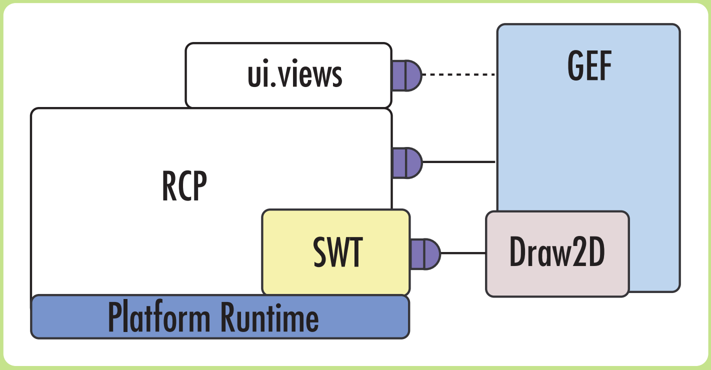
The clean model-view-controller separation has allowed GEF to be model-agnostic. It knows nothing about your model and doesn't place any restrictions on it. It only requires your model to have a notification mechanism (observer pattern). This allows GEF applications to be built with a wide range of disparate models where the controller provides the mediation between the custom domain models and the lightweight Draw2d figures.

GEF's relationship to Draw2d is similar to that of JFace's relationship to SWT. Like JFace, GEF has the concept of a viewer that hosts the construction of a view based on some underlying model. While the analogy isn't perfect (one of the main differences between the two frameworks is that JFace does not allow editing in its viewers, whereas GEF exists mainly for that capability), it's mentioned mainly to convey a proper feel about Draw2d, GEF, and their respective roles (Figure 3).

While GEF is typically used to create activity diagrams, GUI builders, class diagrams, and state machines, it is versatile enough to be used for just about any graphical application. In fact, it can be used anywhere that you can have an SWT canvas, so it's not unusual to see GEF being used in views, dialogs, and other applications. GEF is also currently being enhanced to support the easy creation of rich text editors.

Draw2d, although created mainly to support GEF, requires only SWT, and like SWT, can be used outside Eclipse. The GEF plug-in, on the other hand, requires the Eclipse Rich Client Platform (RCP) and the org.eclipse.ui.views plug-in, which provides the properties and outline views. In the upcoming 3.2 release of GEF, the dependency on org.eclipse.ui .views will be made optional, consequently making GEF available for use in RCP applications.

Figure 2 | GEF components and dependencies



GEF'S CAPABILITIES

Out of the box, GEF supports creating, moving, resizing, and deleting components, undoing and redoing those actions, and other capabilities helpful for creating graphical applications. It provides an easy-to-populate yet customizable palette from which parts can be dragged and dropped into the editing space.

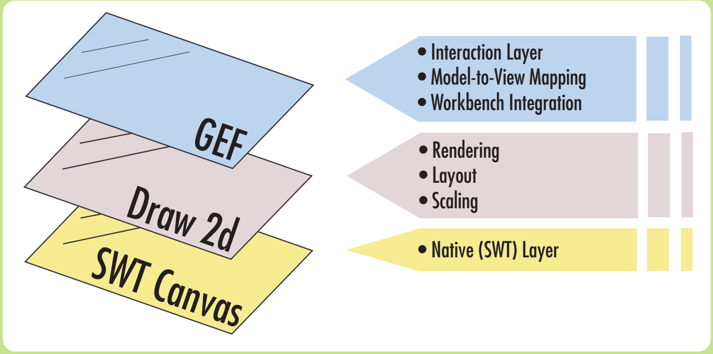
The GEF palette also hosts tools that address tasks that include selection, panning, marquee or group selection, and connection creation. GEF provides for a number of different types of annotations to support a variety of graphical applications. Connections can be annotated with labels, tooltips, and arrowheads, and customized just like any other figure. It can also automatically route connections around components, allow the user to manually specify routing locations (called bendpoints), or some combination thereof.

One lesser known features of GEF is its support for accessibility. In addition to letting users perform graphical interactions via the keyboard, screen readers can detect and describe graphical components (accessibility is dependent on operating system support). GEF also has an overview for quick navigation in large diagrams; zooming; grids; rulers and guides similar to those found in word processors; snapping to grid, guides or other components in the diagram; part cloning; automatic component layout based on a graphing algorithm; and automatic viewer scrolling when dragging a part close to an edge.

GEF makes the job of integrating with the Eclipse workbench easier as well. Its viewers can publish changes to the workbench's selection service, so that views can be appropriately populated. Performing actions via the tool bar, menu bar, and context menu is a relatively simple operation, and GEF provides some generic actions as well.

The framework presents some common utilities for functionality, external to the editor, that most GEF-based applications are likely to support in the Eclipse environment. Examples include a tree viewer for the outline view, undo and redo support for the properties view, and other tools. Future plans include leveraging the new operation history framework provided by the Eclipse platform, providing further integration with the workbench and other tools, such as the Eclipse Modeling Framework (see "A Grand Tour of the Eclipse Modeling Framework," p. 37).

Figure 3 | Overview of GEF components and roles



WHAT YOU SEE IS WHAT YOU GET

Draw2d and GEF allow Eclipse users to quickly get a graphical application up and running. With a simple and extensible toolkit, Draw2d provides a lot of flexibility. GEF adds the interactive element to Draw2d and supplies the hooks to integrate with the Eclipse Workbench. The fact that GEF successfully caters to a large community with diverse needs speaks volumes about its versatility. An elegant architecture and a robust and feature-rich implementation have made it the de facto tool when it comes to graphical editing within Eclipse.

Better Software The Eclipse Way: Using The TPTP Test Tools

"Let's see a show of hands. How many of you believe that software testing is a waste of time?"

No one raises their hand.

"How many of you would like me to review the benefits of thorough software testing?"

Once again, no one raises their hand.

"How many of you believe that the testing done in your organization is sufficiently thorough?"

A few people raise their hands, but others just laugh out loud.

BY BARRY BURD

I've decided not to waste your time enumerating the benefits of good software testing. No one in his or her right mind would say that testing is unnecessary. Besides, if you didn't understand the value of software testing, you probably wouldn't be reading this article.

The question isn't "Why test software?" The question is "How do we test software?" In recent years, unit testing has become the rage. Unit testing means testing each class (each "unit") while the software is being coded. Best practice dictates that you write code to test a class before you write the code to implement the class. (That way, the test code isn't biased toward what you already know the class is capable of doing.) Many software development shops set up their CVS repositories to test each unit nightly. So, testing is an ongoing part of the software development cycle, and not an afterthought when a project's code is cast in stone (well, cast in bits, photons or whatever).

All unit tests have certain characteristics in common with one another, so it makes sense to bake those characteristics into an API. A good unit testing API facilitates the creation and running of unit tests. Ultimately, the tested software is more reliable and more robust.

JUNIT TESTING IN THE ECLIPSE CORE

For Java, the uncontested champion among unit testing APIs is JUnit. Created in the late 1990s by Kent Beck and Erich Gamma, this unassuming API defines tests, test suites, assertions about results, test runners and many other things a unit test needs to have.

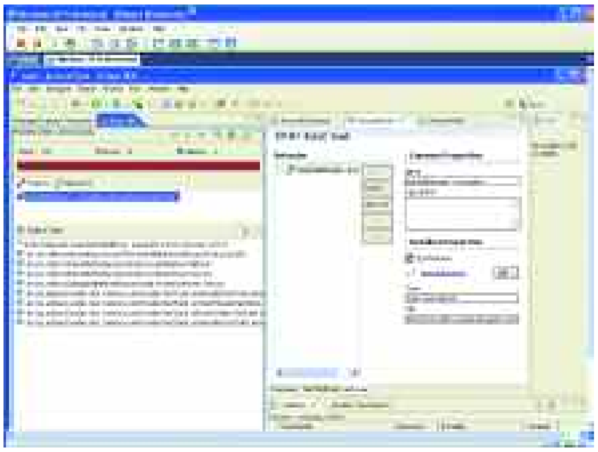
If you're an Eclipse user, and you're not already familiar with JUnit, there's an easy way to get started. Let's assume that you start with a class named Account. (The class stores a customer's name, balance and other such information.)

Start by right-clicking the Account class in the package explorer and then (after a bit more poking around) click to create a new JUnit Test Case. (You can find details in the "Writing and running JUnit tests" section of Eclipse's own documentation pages.)

When you do this, Eclipse creates a skeletal `AccountTest.java` source file, with methods to set up the test, run tests on individual methods inside the Account class and tear down all the testing resources after a run of the test is completed. The only coding you have to do is to instantiate the Account class and add a few calls to JUnit's assert methods (calls like `assertEquals (account.getBalance(), 10.40, 0.01)`). Then, when you select Run As... JUnit Test, Eclipse runs

__Barry Burd__ is a professor in the Department of Mathematics and Computer Science at Drew University in Madison, NJ. When he's not lecturing at Drew University, Dr. Burd leads training courses for professional programmers in business and industry. He has lectured at conferences in America, Europe, Australia and Asia. He is the author of several articles and books, including Java 2 For Dummies and Eclipse For Dummies, both published by John Wiley & Sons.

Figure 1 | The JUnit view in core Eclipse



the new `AccountTest` class and shows the test results in the JUnit view, as shown in Figure 1.

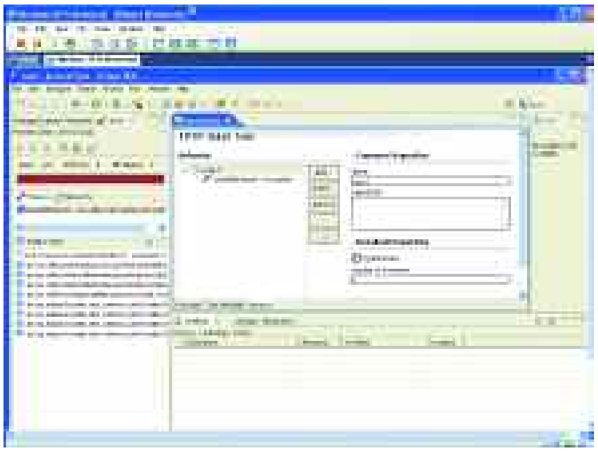
THE TEST & PERFORMANCE TOOLS PLATFORM

The previous paragraph summarized the support for unit testing in Eclipse's core. The support is useful, but it's also primitive. There's no specialized support for editing test code, running tests, or for reporting and analyzing test results. For tasks like these, you need something beyond Eclipse's core. And of course, I have something in mind. You need the Eclipse Foundation's Test & Performance Tools Platform (TPTP). This TPTP project is a reincarnation of the Eclipse Hyades project. TPTP is an ongoing project with four subprojects:

- TPTP Platform
- Monitoring Tools
- Tracing and Profiling Tools
- Testing Tools

Expanding the acronym in the first of these four bullets, we get the "Test & Performance Tools Platform *Platform.*" That's not a typo. This bullet refers to the Platform underlying the all-encompassing Test & Performance Tools Platform. The TPTP Platform subproject provides a common infrastructure for the other three subprojects.

Figure 2 | A portion of the TPTP JUnit Test editor



This article is about testing, so we'll skip past the middle two subproject bullets (Monitoring plus Tracing and Profiling) and cut directly to the last of the four items. This fourth subproject (the Testing Tools subproject) has tools for editing and running tests and additional tools for analyzing and reporting test data.

To show you the Testing Tools basics, let's start again with a class named Account. You use a TPTP JUnit Test wizard to create a test based on your existing Account class. The wizard builds a file named `AccountTest.testsuite` and TPTP automatically opens the TPTP JUnit Test editor—an editor for `.testsuite` files. Under the hood, `AccountTest.testsuite` is a Zip archive containing an XML file. This XML file describes the testing that I intend to apply to my Account class.

Figure 2 shows one of the TPTP JUnit Test editor's pages. Using the page's Add and Insert buttons, you define some testing behavior. The behavior in Figure 2 describes a loop containing a test method invocation. (Later, when you run the test, Eclipse calls the `testAddInterest method` five times. By checking the Synchronous box in Figure 2, you insist that the first method call returns before the second begins, the second call returns before the third begins and so on.)

If you accept certain TPTP options, the code in an `AccountTest.java` file stays in sync with your TPTP JUnit Test editor choices. The syncing isn't foolproof, so you have to be careful. For example, you can add certain methods to the `AccountTest.java` file or the TPTP JUnit Test editor. Either way, the `AccountTest.java` and `AccountTest.testsuite` files stay in sync. But if you tinker with the behavior code in the `AccountTest.java` file (the code that says "invoke this test method at this particular time"), then all bets about file synchronization are off.

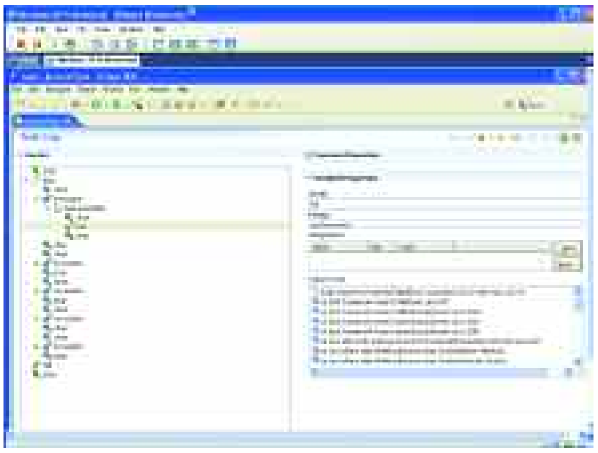
And speaking of the `AccountTest.java` behavior code, this automatically generated code uses reflection. With reflection, TPTP can sink its hooks into your test:

```
HyadesTestSuite loop1 = new HyadesTestSuite(
     "Loop 1");
 accountTest.addTest(new RepeatedTest(loop1, 5));
 loop1.setId("DCEEAB781FB3F62E1C1DC5609D1811DA");

 loop1.addTest(new AccountTest("testAddInterest")
      .setId("E346BEC1401EFED4F6977E509B8511DA")
      .setTestInvocationId(
           "DCEEAB781FB3F62E226262A09D1811DA"));
```

When you run a test using TPTP, you get a log that's more detailed than the JUnit view in core Eclipse. For a look at the TPTP Test Log, see Figures 3 and 4.

With the TPTP Test Log editor, you can navigate quickly from one test event to another, navigate from an event to its source, filter for different kinds of event verdicts (pass, fail, inconclusive, etc.) and perform other handy tasks. You can even save the log for later analysis. The log is saved in its own .execution file, another Zip archive containing an XML file.

Figure 3 | The Events tree in the TPTP Test Log editor



REMOTE TESTING MEANS BETTER RESULTS

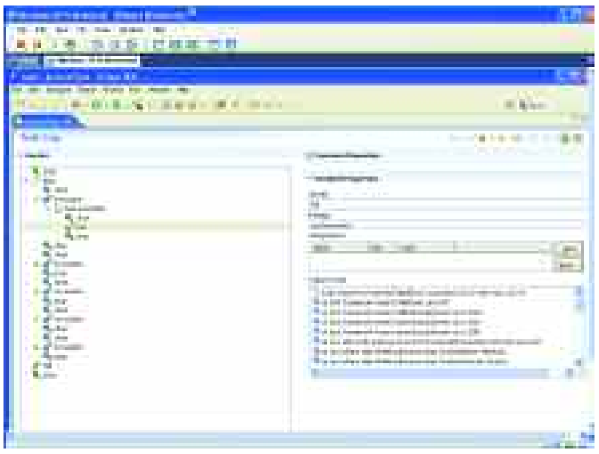
Many applications involve the transfer of data or instructions from one computer to another. Sometimes you can test these applications by creating fake transfers, running the sender and receiver simultaneously on one computer. But fake transfers don't simulate all the effects a network can have on application execution or performance. Instead of faking the transfers on one box, you're much better off using remote testing, where your LAN and WAN network become an integral part of the test.

A suite of remote tests tell you whether an application can or cannot withstand the uncertainties associated with network transmissions. TPTP has facilities for doing remote testing. Here's how it works:

- A *location* is a computer that's running an application to be tested. You can think of a location as a URL referring to a particular computer, but a location has other properties. These properties describe the environment in which the target application runs. (What's the `CLASSPATH`? Which Eclipse plug-ins are enabled? And so on.)
- An *artifact* is a unit of information describing some aspect of a test suite. The `AccountTest.testsuite` file described earlier is an example of an artifact.
- A *deployment* pairs artifacts with locations.

Each location, each artifact and each deployment is called a *test asset*. TPTP provides editors for all kinds of test assets. Figure 5 shows a page of the editor for a deployment asset. As you see in Figure 5, the deployment associates a test suite (part of the `AccountTestArtifact`) with a remote location called `TheServerOnTheOtherSideOfTheWorld`.

Figure 4 | Detailed properties of a test failure



It takes some experience with TPTP to get a pairing like this to work correctly. That's because effective communication between the local and remote machines depends on properties and settings for all of the test assets.

If you run into trouble, you should visit the Deployment Ground Rules page of the TPTP help files. (As with other plug-ins, these help files become available when you install the TPTP plug-in.) Sometimes it helps to post a query on the eclipse.tptp newsgroup. (Point your news client to news.eclipse.org.)

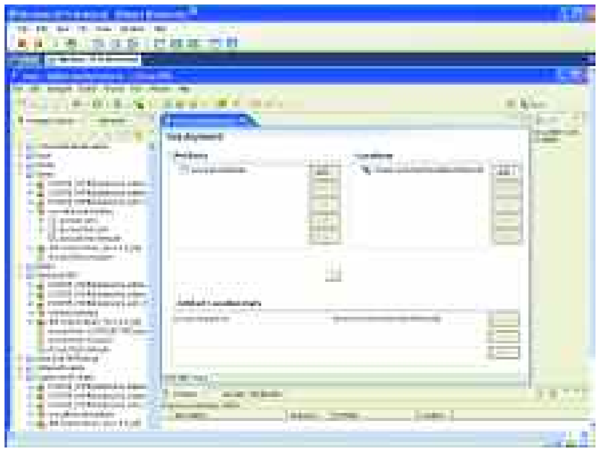
RUNNING TESTS WITH VARIABLE DATA

To test a class, you may want to run the class' methods several times— using different input data each time. One time you make the account balance positive, another time you make it zero or negative, and for another run you omit the balance or try to make the balance be a non-numeric value. Later, you try different pairs of balances and interest rates.

TPTP provides a nice way for you to manage the changing of input data values. It's called a *datapool,* another member of the category called "test assets." Like other kinds of test assets, the datapool has its own editor. This editor includes a form for setting general properties of the datapool and a spreadsheet for modifying datapool values, as you can see in Figure 6.

Like the spreadsheet in a familiar office suite, the datapool spreadsheet is divided into one or more worksheets. Each datapool worksheet is called an *equivalence class*; each equivalence class stores rows of related data values. Of course,

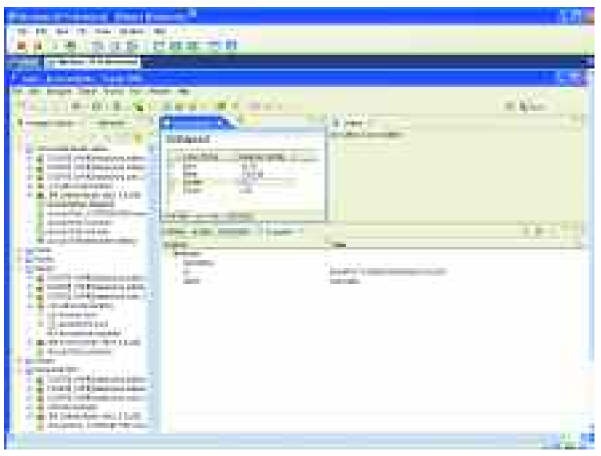
Figure 5 | The Deployment editor



you don't have to type values into the spreadsheet by hand. TPTP lets you import the data from a CSV file.

With a datapool in hand, you modify your testing code to make calls to the datapool API. This API includes an iterator that steps through the rows of the datapool table, grabbing different data each time you make a `getValue` call.

Figure 6 | The spreadsheet page of a datapool editor



AUTOMATE TEST RUN

You can create programs and scripts to perform unattended runs of your test suites. In a Java program, you instantiate a TPTP class named `AutomationClientAdapter` and call the instance's execute method.

In an Ant script, you create a target with a `tptp:test` tag. In a shell script, you run the `AutomationClientAdapter` class using the Java VM. In any of these scenarios, you can run large numbers of tests, change characteristics of tests on-the-fly, integrate test results with other processes, and do all the nice things you can do with programmatic runs.

THE AUTOMATED GUI RECORDER

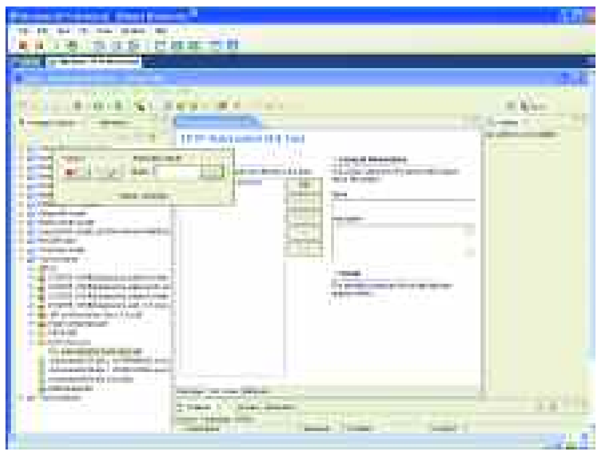
A GUI program can be difficult to test. You don't just tell the program to run, have the program slurp up some data, and then watch the program spew out results. Instead, you expect a user (often a naïve user) to click buttons, fill text fields, and do all sorts of things in some strange, unanticipated order. That's the nature of a GUI program.

So, to test a GUI program, you want to simulate sequences of mouse events and keyboard events. This can be accomplished by installing an add-on to TPTP called the Automated GUI Recorder (AGR). To download the AGR, visit the regular www.eclipse.org/tptp Web page.

To use the AGR, start by creating a new Plug-in Project. (The AGR doesn't run on plain old non-plug-in Java Applications, GUI or otherwise.) Next, you create a new TPTP Automated GUI Test Suite. Within that suite, you create a new TPTP Automated GUI Test.

As you march through the steps in the Automated GUI Test wizard, you specify the Eclipse perspective in which your new plug-in will operate. When you finish the wizard, Eclipse jumps to whatever perspective you specified and adds a floating control center dialog like the one you see in Figure 7.

Figure 7 | The AGR control center dialog



From that point on, the dialog works very much like an old-fashioned macro recorder. You press keys and click your mouse on the Eclipse workbench. The AGR records these events in one of two ways. It records the fact that you click certain objects, or it records the fact that you click at certain coordinate locations. Of course, the click-on-objects way of recording is usually better, so the documentation encourages you to choose this option.

To stop recording, click the little red square in the control center dialog. Eclipse returns immediately to whatever perspective it had before recording started.

Your test is now recorded in an XML document. The document is zipped and encoded inside a `.testsuite` file, but you can easily view the document in the editor for TPTP Automated GUI Tests. If you're satisfied with your recording, you can replay all of its actions. When you do, you get a test log (a `.execution` file) just as you do when TPTP runs other, non-GUI tests.

TEST REPORTING WITH BIRT

You may already be familiar with BIRT, Eclipse's Business Intelligence and Reporting Tools project. Using BIRT, you can generate reports of all shapes and sizes, making tables, charts, graphs and anything else you want from existing data.

What if the "existing data" come from a TPTP test? The newer versions of TPTP integrate nicely with BIRT. So if you don't like the reports shown in Figure 3 and Figure 4, you can create your own. You can combine and illustrate test results in a way that suits your needs.

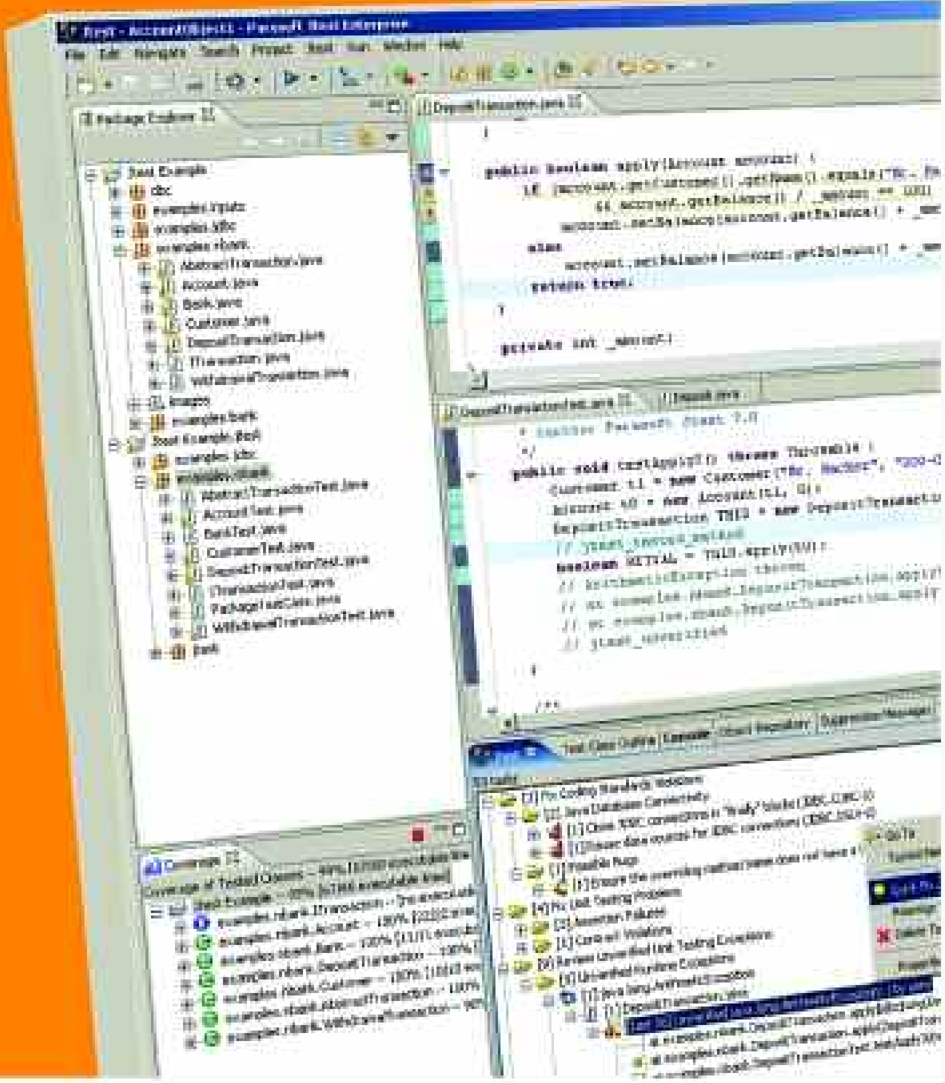
HAPPY TESTING

As the saying goes, "The job isn't finished until the paperwork is done." People used to apply this saying to software testing, not to traditional bureaucracy. But these days, the testing isn't the final paperwork.

Indeed, as we know, testing is a continuous part of the software life cycle. Eclipse with the Test & Performance Tools Platform encourages continuous integrated testing. It makes testing easier, more uniform and more manageable.

Code in quality. Test out bugs.

Take control of your code through Automated Unit Testing and Code Analysis...



```
public boolean apply(Account account) {
    if (account.getCustomer().getName().equals("Mr. Ba
            && account.getBalance() / _amount == 100)
        account.setBalance(account.getBalance() + _amo
    else
        account.setBalance(account.getBalance() + _am
    return true;
}

private int _amount;
```

```
 * @author Parasoft Jtest 7.0
 */
public void testApply1() throws Throwable {
    Customer t1 = new Customer("Mr. Bachmr", "000-0
    Account t0 = new Account(t1, 0);
    DepositTransaction THIS = new DepositTransactio
    // jtest_tested_method
    boolean RETVAL = THIS.apply(t0);
    // ArithmeticException thrown
    // at examples.nbank.DepositTransaction.apply(
    // at examples.nbank.DepositTransaction.apply(
    // jtest_unverified
}
```

PARASOFT **Jtest**®

- Automatically analyzes your code, applying 500+ Java coding best practices that surface poor code constructs affecting reliability, security and performance.
- Automatically unit tests your Java code evaluating code behavior and exposing unexpected exceptions, boundary condition errors and memory leaks.
- Generates extendable, high coverage test cases in JUnit format that can be quickly turned into libraries of reusable test assets.
- 100's of Quick Fix options for fast, accurate resolution of errors.
- Integrated code coverage analysis ensures that your code is thoroughly tested.
- Team collaboration support allows you to define, distribute and enforce code quality standards across entire development teams.

Stop bugs before they start:

Jtest automates the valuable, yet often tedious tasks of code reviews and unit testing allowing development teams to adopt a "test as you go" strategy that promotes testing code as it is developed so that quality is built into the code from its inception and bugs eliminated before they infect the rest of the code base.

The result? Cleaner, safer, more reliable and consistent code. Reduced code rework. Faster, more predictable release cycles. Reliable applications and happier, more productive end-users. **To try Jtest today, call 888-305-0041 (x-3501) or go to www.parasoft.com/EclipseReview.**

Parasoft Jtest clients include: Lehman Bros., Cisco, Disney, Raytheon, ADP, Sabre Holdings, GE and more.

PARASOFT
We make software work.

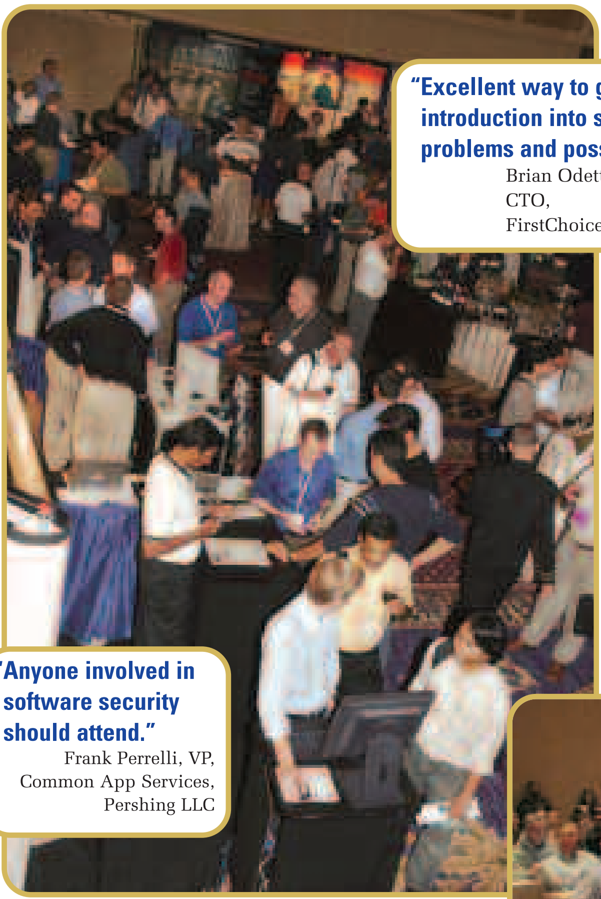
Automated Error Prevention™

Register by April 28 for Super

June 5-7, 2006
Hyatt Regency | Baltimore, MD
www.S-3con.com

"There is real meat in the conference. You also get to meet industry leaders."
Sharon Xia,
Software Developer,
AttachmateWRQ Inc.

The only technical conference focused on software security at the applications level **is coming**



"Excellent way to get a good introduction into software security problems and possible resolutions."
Brian Odette,
CTO,
FirstChoice Solutions

More Than 40 Classes & Tutorials!

"Anyone involved in software security should attend."
Frank Perrelli, VP,
Common App Services,
Pershing LLC

"The conference provides a nice focused look at software security. Very informative."
Paul Waibel,
QA Engineer,
Ceridian

A BZ Media Event

Attend the Software Security Summit!

Early Bird Savings!

Save $300!

SOFTWARE SECURITY SUMMIT EAST

to Baltimore!

"A good place to learn how hackers work and how to stop them."
M. Wan, Development Manager, Thomson Financial

"Eye opening."
Amy Haselhorst, QA Engineer, Ceridian

"It's THE conference for developers and testers who are concerned with software security."
Wei Feng, Senior Architect, GrapeCity Inc. (Japan)

- Learn How to Build Secure Software
- Secure the Software on Your Network
- Test the Security of Your Software
- Implement a Layered Approach to Application Security
- Architect Security Into the Development Life Cycle
- Understand Software Security Vulnerabilities

"If you are interested in writing more secure software, go to this conference."
Richard Morris, Staff Firmware Engineer, Applied Biosystems

Secure Your Software
www.S-3con.com

www.S-3con.com

Subscribe Today!
www.EclipseReview.com

Develop Web Applications FASTER With WTP!

BY TIM WAGNER

Editors, wizards, tools and APIs—thanks to the Web Tools Project, all these facilities are standing by to help you develop and debug Web and J2EE applications faster than ever before.

WTP is divided into two subprojects, Web Standards Tools (WST) and J2EE Standards Tools (JST). The names reflect WTP's charter, namely that it provides tools for standards-based runtimes. In contrast, emerging frameworks such as Spring, Hibernate and Beehive are outside WTP's charter, and are left for commercial and other open source tool projects to address.

WTP is distributed at no charge as open source software, and is licensed for use under the Eclipse Public License (EPL).

WTP INCLUDES EDITING SUPPORT FOR A WIDE VARIETY OF LANGUAGES OTHER THAN JAVA THAT ARE REQUIRED FOR WEB APPLICATION DEVELOPMENT.

Many of the companies behind WTP contributing code and resources are also adopters of WTP themselves; they're building commercial products on top of its frameworks and incorporating some or all of the tool set into to their own products. The result offers a variety of choices: You and your team can choose freely between the standard WTP distribution or one of the specialized versions that offer additional tools, server support and other features. Let's look at what WTP can do, how it works, and what it can do for you.

EDITING AND SUPPORT

All Eclipse projects provide a combination of end-user tools and a platform for adopters, which is a set of APIs that can be used by third parties to build new tools using the project's infrastructure.

Probably the first thing you'll notice after installing WTP is that there's a new perspective designed to help you author and deploy applications, the "J2EE" perspective. This perspective includes the project navigation view—a replacement for the standard package explorer view from Java that provides a specialized view of Web and J2EE projects, including virtual navigation into particular categories, such as the servlets in a project. (Note that when debugging an application, WTP will automatically switch to the debugging perspective and that context's collection of views.)

Use of the new J2EE perspective and its views is optional, of course, and you can always return to the Java perspective.

WTP includes editing support for a wide variety of languages other than Java that are required for Web application development, including JSP, HTML, CSS, JavaScript, XML Schema, WSDL, DTD and XML. The platform makes editing in these languages similar to the experience of editing Java in Eclipse. You get a high fidelity experience that includes "check-while-you-type" capabilities.

The Web Tools Project also supports code assist, formatting, and other "intelligent editing" operations, depending on the language in question. Java snippets in JSP even respond to Java refactoring events (see Figure 1), which helps to keep the application consistent across changes to the model and the UI.

Tim Wagner *is the project lead for the Eclipse Web Tools Platform Project and manages the BEA engineering team responsible for the Workshop product line. Dr. Wagner was the program chair for the 2006 Eclipse Conference and also serves on the Eclipse Architecture and Planning Councils. His professional background and patent contributions span compilers, data integration/XQuery, IDEs and J2EE tools. Research interests include metadata-driven development and visualization techniques for service-based architectures.*

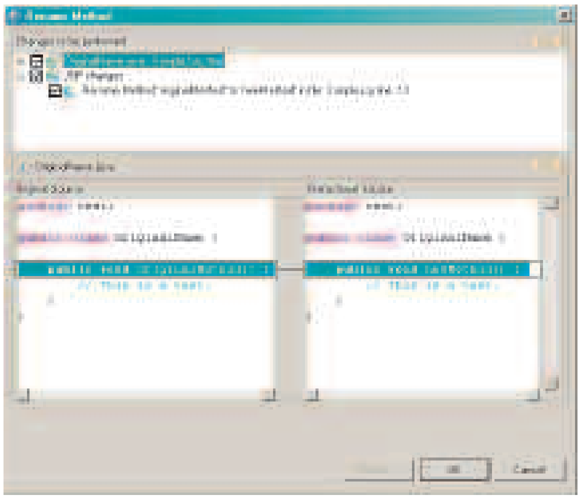
WTP's JSP editor uses the Java editing features from Eclipse's Java Development Toolkit (JDT), so that you get the same experience with Java fragments embedded in JSP that you would in editing a standalone Java file. In addition, WTP's JSP editor provides taglib, HTML and CSS checking. The platform exploits the built-in debugger support for The Java Community Process's JSR 45 ("Debugging Support for Other Languages") for JSP debugging based on either source (JSP) or Java-level views.

In addition to source editing support, WTP provides graphical editing for WSDL and XML schemas that allow even beginners to view, navigate and modify documents written in these languages without a complete understanding of their syntax. The graphical overview, property view and document structure view cooperate to provide a complete navigational and editing package.

SERVER VIEW AND CONFIGURATION

Editing is great, but once your application is written, you'll need to deploy it to your favorite server. While

Figure 1 | JSP participating in Java method refactoring



high-performance implementations of commercial J2EE application servers are generally available from their vendors, basic implementations for several of the many commercial and some open source Web and application servers are available out-of-the-box with WTP. These include Derby, Geronimo, Glassfish, JBoss, JOnAS, Oracle Application Server, Tomcat, BEA's WebLogic and IBM's WebSphere.

WTP wizards support the selection of any of the server types you have installed, as shown in Figure 2. With the server selected, you can configure a particular instance of it using the visual server configuration tool (see Figure 3).

To see what state your various server instances are in, or deploy applications or modules to them, you can use the server view, which is like a miniature built-in console, as shown in Figure 4. The server view provides a dynamic display that captures the list of active

THE PROJECT VISION OF JSF IS TO ADD SUPPORT TO THE ECLIPSE WEB TOOLS PROJECT TO SIMPLIFY DEVELOPMENT AND DEPLOYMENT OF JAVASERVER FACES APPLICATIONS

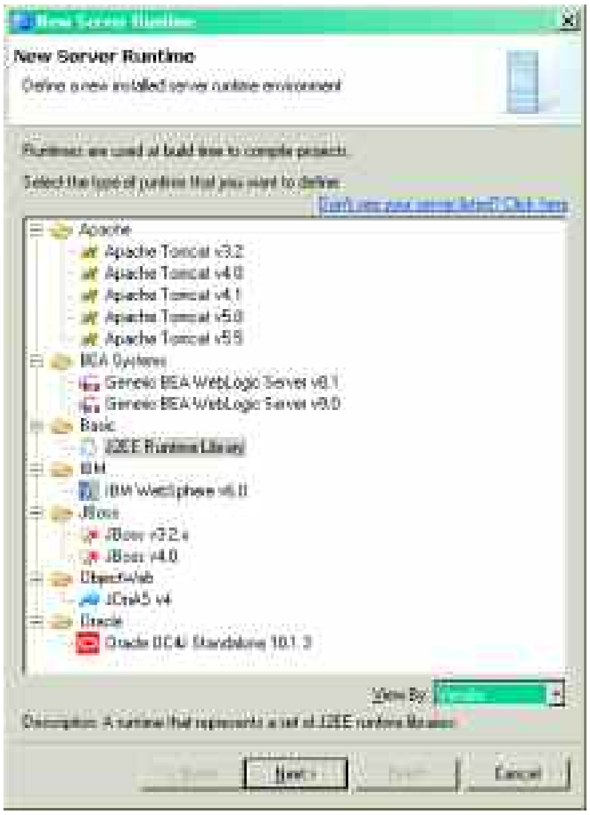
servers, the state of each ("synchronized" means that the server's current state matches that of the corresponding Eclipse project source) and any important status messages. Nested below the server name in the first column are the names of the deployed modules.

FACETS

When working with traditional Java projects, IDE users are accustomed to handling certain environmental details, such as the JRE version (1.4 versus 1.5, for example) or setting the classpath. All of these issues apply in WTP projects.

However, there are also a host of new issues that arise from the server context. For example, Enterprise JavaBeans can be deployed to an application server, such as Geronimo or WebLogic, but cannot be run by Tomcat. Modules designed to exploit a proprietary feature of WebSphere may not work with WebLogic and vice versa. To handle these differences, the WTP team created a way

Figure 2 | Using the WTP wizard to define a new server type



New Server Runtime

Define a new installed server runtime environment

Runtimes are used at build time to compile projects.

Select the type of runtime that you want to define:

```
Apache
    Apache Tomcat v3.2
    Apache Tomcat v4.0
    Apache Tomcat v4.1
    Apache Tomcat v5.0
    Apache Tomcat v5.5
BEA Systems
    Generic BEA WebLogic Server v8.1
    Generic BEA WebLogic Server v9.0
Basic
    J2EE Runtime Library
IBM
    IBM WebSphere v6.0
JBoss
    JBoss v3.2.x
    JBoss v4.0
ObjectWeb
    JOnAS v4
Oracle
    Oracle OC4J Standalone 10.1.3
```

Description: A runtime that represents a set of J2EE runtime libraries to associate additional environmental constraints with a particular project, called “facets.” Facets help users understand which projects (or features used by a project’s source code) can successfully be deployed to which runtimes. The primary benefit is a simpler, more consistent user experience. A sample facets selection screen is shown in Figure 5.

At the API level, the project facets framework helps a plug-in developer think of projects as being composed of “units of functionality” that can be added and removed by the user. This framework avoids the need to have every software company create its own project wizards or activate/deactivate menu items to enable specific functions on a project. So, you can simply select facets supplied by the plug-in writer in the project creation wizard and the new features are activated.

Another major benefit is the ability to accurately model the capabilities of various server runtimes by indicating which facets the server supports. Accurately modeling the server functionality enables the WTP user

THE ECLIPSE TECHNOLOGY PROJECT PROVIDES AN ARENA IN WHICH NEW TECHNOLOGIES CAN BE INCUBATED UNTIL THEY ARE SUFFICIENTLY COMPLETE TO GRADUATE INTO ONE OF THE EXISTING TOP-LEVEL PROJECTS.

interface to prevent the user from accidentally using features that aren’t available on the target runtime.

JAVASERVER FACES

The JavaServer Faces Tools project is a sub-project of WTP. The project vision is to add comprehensive support to the Web Tools Project to simplify development and deployment of JavaServer Faces applications. The JSF project team is actively working to come out of the incubation phase with a technology preview that ships as part of the WTP 1.5 release in June. Figure 6 illustrates the JSF tooling in use.

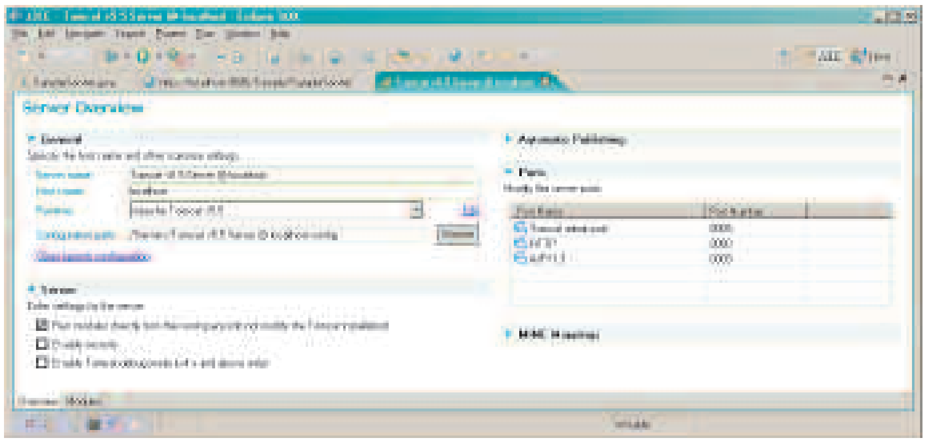
The first release of the JavaServer Faces Tools project will include a productive JSF-JSP Page Source Editor. The planned features include an extensible, metadata-driven Content assist, Quick assist, Quick Fix and Hyperlink for attribute values of specific tags of a taglib. The editor will also feature a component palette to add tags to the source editor.

The JavaServer Faces Tools project will provide a sophisticated multipage editor for the application configuration files that includes a graphical diagram editor for defining the navigation rules and a form-based editor for defining managed beans and other elements in the configuration file. An EMF model for the configuration files will provide rich validation and refactoring capabilities, while enabling ISVs to extend the existing capabilities with features of their own.

EJB 3.0 TOOLS (DALI)

The Eclipse Technology project provides an arena in which new technologies can be incubated until they are sufficiently complete to

Figure 3 | Capturing information about a server instance with the WTP Server Overview



Other projects relating to WTP

There are several related Eclipse projects and project proposals that may be interesting to users of WTP:

The PHP technology project provides editing, debugging and server simulation for the PHP language using the WTP infrastructure.

The DTP top-level project is providing data services, including connection managers, SQL editing and models for this functionality. WTP will be migrating from its existing RDB implementation to DTP after the 1.5 release.

The STP top-level project provides SOA (Service-Oriented Architecture) tools and builds on the Web Services and deployment support in WTP.

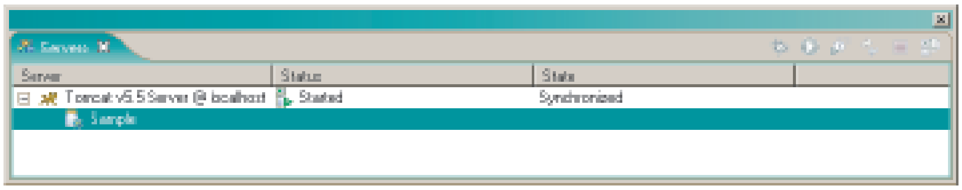
The ATF technology project proposal aims to bring AJAX tooling to Eclipse. As of mid-March 2006, it is currently gathering community feedback prior to its creation review.

The Lazlo technology project is another RIA (Rich Internet Application) technology.

IN ADDITION TO THESE USER TOOLS, DALI PROVIDES MECHANISMS THAT ENABLE ISVS TO ENHANCE AND EXTEND THIS FUNCTIONALITY FOR THEIR OWN EJB 3.0-BASED PERSISTENCE IMPLEMENTATIONS.

graduate into one of the existing top-level projects. One of these projects, Dali, represents tooling for the soon-to-be-completed Enterprise JavaBeans (EJB) 3.0 specifi-

Figure 4 | The WTP server offers a console-like view for server status



Server	Status	State
Tomcat v5.5 Server @ localhost	Started	Synchronized
Sample		

cation. When the specification is complete, the expectation is that Dali's tools will migrate into WTP for further development, shipping as a technology preview along with WTP 1.5.

The Dali EJB Object-Relational Mapping project provides frameworks and tooling for the definition and editing of object/relational mappings for EJB 3.0 Entities. The tooling focuses on minimizing the complexity of creating, editing and updating O/R mappings. Dali includes enhancements to the existing Java editor and property view functionality as well as additional views that help you exploit the default rules built into the EJB 3.0 persistence specification and view/edit this information through the properties view.

You can also leverage relational tools built into WTP to provide context-sensitive values for table and column selections, extend project validation to augment the "Problems" view with errors and warnings that assist them in creating valid entity definitions and automate common tasks. These tasks can include the generation of Entity definitions from Tables or Table generation from Entities to jump start new projects and prototypes. Problems are detected based on mapping rules and by verifying that default values resolve correctly against the database schema in use.

In addition to these user tools, Dali provides mechanisms that let software companes enhance and extend this functionality for their own EJB 3.0-based persistence implementations or to enhance or modify the provided toolset.

WTP has a host of other user tools that we haven't touched on here. Check out the WTP Web site for more extensive information relating to:

- Web services, including wizards to handle top-down (start from WSDL) and bottom-up (start from JavaBean) cases.
- Web services explorer, a built-in Web application that includes UDDI browsing and dynamic WSDL execution.
- EJB Session and Message Bean creation wizards.
- Servlet creation wizards.

The current release of WTP is 1.0.1 (February 2006). The next major release will 1.5; planned to be available in late June 2006 as part of the joint project release known as Callisto, the simultaneous release of Eclipse 3.2 with several top-level projects—more on this later.

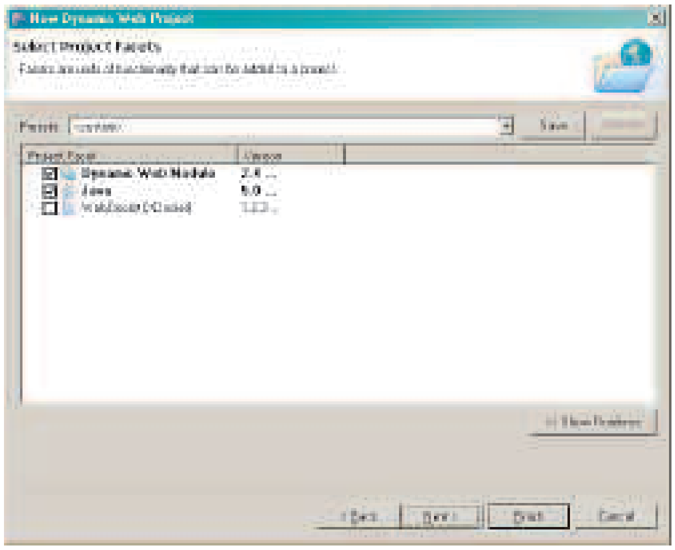
Based on adopter feedback, WTP may also release additional service packs on the 1.0.x line as needed. WTP 1.5 will update the existing WTP capabilities and provide the initial releases (in technology preview status) of JSF and Dali. These previews will enable the community to begin using these features prior to the finalization of their APIs.

OTHER PLANNED 1.5 IMPROVEMENTS

Apart from the two new technology previews described, the remainder of WTP activity for release 1.5 will be focused primarily on improvements to its platform, although the tools will be receiving refreshes, including bug fixes and some performance enhancements.

The Eclipse platform is making major investments in several key areas for its 3.2 release, including improvements to

Figure 5 | The project facet selection page helps users indicate features needed from the server

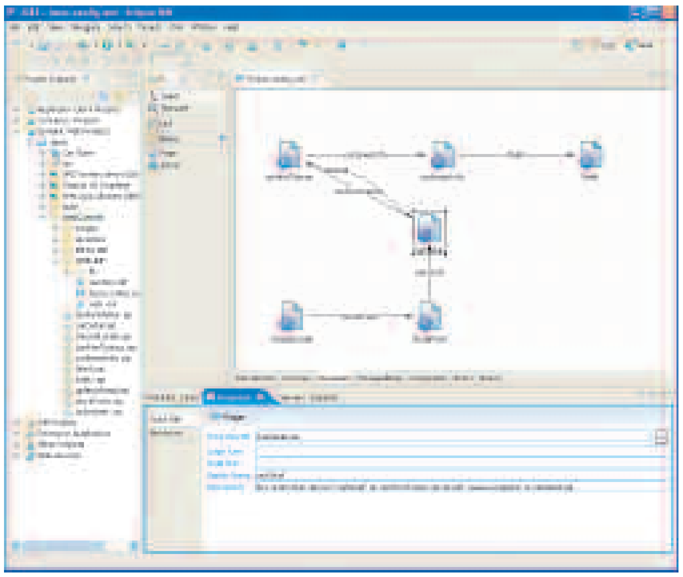


Project Facet	Version
☑ Dynamic Web Module	2.4 ...
☑ Java	5.0 ...
☐ [illegible]	[illegible]

common navigation framework command handling implementation and an updated undo/redo infrastructure.

WTP 1.5 will build on top of Eclipse 3.2 and will incorporate many of its improvements. It will also include changes that will enable validation to run in the background as well as other performance-related functionality.

Figure 6 A graphical diagram editor for navigation rules in the JSF project tools



Updates to the build and deploy infrastructure will be added to improve performance on projects that have many files, along with support for Apache Axis 1.3, an implementation of the Simple Object Access Proptocol (SOAP).

To make it easier for adopters to customize the behavior and appearance of WTP at a fine level of granularity for their own product needs, additional support will be added based on adopter feedback, including more granular capabilities and additional wizard customization hooks.

WTP's ongoing division into individual features will let you "pick and choose" tools or elements that are useful to your work. Software companies will also benefit from the breakdown as part of the enhancements. In addition, the new release will offer better organization of the generic server plug-ins, updates for specific servers, and the removal of old and obsolete server definitions.

WTP AS A PLATFORM

In this article we've focused mostly on the Web Tools Project as a Web developer sees it—a collection of tools that make Web and J2EE application development faster and easier. But as we've mentioned, WTP is also a platform, carefully designed so that software companies can use its framework as part of their own commercial or open-source offerings.

In addition to adhering to the Eclipse Software Development Process, such as the use of Bugzilla, open communication and community involvement, WTP follows several organizing principles to ensure that adopters can successfully leverage the platform. The Eclipse Project has set a high standard for technical excellence, functional innovation and overall extensibility within the Java IDE domain. WTP applies these same standards to Web/J2EE application tooling.

A major goal of WTP is to support a vital application development tools market. WTP's out-of-the-box functionality is useful on its own, but has also been designed from the start to be extensible, so commercial vendors can use what the project delivers as a foundation for their own product innovation.

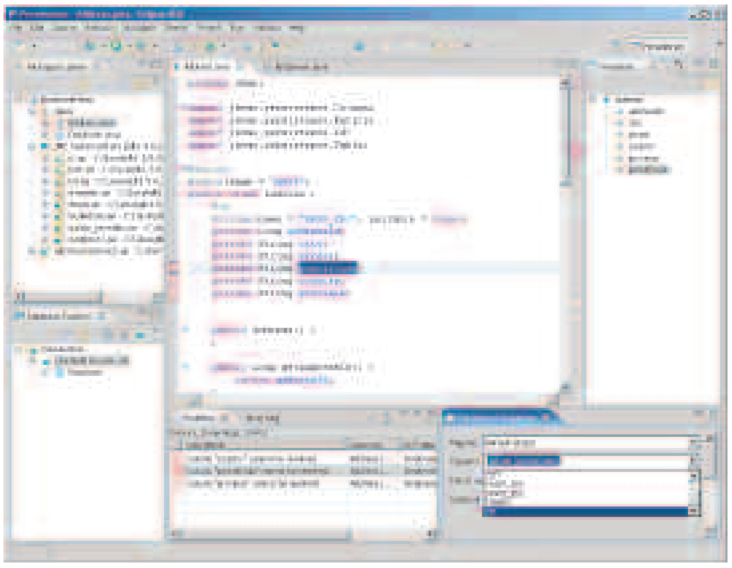
By the same token, vendor neutrality is at the core of WTP. It encourages Eclipse participation and drives Eclipse market acceptance by strengthening the long-term product value propositions of application development vendors.

WTP delivers an extensible, standards-based tooling foundation on which many software companies can create value-added development products for their customers and end users.

Although WT''s focus is on runtime technologies with accepted standards and existing deployments, it also tracks emerging standards where leading-edge tools are desirable. Where multiple technologies are widely used for a given functional need, it attempts to support each, subject only to technical feasibility and the goal of providing the most capable and extensible foundation for the long term.

WTP introduces new APIs in a provisional status for one major release, so that adopters have the opportunity to evaluate, comment and build trial implementations on

Figure 7 An illustration of the Dali tools in action



them. If all goes well, the API is declared in the subsequent release; otherwise, the feedback is used to refine the interface until it meets adopter needs. In other words, no API is released before its time.

The platform's interfaces with other top-level products, including the Eclipse platform itself, are evaluated and rearchitected as needed for each release to eliminate redundancy and ensure overall architectural integrity. In addition to providing its adopters with APIs, WTP treats this part of its community as first-class citizens with adopter "hot list" bug tracking, requirements gathering and API adoption tools. All of these facilities are available to the community as a whole, but they are most useful for companies seeking to use WTP in their products.

THE ECLIPSE PROJECT HAS SET A HIGH STANDARD FOR TECHNICAL EXCELLENCE, FUNCTIONAL INNOVATION AND OVERALL EXTENSIBILITY WITHIN THE JAVA IDE DOMAIN.

GETTING WTP

WTP 1.0.1 and earlier versions are available in a variety of packages from the WTP download site. The WTP standalone binaries and SDKs can be found at download.eclipse.org /webtools/downloads. WTP bundled with the corresponding Eclipse platform and all tool project prerequisites (EMF, GEF and VE) can be found at the same location. An update from 1.0 to 1.0.1 (and eventually from 1.0.1 to 1.5) resides at download.eclipse.org /webtools/updates /index.html. While WTP includes out-of-the-box support for several servers, you will need to download and install those servers separately. See the "Getting Started" section of the Web site (especially the tutorials), the included documentation and the WTP newsgroup (eclipse.webtools) for additional help.

HELPING OUT

As with all open-source projects, WTP advances through donations, especially donations of time. Contributing code is great: If you're interested, please see the "help wanted" ads in the planning documents or contact the development team at wtp-dev@eclipse.org. You can find out more about developing with WTP on www.eclipse.org in the "Development" area.

Not able to help code? There are other ways to improve WTP—even on a very part-time basis. The project can always use help in writing tutorials for the WTP Web site, testing (and filing any bugs you find), incrementally improving the documentation, or even helping out others on the newsgroup (eclipse.webtools). As with all open-source projects, when more people are involved, the result will inevitably be a much more useful tool.

Expediting Tasks in the Eclipse IDE

8 Can't Miss Tips For Using Eclipse

BY DWIGHT DEUGO

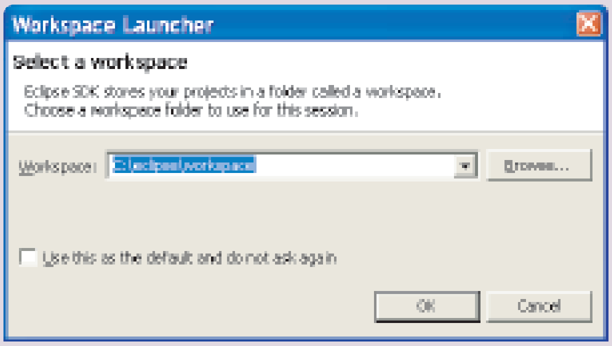
Eclipse is used for different things by different people. For some, it's an integrated development environment for building stand-alone Java applications. Others use Eclipse to build Web applications, C++ applications, embedded applications, plug-ins to extend Eclipse and more. Many companies and individuals also use Eclipse as a tools platform, implementing or porting their tools to work as Eclipse plug-ins by integrating them into the Eclipse SDK as rich client applications. The third category of Eclipse users just want to use the integrated development environment to interact with their favorite tools.

No matter which you are, though, there are many little-known procedures that can make your experience with Eclipse more enjoyable, and we'll cover eight of them here. Some can be performed upon installation. Others will make it easier for you to navigate your way through Eclipse. And some will just make it easier for you to develop Eclipse plug-ins and applications. Before we go any further, let's identify three specific directories we'll use in our eight tips:

- ECLIPSE_HOME_DIRECTORY: the directory where you installed Eclipse.
- ECLIPSE_WORKSPACE_DIRECTORY: the directory containing your Eclipse projects.
- ECLIPSE_RUNTIME_WORKSPACE_DIRECTORY: the directory your Eclipse workbench uses for the worspace when performing runtime testing.

1 PLACE THE WORKSPACE SOMEPLACE USEFUL

Eclipse stores your projects in a folder called Workspace, which is the default name. Provided you never clicked the checkbox, every time you start Eclipse it will ask you which workspace you want to use for the current session. If the workspace location you specify doesn't exist, Eclipse will create one for you. If you accidentally clicked the checkbox and are now unable to specify the workspace location, you can easily return it to its original value. Once Eclipse has started, select the Window->Preferences menu from the workbench. This will open the Preferences dialog. Navigate your way to the Startup and Shutdown preferences and check "Prompt for workspace on startup." Exit Eclipse and start it up again. You should see the Workspace Launcher this time.



Workspace Launcher

Select a workspace

Eclipse SDK stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\eclipse\workspace Browse...

☐ Use this as the default and do not ask again

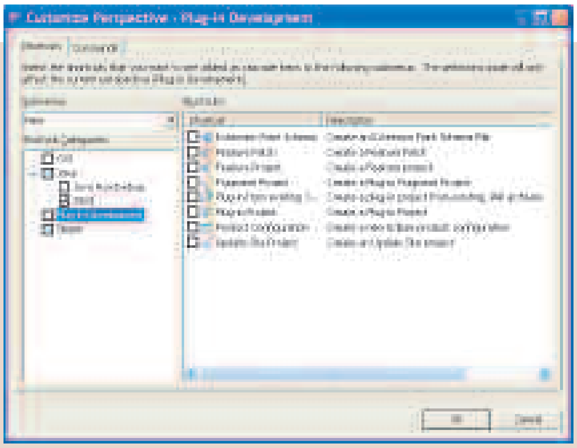
OK Cancel

Just in case you wondered, plug-ins store their preference values in their own directories under the .metadata\.plugins directory of the workbench. For example, look in ECLIPSE_WORKSPACE_DIRECTORY \.metadata\.plugins\. You should see a number of plug-in directories each containing XML files with current preference settings.

This tip helps you identify where you should place your workspace. I create a folder called Eclipse Workspaces and place all of my new workspaces in subdirectories of that folder. It doesn't matter where you put the workspaces. You can follow whatever organizational strategy you want. However, don't put your workspace in your Eclipse installation directory. I have seen people reinstall Eclipse or upgrade to a new version and the first thing they do before grabbing the new download is to delete their old Eclipse installation directory. If your workspace is in there, say goodbye to it when you upgrade! Putting your workspace(s) in a non-Eclipse installation directory will make sure you never suffer this fate.

2

POINT YOUR PLUG-INS



One of three things typically occurs when you start Eclipse with a new plug-in in the ECLIPSE_HOME _DIRECTORY\plugin directory. The best-case scenario is that you start Eclipse and see the new menu items, editors, or views that the new plug-in contributes to Eclipse. In this case, you know that the plug-in is activated and you are ready to use the new functionality.

The next case is that the plug-in is activated, but did not enable selections of its menu, editors, and views. This is actually a feature of Eclipse. Rather than having every new plug-in add new menus, editors and views to the workbench, plug-in developers can initially disable them and let the end-user enable the functionality as needed. To enable a plug-in's features, first select the Window->Customize Perspective menu. This will bring up the dialog.

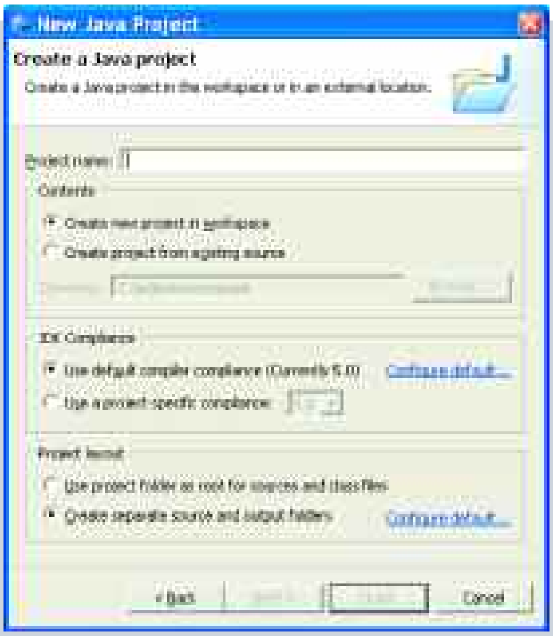
In the Customize Perspective window, you will need to look for your plug-in's category name and enable its functionality. Usually, a close relationship exists between the category name and the name of the plug-in, so you should have no problem finding the category you need to work with. In this window, you can see that the Plug-in Development category is disabled and the corresponding shortcuts are not enabled for the New submenu. By enabling the category, you will find its shortcuts added to the Workbench's File->New menu.

Enable all of the categories in the submenus and then switch over to the Commands tab. Now, enable your plug-in's command groups for the current perspective. After you're done, you should have access to your plug-in's full functionality.

The last thing that can occur with a new plug-in—and the worst—is that you can't find any evidence that it activated. If you run into this problem, you need to navigate to your Eclipse workspace. This is the location you identified when Eclipse started, as shown back in our first tip. Now, look for the file ECLIPSE_WORKSPACE_DIRECTORY\.metadata\.log. If there is a problem with your new plug-in, the contents of the log file should give you some indication. For example, it might give you a stack trace of an exception that occurred while activating the new plug-in. Knowing where the log is will not fix the problem, but it can give you an idea of what to do next.

3

SEPARATE YOUR SOURCE AND OUTPUT FOLDERS


New Java Project

Create a Java project

Create a Java project in the workspace or in an external location.

Project name:

Contents
- [✓ Create new project in workspace]
- Create project from existing source

JDK Compliance
- [✓ Use default compiler compliance (Currently 5.0)] Configure default...
- Use a project specific compliance:

Project layout
- Use project folder as root for sources and class files
- [✓ Create separate source and output folders] Configure default...

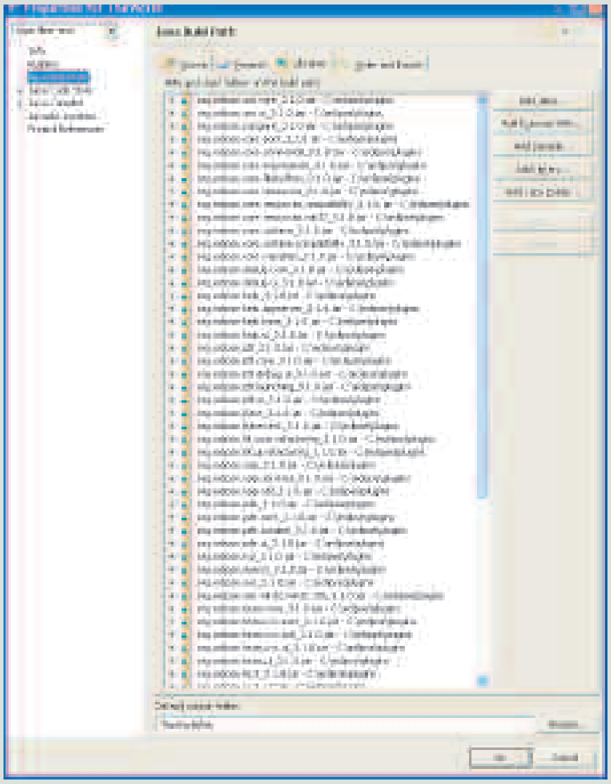
After installing Eclipse, you are ready to create a Java project. Switching to the Java Perspective and selecting File->New->Project brings up the Project Wizard. You can enter a project name and click Next to enter specific Java build settings for a project, or Finish to have the project created using the defaults.

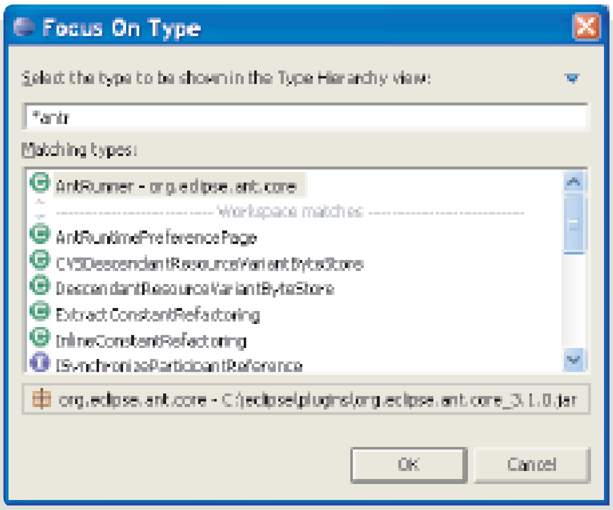
Java source files you create will have their corresponding compiled files located in the same directory. The issue here is that you will have two types of files in the same directory, each with different roles and needs. For example, when you back up your project, you usually don't save the .class files. However, when deploying your application, you need the .class files but not the .java files. You will find that in the future many tasks you perform require you to identify either the source (.src) or the compiled (.bin) files. You can make your life easier by separating your compiled class and java files into different directories from the start.

Separating your source and compiled files is easy. When you create your Java project, make sure to select the radio button that says "Create separate source and output folders" on the New Java Project page. This button is not the default selection after a new installation. This option will create an src directory for your Java packages and source code, and a bin directory for your compiled Java class files. You will see the src directory in the Package Explorer View but not the bin directory. By switching to the Navigator View, you will see they are both present.

4 MAP YOUR PLUG-IN CLASSES

Have you ever wondered how many Eclipse plug-ins are in the standard Eclipse SDK? Just look in your ECLIPSE_HOME_DIRECTORY\plugins directory to get some idea. Each plug-in contributes a number of packages and classes that you can use in your own plug-in development, provided the developers of those plug-ins made their packages exportable. The trick lies in knowing which classes you can use.





Select the type to be shown in the Type Hierarchy view:

*antr

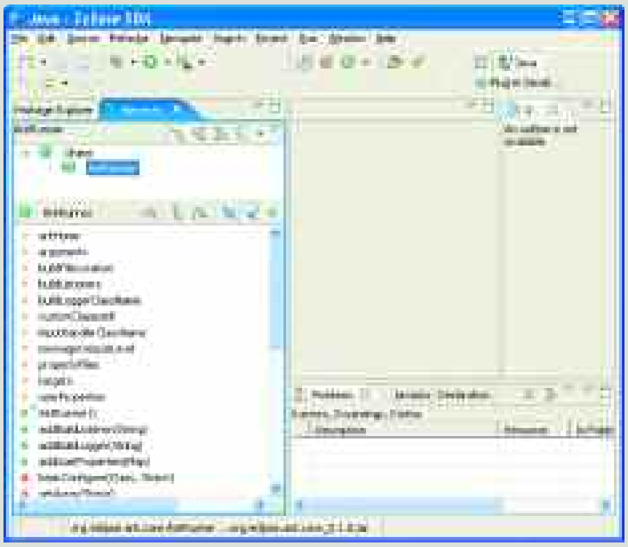
Matching types:

AntRunner - org.eclipse.ant.core
------- Workspace matches -------
AntRuntimePreferencePage
CVSDescendantResourceVariantByteStore
DescendantResourceVariantByteStore
ExtractConstantRefactoring
InlineConstantRefactoring
ISynchronizeParticipantReference

org.eclipse.ant.core - C:\eclipse\plugins\org.eclipse.ant.core_3.1.0.jar

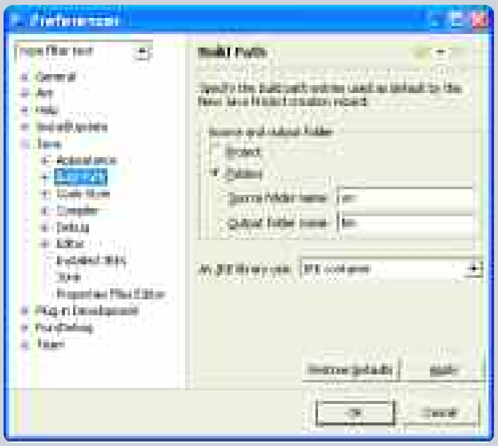
A technique to find the available classes from other plug-ins is to create a plug-in project called TheWorld and then add every plug-in's External JAR file to the Libraries portion of the project's build path. If you get an error saying there are duplicates, simply delete any JARs you just added that created the duplication. This way, whenever I am in the Hierarchy View and use the right button to select Focus On… from the context menu, I am searching all of the available classes. In the Focus On Type dialog, notice that by entering *antr as the search criterion I get all classes beginning with the word Antr from the available plug-ins.

Near the bottom of the Focus On Type window, you can select a class to find out which plug-in it is associated with. Selecting the class in the Hierarchy View will also show you the same information at the bottom of the workbench, as shown below. I believe strongly that the best documentation for Eclipse is Eclipse. So don't wander around in the dark. Get everything where you can find it, see it and use it: in Eclipse.



I almost always forget to select this radio button. To make this the default choice, you can set this Build Path Preference when you first install Eclipse. Selecting the Window -> Preferences menu will bring up the Preferences dialog. Select Java -> Build Path in the Preference dialog and change the radio button from Project to Folders. Result: You will always have your source and compiled code in separate directories.

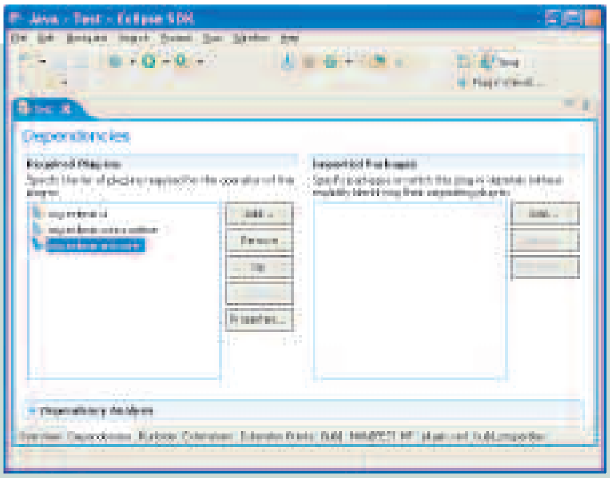
If you prefer, you can use names other than src or bin for these directories, but since these tools are often used by other tools and scripts, such as Ant, think long and hard about using something different. You may not realize it yet, but this small effort will make your code easier to backup, deploy, manage, and integrate with other tools.



Source folder name: src
Output folder name: bin

5

EXECUTE ANT BUILD FILES FROM WITHIN A PLUG-IN


Java - Test - Eclipse SDK
Dependencies
Required Plug-ins
Imported Packages
Add...
Remove
Up
Properties...
Dependency Analysis
Overview Dependencies Runtime Extensions Extension Points Build MANIFEST.MF plugin.xml build.properties

The Eclipse SDK comes with many plug-ins containing classes that you can use when developing your own plug-ins. While you can execute Ant files to build and deploy applications, manipulate files, and a number of other tasks from the Eclipse workbench, how can you do it programmatically from within your plug-in? It's easy. The first step is to add the org.eclipse.ant.core plug-in to the list of dependencies for your plug-in. You can do this by selecting the Dependencies tab in the multi-tab editor that opens when you select your plug-in's plugin.xml file.

Next, add the code in Listing 1, or something similar, to a method that you want to use to execute an Ant build file. The Ant build file build.xml is then identified from the project Test as the build file to execute. There are different ways to get the location, and this is just one example. After getting the location, set the instance of the class AntRunner to use that file's location and then send it the run message. You can use different monitors to increase liveliness. However, the instance of a NullProgressMonitor will get you started. Listing 2 provides imports that will ensure that your code compiles.

Listing 1 | Identifying build.xml as the Ant file to execute

```
IProgressMonitor monitor = new NullProgressMonitor();
IWorkspaceRoot root = ResourcesPlugin.getWorkspace().getRoot();
IProject project = root.getProject("Test")
IFile file = project.getFile("build.xml");
IPath path = file.getFullPath();
String fileLocation = path.toOSString();
AntRunner runner = new AntRunner();

runner.setBuildFileLocation(root.getLocation().toOSString() + fileLocation);
runner.run(monitor);
```

Listing 2 | Imports that will ensure the code compiles

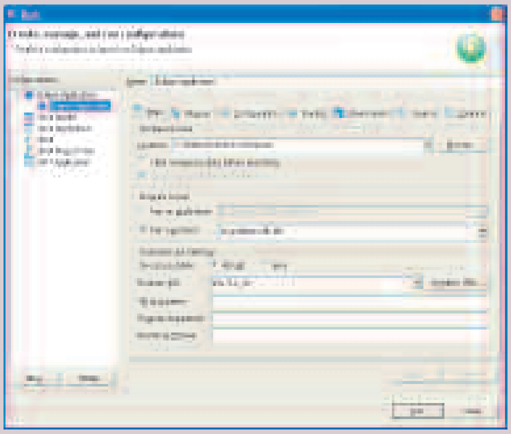
```
import org.eclipse.ant.core.AntRunner;
import org.eclipse.core.runtime.IPath;
import org.eclipse.core.resources.IFile;
import org.eclipse.core.resources.IProject;
import org.eclipse.core.runtime.NullProgressMonitor;
import org.eclipse.core.resources.IWorkspaceRoot;
```

If you are using Eclipse resources such as IFile and IProject to get the project and file paths, add org.eclipse.core.resources to your plug-in's list of dependent plug-ins. If you are using IPath, you will also need to add org.eclipse.core.runtime as a dependency.

6

CLEAR THE RUNTIME WORKSPACE CACHE

Many people new to plug-in development start with Eclipse's Plug-in Development Environment (PDE) perspective, use the associated wizards to build the new plug-in, select the project and select the Run -> Run As -> Eclipse Application from the context menu. This launches a runtime version of Eclipse for testing a plug-in and builds a launch configuration (called Eclipse Application) for you that you can run over and over again by simply clicking the Run button (green button with the white arrow) from the Workbench tool bar.


Run
Create, manage, and run configurations
Configurations:
Name: Eclipse Application
Run
Close

This launch configuration is fine the first time you test your plug-in. However, after that first run, you need to modify the launch configuration to make sure that Eclipse always starts with a fresh runtime environment. At the very least, you want to have the option to clear it before you start your testing. Eclipse caches information about your plug-in when it starts up so that it doesn't have to repeat the effort. However, if your plug-in crashes on start up, Eclipse might use the cached information about it even if you fix the problem and restart the environment.

The tip is to make sure that you are always starting your runtime Eclipse environment with a clean workspace. To make this happen, rather than selecting Run -> Run As -> Eclipse Application to start your

7 GET TO KNOW THE .LOG FILE

Plug-ins, either ones you are building and testing or those you downloaded and just installed, fail to work for all kinds of reasons. Sometimes the reason is that the plug-in's plugin.xml file has a wrong entry. Sometimes it's because a plug-in can't find the other plug-ins it depends on. Other times, the plug-in throws an exception when it starts. So, what can you do when your plug-in doesn't seem to work?

No matter what the reason is or whether it is a development or installed plug-in, your first point of attack is to look at Eclipse's .log file.

If the problem is with a plug-in you downloaded and recently installed, go to your ECLIPSE_WORKSPACE_DIRECTORY\.metadata directory and look for the .log file. If the problem is with a plug-in you are developing, go to your ECLIPSE_RUNTIME_WORKSPACE_DIRECTORY\.metadata to find the .log file. In either case, the content of the .log file will be similar.

What if the .log itself file is missing? It's a safe bet that Eclipse had no problems starting. If the .log file is present, look at its contents for information about what went wrong. When I am developing plug-ins, I often get entries like the .log file shown in Listing 3.

It's easy to see that the first entry suggests I have a null pointer in one of my classes. The second entry shows I have a problem with my DemoPlugin. Other failures also provide information that is somewhat helpful. By looking in the .log file, you will get an idea of what caused the problem. It might not tell you exactly what the problem is, but the logged information should give you some idea of what to do next.

Don't worry about deleting or clearing the contents of the .log file. If Eclipse detects that there is no .log file, it will create one when it needs to write an entry to it. If you just delete the contents, Eclipse will just write the next entry at the beginning of the file.

Listing 3 | Finding problems by looking in the .log file

```
!ENTRY org.eclipse.osgi 2005-08-11 13:51:44.21
!MESSAGE An internal error occurred during: "BankPlugin".
!STACK 0
java.lang.NullPointerException
        at org.bzmedia.test.Bank.run(Bank.java:121)
        at org.bzmedia.test.Worker.run(Worker.java:76)

!ENTRY org.eclipse.osgi 2005-08-14 17:21:32.32
!MESSAGE An error occurred while automatically activating bundle Demo (204).
!STACK 0
org.osgi.framework.BundleException: The activator demo.DemoPlugin for bundle Demo is invalid
at org.eclipse.osgi.framework.internal.core.AbstractBundle.loadBundleActivator(AbstractBundle.java:11)
at org.eclipse.osgi.framework.internal.core.BundleContextImpl.start(BundleContextImpl.java:965)
```

runtime environment, select Run -> Run… This will bring up a Launch Configuration wizard, as shown in the figure. Make sure the Location: entry is set to something other than your current workspace. The default is to use a runtime-workspace. Next, check "Clear workspace data before launching" and "Ask for confirmation before clearing." These selections are not the defaults.

You are now ready to push the Run button. Provided you answered Yes to clearing the runtime workspace data, you shouldn't experience any side effects from the last application test. I have seen perfectly good code fail due to workspace data that was left over from the previous round of tests. It's always hard to debug code that doesn't really have any problems, so remember to clear out the workspace data between tests.

8 PUT ON YOUR LAB COAT, START EXPERIMENTING

As with everything in Eclipse, there are many different ways to do the same thing. I have typically only noted one solution that addresses each challenge, but in most cases there are many more. Take time to explore. Experiment with the IDE. Try new things, test plug-ins, look for shotcuts. Make the toolchain your own. The time you invest in mastering Eclipse will repay you many times over. Perhaps the next Eclipse Review tips article will come from your keyboard, not mine.

Dwight Deugo *is the CEO and Director of Services for Espirity, and is also an Associate Professor in the School of Computer Science at Carleton University. The former editor-in-chief of Java Report, he started the Eclipse Technology Education Project (ECESIS), and is the lead contributor and committer to the project.*

BUSINESS INTELLIGENCE

A BZ Media Publication

6th ANNIVERSARY ISSUE

SD Times ● SOFTWARE DEVELOPMENT

The Industry Newspaper for Software Development Managers

SPECIAL REPORT: DEFECT TRACKING

Search and Destroy

New defect-tracking tools, practices help keep code free of bugs

page 30

FEBRUARY 15, 2006 • ISSUE NO. 144 — www.sdtimes.com • $7.95

Extreme Makeover: New Thinking for JUnit 4

A conversation with software guru Kent Beck

BY EDWARD J. CORREIA

For his books on Extreme Programming, refactoring, test-driven development, Smalltalk and Eclipse, or for efforts toward the founding of the Agile Manifesto and the Three Rivers Institute—it's hard to say how Kent Beck might best be known.

Ironically, many developers may know his name from something he considers simply a hobby: the creation the JUnit testing framework for Java, which he co-authored with Erich Gamma.

Beck characterizes JUnit 4, now in beta, as its most signifi-

▸ continued on page 24

NUnit influenced JUnit, says Beck.

GPL 3.0 Changes How FSF Sees Open Source

Draft addresses license compatibility, software patents

BY JENNIFER DEJONG

The Free Software Foundation has released the long-awaited draft of the GNU General Public License 3.0, setting in motion the first major modification of the widely used open-source software license since the second version was released in 1991.

Unveiled at the Massachusetts Institute of Technology in Cambridge last month, GPL 3.0 addresses compatibility with other open-source licenses, software patents and digital rights management, among other issues.

▸ continued on page 26

WHAT'S NEW IN GPL 3.0?

● **The right to combine GPL 3.0 code with code governed by other licenses.** GPL 2.0 mandated that any code that included GPL-licensed components fell under the GPL. But under GPL 3.0, Apache code would adhere to the Apache license, Eclipse code could follow the Eclipse license, and so forth.

● **A patent retaliation clause designed to discourage patent infringement lawsuits.** Developers who obtain patents for software based on GPL code and use those patents to strike out at others lose their rights to modify and distribute GPL code.

● **A stance on digital rights management.** GPL 3.0 defines encryption and authorization codes (used by some developers to prevent their code from being copied) as source code. As such, it must be made available to other developers. *—Jennifer deJong*

United Beacon Team Shines on SPEM 2.0

Eclipse Process Framework leaders opt unanimously to focus on future OMG spec

BY EDWARD J. CORREIA

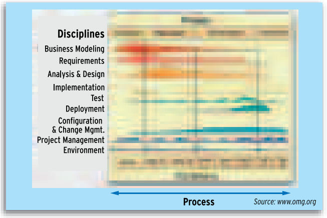
At the first meeting for Project Beacon, the code name for the Eclipse Process Framework technology project, 20 developers from 14 constituent companies created subproject working groups for a tooling metamodel, unified process and agile development.

They also decided at the January meeting to base the tools on a specification not yet finalized by the Object Management Group.

The tooling metamodel will offer a structure for method content and processes using OMG's MetaObject Facility (MOF), UML diagrams and an XML schema. It will be based on IBM's Unified Method Architecture (UMA), which itself is an evolution of OMG's Software Process Engineering Meta-model (SPEM) 1.1, which integrates concepts from IBM Rational's RUP and other technologies.

SPEM 1.1, the current OMG standard, was ratified in January 2005; SPEM 2.0 is expected to be stable next month, but may not be ratified for a year or two. Backers of Beacon, including IBM Rational's Per Kroll, who leads the project, believe it will be better to base the tools on

▸ continued on page 26



The RUP iterative process, which will form a major part of the Eclipse Process Framework, allows work effort to be computed for each discipline over time (from left to right).

Microsoft Hawks Elixir as Cure For Integration

Posts its recipe for CRM-to-Outlook Project

BY ANDY PATRIZIO

Microsoft has posted sample code and documentation for an internal project, called Project Elixir, showing how the company built Web services to let employees access its numerous CRM systems using Outlook.

"The thought behind sharing our experiences on CRM and Elixir was essentially sharing the way we dealt with a set of issues that our customers talk to us

▸ continued on page 23

IN THIS ISSUE

subscribe: www.sdtimes.com

From Modeling to Code Generation—And Back

A Grand Tour of the Eclipse Modeling Framework

■ BY CHRIS ANISZCZYK & BORNA SAFABAKHSH

The Eclipse Modeling Framework is an open-source framework and code-generation tool for building plug-ins and applications based on a structured model. EMF is similar to other XML-binding frameworks like XMLBeans in that it provides tooling to help you manipulate various instances of a given model. We'll explore EMF's tooling, showing you not only how it works, but why you'd want to use it. The code examples used throughout the article are from the library example featured on EMF's Web site, www.eclipse.org/emf.

EMF COMPONENTS

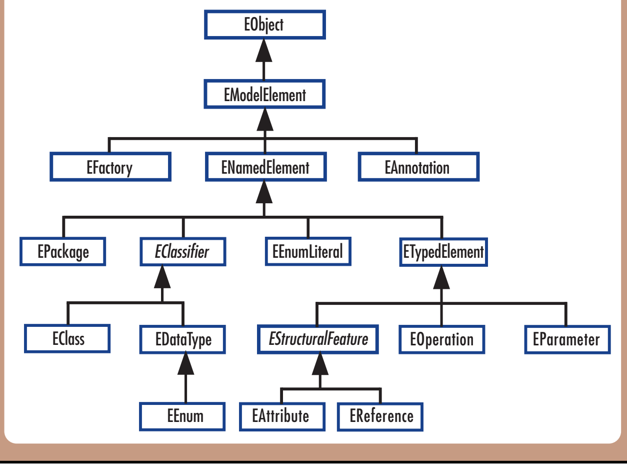
EMF consists of three components: EMF (ECore), EMF Edit and EMF CodeGen. The core portion of EMF includes a metamodel, Ecore, for developing models, as shown in Figure 1. It has a runtime that can manipulate these defined models.

The EMF runtime includes support for tasks like XMI/XML model serialization, reflective model manipulation, observing changes to a model, and recording changes to a model.

The Ecore metamodel (which itself is an EMF model) consists of six fundamental pieces:

- `EClass` represents a modeled object, and has a list of attributes (`EAttribute`) and references (`EReference`). It also contains flags to represent whether the modeled object is abstract or an interface.
- `EAttribute` represents an attribute on a modeled object.
- `EReference` is an association between two objects and has a flag that shows whether the association represents a containment relationship.
- `EDataType` represents a modeled object type like `java.util.Date` or int.
- `EFactory` is a factory that can be used to create instances of your modeled objects (`EClass`).
- `EPackage` is a package that can be thought of as a namespace for modeled objects.

Figure 1 The Ecore metamodel



The EMF Edit framework provides generic and reusable classes for building EMF model editors.

The first set of classes EMF Edit provides are content providers, label providers, property source support, and other convenience classes that allow EMF models to be displayed using Eclipse-based viewers and property views.

The second set of classes includes a command framework that allows EMF-based editors to execute changes to model data, with built-in support for undo and redo.

Also, in case you hadn't realized it, EMF Edit is platform neutral, which means that these components can be reused in something like a Swing graphical user interface.

The EMF code generation framework uses a technology known as Java Emitter Templates (JET) and can generate

Chris Aniszczyk *is a software engineer at IBM's Austin Labs in the realm of Tivoli Security. He is a committer on the Eclipse Modeling Framework Technology / Eclipse Communications Framework (ECF) projects, and also maintains Planet Eclipse (planet.eclipse.org).*

Borna Safabakhsh *is a software engineer within the IBM Software Group. He joined IBM through Extreme Blue after earning a graduate degree in computer science from the Georgia Institute of Technology.*

Listing 1 | The Ecore XMI Library Example

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="library"
    nsURI="http:///org/eclipse/example/library.ecore"
nsPrefix="org.eclipse.example.library">
  <eClassifiers xsi:type="ecore:EClass" name="Book">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="title"
eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="pages"
eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EInt"
        defaultValueLiteral="100"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="category"
eType="#//BookCategory"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="author"
lowerBound="1"
        eType="#//Writer" eOpposite="#//Writer/books"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Library">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name"
eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="writers"
upperBound="-1"
        eType="#//Writer" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="books"
upperBound="-1"
        eType="#//Book" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Writer">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name"
eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="books"
upperBound="-1"
        eType="#//Book" eOpposite="#//Book/author"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="BookCategory">
    <eLiterals name="Mystery"/>
    <eLiterals name="ScienceFiction" value="1"/>
    <eLiterals name="Biography" value="2"/>
  </eClassifiers>
</ecore:EPackage>
```

everything needed to build a complete EMF-based editor. JET leverages the Java Development Tooling (JDT) project in Eclipse to facilitate code generation.

DEVELOPMENT WITH ECORE

In the spirit of flexible efficiency, EMF offers a variety of avenues for defining an application's model. This flexibility lets you describe models in the format you find most comfortable, translating them to Ecore format later.

Using a UML editor capable of exporting an EMF model, the model can be described as a class diagram, specifying classes, attributes of those classes, and associations. This technique is particularly valuable if UML design documentation is available or required, because it provides a visual expression of the model. A UML class diagram for the library example used in this article is shown in Figure 2.

If Java code is your preferred means of model description, EMF lets you annotate existing Java class and interface definitions. By looking for annotation comments such as @model, and assuming standardized "getter" and "setter" naming, the EMF parser and generator can infer model classes, attributes and references.

As application model design evolves, the EMF model can be reloaded to detect changes, and the code merging generator can create code along with the existing implementation:

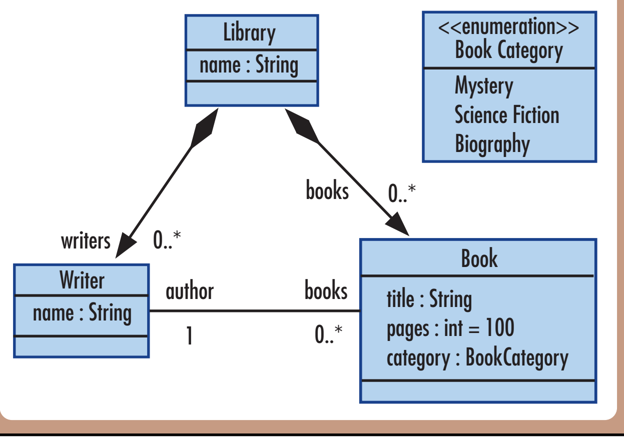
```
/**
 * @model
 */
public interface Book
{
  /**
   * @model
   */
  String getTitle();

  /**
   * @model default="100"
   */
  int getPages();

  /**
   * @model
   */
  BookCategory getCategory();

  /**
   * @model opposite="books"
   */
  Writer getAuthor();
}
```

As a third option, when augmenting existing XML-based applications with EMF where an XSD schema describing the model is already available, you might find it more convenient to build an Ecore using the existing XSD.

Finally, and most directly, the document can be specified in its native XMI format. While this can this be a useful alternative for the XML-inclined developer, this technique requires a significantly higher level of familiarity with EMF modeling concepts and is outside the scope of what we'll cover in this article.

To help understand Ecore better, the library example Ecore XMI is shown in Listing 1.

Figure 2 | A UML class diagram for the library example



Listing 2 Model objectives via convenience methods

```
// create a library
Library library =
LibraryFactory.eINSTANCE.createLibrary();
library.setName("The Eclipse Library");
Writer author =
LibraryFactory.eINSTANCE.createWriter();
author.setName("Ed Merks");
Book book = LibraryFactory.eINSTANCE.createBook();
book.setTitle("Eclipse Modeling Framework");
book.setAuthor(author);

// add the book to the library
library.getBooks().add(book);
```

Listing 3 Printing out the title or name of a book

```
public class TitleAdapter implements Adapter {
public void notifyChanged(Notification notification) {
  switch(notification.getFeatureID(Book.class)) {
      case LibraryPackage.BOOK_TITLE:
        System.out.println("Title was set to: " +
      notification.getNewValue());
        break;
      case LibraryPackage.BOOK_NAME:
        System.out.println("Name was set to: " +
      notification.getNewValue());
         break;
      default:
         System.out.println
       ("Something else happened!");
    }
  }
}
.
// add the title adapter to the list of adapters
myBook.eAdapters().add(new TitleAdapter());
```

GENERATING CODE

Regardless of whether the EMF Ecore model was created based on UML, Java, or XSD, you can generate an EMF generation model (GenModel) using the EMF project and file wizards. GenModel itself is an EMF model that describes the specification for code generation.

While the majority of the information is shared with the Ecore, GenModel contains advanced parameters that can be adjusted for custom code-generation scenarios. However, for standard model implementations, GenModel is more appropriate. Using GenModel, EMF can generate model implementation code, additional edit framework code, additional sample model editor code and model test code.

It's fairly easy to create EMF model objects based on a model. After the code is generated to manipulate the model, `EFactory` must be obtained to create model objects via convenience methods, as shown in Listing 2. The model objects should not be interfaces or abstract.

Every EMF object is also a `Notifier` that implements the famous Observer design pattern for EMF objects. Each EMF object also has a list of `eAdapters()` that can be thought of as a list of observers (or adapters). Listing 3 shows how to print out the title or name of a book when it is set.

By default, EMF provides an XML/XMI serialization mechanism for EMF models. Listing 4 creates an EMF resource set and displays it to standard output. The output in this case can easily be redirected to any valid `OutputStream`. The serialized end result is shown in Listing 5.

One of the special features of EMF is its ability to reflectively manipulate EMF model objects. All generated model objects implement the `EObject` interface, which provides access to the metadata of the instance object in question.

For example, you can use the reflective API methods `eGet()` and `eSet()` to manipulate object data. This concept of reflective manipulation is powerful because it allows you to dynamically modify the instance data of the model as well as the structure of the EMF model during runtime. Listing 6 uses reflective manipulation to set the name of a library.

The `eAdapters()` mechanism offers a way to listen to model changes. The example in Listing 7 listens to all notifications in our library model, and Listing 8 shows sample output that this content adapter would produce via standard output. You can record models using the `ChangeRecorder` class, which lets you determine the changes that should be allowed in your model. Listing 9 creates a library, attaches a change recorder, records a

Listing 4 Creating and displaying a EMF resource set

```
// Create a resource set to hold the resources.
ResourceSet resourceSet = new ResourceSetImpl();
// Register the appropriate resource factory to handle all file extentions.
resourceSet.getResourceFactoryRegistry().getExtensionToFactoryMap().put
  (Resource.Factory.Registry.DEFAULT_EXTENSION,
   new XMIResourceFactoryImpl());
// Register the package to ensure it is available during loading.
resourceSet.getPackageRegistry().put
  (LibraryPackage.eNS_URI,
  LibraryPackage.eINSTANCE);

try {
  Resource resource =
resourceSet.createResource(URI.createURI("http:///eclipse.library"));
  Library library = LibraryFactory.eINSTANCE.createLibrary();
  Book book = LibraryFactory.eINSTANCE.createBook();
  book.setTitle("Eclipse Modeling Framework");
  library.getBooks().add(book);

  Writer author = LibraryFactory.eINSTANCE.createWriter();
  author.setName("Ed Merks");
  author.getBooks().add(book);
  library.getWriters().add(author);

  resource.getContents().add(library);
  resource.save(System.out, null);
  }
catch (IOException exception) {
  exception.printStackTrace();
}
```

Listing 5 The end serialized result created by set output

```
<?xml version="1.0" encoding="ASCII"?>
<org.eclipse.example.library:Library xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:org.eclipse.example.library="http:///org/eclipse/
example/library.ecore">
  <writers name="Ed Merks" books="//@books.0"/>
  <books title="Eclipse Modeling Framework"
author="//@writers.0"/>
</org.eclipse.example.library:Library>
```

Listing 6 Setting the library name using reflective means

```
// create a library using the factory
Library library =
LibraryFactory.eINSTANCE.createLibrary();

// reflectively set the name of the library
library.eSet(LibraryPackage.eINSTANCE.getLibrary_Name()
, "The Eclipse Library");
```

few changes to the model, then revokes the changes via the `ChangeDescription` class.

INTRODUCING EMFT

The Eclipse Modeling Framework Technology project was announced recently to address experimental EMF work without disrupting current EMF development.

EMFT projects are intended to add value that is not found in other modeling frameworks. For example, the EMFT Transaction framework allows for a transactional model of resource management in EMF, where framework users can perform operations on EMF resources in a controlled manner. The framework also integrates with EMFT validation (described below) to maintain the semantic integrity of models. The ability to rollback or perform read-only transactions on EMF models is just an example of what this framework can provide.

EMF already provides a validation framework via the `EValidator` API but certain features are lacking. The EMFT Validation framework provides a method by which live or batch constraints on EMF metamodels can be defined and evaluated using a transactional approach. The framework also makes constraint languages interchangeable and extendable by third parties. Currently, Java and the Object Constraint Language (OCL) can make use of the validation framework.

The EMFT Connected Data Objects (CDO) framework is an object-persistence technology that integrates EMF to transparently store objects in a relational database system. By default, EMF persists models to a XMI/XML resource. CDO takes that capability to the next level, allowing users to persist EMF resources to a database. This persistence can be useful in the case of large models that can benefit from database storage.

The EMFT Object Constraint Language framework includes tooling to evaluate OCL 2.0 expressions against EMF models. Using it, models can be queried for select elements. Constraints can also be validated against a model.

The final EMFT project that we'll discuss is the EMFT Query framework, which provides tooling to construct and execute SQL-like query statements on EMF-based models. These query statements offer an easy way to manipulate models based on a query. For example, if the model represented a library, this framework could be used to query the model for all books written by an author named Ed Merks, as shown in Listing 10.

THE GRAPHICAL MODELING FRAMEWORK PROJECT

The Graphical Modeling Framework (GMF) project offers a powerful framework for visualization and graphical

Listing 7 Listening to notifications in the library model

```
public void notifyChanged(Notification notification) {
  super.notifyChanged(notification);
  System.out.println("Old Value: " +
notification.getOldValue());
  System.out.println("New Value: " +
notification.getNewValue());
  System.out.println();
}
…
// add the content adapter to the list of adapters
library.eAdapters().add(new MyContentAdapter());
```

Listing 8 Sample output from the content adapter

```
// a book was added to the library
Old Value: null
New Value:
org.eclipse.example.library.impl.BookImpl@11121f6
(title: Eclipse Modeling Framework, pages: 100,
category: Mystery)

// the library had its name set initially
Old Value: null
New Value: Old Library Name

// the library had its name set again
Old Value: Old Library Name
New Value: New Library Name
```

Listing 9 Creating a library and working with a change recorder

```
Resource resource =
resourceSet.createResource(URI.createURI("http:///My.library"));
Library library = LibraryFactory.eINSTANCE.createLibrary();
library.setName("The Eclipse Library");
resource.getContents().add(library);
List beforeChanges = new ArrayList(resource.getContents());

ChangeRecorder recorder = new ChangeRecorder(resource);

library.setName("Library of Alexandria");
Book book = LibraryFactory.eINSTANCE.createBook();
book.setTitle("My Book");
library.getBooks().add(book);

ChangeDescription description = recorder.endRecording();
// undo the changes so we are back to our pristine model
description.apply();
```

editing using an existing EMF model. In addition to the domain model of the application, GMF requires a few other models to describe and tailor the generated graphical editor.

First, a diagram definition model describes the objects that are to be visible in the editor, along with their appearance. Next, a tooling definition model describes the available tools, such as palette entries, by which the user can interact with the editor. Finally, a mapping definition model relates elements from the domain model to elements of the other models. It then formalizes the mapping between data elements in the domain and how they are affected by the interactions of the user with the editor.

Once all of the models are defined, GMF builds a generation model and generates code that leverages the GMF runtime, which is based on EMF and GEF. The advantages that the GMF runtime has over plain GEF is that it uses a notational model (separation of business logic from diagram logic), a standard UI for graphical editors and EMF technologies such as EMFT-Transaction.

Through a combination of EMF, GMF and OCL, a wide range of advanced capabilities are available to the developer, including constraints, auditing, and even graphical editing of the editor definition models. GMF builds on model-driven development practices by providing the tooling necessary to go from defining a domain model to generating the complete editor necessary to edit that model.

Listing 10 Querying the model to find all books by a specific author

```
// Set up the condition
EObjectCondition condition = new EObjectReferenceValueCondition(
  new EObjectTypeRelationCondition(LibraryPackage.eINSTANCE.getBook()),
        LibraryPackage.eINSTANCE.getBook_Author(),
        new EObjectAttributeValueCondition(
        LibraryPackage.eINSTANCE.getWriter_Name(), new StringValue(“Ed Merks”)));

// Construct the query statement
SELECT statement = new SELECT(
        new FROM(myEMFResource),
        new WHERE(condition));

// Execute and return the query
return statement.execute();
```

FRAMING THE MODEL

The Eclipse Modeling Frameowrk is a powerful and mature framework with tooling that can be valuable whenever you’re manipulating models within Eclipse. EMF, along with the powerful features inherent in its subprojects, can shave countless hours of development time and make our lives just a little bit easier.

Confusion or Distribution?

Regain control of your development environment with Yoxos Eclipse Distribution.

The Eclipse development environment offers access to a powerful ecosystem of open-source plugins. To fully profit from this ecosystem, you need to know which plugins are the right ones for your development, where to find them, when reliable updates are available and how they will work as part of a unified custom platform. Doing these tasks, not to mention the time to simply download all the software, takes valuable time away from coding.

The solution is Yoxos, a managed distribution of Eclipse and many of the most popular open-source plugins plus an install manager to configure your ideal Eclipse installation.

Visit us at **www.yoxos.com** to unleash the power of Eclipse.

The Eclipse distribution

yoxos
eclipse distribution

www.yoxos.com

© 2006 innoopract
All rights reserved

It's an Ecosystem!

We all know that Eclipse is a great open-source community that provides a lot of cool, high-quality projects. Not everyone knows, however, the role the Eclipse Foundation itself plays in the community. We're often asked who the Foundation serves, what we do, and how we are funded. Let's talk about those questions, and explore some of the activities of the Foundation.

The Eclipse Foundation was established in 2004 as the steward of the Eclipse ecosystem. A set of bylaws and governance structure were put in place to create Eclipse as an open, vendor-neutral environment where competing organizations could work together on on a level playing field.

GOVERNANCE

As a not-for-profit organization, the Eclipse Foundation has no shareholders and is not a charity. The Board of Directors has representatives of the Strategic members and elected representatives of the Add-In Provider and Committer communities. The board is responsible for setting the strategy and policy of the Eclipse Foundation. For example, the board approves the charters of each top-level Eclipse project.

Mike Milinkovich, the executive director, is responsible for the operational aspects of running the Eclipse Foundation. In addition to Mike, 11 people provide essential services such as IT infrastructure, development process refinement, IP due diligence and ecosystem development.

All Eclipse projects are currently organized under one of nine top-level projects. Each top-level project, like the Web Tools Platform and the Test & Performance Tools Platform, is responsible for the development, project management and community building of its technology and its subprojects.

The Foundation doesn't have developers on staff nor does it have direct influence over what is produced within the projects themselves. The project communities, not the Foundation, decide what's needed.

Donald Smith *is Director of Ecosystem Development for the Eclipse Foundation.*

FOUNDATION MEMBERS

The Eclipse Foundation is funded through the annual dues of members. There are four kinds of membership:

- *Strategic members* have a strategic interest in the evolution of the Eclipse technology and want to be heavily involved in the ecosystem.
- ***Add-In providers*** are software vendors, systems integrators and other organizations developing tools, frameworks or applications that make use of or are built on the Eclipse platform.
- *Associate members* are media organizations and not-for-profits interested in leveraging and promoting Eclipse.
- The incredibly important *Committer members* are individuals who submit code on Eclipse projects.

FOUNDATION SERVICES

The Foundation provides a number of services to the community. One of the most visible is running the IT infrastructure, including the eclipse.org Web site. This is no easy task when server page hits are approaching a trillion since 2004 and terabytes are downloaded every month.

Although the Foundation does not employ any developers, we support the open-source projects and their Committers in many ways. The primary developers of the software, Committers are the lifeblood of the Eclipse community.

The Foundation helps Committers by providing the CVS and the Bugzilla database, newsgroups and mailing lists. The Foundation also helps refine the Eclipse Development Process, which is used on the 60-plus projects currently under way. The Eclipse Development Process describes the guiding principles, the project life cycle, and the meritocracy system for working with Committers.

A less visible service, but critical to companies using Eclipse technology, is the due diligence provided on the intellectual property of all the Eclipse projects.

Verifying the IP of Eclipse is a technical and administrative challenge, with code and other intellectual property being donated by people and organizations from around the world. All submissions go through a rigorous IP process to give everyone the confidence that they can use and distribute the technology.

Another important mission is to promote Eclipse to the community at large, and work with members to expand the platform. We do that by managing press relations, organizing the annual EclipseCon conference and other seminars, and hosting or participating in other events.

The ecosystem surrounding the Eclipse platform is the result of a community effort, but the Foundation also has an important role to play.

Eclipse Data Visualization

(No Silly Glasses Required)

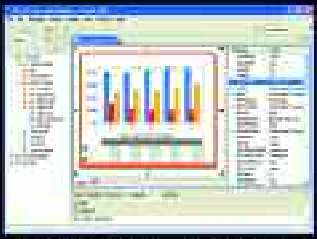


Chart FX for Java Eclipse plug-in

The Leading Charting Solution Now Provides Powerful Data Visualization for Eclipse

The **Chart FX for Java 6.2 Eclipse plug-in** brings enterprise-level data visualization features to the Eclipse IDE. The Designer is integrated into the IDE allowing quick customization of the charts and the required code generation. In addition to a myriad of traditional chart types, the **Chart FX Maps** extension is included to create dynamic, data-driven image maps, such as geographic maps, seating charts or network diagrams, among others. Chart FX for Java 6.2 is available as a Server-side Bean that runs on most popular Java Application Servers. The 100% Java component produces charts in PNG, JPEG, SVG and FLASH formats. The **Chart FX Resource Center** integrates into the Eclipse Help and includes a Programmer's Guide, the Javadoc API and hundreds of samples. This makes Chart FX for Java the most feature-rich, easy-to-use charting tool available for Java development. *Learn more about the seamless integration and powerful features at www.softwarefx.com.*

Chart FX

www.softwarefx.com

New! version 6.2

Now includes Maps!

eclipse READY™
intel® Leap ahead™
Remember when
the sky was the limit?
With Intel® Software Development Products, the Swinburne Center for Astrophysics is showing today's kids a universe filled with unlimited possibilities.

intel® Software

THREADED APPLICATIONS HELP UNRAVEL THE ORIGINS OF THE UNIVERSE.
Take advantage of the power behind multi-core processors by introducing threading to your applications. Threading allows you to use hardware parallelism to improve application speeds. Intel® Software Development Products are on the leading edge of threading technology, giving you the opportunity to discover the performance potential you need. So whether you create applications that model the solar system or enable a gaming system, Intel Software Development Products give your applications the power of parallelism.

Visit www.intel.com/software/products/eclipse for more information.