SYS-CON

WHY 'ECLIPSE DEVELOPER'S JOURNAL' AND WHY NOW? PG 2

# eclipse developer's journal

PREMIER 2005 VOL 1 ISSUE 1

http://eclipse.sys-con.com

# What's New in Eclipse 3.1

# Why "Eclipse Developer's Journal" and Why Now?

## *EDJ's editor-in-chief Introduces world's first independent magazine devoted to the Eclipse ecosystem*

by Bill Dudney

Bill Dudney editor-in-chief of Eclipse Developer's Journal, is a senior Consultant with Object Systems Group. He has been doing Java development since late 1996 after he downloaded his first copy of the JDK. Prior to OSG, Bill worked for InLine Software on the UML bridge that tied UML Models in Rational Rose and later XMI to the InLine suite of tools. Prior to getting hooked on Java he built software on NeXTStep (precursor to Apple's OSX). He has roughly 15 years of distributed software development experience starting at NASA building software to manage the mass properties of the Space Shuttle. You can read his blog at http://jroller.com/page/BillDudney.

When I was first asked to take on the role of editor-in-chief for *Eclipse Developer's Journal (EDJ)* I naturally asked myself this same question. Why do we need a journal dedicated to Eclipse right now?

I believe there are two main answers: 1. the community is large and growing, and 2. the community would benefit from a magazine focused exclusively on the Eclipse ecosystem.

While I searched the Net I found several existing outlets to talk about Eclipse and its related technologies, but none of them are focused exclusively on Eclipse and the community around it. So I said yes and here I am introducing the magazine, which we aim to make the best online resource for Eclipse anywhere on the Web.

To get an idea of how big the community is, I did a bit of poking around and found the following tidbits. Eclipse 3.0 was released on 25 June, 2005 and has been downloaded roughly 1.64 million times. The 3.1 download challenge has hit the 825,00 mark in just over 30 days. That is more than 2 million downloads in 13 months. The 2005 EclipseCon conference was sold out and several new Strategic Developers joined the project. There are currently 28 subprojects with several more in the proposal phase. The bottom line is that the Eclipse ecosystem is huge and growing. The ecosystem will benefit from having an independent magazine devoted to the various aspects of Eclipse and I'm looking forward to EDJ becoming that source.

With that being said I would like to spend the remainder of this introductory editorial on sharing my vision with you on how this thing known as EDJ will start, as well as some of the interesting directions I'd like to take it in the future. And as always I welcome your feedback on what we cover.

The main purpose for *EDJ* is to educate users of Eclipse and developers building plug-ins or Rich Client Platform (RCP) applications. With the help of great columnists and articles we will build out a great information source that you will be able to use to educate yourself on just about anything Eclipse related. In addition to the stuff everyone thinks of as Eclipse (Java Developer Tools, RCP, etc.) I'd also like to spend some significant space getting into the various subprojects going on now. For example on of my personal favorites is the Eclipse Modeling Framework (EMF). I guess I'm just a meta-data geek at heart but the idea of having a whole meta-modeling framework at my disposal is groovy. There are lots of other great projects that I'd like to see covered as well. And finally I'd like to see EDJ become a source of news and information on what is happening in the Eclipse ecosystem.

Thanks for your time and I'm very much looking forward to what we can make EDJ together. )

eclipse developer's journal

**editorial**
**editor-in-chief**
Bill Dudney
**technical editor**
Raymond Camden raymond@sys-con.com
**editor**
Nancy Valentine nancy@sys-con.com
**associate editor**
Seta Papazian seta@sys-con.com
**research editor**
Bahadir Karuv, PhD bahadir@sys-con.com

**production**
**production consultant**
Jim Morgan jim@sys-con.com
**lead designer**
Abraham Addo abraham@sys-con.com
**art director**
Alex Botero alex@sys-con.com
**associate art directors**
Louis F. Cuffari louis@sys-con.com
Tami Beatty tami@sys-con.com
Andrea Boden andrea@sys-con.com

**video production**
**video production**
Frank Moricoo frank@sys-con.com

**contributors to this issue**
Bill Dudney, Ed Burnette, Arthur, Ryman, Jeremy Geelan, Yakov Fain

**president & ceo**
Fuat Kircaali fuat@sys-con.com
**vp, business development**
Grisha Davida grisha@sys-con.com
**group publisher**
Jeremy Geelan jeremy@sys-con.com

**advertising**
**senior vp, sales & marketing**
Carmen Gonzalez carmen@sys-con.com
**vp, sales & marketing**
Miles Silverman miles@sys-con.com
**advertising director**
Robyn Forma robyn@sys-con.com
**advertising manager**
Megan Mussa megan@sys-con.com
**sales & marketing director**
Dennis Leavey dennis@sys-con.com
**associate sales manager**
Kerry Mealia kerry@sys-con.com

**sys-con events**
**president, events**
Grisha Davida grisha@sys-con.com
**national sales manager**
Jim Hanchrow jimh@sys-con.com

**customer relations**
**circulation service coordinators**
Edna Earle Russell edna@sys-con.com
Linda Lipton linda@sys-con.com
**manager, jdj store**
Brunilda Staropoli bruni@sys-con.com

**sys-con.com**
**vp, information systems**
Robert Diamond robert@sys-con.com

**web designers**
Stephen Kilmurray stephen@sys-con.com
Vincent Santaiti vincent@sys-con.com
Shawn Slaney shawn@sys-con.com
**online editor**
Roger Strukhoff roger@sys-con.com

**accounting**
**financial analyst**
Joan LaRose joan@sys-con.com
**accounts payable**
Betty White betty@sys-con.com
**accounts receivable**
Gail Naples gailn@sys-con.com

**Subscribe@sys-con.com**
**Call 1-888-303-5282**

**editorial offices**
**SYS-CON MEDIA**
135 Chestnut Ridge Rd., Montvale, NJ 07645
Telephone: 201 802-3000 Fax: 201 782-9638
ECLIPSE DEVELOPER'S JOURNAL (ISSN #1523-9101) is published monthly (12 times a year) by SYS-CON Publications, Inc.

# What's New in Eclipse 3.1

## *Polishing the apple*

by Ed Burnette

Ed Burnette is the author of the *EclipseIDE Pocket Guide* (to be published later this year by O'Reilly), co-author of Eclipse in Action, and editor of the articles section at eclipse.org. He writes about Eclipse and the Rich Client Platform at his Web site, www.eclipsepowered.org. Ed has programmed everything from multi-user servers to compilers to commercial video games since earning a BS in computer science from North Carolina State University. He is a principal systems developer at SAS, and lives near Research Triangle Park, North Carolina, with his wife, two kids, and a whole bunch of cats.

ed.burnette@sas.com

Since Eclipse's first release in 2001, it has become a popular environment for Java development. In the period between March 10 and May 11, 2005, users downloaded over 17,000 copies of one of the production SDK releases and over 3,500 copies of one of the stable (milestone) SDK builds on average *every day*. A vibrant eco-system of developers, plug-in providers, authors, and bloggers has grown up around it. Eclipse has also gained the backing of the key Java vendors including BEA, Borland, IBM, SAP, and Sybase. Developers like Eclipse because it provides a great platform for building Java applications, and companies like it because it unifies their software tools under one open source umbrella.

In late June of this year, the latest release of the Eclipse Platform, version 3.1, will be available for download from eclipse.org. In this article, I'll highlight some of the more interesting new features it contains. I'll also discuss some of the other Eclipse projects that are releasing new versions at about the same time.

## A New Hope for Developers

One of the major new features of Eclipse 3.1 is full support for the new language constructions in J2SE 5.0 (also called J2SE 1.5 in the old numbering scheme). Generics, annotations, enums, auto boxing, enhanced for loop, etc., – it's all in there, both in the underlying compiler and the user interface and code assistance that Eclipse is known for.

While Eclipse didn't invent the idea of refactoring, it provides one of the most complete implementations. Eclipse 3.1 comes with a number of new and enhanced refactorings, code assistance, and "quick fixes", many in conjunction with its J2SE5 support. For example, you can put your cursor on a conventional *for* loop that iterates over an array (see Figure 1), press Ctrl+1, and Eclipse will offer to convert it to one of the new style *for* loops (see Figure 2).

At the heart of Eclipse's Java support is a fully compliant incremental Java compiler, written in Java and supporting Java language levels 1.3, 1.4, and now 5.0. Having its own compiler brings Eclipse some benefits including fast compilation, smoother debugging and refactoring, and a lot of diagnostic warnings. The compiler has found several uses outside of Eclipse. It's bundled with many popular Linux distributions and commercial applications, and recent versions of Apache Tomcat use it to compile JSPs. It forms the basis of the AspectJ compiler. And I wouldn't be surprised to see the Eclipse compiler used in the recently announced Apache Harmony project as well.

Other usability enhancements make 3.1 more productive. For example, the new release contains a more integrated help system that changes to show help for what you're doing at all times. One of the largest improvements is in the area of Preferences. Addressing a key user request, the Preference dialog now offers the ability to filter by keywords, for example, you can easily find all options having to do with "tabs" by typing that keyword into the filter box. In addition, Web-like navigation has been added to link to related preferences and go forward and backward in the history.

To make preferences easier to find, in Eclipse 3.1 the Preferences dialog can be opened directly from many editors and views through the context menu. For example, if you right-click in the Java editor and select Preferences..., the dialog will appear. Only the options related to Java editing, including those for the text editor that the Java editor inherits, are shown.

Eclipse 3.1 improves its Ant support by including the latest version of Ant, and an Ant script debugger (see Figure 3), plus many editor enhancements. Another welcome addition: the ability to import a project from an Ant build file, and to export and generate a build file from an existing Eclipse project – you can synchronize your CLASSPATH and build.xml with a few clicks. The generated build.xml is simple and clean, with a provision for a build-user.xml that you can override and still keep the benefits of build.xml generation. This is another example of the community in action: the import/export feature is based on the contribution of Richard Höfter, author of the eclipse2ant plugin.

One thing to note is that all these new features don't come with a performance penalty. Eclipse 3.1 is a lot faster and uses far less memory for common operations than version 3.0. Don't believe me? Check out the performance tests results on the download page for any recent build. These improvements are not just for Windows; Mac and Linux users will notice even it even more due to the special attention paid to those platforms. The graphs

don't tell the whole story, however. In normal day-to-day work I've found Eclipse 3.1 to be much snappier than any previous version.

With developers working with ever larger and more complex projects, their IDE needs to keep up. In order to experience and study problems with large workspaces, the Platform team created one consisting of 135 separate projects and 70,000 classes and other resources. Then, using various profiling tools they identified and corrected many bottlenecks, mainly in the area of memory usage and I/O. As a result, Eclipse 3.1 can handle bigger problems in less time than before. Launching the test workspace used to take close to two minutes, but in Eclipse 3.1 it now takes under 10 seconds.

## Return of the Java Client

Java started out on the desktop, and now after a brief vacation on the server side, it's returning to the desktop with a vengeance. The Rich Client Platform (RCP) is helping to spark this renaissance. RCP is a subset of Eclipse that provides a framework for application development. It includes a widget toolkit (SWT), the plug-in loaders, the help system, and other components that you can use in your own programs.

By taking advantage of this free "client middleware," you can focus on your core competencies and reduce your time-to-market. Eclipse's corporate-friendly license (EPL) allows you to reuse the code in your own programs, whether or not they are open source. You can modify and redistribute the code, as long as you return any improvements to the community.

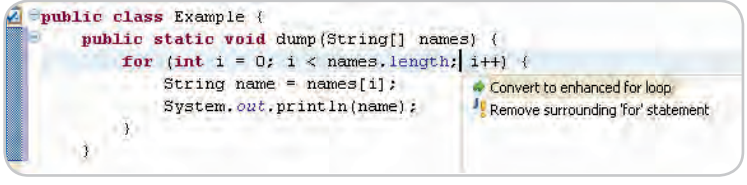


**Figure 1** Press Ctrl+1 to convert a normal array iteration loop.

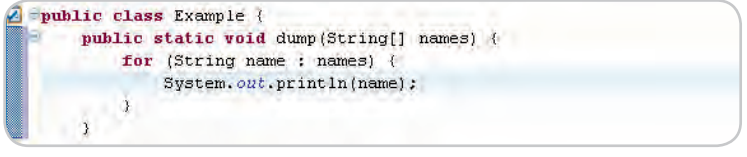


**Figure 2** After conversion, the code uses the enhanced J2SE5 *for* loop.

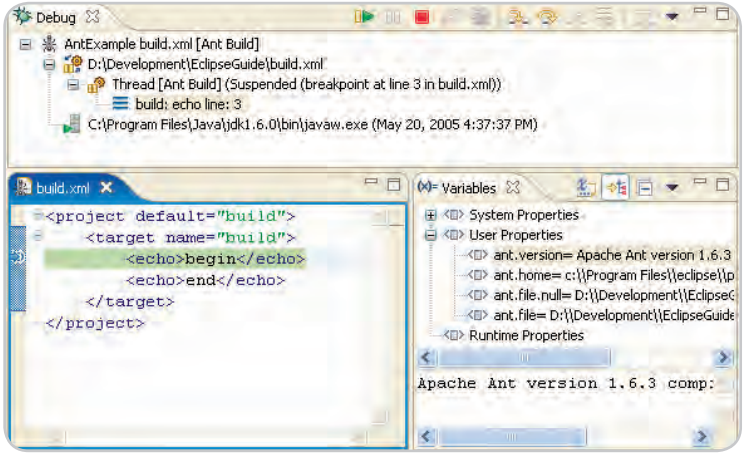


**Figure 3** The Ant debugger lets you single step and examine variables in Ant scripts as you would a Java program.

The biggest change for RCP in Eclipse 3.1 is a set of wizards and editors for creating, building, branding, and deploying RCP applications. To create an RCP application just create a plug-in project, click the checkbox that says "Create an RCP application", select a template, and then click Finish. With a few more clicks you can export the project to create a deployable application. No more trying to figure out plug-in dependencies, tweaking configuration files, and copying plug-ins by hand. All that's handled for you in the new release.

Branded applications are supported through the new Product Configuration editor. You can change the window titles, icons, splash screens, and other branded elements of your program quickly and easily. And with the RCP Delta Pack you can create deployable packages for all supported platforms at the same time (see Figure 4).

RCP applications can take advantage of dynamic plug-ins, that is, plug-ins that come and go at runtime. This provides flexibility to the RCP application delivery model. A large application can be deployed progressively as plug-ins are loaded or on demand when extra functionality is needed. This technology was originally designed for mobile phone provisioning as part of the OSGi Service Platform, and later implemented in Eclipse by the Equinox project team. Eclipse is an active participant in OSGi, and Eclipse 3.1 includes several features slated for version 4 of the OSGi standard.

In one proof-of-concept example shown at EclipseCon, the developers demonstrated a calculator program that started out with only a plus and minus button. Using Eclipse's update manager and dynamic plug-ins, the calculator then downloaded a new plug-in that added a multiply button. All this is done in the running JVM process without a restart.

One of the most frequently asked questions about RCP-based applications is if you can deploy them with Java Web Start. The answer in Eclipse 3.1 is yes. New feature export wizards make this easy; they'll even sign the JARs for you and create a template .jnlp file. In support of Java Web Start, most Eclipse plug-ins have been converted to regular old Java .jar files. Information about extension points, plug-in dependencies, and so on go in manifest files inside the JARs.

In Eclipse 3.1, client developers can take advantage of a slew of UI improvements to make their applications even more functional and better looking than before. For example, SWT includes two new widgets: a Spinner widget for numeric data entry and a Link widget that allows hyperlinks to be included in text labels. A number of other widgets were enhanced.

The Tree widget now supports columns, deprecating the older TableTree widget. This allows a native implementation and helps resolve some of the more subtle problems with the TableTree, including the inability to add an image in the first column. Also the Table widget got a much requested feature: the ability to drag and drop columns to reorder them within the table. Virtual tables with deferred loading are also supported.

The Browser widget continues to get attention as well. This widget wraps the native HTML browser on the current platform (for example, IE on Windows and Safari on the Mac). There have been numerous minor enhancements to the browser including many to its event mechanism. Perhaps the most exciting feature is the ability to execute an arbitrary string of JavaScript within the browser's currently loaded page.

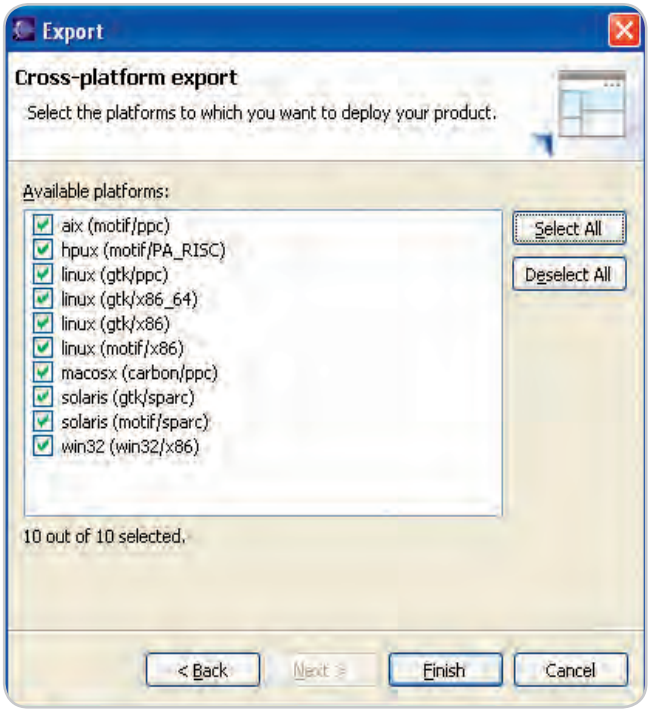


**Figure 4** Create branded, deployable cross-platform products with a few clicks using the new export wizards.

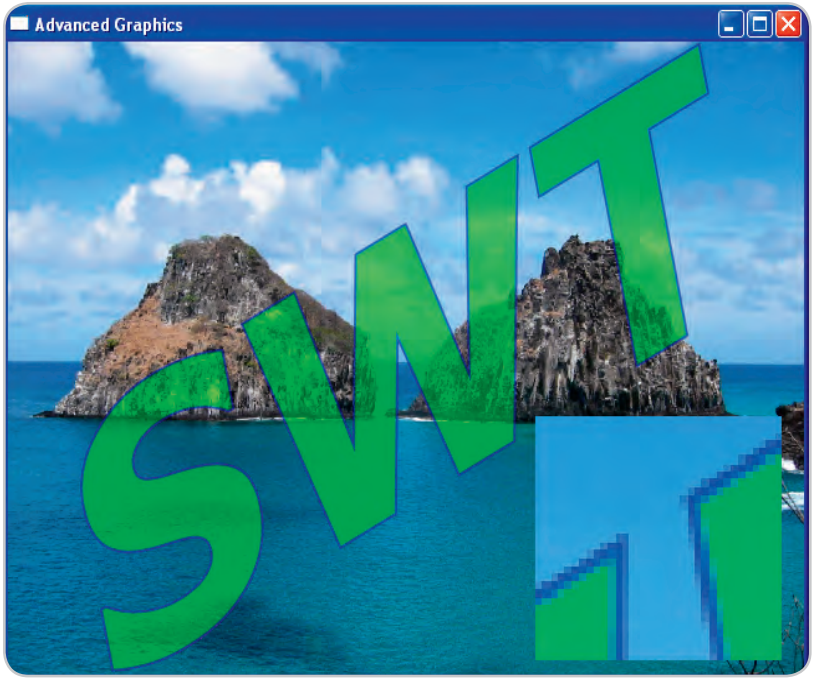


**Figure 5** SWT now supports alpha blending and anti-aliased text (see inset).

Another area that was improved is SWT graphics. Eclipse 3.1 supports alpha-blending, anti-aliasing, paths for geometric shapes and lines, and transformations (see Figure 5). On Windows, using these GC new features takes advantage of the Microsoft GDI+ library (which is included with Windows XP and 2003 but available as a separate download on older systems). On GTK and Motif, the new graphics are implemented with the Cairo graphics library.

## The Community Strikes Back

Community involvement is crucial to the success of Eclipse. One of the things you're expected to do as a good Eclipse community citizen is report your ideas for enhancements along with any bugs you find. Since the source code is available, some take the next step and send code patches as well. Over 7,000 enhancement requests and bug reports have been addressed in release 3.1.

The Eclipse community continues to grow through the addition of new projects. As of this writing, over a dozen new project proposals are pending or have been recently approved. Many of these are "Technology" projects, which are often created by groups of users that band together to fulfill a need. For example, the Mylar project was created at the University of British Columbia to address the problem with information overload by filtering out uninteresting classes and other artifacts while you're programming.

Following on the heels of the 3.1 release of the Eclipse Platform, a number of other Eclipse projects is expected to be released. One of the biggest, the Web Tools Platform project, or WTP for short, is scheduled to release a new version in late July. WTP was initially based on contributions from IBM and ObjectWeb, but many companies and individuals in the community are working on it now, including recent joiner BEA.

The Web Tools Platform currently has two subprojects: Web Standard Tools (WST) and J2EE Standard Tools (JST). WST provides a common infrastructure for Web applications development and provides editors, validators, and document generators for a wide range of Web languages (HTML/XHMTL, CSS, JavaScript, Web services, SQL, XML, XSD, WSDL, etc.). You can also publish and deploy, run and debug, start and stop Web applications on target servers (see Figure 6). WST also includes a TCP/IP Monitor server for debugging HTTP traffic (including SOAP Web services), and a Web services explorer that is very handy for testing. Currently it also has support for relational databases management and queries, though that may be moving to the new Data Tools project soon.

JST extends WST for J2EE applications and servers. Included is a range of tools simplifying development with J2EE APIs including JSP, JCA, JDBC, JTA, JMS, JMX, JNDI, and Web services. It builds on WST to support J2EE servlet engines and EJB containers, including Apache Tomcat, Apache Geronimo, and ObjectWeb Jonas. Server vendors are encouraged to develop adapters for their servers.

> “No more trying to figure out plug-in dependencies, tweaking configuration files, and copying plug-ins by hand; all that is handled for you in the new release”

Another widely anticipated project is the Business Intelligence and Reporting Tools (BIRT) project. BIRT 1.1 is targeted for July, and it will be based on Eclipse 3.1. Currently BIRT includes three components:

- A Report Designer for developing and designing XML report templates
- A Report Engine for generating reports based on the XML template (you can use it standalone or embedded in other applications)
- A Chart Engine for creating charts within BIRT reports or as a standalone API to draw charts in your Swing or SWT applications.

Future plans for BIRT include a Web-based Report Designer.

The Eclipse Test and Performance Tools Platform Project (TPTP), formerly known as Hyades, will launch the 4.0 release in July as well. TPTP delivers components in four areas:

- A platform for building testing tools, with common UI components and standard data models
- Monitoring tools for things like analyzing a Web server
- Test tools, including support for JUnit
- Tracing and profiling tools

TPTP 4.0 delivers better integration with JUnit, new hooks to make it easier to link test cases back with requirements and defects, and usability improvements.

Visual Editor Project (VE) The Visual Editor Project will be releasing version 1.1 approximately two weeks after Eclipse 3.1. Highlights include:

- Support for new SWT controls
- Better support for Swing tables
- Copy/paste support
- Support for editing Eclipse views directly (especially useful for RCP programs)
- Better code generation and reverse parsing (produces code more like what you would write by hand)

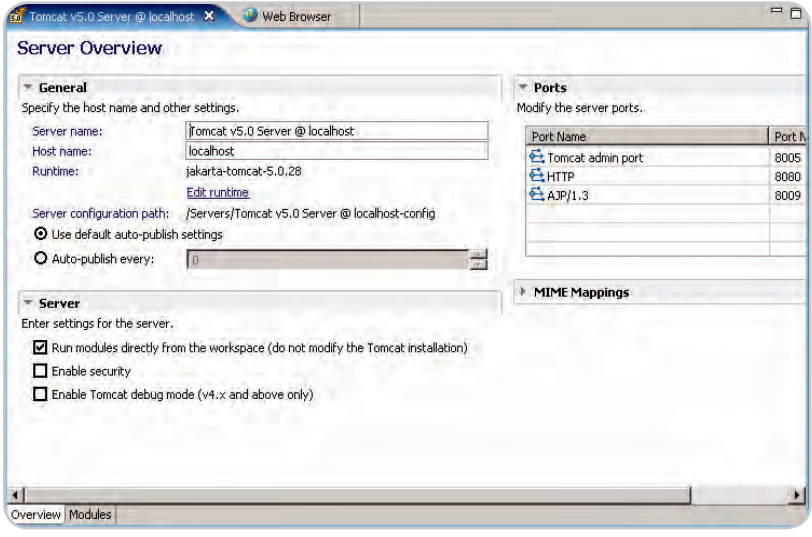


**Figure 6** WST is expected to ship with support for several open source servers like Tomcat

The AspectJ Technology Project will release AspectJ 5.0 soon after Eclipse 3.1 is shipped. The new version includes full support for J2SE5 features, integration of AspectWerkz-style code, better deployment (especially for container-based environments), faster performance, and more comprehensive IDE support. For example, generics are integrated with AOP language features such as join points, pointcuts, advice, and inter-type declarations. Annotations bring AOP to pure Java source files, so you can continue to use your favorite Java compiler and then weave in the aspects in another build step or when classes are loaded. Deployment in J2EE containers is easier and compiling and weaving runs faster and generates better code than before. The class-loading and runtime aspect weaving that made AspectWerkz so convenient should also be supported.

For a gentle introduction to AOP, you may want to check out the Concern Manipulation Environment project (CME) project. It offers powerful code navigation to help you identify cross-cutting aspects in your existing Java code.

## Finale

In four short years since Eclipse exploded onto the scene, it has come to dominate the Java IDE landscape. User groups have sprouted up around the world, and hundreds of books and articles have been written about it (two dozen in Japanese alone!). Eclipse 3.1 is the culmination of a year's worth of development effort on features such as J2SE5 support, performance improvements, and rich clients. If that weren't enough, it will be the base of the next wave of software releases from the Eclipse Foundation and its partners. Whether you're a programmer trying to build the next Killer App or an entrepreneur building a business model on open source, this is an exciting time to be involved with Eclipse.

## Acknowledgments

I wish to thank the many readers of www.eclipse-powered.org who contributed to this article, including Chris Gross, Philippe Ombrédanne, Ng Chin Kiong, Sam Mesh, Bob Foster, David Orme, mgallego, lmandel, and nobodaddy. And a special thanks to Xavier Méhaut, who maintains the Eclipse wiki site, http://eclipse-wiki.info, where we worked on the draft. ❯

# Web Tools Platform:J2EE Development the Eclipse Way

## *The story of WTP 0.7, its scope, design principles, architecture, ecosystem, and plans*

by Arthur Ryman

Arthur Ryman is the leader of the Eclipse WTP Web Standard Tools subproject. He is a software development manager and architecture at the IBM Toronto Laboratory where he has spent the last 10 years working on J2EE development tools. He is an adjunct professor of computer science at York University, a senior member of the IEEE, and a member of the IBM Academy of Technology. He holds a PhD in mathematics from Oxford University. Arthur is a co-author of the book Java Web Services Unleashed, and is currently working on a new book with co-authors Naci Dai and Lawrence Mandel about WTP titled Java Web Application Development with Eclipse.

ryman@ca.ibm.com

The Eclipse Open Source Integrated Development Environment (IDE) (see http://eclipse.org) is rapidly gaining popularity among Java developers primarily because of its excellent Java Development Tools (JDT) and its highly extensible plug-in architecture. Extensibility is, in fact, one of the defining characteristics of Eclipse. As the Eclipse home page says, "Eclipse is a kind of universal tool platform – an open extensible IDE for anything and nothing in particular." Although Eclipse is itself a Java application, all tools, including JDT, are on an equal footing in that they extend the Eclipse platform via well-defined extension points.

Of course, an infinitely extensible, but empty, platform might be interesting to tool vendors, but very boring for developers. Therefore, the initial version of Eclipse came with the JDT and the Plug-in Development Environment (PDE), both examples of how to extend the platform and very useful tools in their own right. JDT supported J2SE development while PDE supported Java-based Eclipse plug-in development. The combination of JDT and PDE fueled the creation of thousands of commercial and Open Source plug-ins for Eclipse, many of which supported J2EE development. For example, IBM released Eclipse-based commercial J2EE products, including WebSphere Studio Application Developer, and Rational Application Developer, while eteration, JBoss, Genuitec, Exadel, and Innoopract among others, released Open Source offerings. However, the profusion of J2EE plug-ins made it difficult for vendors to build on each other and for users to assemble an integrated suite of tools. For example, each J2EE toolset had its own way to support application servers.

As the popularity of Eclipse grew, it became apparent that the next logical step in its evolution was to add platform support for J2EE. This support would provide a common infrastructure for all J2EE plug-ins, with the goal of improving tool integration, reducing plug-in development expense, and simplifying the J2EE development experience for Eclipse users.

In June 2004, based on a proposal from IBM, the Eclipse Management Organization (EMO) agreed to create a new top-level project, the Web Tools Platform (WTP). However, it was believed that for WTP to be truly successful it needed a broad base of vendor support. A search began to engage additional vendors to partner with IBM. WTP was discussed in a BOF session at the first EclipseCon conference held in February 2004, and ObjectWeb agreed to lead the project creation effort. ObjectWeb assembled a set of vendors to join the project and agreed to co-lead the Project Management Committee (PMC). WTP was formally launched in June 2004 based on initial contributions from eteration, Lomboz, and IBM Rational Application Developer.

WTP got further industry endorsement earlier this year when BEA joined the project and announced plans to base a future version of WebLogic Workshop on it. BEA co-leads the PMC along with ObjectWeb. At this year's EclipseCon, Sybase announced the Data Tools

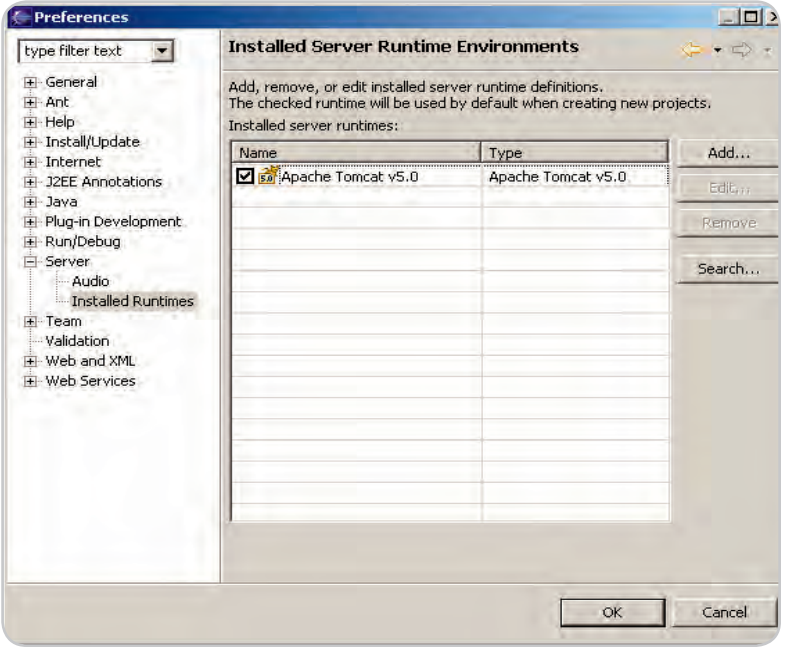


**Figure 1** Server preferences page

Project (DTP), which will add to the data tools in WTP and create a platform layer dedicated to database access. Oracle and Borland also announced Eclipse projects closely related to WTP. With major vendors such as IBM, BEA, Borland, Oracle, and Sybase all co-operating on a shared Open Source tool infrastructure, the center of gravity for J2EE tools has clearly shifted to Eclipse.

WTP 0.7 development is now well underway and has released a series of milestone drivers that can be downloaded from http://eclipse.org/webtools. The final release of WTP 0.7 is on track for a July 2005 delivery. The rest of this article gives you an overview of WTP, its scope, design principles, architecture, ecosystem, and plans.

## A Quick Tour of WTP

One way to understand WTP is that it extends Eclipse along two dimensions, namely execution environments and artifact types. The execution environment dimension defines where code runs. Out-of-the-box, Eclipse lets you develop Java main programs that run in a command shell, applets that run in a Web browser, JUnit tests that run in a JUnit runner, and ANT tasks that run in ANT. WTP extends Eclipse by adding servers in general, and both J2EE and database servers in particular, as new execution environments. In general, you need to install an execution environment, configure it in Eclipse, and associate it with development artifacts that you want to run in it.

The development artifact dimension defines what developers create. Obviously, Eclipse majors in Java source code as a primary development artifact. However other artifacts, such as PDE plug-in manifests and Ant build scripts, are also supported. Each artifact type has associated with it builders, creation wizards, syntax-aware editors, validators, semantic search extensions, and refactoring support. Eclipse users expect editors to provide first-class programmer assistance such as code completion, syntax coloring, error markers, and quick fixes. WTP extends Eclipse with support for the large set of new artifact types encountered in J2EE development. These include HTML, CSS, JavaScript, XHTML, JSP, XML, XSD, WSDL, SQL, and all the J2EE deployment descriptors.

One of the key design goals of WTP is to extend Eclipse seamlessly to support these additional execution environments and artifact types. All of the functions that Eclipse users have come to expect from Java source code should "just work" for the new artifacts. For example, if I select a Java main program, I can Run or Debug it. The same should apply to a JSP. When I select it, the Run command should do something sensible. Specifically for a JSP I expect the Run command to somehow deploy my code into a J2EE server and launch a Web browser with the URL for my JSP. Similarly, the Debug command should run my J2EE server in debug mode and the standard Eclipse Debugger should let me step through my JSP source code. My JSP editor should provide code completion for both JSP tags and inlined Java scriptlets. Furthermore, I expect the code completion for Java scriptlets to work exactly like the code completion for Java source files. I don't want to learn new editing commands simply because I'm editing a new artifact type.

WTP 0.7 achieves many of these goals but there is much work to do to support J2EE fully. Consider the problem of refactoring a J2EE application. An operation as simple as renaming a Java class can have many consequences. If the renaming isn't fully rippled through the application, a runtime error can occur. For example, in addition to references from other Java classes, a Java class can be referenced by JSPs and deployment descriptors. All of these artifacts must be updated to reflect the new name. Suppose the Java class is deployed as a Web Service and that WSDL is generated from it. The WSDL may also need to be regenerated. First-class refactoring of J2EE applications will be an ongoing focus for WTP.

Now let's create a JSP version of "Hello, world." If you'd like to follow along, you'll need to do some setup. Download and install the latest stable driver of WTP from the Web site mentioned above. WTP provides support for many popular commercial and Open Source J2EE servers but doesn't include the runtimes. So you also need to install a server on your machine. For purposes of illustration,

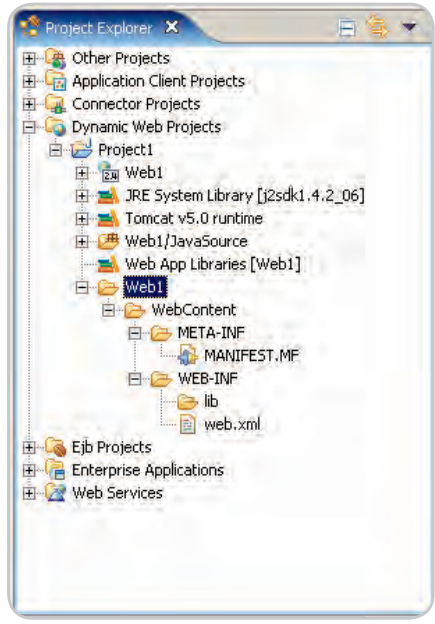


**Figure 2** Project Explorer

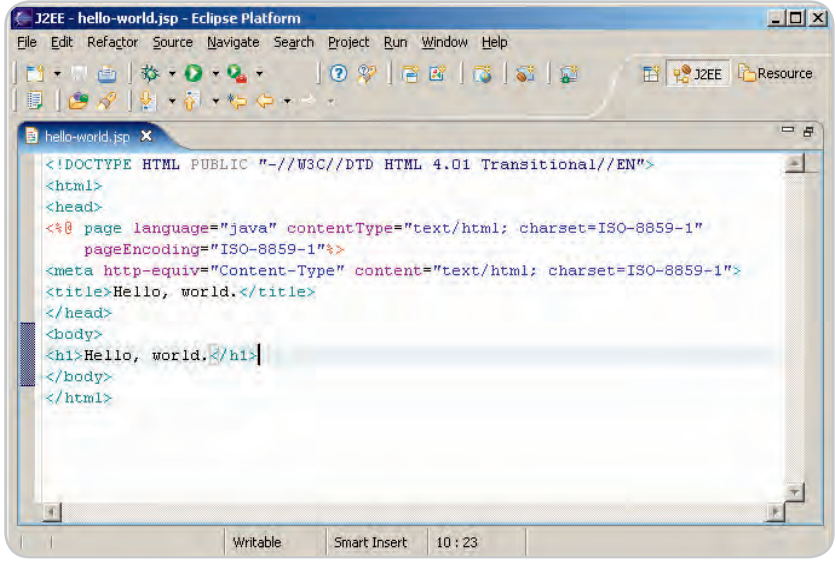


**Figure 3** JSP editor with hello-world.jsp

I'll use Apache Tomcat 5.0.28, which you can obtain from http://jakarta.apache.org/tomcat/. Finally, you'll need a full JDK since JSPs require a Java compiler. I'm using Sun J2SDK 1.4.2_06.

WTP provides a Preference page for Servers. Open the Preference dialog and go to the Server page. Add your Tomcat 5.0 server and configure it to use your JDK (if you use a JRE then JSP compilation will fail). Figure 1 shows the Server Preference page.

Next, create a new Flexible Java Project named Project1 and a new J2EE Web module named Web1 in it. A Flexible Java Project is a J2EE project that can hold several J2EE modules. Figure 2 shows the J2EE Project Explorer after Project1 and Web1 have been created.

Now we're ready to create our JSP. Select the WebContent folder of the Web1 module and use the New File wizard to create a JSP named hello-world.jsp. The wizard fills in the skeleton of a JSP document and opens the file with the JSP editor. The JSP editor has full content assist for HTML and JSP tags, as well as Java scriptlets. Edit the file to say "Hello, world" and save it. Figure 3 shows the JSP editor.

Finally, we're ready to run the JSP. Select hello-world.jsp in the Project Navigator and the Run on Server command from the context menu. You'll be prompted to define the server to be used for the project since this is the first launch. Select the Tomcat server you previously defined and make it the project's default. The Web1 module will be added to the server configuration and the server will start. A Web browser will then be launched with the URL for hello-world.jsp. Figure 4 shows the Web browser with the Web page generated by hello-world.jsp.

Debugging JSPs is also simple. To demonstrate debugging, let's add a Java scriptlet to the JSP. The scriptlet will retrieve a query parameter named "user" and display a welcome message. Listing 1 shows the modified JSP that includes the Java scriptlet.

Set a breakpoint on the line that begins String user = by double-clicking in the left margin. Select the file hello-world.jsp in the Project Explorer and invoke the Debug on Server command from the context menu. The server will restart in debug mode and the Debug Perspective will open with execution halted at the breakpoint. Figure 5 shows the Debug perspective when the JSP is passed the query parameter user=JDJ readers on the URL.

You can now step through Java scriptlets and explore Java variables as usual. In Figure 5, the variable user has been selected in the Variables view that shows its current value JDJ readers. Click the Resume icon to finish processing the JSP. Figure 6 shows the Web browser displaying the resulting Web page.

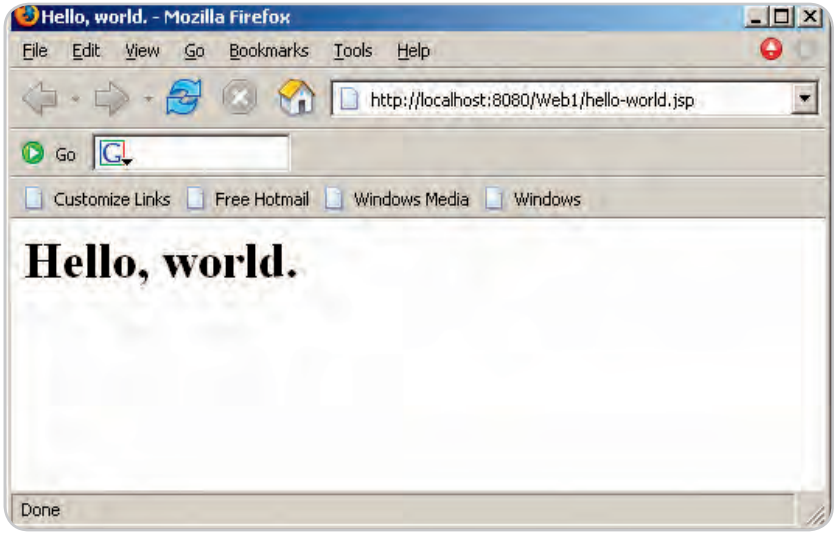


**Figure 4** Web browser with hello-world.jsp

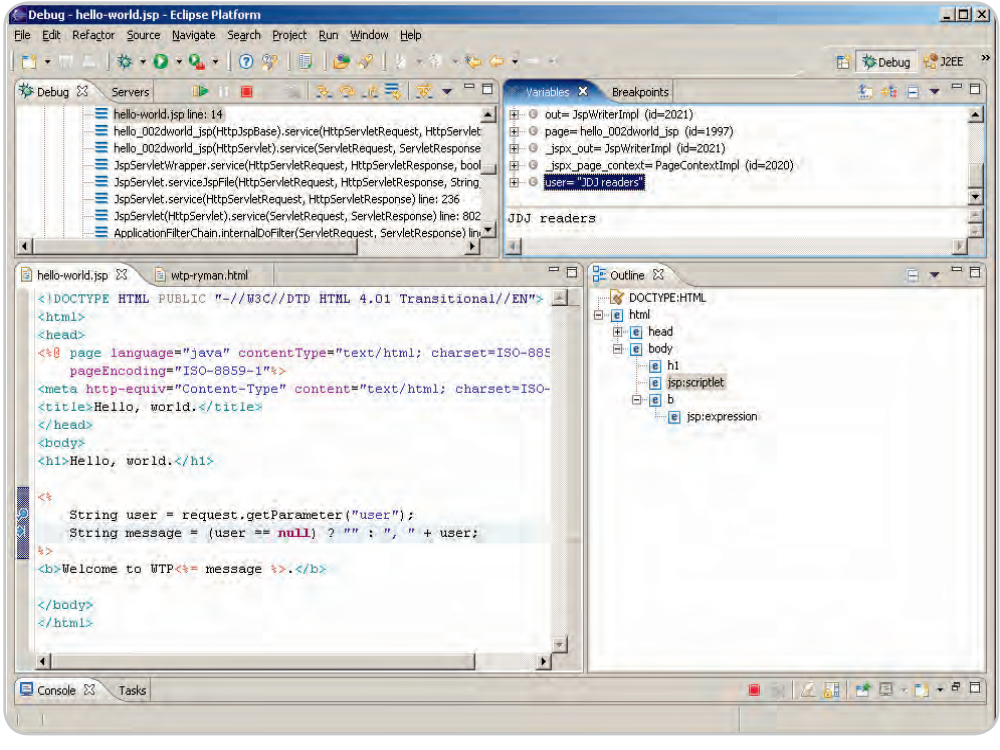


**Figure 5** Debug perspective

## WTP 0.7 Features

The preceding Quick Tour shows a small cross-section of the features available in WTP 0.7. The full set of features in WTP 0.7 includes server, Web, XML, Web Service, J2EE, and data tools. I will now briefly describe these. Consult the WTP Web site for more details.

The server tools let you define and control servers. Servers can be associated with projects, and can be started,

stopped, started in debug mode, and controlled in other ways. Projects can be deployed to servers, or servers can be configured to access your Eclipse workspace content directly. WTP includes server support for Apache Tomcat, Apache Geronimo, and JBoss, as well as many other popular commercial and Open Source J2EE application servers. The server tools include an extension point so that new server types can be easily supported. You can add a new server type by providing either a simple XML configuration file or a Java plug-in.

The Web tools let you create static Web pages based on HTML, XHTML, CSS, and JavaScript. The Web tools include source editors that are based on the WTP Structured Source Editor (SSE) Framework. WTP Web editors provide content assist, syntax highlighting, validation, and other standard Eclipse editor functions. The Web tools also include an embedded Web browser and a TCP/IP monitor that's very handy for debugging HTTP traffic.

The XML tools include source editors for XML, DTD, and XSD that are based on the SSE Framework. Besides source editing, graphical editing is also provided for XSD. The XML tools also include code generators for creating XML instance documents from DTD or XSD.

The Web Service tools include a WSDL editor, a Web Service Explorer, a Web Services Wizard, and WS-I Test Tools. The WSDL editor includes an SSE-based source editor and a graphical editor. XSD editing is seamlessly integrated with WSDL editing. The Web Service Explorer lets you search and publish to UDDI registries, and dynamically test WSDL-based Web Services. The Web Service Wizard ties together the full development lifecycle. It lets you deploy Java classes as Web Services, generate server and client code from WSDL, and generate test clients, as well as being integrated with the Web Service Explorer for publishing and discovery. The Web Service Interoperability (WS-I) Test Tools let you validate WSDL and SOAP for compliance with the WS-I profiles.

The J2EE tools let you create J2EE projects and artifacts like JSPs, servlets, and EJBs, as well as the J2EE deployment descriptors, and deploy these to app servers. The J2EE tools have an SSE-based JSP source editor and a J2EE Project Navigator that displays J2EE components as objects. This provides a higher-level view of your project resources, for example, by displaying all the files related to an EJB as a single EJB object.

The data tools include support for connecting to JDBC-based databases such as Cloudscape, Derby, and other commercial and Open Source databases, and exploring their tables. The data tools also include an SQL source editor that lets you easily execute SQL statements and view the results.

## The WTP Noosphere

In his essay "Homesteading the Noosphere," Eric Raymond likened Open Source developers to homesteaders who stake out their turf in the sphere of ideas. I will therefore describe the turf that WTP has staked out in the Eclipse noosphere.

WTP components are organized along the lines of open standards. WTP classifies the world of standards along two dimensions – technology and formality.

The technology dimension ranges from neutral standards on one extreme to J2EE standards on the other. Technology neutral standards form the foundation of the Web. In fact, the Web has succeeded because it's defined in terms of formats (such as HTML and XML) and protocols (such as HTTP) that specify how systems interact, but don't specify how systems are implemented. This neutrality has allowed vendors to choose the most appropriate implementation technology and compete on the basis of the quality of their implementations. On the other hand, J2EE standards specify application portability rules for J2EE implementations. Both kinds of standard are essential for the viability of the Web.

The formality dimensions define how the standards are created. At one extreme we have the de jure standards bodies such as ISO and IEEE and at the other we have technologies that aren't associated with any formal standards definition organization, but have become de facto standards because of their popularity. Organizations such as W3C, OASIS, and JCP, which define the standards relevant to WTP, have formal processes and are near the de jure end of the spectrum. Popular Open Source technologies such as Struts and Hibernate are at the de facto end.

Figure 7 shows the world of standards classified along the technology and formality dimensions. The turf of WTP is, in principle, all the important standards that are relevant to Web application development. However, the charter of WTP focuses on the standards that are formally defined by recognized standards-definition organizations, i.e., those that cluster towards the de jure end of the spectrum. WTP consists of two sub-projects, Web Standard Tools (WST) and J2EE Standard Tools (JST) that cover the formally defined Web and J2EE standards.

The reasoning behind this scope is that WTP aims to be a platform that many vendors can build on. The formal standards form the building blocks that most vendors want. Support for this base layer of standards


**Figure 6** Web browser with modified hello-world.jsp

is, in a sense, the "table stakes" of any tool. Vendors can cooperate on this base layer and produce high-quality common components while sharing the development expense. Conversely, by not including the de facto standards, WTP leaves room for vendors to innovate and differentiate. For Open Source to succeed contributors must have a way to generate a profit otherwise they won't be able to continue contributing. We hope that this design will yield an excellent set of core Open Source J2EE tools for users, and a solid platform that supports a thriving aftermarket of extenders.

The standards arena is very active and as existing standards are revised and new standards defined WTP will support them based on their market relevance. There may also be a migration of de facto standards to the de jure quadrants. WTP's charter may expand in the future to include new sub-projects. However, immediately, WTP consists of the WST and JST sub-projects.

## The WTP Ecosystem

WTP has the dual goals of providing both tools for the developer community and a platform for tool vendors to extend. Satisfying the needs of vendors requires that WTP define a set of platform APIs. The significance of a platform API is that it will be preserved in future releases. This means that a plug-in that runs in WTP 0.7 will also run – without recompilation – in future versions of WTP. The stability of platform APIs is key to vendor adoption. Clearly if WTP changed its APIs from release to release, vendors would expend significant effort reacting to the changes, and this would slow the rate at which users and vendors move to new versions of the platform.

WTP relies heavily on the user community for testing, bug reports, and enhancement requests, and the development of the user community is one of our main focuses this year. The WTP Web site has tutorials, articles, presentations, and event information. WTP will be well represented on the conference circuit this year. Look for upcoming WTP presentations at events such as EclipseWorld, JavaOne, and the Colorado Software Summit. There are also a couple of WTP books in the works. A thriving user community is a magnet for vendors. As the WTP user community grows so will the number of tools built on it.

Finally, WTP has a role to play in education. Since WTP is free Open Source and supports industry standards, it's an ideal learning tool for the coming generation of J2EE developers. I hope to see universities, community colleges, and even high schools use it for teaching.

The WTP contributor community is drawn from both vendors and users. There are many ways to contribute. You can start by downloading WTP, kicking the tires, and telling your friends about it. If you find a problem or have an idea, open a Bugzilla report. Monitor the newsgroup, and share your solutions to problems with others. If you can write, submit a tutorial or contribute to the online Help system. If you have fixed a problem, submit a patch. If you have time to work on WTP, check Bugzilla for open problems or look at the WTP Help Wanted page. And after you have established a track record of valuable contributions, you can be voted in as a committer.

## What's Next?

WTP 0.7 is scheduled for release in July 2005. We are planning to follow that with WTP 1.0 later in the year. The focus of WTP 1.0 will be on the further development of platform APIs to enable the first wave of products based on WTP. Following that, WTP 1.5 will be released with Eclipse 3.2 in 2006. Candidate items for WTP 1.5 include support for revisions of major specifications such as J2EE 5.0, SOAP 1.2, and WSDL 2.0, as well as new JSRs and Web Service specifications.

We also expect the shape of WTP to change as new projects emerge and mature at Eclipse. New vendors are joining Eclipse and projects are being created at a rapid clip. For example, the data tools in WTP will move into a new Data Tools Project. Technology projects such as those proposed for EJB 3.0 and JSF will likely move into WTP as they mature.

## A Final Word

Like all Open Source projects, the success of WTP depends on the contributions of an enthusiastic community. The project is still in its formative stage and there's much work to do. The project needs users, testers, writers, developers, speakers, trainers, mentors, evangelists, extenders, distributors, and leaders. If you are interested in J2EE development, then please consider this article as your formal invitation to join the WTP community. ❯

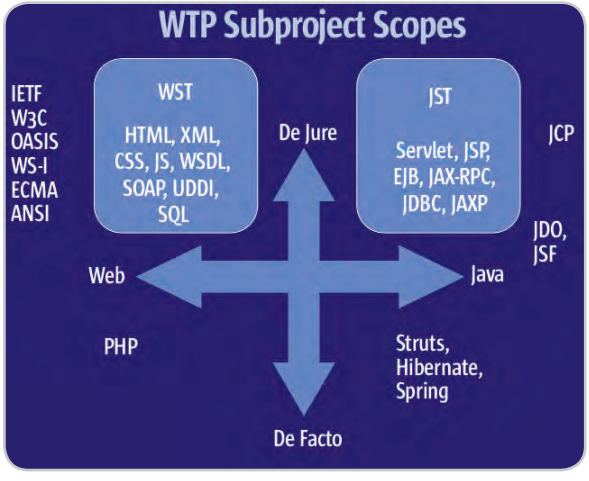


**Figure 7** The scope of WTP

**Listing 1: Hello-world.jsp with Java scriptlet**

```
        <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Hello, world.</title>
</head>
<body>
<h1>Hello, world.</h1>

<%
 String user = request.getParameter("user");
 String message = (user == null) ? "" : ", " + user;
%>
<b>Welcome to WTP<%= message %>.</b>

</body>
</html>
```

# 'Being a Better Platform'

## *Interview with Mike Milinkovich, Executive Director,*

Interview by Jeremy Geelan

Mike, thanks for agreeing to talk to ***JDJ*** and bringing us all up to date with Eclipse since our last interview with you. The best way is probably just to fire off questions and allow you to answer without getting in your way!

**JDJ:** *Overall, how's the independence from IBM going? Since most of Eclipse's committers were IBM employees when you first went independent, how is building a community around Eclipse that is not so IBM focused going?*

**MM:** It's going really well. Frankly, it's going much better and much faster than I had originally anticipated when I started the job. In many ways we have accomplished in the past year what I had expected would take two years or more.

First, the importance of adding companies such as BEA, Borland, and Computer Associates to our board cannot be overstated. Each of these companies competes fiercely with IBM in the marketplace. Each is making million dollar plus investments in Eclipse ($250,000 per year in dues, plus a minimum of eight developers). Each did their own analysis as to whether the Eclipse Foundation was truly independent. And each joined.

Second, the number of projects led by non-IBMers has increased dramatically over the past year. In terms of top-level projects, we have gone from three projects in which two were led by IBM to a total of eight projects with two still being led by IBM.

Third, the number of committers working on Eclipse projects who are IBM employees has steadily dropped over the past year from roughly 75 percent to just over 50 percent. The number will soon drop below 50 percent. This decrease has been mostly the result of increasing the total number of committers. We certainly do not turn away good people from IBM!

In fact, I would also like to recognize the investment that IBM has made in Eclipse. They started this adventure and their continuing investment remains impressive. I also think that the Java community as a whole should recognize the wisdom and strategic thinking shown by IBM in working to establish the Eclipse Foundation as a separate entity. IBM has truly demonstrated how to create a community.

Eclipse *is* truly really completely fully and utterly independent. Anyone who says otherwise has an agenda.

**JDJ:** *How has the assimilation of so many new strategic developers after EclipseCon gone? Is the "plumbing" in place to handle this influx? Are you making adjustments to better handle large groups of people coming on board?*

**MM:** I don't want to sugar coat things. We are having our growing pains as we start up all of these new projects. We need to help get these projects off to a good start and our processes and people are stretched in doing so. However, it's not just the new strategic developers who are causing growth. We have been receiving project proposals from many different directions.

The reaction from the Eclipse community has been outstanding. People with experience within the Eclipse community are stepping up to help us refine our processes and work with the new project leaders and committers to help them get started.

Back in May we had our latest round of Eclipse Council meetings and it was by far the best set of meetings we've had. We were actually debating hard topics like what does it mean for a project to achieve "Eclipse quality," how can we work toward being an even more stable and predictable open source community to further encourage commercial adoption, and what are the tangible things that existing projects can do to help new projects get started. I think it's a very good sign when our community leaders are constructively discussing the tough issues.

**JDJ:** *How much of the themes and priorities specified on your Web site –*www.eclipse.org/org/councils/themes.html *– were you able to accomplish in the 3.1 release?*

**MM:** All of them. None of them.

The themes are not in-

Jeremy Geelan is group publisher of SYS-CON Media and is responsible for the development of new titles and technology portals for the firm. He regularly represents SYS-CON at conferences and trade shows, speaking to technology audiences both in North America and overseas.

jeremy@sys-con.com

> "Certainly at Eclipse we will never stop working on 'Being a Better Platform.' We take the construction, stability, and evolution of the Eclipse APIs very seriously"

> If you are a developer interested in desktop Java development and you have not yet looked at the Eclipse Rich Client Platform, you really need to"

tended to be deliverables. They are intended to provide high-level focus to many different projects at Eclipse to concentrate on similar topic areas and help our entire community move forward. I predict that we will be working on some of those themes for years to come. Certainly at Eclipse we will never step working on "Being a Better Platform." We take the construction, stability, and evolution of the Eclipse APIs very seriously. We want developers to feel comfortable relying on our interfaces.

There are a couple of things about the Eclipse community that are very unique in the open source world that bear repeating.

First, the community is committed to quality and stability. The committers on our projects work very hard to maintain API stability from release to release. Anyone who has ever done this knows that achieving such a goal is a non-trivial effort. But the community honestly believes that API stability is one of the key elements that has made Eclipse so successful.

The second is that we are very focused on predictability. We want our technologies to be adopted commercially. If you want companies to bet their own product plans on an open source project, you need to deliver on schedule. If you look at the Eclipse Platform project, the 3.1 release will be the fourth release in a row where they have hit their dates. If anyone wonders if you can deliver open source projects with several million lines of code on schedule using developers from multiple companies at multiple sites, Eclipse is the project that erases any doubts that it can be done. It is a real testament to the amazing people involved in the project. Pound for pound, I think that Eclipse has some of the best software engineers on the planet.

**JDJ:** *Will the WTP be included with Eclipse by default when it's done?*

**MM:** I don't know. We are still discussing cross-project packaging options within the community.

**JDJ:** *Tell* JDJ*'s readers about the RCP initiative. How has the uptake of the RCP been going? Are you seeing a lot of interest?*

**MM:** If you are a developer interested in desktop Java development and you have not yet looked at the Eclipse Rich Client Platform, you really need to. Eclipse RCP offers a great platform for developing managed applications with a rich user experience. It's doing more to generate interest in Java on the desktop than any other initiative.

We have seen enormous uptake of RCP. I wish we could talk about all of the projects we've seen. One of the most intriguing is the NASA Space Mission Control used for mission planning for the Mars Rover. The most ambitious is the IBM Workplace Client initiative. Macromedia's recent announcement states that they will be building their next-generation Rich Internet Application (RIA) environment on Eclipse RCP.

Eclipse RCP offers a wide variety of features for the application developer. First, it offers a semantically complete and durable component model based on the OSGi bundles standard. The Workbench provides an application shell that developers can plug perspectives, editors, and views into. There are facilities and frameworks for building editors, help facilities, welcome pages, and defining resources.

The best part about the RCP is that it is very real
and it is here now. Developers who are interested in
using Java on the desktop don't have to wait for Mustang. Developers who are concerned about platform lock-in don't have to wait until whenever Longhorn ships. RCP enables multi-platform rich client application development *now*.

RCP has been around for about a year. The big news with 3.1 is really the improvement in tools. The Eclipse Visual Editor now supports the ability to create Eclipse editors using SWT. The Plug-in Development Environment (PDE) has been improved to make it easier to create and deploy Eclipse rich-client applications.

And in case you're wondering, RCP is not just about SWT. You can build RCP applications that utilize Swing. For the Eclipse community, the Swing versus SWT debate is uninteresting. This is not a religious debate. It's about picking the best tool for the job at hand. For those enterprise developers and ISVs who want to build applications with a native look and feel, SWT is a great framework. For those who are less concerned about platform fidelity, we support Swing.

**JDJ:** *With the goal of building out tools for the entire software life cycle, where will the commercial vendors be left to compete? Is there a tension between having the whole life cycle tooled and the vendors that are strategic developers currently competing on that level?*

**MM:** Eclipse is not about supplanting commercial opportunity. It's about creating it.

It's a common misconception that Eclipse is about building tools. That is not the primary focus of our community.

The main objective of Eclipse is to create a universal development platform made up of frameworks and well-constructed APIs. Then we provide exemplary and extensible tools to demonstrate the use of the frameworks. But the point of this

work is to build a platform on which vendors can implement their products. The tools are extensible so they can be further customized to meet the specific needs of commercial platforms.

Now many developers are perfectly happy to construct their toolset on top of Eclipse and open source plug-ins. In the case of our Java development tools, the adoption rate is impressive. But the majority of enterprise development shops are going to be looking for commercially supported and tested tool chains.

**JDJ:** *Turning to Swing/SWT interoperability, can you describe where it is and where it's going?*

**MM:** I really haven't seen a lot of new Swing/SWT interoperability features in Eclipse 3.1. Most of the feedback from our community has been that what we already have is pretty great.

**JDJ:** *Speaking of SWT, do you still think it's worthwhile to try to get cross-platform consistency from native code? Given that all this work has already been done for Swing, do you feel that SWT is still a good bet? Is it wise to repeat work that has been done for so many years in the Java code base?*

**MM:** Of course it is. Java is a great programming language. But the idea that Java should be constrained to only fit one style of application and one way of doing things is just lame. SWT has never been about supplanting Swing. It has been about providing a realistic and high-performance alternative for people whose application requirements demand it.

We believe that there are probably many more Swing applications being built with Eclipse than any other toolset.

The comment about "repeating work" is inaccurate. Swing continues to get better at emulating platforms, which is fundamentally different than the SWT strategy. We are not repeating work. We are implementing an alternative strategy for implementing GUIs with Java.

I believe that the competition from SWT has done more than any other factor to improve Swing. Competition is good. Choice is good. SWT is here to stay.

**JDJ:** *Now that the RCP is well established with the release of 3.1, what direction will Eclipse be headed in the 4.0 time frame?*

**MM:** Instead of talking about a specific release, let me talk about what we want to focus on in the next 12–18 months. A lot of this will be driven by the themes and priorities in our development roadmap. Some of the highlights will include:

1. Continue the evolution and adoption of RCP. We believe the technology around RCP is mature and real now. The focus needs to be on developing frameworks for RCP and having applications adopt RCP as their development platform.
2. Expect to see a lot of work and emphasis on providing tools and frameworks for the embedded developer. We have just launched a new Device Software Development Platform project and I think you'll see some great stuff coming from it over the next 12–18 months.
3. We will continue to extend the set of projects across the software development life cycle
4. Finally, all of the projects are focused on ensuring their tools and frameworks can scale to meet the usability and performance challenges of enterprise development, including support for lots of code and lots of developers.

**JDJ:** *Thanks, Mike, for talking with* JDJ.

**Mike Milinkovich**
*Executive Director*
Eclipse Foundation


# Moving to Eclipse

by Yakov Fain

Yakov Fain is a J2EE architect and creator of seminars "Weekend With Experts" (see http://www.weekend-withexperts.com ). He is the author of the best selling book "The Java Tutorial for the Real World," an e-book "Java Programming for Kids, Parents and Grandparents". Fain also authored several chapters for "Java 2 Enterprise Edition 1.4 Bible." On Sundays, Yakov teaches Java (see http://www.smart-dataprocessing.com ).

yakovfain@sys-con.com

Programmers usually work in a so-called Integrated Development Environment (IDE). You can write, compile and run programs there. An IDE also has a Help thingy that describes all elements of the language, and makes it easier to find and fix errors in your programs. While some IDE programs are expensive, there is an excellent free IDE called Eclipse. You can download it from the Web site www.eclipse.org. I'll help you to download and install Eclipse IDE on your computer, create a project there called Hello World, and after this we'll be creating all our programs there.

Make yourself comfortable in Eclipse - it's an excellent tool that many professional Java programmers use.

## Installing Eclipse

Open the Web page www.eclipse.org and click on the Download menu on the left (http). Click on the link Main Eclipse Download Site and select the version of Eclipse you want to download. They usually have one latest release and several stable builds. The latest release is an officially released product. Even though stable builds may have more features, they still may have some minor problems. At the time of this writing the latest stable build is 3.0M8.

Select this build and you'll see the window shown in Figure 1 .

Click on the link (http) next to the word Windows, Mac, or Linux depending on your computer, and download the file with this long name that ends with .zip to any folder on your disk.

Now you just have to unzip this file into your c: drive. If you already have the program WinZip installed on your computer, *right-click* on this file and select the WinZip from the menu and the option *Extract To*. If you have room on your c: drive, press the button *Extract*, otherwise select another disk that has more space available.

Files with the name suffix *.zip* are archives, and they contain many other files inside. *To unzip the file* means *to extract* the content of this archive on the disk. The most popular archive program is called WinZip and you can download its *trial version* at

Installation of Eclipse is complete! For your convenience, create the shortcut for Eclipse on your desktop. Right-click on the Windows desktop, then press New, Shortcut, Browse, and select the file eclipse.exe in the folder c:\eclipse. To start the program, double-click on the blue icon Eclipse, and you'll see the first Welcome screen (this screen is changing slightly with each Eclipse build) as shown in Figure 2.

If your screen looks different, proceed to so-called Workbench, which is the working area for your Java projects.

## Getting Started with Eclipse

In this section I'll show you how you can quickly create and run Java programs in Eclipse. You can also find a nice

tutorial under the menus Help, Help Contents, and Java Development User Guide.

To start working on a program you'll need to create a new project. A simple project like our HelloWorld will have just one file - HelloWorld.java. Pretty soon we'll create more advanced projects that will consist of several files.

To create a brand new project in Eclipse just click on the menus File, New, Project, and then press the button Next on the New Project window. Now you'll need to enter the name of your new project, for example, My First Project.

Look at the grayed out box Directory. It tells you where the files of this project will be located on the disk. Eclipse has a special folder workspace, where it keeps all files for your projects. Later on, you can create separate projects for a calculator program, a Tic-Tac-Toe game, and other programs.

Eclipse workbench has several smaller areas called perspectives which are different views of your projects.

If you click on the little plus sign by My First Project, it'll expand showing you an item Java Run-time Environment

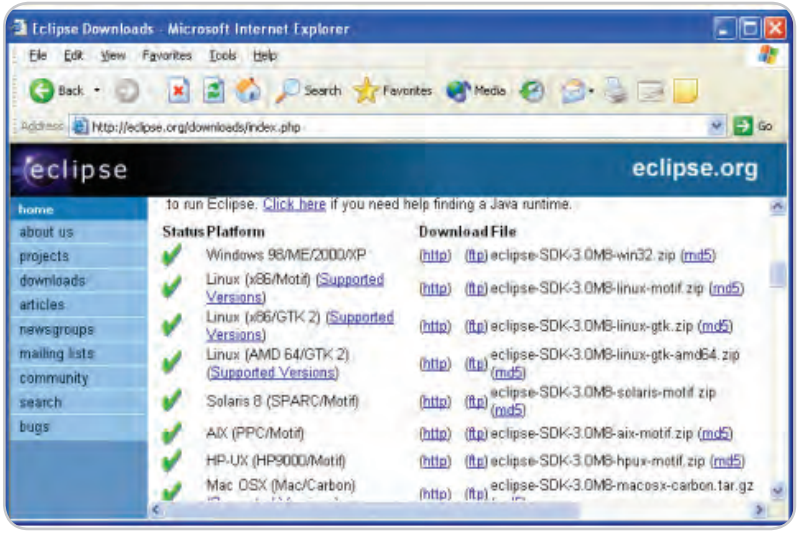


Figure 1

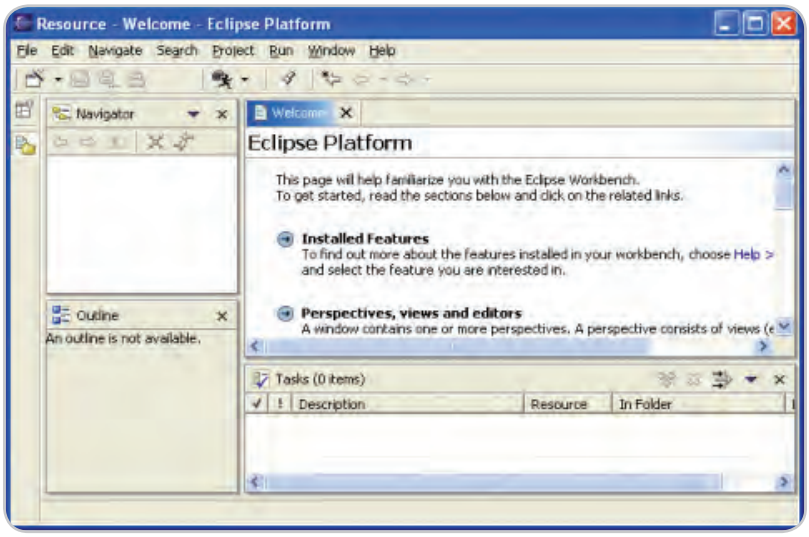


Figure 2

(JRE) System Library which is a part of the project. If for any reason you do not see JRE there, click on the menus Windows, Preferences, Java, Editor, Installed JREs, Add, and, using the button Browse find the folder where you have installed Java, for example c:\j2sdk1.5.0.

## Creating Programs in Eclipse

Let's re-create the HelloWorld program from Chapter 1 of my e-book *Java Programming for Kids, Parents and Grandparents* in Eclipse. Java programs are classes that represent objects from real life.

To create a class in Eclipse select the menus File, New, Class and enter HelloWorld in the field Name. Also, in the section Which methods stubs you would like to create, check off the box.

```
public static void main(String[] args)
```

Press the button Finish, and you'll see that Eclipse created for you the class HelloWorld. It placed program comments (the text between /* and */) on top - you should change them to describe your class. After the comments you'll find the code of the class HelloWorld with an empty method main(). The word method means action. To run a Java class as a program, this class must have a method called main().

```
public class Helloworld {
   public static void main (string [] args)  {
   }
}
```

To complete our program, place the cursor after the curly brace in the line with main, push the button Enter and type the following on the new line:

```
System.out.println("Hello World");
```

To save the program on disk and compile it, just press at the same time two buttons on your keyboard: Ctrl-S. If you did not make any syntax errors, you won't see any messages - the program is compiled. But let's make an error on purpose to see what's going to happen. Erase the last curly brace and hit Ctrl-S again. Eclipse will display the Unmatched Brace error in the tasks perspective, and also it will place a red mark by the line that has a problem.

As your projects become larger, they'll have several files and compiler may generate more than one error. You can quickly find (not fix though) the problematic lines by double-clicking on the error message in the tasks perspective. Let's put the curly brace back and hit Ctrl-S again - voila, the error message is gone!

## Running HelloWorld in Eclipse

Our simple program is a one-class project. But pretty soon your projects will have several Java classes. That's why before running the project for the first time, you need to tell Eclipse which class in this project is the main one.

Select the menu Run, then Run...(make sure that Java

Application is selected in the top left corner), and enter the names of the project and the main class (see Figure 3).

Now press the button Run to start the program. It will print the words Hello World in the console view.

You can run this project by selecting the menus Run, Run Last Launched or by pressing the buttons Ctrl-F11 on the keyboard.

### How HelloWorld Works?

Let's start learning what's actually happening in the program HelloWorld.

The class HelloWorld has only one method main(), which is an entry point of a Java application (program). You can tell that main is a method, because it has parentheses after the word main. Methods can call (use) other methods, for example our method main() calls the method println() to display the text Hello World on the screen.

Each method starts with a declaration line called a method signature:

```
public static void main(String[] args)
```

This method signature tells us the following:

- Who can access the method - public. The keyword public means that the method main() could be accessed by any other Java class or JVM itself.
- Instructions on how to use it - static. The keyword static means that you don't have to create an instance (a copy ) of HelloWorld object in memory to use this method. We'll talk about instances more in the next chapter.
- Does the method return any data? The keyword void means that the method main() doesn't return any data to the calling program, which is Eclipse in this case. But if for example, a method had to perform some calculations, it could have returned a resulting number to its caller.
- The name of the method is main.
- The list of arguments - some data that could be given to the method - String[] args. In the method main() the String[] args means that this method can receive an array of Strings that represent text data. The values that are being passed to a method are called arguments.

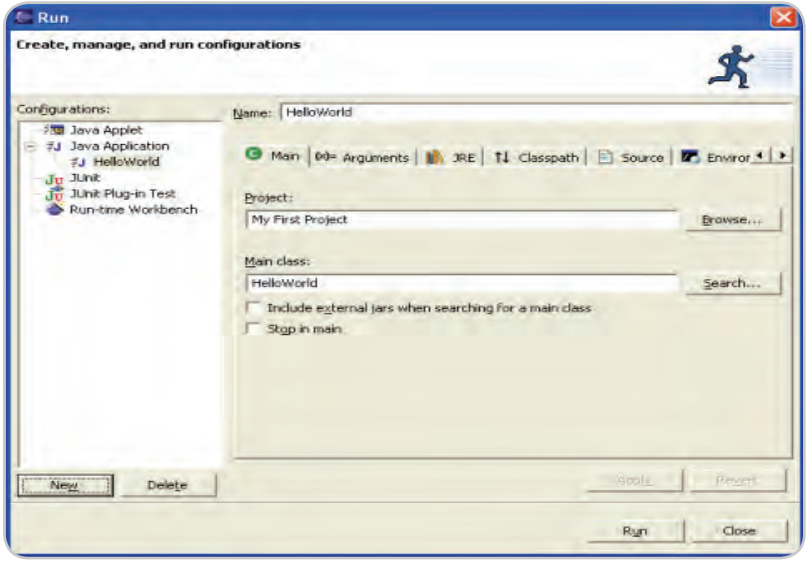


Figure 3

As I said before, you can have a program that consists of several classes, but one of them has the method main(). Java class usually have several methods. For example, a class Game can have the methods startGame(), stopGame(), readScore(), and so on.

The body of our method main()has only one line :

```
System.out.println("Hello World");
```

Every command or a method call must end with a semicolon ;. The method println()knows how to print data on the system console (command window). Method names in Java are always followed by parentheses. If you see a method with empty parentheses, this means that this method does not have any arguments.

The System.out means that the variable out is defined inside the class System that comes with Java. How are you supposed to know that there's something called out in the class System? Eclipse will help you with this. After you type the word System and a dot, Eclipse will show you everything that is available in this class. At any time you can also put a cursor after the dot and press Ctrl-Space to bring up a help box similar to Figure 4.

The out.println() tells us that there is an object represented by a variable out and this "something called out" has a method called println(). The dot between a class and a method name means that this method exists inside this class. Say you have a class PingPongGame that has a method saveScore(). This is how you can call this method for Dave who won three games:

```
PingPongGame.saveScore("Dave", 3);
```

Again, the data between parentheses are called arguments or parameters. These parameters are given to a method for some kind of processing, for example saving data on the disk. The method saveScore() has two arguments -a text string "Dave", and the number 3.

Eclipse will add fun to writing Java programs. The Appendix below has some useful tips and tricks that will speed up your Java programming in this excellent IDE.

## Appendix: Eclipse Tips

Eclipse has many little convenient commands that make Java programming a little easier. I've included some useful Eclipse tips here, but I'm sure you'll find more when you start using this tool.

- If you see a little asterisk in the tab with the class, this means that the class has unsaved code changes.
- Highlight the name of the class or a method that is used in your code and press the button F3 on your keyboard. This will take you to the line where this class or method was declared.
- If some of the lines are marked with red error circles, move the mouse over the circle to see the error text.
- Press Ctrl-F11 to run the last-launched program again.

- Place the cursor after a curly brace and Eclipse will mark the matching brace.
- To change the superclass when creating a new class, click on the button Browse, delete the class java.lang.Object and enter the first letter of the class you'd like to use. You'll see a list of available classes to choose from.
- To copy a class from one package to another, select the class and press Ctrl-C. Select the destination package and press Ctrl-V.
- To rename a class, a variable or a method, right-click on it and select Refactor and Rename from the popup menu. This will rename every occurrence of this name.
- If your project needs some external jars, right-click on the project name, select Properties, Java Build Path and press the button Add External Jars.

## Eclipse Debugger

Rumor has it that about 40 years ago, when computers were large and would not even fit in your room, all of a sudden one of the programs started giving wrong results. All these troubles were caused by a small bug that was sitting inside the computer somewhere in the wires. When people removed the bug, the program started working properly again. Since then, to debug a program means to find out why it does not give the expected results.

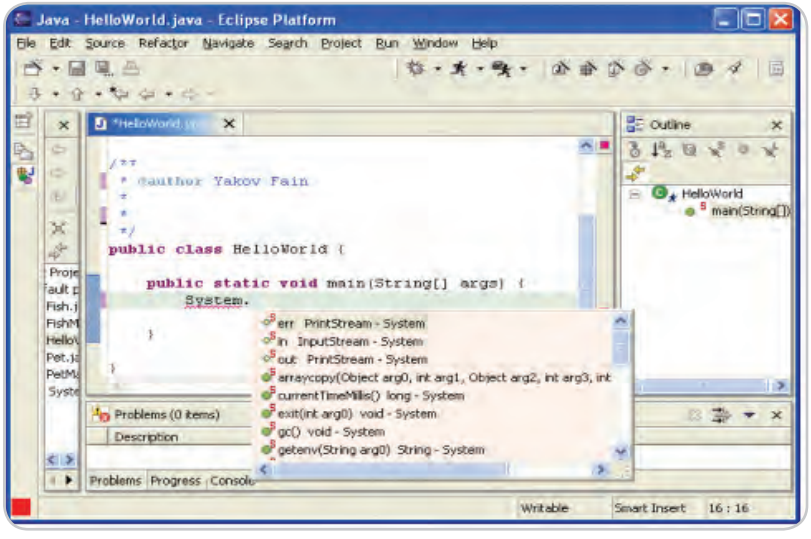


Figure 4

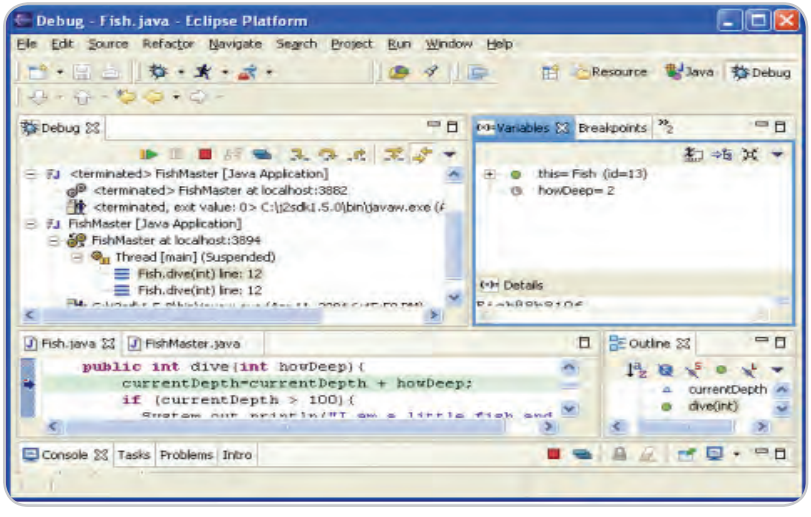


Figure 5

Do not confuse bugs with the compilation errors. Say for example, instead of multiplying the variable by 2, you'll multiply it by 22. This typo will not generate any compilation errors, but the result will be incorrect. Debuggers allow you to step through a running program one line at a time, and you can see and change values of all variables at each point of the program execution.

I'll show you how to use Eclipse debugger using the FishMaster program from Chapter 4 of my e-book *Java Programming for Kids, Parents and Grandparents*.

A breakpoint is a line in the code where you'd like program to pause so you can see/change current values of the variables, and some other run-time information. To set a breakpoint just double click on the gray area to the left of the line where you want a program to stop. Let's do it in the FishMaster class on the line myFish.dive(2). You'll see a round bullet on this line which is a breakpoint. Now, select the menus Run, Debug.... Select the application FishMaster and press the button Debug.

FishMaster will start running in the debug mode, and as soon as the program reaches the line myFish.dive(2), it will stop and will wait for your further instructions.

You will see a window similar to Figure 5.

In the left bottom part of the debug perspective, you see that the line with the breakpoint is highlighted. The blue arrow points at the line that is about to be executed. On the right side (in the Variables view) click on the little plus sign by the variable myFish. Since this variable points at the object Fish, you will see all member variables of this class and their current values, for example currentDepth=20.

The arrows in the top left area allow you to continue execution of the program in different modes. The first bended yellow arrow means step into the method. If you press this arrow (or F5), you'll find yourself inside the method dive(). The window changes and you see the values of the argument howDeep=2 as in the next screenshot. Click on the little plus by the word this to see what are the current values of member variables of this object.

To change the value of the variable, right-click on the variable and enter the new value. This can help when you are not sure why the program does not work correctly and would like to play what if game.

To continue execution one line at a time, keep pressing the next arrow step over (or the button F6).

If you want to continue program in the fast mode, press a small green triangle or the button F8.

To remove the breakpoint just double-click on the little round bullet and it'll disappear. I like using debugger even if my program does not have a bug - it helps me better understand what exactly happens inside the running program.

Where to put a breakpoint? If you have an idea which method gives you problems, put it right before the suspicious line of code. If you are not sure, just put it in the first line of the method main() and slowly walk through the program. ❯

# eclipse developer's journal

**A DIGITAL PUBLICATION**

# Your One-Stop Guide to the Eclipse Ecosystem


The main purpose of *Eclipse Developer's Journal* (*EDJ*) is to educate and inform users of Eclipse and developers building plug-ins or Rich Client Platform (RCP) applications. With the help of great columnists, news you can use, and technical articles, *EDJ* is a great information source that you will be able to use to educate yourself on just about anything Eclipse related

## Subscribe Today!

**FOR YOUR DIGITAL PUBLICATION**

**(AVAILABLE IN DIGITAL FORMAT ONLY)**

### 6 Issues for $19.99

Visit our site at www.eclipse.sys-con.com or call 1-888-303-5282 and subscribe today!

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

http://eclipse.sys-con.com

**SYS-CON.COM**

SYS-CON Media, the world's leading publisher of *i*-technology magazines for developers, software architects, and e-commerce professionals, brings you the most comprehensive coverage of Eclipse

# MyEclipse 4.0 Up On Deck: Exclusive Interview with Maher Masri, President, Genuitec

## *The 4.0 version of MyEclipse is nearing release*

By Bill Dudney

Bill Dudney, editor-in-chief of *Eclipse Developer's Journal*, is a senior consultant with Object Systems Group. He has been doing Java development since late 1996 after he downloaded his first copy of the JDK. Prior to OSG, Bill worked for InLine Software on the UML bridge that tied UML Models in Rational Rose and later XMI to the InLine suite of tools. Prior to getting hooked on Java he built software on NeXTStep (precursor to Apple's OSX). He has roughly 15 years of distributed software development experience starting at NASA building software to manage the mass properties of the Space Shuttle. You can read his blog at http://jroller.com/page/BillDudney.

*Eclipse Developer's Journal:* What is MyEclipse and what's unique about the Enterprise Workbench?

**Maher Masri:** The goal of MyEclipse is to provide a comprehensive, practical, low-cost and fully integrated enterprise development tool suite that anyone can afford to use around the globe. MyEclipse Enterprise Workbench is the first commercial J2EE IDE plugin suite for Eclipse offering an enterprise-class platform and tool suite for developing J2EE and web applications.

My Eclipse is unique in offering:

1) User-centered J2EE and web development features with a primary focus on enterprise development productivity
2) Low cost, subscription-based pricing
3) A corporate commitment to provide the best technical support in the IDE tools industry.

With over 140,000 registered members, our customers are the source of our inspiration and the driver for our quality product and the agile delivery schedule. Our customers play an active role in prioritizing features through forum feedback and interactive polls. It took our customers a little while to realize the flexibility of the low cost, subscription-based pricing model and to appreciate having the freedom to choose an IDE without a lengthy selection process and or having to make a total commitment to a particular tool. Nearly free, the MyEclipse $29.95 annual subscription entitles members full access to all releases and updates as well as free on line support. Customers are free to choose an alternative to MyEclipse at any time, since there's no large expense to justify, which gives us the motivation to continue adding feature and capabilities indefinitely.

**EDJ:** What are your plans for advanced Java5 features?

**MM:** We are really excited about the Java5 support in the recent Eclipse 3.1 release. Our biggest challenge has been, and continues to be, striking the right balance for exploring emerging technologies and standards while addressing immediate customer needs. MyEclipse has provided XDoclet support for well over a year and we are looking very closely at annotation support for EJB and Hibernate capabilities. Our upcoming Hibernate VisualMapper will provide an option to support Java bean mapping through annotations.

**EDJ:** How much of your toolset (JSF, etc) is integrated in the WebTools project (WTP)?

**MM:** We have been involved with Web tools from the very beginning and continue to play an active role on the WTP requirements team. MyEclipse was the first commercial Eclipse-based product to extend early Web tools features and bring enhanced capabilities to our user base. MyEclipse has employed a branch of the WTP based on IBM's initial contribution for over a year and contributed significant improvements to the codebase back to the Web tools project. MyEclipse makes heavy use of the WebTools structured editor and validation frameworks to provide advanced editors for JSP, XML, HTML,

> "Our biggest challenge has been, and continues to be, striking the right balance for exploring emerging technologies and standards while addressing immediate customer needs"

Javascript, and CSS editing. MyEclipse 5.0 planned for late fall '05 will be an upgraded architecture based on an enhanced version of the WTP 1.0 since we expect the WTP APIs and reliability will be stabilized by then.

**EDJ:** What features are supported in the MyUML modeling suite? Which diagrams etc. Does it work on the mac??

**MM:** MyEclipse UML provides a seamless modeling experience with enhanced Java round-trip engineering features between any number of Java projects. We designed MyEclipse UML with the goal of providing the most efficient Agile Modeling environment available. This strategy provides the user complete control over the diagrams in a Java project, their level of detail, and if and when Java code and UML models are synchronized. Users can create class diagrams simply by drag-n-dropping Java classes from Eclipse views onto UML diagrams. The Java forward generation features will automatically recognize your Eclipse Java code format and organization preferences and apply them during the code generation process. MyEclipse UML offers:

- Fully integrated UML tools into MyEclipse platform
- Diagrams include: Use-case, Class, Collaboration, Activity, State, and Deployment
- Uis/Views include: UML Perspective, UML Diagram Editor, UML Outline View, UML PropertiesView
- Full round-trip engineering
- Reverse-engineering support of UML models from Java project source code
  -- Batch process at the following levels: Java project, source folder, package, class
  -- Drag/drop Java classes/interfaces from Eclipse view to any class diagrams
- Consistent styling for classes and Interface details using familiar Eclipse Java class styling
- Forward generated Java code honors your Java coding style preferences
- Export diagrams in popular image formats
- Ability to Print diagrams
- Auto layout option
- Customizable display

While most of MyEclipse's graphical design tools operate on Windows, Linux and Mac, the UML tools only operate on Windows and Linux due to a long-standing Eclipse SWT bug (#67384) that prevents SWT and AWT from interoperating on the Mac. We are actively working with the Eclipse platform team on this issue and hope it will be resolved soon.

**EDJ:** How well does MyEclipse support alternative J2EE technologies such as Hibernate and Spring?

**MM:** MyEclipse has had Hibernate2 support for over a year. The 4.0 release introduces a number of smart editors, wizards, and configuration tools for developing with Hibernate 3, Spring 1.2, and Tapestry 3. We have received very good feedback on these new technologies, contained in our milestone builds, from our user-base. And, while not part of ME 4.0, we are very excited about our upcoming Hibernate VisualMapping Editor that will be released in early fall.

**EDJ:** What demand are you experiencing for Tapestry support?

**MM:** We began supporting Tapestry within the MyEclipse Enterprise Workbench starting with the first milestone release for 4.0. The feedback from tapestry integration is that this addition has been very well received by our user base. Our plans call for tighter integration going forward and we will continue to evaluate priorities based on customer feedback after the final release for MyEclipse 4.0

**EDJ:** What is Genuitec's role on the Eclipse Foundation?

**MM:** Genuitec is a founding member of the Eclipse Foundation and for the last three years has been actively evangelizing Eclipse technology through numerous speaking engagements and magazine articles, including one written in early 2002 that became the genesis of the Eclipse rich client platform. Additionally, Genuitec holds a seat on the Eclipse Foundation Board of Directors where we've recently been re-elected to a second term where we will continue to represent the needs of the Eclipse add-in provider community.

**EDJ:** What are your future plans for MyEclipse?

**MM:** Our customers can count on two things from MyEclipse. First, we will continue to deliver platform independent, standard-based products that will continue to be the best value on the market, at any price. Second, despite tremendous strides thus far, we will continue to add more agile development features with continued focus on improving enterprise development productivity. As such, technology targets for MyEclipse 5.0 and beyond include:

1. Broader life-cycle development support for development management
2. Advanced support for model driven architecture and data modeling
3. Web services and Service Oriented Architecture (SOA) support
4. AJAX support
5. J2EE reporting
6. Advanced rich client design tools ❯

> "MyEclipse UML provides a seamless modeling experience with enhanced Java round-trip engineering features between any number of Java projects"