Vol. 5
Dec'06
www.eclipsemag.net

eclipse MAGAZINE

POWERING THE ECLIPSE ECOSYSTEM

POJO Access Using BIRT

Using BIRT to Access Plain Old Java Objects

Eclipse Power In IBM Workplace/Domino

Preparing for Tomorrow's Challenges

Free Trade with Eclipse

An Introduction to the JFire Open Source ERP Framework

Getting Involved with The Eclipse Community

Join, Participate, Improve the Eclipse Project

Mylar Makes Programmers More Productive

Interview with Mik Kersten, Lead of the Mylar Project

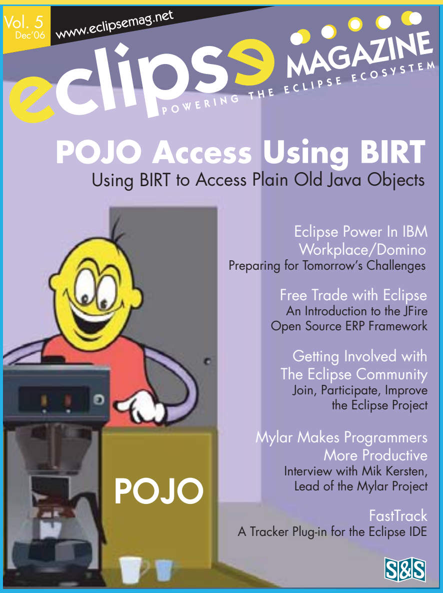
FastTrack

A Tracker Plug-in for the Eclipse IDE

S&S

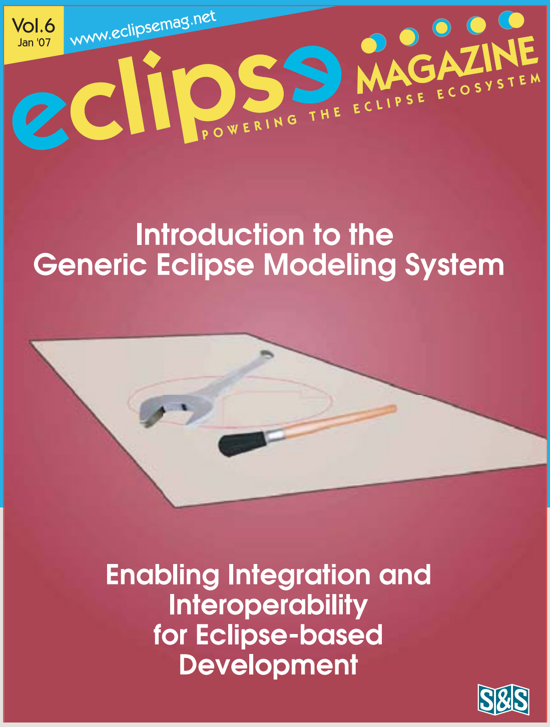
Vol. 5
December 2006

Table of Contents


Vol. 5 Dec'06
www.eclipsemag.net
eclipse MAGAZINE
POWERING THE ECLIPSE ECOSYSTEM
POJO Access Using BIRT
Using BIRT to Access Plain Old Java Objects
Eclipse Power In IBM Workplace/Domino
Preparing for Tomorrow's Challenges
Free Trade with Eclipse
An Introduction to the JFire Open Source ERP Framework
Getting Involved with The Eclipse Community
Join, Participate, Improve the Eclipse Project
Mylar Makes Programmers More Productive
Interview with Mik Kersten, Lead of the Mylar Project
FastTrack
A Tracker Plug-in for the Eclipse IDE
POJO
S&S

Vol. 5
December 2006

Table of Contents


Vol.6
Jan '07
www.eclipsemag.net
eclipse MAGAZINE
POWERING THE ECLIPSE ECOSYSTEM
Introduction to the Generic Eclipse Modeling System
Enabling Integration and Interoperability for Eclipse-based Development
S&S

PREVIEW

COMING UP IN THE JANUARY ISSUE OF ECLIPSE MAGAZINE

- Graphical Model-Driven Engineering (MDE) tools have become extremely popular in the development of applications for a large number domains. In many cases, however, an organization does not have the resources or time available to develop a graphical modeling environment from scratch using the Eclipse Modeling Framework (EMF), Graphical Editor Framework (GEF), or Graphical Modeling Framework (GMF). In other situations, the complexity of the domain limits the feasibility of using a graphical model to describe a domain solution. The Generic Eclipse Modeling System (GEMS), which is part of the Eclipse Generative Modeling Technologies (GMT) project, helps developers rapidly create a graphical modeling tool from a visual language description or metamodel without any coding in third-generation languages.

- Designing, developing, testing and managing business critical applications has become increasingly more complex. To manage this complexity, projects are typically divided into tasks and teams are assigned to execute them. But IT managers also need to ensure all these different teams and team members collaborate effectively as if they were a small team all working in the same room, in the same office, and on the same project. Corona is the right tool to address this IT business problem.

Meeting the Reporting Needs of Real-world Applications

BIRT has been on Eclipse Magazine's radar this year, and for good reason. On the top-half of every enterprise decision maker's priority list this year has been the need for solutions that can meet the reporting needs of real-world applications. BIRT, built from the ground up for web applications, and designed to deliver ease-of-use and flexibility to report developers, who are typically not Java coders, provides that much-needed and sought after functional power. In Volumes 01 through 04 of the magazine, we covered various aspects of Eclipse BIRT — how to harness the enhanced features of Eclipse BIRT 2.1 to effectively create business reports, how BIRT provides end users the ability to drill down to detail, how BIRT can be used to develop rich functionality that gives life and further application value to traditional operation reporting, and more.

In the cover story this issue we move a notch upwards. In an organization, transactional data may be stored in a relational database, and real time information may be scrapped from a web site or captured through a web service or RSS feed. But business intelligence today requires that BI applications draw their information from a variety of disparate sources, and display it to the user in a friendly, unified, format. BIRT makes this possible. This issue, David Trainor and Jason Weathersby look at how BIRT can leverage Plain Old Java Objects (POJOs) to access, meld, manipulate, and present these varying forms of data.

Eclipse is a robust functional platform that IBM Workplace/Domino developers can put to full use in their current and future projects, says Mohamed Khiasudeen. His article focuses on the benefits of Eclipse as a client foundation that has a cross platform, rich UI widget set that is based on native widgets, a rich UI framework, pre-defined dialog basis: Wizards, Preferences, Properties, and other UI: Perspectives, Views, Editors, Workbench (as a base), ActiveX support in SWT on Win32 (platform integration), and a good Help system.

Several companies want to trade arbitrary products in a decentralized network. Imagine that you are assigned the task of having to write the software. This is not an easy task, if specialized GUI for different sectors must be integrated into one application. If you plan to use the Eclipse Rich Client Platform (RCP), it's worth having a look at the ERP framework, JFire, says Marco Schulze.

Open source projects benefit greatly from the participatory nature of the user community. There are many different ways to get involved in the Eclipse community. Wayne Beaton details how you can participate and contribute to the project.

This month we also spoke to Mik Kersten about the key benefits of the Mylar model, what we can look forward to in the 1.0 release, and how Mylar is being converted into a marketable product. At the time of press, the Eclipse Foundation has released Eclipse Mylar 1.0. Read what CodeGear, CollabNet, and a Brazilian multinational company have to say about Mylar on **Page 36**.

In the Plug-in Parade, Antonin Pokorny details FastTrack, a free tracker plug-in for Eclipse that provides basic issue tracking, plus basic project planning and team collaboration, all deeply integrated into the Eclipse IDE.

We hope you enjoy reading this issue of this magazine. If you would like to share your experience of having used Eclipse and how it has helped in your projects, write me at **editors@eclipsemag.net**. We will feature the ten most well written pieces in a future issue of Eclipse magazine.

Till next year, cheers from the Eclipse Magazine team.

[illegible]

Indu Britto
Editor-in-Chief, Eclipse Magazine

Announcements

The Eclipse on Linux (Linux Distro) Project Approved

The Eclipse on Linux (Linux Distro) project has been approved and is currently awaiting provisioning. The Eclipse IDE is perhaps the fastest growing open source application development environment available today. A mailing list posting of February 25, 2006 indicated almost 300,000 downloads of the Eclipse IDE in the preceding 30 day period. High levels of interest combined with an open, versatile platform and a well-run parent organization explain the widespread adoption of this platform. However, Eclipse hasn't enjoyed the type of success on the Linux platform that one might expect. The open source heritage of both Eclipse and Linux would lead one to assume that Eclipse would be highly successful on Linux, but the numbers may surprise. In the same period indicated earlier, only about 40,000 of the downloads were for Linux, comprising just over 13% of the total.

The purpose of this project is to make a case for an Eclipse on Linux Working Group within the Eclipse Foundation. The intent and goal of this working group would be to identify and address issues pertaining to the adoption of Eclipse on Linux, making Eclipse use on Linux more powerful and more widespread.

[MORE INFO]

Micro Focus to Deliver Eclipse Joy to COBOL Programmers

Micro Focus, a provider of enterprise application modernization software, has announced that it will support the Eclipse platform as part of its COBOL development product lines. Micro Focus will deliver the required plug-ins to enable rapid development of COBOL applications or composite Java/COBOL applications within the Eclipse Framework. Micro Focus is adding Eclipse support into its key products for Windows, UNIX, Linux, and mainframe application development. This will include existing Micro Focus facilities for creating Web services from existing applications, and will provide a unified Eclipse-based environment that will standardize the development of traditional workloads and facilitate the move to Service Oriented Architectures (SOA).

Micro Focus support for Eclipse will offer enterprise application development teams a consistent Integrated Development Environment (IDE) regardless of where the application is to be deployed, while taking advantage of the efficiencies inherent in Eclipse to offer best-of-breed productivity, the company said.

Micro Focus is already an Add-In Provider member of the Eclipse Foundation. This announcement means Micro Focus will accelerate its Eclipse integration, a move based on the growing popularity of Eclipse and on an increase in demand from its customer base.

Eclipse support within Micro Focus' distributed COBOL development products, Net Express (for Windows) and Server Express (for UNIX and Linux) is targeted for delivery during the second quarter of 2007. Eclipse support for Micro Focus' IBM mainframe application development environment, Mainframe Express® Enterprise Edition, will follow. Micro Focus will open an Early Adopter Program for its Eclipse integration capabilities in the first quarter of 2007.

"Micro Focus is committed to delivering highly productive COBOL development platforms that offer our customers choice among all leading IDEs," said Stuart McGill, vice president of worldwide marketing at Micro Focus. "We already offer our customers the ability to embrace the market leading Microsoft Visual Studio development environment and I'm delighted that today we can answer the requests of our customers who wish to standardize on the Eclipse platform by announcing our intent to add Eclipse support into our current product suite."

"Eclipse's goal is to create a multi-language development platform that increases efficiency in enterprise software development," said Mike Milinkovich, executive director of the Eclipse Foundation. "As the largest vendor of COBOL solutions off the mainframe, Micro Focus' announcement today is great news for COBOL programmers who can now look forward to enjoying the benefits of the Eclipse Platform while developing, extending and modernizing their enterprise applications."

[MORE INFO]

Eclipse DSDP Announces Three Milestone Releases

The Eclipse Foundation has announced three milestone releases within the Eclipse Device Software Development Platform (DSDP). Founded in 2005 as a top-level Eclipse project, the mission of DSDP is to create an open, extensible, scalable and standards-based development platform to address the needs of the device software market. This series of releases demonstrates the growing momentum and diversity of projects in DSDP. Created by Wind River, the DSDP project now has over 40 committers from ten companies and contains more than 550,000 lines of code.

Beyond highlighting momentum within DSDP, these milestone releases demonstrate an increasing significance of Eclipse to the device software market overall. As they move forward, the DSDP projects are aiming to help developers and vendors create specialized, interoperable solutions so that customers and users of Eclipse-based products can develop device software faster, better and at lower cost.

DSDP Milestone Releases

The three DSDP projects achieving milestone releases include:

- Target Management (TM), release version 1.0: The goal of Target Management is to create data models and frameworks to configure and manage embedded systems, their connections and services. Since there are many different vendors and solutions in the device software space, the main charter of target management is to provide data models and frameworks that are flexible and open enough for vendor-specific extensions. For the 1.0 release, sample implementations will be provided for TCP/IP connections, FTP data transfer and GDB remote launching in the CDT environment. The base technology for the TM project is an open-source version of the IBM Remote System Explorer. 1.0 features include Remote System Explorer framework, CDT remote launch capabilities and integrated Jakarta Commons Net library for FTP access. The contributors: Wind River (project lead), IBM, MontaVista, PalmSource, Symbian and Tradescape. This release will support Windows, Solaris, Linux and Mac.
- Embedded Rich Client Platform (eRCP), release version 1.0: the goal of this project is to extend the Eclipse Rich Client Platform (RCP) to embedded devices. eRCP enables the same Eclipse development model used to create applications on desktop machines to also be used on devices. The project includes a subset of RCP components tailored to mobile devices. 1.0 features include: Utilizes the familiar Eclipse programming model of applications providing Views to a workbench, reduces RCP footprint to fit on constrained devices, provides patches to core components to enable running those components on JME CDC/Foundation Profile mobile devices, and enables application binary compatibility across a range of devices with different input mechanisms and screen types/sizes. Contributors: IBM (project lead), Nokia and Motorola. This release supports Windows Mobile 2003/2005, Nokia Series 80 and S60.
- Mobile Tools for the Java Platform (MTJ), release version 0.7: the goal of MTJ is to extend the Eclipse platform to support mobile device Java application development. The purpose is to develop both frameworks that can be extended by tool vendors and tools that can be used by third party mobile java application developers. Mobile Java domain contains several combinations for configuration (CLDC and CDC) and profile (MIDP, Foundation Profile and Personal Profile). Currently the most common combination is CLDC+MIDP. 0.7 Features: A device and emulator framework, a deployment framework, generic build processes for mobile application development, mobile device debugging, application creation wizards, UI design tools, localization, optimization and security. Contributors: Nokia (project lead), IBM and SonyEricsson. This release supports Windows tools, Mobile deployment (Nokia).

"The DSDP project is crucial to fulfilling Eclipse's goal of creating a universal development platform for increasingly complex software," said Mike Milinkovich, executive director of the Eclipse Foundation. "DSDP has gained rapid momentum, and with these three major releases, the project now provides a broad foundation for commercial device software."

"These three releases are important milestones in DSDP's progress as a top-level Eclipse project," said Doug Gaff, leader of the DSDP Project Management Committee (PMC) from Wind River. "Wind River is particularly pleased that because of Target Management's successful release, device software developers now have an open source framework and a set of views for managing remote embedded systems from Eclipse. Wind River plans to adopt the Target Management technology in our next release of Wind River Workbench."

"With the eRCP and MTJ project releases, mobile Java developers have two new open source projects to facilitate the development and execution of Java ME applications," said DSDP Project Management Committee member Mark Rogalski. "IBM is using the eRCP project as the base runtime for IBM's Lotus Expeditor Client for Devices. Lotus Expeditor provides a programming model that delivers a universal client experience across the Lotus client portfolio, including Websphere Portal, Lotus Sametime and the new version of Lotus Notes code-named Hannover. The Lotus Expeditor Toolkit plugs directly into Eclipse or Rational Application Developer."

"Nokia sees the first GA releases of these projects as major progress in combining the power of open source development and commercial products for development in the device market," said Mika Hoikkala, project lead for Mobile Tools for the Java Platform from Nokia. "Though developed in our open source Eclipse project group, the frameworks from MTJ will provide a foundation for Nokia's Carbide.j commercial mobile development tools."

[MORE INFO]

TRANGO Systems Joins The Eclipse Foundation

TRANGO Systems has joined the Eclipse Foundation as an Add-In Provider. Specializing in embedded hardware virtualization, TRANGO Systems helps semiconductors and devices manufacturers to build up scalable and secure platform. By joining Eclipse Foundation TRANGO Systems will contribute to the open source IDE growing adoption among embedded developers community. Since April 2006, TRANGO Systems products are shipped with Eclipse plug-ins for system integration, target control, monitoring and virtual-JTAG debugging.

The TRANGO hypervisor allows multiple execution environments to run securely side by side on the same processor core, which, as a result, enables to securely deploy new services and reduce Bill of Material. By offering a safe environment for secure deployment of new services such as DRM or Device Management, TRANGO Systems helps operators and content providers growing new revenues. The TRANGO Hypervisor also allows systems to be certified and thus helps compliancy with security standards and requirements.

"We warmly welcome TRANGO Systems as a member of the Eclipse Foundation. As the company is working with worldwide leading semiconductors manufacturers, we expect that the TRANGO Hypervisor adoption will help strengthening Eclipse IDE use within semiconductors and devices industry" stated Mike Milinkovich, Executive Director of the Eclipse Foundation. Available on ARMv5 and MIPS32 / MIPS64 embedded architectures, the TRANGO Hypervisor and its Eclipse tools support Linux, Windows CE 5.0, as well as eCos and µC/OS-II.

[MORE INFO]

VTT Joins Eclipse Foundation

VTT Technical Research Centre of Finland has joined the Eclipse Foundation as an associate member. VTT's first contribution to the Eclipse community is a tool designed for managing a software architectural knowledge base. Eclipse represents part of the implementation process for VTT's technology policy. In order to express VTT's support to the Eclipse community, VTT has joined the Eclipse Foundation as an associate member.

VTT has recently contributed to the Eclipse ecosystem by developing a tool called "Stylebase for Eclipse". The tool provides assistance for software architects and designers. It has been implemented as an Eclipse extension and published under the GNU General Public License, the most well-known open source software license.

Eclipse is known as the most popular open source development environment and a universal platform for tool

integration. Eclipse forms an independent open eco-system around royalty-free technology. The Eclipse community has developed new features that have helped evolve the platform towards integrating not only tools but also applications and services. New features also make Eclipse a strong force in the embedded market and expands the community even further internationally. Today, Eclipse is more than "just" a development environment, it has become a powerful platform that serves the entire application life cycle.
[MORE INFO]

Software AG's Natural for Eclipse: Create SOA-Ready Enterprise Apps

Software AG has rolled out Natural for Eclipse, a full Eclipse-based environment for Natural 2006 application development. The development and deployment of Natural and Java business applications from a common development platform will ease the creation of SOA-ready, mission-critical enterprise applications for both mainframe and open systems, said the company. Software AG has partnered with innoWake, based in Ulm, Germany, to provide technology for Natural for Eclipse.

With Natural for Eclipse, customers can hope to extend and integrate their enterprise systems with the latest modern technology infrastructures, such as a Service-Oriented Architecture (SOA), while maintaining the highest levels of performance and reliability. "The product is the latest addition to the Natural Productivity Package which improves application developer productivity and helps customers reduce time and costs for application development and maintenance", said a company spokesperson.

In the Natural Productivity Package Professional Edition, Natural for Eclipse provides an open source based development environment for Natural applications deployed on Mainframe, Unix, Linux or Windows platforms. In the Enterprise Edition, Natural for Eclipse enables Java and Natural developers to collaborate on the development of Service-oriented applications and AJAX-based rich internet applications.

Software AG's partner innoWake is a provider of Java and web-based development tools using open source environments like Eclipse. Joseph Gentry, vice president of Enterprise Transaction Systems for Software AG said, "Natural for Eclipse is an enterprise-class product that allows our customers to make use of the latest development tools and approaches while protecting and extending the investments they have made in applications and skills." Software AG's Natural for Eclipse is currently available on the Windows and Linux platforms and can deploy onto mainframes, Unix, Linux and Windows.
[MORE INFO]

IBM Rolls Out Eclipse-based Alternative to Microsoft .NET Client Software

IBM has unveiled Lotus Expeditor, a development platform for creating Eclipse-based and Web 2.0 applications. Lotus Expeditor's open standards-based programming model is expected to enable enterprises to integrate existing and new applications and deliver them to a variety of connected and disconnected devices.

Lotus Expeditor provides the underlying programming model for a universal, end user experience across all the Lotus product portfolio including Lotus Sametime 7.5, WebSphere Portal 6.0, and the upcoming release of Lotus Notes, code-named Hannover.

Software applications that do not have a common language or structure make sharing data and information difficult, if not impossible. Lotus Expeditor is aimed at helping enterprises address these constraints by giving developers the ability to create a personalized user experience that spans across desktop platforms and devices providing access to critical business applications and data anytime, anywhere. For example, with Lotus Expeditor users can create composite applications combining existing and new software assets within a service oriented architecture (SOA), add extensions to existing databases that allow mobile workers to access key financial or sales information from any mobile device, or augment existing applications using Eclipse or Web 2.0 technology to create a Voice-Over Internet Protocol (VoIP) plug-in for an instant messaging environment.

New features in Lotus Expeditor:

- Developer Toolkit: An enhanced toolkit, which installs as an Eclipse feature, providing wizards for portal and web services-based applications and templates that developers can use to speed up the creation of a new class of off-line applications.
- Composite applications: Facilitates the creation of composite applications and mashups into integrated views and processes across different application types. These applications are delivered in many ways but within a single user interface accessed via a web browser, a rich client or a text terminal. For example, Lotus Expeditor helps a bank teller more efficiently fulfill a customer request by providing an integrated view comprising multiple customer-care applications.
- Mobile device support: Enables use of applications on mobile devices, such as PDAs and Smartphones. Unlike .Net and Visual Basic applications, Lotus Expeditor can centrally manage and distribute applications with pre-defined user access levels based on an employee role or function. Without the appropriate security access, users are locked from viewing or updating sensitive or confidential data.

"An open alternative to Microsoft .NET client software, Lotus Expeditor provides the flexibility that comes from service oriented architecture (SOA) and open standards-based software, giving all users a universal client experience and allowing WebSphere Portal, Lotus Forms and Lotus Sametime users to extend their current applications world beyond the desktop," said Ken Bisconti, vice president Lotus Software, IBM.

Complementing Lotus Expeditor is Lotus Mobile Connect, software that extends the use of Expeditor-based solutions by adding secure network roaming for a wide variety of mobile and wireless devices. This product helps users become more productive through mobile access to IBM Workplace Forms, IBM WebSphere Portal via laptops, tablets or desktops, and other applications, allowing users to seamlessly continue with their applications when they are reconnected. As an example, Lotus Mobile Connect allows users to continue working with Lotus Sametime Instant Messenger where they last left off without needing to log in again, even when networks are sporadic.

Pricing on Lotus Expeditor and Lotus Mobile Connect will be available at the time of product availability in 2006. [MORE INFO]

Helmi Technologies On Board the Eclipse Foundation

Helmi Technologies, a provider of the Open Source RIA Platform, has joined forces with the Eclipse Foundation as in Add-in provider member. Integrating Helmi's Ajax-based Platform leverages existing skills and the familiar Eclipse development environment. Using the Eclipse environment and Helmi's Platform allows customers to rapidly create browser-independent, high- impact web applications, while cutting development time by over 50 percent, said the company.

"Companies such as Helmi play a vital role in the Eclipse Foundation," said Mike Milinkovich, executive director at Eclipse Foundation. "Helmi's technology and source code is now available to the Eclipse community. Ajax solutions are particularly in high demand and Helmi is offering Eclipse users an Ajax-based solution to rapidly and cost-effectively create browser- independent AJAX rich Internet applications."

The Helmi Open Source RIA Platform delivers on the promise of Web 2.0. Client- and server-side engineers can create high performance applications because they can use the same components and communicate effectively through an object-oriented framework, said a company spokesperson.

"We are excited to be joining the Eclipse Foundation," said Jorden Woods, CEO of Helmi Technologies in the U.S. "Our goal when developing our open source product was to allow companies to leverage their development team's knowledge base and shorten the learning curve to maximize productivity. We share the same vision as the Eclipse Foundation which is to enable developers to easily create derivative works from our Platform in an open source environment."

[MORE INFO]

New Releases

Eclipse Releases New Device Development Tools

The Eclipse Foundation has released two plug-ins aimed at making the open-source Eclipse Integrated Development Environment (IDE) more useful to device developers, and a third has achieved a milestone pre-release, version 0.7. Two of the three need more community participation to improve their Linux support, the organization claimed.

According to the foundation, the three newly available device-oriented Eclipse plug-ins are:

- Target Management 1.0
- Embedded Rich Client Platform (eRCP) 1.0
- Mobile Tools for the Java Platform (MTJ) 0.7

All three plug-ins were developed under the aegis of a top-level Eclipse Foundation project called the Device Software Development Platform (DSDP), the organisation said.

Spearheaded by long-time development tools leader Wind River, the DSDP project was founded in mid-2005, and has since attracted about 40 'committers' (significant software contributors) representing ten companies. Like other top-level Eclipse projects, it organises multiple sub-projects aimed at creating plug-ins for the modular, open source Eclipse IDE. The IDE has been widely adopted by embedded tools vendors, in part because it runs on Linux, Windows, Solaris, and Macintosh development hosts, saving vendors the pain of maintaining their tools on multiple platforms, the foundation claimed.

Eclipse plug-ins enable commercial tools vendors and individual users to customise the modular, extensible Eclipse IDE according to their needs mixing and matching components from multiple third party providers, as well as from the open source community. Plug-ins typically used by device developers include code editors for various languages (C/C++ and Java are popular), debuggers, compilers, and test suites and automation frameworks, the foundation added.

Doug Gaff, a Wind River engineering manager who heads up the DSDP's organising committee, says today's milestone releases signal gathering DSDP momentum, as well as the increasing significance of Eclipse to device developers. [MORE INFO]

Borland Redesigns Java IDE, JBuilder 2007 Built on Eclipse Platform

CodeGear, formerly known as the Developer Tools Group of Borland Software Corporation, has unveiled JBuilder 2007, an Integrated Development Environment (IDE) built on Eclipse to make development fast and reliable for Java, open source and the web, the company is said to have quoted.

According to the company, CodeGear JBuilder 2007 is said to accelerate developer velocity with Visual EJB and web services GUIs, provides balance between the worlds of open source and commercial software with TeamInsight, an integrated collaboration portal. Improving on Rapid Application Development (RAD) and team collaboration capabilities, JBuilder 2007 (formerly codenamed 'Peloton') has been completely redesigned to leverage the Eclipse platform.

Borland says that two trends are driving change in software development today. Firstly, open source is more pervasive than ever. Secondly, development teams are increasingly distributed around the world.

Mike Milinkovich, executive director of the Eclipse Foundation, said, "JBuilder 2007 aims to deliver enterprise-class capabilities that complement and extend the Eclipse platform, helping organisations and developers to better manage distributed teams and development".

Completely rebuilt and re-architected to take advantage of Eclipse, CodeGear's JBuilder 2007 combines the advantages of an open platform with JBuilder's trademark. According to the company, key benefits and features include:

- Velocity
- Confidence
- Balance

JBuilder 2007 will be generally available by the end of the year, and will be backward compatible with applications built in previous versions of the product.
[MORE INFO]

Wolfram Workbench, Based on Eclipse, Rolls Out

Wolfram Research today announced the release of Wolfram Workbench, the state-of-the-art integrated development environment (IDE) that incorporates Mathematica technologies. Complementing Mathematica's uniquely powerful, high-level programming language and world-renowned features for computation, visualization, and modeling, Workbench is ideal for the development of large-scale technical solutions.

"Workbench has unlocked Mathematica's potential as the way to build technical software," said Tom Wickham-Jones, director of kernel technology and lead Workbench developer. "If you're working on technical application development, we've built the definitive system." Together, Workbench and Mathematica enable the creation of innovative technical applications in areas such as engineering, science, finance, and education.

Built on Eclipse, a leading IDE platform, Workbench makes it easy for Eclipse adopters to use Mathematica for their applications. Existing Mathematica-based developers gain access to state-of-the-art development tools, resulting in more efficient project construction and increased productivity.

"Mathematica is the only system capable of spanning the development spectrum, from small to large projects," said Conrad Wolfram, director of strategic marketing at Wolfram Research. "Workbench boosts Mathematica's capability for more complex projects, taking technical development into the twenty-first century, away from Fortran and its derivatives."

Key features in Workbench enable users to:

- Group files, code, and other Mathematica resources into a single project
- Perform source code editing with syntax highlighting, error reporting, local variable coloring, and many more options
- Study code as it runs to easily detect and fix any problems
- Profile the code's execution and develop and run tests, with an array of insightful reporting methods

- Manage multiple versions of files and access version histories
- Build and deploy Mathematica packages

Features that make Mathematica ideal for development include:

- Multi-paradigm programming that lets users program as they think, not as the language dictates
- Short, readable code for faster, simpler implementation
- The world's most extensive computation library, with symbolic as well as numerical and graphical functionality
- Built-in connectivity to other languages and extensions
- Cutting-edge deployment options

"Workbench has quickly become an indispensable tool we wouldn't want to live without. We can now develop both Mathematica and Java code in the same IDE and profit from the version control and build-management infrastructure we have already established around Eclipse," said Sascha Kratky of uni software plus, a developer of custom software solutions in finance, engineering, and manufacturing.
[MORE INFO]

JasperSoft Debuts Report Designer For Eclipse IDE

JasperSoft, a provider of open source business intelligence, has launched iReport Plug-in for Eclipse, a graphical report designer that lets developers using Eclipse quickly add reports to their applications, the company said.
JasperSoft's new Eclipse plug-in, as well as the iReport designer application and the JasperReports reporting libraries are all freely available for download and use under open source licenses. This is the first integration of Eclipse with JasperSoft's open source BI products, which have been downloaded more than 1.7 million times, the company added.

According to the company, Eclipse users can now create, edit, and run reports from the same familiar development environment they use everyday, speeding up their workflow and reducing project development times. Java developers can access the iReport graphical report editor to give their applications rich reporting capability based on JasperReports library that can create Web and pixel-perfect ready to print reports in PDF, HTML, XLS, CSV, TXT and XML. In addition to easy report creation, the new iReport Plug-in for Eclipse also gives developers access to the entire JasperIntelligence suite of open source BI applications to manage interactive reports from a server, schedule reports, and analyse data.

"JasperSoft developed the iReport Plug-in for Eclipse in response to demand from Java developers to get direct access to the world's most popular open source reporting library with the most popular open source IDE platform", said Giulio Toffoli, lead committer for the iReport project and iReport engineering lead at JasperSoft.
[MORE INFO]

MyEclipse 5.1 with All-in-One Installer Released

The release of MyEclipse 5.1 is now available for immediate installation. Windows, Linux, and Mac OSX support is available. MyEclipse is an Integrated Development Environment (IDE) based on the Eclipse platform primarily aimed at improving developers' productivity by simplifying development lifecycle in the delivery of UML, Web, J2EE, XML, JSP , JSF, Struts, Hibernate and database applications. The MyEclipse 5.1 release includes enhancements and bug fixes for the MyEclipse environment over the previous 5.0 GA release, including:

- Eclipse 3.2.1 compatability
- All-in-One Installer, including MyEclipse 5.1, Eclipse 3.2.1 & Java 5 JRE
- 130+ bug fixes
- Wizard to create Web Service Client from any WSDL file, local or remote (URL)
- Upgraded Web Services XFire to latest release (1.2.2)
- Matisse4MyEclipse now supports custom user forms, custom controls, improved usability

- Integration of Spring IDE 1.3.6
- Supports user deployment of prepackaged EJBs along with MyEclipse enterprise projects
- Improved Sybase SQL syntax support
- Improved stability issues of Visual Web Designer on Linux
- Improved validation framework performance; JSP now 7x faster and includes the ability to exclude resources from validation
- Enhanced Hibernate HQL Editor supports multi-project configuration
- Optimized application redeployment feature that provides reload effect and virutally eliminates common appserver Jar locking issues
- Web 2.0 Browser now with HTTPS support
- New Tomcat 6 Server Connector
- MyEclipse now fully I18N enabled

Though Matisse4MyEclipse is fully integrated into MyEclipse, Macintosh users will be unable to utilize the Matisse4MyEclipse, MyUML, and MyEclipse Image Editor functions due to the long-standing Eclipse SWT_AWT bug #145890. The MyEclipse 5.1 release is available to all free trial users. Please see the installation instructions above for instructions on how to try out MyEclipse!
[MORE INFO]

oXygen XML editor 8.0 (Eclipse plugin)

oXygen is an XML editor that supports any XML document, and works with XML Schemas, DTDs, Relax NG schemas, and NRL Schemas. It has powerful transformation support that allows you to edit XSLT and XSL-FO documents and to obtain documents in the desired output format (such as HTML, PS, or PDF) with just one click. It also includes a complete Subversion client, support for flattening XML Schemata, an XML Schema instance generator, integration with the X-Hive/DB, MarkLogic and TigerLogic XML databases, editing actions on the diagram, and a rename refactoring action.

oXygen XML editor 8.0 is a major feature enhancement edition. The new grid editor allows you to edit repetitive XML content in a special layout similar to a spreadsheet application, without the need to interact with the markup. Using the new Database perspective you can browse tables or collections from databases, execute XQuery or SQL queries, inspect or modify data, specify XML Schemas for the XML fields and collections. The database support includes many of the popular servers, operating either as native XML storage (Tamino, XHive, MarkLogic, TigerLogic, eXist, Berkeley) or mixed, as relational and XML at the same time (DB2, SQLServer, Oracle).
[MORE INFO]

OpenXava Eclipse Plugins 0.1 Rolled Out

The OpenXava Eclipse Plugin provides a database reverse engineering utility for the OpenXava framework. This plugin allows the user to generate an OX component from existing database tables. Using it in conjunction with the OpenXava framework, you can easily generate a J2EE application from an existing database. This is the first version, and it includes support for connecting to a database to retrieve metadata from Eclipse, full customization for generated code, and support for advanced mapping features.
[MORE INFO]

Topcased 0.11.0 Released

Topcased is an integrated System/Software engineering toolkit compliant with the requirements of critical and embedded applications. It covers the stages from requirements analysis to implementation, as well as some transversal activities like anomaly management, version control, and requirements traceability. Topcased is strongly model-oriented. Not only does Topcased provide model editors, model checkers and model transformations, it is also by itself based on modelling and code generation. With Topcased, you can develop your own graphical editors and model transformation using Topcased. The Topcased team has announced the release of Topcased 0.11.0. Two archives are available: a toolkit version that only contains the binaries, and the SDK that contains the toolkit and the source code, this last version is aimed for developers.
[MORE INFO]

jax® INNOVATION AWARD 2007

20,000 Euros for your Innovation!

In the era of Web 2.0, it is time to seek out the best innovations that can take us to the next generation. The JAX Innovation Awards, endowed with a prize money of 20,000 Euros, has been constituted to throw the spotlight on the most striking contributions to the world of Java, Eclipse, and SOA. The awards will showcase technological innovations from the European market.

Who can participate?

Contributions can come from

- Individuals
- Companies
- Universities/Research institutes
- Other organizations

Hand in your contributions and win big!

jax-award.com

Deadline: 19 March 2007

S&S

Eclipse Power In IBM Workplace/Domino

Preparing for Tomorrow's Challenges

By Mohamed Khiasudeen

Eclipse is a robust functional platform that IBM Workplace/Domino developers can put to full use in their current and future projects. In this article, we focus on the benefits of Eclipse as a client foundation that has a cross platform, rich UI widget set that is based on native widgets, a rich UI framework, pre-defined dialog basis: Wizards, Preferences, Properties, and other UI: Perspectives, Views, Editors, Workbench (as a base), ActiveX support in SWT on Win32 (platform integration), and a good Help system. Eclipse as a client foundation is an extensible platform that features a plug-in extensibility model, shared programming model with tools development, education that is already developed for tools offerings, core services, extension points, core frameworks, production quality platform with two major releases in the market, and an Open Source code base.

Introduction

Eclipse is a true Open Source development framework or workbench, along with set of widgets (SWT), designed for tools developers to leverage code reuse, a consistent user interface, and a plug-in architecture for developing new packages. However, while Eclipse was originally targeted and designed for tools developers, it has now evolved to become a more robust general-purpose application platform with its powerful Rich Client Platform (RCP).

IBM Workplace Designer 2.5 is a true Eclipse-based tool that brings the concepts of Domino Designer to the J2EE and Service-Oriented Architecture (SOA) space. With the power of Eclipse, IBM Workplace Designer is able to provide the Domino developer community with rich functionality such as code complete, color-coding of classes and objects within classes, and an advanced IDE experience.

IBM Workplace Designer constitutes a core part of the IBM Workplace product umbrella, which includes the Notes/Domino family, the Web Sphere Portal family and WorkSpace Collaborative Services.

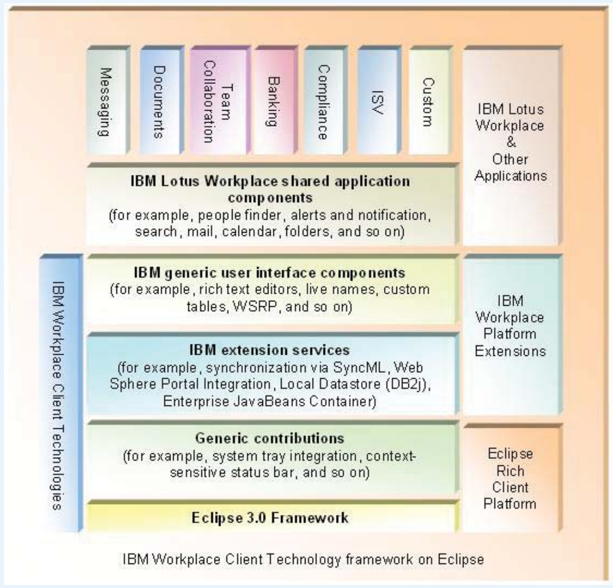
Eclipse In IBM Workplace

The core platform of IBM Workplace is based on Eclipse technology, which provides the Java run-time environment for general desktop applications, an application user interface, and a flexible architecture that is easily extended and supports multiple operating systems.

Workplace Client Technology is a collaboration of:

- **A secure data store (a Java relational database)**. This data store is a zero-administration, pure Java relational database.
- **The Eclipse rich client framework**.
- **A personal EJB container for running local applications**. The lightweight EJB container, Extension Services for Web Sphere Everyplace (ESWE) was built from the ground up by the Web Sphere team for resource constrained devices such as PDAs and takes up less than 1 MB of disk space.
- **The ability to download layouts and application components from Web Sphere Portal**.
- **A synchronization framework**. Synchronization of the data between the local data store and remote applications is achieved via an implementation of SynchML. Synchronization allows new features to be pushed to desktops.

Fig. 1: IBM Workplace Client Technology Framework on Eclipse Platform


Messaging
Documents
Team Collaboration
Banking
Compliance
ISV
Custom
IBM Lotus Workplace & Other Applications
IBM Lotus Workplace shared application components
(for example, people finder, alerts and notification, search, mail, calendar, folders, and so on)
IBM generic user interface components
(for example, rich text editors, live names, custom tables, WSRP, and so on)
IBM Workplace Platform Extensions
IBM extension services
(for example, synchronization via SyncML, Web Sphere Portal Integration, Local Datastore (DB2j), Enterprise JavaBeans Container)
Generic contributions
(for example, system tray integration, context-sensitive status bar, and so on)
Eclipse Rich Client Platform
Eclipse 3.0 Framework
IBM Workplace Client Technologies
IBM Workplace Client Technology framework on Eclipse

- **Auto-provisioning capabilities.** Provisioning features allow for the creation of accounts based on policies defined for users.

The Eclipse platform provides building blocks and frameworks to facilitate the development of new tools via plug-ins. These plug-ins consist of structured bundles of code and data that allow for the extensibility of this framework to read the plugin.xml for each plug-in. The extension services layer also includes new non-Eclipse plug-ins created specifically for Workplace Client Technology.

Eclipse Framework in IBM Workplace

Additionally, the Eclipse framework has incorporated SMF into its underpinnings, so ultimately, all of the plug-ins that run in Eclipse run as bundles, and can be dynamically loaded, unloaded, started and stopped. This adds an additional element of flexibility and choice to the application developer, as well as a more robust platform underneath.

IBM Workplace Client technology is a major and important component of the IBM Workplace family. With Workplace Client Technology, we can develop, deploy, and centrally manage rich end-user client applications. This allows you to extend management and security features found in server-based platforms to user desktops and to other devices. These rich client applications give you the flexibility and portability of client-side applications combined with server-side control and cost-savings. Workplace Client Technology comes in two editions: Micro Edition for mobile devices and Rich Edition for desktops.

IBM Workplace Client Technology incorporates standards-based components and technologies, including Java and the Eclipse 3.0 open standard. It includes central policy-based administration features such as provisioning and synchronization. Applications can run locally on the end user's machine. By leveraging the embedded EJB (Enterprise JavaBean) container and encrypted relational database, users can work offline and online. Workplace Client Technology also features a synchronized, policy-based secure data store that allows data to be stored locally until the user is ready to synchronize the data with the server (this is similar to DB2's synchronization services of Web Sphere Everyplace Connection Manager).

Open Standards in IBM Workplace

The Eclipse platform was created as open-source software originally intended for use as a GUI interface to support custom-created development tools. IBM has built the Workplace Client Technology on its open architecture to take advantage of its ability to provide UI and execution environment for plug-in modules. An Eclipse plug-in operates on files in the workspace and provides a tool-specific UI. When the platform is launched, the user is presented with an integrated set of available plug-ins - the desktop analogy to a server-based portal.

The plug-ins in the extension services layer includes Eclipse plug-ins as well as plug-ins created especially for Workplace Client Technology that are not part of core Eclipse. Eclipse plug-ins include the update manager, help system, preferences, and workbench components. The update manager centrally manages deployed applications. Help integration provides the client with methods to define context so that the online help will automatically launch with the appropriate context. Alerts notify the user that an action has happened, and preferences customize the desktop to present to the end user.

Eclipse UI Frameworks in IBM Workplace

- Workbench Parts
- EditorsParts – For Edit or Browse a resource
- ViewParts – Navigate Content From defined input
- JFace Text Framework – Support, display and editing of source text

JFace UI Framework

- Viewers – Support for building model-based visual parts with standard viewers
- Actions and Contributions – Support for creating and managing shared resources (menu bar, tool bar, status area)
- Dialogs and Wizards – Single or multi page dialogs that provide task guidance or value management
- Images and Fonts – Support for management and construction of resources such as SWT fonts and images
- Standard Widget Toolkit – Portable API to platform Native widgets

IBM Workplace Messaging

Workplace Messaging provides users with integrated mail, personal calendar, and personal address book. Within Workplace Messaging, Workplace Client Technology uses Eclipse views to provide navigation, management, and display of mail folders such as the Inbox and user-defined folders. Editors to create, edit, and view emails are provided with both Eclipse controls and a pluggable rich-text editor.

Summary

Eclipse is a robust functional platform that IBM Workplace/ Domino developers can harness for their current and

Fig. 2: Workplace Client Technology Architecture

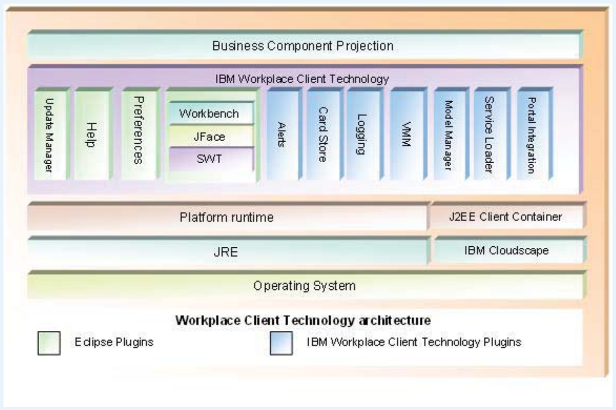
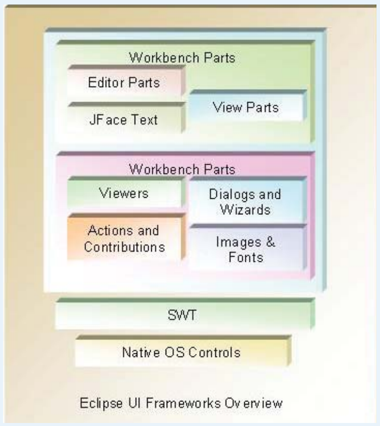


Fig. 3: Eclipse UI Frameworks - Workplace Client Technology



future projects. Developers benefit through a better way of organizing projects, sharing code, and taking the full potential advantage of Java design techniques like interfaces and code separation. As Lotus Workplace and Lotus Notes/Domino continue to become more integrated, Java and Eclipse will play an important role. Notes developers can build on their collaboration and groupware skills today while preparing for the challenges of tomorrow.

Resources & References

[1] http://www.eclipse.org/
[2] http://www.ibm.com/
[3] http://www.notes.net /
[4] http://www.lotus.com/

Mohamed Khiasudeen

works as a Consultant at Wipro Technologies, India. With over 15 years of experience in Information Technology, Khiasudeen has extensively worked with IBM Domino and IBM Workplace technologies.

Free Trade with Eclipse

An Introduction to the JFire Open-Source ERP Framework

By Marco Schulze

Imagine several companies want to trade arbitrary products in a decentralized network, and you have to write the software. Not an easy task, if specialized GUI for different sectors must be integrated into one application. If you plan to use the Eclipse Rich Client Platform, it's worth having a look at the ERP framework JFire.

Introduction

If you have already written a shop system or a similar project, you know that there are a lot of requirements that are more or less common to all these systems. Thus, it would be desirable to have a platform, which fulfills many of these requirements and eases extensions into every direction. But what do these common requirements look like? Some of them can easily be formulated, such as the concepts of 'customer' or 'order'. Others, however, are harder to find. For example, a price configuration can be highly varied as the case arises. Despite these variances, common ground emerges from generalization and abstraction. JFire's goal is to bundle solutions for these common requirements and to provide them as a modular ERP framework that is easily extensible. It aims to cover most usual cases and allows for extraordinary solutions by providing numerous interfaces and extension points.

Basic Set Up

In order to understand the structure of the client with its extension points, we first need to look at the big picture.

JFire's[1] server side is based on the Java 2 Platform Enterprise Edition (J2EE 1.4) and Java Data Objects (JDO 2.0). Due to this architecture, it's easy to extend the data model by Java inheritance without worrying about SQL. The communication between client and server or between two servers is handled comfortably by J2EE. This backend allows to easily implement interfaces to other systems or clients (for example, a Web shop).

At the moment, JFire contains an Eclipse-RCP-based client. It also includes five perspectives for system administration (users, access rights, and so on), product configuration, order processing, report design, and a 2D editor. Since most modules are split into a user and an administrator plug-in, an installation can be closely adjusted to the operator's needs. This does not only save time when updating, but helps to keep a normal user's GUI simple as well.

Separate Cooperation

In order to enable multiple companies to trade without disclosing confidentiality, JFire manages them as so-called *Organizations* in separate databases. The organizations can even be hosted on different J2EE servers. To communicate, they only need to be connected to a common network (for example, the Internet) and to be acquainted with one another. This potentiates users to set up a flexible infrastructure, which is extremely scalable and seldom requires expensive clustering.

Elucidations

- **Extension Point**: An extension point is a plug-in's interface description that allows other plug-ins to integrate additional functionality into the existing logic. The Eclipse RCP uses this mechanism, for example, to permit populating the main menu with own entries.
- **Extension**: An extension is used to extend a previously defined extension point. You can declare an unlimited number of extensions for each extension point.
- **JBoss**: JBoss[2] is a widely used open source J2EE server.
- **Java EE**: Java EE[3] is the abbreviation for Java 2 Platform Enterprise Edition. This standard was developed by Sun in order to simplify the development of distributed applications.
- **JPOX**: JPOX[4] is open source and was selected by Sun to be the reference implementation of JDO.
- **JDO**: Java Data Objects[5] is a standard for data persistence developed by Sun. It allows for storing and retrieving Java objects to / from an arbitrary database. Currently, JDO is mostly used for object relational mapping, because relational databases are still most prevalent. However, persisting into object databases or even flatfiles using the same API is possible.
- **EJB**: Enterprise Java Beans (EJB) are components managed by a J2EE server. They ease the development of distributed applications.

Users, Access Rights, and Workstations

A J2EE server is already able to check access rights and, if necessary, to reject method calls. In order to use this functionality, JFire provides a JDO-based login module for JBoss. Other J2EE servers will be supported in the medium-term. See Figure 1.

The plug-in *org.nightlabs.jfire.base.admin* provides the perspective 'System Administration', in which you can edit users or user groups and assign access rights to them. Additionally, you can manage your workstations here. In the long run, the administrator will be able to configure which plug-ins shall be installed (and automatically updated) on which workstation. This feature is based on the Eclipse update system; however, it's not completed yet.

Central Configuration

Every Eclipse user knows the dialog 'Window' / 'Preferences...'. Certainly, the JFire configuration is accessible here, too. But instead of using the usual **.prefs* files, it stores (almost) all settings in JDO objects on the server. Therefore, JFire contains a framework, linking the so-called configuration modules to users or workstations. This empowers the administrator to centrally control not only access rights, but all settings of the users in the perspective 'System Administration'. She can even configure which settings a user can modify himself.

To register your own preferences pages for the JFire configuration, create an extension to *org.eclipse.ui.preferencePages*. However, you need to inherit *AbstractUserConfigModule PreferencePage* or *AbstractWorstationConfigModulePreference Page* from the package *org.nightlabs.jfire.base.config*.

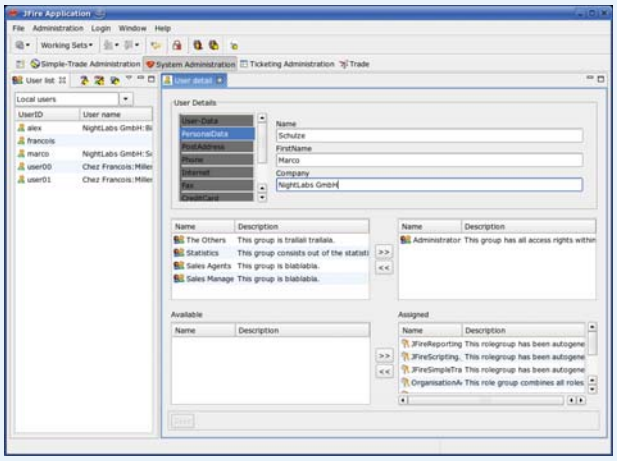


Fig. 1: User management with personal data, user groups and access rights

UserID	User name
alex	NightLabs GmbH: Bi
francois	
marco	NightLabs GmbH: S
user00	Chez Francois: Miller
user01	Chez Francois: Miller

Name: Schulze
FirstName: Marco
Company: NightLabs GmbH

Name	Description
The Others	This group is trallali trallala.
Statistics	This group consists out of the statisti
Sales Agents	This group is blablabla.
Sales Manage	This group is blablabla.

Name	Description
Administrator	This group has all access rights within

Name	Description
JFireReporting	This rolegroup has been autogene
JFireScripting	This rolegroup has been autogene
JFireSimpleTra	This rolegroup has been autogene
OrganisationA	This role group combines all roles

Client/Server Compatibility by RemoteClassLoader

The JDO data model and the EJB interfaces must be accessible in the server as well as in the client. How can we avoid the resulting maintenance and deployment overhead? The JFire client uses a RemoteClassLoader that acquires classes and resources via J2EE at runtime. You only have to publish the resources on the server by means of an XML descriptor. This ensures that client and server operate on the same classes and thus reduces not only maintenance effort but also eradicates potential bugs.

Specialised Product Types

The attributes of miscellaneous product types offer significant differences. A car, for example, is characterised by performance and mileage, whereas those properties don't apply to a plane ticket at all. To represent all possible kinds of products, the base classes *ProductType* and Product are abstract and must be inherited for your own purposes.

The module *JFireSimpleTrade* serves as inspiration for new product types; it provides simple but versatile products that are labelled with a localisable name. The corresponding sales and administration GUI is contributed by the plug-ins *org.nightlabs.jfire.simpletrade* and *org.nightlabs.jfire.simpletrade.admin.*

Sophisticated Sales GUI

Although different product types mostly have little in common, it should be possible to book them in the same order. See Figure 2. However, no GUI element is able to generically display all products. Furthermore, especially the particular attributes of the respective products should be made available to the users. Thus, it must be possible to assign to each kind of product its own GUI element, which can be presented concisely in the same view. To achieve that goal, the plug-in *org.nightlabs.jfire.trade* defines a set of extension points. The most important ones are:

- *org.nightlabs.jfire.trade.articleAdderFactory:* an *Article Adder* provides GUI (red arrow), which adds products to an order. JFire denotes the resulting order line as Article.
- *org.nightlabs.jfire.trade.articleEditFactory:* an *ArticleEdit* serves to create GUI elements for displaying and editing order lines. Two *ArticleEdits* for different product types can be seen in the screenshot (yellow arrows).
- *org.nightlabs.jfire.trade.articleContainerAction:* this extension point allows to register actions referring to the complete order (for example, print). Since orders as well as offers and invoices can be addressed, the term *ArticleContainer* is used. All those actions can be found in the context menu, the coolbar and the main menu (green arrows).
- *org.nightlabs.jfire.trade.articleEditAction:* besides actions for the *ArticleContainer* there have to be Actions able to manipulate single items (e.g. release or delete). They are also accesible via context menu as well as cool bar and main menu (blue arrows).

Fig. 2: Perspective for purchase and sale

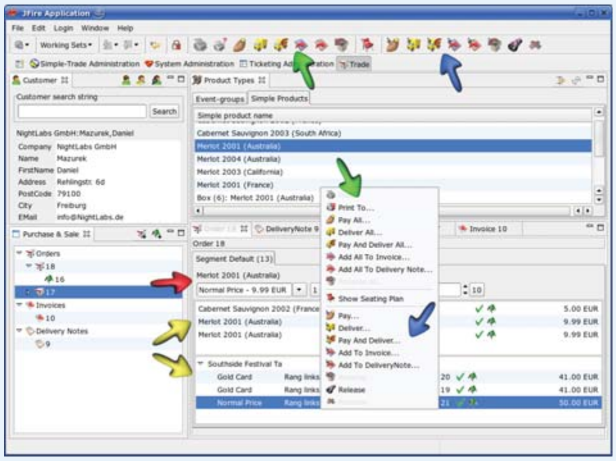
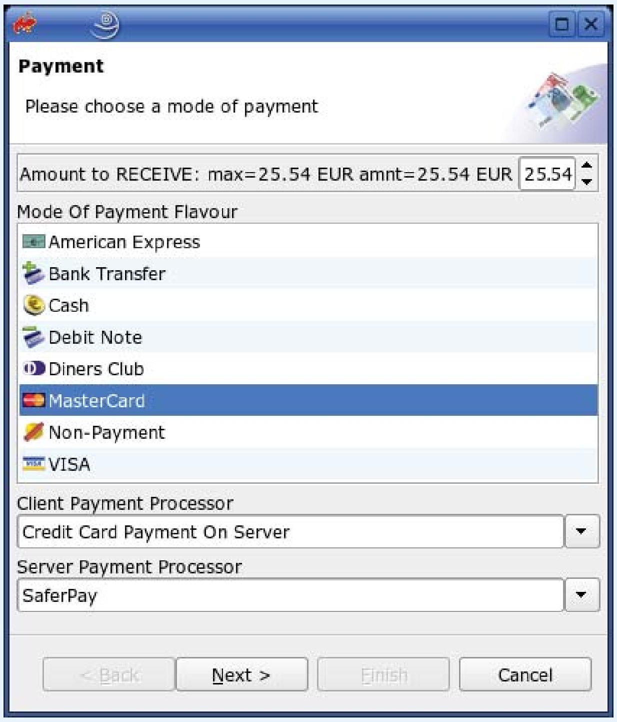


Fig. 3: Page 1 of the payment wizard



Payment

Please choose a mode of payment

Amount to RECEIVE: max=25.54 EUR amnt=25.54 EUR 25.54

Mode Of Payment Flavour

- American Express
- Bank Transfer
- Cash
- Debit Note
- Diners Club
- MasterCard
- Non-Payment
- VISA

Client Payment Processor: Credit Card Payment On Server

Server Payment Processor: SaferPay

< Back | Next > | Finish | Cancel

Wares for Money

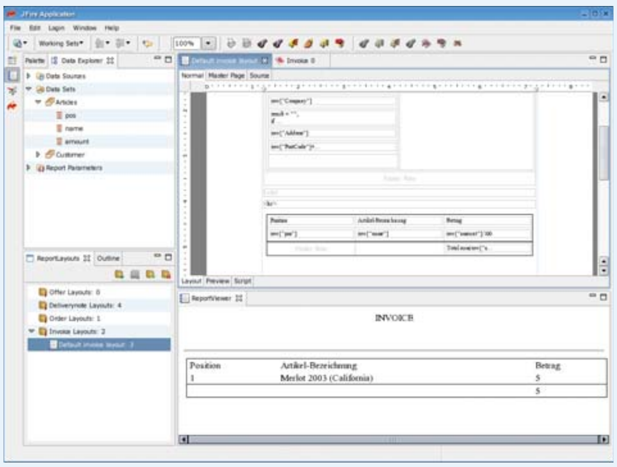
The selected goods have to be paid sometime. For this purpose, there is a payment framework, which can be easily extended by backend as well as frontend modules. At first, JFire knows various modes of payment (class *ModeOfPayment*). Every one of them can contain multiple sub-modes called *ModeOfPaymentFlavour* (managed by a JDO class with the same name). An example could be 'VISA', 'MasterCard' or 'American Express' as sub-types for the mode of payment 'Credit Card'. To interface with an external payment service, extend the classes *org.nightlabs.jfire.accounting.pay.ServerPaymentProcessor* or *org.nightlabs.jfire.trade.transfer.pay.ClientPaymentProcessor.* A ClientPaymentProcessor must be registered, using the extension point *org.nightlabs.jfire.trade.clientPaymentProcessorFactory.* It can handle payments either completely on the client side (for example, via an EC terminal) or confine itself to collect information for a backend payment. Therefore, it can add an unlimited number of pages to the payment wizard.

Analogical to the payment, there exists a delivery framework. On the client side, you need to use the extension point *org.nightlabs.jfire.trade.clientDeliveryProcessorFactory.* This way you could, for example, send a non-material product directly via e-mail.

Reports with BIRT

If you ever created reports, statistics or printer forms, you might have already heard about the Business Intelligence and Reporting Tools, or BIRT[6]. For a very long time, JasperReports was probably the best choice open-source reporting too. Since its release in November 2004, BIRT has come to be recognized as an interesting alternative. JFire opted for BIRT mainly because of the excellent Eclipse integration. It enriches the report engine by JDO and server-side JavaScript datasources. It not only uses BIRT for the creation of documents like invoices and delivery notes, but also provides an extended report design perspective, which empowers users to create their own reports and to store them on the server. See **Figure 4.**

Fig. 4: Report design perspective



INVOICE

Position	Artikel-Bezeichnung	Betrag
1	Merlot 2003 (California)	$
		$

Dynamic Wizard

A static wizard can quickly be implemented using *org.eclipse.jface.wizard.Wizard* and *org.eclipse.jface.wizard.WizardPage.* Even if you want to skip pages under certain conditions, you only need to override the method *getNextPage()*. But if you want to add and remove arbitrary pages while the wizard is visible, this approach becomes messy, very quickly. To avoid this and keep it concise, JFire breaks the workflow of all dynamic wizards (for example, the payment wizard) down into hops (class *WizardHop*). One hop represents a locally managed workflow part that supports runtime adding or removal of pages without knowing or even touching other hops. Because *WizardHops* can be nested, even complex wizards remain clear. A concrete example is the mentioned payment wizard. The user can distribute one payment to several modes of payment by simply reducing the amount in one page. This adds a new payment page with the remaining amount. Additionally, a mode of payment can require other wizard pages (for the credit card number, for example). Of course, all these pages need to be displayed in the correct order—*org.nightlabs.base.wizard.WizardHopPage* and *org.nightlabs.base.wizard.WizardHop* take care of this.

Exceptions and Other Errors

In case something goes wrong, most programs can send an error report to the developers. These reports are extremely useful, especially when the malfunction rarely occurs. So it's no wonder that JFire implements an application-wide exception handling framework. The plug-in *org.nightlabs.base* provides the extension point *org.nightlabs.base.exceptionhandler*, which facilitates the registration of handlers for classes extending *Throwable*. There is a default handler which sends an error report over a configurable backend. Hence, interfaces to bug-tracking systems are easy to be implemented. So far, however, only an e-mail implementation exists. Unfortunately, it's not yet possible to catch all exceptions, because Eclipse has some internal error handlers that do not yet allow the interception.

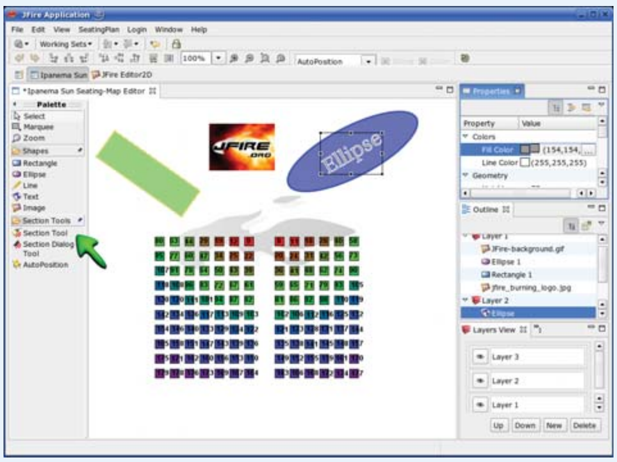
A Picture is Worth a Thousand Words

In order to sell a product, you often need a graphical view. For example, when buying a car's spare part, an interactive construction plan might be very helpful. To prevent the picture from becoming blurry when zooming, and to ease linking between logical and graphical objects, it makes sense to use a vector data model. Definitely, you need an editor that can create and modify this model. JFire contains a GEF-based 2D editor with standard tools like select, rectangle, polygon, ellipse and image. You can easily addyour own tools—in the screenshot shown in **Figure 5**, the 'Section Tools' (green arrow) are extensions provided by the JFire-based Ipanema Seating Plan Editor.

Although the editor uses GEF, all objects can be drawn by Swing, too. This makes coding an applet for a web page easier.

GEF is a slightly oversized for a simple sales-view, where most of its features are not needed; instead, a high priority is set on performance. That's why JFire contains a narrow viewer with small footprint. It can register listeners that are triggered, whenever a graphical object is selected.

Fig. 5: Seating plan editor based on JFire 2D editor



JFire in Practice

Now that the basic set-up and a few details have been made clear, it will be interesting to see how JFire can be used in practice.

Besides the module *JFireSimpleTrade*—which is not only a sample-implementation, but also aims at covering many of the lesser complex application areas—the development at the proprietary software *Ipanema 2* is in full activity. It is a JFire-based ticket distribution system that empowers theatres and cinemas, as well as concert, sport event and other organisers to sell their tickets over various channels of distribution (for example, box offices, call centers, and web shops). Ipanema 2 taps the full potential of JFire and makes it possible to create an extremely scalable distribution network, in which every business partner can (re)sell tickets for all events of all other participants.

Because JFire is Free Software, everyone is free to create sectoral solutions and either create independent distribution networks or to join an existing one. Some extension points already exist for interfacing to other systems in order to make heterogeneous trading platforms feasible.

Free Software in Development

JFire is still in the "pre-alpha" stage. But for someone intending to create another complex solution on it, now might be the perfect time to start. The community is happy to have newcomers.

Resources & References

[1] http://www.jfire.org
[2] http://www.jboss.com
[3] http://java.sun.com/javaee
[4] http://www.jpox.org
[5] http://java.sun.com/products/jdo
[6] http://www.eclipse.org/birt
[7] http://xdoclet.sf.net

Marco Schulze

is CEO of NightLabs Ltd. and deals with ticket distribution and other trading systems for more than 10 years. Additionally, he's committer at JPOX and XDoclet.

Getting Involved with the Eclipse Community

Join, Participate, and Improve the Eclipse Project

By Wayne Beaton

The 'Purposes' section of the Eclipse Foundation's bylaws state that, "The purpose of Eclipse Foundation is to advance the creation, evolution, promotion, and support of the Eclipse Platform and to cultivate both an open source community and an ecosystem of complementary products, capabilities, and services." Open source projects benefit greatly from the participatory nature of the user community. There are many different ways to get involved in the Eclipse community, and the article details how you can participate and contribute to the project.

Introduction

As a developer, your first (and best) way to get involved with the community is to use the Eclipse IDE for your development and explore what the many Eclipse projects have to offer you and your fellow developers. Let your interest areas guide you; for instance, if you are interested in modeling and visual editing, consider the Eclipse Metadata Framework (EMF), Graphical Editing Framework (GEF) and Graphical Modeling Framework (GMF) projects, and if you need to generate reports, consider the Business Intelligence Reporting Tools (BIRT) project. There are many projects; as you get more familiar with the community you'll develop a natural tendency to explore them all.

Explore the Community's Projects

Every project has its own web site that is maintained by the project itself. The web site is a great place to find tutorials, information about the project team, and a project plan. The project plan lays out the goals for the next release of the project. Project web sites are of the form:

http://www.eclipse.org/[project_name]

Example: http://www.eclipse.org/datatools for the Data Tools Project, or
http://www.eclipse.org/higgins for the Higgins Trust Framework project

The projects landing page[1] is a good entry point to help you figure out the project landscape. The Eclipse Wiki[2] and the 'Planet Eclipse'[3] blog aggregator are other great sources of information.

Get Directly Involved

Don't wait too long to get directly involved with the community. Probably the best way to start getting involved with the community is through **newsgroups**.

Seasoned Eclipse users and developers monitor the **newcomer** news group[4] and they are happy to answer your questions. While it is generally better for you to post well thought out questions, the developers who monitor the newcomer group tend to be accommodating and will prompt you for additional information if any is required (or will direct you to a more appropriate newsgroup if necessary).

There are other newsgroups on the server as well; in fact, each Eclipse project has its own newsgroup. If, for example, you have questions about the C/C++ Development Tools (CDT) Project, you can post questions on the **eclipse.tools.cdt** newsgroup. If you're not sure what group would be best to answer your question, post it on the newcomer list and somebody should point you in the right direction.

"Be sure to read "How to Ask Questions The Smart Way" before you post to project newsgroups; well-formed questions tend to be answered while poorly worded ones may not. If your question goes unanswered, consider rewording the question, or adding additional information. Always be polite. Don't be afraid to offer answers to questions from others."

Resources for Members

Perhaps your most valuable resource as a member of the Eclipse community is Bugzilla[6]. It is your main interface to the developers; it is how you contribute directly to projects. As the name implies, Bugzilla is primarily used to report bugs, but it is used for more. You can request new features and even pose questions through Bugzilla (though questions are probably better suited for the news group). You should check if an existing bug report covers your issue before you create a new one. If you have additional information that may add value to an existing bug report, add a comment.

If there is no bug report that covers your issue, create a new one. You can find guidance to help you write a good bug report in the Bug Report FAQ[7]. At a minimum, a good bug report should contain information about your environment and describe the issue you are having as thoroughly (yet concisely) as possible.

Consider including a stack trace, or attaching a screen shot. Try to determine the right target project for the bug. If, for example, you uncover a bug while designing a report, file the bug under the BIRT project. If you misfile the bug, a committer will typically try to reassign the bug (it may take several reassignments for the bug to find the right home). As changes are made and comments are added to the bug you will be sent e-mail notifications so that you can keep up with the progress.

Bug patches and code for new features are also contributed through Bugzilla. If you have code that actually fixes a bug (ideally including appropriate unit tests that test the functionality), you can contribute them as attachments to a bug report. Ultimately, an authorized committer must decide to make your code part of the project's released code. By making consistent, high-quality contributions, a developer may be invited to become a committer on the project.

Increase the Scope of Your Contribution

As you get more involved with the community, you can explore other avenues. If you're building plug-ins, you can make them available to the broader community through the Eclipse Plug-in Central[8]. You might consider writing articles for Eclipse Corner[9] or Eclipse Magazine[10]; or you could start your own blog discussing Eclipse topics and technology and include it on Planet Eclipse. There are many different ways to get involved in the Eclipse community; don't be shy.

Wayne Beaton

is employed by The Eclipse Foundation where he works as an evangelist, spreading the word and helping folks adopt Eclipse technologies. Wayne has extensive experience in object-oriented software development and is a strong proponent of refactoring, unit testing, and agile development methodologies.

Resources & References

[1] Eclipse Projects Page: http://www.eclipse.org/projects
[2] Eclipse Wiki: http://wiki.eclipse.org/index.php/Main_Page
[3] Planet Eclipse Blog Aggregator: http://planeteclipse.org/planet
[4] 'Newcomer' News Group: news://news.eclipse.org/eclipse.newcomer
[5] "How to Ask Questions The Smart Way": http://www.catb.org/%7Eesr/faqs/smart-questions.html
[6] Bugzilla: https://bugs.eclipse.org/bugs/
[7] Bug Report FAQ: http://wiki.eclipse.org/index.php/Bug_Reporting_FAQ

Announcements

Eclipse Mylar 1.0 Introduces Major New Innovation For Developer Productivity

The Eclipse Foundation has released version 1.0 of Eclipse Mylar. Mylar is a tasked-focused user interface that enhances developer productivity by reducing information overload and enabling easy multi-tasking.

Today's IDEs overload developers with information by presenting too many details of a system's complexity. As a result, developers often waste an inordinate amount of time searching and browsing through hundreds of files to find the information relevant to the task-at-hand. Mylar provides a new task-focused user interface that filters Eclipse IDE views to show only the elements relevant to a specific task. Thereby focusing the attention of the developer and reducing the amount of scrolling and searching required to fix a bug or add a new feature.

"The key intuition behind Mylar is that when a developer fixes a bug or adds a feature, they only care about a subset of the system," says Mik Kersten, Mylar project leader. "Mylar's innovation is to make this subset explicit by automatically managing task context on the developers' behalf. The end result is that the developers are able to focus on their programming instead of being forced to constantly find and re-find the information they need to get work done."

Mylar simplifies the task management complexity of a developer's daily work. It enables multi-tasking by providing a task management view that allows developers to easily switch between tasks and share task context with other developers. Mylar also unifies the different task lists a developer typically uses, such as bug tracking and e-mail systems, by integrating these systems into a common Mylar Task List. This unified task list leverages Eclipse's rich client capabilities to allow offline access and change notifications, enabling developers to spend more of their day in the Eclipse IDE where they are productive instead being forced to constantly switch back and forth between the browser and email inbox.

"Mylar is the next killer feature for all IDEs," says Willian Mitsuda, developer at IBOPE, a large Brazilian multinational company. "When I started using Mylar I just couldn't believe how I managed to work on large project without it. This is definitely one of the most important innovations since refactoring."

Mylar already integrates with issue tracking software such as Bugzilla, JIRA and Trac. Organizations are also beginning to integrate or plan to integrate support for Mylar into their products, including CodeGear, Collabnet and Maven.

"In JBuilder 2007 we focused on making it easier to view, understand and manage daily tasks both from a team and an individual developer's perspective and Mylar is an important component in the solution," said Michael Swindell Vice President of Products and Strategy for CodeGear. "Mylar provides us with an innovative means for viewing project responsibilities and a foundation on which we can integrate different lifecycle components into the JBuilder experience and fine tune the user's view of project artifacts."

"Mylar's compelling new approach and CollabNet's collaborative development platform will organize all aspects of the Eclipse developer environment. Together with Subversion, it connects developers directly with the rest of the project team and the development processes used by their organization," said Jack Repenning, architect at CollabNet.

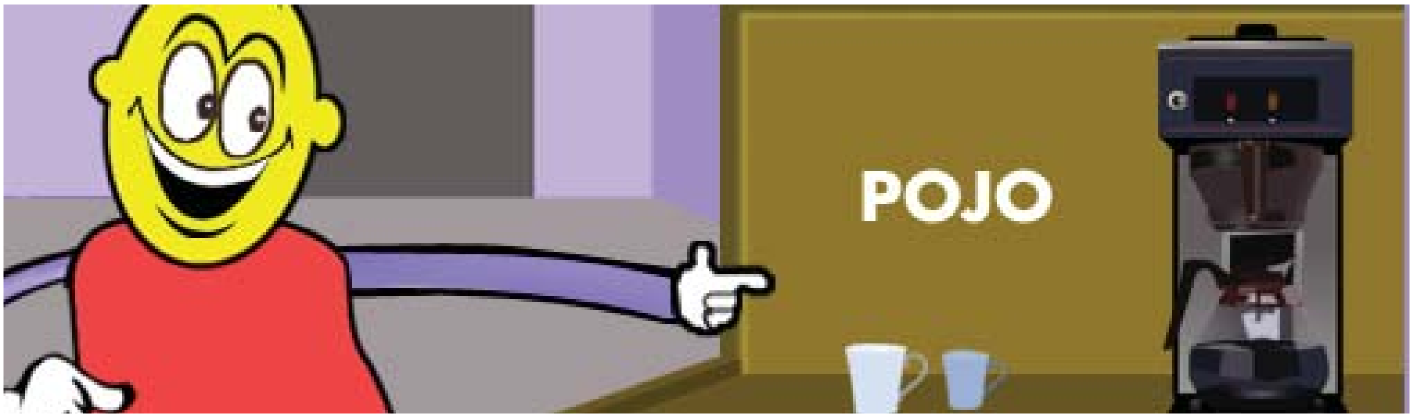
Eclipse Magazine spoke to Mik Kersten a few days before the release of Eclipse Mylar 1.0. Read the interview on Page 36, where Mik elucidates on the key benefits of the Mylar model, what was planned for the 1.0 release, and how Mylar is being converted into a marketable product.

POJO Access Using BIRT

Using BIRT to Access Plain Old Java Objects

By David Trainor and Jason Weathersby

Business intelligence today requires that BI applications draw their information from a variety of disparate sources. Transactional data may be stored in a relational database, and real time information may be scrapped from a web site or captured through a web service or RSS feed. When developing compelling Web-based BI applications, this information needs to be accessed, combined, and displayed to the user in a friendly, unified, format. Fortunately Business Intelligence and Reporting Tools (BIRT) technology makes this possible. In this article we will look at how BIRT can leverage Plain Old Java Objects (POJOs) to access, meld, manipulate, and present these varying forms of data.



Introduction

BIRT is a flexible, 100% pure Java reporting tool for building and publishing reports against data sources ranging from typical business SQL databases, to XML data sources, to inmemory Java objects. BIRT is the Open Source Eclipse Business Intelligence Reporting and Tools[1] project and benefits from the rich capabilities of the Eclipse platform and a very active community of users.

BIRT includes powerful report and chart designers, as well as runtime components for generating and deploying BIRT reports. The BIRT Report Designer provides easy-to-use wizards and point-and-click layout capabilities that are similar to many web page design tools. This unique approach to report creation makes BIRT particularly easy for all levels of developers in an organization. In addition, powerful scripting capabilities and an extensive API allow the Java developer and ISVs to tailor BIRT to specific needs. The runtime component supports the deployment of a variety of reports that, when embedded in an application, adopt the native look and feel of that application. Finished reports are easily integrated with Java Server Pages, servlets, or existing Java applications and leverage existing application server infrastructure.

This article describes the many ways BIRT provides access to information that is not stored in traditional formats such as databases or files. Using BIRT in conjunction with Plain Old Java Objects (POJOs) you can build virtually any report from any source of information. Because BIRT is Java based and designed to be completely extensible and customizable, the possibilities are limitless.

We will discuss how you can use POJOs to retrieve the report's data as well as how you can use POJOs to control the display, layout, or properties of the report based on business logic.

BIRT Events

When rendering reports, the BIRT engine opens an XML report design, connects to the report's data sources, collects parameters, transforms report elements, and generates the output. During this process the engine provides several ways of incorporating external objects and code to modify the default behavior of most report elements. One of these methods is the ability to write handlers in JavaScript or Java at predefined events during the lifecycle of report generation and presentation. In this article we will focus on using JavaScript to collect data and modify the presentation of report designs.

BIRT uses Mozilla's **Rhino** engine to allow scripting at various events during the report generation pipeline. These include events such as Initialize, at the beginning of report generation, to afterRender, when the rendering phase of report creation is complete. These event handlers can be coded in the script editor of the BIRT Report Design Perspective, and can be accessed by selecting a report element and clicking on the script tab located at the bottom of the report layout view. As an example, writing an onCreate event handler script of a Label report element can change the title of report. If the designer wants the title to be tied to a report parameter this script might look something like that shown in Figure 1.

It is important to understand that, not only can these events be written in JavaScript, but the Packages and importPackage keywords provided by Rhino allow direct access to any external Java objects that exist in the classpath. To access a Java object called MyObject you could write JavaScript similar to the following:

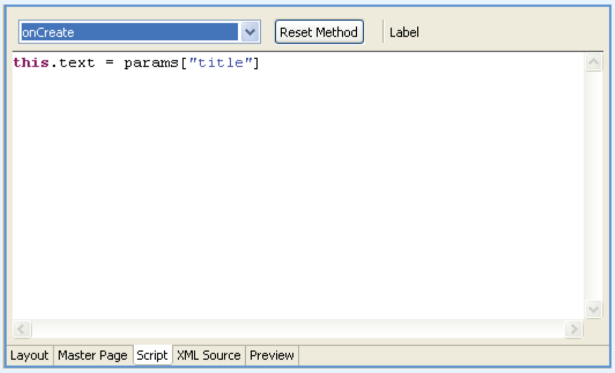
```
importPackage(Packages.my.external.package);

MyObj = new MyObject(params["myConstructParm"]);
this.text = MyObj.myMethod();
```

To build an example report that accesses objects we will first look at what events are available to retrieve data for a report.

Requirements to Deploy and Run Examples

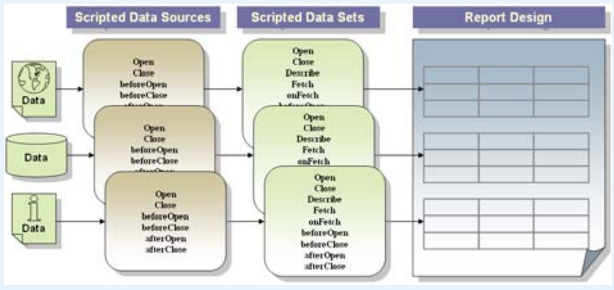
- Eclipse BIRT 2.1.x
- GooglePOJO Report
- Java Source Code and/or Jar File
- Internet connection

Fig. 1: Script Editor



```
this.text = params["title"]
```

Fig. 2: BIRT Scripted Data Flow



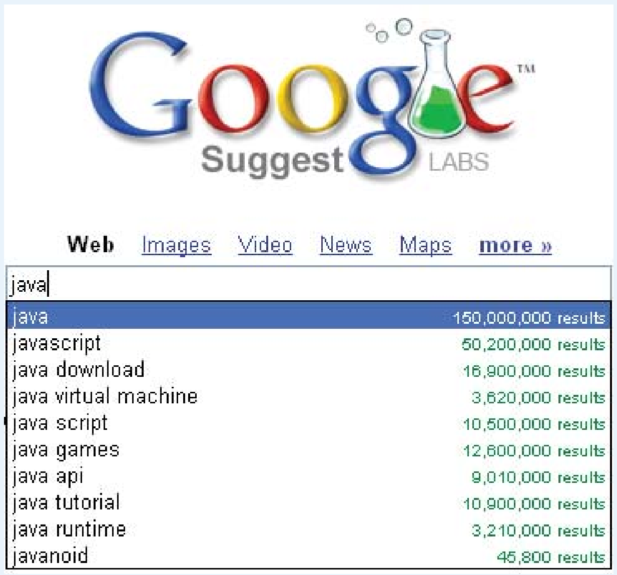
Scripted Data Source / Scripted Data Set Using a POJO

BIRT provides data access drivers for several data sources such as JDBC, XML and Flat Files. As with most constructs within BIRT this list is extensible, allowing designers to implement their own data drivers. Writing a custom driver from scratch can be time consuming, so BIRT provides a specialized driver for generic data access. This driver is called a **Scripted Data Source** and allows event handlers to be implemented to access external objects, files, and databases to collect data. As with all BIRT data sources, Scripted Data Sources are associated with Data Sets, which represent the collected data in tabular format. Scripted Data Sources and Data Sets each have events that can be overridden to retrieve data. In particular, the Scripted Data Source provides two methods that we will use to illustrate data collection and these are open and close. These two events allow us to implement handlers when the data source is opened and when it is closed. We will also use the fetch event of the Scripted Data Set, which is called to retrieve a row of data, and allows us the ability to set the row of data to anything we desire. See **Figure 2**.

We will create a report that implements a handler for the open event of a Scripted Data Source to instantiate a POJO. Then we will override the fetch event of the Scripted Data set to retrieve data from the POJO, and finally override the close event of the Scripted Data Source to do clean up.

The data for our report is going to come from constructing a URL and submitting to Google's Suggest web page in our POJO. (Google Suggest is a utility to help narrow down the choices when keying in a search term. To try Google Suggest go to: http://www.google.com/webhp?complete=1&hl=en). **Figure 3** is a screenshot of Google Suggest in action.

Fig. 3: Google Suggest



java

java	150,000,000 results
javascript	50,200,000 results
java download	16,900,000 results
java virtual machine	3,620,000 results
java script	10,500,000 results
java games	12,600,000 results
java api	9,010,000 results
java tutorial	10,900,000 results
java runtime	3,210,000 results
javanoid	45,800 results

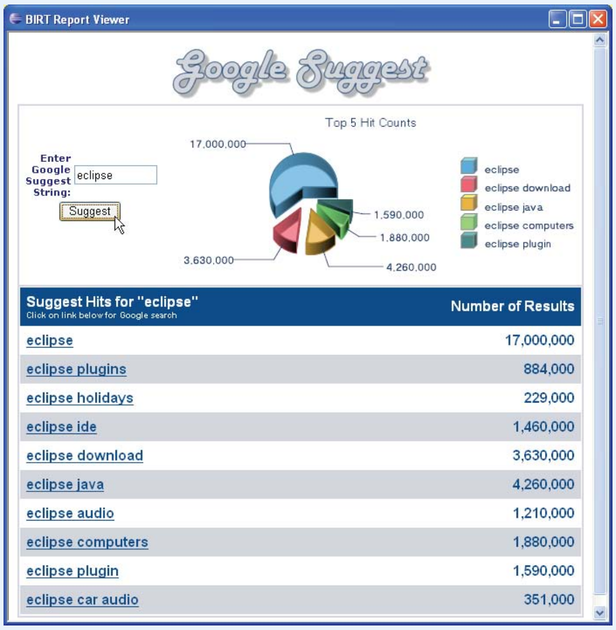
Google's Suggest web page will return the results of the suggest query string that our POJO will parse and then return a vector of results back to the report for rendering. The URL that we will be submitting is constructed as follows:

```
"http://www.google.com/complete/search?hl=en&js=true&qu="
+ suggestQuery
```

The suggestQuery variable is passed into the POJO from a user input box on the report itself. The suggestQuery value is the string that Google Suggest uses to retrieve a list of 10 suggested web links that we will display on the report. **Figure 4** shows the report we are going to build.

The report layout we are starting with is shown in **Figure 5**. Note that the report has a single table with a header, detail, and footer row. The table has two columns: one for the 'Hit' data field and one for the 'Count' data field, which we will describe later. The report has also been set up with a single report parameter that will contain the queryString passed into the report. The report also has a text element that contains HTML and JavaScript code to create a form, as well as a pie chart of the top five hits. The interaction of the form and

Fig. 4: Report built using a POJO as a data source



Suggest Hits for "eclipse" Click on link below for Google search	Number of Results
eclipse	17,000,000
eclipse plugins	884,000
eclipse holidays	229,000
eclipse ide	1,460,000
eclipse download	3,630,000
eclipse java	4,260,000
eclipse audio	1,210,000
eclipse computers	1,880,000
eclipse plugin	1,590,000
eclipse car audio	351,000

the report parameter allows the report to be rerun passing in the form's input control value as the report parameter, which will be used as the queryString to run.

Creating a Scripted Data Source

The starting point for any report is defining a data source. In this example we will be creating a Scripted Data Source that will instantiate our POJO in its open method and do cleanup in its close method. To create a Scripted Data Source, right click on the 'Data Sources' in the 'Data Explorer' view and choose 'New Data Source' from the popup menu. Once the 'New Data Source' dialog appears select 'Scripted Data Source' as the data source type, give it a name—in this case we're calling it srcGoogleSuggest—¬and click Finish as shown in Figure 6.

Creating a Scripted Data Set

Next we need to create a Scripted Data Set based on the Scripted Data source we just created. To do this, right click on 'Data Sets' in the 'Data Explorer' view and choose 'New Data Set' from the popup menu. Once the 'New Data Set'

Fig. 5: Report Layout

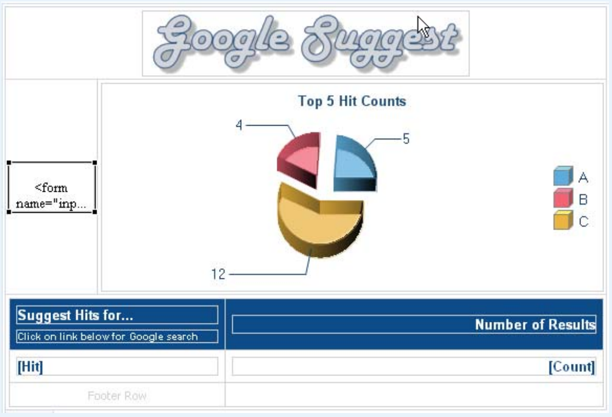
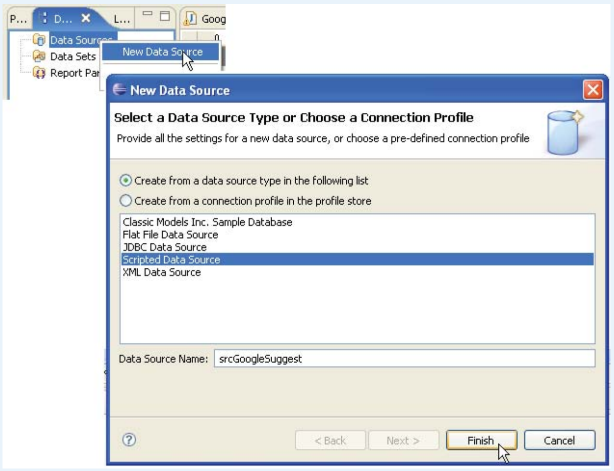


Fig. 6: New Data Source Dialog



New Data Source

Select a Data Source Type or Choose a Connection Profile

Provide all the settings for a new data source, or choose a pre-defined connection profile

(●) Create from a data source type in the following list
( ) Create from a connection profile in the profile store

Classic Models Inc. Sample Database
Flat File Data Source
JDBC Data Source
Scripted Data Source
XML Data Source

Data Source Name: srcGoogleSuggest

< Back | Next > | Finish | Cancel

More Information on BIRT

This article will not go into the details of creating a BIRT project, creating a new BIRT report, basic data source and data set creation, report parameter creation, or placing report elements such as images, grids, tables, etc. To get a basic understanding of report development, please see the tutorials and demos located on the BIRT eclipse web site: http://www.eclipse.org/birt, and read previous articles published in Eclipse Magazine:

- **Introduction to Eclipse BIRT 2.1**: Report writing is the main influence on how an application is judged. Jason Weathersby's article in Volume 1 of Eclipse Magazine explains how to use the enhanced features of Eclipse BIRT version 2.1 to effectively create business reports.
- **BIRT Reporting Interactivity**: Every organization knows the importance of operational reporting to deliver information totheir business users. While traditional operational reports provide a static view into data, BIRT gives end users the ability to drill down to detail, jump to bookmarks, navigate a table of contents, show tool tips, and export data. In Volume 2, David Trainor and Jason Weathersby, demonstrated how to use BIRT to develop rich functionality that gives life and further application value to traditional operation reporting.

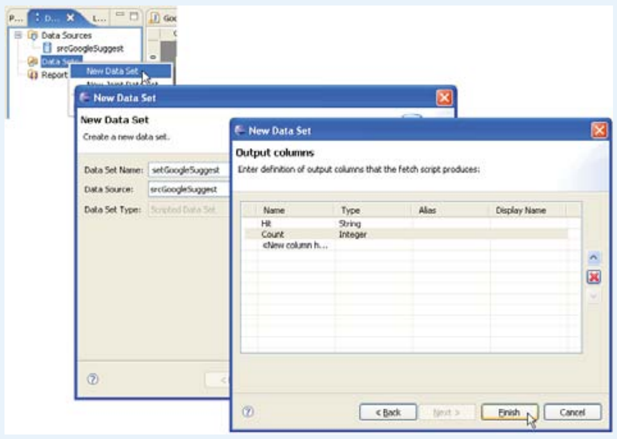


Fig. 7: New Data Set

New Data Set

Create a new data set.

Data Set Name: setGoogleSuggest
Data Source: srcGoogleSuggest
Data Set Type: Scripted Data Set

Output columns

Enter definition of output columns that the fetch script produces:

Name	Type	Alias	Display Name
Hit	String		
Count	Integer		
<New column h...			

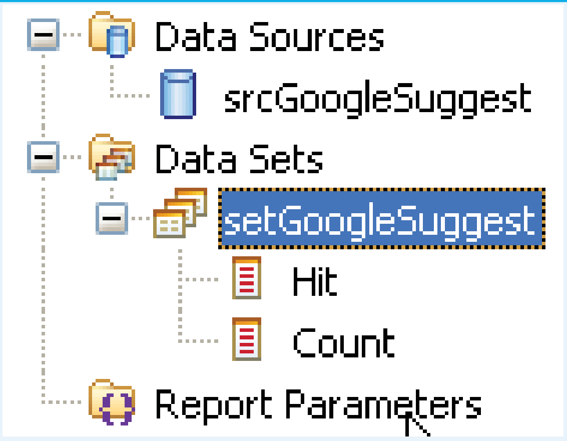


Fig. 8: Data Explorer View

dialog appears give the data set a name (in this case we're calling it setGoogleSuggest). Next, ensure that the name (srcGoogleSuggest) is selected for the data source, and then click 'Next'. The dialog that appears is the place where you define the names, data types, aliases, and display names for the columns or data fields that you will place in the report. In this example we will have two columns, one named 'Hit' and another named 'Count'. The 'Hit' column will contain the suggested web link text and we will give it a data type of 'String'. The 'Count' column will contain the number of results for that particular hit, so we'll give it a data type of 'Integer'. Figure 7 shows the completed dialog boxes.

At this point, the data fields are available in our 'Data Explorer' view, as shown in Figure 8. These data fields can now be dragged and dropped on the detail row of the table on the report.

But if we simply place these fields on the report's table and run the report, there would be no detail data and the report would be empty. This is because we have not defined how these data fields will get populated—this involves overriding the Scripted Data Source/Data Set events to instantiate and call a POJO.

Plain Old Java Object (POJO) Source Code

Let's take a quick look at the POJO we are going to use. Listing 1 includes the bulk of the object's Java code. Note that the Java class name is GoogleSuggestPOJO, and we have a single method named readData that takes a String as a parameter and returns a Vector. The method takes the string passed in and appends it to the Google Suggest URL as the queryString for Google Suggest. The code then opens a stream to the URL and reads the results from the Google Suggest page, parses them, and adds them to the Vector, which it then returns.

Scripted Data Source Event Handlers – open(), close()

The code for the open method of the Scripted Data Source is shown in Figure 9. To retrieve data from the POJO into the data row of our Scripted Data Set, instantiate the GoogleSuggestPOJO class described earlier in the Scripted Data Source's open method. Once the POJO is instantiated, we'll call the readData method, passing in the value of the report parameter and storing the resulting Vector in memory as an array. Finally we'll set up some row counting variables to know when we're finished fetching all the rows in the fetch method of the Scripted Data Set, which we'll look at a little later.

The code for the close method of the Scripted Data Source is shown in Figure 10. All we need to do in the close method is memory cleanup by assigning Java objects to NULL.

Scripted Data Set Method Event Handlers – fetch()

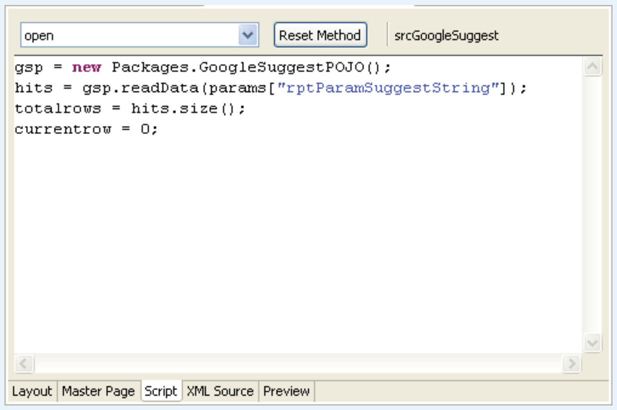
The fetch method in a BIRT report is responsible for populating a data row. A data row is the collection of data fields or columns retrieved from a data source. In a typical JDBC data set, the BIRT report engine handles this automatically. In our example, however, we have to populate

Listing 1

```
public class GoogleSuggestPOJO {
        public Vector readData(String searchString) {
                Vector rtnV = new Vector();

                String qu = searchString;
                boolean isError = false;

                URL u;
                InputStream is = null;
                DataInputStream dis;
                String s;

                try {
                        qu = URLEncoder.encode(qu, "UTF-8");
                        u = new URL(
                         "http://www.google.com/complete/search?hl=en&js=true&qu=" + qu);

                        is = u.openStream(); // throws an IOException
                        dis = new DataInputStream(new BufferedInputStream(is));

                        String resultStart = "new Array(";

                        while ((s = dis.readLine()) != null) {
                                int startHitIndex = s.indexOf(resultStart)
                                                + resultStart.length();
                                int endHitIndex = s.indexOf("),");

                                String countString = s.substring(endHitIndex + 13);

                                int startCountIndex = 0;
                                int endCountIndex = countString.indexOf("),");

                                String resultHits = s.substring(startHitIndex, endHitIndex);
                                String resultCounts = countString.substring(startCountIndex,
                                                endCountIndex);

                                StringTokenizer stH = new StringTokenizer(resultHits);
                                StringTokenizer stC = new StringTokenizer(resultCounts);

                                while (stH.hasMoreTokens()) {
                                        String count = stC.nextToken("\"");
                                        rtnV.add(new String[] { stH.nextToken("\""),
                                                count.substring(0, count.indexOf(" ")).replaceAll(",","") });
                                        if (stH.hasMoreTokens()) {
                                                stH.nextToken("\""); // skip blanks
                                                stC.nextToken("\"");
                                        }
                                }
                        }
                } catch (MalformedURLException mue) {
                        System.out.println("-----------MalformedURLException happened.");
                        mue.printStackTrace();
                        isError = true;
                } catch (IOException ioe) {
                        System.out.println("--------------IOException happened.");
                        isError = true;
                        //below is to provide some values if google suggest messed up
                        for(int i = 0; i < 10; i++) {
```

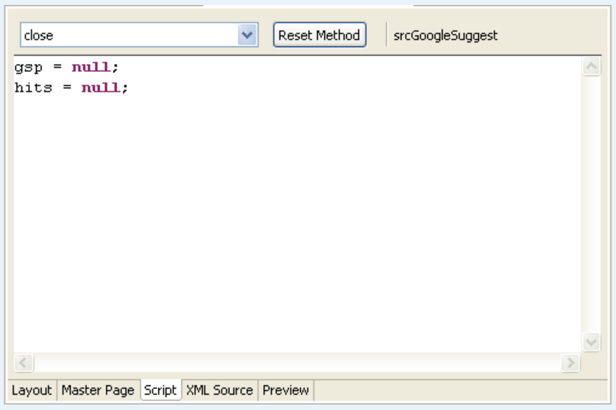
```
                                        rtnV.add(new String[] {searchString,"0"});
                                }
                        } finally {
                                try {
                                        if(is != null)
                                                is.close();
                                } catch (IOException ioe) {
                                        // just going to ignore this one
                                        System.out.println("----------IOException happened again.");
                                        ioe.printStackTrace();
                                }
                        }

                        if (isError) {
                                System.out.println("alert(\"Server Error - Please Try Later\");");
                        } else {
                        }
                        return (rtnV);
                }
        }
```

Fig. 9: Scripted Data Source Open Method



```
gsp = new Packages.GoogleSuggestPOJO();
hits = gsp.readData(params["rptParamSuggestString"]);
totalrows = hits.size();
currentrow = 0;
```

Fig. 10: Scripted Data Source Close Method



```
gsp = null;
hits = null;
```

the data row ourselves—in this case, from the in memory array that we retrieved from our POJO. There are two data fields in our example data row—'Hit' and 'Count'. Figure 11 shows the fetch method of the Scripted Data Set.

As you can see, the first thing we do in fetch is to check whether we are at the end of our data set. Earlier, we retrieved the total number of rows in the open method of the Scripted Data Source and stored it in a global variable. Also in the open method, we defined a global variable to keep track of the current row we are on. In fetch if we are done with all rows we return FALSE, telling the BIRT report engine to stop processing rows. If we are not at the end of our rows, we return TRUE telling the engine to keep processing rows.

After the row count check, we make a call to the get method of the array to retrieve the next row, and assign the first element of the array to the 'Hit' column and the second element of the array to the 'Count' column. This populates the entire data row so we increment the row counter and return TRUE. This processing will take place for each row and will display as an individual detail row in the report's table.

This is all we need to do to complete our report using a POJO to retrieve data. In this case, the data comes from a URL stream from Google. We could just as easily have called Amazon's APIs, read an RSS feed, called a Web Service, or accessed a business EJB.

Figure 12 shows the finished report.

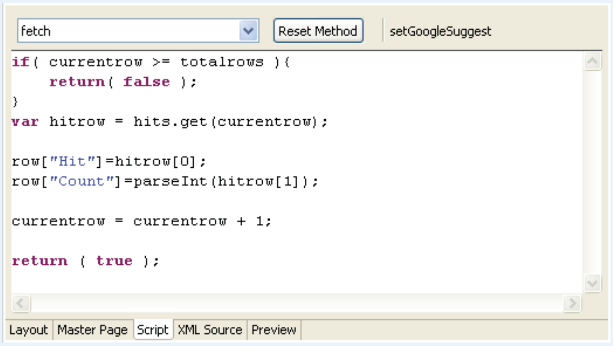
Using Java Objects within Expressions

As shown earlier, BIRT allows POJO access from all events, not just within the data collection. In addition, Java objects can be accessed with BIRT Expressions to control report presentation. As a simple example, we will demonstrate accessing an object stored in the Session to manipulate the report output.

Web applications often store information within the session object to tailor the presentation for the given consumer. As an example, a shopping cart may hold selected items or a financial application may store a buy request before its execution. BIRT can also make use of the session object for visual and logic customizations. Suppose the user's role in the organization is stored within session, a BIRT report can be developed to make use of this field to display one set of information for an analyst and another for a member of the executive team. To illustrate this concept, we can further refine the Google Suggest report, by modifying the visibility property of the chart to only display when a user with the 'Analyst' role is viewing the report. But before demonstrating this, it is important to understand a little more about BIRT expressions.

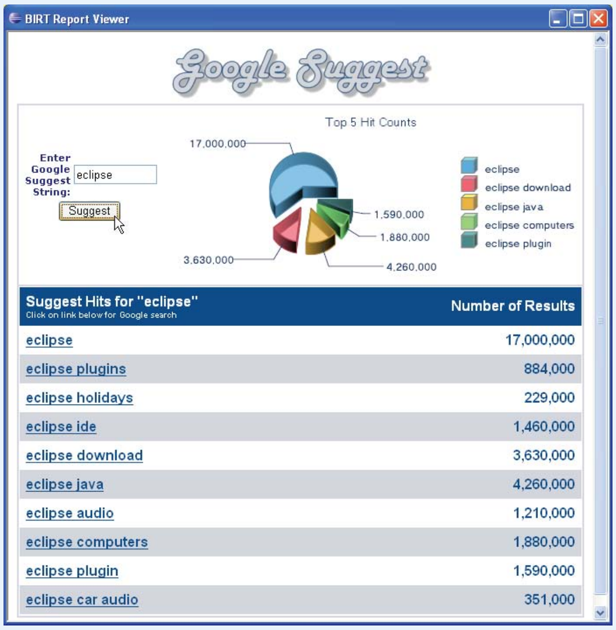
As described earlier, BIRT Event Handlers can be implemented in JavaScript, which uses the Mozilla Rhino engine. This same engine is used within the BIRT Expression Builder, which is used to set properties and values on report items. The fundamental difference between Expressions and Event Handlers is that Expressions are primarily used to return a value, such as an image location or a database URL. Expressions are used throughout BIRT to return values for parameters, hyperlinks, table of contents entries, highlighting and mapping conditionals, and properties. One example of an expression that is useful for our example would be the visibility property of a report element. This property allows hiding a report element either for all report output types or a specific report output. This is useful when reports need to display certain elements in one format, but not in another. A good example of this is the search form within the Google Suggest report, which should not be displayed when rendered as PDF. A visibility expression should return TRUE or FALSE. If the returned value is TRUE, the report item will not be rendered; conversely, if the return value is FALSE, the item will be displayed. See Figure 13.

Fig. 11: Scripted Data Set Fetch Method



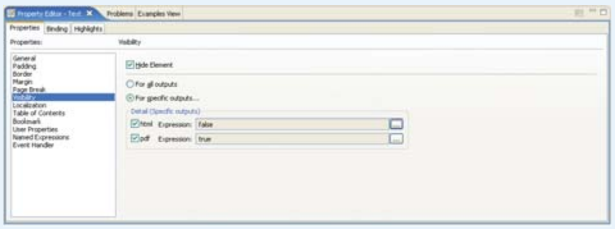
```
if( currentrow >= totalrows ){
    return( false );
}
var hitrow = hits.get(currentrow);

row["Hit"]=hitrow[0];
row["Count"]=parseInt(hitrow[1]);

currentrow = currentrow + 1;

return ( true );
```

Fig. 12: Finished Report



Suggest Hits for "eclipse" Click on link below for Google search	Number of Results
eclipse	17,000,000
eclipse plugins	884,000
eclipse holidays	229,000
eclipse ide	1,460,000
eclipse download	3,630,000
eclipse java	4,260,000
eclipse audio	1,210,000
eclipse computers	1,880,000
eclipse plugin	1,590,000
eclipse car audio	351,000

Fig. 13: Visibility Property



[✓] Hide Element

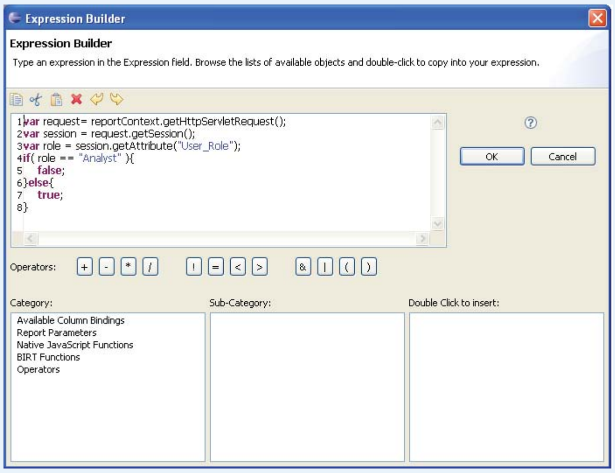
For all outputs
[✓ For specific outputs...]

Detail (Specific outputs)

[✓] html Expression: false

[✓] pdf Expression: true

Fig. 14: Expression Builder

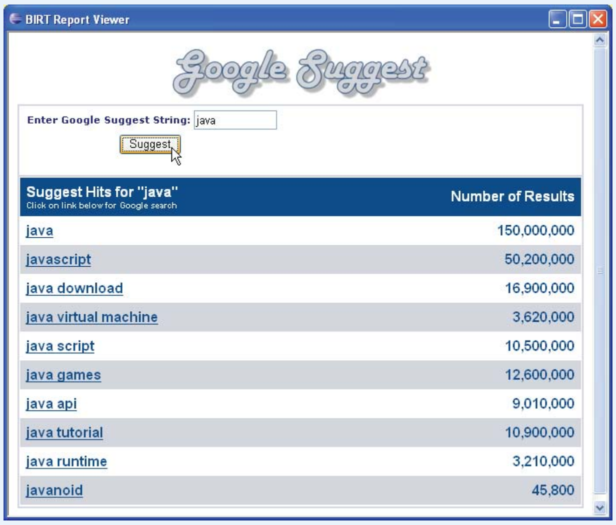


Expression Builder

Type an expression in the Expression field. Browse the lists of available objects and double-click to copy into your expression.

```
var request= reportContext.getHttpServletRequest();
var session = request.getSession();
var role = session.getAttribute("User_Role");
if( role == "Analyst" ){
    false;
}else{
    true;
}
```

Fig. 15: Analyst View of Report – No Chart



Suggest Hits for "java" Click on link below for Google search	Number of Results
java	150,000,000
javascript	50,200,000
java download	16,900,000
java virtual machine	3,620,000
java script	10,500,000
java games	12,600,000
java api	9,010,000
java tutorial	10,900,000
java runtime	3,210,000
javanoid	45,800

Continuing with the example, if a session attribute contains a variable named User_Role, an expression for the chart visibility could be built as follows to only display the chart for the 'Analyst' role. See Figure 14.

When the report is executed the chart will be displayed if the session attribute User_Role is set to 'Analyst'. If the value does not exist or is set to something else the chart will not be displayed. See Figure 15.

Summary

As you can see, using BIRT's Java foundation we are able to leverage a wealth of information that exists in formats other than the traditional database. Using Plain Old Java Objects, BIRT allows you to build virtually any report from any source of information. In this article we've only looked at a couple of ways to use these capabilities, but there are countless others. We encourage you to download BIRT today and experiment with your own information delivery.

Resources & References

[1] Eclipse BIRT Project Home Page: http://www.eclipse.org/birt
[2] Eclipse BIRT 2.1 Download: http://download.eclipse.org/birt/downloads

Jason Weathersby

is the BIRT Evangelist at Actuate Corporation and a member of the Eclipse Business Intelligence and Reporting Tools (BIRT) Project Management Committee (PMC). Jason has over 15 years experience in the software development field, ranging from real time process control to business intelligence software. At Actuate, Jason is currently responsible for educating the Open Source community on BIRT and encouraging its adoption.

David Trainor

is the BIRT Technical Marketing Manager at Actuate Corporation. David has 20 years ex-perience in the software development field with specific technologies such as J2EE, .NET, Web Services, and Business Intelligence. At Actuate, David is currently responsible for delivering solutions and support to the sales force for Actuate technologies including BIRT, Business Report Studio, and iPortal.

Mylar Makes Programmers More Productive

Interview with Mik Kersten, Lead of the Mylar Project

By Eclipse Magazine

The goal of the Mylar project is to evolve a focused user interface and task management for the Eclipse platform. At the core of Mylar is a mechanism that that makes our interaction with a system explicit. Mylar has been created by Mik Kersten as a part of his PhD thesis, supervised by Gail Murphy and supported by the Software Practices Lab at UBC, the IBM Centre for Advanced Studies, and NSERC. Mik spoke to Eclipse Magazine about the key benefits of the Mylar model, what we can look forward to in the 1.0 release, and how Mylar is being converted into a marketable product.

Eclipse Magazine (EM): There are several research tools that provide help to the programmer to explicitly declare the elements related to work task. How is Mylar's offering different from other models? What are the key benefits of the model?

Mik Kersten (MK): Mylar has two key features that, when combined, distinguish it from previous approaches. First, it centers our interaction with an IDE around tasks. Similar to how IDEs have made source files a first-class part of the interaction—enabling easy browsing, searched, and shared—Mylar has made tasks a first class part of the IDE. Mylar's task management integrates both personal tasks and shared tasks that reside in repositories such as **Bugzilla, JIRA**, and **Trac**. Once integrated, the editing and navigation between tasks and source code become seamless. For example, as soon as a bug report is assigned to me in Bugzilla, a reminder pops up on my screen, directs me to the new bug report in my Task List, where I can edit it directly and click on entries in the stack trace to take me to the source code.

Once tasks are integrated, Mylar's automatic context management works to reduce information overload in the IDE by showing you what you are working on. We believe that that less is more—when working on a large system you only need to see the parts of the system related to your task. Instead of burdening you with defining those parts manually, as many other tools have done, Mylar watches your programming activity to build up a task context automatically. It actively maintains this context so that it shows you only the most relevant parts, and does so in a very predictable way with a degree-of-interest ranking. This task context is then used to focus the entire Eclipse UI on the task-at-hand. For example, instead of showing hundreds of elements, the Package Explorer, Mylar only shows you those that you have selected and edited. And the best part is that these contexts can be stored and shared with a single click. If a new patch comes in while I'm working on adding a feature, I simply activate the task with the patch, retrieve the patch and context from the Bugzilla report, and see exactly what the contributor saw when they created the patch. Once I have reviewed the patch, it only takes one more click to get back to the context of the previous task, where Mylar will restore all the editors that were previously open, reopen the perspective used for that task, and bring me right back to where I left off. We often work on multiple tasks a day, with collaboration and changing priorities being a constant source of distraction. Mylar's support for task contexts makes this kind of multitasking much easier.

EM: In the first prototype, a single Mylar context existed per Eclipse workspace, but programmers have indicated a desire to extend Mylar's model to support multiple tasks. Are there any plans of expanding the Mylar model to incorporate this?

MK: We tested this early version, Mylar 0.1, on a small group of professional programmers at IBM. Both our own use of Mylar and the user study feedback indicated that supporting multi-tasking was key. So we combined it with the Bugzilla plug-in that had been created at UBC for a research project called **Hipikat**, and that gave us enough of a platform to test this new idea of task context out. We created Mylar 0.2 for our own bootstrapping and presented it at EclipseCon 2005 to gather subjects for a user study, then made a private release of Mylar 0.3 for anyone willing to participate in the study. Mylar 0.3 had the implementation of task context that we have today, and validated with statistical significance that Mylar makes programmers more productive. A paper detailing the results of this study will soon be available for download from the Mylar homepage.

Making Mylar Marketable

Several companies that provide issue trackers have expressed interest in integrating with Mylar, so they do not have to reinvent the wheel in creating well-integrated task management for Eclipse. Our group at University of British Columbia (UBC) will form a company to provide the consulting and training services requested by these companies. We also plan on providing services to companies wanting to integrate or embed Mylar with their Eclipse-based tools. The team get many requests from users wanting Mylar integration with commercial operating systems, e-mail clients, and other information management tools, and we plan on making connectors for these tools available as well.

EM: Although most users of Mylar liked what the views exposed, there has been a mixed feedback with regards to the highlighting scheme. Some have found it visually loud. Are there any changes made with regard to this in the later releases?

MK: The highlighting of the degree-of-interest seemed like a good idea at first, because varying the background color of elements could be used to indicate the spectrum of interest levels in the task context. However, putting too much color in the UI is another way of overloading the user with information. So we took the less is more approach again and simplified this scheme to made the most interesting elements appear in bold font, as a sort of implicit bookmark that we call a landmark, and to make uninteresting elements appear gray.

EM: Mylar won you $10000 in "The Extra Chapter Challenge" contest. Mylar can also be used by non-programmers to keep track of the user's files and Internet searches. Can you elaborate on how individual computer users can benefit from Mylar?

MK: In addition to the user studies that we ran on programmers, we conducted another study where we took the core parts of Mylar and integrated them into a prototype file and web browsing application. We were very pleased to see that explicit tasks and context can also support more generic kinds of knowledge work, and that when given tool support for task context most of our study subjects voluntarily worked in a task-focused way.

EM: You also made a statement that several companies have expressed interest in Mylar. Which are these companies, and how are you planning to convert Mylar into a marketable product?

MK: Mylar is and always will be an open source framework, and provide integration with the Eclipse SDK and task repository connectors supported by open source contributions. What we want to do as a project is to ensure that Mylar connectors are also created for the broad range of commercial task and source repositories in order to support the many users who need these connectors to work with Mylar. Several companies that provide issue trackers have expressed interest in integrating with Mylar, since it ensures that they do not have to reinvent the wheel in creating well-integrated task management for Eclipse. Our group at University of British Columbia (UBC) is in the process of forming a company that will provide the consulting and training services requested by these companies.

In addition, Mylar's context framework can be extended to better integrate with programming tools, such as those for JEE development. So we also plan on providing services to companies wanting to integrate or embed Mylar with their Eclipse-based tools. Finally, we get many requests from users wanting Mylar integration with commercial operating systems, e-mail clients, and other information management tools, and plan on making connectors for these tools available as well. I expect the tools that we build on Mylar, as well as the integration and embedding of Mylar by other companies will drive improvements in the Mylar framework and core tools.

While these commercialization will extend the reach of the Mylar technology, the Mylar open source project will continue

Mik Kersten is the lead of the Mylar eclipse.org project, and a committer on the AspectJ and AJDT projects. While working Xerox PARC Mik created the IDE support for AspectJ, and plug-ins for JBuilder, NetBeans, VisualStudio, and Emacs. He is now completing his PhD at the University of British Columbia, and focused on helping Eclipse reduce information overload by making the context of the tasks we work on explicit.

as an openly developed, transparent, and very permeable project. We're creating a new technology on this project, and integrating this technology with the very diverse ways that people work could not happen so quickly if not for the couple of thousand of submitted bug reports that have helped define the way Mylar should work, and the couple of hundred patches that have helped us get it there.

EM: Some of our readers wrote in to find out what plan the Mylar project has, in terms of making issue tracker, project management, and development environment work seamlessly together?

MK: Mylar integrates with the developer's end of project management by integrating issue trackers. For example, if you manage projects with target milestones, you can create queries in your Task List to make those milestones explicit. Having a UI for working with the tasks integrated with Eclipse is key because development tasks are closely related to source code, so we need rich support for navigating between the two. In contrast, project planning is often easy to do independently of the source, so a Web UI for project planning often suffices, especially since the output of planning sessions immediately shows in developers' IDE via our synchronization with the task repository. We are focused on streamlining the developer's experience, and ensuring that we integrate with project management and project planning tools.

EM: Apart from Mylar there are a lot of related projects that are either building on or expanding Mylar. Can you throw some light on a few of them like the Bug Triage?

MK: Mylar provides three extensible frameworks—the Task Framework for integrating with repositories such as Bugzilla, the Context Framework for extending the task context model to other kinds of resources, and the Monitor Framework for running user studies. Parts of the frameworks can be used headless, for example, in a server side application.

The Bug Triage project from the University of British Columbia (UBC) has taken the Tasks Framework and Bugzilla Connector, and used it as a Java API for accessing Bugzilla. They then implemented a very neat server side tool that automatically assigns bugs to people based on past assignments and other heuristics.

EM:You have experience in building IDEs for AspectJ, the AOP extension to Java. Eclipse and NetBeans both claim to be universal tool platforms. Can you compare the two as extensible Java IDEs?

MK: Yes, and in addition to Eclipse and NetBeans, I have built plug-ins for other IDEs including JBuilder, VisualStudio, IDEA and Emacs. Beyond the usual Swing/SWT debates, what I have learned from those experiences is that there is a huge spectrum in how easy it is to build extensions to an IDE. Ease of extensibility requires the IDE source code to be available and easy to navigate, because the sources are the best documentation on how to use the APIs, especially when you are using them in ways that were not pre-conceived by the IDE developers. Eclipse's JDT makes browsing the structure of the Eclipse platform itself very easy. Another very big factor is the modularity of the IDE. The better the modularity, the easier the IDE is to extend. For example, Eclipse's abstractions for views, filters, and decorators mean that we can consistently focus every view on task context. The OSGi-based plug-in mechanism enables us to loosely couple to parts of the platform, and to make our own plug-ins easy to extend. Finally, the ability to self-host, which Eclipse's PDE provides, is key to productivity when building plug-in extensions. Other IDEs have various combinations of these features. However, the fact that Eclipse developers have been eating their own dog food for so long has resulted in very mature and streamlined platform and tool support for plug-in developers. I can safely say that it would be either extremely hard or impossible to have implemented Mylar on anything other than Eclipse without requiring major changes from the underling platform.

EM: Since your presentation at EclipseCon, over 75 people signed up for the summer user study, which made up the initial user community. What are the inputs received from the study? How has this helped fine-tune the later releases of Mylar?

MK: That initial user study was critical in helping us understand how to integrate this new idea of task context with Eclipse, and ongoing feedback is continuing to do so. We have learned a lot about how thoroughly and consistently task context needs to be integrated with the IDE. For example, the idea that change sets—the incoming and outgoing version control changes—are a lot easier to use when automatically managed with task context came from the user study. An interesting surprise that resulted from the study was that while we thought it was key that Mylar's support for allowing more than one task to be active was not very useful—and it is now gone from the UI—users needed the ability to multi-task very easily, but not the ability to work on two tasks at the same time. In addition to validating the technology, what the user studies and the ongoing feedback provide is input on which features to add and which to avoid or take away.

Mylar and NetBeans as Extensible Java IDEs

Ease of extensibility requires the IDE source code to be available and easy to navigate, because the sources are the best documentation on how to use the APIs. Eclipse's JDT makes browsing the structure of the Eclipse platform itself very easy. Another very big factor is the modularity of the IDE. The better the modularity, the easier the IDE is to extend. Finally, the ability to self-host, which Eclipse's PDE provides, is key to productivity when building plug-in extensions. Although other IDEs have various combinations of these features, the fact that Eclipse developers have been eating their own dog food for so long has resulted in very mature and streamlined platform and tool support for plug-in developers.

EM: Additional Mylar tools extend the core model and UI facilities and add tool-specific interest encoding and search facilities. The future extensions are supposed to include debugging support, Dynamic Help integration, interest model visualization and C/C++ development support. Can you throw some light on the developments in these aspects?

MK: Our goal is to provide a complete implementation of task context for the Eclipse SDK. We're almost there, but still need to extend the Debugging integration; for example, to support automatically enabling and disabling break points with task activation. Help integration has not been requested much, but given the size of the Eclipse docs it could benefit from focusing based on task context. We will support extensions for other languages, such as C/C++, being built on top of Mylar and consider the Java and XML support in Ant and PDE to be the reference implementation.

EM: What are the key limitations of Mylar? What are the challenges that the users have brought to light?

MK: The key challenge has been integrating a new kind of UI and interaction technology in a way that integrates seamlessly with the existing IDE. IDEs have never had a first class notion of user-defined tasks, and the task context idea is totally new.

All of the Mylar implementation since the initial 0.1 prototype has been done with Mylar, and this bootstrapping has helped streamline the integration. But without a lively user community it would have biased it around the work practice of the committers. As a result we thrive on the continuous stream of bug reports that we get on how to better integrate and improve Mylar. Thanks to Mylar's facilities for working with bug reports, creating patches, and sharing contexts to facilitate reviewing patches, we have made it very easy to contribute to the project. So getting around this challenge and reducing the time it takes to go from a new technology to a usable tool in this short amount of time has been made possible by the very tight feedback loop that we have for users with our Bugzilla integration, and by facilitating contributions.

EM: Mylar 0.6.1 was made available since July. The first release candidate for Mylar is slated for November 24, with the stable 1.0 rollout planned for early December. Can you take us through the future roadmap for Mylar, and what can be expected in the 1.0 release?

MK: I want to mention that we have considered all the weekly builds of Mylar since 0.1 to be 'stable', and our practice is never to destabilize. This makes it possible for us to continually eat our own dog food—Mylar committers and contributors work bootstrapped straight from the latest version in CVS. But what has been evolving rapidly over the 12 months since our first public release is the Tasks and Context frameworks, the flexibility and integration of the support for task management, and the consistent integration of task context throughout the Eclipse SDK. In addition, the Bugzilla connector now provides all the core functionality available via the Web UI, the Trac connector has nearly all of the facilities provided by Bugzilla such as rich editing and offline support, and the JIRA connector will soon have this support as well.

Mylar 1.0 will mark a level of maturity in our continually improving UI for provides tasks management and reduces information overload in Eclipse. Since we expect our user community to continue growing at a rapid pace, priorities post 1.0 will largely be determined by the ongoing feedback and needs of those adopting the 1.0 release.

FastTrack

A Tracker Plug-in for the Eclipse IDE

By Antonin Pokorny

FastTrack is a FREE tracker plug-in for Eclipse. It provides basic issue tracking, plus basic project planning and team collaboration, all deeply integrated into the Eclipse IDE.

Introduction

The capability to assign and track the process of Work Items is integral to collaborative software development. "Work Item" is a general term that refers to a specific granular unit of work which must be accomplished in order to successfully realize a particular software objective (a project or release, for example). One commonly used term is "issue". However, this term is a bit vague because it can actually refer to related, yet different types of work which it can be advantageous to track separately.

In FastTrack, "Work Items" can be user-defined to fit a development organization's process, but by default they will be typed as follows:

- Requirements
- Tasks
- Change Requests

Examples of additional types that could be defined might include:

- Defects (or Bugs)
- Feature requests
- Tips for Improvement

Requirements are a less granular type of Work Item because the resolution of some combination of other types (Tasks, Change Requests, etc.) will generally be required in order to resolve a Requirement.

There are many open-source and commercial "issue" tracking systems, but none of them provide one additional and highly desirable capability which FastTrack has: **traceability**. This is the capability to trace the development process from the initial requirement, through any modifications of the requirement, on into the tasks, change requests, etc. that were done to implement the requirement, right down to the source code that was produced to accomplish each item. The reverse is also possible: to trace, for example, a defect backward through the process to the original requirement.

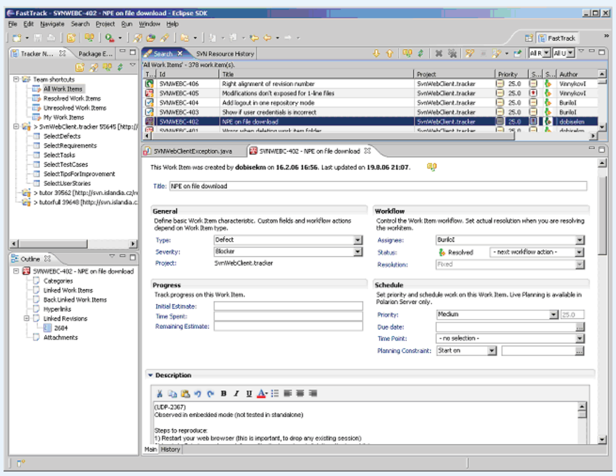


Fig. 1: The Work Item Editor in FastTrack Perspective

FastTrack - SVNWEBC-402 - NPE on file download - Eclipse SDK

'All Work Items' - 378 work item(s).

T...	Id	Title	Project	Priority	S...	S...	Author
	SVNWEBC-406	Right alignment of revision number	SvnWebClient.tracker	25.0			VinnykovI
	SVNWEBC-405	Modifications don't exposed for 1-line files	SvnWebClient.tracker	25.0			VinnykovI
	SVNWEBC-404	Add logout in one repository mode	SvnWebClient.tracker	25.0			BurloI
	SVNWEBC-403	Show if user credentials is incorrect	SvnWebClient.tracker	25.0			BurloI
	SVNWEBC-402	NPE on file download	SvnWebClient.tracker	25.0			dobisekm

This Work Item was created by **dobisekm** on **16.2.06 16:56**. Last updated on **19.8.06 21:07**.

Title: NPE on file download

General
Define basic Work Item characteristic. Custom fields and workflow actions depend on Work Item type.
Type: Defect
Severity: Blocker
Project: SvnWebClient.tracker

Workflow
Control the Work Item workflow. Set actual resolution when you are resolving the workitem.
Assignee: BurloI
Status: Resolved - next workflow action -
Resolution: Fixed

Progress
Track progress on this Work Item.
Initial Estimate:
Time Spent:
Remaining Estimate:

Schedule
Set priority and schedule work on this Work Item. Live Planning is available in Polarion Server only.
Priority: Medium 25.0
Due date:
Time Point: - no selection -
Planning Constraint: Start on

Description
(UDP-2367)
Observed in embedded mode (not tested in standalone)

Steps to reproduce:
1) Restart your web browser (this is important, to drop any existing session)

File Edit Navigate Search Project Run Window Help
FastTrack
Tracker N... Package E...
Search SVN Resource History
Team shortcuts
All Work Items
Resolved Work Items
Unresolved Work Items
My Work Items
> SvnWebClient.tracker 55645 [http://
SelectDefects
SelectRequirements
SelectTasks
SelectTestCases
SelectTipsForImprovement
SelectUserStories
> tutor 39562 [http://svn.islandia.cz/
> tutorfull 39648 [http://svn.islandia.c
Outline
SVNWEBC-402 - NPE on file download
Categories
Linked Work Items
Back Linked Work Items
Hyperlinks
Linked Revisions
2684
Attachments
SVNWebClientException.java
SVNWEBC-402 - NPE on file download
Main History

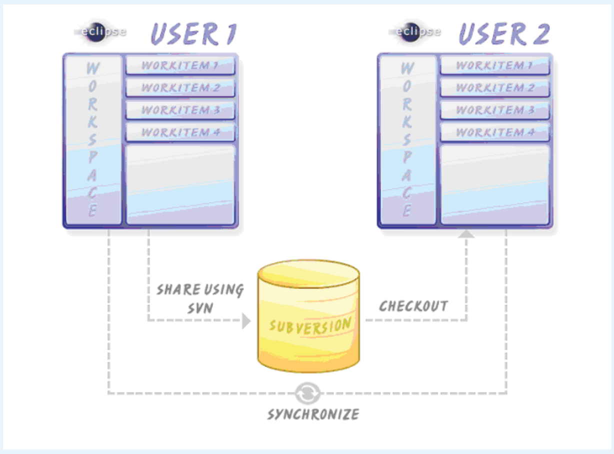


Fig. 2: Team development with the Subversion server

Architecture

Although the term "Tracker" is probably more likely associated with relational database systems, it is not case of Fast-Track. During design phase of FastTrack, there was maximum focus on simplicity, optimal integration into the Eclipse IDE and compatibility with existing Eclipse concepts.

This approach resulted in an innovative architecture, based on a representation of every Work Item as a common XML file stored in the Eclipse workspace.

Today, Subversion or CVS are often used as a version control system providing file versioning and sharing. This is a standard approach for sharing and version control of source code. In FastTrack, Work Items are treated in the same way.

File sharing and versioning is standard functionality of Subversive – the Subversion Team provider for Eclipse which is bundled with FastTrack.

Project administration and Work Items tracking is deeply integrated into the Eclipse IDE in an excellent way thanks to file oriented FastTrack architecture.

File Sharing Through the Subversion Server

FastTrack guarantees team collaboration and Work Item sharing among users in the simplest way possible. The XML files representing individual Work Items, are stored in the Subversion repository – to be precise in a special folder within the particu-lar development project.

Other users are notified about newly created Work Items and about modifications of already existing Work Items via the standard Subversion system mechanisms. This means that if User No. 1 performs a modification, commits the change into the repository, User No. 2 will perform synchronization or possibly an update in a standard way using the function "Synchronize with Repository" – the modifications will become visible in her/his development environment.

Tracker

FastTrack is an easy, but very efficient tracker solution for individual programmers or small programming teams using the Eclipse IDE. All necessary functions are available directly in the IDE, including:

- Searching within the Work Item contents
- Storing of frequently used queries
- Linking of Work Items
- Editing Work Items
- Adding comments
- Adding attachments

The following paragraphs briefly describe major FastTrack functionality.

Searching

FastTrack provides searching of Work Items similarly to the standard search support in the Eclipse IDE, except that searches are focused on artifacts administered by FastTrack. The search can be easily restricted, using the criteria like 'all'/

'created by me'/'assigned to me', or ‚resolved'/'unresolved'. FastTrack provides the user with a query language support-ing the definition of complex queries, using logical operators, or other search criteria for individual Work Item attributes.

Shortcuts

Shortcuts in FastTrack represent a stored query and the configuration of the rendered result, i.e. sorting, column order, and column widths. Shortcuts are displayed in the navigator enabling quick access to the respective sets of Work Items.

Shortcuts can be defined on the project level or on the personal level for each user. Shortcuts are stored in the Subversion repository.

Outline

The Work Item Editor usually shows relatively huge variety of information, and scrolling the pane to reach the comments or the list of linked Work Items is often unavoidable. Therefore FastTrack provides an Outline View, which easily and quickly exposes the major structural elements of a Work Item.

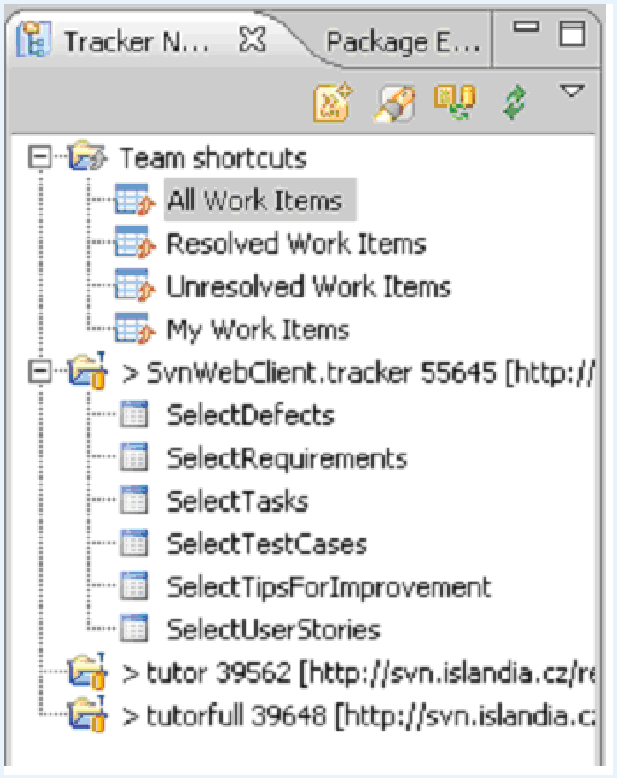
Work Item Editor

The Work Item editor is used to view and edit basic work item properties and also to control all other operations like switching workflow status, adding links, comments, etc.

Editors of basic Work Item attributes are located in the upper section of the Editor. These are, specifically, the Title, De-scription, Assignee and other attribute groups according to their meaning, i.e. Workflow, Time tracking, Scheduling. See the Figure 1.

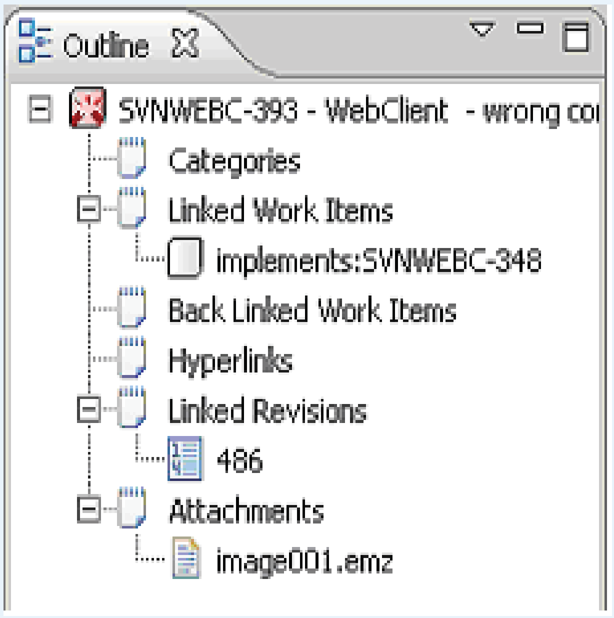
A very important feature of FastTrack is the ability to link Work Items to each other. For example it is possible to create a link between a Task and a Requirement and thus express that the Task represents an element of work leading to implementation of the Requirement. As already mentioned, FastTrack does not limit either types of Work Items or link types among them. Linking can be adapted to the enterprise specific requirements by the FastTrack configuration.

Fig. 3: Shortcuts with Predefined Queries



Team shortcuts
All Work Items
Resolved Work Items
Unresolved Work Items
My Work Items
> SvnWebClient.tracker 55645 [http://
SelectDefects
SelectRequirements
SelectTasks
SelectTestCases
SelectTipsForImprovement
SelectUserStories
> tutor 39562 [http://svn.islandia.cz/re
> tutorfull 39648 [http://svn.islandia.c

Fig. 4: Outline View



SVNWEBC-393 - WebClient - wrong co
Categories
Linked Work Items
implements:SVNWEBC-348
Back Linked Work Items
Hyperlinks
Linked Revisions
486
Attachments
image001.emz

Workflow

Workflow support is one of the essential functions of each tracker. FastTrack enables users to define lists of possible states and the transitions allowed between them. This way, FastTrack allows only such workflow transitions as are allowed by the workflow definition.

History

Because FastTrack uses Subversion for its underlying data storage, a complete history of all operations is always available. FastTrack provides a view on old Work Item states. Historical states can be compared with each other in the user interface, which is visually aligned to the Work Item structure.

Commit Office – No Delayed Programming Tasks Any More

FastTrack introduces a brand new commit dialog called Commit Office. It is a set of several integrated pages aggregating use-cases typically performed during commit. Besides a standard commit message page, new pages for linking of Work Items and for validation of commit policies are provided.

Standard Commit

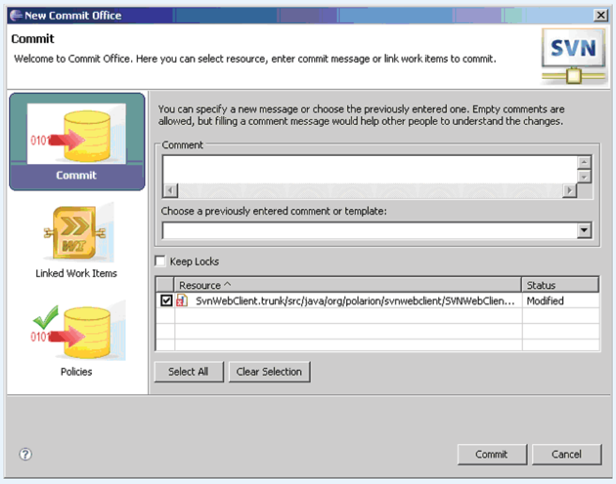
The main view of the Commit Office Environment is, in fact, exactly the same as the Subversive (Subversion client) com-mit dialog. This page is typically used for composing the Commit message for every Commit.

Work Item Linking

The typical usability problem of IDEs and Tracker systems is the fact that developers spend majority of time in the IDE, but have to switch to another environment in order to update Work Items, perform workflow operations, link Revisions, etc.

That's why FastTrack provides directly, within the Commit dialog, a specific Search area for Work Items, which are pre-configured for the searching criteria "assigned to me". Work Items, which are related to Commits, can be selected by clicking on checkboxes within this table. If applicable, it is possible to edit the found Work Items directly in the Commit Office, to append comments, to switch the workflow state – for example into the state "Resolved" – or to edit all Work Item attributes in general.

Fig. 5: Commit Office



New Commit Office

Commit

Welcome to Commit Office. Here you can select resource, enter commit message or link work items to commit.

You can specify a new message or choose the previously entered one. Empty comments are allowed, but filling a comment message would help other people to understand the changes.

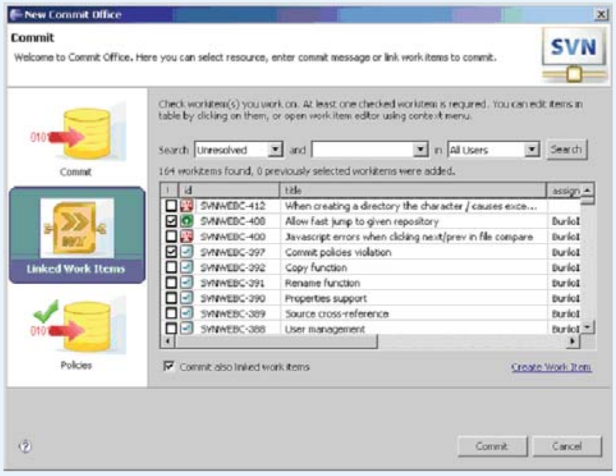
Comment

Choose a previously entered comment or template:

☐ Keep Locks

	Resource ^	Status
☑	SvnWebClient.trunk/src/java/org/polarion/svnwebclient/SVNWebClien...	Modified

Fig. 6: Work Items Linking



New Commit Office

Commit

Welcome to Commit Office. Here you can select resource, enter commit message or link work items to commit.

Check workitem(s) you work on. At least one checked workitem is required. You can edit items in table by clicking on them, or open work item editor using context menu.

Search Unresolved and in All Users

164 workitems found, 0 previously selected workitems were added.

!	id	title	assign
☐	SVNWEBC-412	When creating a directory the character / causes exce...	
☑	SVNWEBC-408	Allow fast jump to given repository	Burlol
☐	SVNWEBC-400	Javascript errors when clicking next/prev in file compare	Burlol
☑	SVNWEBC-397	Commit policies violation	Burlol
☐	SVNWEBC-392	Copy function	Burlol
☐	SVNWEBC-391	Rename function	Burlol
☐	SVNWEBC-390	Properties support	Burlol
☐	SVNWEBC-389	Source cross-reference	Burlol
☐	SVNWEBC-388	User management	Burlol

☑ Commit also linked work items

Commit Policies

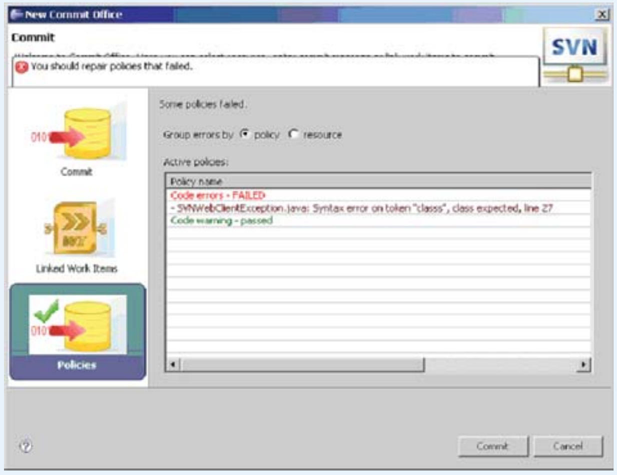
The Eclipse IDE provides the user with a support for programming policy checks on Java source codes. Any violations found are high-lighted in the source code editor and presented in the Problems view. The Commit Policies in FastTrack use the same information and perform a check to see whether the policies are met before every Commit. In case of a policy violation, the user will be shown a warning message. Occurrence of a policy violation doesn't prevent the commit, but the user does have to enter a reason for committing changes with failed policies. This information is stored in the commit message and can be easily found later.

Test the FastTrack

FastTrack can be used as a single-user version or to support a small programming team having a Subversion repository in place.

The installation procedure in case of a single user version is quite simple. All that is needed is the Eclipse IDE version 3.2. The Subversive plugins is required for the Subversion access, but it comes in the standard binary distribution of FastTrack.

Fig. 7: Commit Policies



You should repair policies that failed.

Some policies failed.

Group errors by [✓ policy] resource

Active policies:

```
Policy name
Code errors - FAILED
- SVNWebClientException.java: Syntax error on token "class", class expected, line 27
Code warning - passed
```

In case of a programming team implementation, it is necessary that a functional Subversive server be provided which can be accessible through an Apache server.

As soon as the Eclipse IDE and SVN are correctly installed and configured, it will be necessary to install the Subversive system by starting the Update Manager of Eclipse and accessing the FastTrack update site: http://www.polarion.org/projects/fasttrack/download/update-site

After successful FastTrack installation it is necessary to obtain the license file, which can be requested in following direc-tory: Windows/Preferences/Tracker/License/GetLicense. An internet site pops up, where you have to enter your user data. The license information will be sent automatically to the entered e-mail address.

After this procedure, you can easily swap to the FastTrack View and start working. The first recommended step is to create a new project by New/Tracker/Tracker Project.

Resources & References

[1] http://www.polarion.com
[2] Polarion Community for Subversion--this is the homepage and the site for the Subversive project and for many other SVN oriented open source projects as well: http://www.polarion.org

Antonin Pokorny

is a VP R&D at Polarion Software. He has many years of experience in the field of application Lifecycle product development. In addition to that he manages many distributed development projects and he is the founder of the Subversion Eclipse Plug-in "Subversive".

SHARE ECLIPSE INFORMATION WITH THE COMMUNITY AND WIN BIG!!!

Advertisers

10	**Eclipse Magazine** http://www.eclipsemag.net
14	**JAX Innovation Award** http://jax-award.com
18	**SDA Asia** http://www.sda-asia.com
41	**JAX Magazine** http://www.jaxmag.com
46	**International PHP Magazine** http://www.phpmag.net
48	**S&S Media** http://www.softwaresupportmedia.com

Imprint

Editor-in-Chief	Indu Britto
Associate Editor	Alexander Neumann
Managing Editor	Dilip Thomas
Authors	Mohamed Khiasudeen, Marco Schulze Wayne Beaton, David Trainor, Jason Weathersby, Antonin Pokorny
Layout	S. Ganesh, Gwendolyn Vaz

How to Contact Us
We invite you to submit letters, articles, announcements, and press releases to Eclipse Magazine.

E-Mail:	editors@eclipsemag.net
Address:	Software & Support Verlag GmbH Geleitsstraße 14 60599 Frankfurt am Main Germany
Tel:	+49 (0) 69 63 00 89 0
Fax:	+49 (0) 69 63 00 89 89
Web Site:	www.eclipsemag.net

Advertising

E-Mail:	advertise@eclipsemag.net, noutmani@entwickler.com
Tel:	+49 (0) 69 63 00 89 0
Fax:	+49 (0) 69 63 00 89 89

eclipse MAGAZINE POWERING THE ECLIPSE ECOSYSTEM