SYS-CON **THE BUSINESS VALUE OF AN SOA STRATEGY** PAGE 12

No. 1 *i*-Technology Magazine in the World

JDJ

JDJ.SYS-CON.COM VOL.12 ISSUE:7

REALWORLD JAVA SEMINAR
REALWORLDJAVA.COM
THE LARGEST JAVA DEVELOPER EVENT ON THE EAST COAST
AUGUST 13, 2007
ROOSEVELT HOTEL, NEW YORK
SEE PAGE 29 FOR DETAILS

EFFECTIVE DEVELOPMENT OF JAVA CONFORMANCE TESTS WITH META-PROGRAMMING

Java Annotations
+ Compiler API
+ Annotation Processing

Remarkable Results

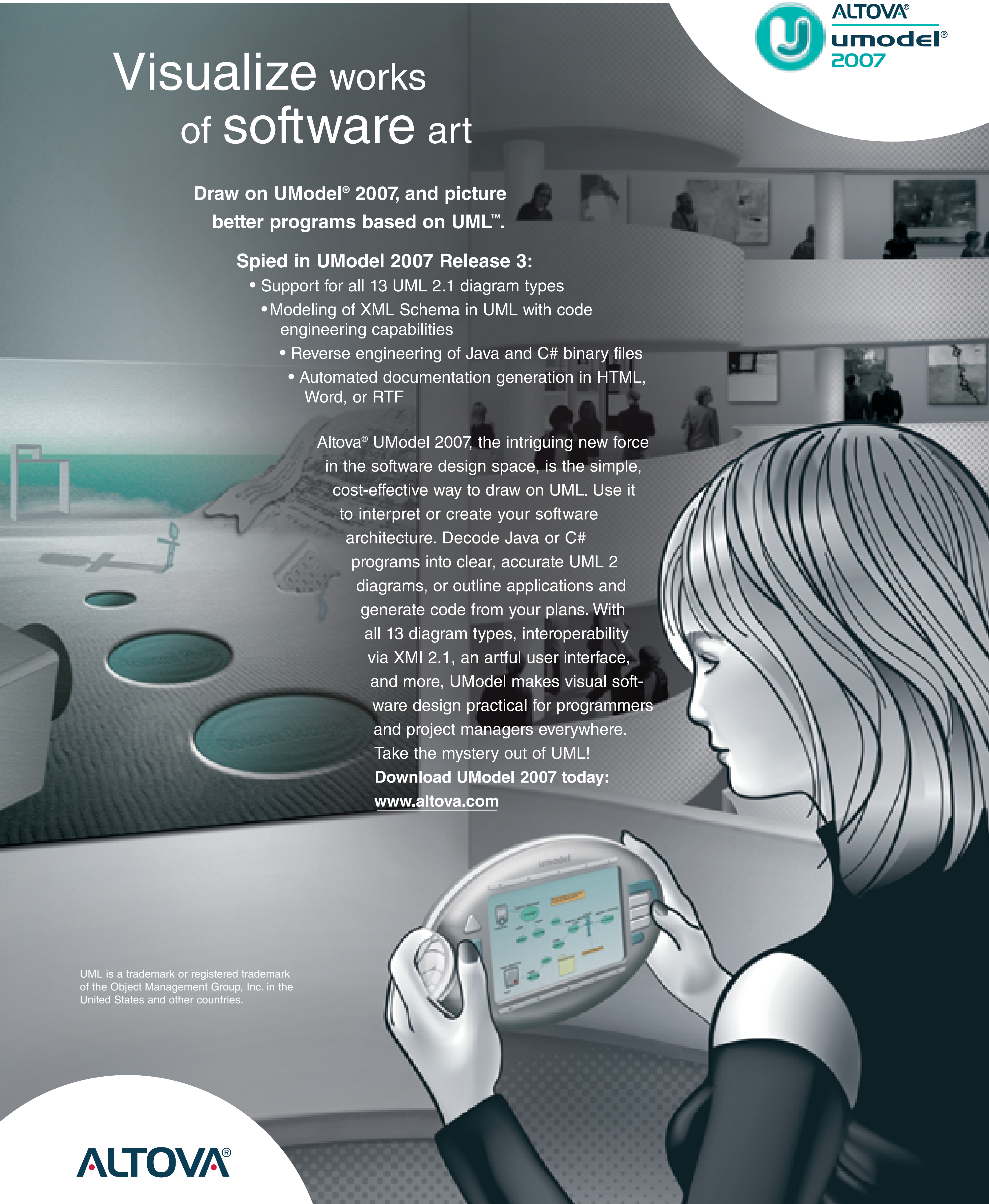
RETAILERS PLEASE DISPLAY UNTIL SEPTEMBER 30, 2007
$5.99US $6.99CAN

PLUS...

- Desktop Java Slims Down to Enter the Special FX Race
- The Evolution of SIP Servlets for Converged Voice and Data Apps
- Skilled Listening in Java

Altova® UModel® 2007 – The starting point for successful software development.

ALTOVA® umodel® 2007



Visualize works of software art

Draw on UModel® 2007, and picture better programs based on UML™.

Spied in UModel 2007 Release 3:

- Support for all 13 UML 2.1 diagram types
- Modeling of XML Schema in UML with code engineering capabilities
- Reverse engineering of Java and C# binary files
- Automated documentation generation in HTML, Word, or RTF

Altova® UModel 2007, the intriguing new force in the software design space, is the simple, cost-effective way to draw on UML. Use it to interpret or create your software architecture. Decode Java or C# programs into clear, accurate UML 2 diagrams, or outline applications and generate code from your plans. With all 13 diagram types, interoperability via XMI 2.1, an artful user interface, and more, UModel makes visual software design practical for programmers and project managers everywhere. Take the mystery out of UML! **Download UModel 2007 today: www.altova.com**

UML is a trademark or registered trademark of the Object Management Group, Inc. in the United States and other countries.

ALTOVA®

Desktop Java Slims Down to Enter the Special FX Race

Joe Winchester

A number of very significant development efforts are underway that bode well for Desktop Java's future.

On the language side is the Java FX script project http://www.sun.com/software/javafx/index.jsp. Java FX is neat because it provides a high-level scripting interface that runs on top of the Java 2D API. From the users' viewpoint it means they don't have to write Java code and, for better or worse, understand the intricacies of threads, Java 2D or Swing class hierarchies, timing frameworks, and so forth. Instead they just write script that describes the desired GUI at a high level, including animation effects, graphic effects such as gradients or noise, and data binding in the base API. There are some nice tools out there to help teach the language syntax that's available on https://openjfx.dev.java.net/; the goal of the FX team is to provide a syntax and experience that is palatable to graphic designers and folks who are interested in finessing the visual experience of the application. Currently at runtime it operates as a kind of fourth-generation language because an interpreter creates Java code from the FX script and then calls the appropriate Java 2D APIs. The long-term goal of the team though is to compile from FX script straight to byte codes.

On its own Java FX is s a nice addition to the desktop Java stable; however, it still relies on the presence of a JRE that can run the code it generates. To those who see FX as being an entrant into the rich Internet application space, competing with Adobe Flex for example, there remains the difficult problem of how to get the install experience and runtime performance of desktop Java comparable to other rich Internet application architectures. Fortunately this problem is being tackled by another very interesting project, the Java kernel.

The Java kernel initiative recognizes the fact that most Java applications typically use a subset of the JRE, an example being Limewire (a file-sharing client app) that weighs in at about 1/3 of a total JRE size. The kernel will repackage the JRE such that only the portions required to run an app are downloaded as and when they are needed. The base portion that every program needs would be downloaded as a minimum for the JRE to start, and an application can either explicitly specify additional dependencies it needs, or rely on ClassNotFound exception magic to cause missing classes to be downloaded on demand. This is all great news to help slim down and make more efficient a Java program's delivery to a user's desktop. In addition to tackling how Java programs are downloaded, the kernel is also looking at the startup performance of Java programs. While warm start times have decreased significantly over the last few releases, a cold start JRE still requires a delay of many seconds before the application's main(String[]) is called. The Java kernel is attempting to tackle this by effectively turning a cold start experience into a warm one, using techniques such as pre-loading the JRE from disk into memory, as well as having operating system services whose job it is to monitor available memory and JRE activity to get the best performance for the user. In addition to these, the kernel is also providing a new browser plug-in that gives Web start developers more information about what's on the desktop, the available JRE levels and vendors in detail, and generally makes the whole JNLP experience slicker and more robust.

Both the Java kernel and Java FX are great news for Desktop Java. The kernel is good because it's recognizing that Swing has reached a stage of maturity where the problem isn't one of missing features, but how to help developers get their applications onto a user's desktop as quickly and efficiently as possible. FX is great news because it opens up a whole new set of developer to Java; those who are comfortable with scripting languages and want to do advanced animation and graphics effects easily with a few lines of code. The end result of both will be more applications written in Java running on more users' machines, faster and easier for both developer and user.

Joe Winchester is a software developer working on WebSphere development tools for IBM in Hursley, UK.

joewinchester@sys-con.com

Subscriptions

For subscriptions and requests for bulk orders, please send your letters to Subscription Department:

888 303-5282
201 802-3012
subscribe@sys-con.com

Cover Price: $5.99/issue. Domestic: $69.99/yr. (12 Issues) Canada/Mexico: $99.99/yr. Overseas: $99.99/yr. (U.S. Banks or Money Orders) Back Issues: $10/ea. International $15/ea.

Editorial Offices

SYS-CON Media, 577 Chestnut Ridge Rd., Woodcliff Lake, NJ 07677
Telephone: 201 802-3000 Fax: 201 782-9638

Java Developer's Journal (ISSN#1087-6944) is published monthly (12 times a year) for $69.99 by SYS-CON Publications, Inc., 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677. Periodicals postage rates are paid at Woodcliff Lake, NJ 07677 and additional mailing offices. Postmaster: Send address changes to: Java Developer's Journal, SYS-CON Publications, Inc., 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677.

Embedded Reporting and Analytics

Operational Business Intelligence for Java Applications

Only JReport has what it takes to make both developers and users happy.

Developers are happy to embed reports with sophisticated layout options, Web controls and integrated security. JReport makes their life easier with top performance, scalability and seamless Java EE deployment.

Users are happy to get their reports pre-scheduled or on demand; view them with cascading parameters, dynamic drilling, sorting, filtering and precise exporting. JReport's self-service ad hoc reporting lets users create analytic reports intuitively and work with them interactively.

The bottom line: JReport makes the business more efficient and more productive. And that makes everyone happy.

See for yourself, go to www.jinfonet.com/fd and see the power of JReport.

Visit www.jinfonet.com/fd for a Test Drive

JReport®
JINFONET SOFTWARE

JULY 2007 VOLUME:12 ISSUE:7

JDJ contents

JDJ (ISSN#1087-6944) is published monthly (12 times a year) for $69.99 by SYS-CON Publications, Inc., 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677. Periodicals postage rates are paid at Woodcliff Lake, NJ 07677 and additional mailing offices. Postmaster: Send address changes to: JDJ, SYS-CON Publications, Inc., 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677.

Skilled Listening in Java

How to generate events for listeners correctly and efficiently

by Dr. Warren D. MacEvoy

There are many good resources on using events and implementing event listeners. Unfortunately, there are nearly as many resources that incorrectly or inefficiently implement event generators. These implementations introduce subtle nondeterministic errors, execute slowly, and/or fill the virtual machine with unnecessary garbage. The goal of this article is to address this problem once and for all – as a pattern and code generator for JDKs before 1.5, and as a set of generic types in JDK 1.5 and later.

Most Java programmers learn how to *listen* early on, usually thanks to the Swing classes. The elements are as follows:

- *An event*, which is the information to exchange.
- *A listener*, which receives the information.
- *A generator*, which sends the information. Listeners can be added (or removed) from generators as they become interested (or disinterested) in the events they send.

Abstractly, at any given time, the set of generators and listeners for an event form a directed graph, where edges go from generators to listeners. Usually, this graph is essentially static. That is, listeners are added to generators as part of some initialization phase, and events don't start moving through the graph until after all the connections have been made. Almost all of the implementations you see of generators work correctly, although not necessarily efficiently, in this situation.

Here's the rub. These things are often not correctly accountedfor:

1. Adding and removing listeners may happen asynchronously. That is, different threads may add or remove listeners at somewhat inconvenient times.
2. In response to listening to an event (and so synchronously), a listener may add or remove itself or other listeners from a generator.
3. Generators may choose to send events asynchronously. These are all issues of correctness. Issues of efficiency are:

1. Most generators spend most of their life sending events (as opposed to adjusting who is listening).
2. The average number of listeners for each generator is significantly less than one.

The humble truth is, most of the generators in your application are not being listened to by anything, and most of the rest have one listener. For example, of the 18 events generated by a JButton, how many do you actually pay attention to?

Correct

Correct is more important than efficient. Listing 1 shows a typical implementation of a correct, but inefficient, generator.

Let's see why this is correct. First, the synchronized(lock) blocks synchronized access to the listeners' list. It is very important that *only the clone step of send is synchronized*. This is because completing a send step may require arbitrary adds or removes. If these adds or removes happen from a different thread compared to the send, they would deadlock if send(Event) is synchronized. To avoid this problem, we copy the listeners list before notifying them, thus eliminating the resource contention. It is also important that *we do not use the synchronized keyword in front of the add, remove, and clone methods*. This is because using synchronized in front of a method uses the object for mutually exclusive access to the methods. Since you have no control over how the DefaultGenerator will be used, it's pretty easy to set up a deadlock situation by having this public mutex locked by something else. By sticking to an internal private object for a lock, we know we're the only ones who affect it.

–continued on page 10

Warren D. MacEvoy is a CS professor in the department of computer science, mathematics and statistics at Mesa State College, Grand Junction, Colorado.

wmacevoy@mesastate.edu

SYS-CON MEDIA

President and CEO:
Fuat Kircaali *fuat@sys-con.com*

President and COO:
Carmen Gonzalez *carmen@sys-con.com*

Group Publisher:
Roger Strukhoff *roger@sys-con.com*

Advertising

Advertising Sales Director:
Megan Mussa *megan@sys-con.com*

Associate Sales Manager:
Corinna Melcon *corinna@sys-con.com*

Events

Events Manager:
Lauren Orsi *lauren@sys-con.com*

Events Associate:
Sharmonique Shade *sharmonique@sys-con.com*

Editorial

Executive Editor:
Nancy Valentine *nancy@sys-con.com*

Production

Lead Designer:
Tami Lima *tami@sys-con.com*

Art Director:
Alex Botero *alex@sys-con.com*

Associate Art Directors:
Abraham Addo *abraham@sys-con.com*
Louis F. Cuffari *louis@sys-con.com*

Web Services

Vice President, Information Systems:
Bruno Decaudin *bruno@sys-con.com*

Information Systems Consultant:
Robert Diamond *robert@sys-con.com*

Web Designers:
Stephen Kilmurray *stephen@sys-con.com*
Richard Walter *richard@sys-con.com*

Accounting

Financial Analyst:
Joan LaRose *joan@sys-con.com*

Accounts Payable:
Betty White *betty@sys-con.com*

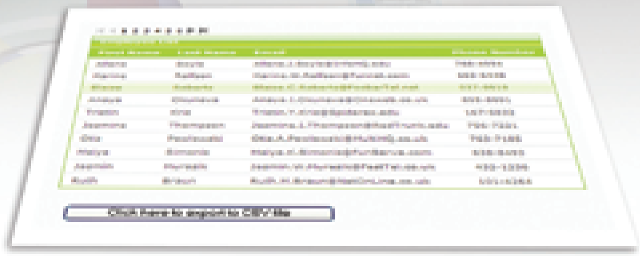
Customer Relations

Circulation Service Coordinator:
Edna Earle Russell *edna@sys-con.com*
Alicia Nolan *alicia@sys-con.com*

"I bring meaning and comprehension to my corporate data using advanced visualization tools."

- Application Developer

WebChart™ - 3D Overlay Donut Chart



WebGrid™ Export to CSV Format

NetAdvantage® for JSF 2007 Volume 1

Consistent Multi-Platform User Experience

WebCharting – Add high impact 2D/3D visualizations with over 20 different chart types and views

Improved Data Reporting – Easily export data within a grid to any application that supports the import of CSV such as spreadsheets and databases

Application Performance – Leverage our AJAX framework to turbo-charge your Web applications

Accessibility – Support of US Code Section 508 across all Infragistics controls

learn more: infragistics.com/jsf

Infragistics®
Powering The Presentation Layer

Infragistics Sales - 800 231 8588
Infragistics Europe Sales - +44 (0) 800 298 9055

WINDOWS FORMS ASP.NET WPF JSF

grids scheduling charting toolbars navigation menus listbars trees tabs explorer bars editors

_INFRASTRUCTURE LOG

_DAY 72: We wrote our software but didn't build it to fit with the broader IT architecture requirements. Now we don't have the flexibility to reuse our assets. We're not moving forward. Why did we lock ourselves in like this?

_I never knew being stuck at work could be so literal.

_DAY 73: Here's something less confining. IBM Rational unifies all aspects of our SOA software design and development. Now we can ensure global architecture integrity using a new, simpler, modular systems development approach. And we're speeding our results with sound architectural design and automated service delivery and maintenance.

_I'm glad we're free. Was never sure where to put "stuck to my coffee cup" on my time sheet.

IBM.COM/**TAKEBACKCONTROL**/INNOVATE

–continued from page 6

Safety pays in correctness (which is a good thing), but it suffers from some serious inefficiencies:

1. The listeners list is cloned on every send. Most of the time, nobody changes the listeners during a send, so the clone does not usually matter.
2. An Iterator is created on every send. This is another object to create and discard. Using a (pre JDK 1.5) iterator also requires runtime safety checks while going through the List.
3. An Event is passed to every send. Most of the time, nobody is listening to the generator, so that event usually just lands in the garbage pile.

And Efficient

Sometimes correctness costs efficiency, but we are lucky here, because we can have both (see Listing 2).

To use this DefaultGenerator, we would write:

```
class MyGenerator
  extends DefaultGenerator {
  void send() {
     if (listening()) send(new Event());
  }
}
```

There is more code, but what have we gained? This implementation is also correct, but look at send(), listening() and send(Event): no synchronization, no copies, no type-casts, and we only create an Event if (almost always – see sidebar) we have listeners to receive it. Instead of duplicating the listener list every time a send happens, we make copies when adding and removing from the listeners array. This works because object reference assignment is an atomic action. We use a plain old array because this eliminates unnecessary runtime type casts and the iterator overhead.

We have a generator that is still safe, but faster than the naïve safe solution. But it's exactly the kind of code that is easy to get wrong. Why would we want to rewrite something a bunch of times anyway? JDK 1.5 helps us with the use of generics. Instead of using the above pattern again and again for each kind of event, we can write it once and be done. An implementation is given in the resources.

With the generics at hand, you can stop worrying about making correct and fast generators and just use them. For example, if you wanted to a DateGenerator that sends the current Date on every tick(), a listener that printed out the date, and some main code to put them together:

```
class DateGenerator
  extends DefaultGenerator < Date > {
  public void tick() {
   if (listening()) send(new Date());
  }
}

class DateListener implements Listener <
Date > {
    public void receive(Generator < Date >
generator , Date date) {
          System.out.println(date);
    }
}

public class Main {
    public static void main(String[] args)
    {
     DateGenerator generator =
       new DateGenerator();
     DateListener listener =
       new DateListener();

     generator.addListener(listener);
     generator.tick();
    }
}
```

With the generic implementation, you can think about the actual messages and what they do, rather than the details of the mechanism for every new message type.

For those who are required to be compatible with a pre 1.5 JDK, it can be a lot of error-prone typing to write the boiler-plate code for every Event type you happen to need. In the resources, there is a tool for writing this code (built using the BeanShell Preprocessor, but you don't need BPP or BeanShell to use it).

What's Been Said...

So – generators are poorly or incorrectly implemented all the time. Using these generators result in solutions that are less stable, slower, and consume more memory. For older (pre 1.5) JDKs, this can be solved by adopting the fast generator pattern. These patterns can be built automatically using meta. GenerateListeners. For new (post 1.5) JDKs, this can be solved using generic classes. These tools are given in the resources.

That's it – happy listening in a dynamic world!

Resources

- *Online version of this paper:* http://www.mesastate.edu/%7Ewmacevoy/listen/paper.html
- *JavaDOCs for meta package:* http://www.mesastate.edu/%7Ewmacevoy/listen/meta/doc/index.html
- *JavaDOCs for classes generated by meta package:* http://www.mesastate.edu/%7Ewmacevoy/listen/usemeta/doc/index.html
- *JavaDOCs for events package:* http://www.mesastate.edu/%7Ewmacevoy/listen/generic/doc/index.html
- *Source/binary/doc zip file:* http://www.mesastate.edu/%7Ewmacevoy/listen.zip
- *Writing Event Listeners:* http://java.sun.com/docs/books/tutorial/uiswing/events/index.html

– Listings 1 and 2 can be downloaded from the online version of this articlw at http://java.sys-con.com/read/400083.htm

"almost always"

I stated that, almost always, if (listening()) send(new Event()) will create and send an event only when there are listeners to receive it. But notice that there is no synchronization between the listening() and the send(new Event()), and so the listeners list may change arbitrarily between the check for listeners and the send. First, realize this is a very subtle timing issue and so, almost always, the listening() call will correctly determine if there is anyone listening. Second, realize this does not effect the correctness of the generators, because only two things will (very rarely) happen:

- **listening() returns true, but the last listener is removed before the send(Event).**
  The Event is created and unused. This has no impact on correctness, and negligible effect on efficiency.
- **listening() returns false, which skips a send(Event), because a listener is added just before the send would have occurred.**
  This same argument could be made in the send(Event) method since we copy the reference to the listeners array before notifying listeners. At some point (shortly before the sending) you have to decide if you will or won't actually send. Since it is convenient to decide if you will send before constructing the event, this has no impact on correctness, but a significant positive impact on efficiency.

RARE OCCURRENCE.

For a limited time, upgrade to Crystal Reports® XI for only $99. Create brilliant reports in minutes and speed report integration so you can focus on what you do best – core application coding. A great price with this depth of features is a rare occurrence, indeed.

- .NET, Java™ and COM SDKs
- Unlimited free runtime for internal corporate use
- Includes Crystal Reports Server – embed report management services
- Includes crystalreports.com – share reports over the web
- Unlimited installation-related support

NOW $99

UPGRADE
or
$395 NEW

Act fast. Go to www.businessobjects.com/rareoccurrence or call 1-888-333-6007 today.

The Business Value of an SOA Strategy

Interview with Sandy Carter, vice president, SOA & WebSphere Strategy, Channels and Marketing, IBM Corporation

Interviewed by
Roger Strukhoff

After IMPACT 2007 in Orlando, *Java Developer's Journal* had the opportunity to talk with Sandy Carter about IBM's new SOA announcements at the event, as she is responsible for driving IBM's cross-company, worldwide SOA marketing initiatives.

JDJ: *Please outline for us the new SOA roadmaps that IBM announced at the recent IMPACT event in Orlando. What do they contain? Are they targeted at vertical markets, and, if so, how else are they targeted?*

Sandy Carter: IBM has just announced eight new industry-specific SOA Roadmaps spanning six industries. Each of the SOA roadmaps contains a business blueprint, which helps customers map the business side of an SOA strategy, and an industry-specific framework, which includes core technology used to execute the business blueprint. The new SOA Roadmaps focus on critical business process areas within a given industry. Some examples include online booking for the insurance industry, member enrollment and benefits/eligibility for health care, payments for banking, personal shopping for retail, service provisioning and service delivery for telecommunications, and supply chain collaboration for industrial.

The business blueprints begin with detailed research that outlines industry challenges and how those industries can benefit from SOA. Each piece of research examines specific business scenarios, describing how those processes are typically executed today and how they could be improved with SOA. The business blueprint also recommends an SOA entry point to make it easier for businesses to know the best place to start to deliver the most immediate business benefits. IBM's SOA entry points make it easier for customers to approach and initiate SOA projects. The three business entry points are people, process, and information. The two technical entry points are connectivity and reuse.

Complementing these offerings, IBM also recently announced the IBM SOA Industry Frameworks. These frameworks include technology from IBM and IBM Business Partners to help use SOAs. They provide customers with reusable, industry-specific software modules called business services that are based on the WebSphere Business Services Fabric. These modules perform individual tasks tailored to the industry's users, policies, and methods. New SOA Industry Frameworks are specific to the banking, health care, telecom, retail, and insurance industries to add to the first framework for Product Lifecycle Management, which was announced late last year. Additional frameworks will follow later this year.

SANDY CARTER
Vice president, SOA & WebSphere Strategy, Channels and Marketing,
IBM Corporation

JDJ: *IBM also announced six new SOA Professional Services. Can you provide an overview on each of these?*

Carter: IBM's six new SOA professional services are focused on SOA Diagnostic, SOA Strategy, SOA Implementation Planning, Business Process Management Enabled by SOA, SOA Design Development and Integration, and SOA Management. New capabilities include infrastructure and strategy workshops for SOA Strategy professional services, Web application and portal infrastructure services for SOA Design, Development and Integration, and a new testing center of excellence for SOA management.

JDJ: *IMPACT had a flurry of activity regarding SOA. Is this in response to general projections of growth for SOA, customer input, the competitive landscape, or all three?*

Carter: IBM's activity around SOA has been driven by strong customer demand. SOA has been a growth engine for IBM as well as our customers because it gives companies the much-needed flexibility to focus on achieving business results without being hindered by the constructs of established infrastructures. IBM's competitive differentiation is in its ability to address business challenges using the right balance of business and technical skills along with an unmatched, multi-pronged approach to meeting customers' needs.

"IBM has just announced eight new industry-specific SOA Roadmaps spanning six industries"

Roger Strukhoff is Group Publisher and Editorial Director of SYS-CON Media. He spent 15 years with Miller Freeman Publications and The International Data Group (IDG), then co-founded CoverOne Media, a custom publishing agency that he sold in 2004. His work has won awards from the American Business Media, Western Press Association, Illinois Press Association, and the Magazine Publishers Association. You can read his blog at "rssblog.linux.SYS-CON.com" and contact him at roger(at)sys-con.com.

roger@sys-con.com

"IBM recognizes that a successful SOA strategy requires both IT and business executives to collectively map out goals **and illustrate the value of the SOA project"**

JDJ: *What sort of time frame or range of time frames do you typically see with SOA, from original outline to deployment? What sort of preference have you seen for companies to do a pilot project versus, say, a complete departmental SOA design and deployment?*
Carter: The time frames vary depending on the organization's needs and the alignment of business and IT. Working with more than 4,500 customers worldwide on their SOA strategy, we recommend that companies first begin with one of the five SOA entry points. These entry points are people, process, information, connectivity, and reuse. In addition, we recommend forming a cross-company team of business and IT leaders so that the SOA strategy is mapped out and agreed upon before deployment.

JDJ: *What are the most common questions that you receive from customers who are just setting sail along the SOA waters? What are the most common questions you receive from customers who have deployed a SOA and are now in the process of maintaining it or expanding it?*
Carter: The five most common questions from customers who are beginning their SOA journey are usually:

- Can I still use my existing software and hardware?
- Where do I begin with SOA?
- Is SOA for large companies only?
- How do I gain the much-needed SOA skills?
- How do I prove the value of SOA to the executives in my organization?

First, SOA is specifically designed to protect existing IT investments in software and hardware. The most successful SOA deployments are found at those companies that begin with a view of SOA as a long-term business strategy. These companies approach SOA incrementally, through one of the five entry points cited above, and continuously measure and monitor its success at every step along the way.

Interestingly, in addition to large organizations, we're seeing more and more companies in the small- to medium-sized business market deploying SOA as a way to build a more flexible environment that makes the most use of technology and staff skills.

For those companies that are expanding their SOA, the questions are usually about, "how do I document and share my business processes?" To do this, it's important that the SOA initiative includes a strong governance strategy to make sure that its services can be easily managed, monitored, and reused while ensuring only best practices are shared.

In terms of developing skills, it's clear that the industry is facing a serious SOA IT skills shortage. In fact, a recent study found that 56 percent of IBM customers cited lack of skills, mainly individuals with a blending of IT technical understanding and business process acumen, as the leading inhibitor to SOA.

To address the SOA skills shortage, IBM recently announced new tools and certification programs to help organizations develop teams of individuals with so-called "T-shaped" skills, which encompass both deep business skills, represented by the horizontal line of the "T," and technical understanding, represented by the vertical line.

The new skills-centric offerings include an interactive SOA game called Innov8, which is a Business Process Management (BPM) Simulator. Innov8 is an interactive, 3D educational game simulator designed to bridge the gap in understanding between IT teams and business leaders in an organization. This type of serious gaming – simulations that have the look and feel of a game but correspond to non-game events or processes such as business operations – has emerged as a successful method to retrain or develop new skills. Of note, this simulator is the result of the annual IBM SOA case study competition among graduate students at Duke University and the University of North Carolina. The game, which is played with a joy stick, is based on advanced, commercial gaming technologies and allows players to visualize how an SOA affects different parts of the organization. Together, users can literally see business processes, identify bottlenecks, and explore "what if" scenarios before the SOA is deployed. In addition, IBM has introduced "IBM TV: Impact Channel," an online portal that contains Webcasts, podcasts, demos, white papers, and other resources targeted to business and IT professionals as well as those looking to develop T-shaped skills.

Further, IBM has enhanced its SOA certification and education programs with new, self-paced and instructor-led courses conducted online and in classrooms. With more than 218 SOA-based courses for every level in an organization, IBM's SOA curricula provide the roadmap to master the most highly sought-after SOA industry skills. IBM also continues to foster relationships with higher education institutions and is working with hundreds of colleges and universities around the world on SOA curricula.

IBM recognizes that a successful SOA strategy requires both IT and business executives to collectively map out goals and illustrate the value of the SOA project. To this end, IBM offers free, online assessment tools to help executives at companies of all sizes determine their organization's business and IT needs. For example, IBM's SOA Business Analyzer helps an organization identify its greatest assets and opportunities that are ripe for an SOA project.

Also, the IBM BPM with SOA ROI Assessment Tools helps organizations assess their Business Process Management (BPM) readiness through 10 simple questions. It then delivers a BPM score, assesses an organization's position to realize benefits from BPM, and provides recommended next steps. Through this assessment, both IT and business executives can better understand the business value of an SOA strategy.

The Evolution of SIP Servlets for Converged Voice and Data Applications

Upgrading the SIP Servlets 1.0 standard

by Chris Boulton

It's widely recognized that the telecommunications industry is riding the crest of change and evolution on the back of new access technologies such as 3G, GPRS, and Wi-Fi. Such IP-centric access mechanisms must also be considered in conjunction with the emergence of feature-enabling technologies such as VoIP, instant messaging, and presence. A whole new converged telecommunications world is emerging that requires an appropriate complementary IP-based infrastructure as opposed to the legacy, circuit-switched solutions of the past.

The disruptive effects that such an evolution is having on telecommunications service providers' revenue streams are also worth mentioning. The inevitable migration to IP-based services that can support converged applications has introduced a massively competitive marketplace where new network specialists such as Skype (www.skype.com) and Freewire (http://www.freewiretv.com/) are eating away at voice-related revenues and turned "voice" as a service into a commodity. As a result, to stay competitive, Service Providers are looking to alternative service offerings that can be developed in a relatively short space of time but that can fully utilize the flexibility of packet-switched networks, which also translates to new opportunities for Java developers.

The Session Initiation Protocol (SIP) protocol has emerged as the de facto standard for enabling converged voice and data applications in telecommunications. SIP servlets provide a key component to rapidly developing next-generation converged services, and SIP servlets will become more important as the next-generation technology emerges.

In 2003 when the SIP Servlets 1.0 standard was published (JSR 116), it soon became apparent that a technology was emerging that would change the telecommunications industry. SIP Servlets 1.0 provided the foundation for an exploratory technology step that allowed the rapid development of SIP-based telecom applications that would enable VoIP services as well as converged voice and data applications.

Using the well-established servlet model from J2EE, the adapted SIP servlet container inherits important functionality such as lifecycle management and configuration. SIP Servlets 1.0 also provides an appropriately abstracted API for SIP Servlet developers. The complexity of SIP is therefore abstracted away, providing functionality that doesn't require in-depth knowledge of the protocol's intricacies. The SIP Servlet 1.0 specification and API forms the basis of many application servers used in early IP-based telecom services deployments. Since it was first introduced, the technology has been widely deployed in a variety of both fixed and mobile networks, especially as part of the IP Multimedia Subsystem (IMS) reference architectures defined by the 3rd Generation Partnership Project (3GPP).

SIP Servlets 1.0 was a revolutionary step in providing a platform with application-level constructs. But as with any evolving technology, requirements and industry demands are extremely dynamic and it became apparent that SIP servlets needed an overhaul to maintain its technological advantage.

The Java Community is currently defining the newest-generation SIP Servlets 1.1 (JSR 289). In January 2006 an expert group was formed to deliver SIP Servlets 1.1 with the goal of producing a technology that built on the success of SIP Servlets 1.0. SIP Servlets 1.1 (JSR 289) would also forge closer technological alliances with J2EE and provide an appropriate vessel to carry SIP into NGN (Next-Generation Networks) and beyond. The objective was to overcome the problems encountered since the introduction of SIP Servlet technology. The result is the upcoming JSR 289 release.

Why a New Version?

It has become clear that the functionality of SIP Servlets 1.0 has been outpaced by the rapid evolution of the VoIP industry, but where exactly does it fall short? First, some background and an explanation of the problems that have been identified as a result of experiences with both SIP servlet container and application implementations.

As a starting point, the SIP Servlet 1.0 specification was a particularly revolutionary technology. Like so many early representations of technology, the fundamental concepts were good but continual refinement is required to improve the quality of the solution. The original text in the SIP Servlet 1.0 specification doesn't provide the explicit guidelines that container developers need to deliver consistent operations across a range of vendor platforms. Stronger language throughout the document is required to set tighter boundaries in a similar style to standards bodies like the Internet Engineering Task Force (IETF). The areas of confusion introduced into the specification have led to a wide variety of implementation variance. As a result, there is little chance of inter-vendor interoperation when porting applications between container implementations.

Application developers have also uncovered key areas in the SIP Servlet API that need to be tailored to provide more assistance. For example, a Back-To-Back User Agent (B2BUA) is one of the most commonly used constructs implemented using the SIP Servlet API. B2BUA involves two separate SIP user agents bound together to act as a single signalling entity (as opposed to a pure proxy). Most

Chris Boulton is part of the CTO's development team at Ubiquity Software, creators of the Ubiquity SIP Application Server, which is being used for converged applications by more carriers than any other SIP platform in the world.

SIP applications written for the telecom industry today have some form of B2BUA functionality (e.g., pre-pay, conferencing). In version 1.0 of the API, the process of implementing simple B2BUA functionality was quite cumbersome when you consider how frequently it was used. This was primarily due to the complexity of managing the protocol sessions and state associated with B2BUA functionality.

As with HTTP servlets, SIP servlets have the concept of Sessions, which provide association for multiple actions. SIP servlets have application sessions that act as an umbrella for a specific application instance. Zero or more associated protocol sessions (SIP Sessions) can exist in an application session that loosely map to a SIP call. When a SIP servlet container receives a new SIP request, an application session and SIP session are automatically created. In certain circumstances, say a conferencing application, it's appropriate to associate new SIP protocol sessions with an existing application session instead of creating new sessions. In SIP Servlets 1.0 this is achieved using an Encode URI mechanism. The encoded URI is included in the SIP message and when it arrives at the SIP servlet container, it's routed directly to the appropriate application instance. This may seem like a useful mechanism but it has proven to be extremely troublesome and difficult to use (e.g., for distributing an encoded URI). Using an encoded URI also has the added drawback of skipping application composition. When the SIP servlet container gets an encoded URI in a SIP request, it routes the message directly to the targeted destination without evaluating the request for alternative application processing. This can create problems in application composition where, for example, you might have a security or prepay application that must be invoked first.

Convergence is a word often used in association with SIP servlets, and it's usually incorrectly conveyed. SIP Servlets 1.0 correctly introduced the concept of convergence, but that definition only covers half the story. It only defines and discusses convergence of SIP servlets with HTTP servlets that coexist in a single monolithic solution. While useful for some simple applications, this type of deployment isn't appropriate for large-scale deployments where integrating with high-throughput J2EE applications requires a clear boundary to separate performance and maintenance. The original SIP Servlet specification clearly lacked the ability to externally access SIP servlet resources within a Service Oriented Architecture/J2EE environment. This prohibits certain architectures and limits integration between different IT infrastructures and core telecom network infrastructures.

One of the most important and powerful features of SIP servlets is the application composition model. Application composition refers to the ability of a SIP request to visit multiple servlet applications as part of a larger combined solution or service. SIP Servlets 1.0 uses much of the functionality inherited from HTTP servlets, where incoming requests for service are judged on the XML configuration contained in the deployment descriptor. On reflection, this kind of static, hard-wired, and complex configuration of applications isn't entirely appropriate for the real-time world of telecommunications application composition. A more flexible approach is needed that allows a dynamic decision-making process; possibly a process based on related activities and external triggers that have occurred (e.g., access to external data source).

SIP is now an extremely mature protocol with a plethora of extensions and modifications. SIP Servlets 1.0 is based on one of the later versions of RFC 2543, which was later made obsolete by RFC 3261. It's understandable that, since the creation of SIP Servlets 1.0, the requirements demanded of the protocol have progressed. Over the past few years a number of new SIP methods have been introduced (e.g., UPDATE) along with extensions that provide important functionality (e.g., the SIP PATH header in RFC 3327). These modifications need to be incorporated into future releases of the standard to provide the level of functionality required for today's telecom VoIP applications. It's also fair to say that SIP's evolution has led to a number of mismatches in the SIP Servlet API that need to be resolved.

Clearly, there are a number of key areas in the SIP Servlet API that need modification too. An expanded list of all the requirements, both large and small, clearly indicates that JSR 116 was ready for a major overhaul.

What's New in SIP Servlets 1.1?

Given the advances in telecommunication applications, it was clear that SIP Servlets 1.0 was ready for a spring cleaning. SIP technology had been widely adopted in telecommunication architectures and with appropriate modifications would provide a central core standards platform. However, to address some of the shortcomings in the initial standard, there are key features that will appear in SIP Servlets 1.1 because of the problems identified.

As we said, the application composition model used in SIP Servlets 1.0 wasn't flexible enough to support emerging telecom applications. To overcome these hurdles, SIP Servlets 1.1 will incorporate a radically new application composition and routing scheme that provides additional functionality and flexibility. Figure 1 shows how applications are composed and routed in SIP Servlet 1.0.

The request enters the SIP stack (1) and is passed to each application loaded in turn (2-4). When a SIP request is passed to an application, it's validated against the deployment descriptor XML file. The container will check to see if the specific application (a SIP Servlet Archive or sar file) is interested in servicing the SIP request. If it does it's triggered, but if the application isn't interested it's simply passed to the next one in the chain.

The application composition and routing mechanism developed in SIP Servlet 1.1 is vastly different from the model in SIP Servlet 1.0. The new mechanism introduces a new logical entity called an Application Router (AR). Instead of dispatching SIP requests to servlet applications as a result of complex XML files, the container uses the new AR. A common API is provided so the container can interact appropriately with the AR.

Figure 2 provides a simplified example of how the new application composition mechanism operates in SIP Servlets 1.1. The container gets a SIP request (1) as in the previous example. The SIP request is then passed to the SIP servlet container (2) for further processing. The container uses the common Application Router API to request which SIP Servlet application should get the request (3). In Figure 2, "App 1" is returned and the request is then serviced. It should be noted that the XML mappings for requests are no longer used as the primary selection mechanism. Instead, a default SIP servlet in the application is always triggered. Once App 1 has finished with the request the process is repeated (4-8) until no more applications want to make use of the SIP request. At this point the request can be routed externally (9).

The AR is an interesting concept that lets implementations be flexible in how they select and compose services (e.g., a push-to-talk service could consist of several SIP servlet applications, say conferencing, presence, and floor control). The decision as to exactly what is returned by the AR to the SIP servlet container could be very simple or based on profile information from a Home Subscriber Server (HSS) or some other data store. As long as the AR exposes the standardized API interface to

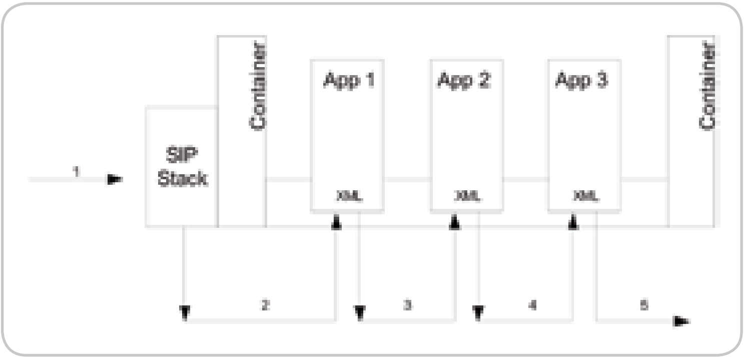


Figure 1 Application composition in SIP Servlet 1.0

the container, it's free to do what it wants when making decisions. With the new standards definition, service providers will most likely be able to tailor an AR to act in an appropriate manner based on any number of required variables. This approach provides an extremely powerful alternative to the static XML mappings provided in SIP Servlet 1.0.

It's fact that a B2BUA is the most popular form of SIP servlet application, and yet it's not all that easy to construct. It was obvious that application developers needed a helping hand to produce such functionality easily. A new B2BUA helper class has been introduced in SIP Servlets 1.1 specifically for this purpose.

A B2BUA consists of multiple protocol sessions that have to be linked to mimic a single-network entity. The B2BUA helper class provides convenient methods that enable implicit linking to take place that reduces complexity and eases the flow between linked protocol sessions in a B2BUA application. The B2BUA helper class also provides previously unavailable functionality, including the ability to send multiple SIP provisional response codes when acting as a User Agent Server (UAS). This was an important feature that suffered from limitations in SIP Servlets 1.0 that has now been addressed.

This article has highlighted the weakness of the encode-URI mechanism and the ability to specifically associate new protocol sessions with an existing application session. SIP Servlets 1.1 introduces a new "Session Targeting" feature that lets an application specify a protocol session association with an existing application session. Session Targeting is achieved by including a specialized Java annotation in applications that checks the protocol session associated with a new SIP request and makes a decision on whether to associate or create a new application session. When applied to conferencing, for example, a user dialing into a conference instance can now be part of an existing application session if required.

One of the primary benefits of the new Session Targeting over the previous Encode-URI mechanism is that it has no impact on the application composition chain. Requests containing an encoded URI are immediately directed to the correct application instance, while session-targeted requests traverse the composition path normally until they reach the appropriate application (as denoted using the Java annotation). The ability to traverse the full composition path and not skip vital steps provides a powerful protocol session association mechanism.

Convergence was a key theme introduced in SIP Servlets 1.0 and continues to be important in 1.1. As discussed, a converged HTTP and SIP container doesn't holistically fulfil the requirements of SOA/J2EE-based telecom solutions. The term "convergence" is redefined in SIP Servlets 1.1 to accommodate both a converged HTTP/SIP container and include convergence from a J2EE perspective. For example, an enterprise archive (EAR) that contains both SIP and HTTP servlets that communicate using EJB is now considered a "converged" application. This provides a true holistic definition of convergence.

A number of subtle changes were required to accommodate such a redefinition. First, the primary interface used to create and carry out SIP functions (SipFactory) is now made available via a specific Java annotation (@SipFactory) as well as from the Servlet Context (as defined in SIP Servlet 1.0). The annotation providing SIP functionality can then be used anywhere in a converged application (convergence as previously defined in this paper).

It's also important to note that a converged application can access an existing instance of an application. Each application session is uniquely identified by a key. SIP Servlets 1.1 lets a converged application access an application session using the getAppplicationSession method available via the SipSessionUtil interface. The method call lets a converged application retrieve the session using the unique application session identifier and is returned the appropriate access to a unique Application Session instance. It functions like 'SipFactory' access, so this utility is made available to the converged components of a converged application and can be injected via a specific Java annotation (@SipSessionsUtil).

Clearly, annotations and the resource injection available in Java 5 are needed by the SIP Servlet 1.1 specification. For this reason Java 5 is the mandatory supported platform.

Backwards compatibility has also been a major consideration in designing the next generation of SIP servlet containers. While every effort has been made to maintain legacy applications, there are certain areas that have been clarified in both the API and container behavior. The bottom line is that a well-behaved SIP Servlet 1.0 application can be successfully deployed in a SIP Servlet 1.1-compliant container. In all other cases some small code modifications might be required.

While this discussion isn't a definitive guide to the changes incorporated in JSR 289, it does highlight some of the primary modifications. These changes help to form a more complete picture of the technological direction of SIP servlet containers.

Conclusion

SIP servlet technology was originally created to provide an appropriate platform for rapidly creating telecommunications services and improving the application development infrastructure. It's true to say that the first generation of SIP servlets achieved its objective by establishing a sound technology platform that has seen wide adoption and a high level of implementation from both container and application developers. It's also clear that, as the technology and deployments evolved, those first-generation servlets needed to be revised to take advantage of new innovations and the insights gained by working with SIP Servlets 1.0.

SIP Servlets 1.1 will mark a new departure in telecommunications technology. The ability to provide true convergence in an application and to compose service-level constructs will provide a powerful tool that will aid Java developers and carriers for years to come. In conjunction with other new features such as B2BUA help, an API cleanup, and SIP access from a converged application, SIP servlet technology will now offer the versatility to move SIP application development to the next level.

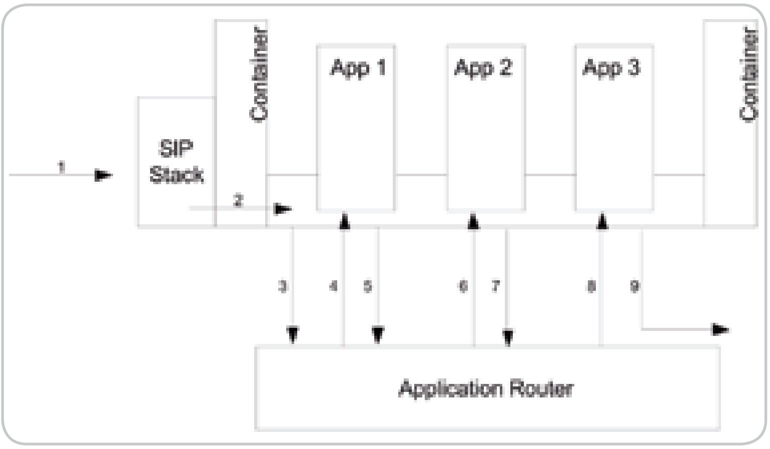


Figure 2 Application composition in SIP Servlets 1.1

References

- http://www.jcp.org/en/jsr/detail?id=116
- http://www.jcp.org/en/jsr/detail?id=289

Help is here.

At last!

- *Lightweight*
- *Java™ browser SDK*
- *Based upon Mozilla™*
- *Faithful robust rendering*

Swing Edition also features:

- *Pure Swing lightweight rendering*
- *Web Standards compliance*
- *W3C DOM*
- *Easy to embed*
- *Java Applet and plugin support*
- *Extensive API*

All WebRenderer products support the following standards: HTML, XHTML, CSS 1 & 2, JavaScript (inc AJAX), DOM, HTTP, HTTPS, XML, XSL, plugins etc. with faithful and always predictable rendering. Check out our other products:

***PLUS** Desktop Edition*

- Supports *Internet Explorer™*, *Mozilla™* & *Safari™* spawns
- Fast and robust SDK
- Micro edition available
- Native font support
- Faithful, predictable rendering

***PLUS** Server Edition*

- Server optimized and scalable
- J2EE support
- Leading server-side Java browser
- Render to thumbnail image
- Headless print support SDK

JadeLiquid™ Software

DOWNLOAD FREE 30 DAY TRIALS NOW! WEBRENDERER.COM

Introduction to **Maven**

Part II – How to use Maven in product development

by Murali Kashaboina
and Geeth Narayanan

In part one of this article we discussed the core competencies of Maven and how it fits into application development and build management. Now we'll delve into some of the practicalities in exercising Maven using a practical example that will walk you through some of the common tasks encountered in typical project development and how they can be done using Maven.

Downloading & Installing Maven

The latest version of Maven 2 can be downloaded at http://maven.apache.org/download.html, Maven's official site. When this article was written, Maven 2.0.6 was the latest version available. It can be downloaded and installed following these steps:

1. Download the appropriate binary file
2. Unarchive the binary file and install it in a local directory
3. Add a maven2InstallationDirectory /bin directory to the system PATH variable
4. By default, Maven 2 uses ~/.m2/ repository as its local repository directory. In case the local repository should point to some other directory, edit maven2InstallationDirectory/conf/settings.xml, uncomment the <localRepository> element and set its value to point to the path location of the new repository directory. For example, <localRepository>/home/mavenuser/m2_repository<localRepository>
5. Run the command mvn -help at a command prompt to verify that the installation was successful. Information as shown in Figure 1 should be displayed.

Installing the Maven 2 Plug-in for Eclipse IDE

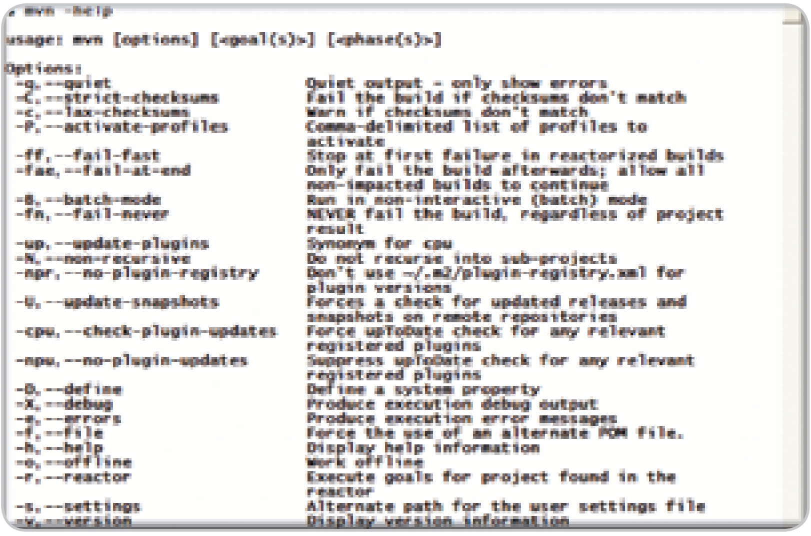
We'll use Eclipse to demonstrate the example application. The first

thing to do would be to install the Maven 2 plug-in for Eclipse so that Maven tasks can be done from inside Eclipse. The benefits of using this plug-in are, as listed on the plug-in's official site (http://m2eclipse.codehaus.org/):

- Launching Maven builds from within Eclipse
- Dependency management for the Eclipse build path is based on Maven's pom.xml
- Resolving Maven dependencies from the Eclipse workspace without installing to a local Maven repository
- Automatically downloading the required dependencies from the remote Maven repositories
- Wizards for creating new Maven projects, pom.xml, or enabling Maven support on plain Java project
- Quick search for dependencies in Maven remote repositories
- Quick fixes in the Java editor for looking up required dependencies/jars by the class or package name

The following steps will walk you through the installation process:

1) On the Eclipse file menu, click on **Help > Software Updates > Find and Install** as shown in Figure 2.
2) The install and update window will open. Select the option Select for new features to install as shown in Figure 3. Click on the Next button.
3) The update sites window will be opened as shown in Figure 4. Click on the New Remote Site button.
4) The new update site dialog window will be opened as shown in Figures 5 and 6. Enter Maven-Eclipse Plugin in the name textbox



```
> mvn -help

usage: mvn [options] [<goal(s)>] [<phase(s)>]

Options:
 -q,--quiet                        Quiet output - only show errors
 -C,--strict-checksums             Fail the build if checksums don't match
 -c,--lax-checksums                Warn if checksums don't match
 -P,--activate-profiles            Comma-delimited list of profiles to
                                   activate
 -ff,--fail-fast                   Stop at first failure in reactorized builds
 -fae,--fail-at-end                Only fail the build afterwards; allow all
                                   non-impacted builds to continue
 -B,--batch-mode                   Run in non-interactive (batch) mode
 -fn,--fail-never                  NEVER fail the build, regardless of project
                                   result
 -up,--update-plugins              Synonym for cpu
 -N,--non-recursive                Do not recurse into sub-projects
 -npr,--no-plugin-registry         Don't use ~/.m2/plugin-registry.xml for
                                   plugin versions
 -U,--update-snapshots             Forces a check for updated releases and
                                   snapshots on remote repositories
 -cpu,--check-plugin-updates       Force upToDate check for any relevant
                                   registered plugins
 -npu,--no-plugin-updates          Suppress upToDate check for any relevant
                                   registered plugins
 -D,--define                       Define a system property
 -X,--debug                        Produce execution debug output
 -e,--errors                       Produce execution error messages
 -f,--file                         Force the use of an alternate POM file.
 -h,--help                         Display help information
 -o,--offline                      Work offline
 -r,--reactor                      Execute goals for project found in the
                                   reactor
 -s,--settings                     Alternate path for the user settings file
 -v,--version                      Display version information
```

Figure 1

Murali Kashaboina is a lead architect at Ecommerce Technology, United Airlines, Inc. He has more than 10 years of enterprise software development experience utilizing a broad range of technologies, including JEE, CORBA, Tuxedo, and Web services. Murali previously published articles in WLDJ and SilverStream Developer Center. He has master's degree in mechanical engineering from the University of Dayton, Ohio.

murali.kashaboina@united.com

and http://m2eclipse.codehaus.org/update/ in the URL textbox. Click OK on the New Update Site dialog window. This will add the new site to the list of sites in the Update sites window. Click the Finish button.

5) The search results window with the Maven Plugin Installation option will be opened as shown in Figure 7. Select the plug-in option and click Next.
6) The Maven 2 plug-in license agreement window will open. Review and accept it and click the Next button (see Figure 8).
7) The installation verification window will be opened as shown in Figure 9. Click the Finish button.
8) The Eclipse download manager will start downloading and installing the Maven 2 plug-in from the specified site. Once the installation is complete, it will prompt you to restart Eclipse. The installation of the Maven 2 plug-in for Eclipse will complete and Maven will be ready to be used after restarting.

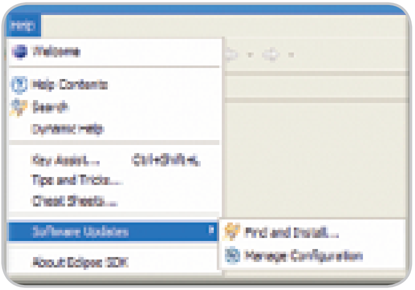
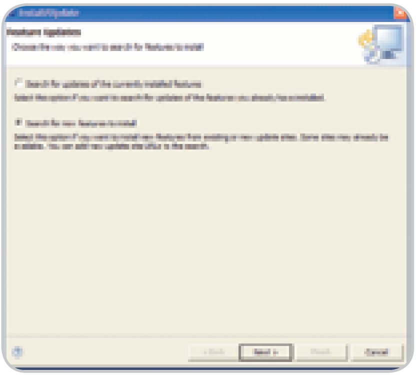


Figure 2


Install/Update

Feature Updates

Choose the way you want to search for features to install

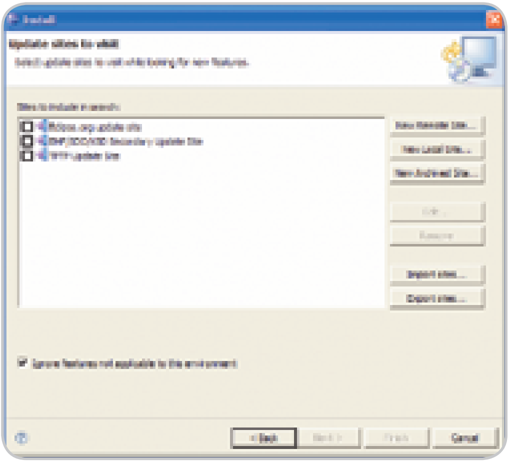
Search for updates of the currently installed features

Select this option if you want to search for updates of the features you already have installed.

● Search for new features to install

Select this option if you want to install new features from existing or new update sites. Some sites may already be available. You can add new update site URLs to the search.

Figure 3


Install

Update sites to visit

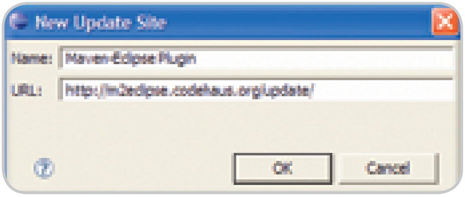
Select update sites to visit while looking for new features.

Sites to include in search:

☐ Eclipse.org update site
☐ EMF (SDO/XSD) Secondary Update Site
☐ WTP Update Site

✓ Ignore features not applicable to this environment

Figure 4


New Update Site

Name: Maven-Eclipse Plugin

URL: http://m2eclipse.codehaus.org/update/

Figure 5

Hands-on Example

In this article we'll make an attempt to provide an example that will walk you through some of the common build tasks encountered in typical project development and how they can be executed using Maven.

Most J2EE developers might have been involved in creating and deploying J2EE deployable units such as JARs, WARs, RARs, and EARs. In our example, we considered a J2EE scenario that contains

THE | POWER | TO CREATE | REALITY™

Go Ahead...

TOUCH!

GL Studio for Java, developed by DiSTI, lets you quickly create real-time, interactive graphical elements for your applets and web pages. Now your clients can interact with your products in a way never before possible.

GL Studio for Java enables you to generate pure Java code for rapid, client-side rendering. The result is high fidelity, realistic graphic solutions and a distinct competitive advantage for your organization. The power to create reality is within your reach.

Go Ahead... TOUCH at: www.simulation.com/jdj

DiSTI

www.simulation.com

11486 Corporate Blvd. Suite 190, Orlando, FL 32817 • PH: 407-206-3390

Archetype	Group ID	Description
maven-archetype-quickstart	org.apache.maven.archetypes	Creates a default project for a Java application
maven-archetype-j2ee-simple	org.apache.maven.archetypes	Creates a project for a simple J2EE application
maven-archetype-webapp	org.apache.maven.archetypes	Creates a project for a Web application
maven-archetype-mojo	org.apache.maven.archetypes	Creates a project for Maven plug-in development
maven-archetype-simple	org.apache.maven.archetypes	Creates a project for a simple application
maven-archetype-portlet	org.apache.maven.archetypes	Creates a project for a portlet application

Table 1

POM for 'xmlBinding" Submodule

```
<project>
<parent>
 <artifactId>EmployeeInfo</artifactId>
 <groupId>com.somecompany</groupId>
 <version>1.0</version>
</parent>
 <modelVersion>4.0.0</modelVersion>
 <groupId>com.somecompany</groupId>
 <artifactId>xmlBinding</artifactId>
 <name>xmlBinding</name>
 <version>1.0</version>
 <url>http://maven.apache.org</url>
<dependencies>
 <dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>3.8.1</version>
  <scope>test</scope>
 </dependency>
</dependencies>
</project>
```

POM for parent 'EmployeeInfo' module

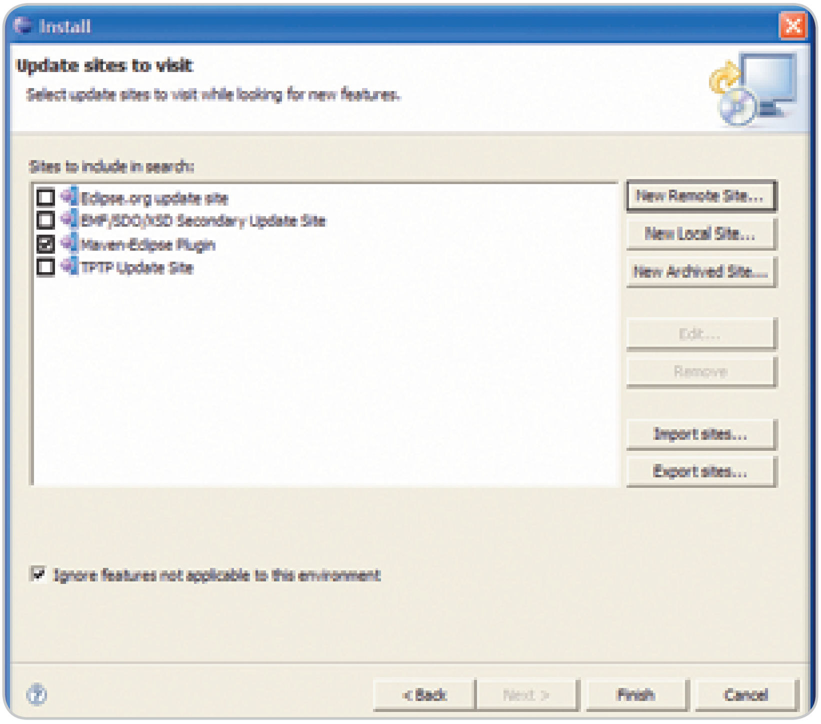
```
<project >
 <modelVersion>4.0.0</modelVersion>
 <groupId>com.somecompany</groupId>
 <artifactId>EmployeeInfo</artifactId>
 <packaging>pom</packaging>
 <version>1.0</version>
 <name>EmployeeInfo</name>
 <url>http://maven.apache.org</url>
<dependencies>
 <dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>3.8.1</version>
  <scope>test</scope>
 </dependency>
</dependencies>

<modules>
 <module>xmlBinding</module>
</modules>
</project>
```

Table 2

Geeth Narayanan is a senior architect at Ecommerce Technology, United Airlines, Inc. He has 10 years of experience in the IT industry, specializing in solutions using Java EE technologies. Geeth has master's degree in electrical engineering from the University of Toledo, Ohio.

geethakrishn.narayanan@united.com



Update sites to visit

Select update sites to visit while looking for new features.

Sites to include in search:

- [ ] Eclipse.org update site
- [ ] EMF/SDO/XSD Secondary Update Site
- [✓] Maven-Eclipse Plugin
- [ ] TPTP Update Site

[✓] Ignore features not applicable to this environment

Figure 6

a simple Web application that gets and displays employee details using employee ID. Even though the use case is fairly simple, to illustrate a few of the cornucopia of features available in Maven, we added some complexity by including other components such as EJB, RAR, and an XSD-bound data model that are typically encountered in most J2EE-based applications. The sample application, known as "EmployeeInfo," consists of the following modules:

- The xmlbinding module contains an XSD schema that is used to generate XML-bound Java classes using Castor. This is another concept that had been widely used in Java projects lately that we thought would be useful to illustrate. This module generates a JAR artifact containing XML-bound Java classes.
- The connector module contains a JCA adapter that fetches employee details. This module generates a RAR artifact and uses XML objects interally to represent employee information. Of course, it returns hard-coded values for the details, since the purpose of this example is to illustrate Maven's use in different developmental scenarios.
- The ejb module contains a GetEmployeeDetailsEJB stateless session bean. This module generates a EJB JAR artifact and internally invokes a JCA connector to retrieve an employee info XML object.
- The web contains the JSPs that take in the employee ID user input and displays employee details by calling a stateless session EJB. This module generates a War artifact containing the web application.
- The ear module constitutes the enterprise application that contains the Web application, enterprise java beans, connector, and other dependencies as needed.

In this installment of the article, we will explain how to set up the top level project directory, the 'xmlBinding' module directory structure and how to execute Ant tasks during Maven build process. The rest of the module

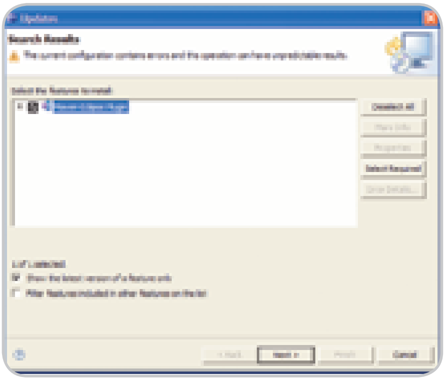


Figure 7

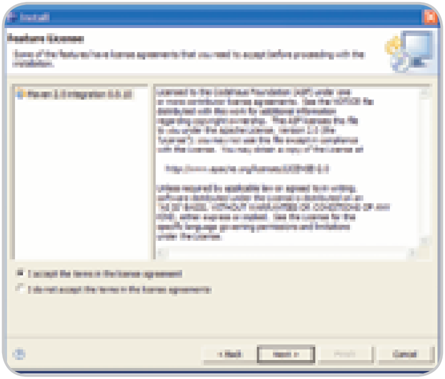


Figure 8

details along with examples of typical developmental tasks that can be easily accomplished using Maven will be elaborated in the final installment of the article to be published in the next issue of *JDJ*.

Using Maven Archetypes To Create a Top-Level Project Directory

Archetype is a project templating toolkit. There are built-in Archetypes in Maven that will let developers quickly create a standard project directory layout based on project type. For example, there are Archetypes to create WAR projects and Maven plug-in projects. Table 1 shows some common Archetypes used with Maven. There's a comprehensive list of Archetypes at http://docs.codehaus.org/display/MAVENUSER/Archetypes+List.

Archetype by itself is a Maven plug-in component with typical Maven artifact coordinates. Unlike regular Maven plug-ins that run during Maven build lifecycle phases, An Archetype plug-in gets executed when a Maven project is created. Using Maven Archetypes in creating a project directory structure inculcates consistency in the way project directories are laid out and saves time deciding what the project structure should be.

We'll use the built-in maven-archetype-quickstart arechetype to create a directory structure representing a typical Java project. Open a command prompt and change the directory to the working directory where the project directory should be created. At the command prompt, execute this Maven command:

```
mvn archetype:create -DarchetypeGroupId=org.
apache.maven.archetypes -DarchetypeArtifactId=mav
en-archetype-quickstart -DgroupId=com.somecompany
-DartifactId=EmployeeInfo -Dversion=1.0
```

On the command we provided the Maven coordinates for our project; com.somecompany as the group ID, EmployeeInfo as artifact ID, and 1.0 as the artifact version. Executing the command above creates this directory structure as shown in Figure 10.

A default POM file is created for our project as shown below. The directory structures src/main/java/ and /src/test/java are created with directories for the default package. Note that Maven Archetype uses a group ID for default package names unless a –DpackageName property is explicitly specified on the command. Note that by default Archetype includes Junit as a dependency and creates App.java in the main source directory and a Junit test case AppTest.java in the test source directory:

```
<project xmlns="http://maven.apache.org/
POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://
maven.apache.org/POM/4.0.0 http://maven.apache.
org/maven-v4_0_0.xsd">
 <modelVersion>4.0.0</modelVersion>
 <groupId>com.somecompany</groupId>
 <artifactId>EmployeeInfo</artifactId>
 <packaging>jar</packaging>
 <version>1.0</version>
 <name>EmployeeInfo</name>
 <url>http://maven.apache.org</url>
 <dependencies>
  <dependency>
   <groupId>junit</groupId>
   <artifactId>junit</artifactId>
   <version>3.8.1</version>
   <scope>test</scope>
  </dependency>
 </dependencies>
</project>
```

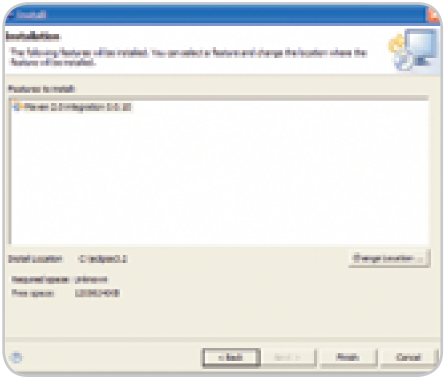


Figure 9

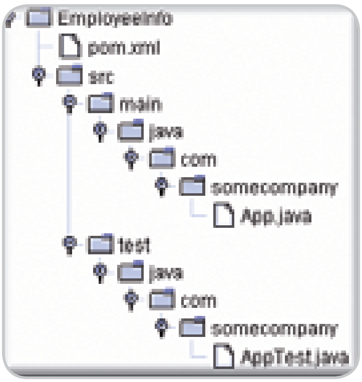


Figure 10

Importing a Maven Project into Eclipse

The following steps show how to import a Maven project into Eclipse:

1. The first thing would be to create

Figure 11

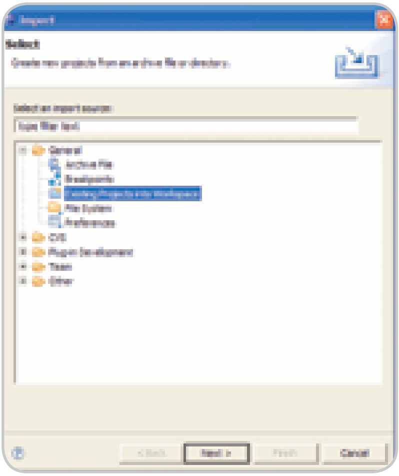
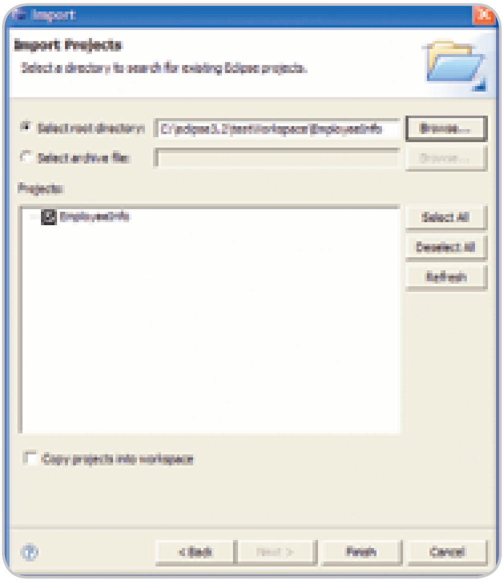


Figure 12



Import

Import Projects

Select a directory to search for existing Eclipse projects.

Select root directory: C:\eclipse3.2\test\workspace\EmployeeInfo

Select archive file:

Projects: EmployeeInfo

Copy projects into workspace

Figure 13

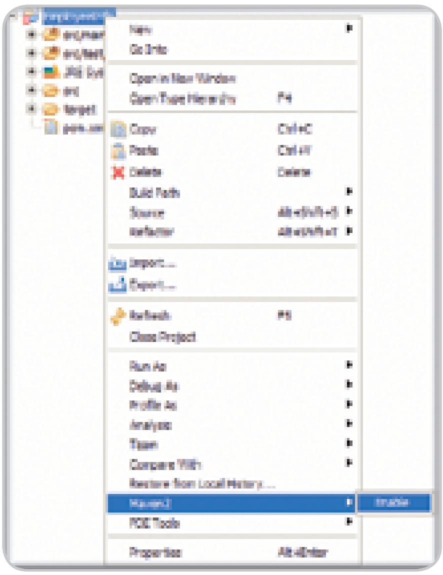


Figure 15

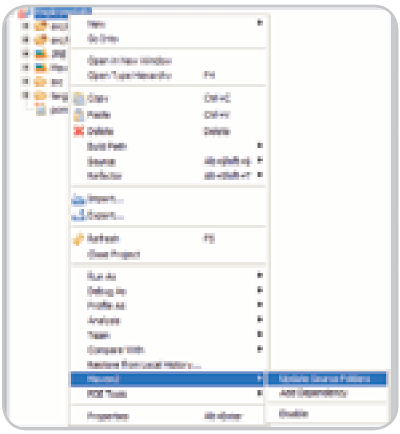


Figure 16

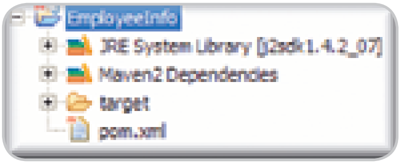


Figure 17

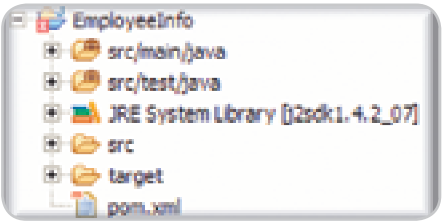


Figure 14

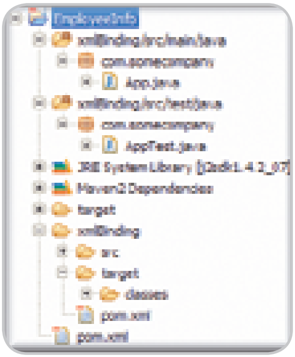


Figure 18

Eclipse-specific project files based on Maven's POM information so that Maven project can be imported into Eclipse. This is fairly easy to do using Maven's Eclipse plug-in. At the command prompt change the directory to the EmployeeInfo directory and run the Maven command to execute the Eclipse plug-in with eclipse goal as shown in Figure 11. Executing the Eclipse plug-in with an eclipse goal will result in creating the .class-path and .project files that Eclipse uses.

2. Import the newly created project into Eclipse. Click on **File > Import** to open the Eclipse import wizard and select Existing Projects into Workspace as shown in Figure 12.
3. In the next window browse through the directory to select the EmployeeInfo project directory to be imported. Optionally, you can select the Copy projects into workspace checkbox if you have to copy the project directory (see Figure 13).
4. Eclipse will import and open the project in the explorer package as shown in Figure 14. Note that Eclipse displays a project failure flag indicating that the project isn't Maven-enabled yet
5. The next step would be to enable Maven for the project by right clicking on the project and selecting **Maven > enable** option as shown in the Figure 15.
6. Once the project is Maven-enabled, Maven tasks can be done from Eclipse. The first thing would be to update the Maven source directories using the Maven option as shown in Figure 16. Enabling Maven and updating the Maven sources will set the Maven class-path to be used according to Eclipse and will update all the source directories including the test source directories.
7. Now that we have a parent directory structure laid out for the EmployeeInfo project, we have to make a slight change to the POM so it can start accepting other modules as children. By default the POM file has a jar packaging type. The packaging type for a parent POM that can contain other Maven modules should be set to

"Archetype is a project templating toolkit. There are built-in Archetypes in Maven that will let developers quickly create a standard project directory layout based on project type"

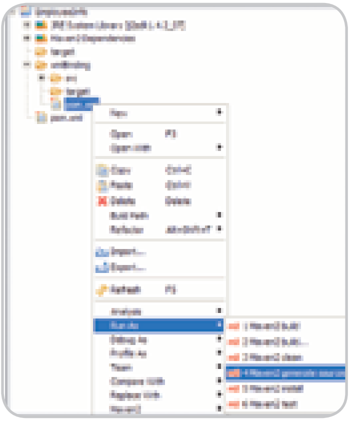
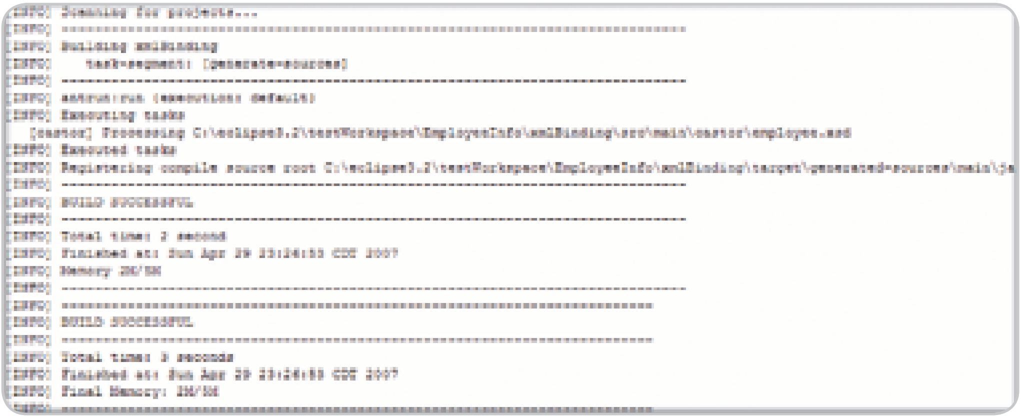


Figure 19


```
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Building xmlBinding
[INFO]    task-segment: [generate-sources]
[INFO] ------------------------------------------------------------------------
[INFO] antrun:run (execution: default)
[INFO] Executing tasks
  [castor] Processing C:\eclipse3.2\testWorkspace\EmployeeInfo\xmlBinding\src\main\castor\employee.xsd
[INFO] Executed tasks
[INFO] Registering compile source root C:\eclipse3.2\testWorkspace\EmployeeInfo\xmlBinding\target\generated-sources\main\ja
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 2 second
[INFO] Finished at: Sun Apr 29 23:24:53 CDT 2007
[INFO] Memory 2M/5M
[INFO] ------------------------------------------------------------------------
[INFO] ====================================================================
[INFO] BUILD SUCCESSFUL
[INFO] --------------------------------------------------------------------
[INFO] Total time: 3 seconds
[INFO] Finished at: Sun Apr 29 23:24:53 CDT 2007
[INFO] Final Memory: 2M/5M
[INFO] ====================================================================
```
Figure 20

pom. By changing the packaging type to pom, there will be no specific artifact coming out of the parent POM and so the src directory, as it is, can be removed. However, specifying the group ID and version in the parent POM will let all sub-modules inherit such attributes and make the parent POM a central place to specify such attributes. Once the packaging type is changed to pom and the source directories deleted, Maven source directories should be updated using the Maven option shown in Figure 16.

8. The project in Eclipse should look like Figure 17. Note that by default Maven uses a target/classes directory to store main Java classes and a target/test-classes directory to store test Java classes.

Creating Project Modules

Now that we have the parent project it's time to create the individual module projects inheriting the POM from the parent project.

Setting Up a Castor-based xmlBinding Module

Since an xmlBinding module generates a JAR artifact, we need a simple JAR project directory structure. To create this, go to the EmployeeInfo directory on the command prompt and execute the following Maven command:

```
mvn archetype:create -DgroupId=com.somecompany -DartifactId=xmlBinding
–Dversion=1.0
```

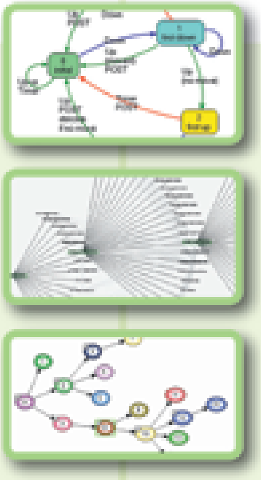
This creates a simple project structure with src/main/java and src/test/java directories containing a sample Java class and its Junit test case respectively. Now refresh the EmployeeInfo project in Eclipse and update the Maven source directories using the Maven option in the right-click menu. Eclipse will add the xmlBinding directory to the project and show both xmlBinding/src/main/java and xmlBinding/src/test/java as Eclipse Java source directories as shown in Figure 18. Since we intend to generate XML-bound Java classes using Castor, the xmlBinding/src/main/java and xmlBinding/src/test/java default Java source directories can be deleted.

Let 's compare the xmlBinding POM file with the parent EmployeeInfo POM file (see Table 2).

The POM for the xmlBinding sub-module has a parent element that refers to the EmployeeInfo module. Also note that the parent EmployeeInfo module refers to the name of the sub-module in the <modules> element. This relationship essentially instructs Maven to look for the parent POM while processing the xmlBinding POM and for all sub-module POMs specified under the <modules> element while processing the EmployeeInfo POM. This sub-module nesting can continue to any level based on the extent of the hierarchical modularization of concerns.

Build interactive diagrams easier than you ever imagined

Create custom interactive diagrams, network editors, workflows, flowcharts and design tools. For web servers or local applications. It's easy-to-use and very flexible. We're the first developer of diagramming components and still the best. Find out for yourself: download our FREE fully functional evaluation kit, with full support at www.nwoods.com.

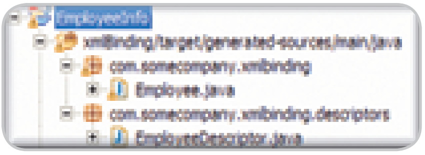


FREE Download With Full Support!

GoDiagram

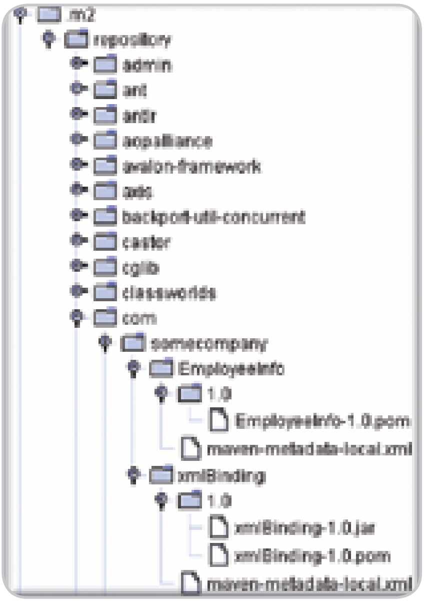
Interactive diagram components

www.nwoods.com



```
EmployeeInfo
  xmlBinding/target/generated-sources/main/java
    com.somecompany.xmlbinding
      Employee.java
    com.somecompany.xmlbinding.descriptors
      EmployeeDescriptor.java
```

Figure 21



```
m2
  repository
    admin
    ant
    antlr
    aopalliance
    avalon-framework
    axis
    backport-util-concurrent
    castor
    cglib
    classworlds
    com
      somecompany
        EmployeeInfo
          1.0
            EmployeeInfo-1.0.pom
          maven-metadata-local.xml
        xmlBinding
          1.0
            xmlBinding-1.0.jar
            xmlBinding-1.0.pom
          maven-metadata-local.xml
```

Figure 22

Comparing the two POMs it's evident that both declare the same Junit dependency and specify the same <version> element artifact. It's common that all sub-modules under the one main parent module get released under one version. In this context, it becomes redundant to declare versions in the sub-modules. Maven comes to rescue in such situations by providing a project inheritance model in which sub-modules can inherit build process state attributes from the parent module – group ID, version, dependencies, and other deployment information. Since the xmlBinding module already declares EmployeeInfo as the parent in the POM, to take advantage of project inheritance we just have to make slight changes to the xmlBinding POM. Essentially removing the version, group, and common dependency information from the POM. Below is the modified POM for xmlBinding:

```
<project>
 <parent>
  <artifactId>EmployeeInfo</artifactId>
  <groupId>com.somecompany</groupId>
  <version>1.0</version>
 </parent>
 <modelVersion>4.0.0</modelVersion>
 <artifactId>xmlBinding</artifactId>
 <name>xmlBinding</name>
</project>
```

The next step would be to create the XSD and use Castor to generate sources for XML-bound Java classes. Create a new folder in Eclipse called castor under the xmlBinding/src/main directory. Then create the XML schema file employee.xsd under the newly created castor directory. The employee.xsd will contain following schema structure:

```
<xs:schema version="1.0"  xmlns:
xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://schema.somecompany.com/"
targetNamespace="http://schema.mytestcom-
pany.com/" elementFormDefault="qualified"
attributeFormDefault="unqualified">
 <xs:element name="employee">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="id" type="xs:
string"/>
    <xs:element name="salary" type="xs:
string"/>
   </xs:sequence>
   <xs:attribute name="firstName"
type="xs:string" use="required"/>
   <xs:attribute name="lastName"
type="xs:string" use="required"/>
   <xs:attribute name="department"
type="xs:string" use="required"/>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Next create an employeeBinding.xml Castor binding file in the xmlBinding/src/main/castor directory with the following binding information:

```
<cbf:binding xmlns:cbf="http://www.castor.
org/SourceGenerator/Binding" defaultBindin
gType="element">
 <cbf:package>
  <cbf:name>com.somecompany.xmlbinding</
cbf:name>
  <cbf:namespace>http://schema.somecom-
pany.com/</cbf:namespace>
 </cbf:package>
</cbf:binding>
```

Running Ant Tasks

Now it's time to generate Java sources using the Castor binding. There are different ways to do this using Maven. We chose to use a Castor Ant task to generate Java sources to illustrate how Ant tasks can be executed using maven-ant-run-plugin. The Maven artifact for the Castor Ant task can be found in the http://repository.codehaus.org repository. We'll use version 1.1 of the castor-codegen-anttask artifact that's under the org.codehaus.castor group. To use this artifact, we must first add the Codehaus repository information in the POM file as follows:

```
<repositories>
  <repository>
   <snapshots>
    <enabled>false</enabled>
   </snapshots>
   <id>central</id>
   <name>Maven Central Repository</name>
   <url>http://repo1.maven.org/maven2</
url>
  </repository>
  <repository>
   <id>codehaus </id>
   <name>Codehaus Maven Repository</name>
   <url>http://repository.codehaus.org/</
url>
  </repository>
</repositories>
```

Next add runtime dependencies in the POM as shown below needed

"We chose to use a Castor Ant task to generate Java sources to illustrate how Ant tasks can be executed using maven-antrun-plugin"

“To illustrate a connector’module that yields a RAR artifact we’ll start with a default Maven project”

to execute Castor tasks:

```
<dependencies>
  <dependency>
    <groupId>org.codehaus.castor</groupId>
    <artifactId>castor</artifactId>
    <version>1.1</version>
  </dependency>

  <dependency>
    <groupId>org.codehaus.castor</groupId>
    <artifactId>castor-codegen-anttask</arti-
factId>
    <version>1.1</version>
    <scope>runtime</scope>
  </dependency>
</dependencies>
```

Below is the core part where we’ll add maven-antrun-plugin, which can be used to execute Ant tasks. The plug-in can be added to the POM as shown below:

```
<build>
 <plugins>
   <plugin>
     <artifactId>maven-antrun-plugin</arti-
factId>
     <executions>
       <execution>
         <phase>generate-sources</phase>
         <goals>
              <goal>run</goal>
         </goals>
         <configuration>
              <tasks>
                          <task-
def classname="org.castor.anttask.
CastorCodeGenTask" name="castor">
                                <classpath
refid="maven.test.classpath"/>
                                </taskdef>
                                <castor
types="j2"

 warnings="false"

   todir="${project.build.directory}/gener-
ated-sources/main/java"

  bindingfile="${basedir}/src/main/castor/
employeeBinding.xml">


<fileset dir="${basedir}/src/main/castor/">


<include name="*.xsd"/>


</fileset>
                                </castor>
                      </tasks>
                      <sourceRoot>${project.
build.directory}/generated-sources/main/
java</sourceRoot>
              </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
 </build>
```

As evident from the POM snippet above, the plug-in is configured to execute the run goal of the plug-in when the generate-sources lifecycle phase is triggered. The maven-antrun-plugin will execute any Ant tasks declared in the configuration/tasks element. In this case we define an Ant task named castor using the org.castor.anttask. CastorCodeGenTask Java class. Note that we refer to the maven.test.classpath provided by Maven as a runtime property. This particular classpath will contain all the dependencies including test dependencies. While invoking the declared castor task, we refer to some of the other properties provided by Maven such as the project.build. directory. By default the project.build. directory will point to the currentProjectModule/target directory where project outputs such as classes and reports are usually stored. Note that the POM state information is accessible in the form of runtime properties. For example, the version of the current artifact can be referred to the project. version property. And note that the generated source files will be put in the ${project.build.directory}/generated-sources/main/java directory and by specifying the <sourceRoot>’ element under the plug-in configuration, we tell the Maven plug-in to add a generated source directory as a project source directory. To generate sources right-click on the xmlBinding POM file in Eclipse and then in the Run As options select Maven 2 generate-sources as shown in Figure 19.

Eclipse will run a generate-sources lifecycle phase on the xmlBinding POM. The Eclipse console output from the Maven execution will be as shown in Figure 20.

The Castor-generated source files can be found under the target directory and when Maven sources are updated using Maven option in the right-click menu, Eclipse should display the generated source directory in the package explorer as shown in Figure 21.

To deploy the xmlBinding JAR artifact to the local repository for the first time right-click on the parent EmployeeInfo module’s POM file in Eclipse and then in Run As options select Maven 2 install. This will run the Maven lifecycle phases till the install phases for the parent and for all the child modules. Maven builds the JAR artifact and copies it over to the local repository along with the POM information as shown in the Figure 22.

Conclusion

In this installment of the article, we have shown how to download and install Maven 2, how to install Maven 2 plugin for Eclipse and how to go about setting up a project directory structure using Maven 2. We used a simple use case of displaying employee details on the web given an employee ID but deliberately made the design a bit complex by introducing design concepts such as XML binding, EJBs, and JCA connectors to illustrate a few of the many features offered by Maven. In the final installment of the article, we will discuss remaining modules from the example application and will illustrate how Maven 2 helps in accomplishing typical development tasks in a fairly easy manner that otherwise would demand significant time and effort to accomplish.

Effective Development of Java Conformance Tests with Meta-programming

Java Annotations
+ Compiler API
+ Annotation Processing

Remarkable Results

by Dmitry A. Fazunenko

This article presents a case study of the use of meta-programming in Java compatibility testing. It shows how parts of the source code can be shared between different products and modified to generate programs targeting specific functions and describes the approach Sun Microsystems has used for building Technology Compatibility Kits (TCK) for more than five years.

In modern Java TCKs, testing components for multiple products are stored in a single repository not as pure Java language, but as XML files where code is annotated with attributes called metadata. A set of tools is used to select only tests that are applicable for a particular product and transform them into final Java programs. These post-processing tools written in Java provide a simple and efficient way to generate testing components that match specific product needs.

Here we'll illustrate Sun's approach by providing examples of test storage formats and the processing scheme used by the conversion tools. We'll also draw a comparison with Sun's former approach, which used Perl for test generation, and explain why Sun moved to the XML-based methodology.

TCKs

The international community develops and evolves Java technology specifications using the Java Community Process (JCP). The JCP produces high-quality specifications in Internet time, using an inclusive consensus-building approach that produces a specification; a reference implementation that demonstrates that the specification can be implemented; and a technology compatibility kit, or TCK, a suite of tests, tools, and documentation used to test implementations for compliance with the specification.

There are as many TCKs as there are Java technologies, and these technologies might be deeply intertwined despite their differences. For example, both the Java Platform Standard Edition (SE) and the Java Platform Micro Edition (ME) include the Java Virtual Machine (JVM) and the java.lang package, so most tests written for the Java SE VM can be reused to test the Java ME VM.

Each separate TCK has its own requirements for tests such as performance targets or maximum memory size. As a rule, the kernel of the test is the same across TCKs, but the test might take different forms. For example, for Java SE several tests can be combined in one class; while for Java ME, due to a restriction on class file size, each test case should be put in its own class.

Sun Microsystems has a long history of successful TCK development. During the past 10 years, about 100 TCKs with millions of tests were shipped. This success would have been impossible without effective reuse or sharing of tests between TCKs. At the heart of the approach applied at Sun for TCK development is meta-programming, which implies that sources are generated from metadata.

This article describes how this approach is implemented. Even though the area of conformance testing is narrow, the underlying principles can be generalized:

- All tests are stored as templates, not as tests to be included in the product.
- Each test is represented as code with attributes (meta information).
- Templates are stored in a single repository shared across different TCKs.
- Tests applicable for a particular technology are selected by attributes at the TCK build stage based on selection criteria set by the TCK.
- All TCKs use the same tools for extracting test sources from templates, but tools can be customized according to particular TCK needs.
- All TCKs abide by the general policy regulating the rules for writing tests.

TCK Structure

Each TCK product consists of the following components:

- **Test Monitor:** Tool for test configuration and execution
- **Tests:** Programs returning either a passed or failed status
- **Libraries:** Code shared by tests
- **Product Documentation**

Each test consists of:

- **Test sources:** Java source code and data files
- **Class files:** Compiled sources
- **Test descriptions:** HTML files that consist of Test Monitor instructions showing how to execute tests, links to test sources and data files, and human-readable test documentation

The minimal test unit is a method or Test Case. Test Cases with similar functionality are combined in one class or Test

Dmitry Fazunenko is a lead TCK programmer at Sun Microsystems Inc. His expertise is in developing Technology Compatibility Kits for Java SE platforms. He has been working in this area for a decade and during the past six years, he has focused on developing and improving techniques for effective test development.

dmitry.fazunenko@sun.com

Group. Listing 1 provides a sample fragment of the test source code.

Some tests require arguments be passed or some action performed prior to execution. This information is specified in the Test Description and stored in an HTML file. The Test Monitor reads the Test Description and runs the test according to the instructions. Besides execution instructions, the HTML file contains test documentation, which lets one understand what the test checks for and how without reading the sources.

Here is a sample fragment of an HTML file:

Test Specifications and Descriptions for Applet

* public Applet()
* public void start() List of tested methods
* public void stop()

public Applet()

Assertion	Test Case ID
Constructs Applet instance	Applet001
Throws: HeadlessException - if GraphicsEnvironment.isHeadless() returns true	Applet002

Table 1 Domain testing of class Applet, constructor public Applet()

Test Description

Item	Value
Title	Applet constructor tests
Source	CtorTests.java
ExecuteClass	javasoft.sqe.tests.api.java.applet.Applet.CtorTests
ExecuteArgs	-TestURL $testURL
keywords	runtime positive

Table 2 Test Monitor instructions for running Applet constructor tests

Test Development

Tests aren't only the largest part of any TCK, but the main component. Their number and quality define how well the specification is covered and, as a result, the likelihood of incompatibilities in the implementation. Some TCKs are very large. For example, the TCK for Java SE 6.0 contains more than 100,000 tests. Effective management of such big products is impossible without applying special techniques that let one to do mass updates or easily add or remove tests without breaking other functionality.

To develop a high-quality TCK you have to produce a huge number of tests. There are three ways of creating new tests: developing new ones, reusing existing tests, or automatically generating tests from the specification. The last option is the best, but unfortunately it exists only in theory. Reusing tests is cheaper than developing new ones so the development of most TCKs starts with a search for existing tests that can be used. The more tests you can reuse, the less you need to develop. But when dealing with thousands of files, the task becomes daunting.

Java technologies tend to evolve and most of them have several releases. Each release requires a separate TCK that might include tests for new features as well as bug fixes and tests that improve the coverage in existing areas. So it's important to be able to include tests easily and incorporate newly developed code and bug fixes. Meta-programming can help.

Meta-programming

Wikipedia defines meta-programming as:

The writing of programs that write or manipulate other programs (or themselves) as their data or that do part of the work that is otherwise done at compile time during runtime.
-http://en.wikipedia.org/wiki/Metaprogramming

Representing the test as data for further processing enables the program to achieve the expected results by modifying control programs of several kilobytes, not by adopting thousands of tests. When all existing and applicable tests are found, tests are decorated according to TCK requirements and HTML files don't contain orphans.

Additionally, not keeping data that can be calculated guarantees consistency and enables the program to write less code. It also enables the generation of similar tests from one template.

Honestly, it's more convenient for engineers to develop code in pure Java than in any other format. While applying this approach will increase development time minimally, it will save resources for further reuse of tests by other TCKs.

Sun represents a TCK test as its code and related attributes (meta-information). In the past Sun used Perl as a language for test data definitions. Developers wrote simple Perl programs consisting mostly of variable settings and invocations of library methods:

```
require $ENV{TESTGEN}; # import test generation library
$package = "applet";         # meta information describing
$class = "Applet";           # api under test
$method = "public Applet()"; #
$code = '{
a = new Applet();
if (a.isVisible() == true && a.isEnabled() == true
&& a.isValid() == false) { // Check result
return Status.passed("OKAY");
} else {
return Status.failed("Incorrect ' . $class . ' object created");
}
}';
&gen; # invocation of test generation method
```

All generation functionality was implemented in the Perl library. It was possible to customize generation by defining the TESTGEN environment variable pointing to the alternative library. Behind the simplicity are a number of serious restrictions: the necessity of manually synchronizing generation libraries between different TCKs, the difficulty in reading test attributes, and unchecked variable names.

"The international community develops and evolves Java technology specifications by using the Java Community Process (JCP)"

The XML language fits the purpose of specifying meta-info better. The XML language lets us keep data in a well-defined format unified across all TCKs. In its turn the unified format results in a set of shared tools that implement processing functionality common for all TCKs (see Listing 2). Tools are customizable, so all TCKs use the same tools and just provide their own specific plug-ins.

Figure 1 demonstrates how tests are processed during the build.

"Meta-programming is power"

Tests for all TCKs are stored in a common **Test Repository** so newly developed tests as well as bug fixes made by one TCK are available for all TCKs. A set of tools is shared between all TCKs: the **Filtering Tool** and the **Generation Tool**. TCK provides **test selection criteria** to the **Filtering Tool**, which selects only those tests applicable to the technology. Selected tests are passed to the **Generation Tool**, which generates test sources as they need to be presented in TCK. The **Generation Tool** is implemented to extract tests from templates in a way suitable for most TCKs, but it can be customized when needed through **test generation customization**.

In this approach, TCK needs to define only its own **test selection criteria** and **test generation customization**. Test sources and tools can be taken from a general repository.

Implementation

This section covers the technical implementation of shared tools.

- **Internal Test Representation:** A DTD defines about 50 elements that can be used in XML test descriptions. For each element a corresponding Java class was created. The specially developed parser reads XML test descriptions and creates a Java object representing the test. An XML emitter can write Java objects back to XML. This technique is known as marshaling and unmarshaling. It lets tools use the Internal Representation API and doesn't involve XML.
- **Filtering Tool:** The purpose of the Filtering tool is to select tests that apply to a specific technology. Based on TCK properties, the Filtering tool creates an instance of *AttributeFilter:*

```
public interface AttributeFilter {
    public boolean accept(TestGroup tg);
    public boolean accept(TestGroup tg, TestCase tc);
}
```

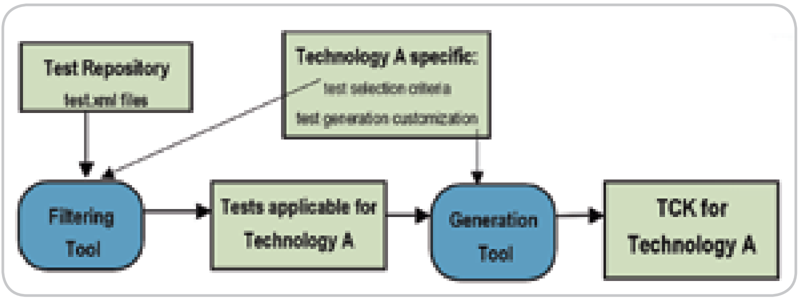


Figure 1 Scheme of the test processing

Then, the Filtering tool parses the passed XML file to get an instance of a TestGroup object. After that, it applies *AttributeFilter* to all TestCases in the TestGroup to detect which of them doesn't meet the selection criteria. Rejected TestCases are removed from the TestGroup. The whole TestGroup can be rejected if it doesn't meet the selection criteria or if all the TestCases are rejected. When *AttributeFilter* has been applied, the Filtering Tool returns the TestGroup to XML for further processing. Newly created (intermediate) XML necessarily contains only tests that apply to the technology.

With Internal Representation, the API development of TCK-specific filters is very simple. The following is an example of the filter that accepts tests marked with Java Architecture for XML Binding (JAXB) technology version 2.0 and compatible versions:

```
public class JaxbTechnologyFilter implements AttributeFilter {
    String ver = “2.0”;
    public boolean accept(TestGroup tg) {
        return FilterUtil.checkTechnology(tg.getTechnologies(),
“JAXB”, ver);
    }
    public boolean accept(TestGroup tg, TestCase tc) {
        return FilterUtil.checkTechnology(tc.getTechnologies(),
“JAXB”, ver);
    }
}
```

It's unnecessary to list all technologies to which the test applies. A test can be selected or rejected by the class or method under test.

The following example demonstrates how to select tests that cover only the JAXB API specification (the javax.xml.bind package):

```
    public boolean accept(TestGroup tg) {
      String testedPackage = tg.getTestedPackage();
      return testedPackagetg.getTestedPackage().startsWith(“javax.
      xml.bind”);
 }
```

Generation Tool: The purpose of the Generation tool is to generate test sources to include in the product. The functionality of the Test Generator tool itself is small enough. It does the preliminary work, such as parsing XML, detecting the type of test, and instantiating and invoking an appropriate test emitter. The emitter does the real work of test generation. All test generation emitters implement the JavaEmitter interface:

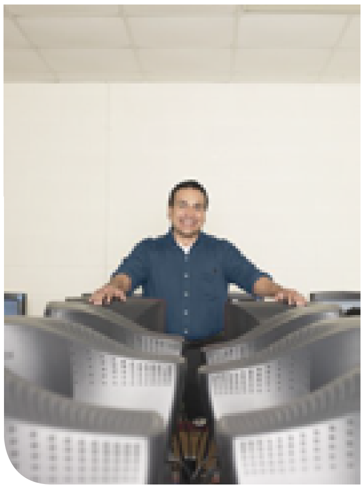
```
public interface JavaEmitter {
    public void generate(TestGroup[] testGroups);
}
```

The default JavaEmitter generates one HTML file for all passed TestGroups. The JavaEmitter checks all test at-

RealWorld JAVA SEMINAR
REALWORLDJAVA.COM
THE LARGEST JAVA DEVELOPER EVENT ON THE EAST COAST

Register Today!
Early Bird: Save $100
www.realworldjava.com

A Roadmap for Java Professionals



Hands-On Education:
The Largest Java Developer Event on the East Coast

A seasoned Java professional has to know more than just the syntax of the Java language. Java EE offers a set of standardized technologies for enterprise development. A number of open-source frameworks such as Spring or Hibernate are widely used in a variety of Java applications. Familiarity with new "beyond-Java" languages and technologies will widen your horizons and make you a more valuable Java professional.

— Sponsored by —
No. 1 i-Technology Magazine in the World
JDJ

— Produced by —
SYS-CON EVENTS

For this and other great events visit www.EVENTS.SYS-CON.com

LEARN FROM THE MASTERS:

- Java 6.0 - New Features
- EJB 3, Spring and Hibernate: A Comparative Analysis and Recommendations
- Adobe Flex for Java Developers
- Enterprise Service Bus as a Centerpiece of SOA Implemented with Java
- Code Quality: Pay Now or Pay Later
- EJB 3. and Java Persistence API
- Concurrent Programming in Java
- XML Processing in Java
- Programming with Spring Framework
- AJAX for Java Developers
- Ruby on Rails for Java Developers

tributes and finds all tested methods. For each method it creates combined descriptions about how it's tested and by which test cases. The JavaEmitter is also responsible for creating Test Monitor instructions based on test attributes.

The Java Emitter is also responsible for generating Java sources. The default Java Emitter generates a separate Java file for each TestGroup and does much routine work, such as determining the package, inserting necessary imports, adding copyright blocks in the correct format, and creating the *main(String[] args)* method. Each test case is extracted as a method. The JavaEmitter generates well-formed JavaDoc comments from test attributes.

Tests can be differentiated by extraction type, and each test type can be associated with its own emitter. So one can either introduce a new type or assign an existing type to an emitter.

A typical situation when an alternative emitter is required would be the following: on some Micro Edition Platforms a restriction is placed on available memory and consequently on class file size. For such TCKs the emitter creates one class for each separate test case. The emitter increases test execution time but makes it possible to run tests.

Known Issues

One major issue in the described approach is the difficulty of writing tests in XML, which is very good for machine processing, but not convenient for people. An IDE can't be used without installing specially designed plug-ins. Plug-ins help, but they don't allow the use of an IDE's many useful features.

But this problem can be solved! Metadata can be specified with annotations. Java SE 6 provides an excellent and powerful mechanism for processing Java sources: the Compiler API and Annotation Processing (JSR 199 and JSR 269). This processing makes it possible to keep attributes in Java and generate XML for further processing from Java. Such an approach will make test development easier and preserve existing functionality: selecting applicable tests and customizing generation (see Listing 3).

Conclusion

Disposable data that can be auto-completed increases product quality and makes product support easier. Metadata lets code be shared between products and decorates that code dynamically. XML is a good language for data representation, but it's more effective for the intermediate level of processing. Using Java Annotations in combination with the Compiler API and Annotation Processing can give you remarkable results.

Meta-programming is power.

Listing 1: Example fragment of a TCK test

```
 /*       Copyright Block
 * Copyright 2006 Sun Microsystems, Inc. All rights reserved.
 * SUN PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *     Description of Test Group
 * Applet constructor tests, Testing Applet creation
 */
package javasoft.sqe.tests.api.java.applet.Applet;

public class CtorTests extends ReadTest {  Class of Test Group
    /**
     /**     Description of Test Case
     * Assertion testing for public Applet(),
     * Constructs Applet instance.
     */
    public Status Applet001() { Test Case #1
        Applet a = new Applet();
        if (a.isVisible() == true && a.isEnabled() == true
                && a.isValid() == false) { // Check result
            return Status.passed("OKAY");
        } else {
            return Status.failed("Incorrect Applet object created");
        }
    }
    public Status Applet002() { Test Case #2
...
```

Listing 2: Example fragment of a test in XML format

```
<TestGroup ID="Ctor">
      attributes of Test Group
<Description>Testing Applet creation</Description>
    <Package>java.applet</Package>
    <Class>Applet</Class>
    <Method>public Applet()</Method>
    <TestCase ID="Applet001"> attributes of Test Case
         <Assertion>Constructs Applet instance</Assertion>
         <TestCode><![CDATA[  Java source code of Test Case
    Applet a = new Applet();
    if (a.isVisible() == true && a.isEnabled() == true
            && a.isValid() == false) { // Check result
        return Status.passed("OKAY");
    } else {
        return Status.failed("Incorrect Applet object created");
    }]]></TestCode>
    <TestCase ID="Applet002"> Next Test Case
    ...
</TestGroup>
```

Listing 3: Example fragment of an annotated test source with meta-info

```
 package javasoft.sqe.tests.api.java.applet.Applet;

@TestGroup(
  testedClass=java.applet.Applet.class,
  testedMethod="public Applet()"
)
public class CtorTests extends ReadTest {
   /**
    * Assertion testing  for public Applet().
    * Constructs Applet instance.
    */
    @TestCase
    public Status Applet001() {
        Applet a = new Applet();
        if (a.isVisible() == true && a.isEnabled() == true &&
                a.isValid() == false) { // Check result
            return Status.passed("OKAY");
        } else {
            return Status.failed("Incorrect Applet object created");
        }
    }
```

Web 2.0 and Rich Internet Apps

CALL FOR PAPERS NOW OPEN!

September 24-26, 2007

Santa Clara Convention Center
Hyatt Regency Silicon Valley
Santa Clara, CA

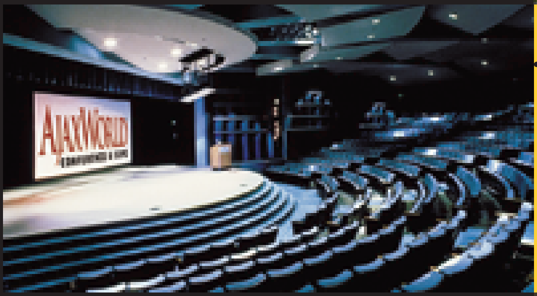
To Register
Call 201-802-3020 or
Visit www.AJAXWorld.com

Hurry! Limited Seating
This Conference Will Sell-Out!

Hyatt Regency Silicon Valley
Santa Clara, CA

Providing developers and IT managers alike with comprehensive information and insight into the biggest paradigm shift in website design, development, and deployment since the invention of the World Wide Web itself a decade ago.

On September 24-26, 2007 over 1,000 developers, architects, IT managers, and software professionals of every stripe will be converging in Santa Clara, CA to attend the West coast AJAXWorld Conference & Expo -- the most comprehensive meeting on the most significant technology subjects of recent times: AJAX, Rich Internet Apps & Web 2.0.



Join Over 3,600 Early AJAX Adopters Who Attended the #1 Web 2.0 Event in the World!

Experience AJAX, RIA, and Web 2.0 Knowledge and Best Practices at AJAXWorld Conference and Expo 2007

Delegates will hear first-hand from the creators, innovators, leaders and early adopters of AJAX, and a slew of leading vendors will provide sneak-peeks of the very latest frameworks, tools, products and applications during the conference, which will have over 100 sessions and presentations given by 125 different speakers - the largest AJAX-focused speaker faculty ever assembled in one place at one time.

AJAXWorld Conference & Expo 2007 West will be jump-started with a full one-day "AJAX University Boot Camp," followed by the Main Conference and Expo over the following 2 days.

Highlights Include:

- ***100+ conference speakers** selected from among the world's leading AJAX technologists and practitioners, including high-level IT executives, industry thought leaders, VCs and analysts*
- ***Topical, Interactive "Power Panels"** simulcast on SYS-CON.TV*
- ***Demos from 30+ vendors** both in General Session and in the Exhibit Hall/AJAX Showcase*
- ***Pre-Conference "AJAXWorld University Bootcamp"** for those wanting an all-day, hands-on encounter with AJAX*
- ***Leading-edge sessions** on issues like Offline AJAX, Mobile AJAX, Enterprise AJAX, AJAX Security, Desktop AJAX, Semantic Mash-Ups, Next-Generation SaaS, JavaScript & XML, Google Web Toolkit, Adobe Apollo, Adobe Flex, AJAX RIA GUIs, and Appropriate vs Inappropriate Use of AJAX.*

Early Bird Price: $1,495
(Register Before May 25, 2007)

SYS-CON EVENTS
For more great events visit www.EVENTS.SYS-CON.com

VISIT WWW.AJAXWORLD.COM FOR THE MOST COMPLETE UP-TO-DATE INFORMATION

JCP's Annual Awards Winners

JSRs: The new, the newer, the newest

Onno Kluyt

In the May column I introduced the candidates nominated by the community for the top 2007 JCP Awards– *the JCP Program 5th Annual Awards*. For those of you who missed the grand finale at the Community Event organized by the JCP at JavaOne, here's the line-up of winners.

JCP Member of the Year: Apache Software Foundation

The Apache Software Foundation (ASF) won the JCP Member of the Year award in recognition of its contributions in 2007 and its active participation in the JCP community overall. ASF representatives have served on numerous Expert Groups and helped implement JCP specifications through, for example, Apache Tomcat, Apache Geronimo, Apache MyFaces, Web services, and portlets projects. The award was accepted by Geir Magnusson Jr., director of the ASF, and chairman/founder of the Apache Geronimo project. He represents the ASF on the Java Standard Edition/Enterprise Edition (SE/EE) Executive Committee (EC). The runners up in this category were Nokia and Orange France.

JCP Participant of the Year: Wayne Carr

This year's Annual Awards premiered a new category, JCP Participant of the Year, the winner of which was Wayne Carr, architect in Intel's SSG/Enterprise Software Solutions Division. Wayne has coordinated Intel's JCP participation since 2002. In that time Intel has participated in 22 JSRs, with Wayne representing Intel on JSR 250, Common Annotations for the Java Platform; JSR 270, Java SE 6 Release Contents; and the early stages of JSR 277, Java Module System. He has served on the Java SE/EE EC for the past two and a half years, and before that he served a year on the Java Micro Edition (ME) EC. In the Executive Committees, Wayne has focused on making the JCP program more transparent and open, on promoting open source, and on ensuring fairness for independent implementations. The runners up in this category were Jean-Marie Dautelle and Doug Lea.

Onno Kluyt is director of the Community Growth group at Sun Microsystems and the Chair of the JCP.

onno@jcp.org

Most Outstanding Spec Lead for Java SE/EE: Nasir Kahn

In this category the votes went to BEA Systems' Nasir Khan, architect of the WebLogic SIP Server based on the Internet standard called Session Initiation Protocol (SIP). A member of the JCP community since 2000, Nasir moved up through the ranks, starting as a member, then working in the BEA-led Expert Group for JSR 309, Media Server Control API, and becoming a Spec Lead. In 2007, he was also recognized as a Star Spec Lead for his work on JSR 289, SIP Servlet v1.1. The runners up for this award were Alan Bateman and David Nuescheler.

Most Innovative JSR for Java SE/EE: JSR 308

The votes cast in this category designated JSR 308, Annotations on Java Types, as the winner. The Spec Leads Danny Coward of Sun Microsystems and Michael Ernst along with the supporting JSR Expert Group are working on a standard that enriches the Java annotation system. It will permit annotations to appear not just on declarations, but on any use of a type, such as generic type arguments, typecasts, and method receivers. The JSR 308 Expert Group is considering other extensions, such as permitting an annotation to be specified multiple times at a single location, or permitting annotations on statements as well as on whole methods. The JSR 308 changes will make the Java annotation system more expressive and hence more useful, for example, by making bug-finding tools and annotation processors more effective. Another innovative aspect of JSR 308 is that it will permit programmers to customize the Java type system to provide extra guarantees, while retaining backward compatibility. This gives a guarantee about any program that uses the annotations. For more details about JSR 308, see http://pag.csail.mit.edu/jsr308/. The other JSRs in the race for this award were JSR 299, Web Beans, and JSR 309, Media Server Control API.

Most Outstanding Spec Lead for Java ME: Mike Milikich

Mike Milikich, the Java ME Technology Development Manager for Motorola Mobile Devices, won the top award in this category. Since 2001, when he began participating in the JCP program, Mike has served in various ways, including representing Motorola on the Java ME EC and contributing to eight JSRs: 37, 118, 180, 248, 249, 253, 271, and 307. He is the Star Spec Lead for JSR 271, Mobile Information Device Profile 3, whose Expert Group encompasses over 120 participants, and he is co-leading JSR 307, Network Mobility and Mobile Data API. The runners up in this award category were Shai Gotlib (JSR 190) and Antti Rantalahti and Ivan Wong (JSR 272).

Most Innovative JSR for Java ME: JSR 307

Mike Milikich was called twice to the winners' podium at the JCP Program 5th Annual Awards gala at JavaOne. The second time to accept with Eric Overtoom, his co-Spec Lead, the award for JSR 307, Network Mobility and Mobile Data API. This JSR aims to provide better control

–continued on page 34

JAVA DEVELOPER'S JOURNAL Advertiser Index

Advertiser	URL	Phone	Page
AJAXWorld Conference & Expo	www.ajaxworld.com	201-802-3022	31
Altova	www.altova.com	978-816-1600	Cover II
Business Objects	www.businessobjects.com/rareoccurrence	888-333-6007	11
DiSTI	www.simulation.com	407-206-3390	19
IBM	www.ibm.com/takebackcontrol/innovate		8-9
Infragistics	www.infragistics.com/jsf	800-231-8588	7
InterSystems	www.intersystems.com/ja2p		Cover IV
Java Developer's Journal	www.jdj.sys-con.com	888-303-5282	33
JadeLiquid	www.webrenderer.com		17
Jinfonet Software	www.jinfonet.com/fd	240-477-1000	4
Northwoods Software Corp.	www.nwoods.com	800-434-9820	23
Real-World Java Seminar	www.realworldjava.com	201-802-3020	29
Software FX	www.softwarefx.com	561-999-8888	Cover III

–continued from page 32

over how network connections and data sessions are established for applications. This aspect is becoming increasingly important as devices are being introduced that support multiple ways of communicating with the world, over cellular data services, WiFi, WiMax, Bluetooth – each having different characteristics and capabilities of routing data to particular destinations. This JSR also sets out to provide the means for applications to select particular connections, such as a game that expects to use a particular cellular connection configuration, especially a configuration that is not the default for the device. To see the additional new capabilities this JSR plans to introduce, visit the JSR public page at http://jcp.org/en/jsr/detail?id=307. The runner-up JSRs in this award category were JSR 298, Telematics API for Java ME, and JSR 300, DRM API for Java ME.

Join me in congratulating both winners and runners up. Bookmark these JSRs should you want to follow their progress and get in touch with the Spec Leads and Expert Groups to provide input and feedback.

JSRs: The New, the Newer, the Newest

Now for your summer Java standards blotter, a few of the most recently submitted JSRs on which you may want to keep an eye. Some of them, you'll recognize, were among the candidates for the top JCP awards.

JSR 315, Java Servlet Specification 3.0

This JSR was submitted by Sun Microsystems in June. At the time of writing the JCP EC vote is expected to rule on it by July 2. Early reviews of the proposal in several Java developer outlets note that there are lots of interesting bits in it including the ability to programmatically log in and out, configuration through annotations, and asynchronous communications. Check it out at http://jcp.org/en/jsr/detail?id=315.

JSR 314: JavaServer Faces 2.0

Kicked off in May, JavaServer Faces 2.0, JSR 314, is in Expert Group formation mode at the time of writing. Go to its public page at http://jcp.org/en/jsr/detail?id=314 to check what it holds in its development road map and apply for joining the Expert Group should you want to contribute to what the Spec Leads characterize as a "major revision of the JavaServer Faces specification." This new work on JavaServer Faces technology occurs at an interesting time when the form that many of the AJAX runtime frameworks have chosen to take in the Java EE world is JavaServer Faces technology based, so say some of the Java technology watchers.

JSR 312: Java Business Integration 2.0 (JBI 2.0)

Started in March 2007 this new version of JBI – JBI 2.0 – is intended to "augment the standard to address new requirements." The Expert Group of this JSR is also in formation, an opportunity for anyone with an interest to apply to join the Expert Group and be part of the JSR development. The goal of the JBI 2.0 Expert Group will be to investigate, identify, and pursue direction through which the JBI architecture can be kept simple but new component (bindings and services) functionality enhanced for various usage audiences. This specification sets out to enhance the capabilities of Java Business Integration, allowing developers to provide more sophisticated applications and achieve better integration with other Java platform technologies.

JSR 311: JAX-RS, Java API for RESTful Web Services

This JSR was filed relatively recently at the end of February, was approved and is in Expert Group formation mode at the time of writing. Its goal and road map were presented at a BOF at JavaOne – to provide a high-level declarative programming model for RESTful services that is easy to use and encourages development according to REST tenets. According to the Spec Leads the intent is to make services built with the API deployable by use of a variety of Web container technologies and benefit from built-in support for a variety of HTTP usage patterns and conventions. Check out this JSR's public page for details regarding how the Spec Leads and Expert Group plan to address the anticipated needs of the next-generation Web.

JSR 310: Date and Time API – Start 13 February 2007

A few months young, this JSR sets off to address a few issues with the Java Date and Time API: the fact that currently Java SE has two separate date and time APIs – java.util. Date and java.util.Calendar – and they are described as difficult to use by Java developers on Weblogs and forums. At the end of the development process the Spec Leads and Expert Group hope to deliver a comprehensive date and time model including dates and times (with and without time zones), durations and time periods, intervals, formatting, and parsing.

JSR 309: Media Server Control API

In early February BEA and HP filed the Media Server Control API, JSR 309. Of interest to the telecom industry, this specification is planned as a protocol-agnostic API for Media Server Control and will be designed to provide the developer community with an API that standardizes access to external media server resources from services built on Java application servers. Also in its Expert Group formation mode, the JSR has signed up a few members of the telecommunications community, media server companies, application companies, network equipment providers, and service providers. For a complete list of supporters and to join the Expert Group, visit http://www.jcp.org/en/jsr/detail?id=309.

Stay tuned for the next column, the summer will have its JSR news no doubt and more.

"Stay tuned for the next column, the summer will have its JSR news no doubt and more"

TITLE UNIFICATION FOR ECLIPSE & NETBEANS INTEGRATION

☆SHOWDOWN☆

JOE "CHART FX" CODER

VS

DATA VISUALIZATION

A FULL 6.2 ROUNDS OF CHAMPIONSHIP HEAVYWEIGHT DATA VISUALIZATION

Chart FX

Undisputed Champion

BOX OFFICE
www.chartfx.com

A KNOCKOUT COMBINATION OF CHARTS, GAUGES & MAPS FOR JAVA

BROUGHT TO YOU BY
SoftwareFX

Call (561) 999-8888 for more information or download a trial version at www.softwarefx.com

Innovations by InterSystems

Make Java Applications More Valuable

For Java developers, Caché offers a wealth of benefits.

When you embed Caché in your applications, they become more valuable. Caché dramatically improves speed and scalability while decreasing hardware and administration requirements. This innovative object database runs SQL queries faster than relational databases. And with InterSystems' JALAPEÑO™ technology for Java developers, Caché eliminates object-relational mapping. Which means Caché doesn't just speed up the *performance* of applications, it also accelerates their development. Caché is available for Unix, Linux, Windows, Mac OS X, and OpenVMS – and it is deployed in more than 100,000 systems ranging from two to over 50,000 users. Embed our innovations, enrich your applications.

InterSystems CACHÉ®

Download a free, fully functional, no-time-limit copy of Caché, or request it on CD, at **InterSystems.com/Ja2P**