//SEPTEMBER/OCTOBER 2013 /

Java™ magazine

By and for the Java community

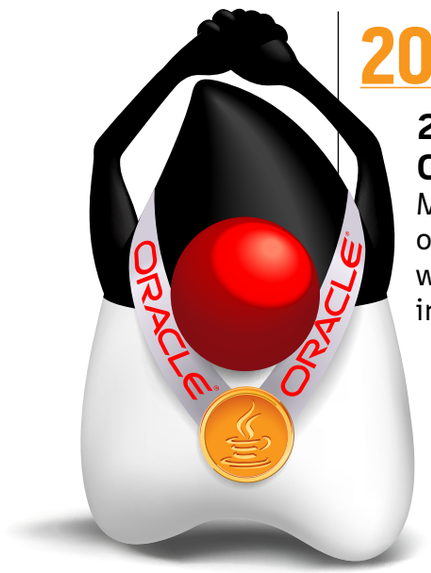
Seize the Cloud

Develop, deploy, and manage your applications in the cloud with Java EE and Oracle Java Cloud Service

ORACLE.COM/JAVAMAGAZINE

ORACLE

//table of contents /



COVER ART BY I-HUA CHEN; PHOTOGRAPHY BY BOB ADLER

EDITORIAL

Editor in Chief
Caroline Kvitka

Community Editors
Cassandra Clark, Sonya Barry, Yolande Poirier

Java in Action Editor
Michelle Kovac

Technology Editors
Janice Heiss, Tori Wieldt

Contributing Writer
Kevin Farnham

Contributing Editors
Claire Breen, Blair Campbell, Karen Perkins

DESIGN

Senior Creative Director
Francisco G Delgadillo

Senior Design Director
Suemi Lam

Design Director
Richard Merchán

Contributing Designers
Jaime Ferrand, Nicholas Pavkovic, Arianna Pucherelli

Production Designers
Sheila Brennan, Kathy Cygnarowicz

PUBLISHING

Vice President
Jeff Spicer

Publisher
Jennifer Hamilton +1.650.506.3794

Audience Development and Operations Director
Karin Kinnear +1.650.506.1985

ADVERTISING SALES

Associate Publisher
Kyle Walkenhorst +1.323.340.8585

Northwest and Central U.S.
Tom Cometa +1.510.339.2403

Southwest U.S. and LAD
Shaun Mehr +1.949.923.1660

Northeast U.S. and EMEA/APAC
Mark Makinney +1.805.709.4745

Advertising Sales Assistant
Cindy Elhaj +1.626.396.9400 x 201

Mailing-List Rentals
Contact your sales representative.

RESOURCES

Oracle Products
+1.800.367.8674 (U.S./Canada)

Oracle Services
+1.888.283.0591 (U.S.)

Oracle Press Books
oraclepressbooks.com

ARTICLE SUBMISSION

If you are interested in submitting an article, please e-mail the editors.

SUBSCRIPTION INFORMATION

Subscriptions are complimentary for qualified individuals who complete the subscription form.

MAGAZINE CUSTOMER SERVICE

java@halldata.com **Phone** +1.847.763.9635

PRIVACY

Oracle Publishing allows sharing of its mailing list with selected third parties. If you prefer that your mailing address or e-mail address not be included in this program, contact Customer Service.

Java Magazine is published bimonthly with a free subscription price by Oracle, 500 Oracle Parkway, MS OPL-3C, Redwood City, CA 94065-1600.

Digital Publishing by Texterity

The IT forecast is for clouds. Public clouds and private clouds are here to stay. In this issue, we look at what the cloud means for Java developers and give you the tools to seize the opportunity that cloud presents.

First, Oracle's Cameron Purdy weighs in on why developers need to change the way they think, in the interview "Cloud: Challenge and Opportunity." Applications must be built so that they can dynamically scale out and dynamically balance the load across multiple servers, he says, adding that developers must also explicitly design the ability for applications to scale as close to linearly as possible. Purdy also discusses why developers should pay attention to platform as a service. Plus: developer, Java Champion, and Netherlands JUG leader Bert Ertman tells us how he is using the cloud today.

Next, Java Champion Harshad Oak helps you get started with "Hands On with Oracle Java Cloud Service." While it was once unheard of to suggest that a Java EE application could run in a shared environment, says Oak, it is now actually fashionable to do so.

We also bring you the 11 winners of the Duke's Choice Awards (now in its 11th year). This year's winners of the award, which honors innovation in Java development, range from the serious and cerebral to the practical, amusing, and entertaining. The winners are pushing the frontiers of technology by simulating the human brain and musculoskeletal system; providing guidance to cars, satellites, and robotic fish; training tomorrow's Java programmers; making Java applications more secure; and building communities. We are honored to recognize them here.

How are you using the cloud? How are you innovating with Java? Let us know.

Caroline Kvitka, Editor in Chief BIO

//send us your feedback /

We'll review all suggestions for future improvements. Depending on volume, some messages may not get a direct reply.

PHOTOGRAPH BY BOB ADLER

CMS or Portal? Choosing the Right Technology

dotCMS CTO Will Ezell takes a look at trends driving change for content management

Will Ezell, CTO, dotCMS

Q› Why do companies have trouble deciding between these systems?

There's a fair amount of overlap between a CMS and a portal—many portal tools contain rudimentary content management tools.

But while companies may be able to use a portal in place of a CMS system, getting it to look and act the way you like might be difficult. You might end up trying to hammer a square peg into a round hole.

How so?

Companies often choose to use portals for their document libraries or content-rich intranets. The issue is that portals are really just frameworks intended to provide an aggregated view into multiple enterprise applications based on a user's profile. Think of the old My Yahoo pages, which contained a lot of customizable, lightweight widgets—it turns out that the widgets were too lightweight and users inevitably would leave for the fuller experience of the native sites.

The real problem is that today's business users expect any site—web, mobile, even intranets—to look and interact in ways that portals weren't designed for.

So it's a question of function not meeting need?

Portals are good at what they are designed for, but can be cumbersome to build in. They require your sites to function in a specific way, which makes it hard on your users. And these days, users are used to dealing with websites and apps that are flexible, intuitive, and very easy to use. Turns out, your workforce wants their work-related web experiences to be just as simple and easy as their consumer experiences.

So CMS systems like dotCMS are built to address ease of use?

Absolutely. If your team has ever wrestled with the look, feel or interactivity of your portal-driven site or intranet, you should consider dotCMS. DotCMS is designed from the ground up to manage and seamlessly deliver role-based content, sites, mobile sites, intranets and applications without impacting interactivity or imposing rigid look-and-feel requirements. It moves content and user experience to the center of the equation, where it belongs. We basically provide the development tools and get out of the way.

How do Java developers react to working with dotCMS?

When J2EE developers hear from a business unit, "I need ABC web site that does XYZ," they instinctively reach for a portal framework.

It turns out that a CMS might offer an easier, more flexible platform for delivering such sites and apps. DotCMS is based on familiar open standards such as OSGi, CMIS, Spring, Struts, Hibernate, Velocity and Elasticsearch, which also means that developers can take our code base and customize it to do whatever they need.

Our UI is very straightforward and easy to comprehend; we tried to make dotCMS as approachable and simple as possible. Plus, our use of loosely coupled RESTful APIs means that you can read/write content and apps to any 3rd-party web-based system, be it php, .NET or J2EE. It's just much easier to respond to changes and manage any given site or content.

And this may be the best part: it's a fraction of the price of most portal solutions in this space.

For more information visit **dotCMS.com**

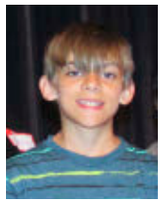
dotCMS

Top: Teens get hands-on practice using Alice at the Make the Future Java Summer Workshop. Bottom: David Culyba of Carnegie Mellon University describes the programming process.

PHOTOGRAPHS BY DON SLATER

JAVA'S FUTURE IS BRIGHT

1,000 San Francisco Bay Area teenagers learned how to program in Java at the Make the Future Java Summer Workshop, held July 30–August 1 at Oracle headquarters in Redwood Shores, California. Oracle Academy staff members and Carnegie Mellon University (CMU) computer science professors helped participants learn Java using the award-winning Alice visual 3-D educational tool, which uses a drag-and-drop interface to create animations. On August 1, students attended a Minecraft workshop, where they learned how to extend the popular online game Minecraft by using Java to write modifications.



Nick Rocha, a 13-year-old programmer from San Antonio, Texas, presented a session to all attendees called Inspired By Alice. Rocha, who has a passion for computer-based and video games, learned to program a year ago through a Learning to Program with Alice course at Homeschool Arts & Academics Class Day. He was instantly hooked on Alice. "If you like games, 3-D animation, and being creative, I bet you'll like Alice," Rocha told the audience. He demoed several of the animations and games that he has built using Alice. "Alice can be tough," he explained, "but it's both fun and rewarding."

Attendees also heard from **David Culyba**, one of the primary Alice developers at CMU in Pittsburgh, Pennsylvania. Culyba, who worked on Alice as an undergraduate at CMU, worked in the industry as a game developer before joining the Alice team, of which Oracle is a long-time supporter. Culyba focuses on how to make programming less frustrating and more fun. When the audience was asked if he had succeeded, based on their day's experience

Nick Rocha talks about his experiences using Alice.

with Alice, the response was an enthusiastic "yes!"

"Programming is a problem-solving process," he said. "Think about the problem that you are trying to solve, and then think about how to get a computer to solve that."

Workshop participants included 24 students from the Taleem Initiative, a community-based program that promotes higher education among students who aspire to be the first in their families to attend college. "The Future of Java workshop introduced our students to a world of new opportunities," said Dr. Arvinderpal S. Wander, principal engineer at Oracle Labs and cofounder of the Taleem Initiative.

Workshop participants were rewarded for their participation with T-shirts, gift cards, thumb drives containing software, and plenty of Java jelly beans.

Learn more at the Make the Future Java learning resource center.

THE DEVELOPER CONFERENCE 2013

The Developer Conference (TDC) 2013 was held in São Paulo, Brazil, July 10–14. More than 3,200 developers attended this five-day multicommunity developer conference, while another 3,000 developers watched the live streaming sessions.

"Developers from different communities come to TDC to learn about new development methodologies and new technologies," said **Yara Senger** (pictured, top left) of conference organizer Globalcode. Tracks included Java, service-oriented architecture (SOA), mobile, Ruby, Python, agile, open source embedded, big data, and digital TV.

TDC was full of surprises, creativity, and fun. For example, in a two-day Raspberry Pi challenge sponsored by Oracle, four teams of developers built a heart rate monitoring application, facial recognition software, an automated pet feeder, and a music jukebox application. One of the challenge participants, **Inacio Junior**, performed calisthenics on stage to animate a throbbing raspberry fruit graphic and heart rate graph in a JavaFX interface. In addition, Globalcode's **Vinicius Senger** created a voice-triggered popcorn machine using a Raspberry Pi, an Arduino, and sensors. He also developed an Oracle Java Embedded Suite application to control the booth lights and tracking room temperature using a Raspberry Pi, an Arduino, and sensors.

Inacio Junior talks about the heart rate application his team created.

PHOTOGRAPHS COURTESY OF TDC

JAVAONE SHANGHAI

JavaOne Shanghai was held July 22–25 at the Shanghai Expo Center. Here are a few highlights:

Geeks ride. Now a tradition around the world, the Geek Bike Ride preceded JavaOne. In matching bike jerseys (pictured above), more than 50 riders enjoyed a ride near the Huangpu River. In deference to more than 90-degree heat and high humidity, the group stopped for refreshments halfway through the ride. It was great fun and camaraderie.

Java user group (JUG) meeting. On Sunday, the Green Tea JUG took advantage of the speakers coming to China for JavaOne. More than 150 developers heard speakers from Oracle, IBM, and Alibaba on topics ranging from Java EE new features to OpenJDK best practices. JUG leader **Leo Yu** said, "It was a great event!"

Embedded discussion. A panel discussion on "Embedded to the Enterprise" was held at the Asia Pacific Oracle User Group Summit. The discussion focused on the instrumental role of Java EE for the Internet of Things and the cloud going forward.

Java EE 7. Just released, Java EE 7 was a hot topic at the technical keynote. Java EE 7 content included a hands-on lab and sessions on the platform, WebSocket, and batch processing. An Oracle Java Cloud Service hands-on lab was also offered.

Alex Mironenko shows demos of embedded Java in action.

Simon Ritter goes head to head with Tori Wieldt at JavaOne Shanghai.

PHOTOGRAPHS BY STEPHEN CHIN AND HARTMANN STUDIOS

Meet Scientist Duke

Each year, Oracle releases a personality for Duke, the official mascot of Java technology. The latest is Scientist Duke. Much like a genius developer would with Java technologies, Scientist Duke is engaged in developing, mixing, and testing the latest ingredients to create the greatest formula/product/innovation ever. Last year, Adventure Duke was off on a mission to find the latest Java wonders around the world. 2011 unveiled Future Tech Duke, and 2010 brought us Surfing Duke.

Oracle Java ME Embedded Update

Oracle has released Oracle Java ME Embedded 3.3 and Java ME Software Development Kit (SDK) 3.3, a complete client Java runtime and toolkit optimized for microcontrollers and other resource-constrained devices. This release includes improvements of interest to developers, including ways to not have to build so much "core plumbing" for an app, and more information about memory and network usage, which can be critical for low-power apps.

Oracle is also introducing the Oracle Java Platform Integrator program to provide partners with the ability to customize Oracle Java ME Embedded products to reach different device types and market segments.

JAVA CHAMPION PROFILE

DR. MATJAZ B. JURIC

Dr. Matjaz B. Juric *is head of the Cloud Computing Center and the SOA Competence Center and a professor at the University of Ljubljana in Slovenia. He was named a Java Champion in May 2010.*

Java Magazine: Where did you grow up?

Juric: I grew up in Ptuj, a beautiful town with an old castle and a historic center with narrow streets and old buildings.

Java Magazine: How did you first become interested in computers and programming?

Juric: When I was around 10 years old, home computers became available in our country, and the first magazines on computing started to appear. I was immediately overwhelmed, and since then my professional life has been devoted to computer science.

Java Magazine: What was your first computer and programming language?

Juric: A Sinclair ZX Spectrum with 48 KB of memory, with BASIC preloaded. A year later, I switched to Commodore 64.

Java Magazine: What was your first programming job?

Juric: I developed an accounting application on a PC. It grew into a substantial and successful project: some companies still use it today.

Java Magazine: What do you enjoy for fun?

Juric: In the winter, I love snowboarding and skiing. In the summer, I'm a passionate kite surfer and scuba diver.

Java Magazine: What happens on your typical day off?

Juric: I'm the father of a lovely two-month-old daughter. I spend all my free time with my family. It's amazing to raise a child.

Java Magazine: What "side effects" of your career do you most enjoy?

Juric: First, attending conferences and meeting all the passionate developers, talking with them, and generating new ideas. Second, traveling: visiting new places is relaxing, giving you positive energy and freeing you from routine.

Java Magazine: Has being a Java Champion changed anything for you with respect to your daily life?

Juric: Being a Java Champion is a respected position, which can give you community recognition. Also, this status eases communication with other experts in the field. Last but not least, Java Champions are invited to JavaOne, which is a one-of-a-kind experience.

Java Magazine: What, in your view, is most significant about the recent Java EE 7 release?

Juric: Bringing important enhancements to REST services and JMS [Java Message Service] are two of the most welcome features. However, I'm looking forward to Java EE 8 and the announced cloud extensions.

Java Magazine: What else are you looking forward to?

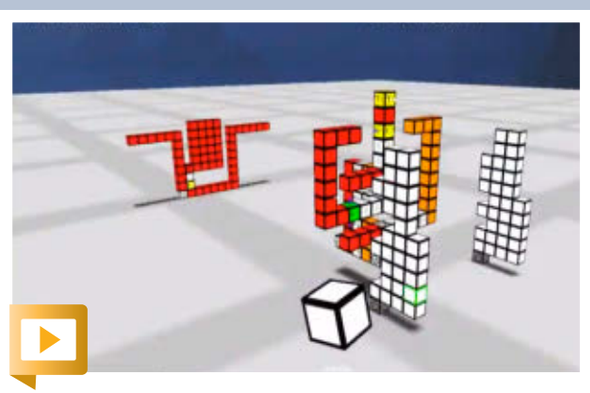
Juric: I think a significant shift in enterprise application development will be required to keep up with the development progress in other areas and to align enterprise applications and systems with new devices, interfaces, and other innovations. It will be a great opportunity for Java to evolve and to keep its leading position on the server side.

jMonkeyEngine Combines Gaming and Community

If you like game development and open source, and you want to participate in an active Java community, check out jMonkeyEngine. It is an all-Java game engine library written especially for game developers who want to create 3-D games using modern technology standards. The core team consists of nine people, led by architect **Kirill Vainer** and community builder **Erlend Sogge Heggen**.

One of the unusual features of the project is the completeness of the software documentation. For example, the recently released Version 3 includes the *jMonkeyEngine 3.0 Beginner's Guide*, a book written by jMonkeyEngine documentation specialist **Ruth Kusterer**.

The jMonkeyEngine community site includes a blog, documentation, forums, a download section, and ways to contact the project leaders. There are many ways to get involved: the jMonkeyEngine Contributor's Handbook describes roles for modelers, web developers, technical writers, programmers, and anyone else who'd like to help.

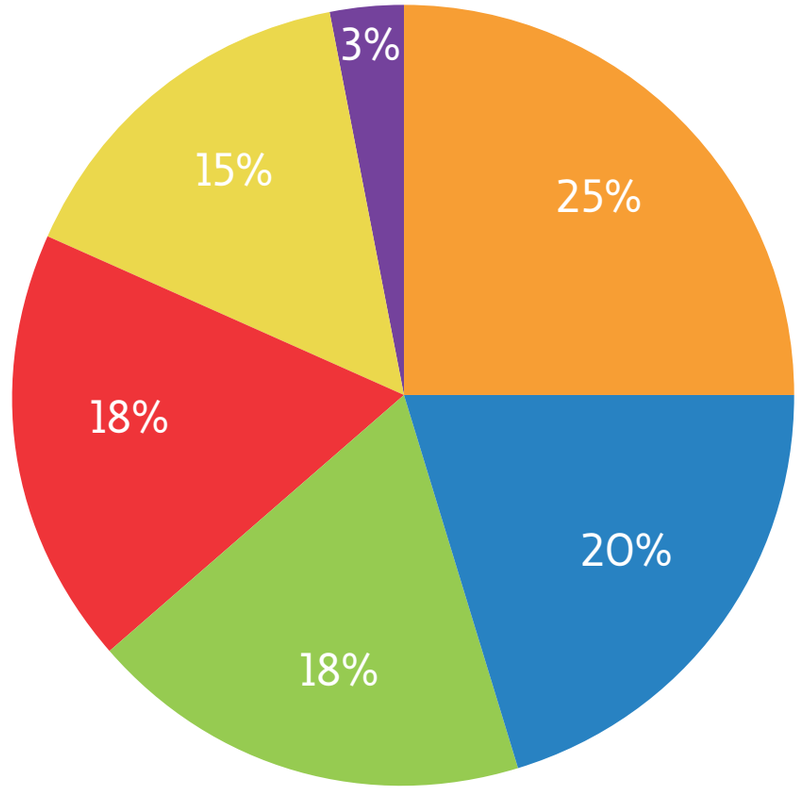


Get a look at what you can do with jMonkeyEngine.

JAVA.NET POLL

THE CLOUD: MORE THAN BUZZ

The results of a recent Java.net poll suggest that developers have widely varying views on the significance of the cloud, ranging from "It changes everything" to "It's just buzz." A total of 177 votes were cast during the two-week survey in response to the question "Does 'the Cloud' change anything?" Here are the final results:



25% It's a pretty significant development.

20% It changes everything: ultimately, it will eliminate most corporate data centers.

18% It's too early to tell.

18% It's not as important a development as many people think.

15% No, it's just buzz.

3% Other

FEATURED JAVA USER GROUP

CHICAGO JAVA USERS GROUP

The Chicago Java Users Group (CJUG) was founded in 1995 by **Philip McGlauchlin**. Today, CJUG is led by **Scott Kramer** (president) and **Freddy Guime** (community leader). "I'm more of the constituency-facing guy, while Scott concentrates more on the business side, getting sponsors, and making sure that CJUG has some money in its coffers for me to spend," says Guime.

The group, which has about 360 active members and thousands of members in total, typically meets twice a month, "but we're really making inroads to having more meetings," Guime says. The agenda for meeting topics is broad, including speakers series that focus on topics such as "Learning About Our Craft," "Making a Difference in the Java Community," and "Growing as a Java Professional."

"As we keep getting more volunteers and funding, we keep going down the list and making Chicago the best place to be a Java developer," Guime says.

Through the CJUG mentorship program, any member can request help or advice on a particular issue. "This program is run by volunteers," Guime notes, "and it's starting to bear fruit." CJUG's outreach doesn't end with its own membership, though. The group works with computer science departments at Loyola, DePaul, University of Illinois at Chicago, and Illinois Institute of Technology to provide guest speakers and to help grow awareness among the students that their city has a community they can rely on for their future professional needs.

CJUG reaches out to potential sponsors in part through its sponsorship program. Publicizing the program on the CJUG site allows potential sponsors to understand the "rules" of sponsorship. Guime notes, "While this might alienate certain sponsors, it also helps us save time—which in our JUG is the most precious resource."

CJUG's focus has been on building a community, and "that's hard," Guime says. "It has required a tremendous amount of time, work, conflict, and commitment. But once the seeds are set, the group starts growing. As we see it, a JUG is as healthy as its leaders. Keep the leadership engaged, and you can create a great community with great results."

African JUGs Embrace Making the Future Java

African Java user groups (JUGs) are taking advantage of Oracle's "Make the Future Java" initiative, organizing events around the Java EE 7 launch and other Java-related technologies. Due to the unique problems many African developers and user groups face with respect to transportation and communications, the tools provided by Oracle (including "Make the Future Java EE 7 Tool Kit," online events, and libraries of on-demand webcasts) are an invaluable resource.

Lamine Ba, general manager of the 5,000-member umbrella group JUG Africa, said, "We find it extremely helpful that Oracle provides online events and webcasts, as well as resources for us to host local events, so our community members can come together to learn about the new features in Java EE 7, as well as socialize and exchange ideas with each other."

EVENTS

Devoxx *NOVEMBER 11–15*
ANTWERP, BELGIUM

Devoxx has been running for the last 12 years and is the largest Java conference in Europe. The five-day conference attracts world-class speakers and developers from 40 different European countries. Its program has a packed schedule with university sessions during the first two days and keynotes, sessions, hands-on labs, quickies, and birds-of-a-feather sessions over two days. Tracks include Java, Android, HTML5, and more.

PHOTOGRAPH BY GETTY IMAGES

Silicon Valley Code Camp
OCTOBER 5–6
LOS ALTOS HILLS, CALIFORNIA
At this free community event, developers learn from each other. In addition to programming training, the conference tackles topics such as software branding and legal issues around software.

CodeMotion
OCTOBER 17–19
MADRID, SPAIN
This conference is open to users of all languages and platforms. It offers full-day workshops during the first day, and keynotes, conference sessions, and hackathons in eight tracks over two additional days.

Jazoon
OCTOBER 22–23
ZURICH, SWITZERLAND
This international conference for the software community offers content on programming, frameworks, tools, platforms, and methods as well as about the professional development process for enterprises, the web, the cloud, mobile devices, and desktop applications.

JAX London
OCTOBER 28–30
LONDON, ENGLAND
JAX is about a new generation of web technologies and cloud-based services; sophisticated architectural decisions for maximum flexibility and scalability; integration of big data technologies; and on the client side, the rise of powerful web apps and mobile post-PC devices.

Oredev 2013
NOVEMBER 4–8
MALMO, SWEDEN
The theme of the conference is "Year of the Arts." Developers, testers, and project managers are invited to reflect on the art of the programming processes, languages, tools, and platforms. The organizers promise sessions and keynotes that are creative, provocative, and innovative.

W-JAX Munich
NOVEMBER 4–8
MUNICH, GERMANY
W-JAX is a leading conference that focuses on the Java platform, web, architecture, agile, and the cloud.

J-Fall
NOVEMBER 6
NIJKERK, THE NETHERLANDS
J-Fall is the annual Java developer conference organized by NLJUG, the Dutch Java user group. More than 40 technical presentations will be offered to the 1,200 expected attendees by local and international Java superstars. The conference marks the tenth anniversary of NLJUG and J-Fall.

Java EE Ambassadors Visit Chinese Java User Groups

Oracle's **Anissa Lam** (project lead for the GlassFish Administration Console) and **Shing Wai Chan** (spec lead for JSR 340, Java Servlet 3.1, and implementation lead for the GlassFish web container) recently toured China, visiting Java user groups (JUGs) to promote Java EE 7 and GlassFish technologies.

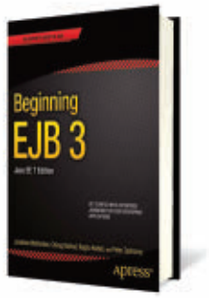
Just before JavaOne Shanghai, the Shanghai JUG and the Green Tea JUG (from Hangzhou) co-organized an event that included presentations by Lam and Chan.

Following JavaOne Shanghai, Lam and Chan visited the just-formed Nanjing JUG. Many of its founding members work at Fujitsu Nanjing, which is a great supporter of Java EE and the GlassFish project. Nanjing JUG members have worked with Lam and other members of the GlassFish team for years.

During the trip, Lam and Chan also visited the Guangdong JUG; and at JavaOne Shanghai, they had the opportunity to meet with members of the Duke's Choice Award–winning OSChina team.

Anissa Lam discusses her activities at JavaOne Shanghai.

JAVA BOOKS



BEGINNING EJB 3, JAVA EE 7 EDITION
By Jonathan Wetherbee, Chirag Rathod, Raghu Kodali, and Peter Zadrozny
Apress (May 2013)
Develop powerful, standards-based, back-end business logic with *Beginning EJB 3, Java EE 7 Edition*. Written by an author team with 20 years of combined Enterprise JavaBeans (EJB) experience, this book teaches you how to use the new EJB 3.2 APIs. You'll gain the knowledge and skills you need to create the complex enterprise applications that run today's transactions and more. This book is packed with practical insights, strategy tips, and code examples.

JAVAMAIL API, SENDING AND RECEIVING EMAIL WITH JAVA
By Elliotte Rusty Harold
O'Reilly (July 2013)
Send and receive e-mail from Java applications using the JavaMail API. With this concise book, you'll learn how to communicate with existing SMTP, POP, and IMAP servers, and how to write your own. Whether you need to build an e-mail–centric application such as a mailing list manager or simply add e-mail notification to a larger product, JavaMail is the answer. Packed with code examples, this book shows you how JavaMail enables you to avoid low-level protocol details, so you can focus on what you actually want to say in a message.

JAVA CODING GUIDELINES, 75 RECOMMENDATIONS FOR RELIABLE AND SECURE PROGRAMS
By Fred Long, Dhruv Mohindra, Robert Seacord, Dean Sutherland, David Svoboda
Informit (August 2013)
Reflecting pioneering research on Java security, *Java Coding Guidelines: 75 Recommendations for Reliable and Secure Programs* offers updated techniques for protecting against both deliberate attacks and other unexpected events. You'll find best practices for improving code reliability and clarity, and a full chapter exposing common misunderstandings that lead to suboptimal code.

Oracle Publishers Program

Oracle Publishers Program works with participating independent publishers to support the creation of high-quality technology books and related materials that focus on Oracle technologies. Independ-ently published technology books are valuable in many ways, including raising awareness of Oracle products; making Oracle technology approachable for beginners; helping our customers be more successful; increasing product adoption; and generating buzz for new products and technologies. Oracle believes that publishers and authors are critical evangelists for Oracle, and that it is important to support the publication of Oracle technology books for the retail marketplace.

The Oracle Publishers Program provides support to member publishers by providing access, whenever possible, to product information, early product releases, and Oracle experts, thus helping improve time to market for Oracle technology titles.

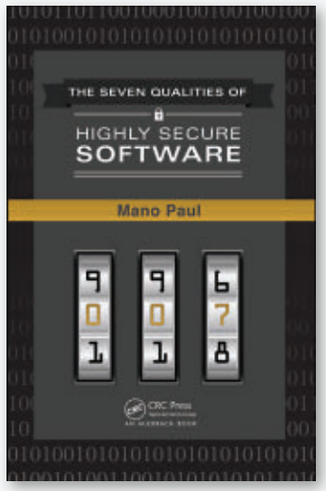
Publishers Program members and their nominated authors participate in our annual Oracle Publishers Seminar held in San Francisco and have the opportunity to attend Oracle's annual technology conferences—Oracle OpenWorld and JavaOne—which run at the same time as the seminar.

Book Promotions and Discount Channels

- Oracle Author Podcast Series
- *Oracle Magazine* Book Beat
- Java Nation
- Oracle Technology Network Member Discount
- Oracle Technology Network Member New Offers
- Oracle ACE Newsletter
- Oracle OpenWorld and JavaOne bookstore exposure and author signing sessions

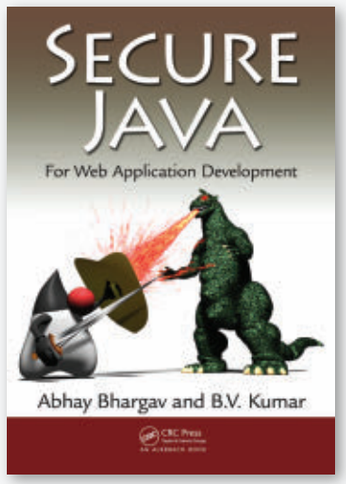
SAVE 25%

Save on the Books You Need to Design, Develop, and Deploy Secure Software



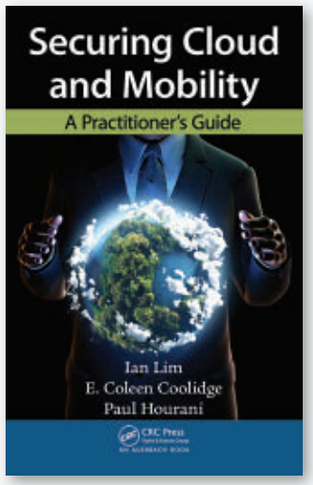
The Seven Qualities of Highly Secure Software

ISBN: 978-1-4398-1446-8, 160 pp.
$49.95 / £31.99



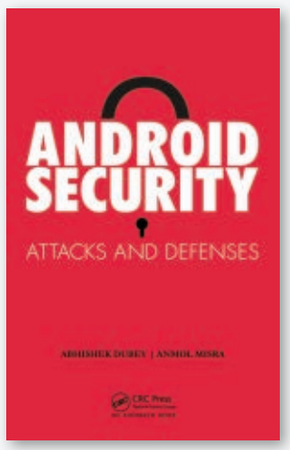
Secure Java For Web Application Development

ISBN: 978-1-4398-2351-4, 308 pp.
$76.95 / £48.99



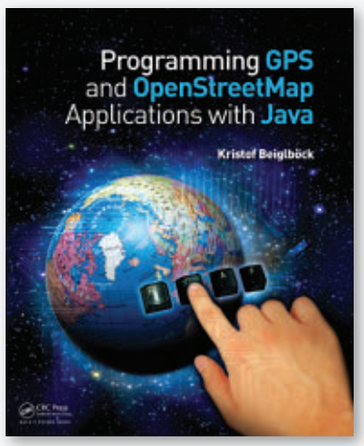
Securing Cloud and Mobility

ISBN: 978143985055-8, 228 pp.
$79.95 / £49.99



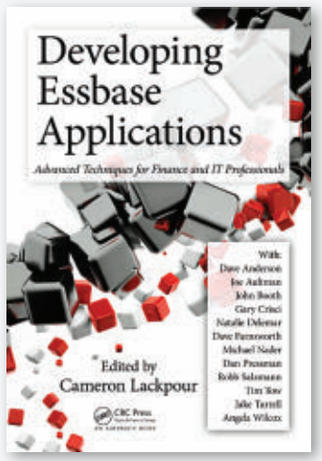
Android Security

ISBN: 978-1-4398-9646-4, 280 pp.
$59.95 / £38.99



Programming GPS and OpenStreetMap Applications with Java

ISBN: 978-1-4665-0718-0, 248 pp.
$59.95 / £38.99



Developing Essbase Applications

ISBN: 978-1-4665-5330-9, 445 pp.
$69.95 / £44.99

Sign Up for our Free Newsletters and Connect with CRC Press IT Books on Facebook, Twitter, and LinkedIn to Keep the Discounts Coming!

INFORMATION SECURITY TODAY

The final word in information systems security

IT Today

In-depth... insightful...for today's technology leaders

IT Performance Improvement

Improving organizational performance through IT

WWW.CRCPRESS.COM | CRC PRESS Taylor & Francis Group

Order online and enter discount code **GVM24** to **SAVE 25%**

JCP Executive Series

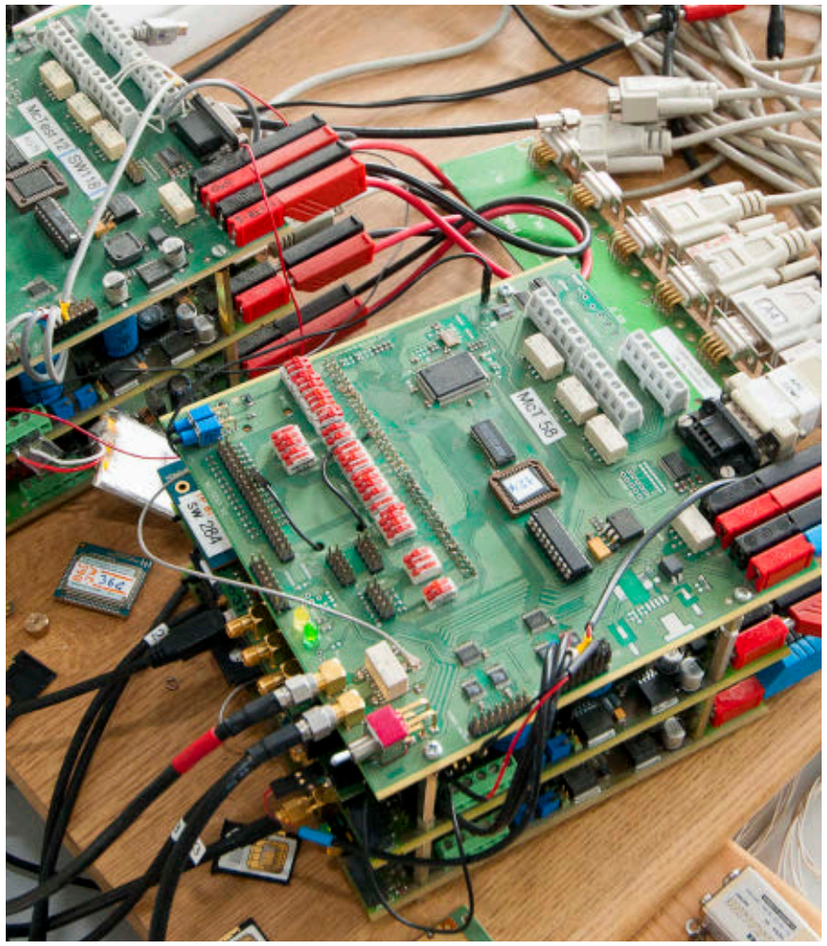
JCP and Gemalto M2M—A Winning Combination

Gemalto M2M's Thomas Lampart and Florian Denzin discuss the multifaceted interactions between M2M hardware, Java software, and the JCP. **BY STEVE MELOAN**

PHOTOGRAPHY BY TON HENDRIKS

Thomas Lampart is senior Java architect at Gemalto M2M (formerly Cinterion Wireless Modules) in Berlin, Germany. He has been involved in the development of Java-enabled machine-to-machine (M2M) data communication modules since 2002. During that time, he played a significant role in the creation of the two Java M2M JSRs for Java ME: JSR 195 (Information Module Profile, or IMP) and JSR 228 (Information Module Profile - Next Generation, or IMP-NG). He is the maintenance spec lead for JSR 228. In addition, Lampart is a member of the Expert Groups for JSR 360 (Connected Limited Device Configuration 8) and JSR 361 (Java ME Embedded Profile). Since 2012 he has also represented Gemalto M2M on the Java Community Process (JCP) Executive Committee.

Florian Denzin is a product portfolio director at Gemalto M2M. He has extensive experience in M2M communication in a variety of positions, including product definition and product strategy and innovation. His focus includes vertical industries such as automotive and metering. Denzin is also an active member in several industry forums and



Left: A stack of development boards. Right: Thomas Lampart (left) talks with Florian Denzin in the Gemalto M2M lab.

standardization organizations such as the European Telecommunications Standards Institute (ETSI) and GSMA, an association of mobile operators and related companies.

Gemalto M2M products and services enable devices, equipment, machines, and vehicles to communicate via cellular networks, facilitating edge-to-enterprise (E2E) connectivity and helping to create the Internet of Things. Gemalto M2M's offerings span automotive, mHealth, smart energy, security, payment solutions, remote monitoring and control, and mobile computing.

In this interview, Lampart and Denzin discuss their experiences and insights into embedded application development and the evolution of Java ME through the JCP.

Java Magazine: What motivated you to join the JCP, and how has it benefited Gemalto M2M?

Gemalto M2M: We joined the JCP in 2002, when we were still part of Siemens Communications. Our goal was to help Java become more applicable to the M2M space. So we created JSR 195 in conjunction with Nokia. The JSR addressed the needs of Information Module users working with such devices as modems, home electronics, industrial metering devices, and so on. Many of these users want to access Java runtime environments to speed development and increase portability; however, these devices typically have no graphical display capabilities or user-input mechanisms required by Mobile Information Device Profile [MIDP] 1.0. So JSR 195 remedied this, making such devices more broadly and easily applicable for Java developers. We also contributed JSR 228 with Nokia, which had similar goals, offering a subset of MIDP 2.0. These JSRs were the first step in helping Java to accommodate the needs of M2M technologies.

Creating standards is crucial for the success and growth of the M2M space.

The JSRs we participated in helped to move things forward, but the consumer market is very dynamic. Due to various target device constraints, Java ME has been somewhat behind the leading edge of Java technology. To remedy this, in 2012 Oracle announced its new Java ME strategy, which will allow greater software flexibility and broader device support. And last year, Gemalto M2M joined the Java Executive Committee to participate in the evolution of Java ME and help shape new directions.

Java Magazine: As an M2M technology provider, what unique perspectives does Gemalto bring to the JCP process?

Gemalto M2M: We are providing Java modules to a broad array of industries, which gives us unique insights into the needs of enterprises such as automotive, cellular M2M, mHealth, metering, and so on. Our experience with these application areas provides insight and perspective on how Java can be used to interconnect millions of tiny machines. And we try to pass this experience on by creating JSRs and participating in the Java Executive Committee. We are currently active members of the JSR 360 and JSR 361 Expert Groups. The goal of these JSRs is to bring the power and flexibility of Java 8 language features to Java ME, while maintaining a small footprint and a consistent developer environment.

Java Magazine: What have you gleaned from your experiences that you would like to pass on about the JCP community?

Gemalto M2M: I think we all agree that open standards have been a key ingredient in the growth and success of Java. It has led to an entire ecosystem of freely available tools and code applications that have promoted Java's acceptance. But now the challenge is to move Java forward in this same way for the embedded space, currently the fastest-growing application area. The JCP process is the key. Developers must be encouraged to contribute their experiences and expertise toward making Java as flexible and applicable as it can be for the explosion of new devices coming online. We need to focus more on promoting the JCP to embedded developers at Java conferences and through technology publications.

Java Magazine: Gemalto M2M is a leading provider of M2M technology that is also used for mHealth solutions. With explosive growth on the horizon, how has this application area affected your involvement with the JCP?

Gemalto M2M: Supporting the delivery of healthcare with mobile devices [mHealth] is an incredibly broad field. Besides real-time monitoring of patient data, and telemedicine, it also includes the vast domain of collecting and administering patient data using

Lampart checks his schedule and catches up on e-mail.

M2M Modules

In 2010, Gemalto M2M (then Cinterion) won the Duke's Choice Award at JavaOne in the Innovative Java Building Block category. Their TC65i and EGS5 Java ME–powered modules bring always-on wireless M2M solutions to the mHealth domain. Automated Diabetes Management Systems (ADMS), for example, use the TC65i. In a remote data center, automated care algorithms are triggered to deliver crucial feedback to help medical practitioners bring blood sugar back into the normal range. Gemalto M2M modules offer open standards advantages in advanced telemedicine applications, enabling 24/7 wireless communications between medical devices and care data systems.

Denzin, Lampart, and Gemalto M2M team members connect a wind generator to a wireless gateway.

mobile technologies. Our area of focus requires a reliable and secure programming environment with a lot of supportive functionalities. And that's why we chose the Java platform.

Here's an application that illustrates what we're talking about. The Philips Respironics Sleep System One uses the Gemalto M2M TC65i Java module with the GSM/GPRS network to allow doctors to remotely analyze sleep apnea patients' breathing data and make therapy adjustments over the air. These modules have IP-based cellular data connectivity and can even integrate GPS data and other information. Software development is done using standard IDEs such as Eclipse or NetBeans, and all other familiar Java skills apply. And of course, Java security is also built in. But as the mHealth application domain continues to evolve, participation in the JCP is essential to make sure that Java ME is in touch with the special needs of this important application area. And we are doing that, and encouraging others to do it.

Java Magazine: How will the upcoming convergence between Java SE and Java ME affect your enterprise?

Gemalto M2M: A number of new and useful features will become available for Java ME users through this planned convergence, which is detailed in JSR 360 and JSR 361. Having a common evolution path for both Java SE and Java ME will build a bridge for Java SE developers toward Java ME. This is an essential step to address the vast, fast-growing, and diverse M2M application space. Some of our customers are already familiar with Java SE, and will find it even easier to migrate into Java ME for their resource-constrained applications.

Gemalto M2M's own development activities will also be enhanced by this convergence. It will mean that code we have previously developed using Java SE may become useful in the embedded domain.

Also later, there has to be cross-pollination between the embedded space and Java SE. Some of the JSRs from the embedded world will be beneficial to Java SE, such as location and messaging functionalities. So, both worlds can benefit from this alignment.

Java Magazine: Any closing thoughts on the JCP and the embedded space?

Gemalto M2M: The JCP encourages participants to become engaged with one another and to ask, "What are you working on? Here's what we are doing. Let's compare notes." The needs of the automotive industry can be very different from what's going on in mHealth or metering applications. But there is something to be gained from all these areas mixing. This is done through formal channels within the JCP process, and it's also done informally where individuals just shoot e-mails back and forth or meet face-to-face at various conferences. All these interactions lead to a deeper understanding of the technical challenges and solutions. It's a productive ecosystem that commingles ideas and inspires better work from everyone. `</article>`

Steve Meloan is a former C/UNIX software developer who has covered the web and the internet for such publications as *Wired*, *Rolling Stone*, *Playboy*, *SF Weekly*, and the *San Francisco Examiner*. He recently published a science-adventure novel, *The Shroud*, and regularly contributes to *The Huffington Post*.

LEARN MORE

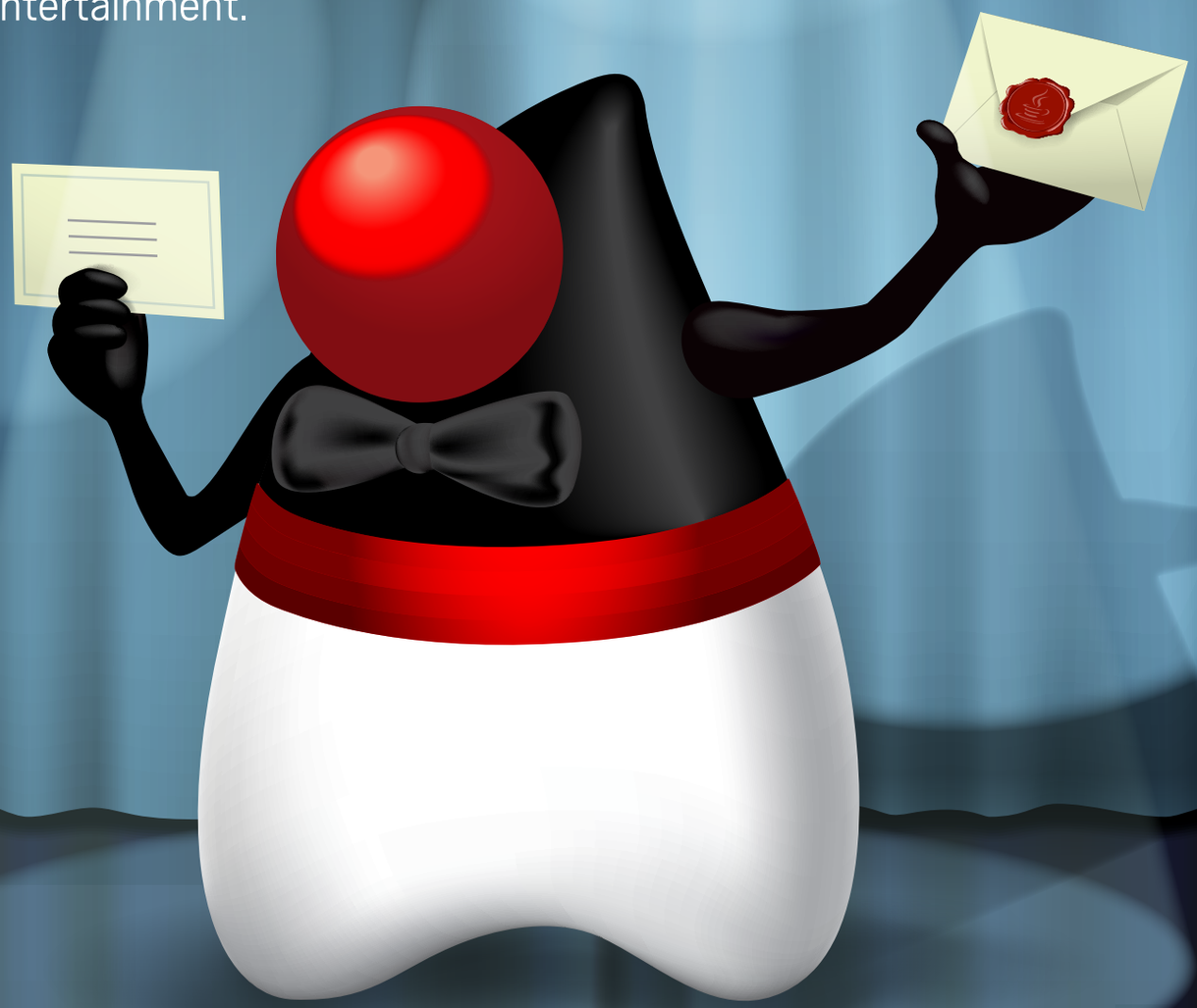
- Java Community Process
- Gemalto and Oracle

2013 DUKE'S CHOICE AWARDS

From the serious to the whimsical, the 2013 Duke's Choice Award winners include Java projects in medicine, technology, transportation, aeronautics, education, and entertainment.

BY PHILIP J. GILL

This year's Duke's Choice Award winners honoring innovation in Java development range from the serious and cerebral to the practical, amusing, and entertaining. These projects are pushing the frontiers of medicine and technology by simulating the human brain and musculoskeletal system; providing guidance to cars on the highway, satellites in space, and robotic fish under water; training tomorrow's Java programmers; making Java applications more secure; and building communities.



ART BY I-HUA CHEN

THIS YEAR'S WINNERS

(in alphabetical order by organization name)

Contrast, Contrast Security

Devoxx4Kids, DEVOXX

The Dutch Java User Group

ISIM, ISBAK

Bintray, JFrog

jCardSim, Licel

GEONS Ground Support System, National Aeronautics and Space Administration

OpenSim, The National Institutes of Health Center for Biomedical Computation and National Center for Simulation in Rehabilitation Research

openHAB

Jessikommand, Robotswim

Neuroph, University of Belgrade's Faculty of Organizational Sciences

SERIOUS AND CEREBRAL

Musculoskeletal disorders and diseases such as rheumatoid arthritis and osteoporosis will affect one in two Americans during their lifetimes, and they are a leading cause of disability and healthcare costs, according to the publication *The Burden of Musculoskeletal Diseases in the United States* (American Academy of Orthopaedic Surgeons, 2011). To address this issue, the National Institutes of Health (NIH) funds various research initiatives, including the NIH Center for Biomedical Computation (known as Simbios) and the National Center for Simulation in Rehabilitation Research (NCSRR) at Stanford University, in Stanford, California.

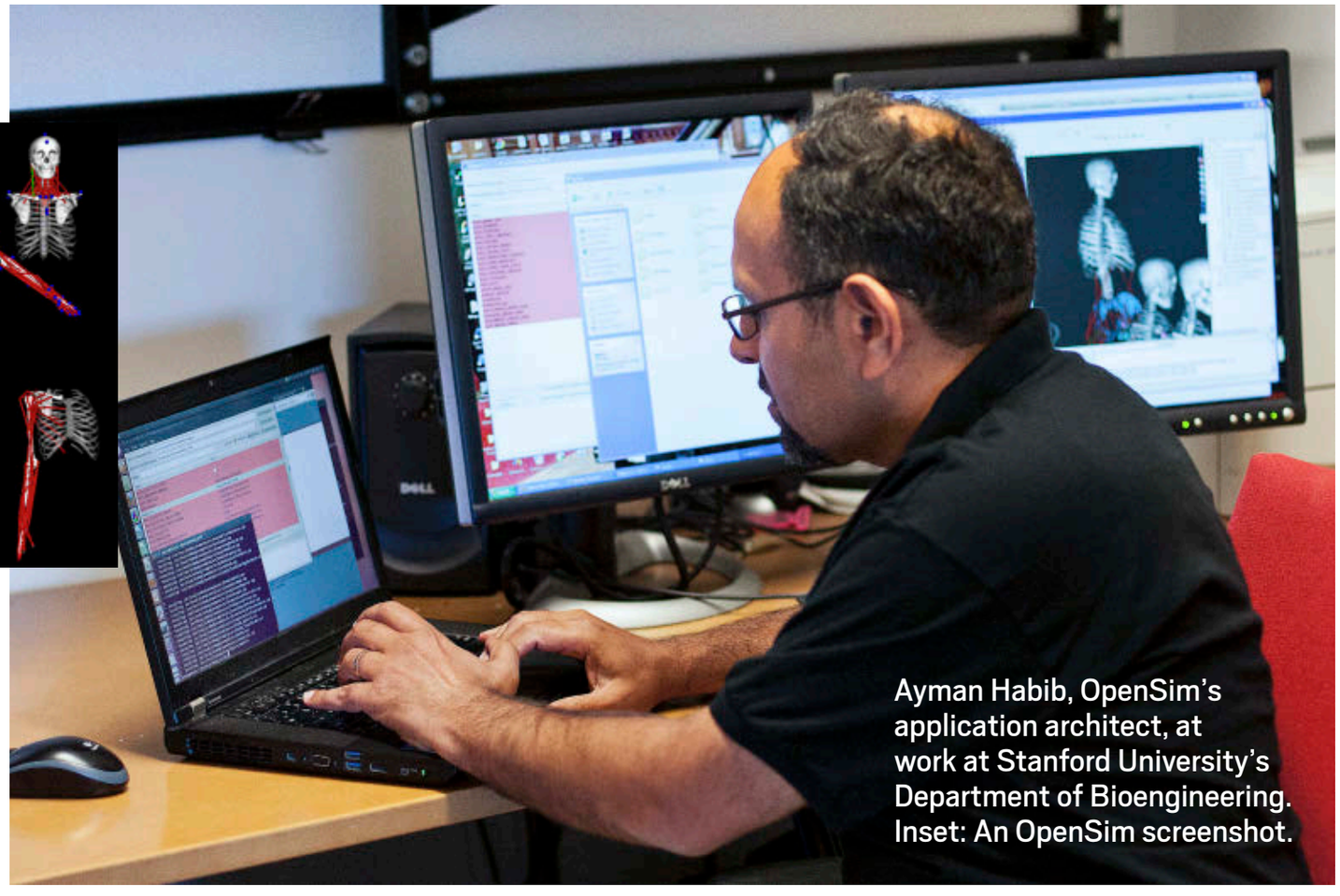
The team at Simbios and the NCSRR created **OpenSim**, an application for modeling the muscles, joints, and bones that make up the body and simulating how humans move. This free tool enables researchers, therapists, students, and product designers to develop, analyze, simulate, and share information to find treatments—and, perhaps one day, cures—for a variety of musculoskeletal disorders and diseases. In addition, teams from DARPA's Warrior Web effort are using the software to help design the next generation of "smart suits" for soldiers to reduce injury risk and fatigue.

DECIDING FACTOR

"Cross-platform was key to our selection of Java as the GUI front end to OpenSim."

—Ayman Habib, Application Architect, OpenSim

"OpenSim is an open platform that can be easily shared because it is built on an open-system, open source approach that includes Java technology and the NetBeans Platform," explains Ayman Habib, OpenSim's application architect and a member of the Neuromuscular Biomechanics Lab in Stanford University's Department of Bioengineering. "OpenSim models are composed of components that can be shared," Habib notes. If researchers develop a new type of muscle model, for example, they can write the



Ayman Habib, OpenSim's application architect, at work at Stanford University's Department of Bioengineering. Inset: An OpenSim screenshot.

PHOTOGRAPH BY BOB ADLER; OPENSIM SCREENSHOT COURTESY OF JEN HICKS AND KEVIN XU

OpenSim's Jen Hicks, R&D manager, and Habib discuss plans for the release of OpenSim software.

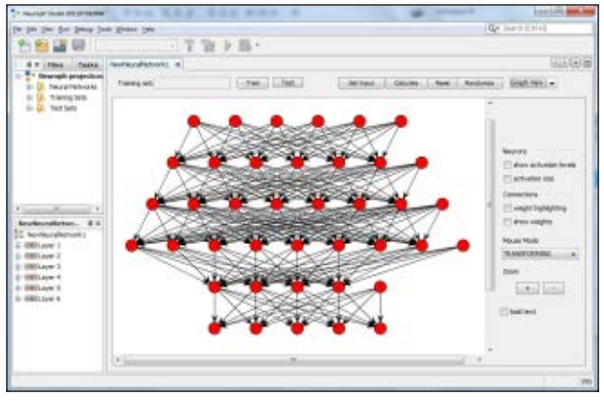
PHOTOGRAPH BY BOB ADLER

new component as a C++ class, then compile it as a dynamic library that is loaded into the application and its GUI, which is written in Java.

"Cross-platform was key to our selection of Java as the GUI front end to OpenSim," says Habib. "Responsive GUI was a big issue that wouldn't have been possible using other technologies, especially when running computation-intensive tasks. The NetBeans Platform adds modular development, plug-ins, and APIs to support selection, editing, preferences, layouts, docking, and more."

Neuroph, an all-Java neural network framework and integrated development environment (pictured below), is helping researchers and scientists simulate brain activities and simplified brain-like structures that can be used for problem-solving, recognition, prediction, control, modeling, and functional approximations in medicine, robotics, finance, and software.

The Neuroph project started as a university research project at the University of Belgrade's Faculty of Organizational Sciences, in Belgrade, Serbia. It has since evolved into a leading open source project in its field, with contributors from across the globe.



Community Choice 2013 a Tie!

The Java community has spoken, and it's a tie!

This year the two winners of the Community Choice Award are Bintray, a social network for developers from Netanay, Israel–based JFrog, and Contrast, a Java EE security plug-in from Columbia, Maryland–based Contrast Security.

Taking a cue from Facebook, Bintray provides a free, cloud-based social networking platform that enables software developers to download, store, promote, and share executable binary code and libraries. Profile pages list members, available binaries and downloads, relevant websites, bug trackers, community reviews, and the option to watch for future updates.

The Contrast plug-in invisibly monitors applications during testing and automatically identifies security vulnerabilities. Its patented technology weaves "security sensors" into the Java Virtual Machine (JVM), libraries, and the application's custom code, and reports suspected and known vulnerabilities.

PROVIDING GUIDANCE

While OpenSim simulates the human body and Neuroph the human brain, one other winner this year seeks to simulate and analyze something that is at times far more complex and frustrating: traffic. **ISIM**, from intelligent transport system developer ISBAK of Istanbul, Turkey, is an all-Java traffic planning, simulation, and analysis tool that lets users simulate, plan, and construct road networks using different road, junction, car, speed, and other relevant parameters for maximum results.

In space, the **GEONS Ground Support System (GGSS)** is an analysis and mission operations tool that uses the GPS-Enhanced Onboard Navigation System (GEONS) from the National Aeronautics and Space Administration (NASA). GGSS uses the NetBeans Platform, developed in the NetBeans IDE, as the basis of its ground system software and will support

Left: A Jessiko robotic fish. Right: Robotswim CEO and Founder Christophe Tiraby watches robotic fish with visitors to the RobotWorld conference in Seoul, South Korea.

PHOTOGRAPHS COURTESY OF ROBOTSWIM

JAVA-BASED COMMANDS

"You can use Jessikommand to create movement scenarios and then to send the orders to the robots."

—Guillaume Genty, Lead Software Engineer, Robotswim

2013 DUKE'S CHOICE AWARDS

Left: A school of robotic fish are powered by a Java-based command and control software. Right: Robotswim's Guillaume Genty and Tiraby take care of the robotic fish environment.

PHOTOGRAPHS COURTESY OF ROBOTSWIM AND BY GUILLAUME BONN/GETTY IMAGES

a 2014 launch of the Magnetospheric Multiscale (MMS) mission.

Developed by NASA partner a.i. solutions, the GGSS is deployed in the MMS mission operations control room at the Goddard Space Flight Center, in Greenbelt, Maryland. "Combining JDK 7, the NetBeans Platform, and JavaFX saved an estimated 35 percent in time versus estimates for primary software development," says a.i. solutions Senior Software Engineer Sean Phillips.

Three-year-old French startup Robotswim in Palaiseau, France—just outside of Paris in what's known as France's Silicon Valley—has produced the world's smallest commercially available robotic fish, Jessiko, measuring 22 centimeters in length. Jessiko is also the only robotic fish that can be programmed to swim in groups, or schools, through the power of **Jessikommand**, a Java-based command and control software. Jessikommand communicates with the fish using a beacon network.

"Jessikommand controls the beacon network and, using those beacons, sends real-time orders to robots and asks them their status, using a patented, two-way optical communication system," explains Guillaume Genty, Robotswim's lead software engineer.

"You can use Jessikommand to create movement scenarios and then to send the orders to the robots, and it also helps robots to stay coordinated with each other," Genty explains. "For example, in some scenarios, we mark a fish as a master and give it real-time precise orders, and then ask other fish to follow it."

To date, the largest school of Jessiko robotic fish the company has deployed at one time is 50. The typical use is for consumer entertainment, where a school of Jessiko fish are programmed to swim in patterns in aquariums, changing colors as they go. "To create a real school, it is better to have more than 30 robots because some of them are sometimes capricious," Genty adds. "They get lost and continue on their own before going back to the school. It's really artificial life."

Devoxx4Kids Founder Stephan Janssen (center) problem-solves with workshop participants.

THE FUTURE OF JAVA

Some of this year's winners have a decidedly forward-looking stance. **Devoxx4Kids**, for example, is a program from the team behind DEVOXX, one the world's largest Java developer conferences. These education sessions teach children between the ages of 8 and 14 computer programming and logic in languages other than English. In these sessions, young people create computer games, program robots, and learn about electronics.

The **jCardSim** program from Licel, a Moscow, Russia–based independent software developer specializing in development tools for Java Card technology, was developed to help students and to encourage them to use Java Card, the world's most widely used platform for smartcard and other devices with limited memory capacity. Licel's jCardSim is a Java Card technology software simulator that enables students to prototype and test applications across multiple platforms.

The next step in the evolution of Java is its leading role in the emerging Internet of Things. The Internet of Things holds great promise, but the realization of its benefits is threatened by the proliferation of single-purpose devices that have their own user interfaces and back-end ecosystems.

To meet this challenge, the talented contributors to the **open Home Automation Bus (openHAB) project** have developed a central integration point so that developers can easily integrate devices and applications. openHAB is a pure-Java home automation solution based on OSGi standards; the core of the openHAB runtime is the Equinox OSGi runtime and Eclipse Jetty web server. openHAB comes with a scripting language so that developers can easily define any kind of automation logic they have in mind.

PHOTOGRAPH COURTESY OF DEVOXX

JUDGES AND PROCESS

The winners of the 11th annual Duke's Choice Awards were selected in a three-part process. All members of the Java community were first invited to submit nominations to this year's judges. Next, the judges selected nine winners and five candidates for the second annual Community Choice Award. Finally, Community Choice Award nominees were posted on Java.net, and all members of the Java community were invited to vote for their favorite.

This year's judges were

- **Yara Senger**, SouJava
- **John Yeary**, president and founder, Greenville Java Users Group
- **Martijn Verburg**, London Java Community
- **Michelle Kovac**, Java marketing and operations
- **Arun Gupta**, Java evangelist and GlassFish community member
- **Sharat Chander**, Java evangelist team manager

The winners will be honored at JavaOne September 22–26 in San Francisco, California.

Duke's Choice Award China Winners

Fall 2012 marked an expansion of the Duke's Choice Award program, which now includes regional awards that are announced in conjunction with each international JavaOne conference. One of the highlights of JavaOne Shanghai was the presentation of the very first Duke's Choice Award China. The 2013 winners were the Moco technology integration server project; the X Fantasy MMORPG real-time web-based game; and the OSChina.NET open source community infrastructure.

Zheng Ye, lead developer of the Moco project, describes Moco as a tool for facilitating the development of applications that interact with web services. "Using web services requires integration, which involves communications between the different ends," Ye explains. "The consumer of a web service needs to communicate with something to support its development. But what if the server side of the web service is also still under development? Moco is designed to address these issues. Moco enables testing and integration to be completed in a smaller scope, which means simplification."

From left: China Duke's Choice winners Jia Ke, Zhang Hailong, and Zheng Ye

Jia Ke, lead developer of the X Fantasy game, presented a JavaOne Shanghai session, "Lessons from Developing the X Fantasy Web Game in Java." When asked what impact he expects Java SE 8 lambda expressions to have on Java game programming, Ke answered: "What lambda expressions can do is simplify or polish anonymous classes. However, less than one-thousandth of the code in X Fantasy uses anonymous classes, so we don't expect much immediate impact on our own development. Still, we agree that the support of lambda expressions in Java 8 will bring Java experts more flexibility and choices for optimizing code, which we think is very valuable."

Zhang Hailong, who leads the Guangdong Java User Group, is a cofounder of OSChina—which, with more than a million users, is the largest open source community in China. "The OSChina website is built on top of Java technology—for example, the JDK, Tomcat, and Velocity," Hailong says. "We've expended lots of effort optimizing the website using specific architectures and caching technology. OSChina demonstrates how to build a low-resource-consuming, lightweight, and very fast website using Java technology. We have open-sourced a lot of our code to let other people utilize what we have done." Hailong also notes that OSChina has a forum where Java developers can ask questions and discuss Java technologies. "We also organize offline meetups to enhance the technical atmosphere and help Java developers improve their skills," he says.

—Kevin Farnham

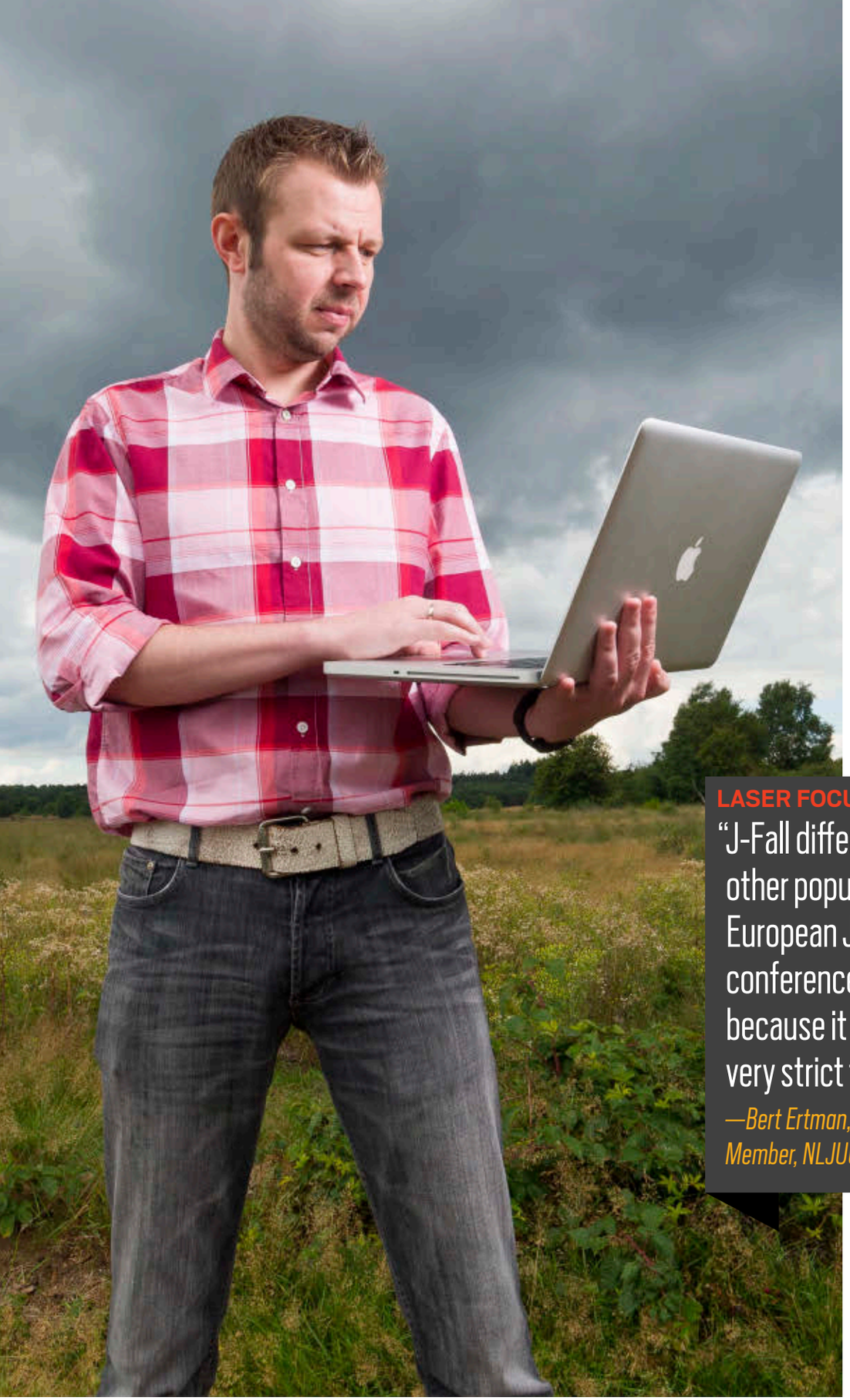
PHOTOGRAPHS BY FEIFAN ZHOU/24 EIGA AGENCY AND COURTESY OF NLJUG

NLJUG's J-Fall conference

"LOCAL FOR LOCAL"

Last year the Duke's Choice Awards established a new tradition by announcing its first-ever awards to a Java user group (JUG)—two, in fact: the London Java Community and JDuchess. This year that tradition continues, with an award going to the **Dutch Java User Group (NLJUG)**—an organization with national reach throughout the Netherlands. Since its founding in 2004, the organization has grown from about 100 to almost 3,500 members.

"NLJUG considers itself to be a platform from which other initiatives



PHOTOGRAPH BY TON HENDRIKS

LASER FOCUS

"J-Fall differs from other popular European Java conferences because it has a very strict focus."

—Bert Ertman, Board Member, NLJUG

2013 DUKE'S CHOICE AWARDS

Ertman discusses strategies for managing a large Java user group.

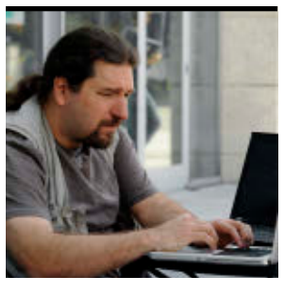
can be organized," says Bert Ertman, NLJUG board member and a fellow at Luminis, a software development and consulting firm based in Apeldoorn, the Netherlands.

NLJUG is particularly known for its annual J-Fall conference, the leading Java conference of its kind for the Dutch-speaking community. Started in the same year as NLJUG, the conference now attracts about 1,200 people in a typical year.

"J-Fall differs from other popular European Java conferences because it has a very strict focus," says Ertman. "We chose to be 'local for local,' meaning that most of the speakers are local and a lot of the sessions are presented in Dutch. While most people in the Netherlands are able to speak and understand English very well, it still adds value to have content presented in your native language and have the bar for interacting as low as possible." Plus, the 10-member program committee comes from the local community. "By rotating the committee," Ertman continues, "we strike a nice balance and mirror the tastes of the Dutch Java community."

Another important reason for featuring local speakers is to let J-Fall serve as a breeding ground for speakers and talent. "It is getting harder and harder to become a speaker at well-known conferences such as DEVOXX and JavaOne if you don't have a track record as a speaker," Ertman says. "J-Fall is that place where talented Dutch presenters can have a taste of what it's like to present and can further master the skills required. I am very proud to see that over the past 10 years we have had some starting presenters who have now become very well-known, internationally acclaimed conference speakers." `</article>`

Philip J. Gill is a San Diego, California–based writer and editor who has been following Java technology for more than 20 years.



Part 2

Java 8: Lambdas

Learn how to use lambda expressions to your advantage.

TED NEWARD

BIO

The release of Java SE 8 swiftly approaches. With it come not only the new linguistic lambda expressions (also called *closures* or *anonymous methods*)—along with some supporting language features—but also API and library enhancements that will make parts of the traditional Java core libraries easier to use. Many of these enhancements and additions are on the Collections API, and because the Collections API is pretty ubiquitous across applications, it makes the most sense to spend the majority of this article on it.

However, it's likely that most Java developers will be unfamiliar with the concepts behind lambdas and with how designs incorporating lambdas look and behave. So, it's best to examine why these designs look the way they do before showing off the final stage. Thus, we'll look at some before and after approaches to see how to approach a problem pre-lambda and post-lambda.

Note: This article was written against the b92 (May 30, 2013) build of Java SE 8, and the APIs, syntax, or semantics might have changed by the time you read this or by the time Java SE 8 is released. However, the concepts behind these APIs, and the approach taken by the Oracle engineers, should be close to what we see here.

Collections and Algorithms

The Collections API has been with us since JDK 1.2, but not all parts of it have received equal attention or love from the developer community. Algorithms, a more functional-centric way of interacting with collections, have been a part of the Collections API since its initial release, but they often get little attention, despite their usefulness. For example, the Collections class sports a dozen or so methods all designed to take a collection as a parameter and perform some operation against the collection or its contents.

BE ATTENTIVE

Algorithms, a more functional-centric way of interacting with collections, have been a part of the Collections API since its initial release, but they often get little attention, despite their usefulness.

Consider, for example, the Person class shown in **Listing 1**, which in turn is used by a List that holds a dozen or so Person objects, as shown in **Listing 2**.

Now, assuming we want to examine or sort this list by last name and then by age, a naive approach is to write a for loop (in other words, implement the sort by hand each time we need to sort). The problem with this, of course, is that this violates DRY (the Don't Repeat Yourself principle) and, worse, we have to reimplement it each time, because for loops are not reusable.

The Collections API has a better approach: the Collections class sports a sort method that will sort the contents of the List. However, using this requires the Person class to implement the Comparable method (which is called a *natural ordering*, and defines a default ordering for all Person types) or you have to pass in a Comparator instance to define how Person objects should be sorted.

So, if we want to sort first

PHOTOGRAPH BY PHIL SALTONSTALL

by last name and then by age (in the event the last names are the same), the code will look something like **Listing 3**. But that's a lot of work to do something as simple as sort by last name and then by age. This is exactly where the new closures feature will be of help, making it easier to write the Comparator (see **Listing 4**).

The Comparator is a prime example of the need for lambdas in the language: it's one of the dozens of places where a one-off anonymous method is useful. (Bear in mind, this is probably the easiest—and weakest—benefit of lambdas. We're essentially trading one syntax for another, admittedly terser, syntax, but even if you put this article down and walk away right now, a significant amount of code will be saved just from that terseness.)

If this particular comparison is something that we use over time, we can always capture the lambda as a Comparator instance, because that is the signature of the method—in this case, "int compare(Person, Person)"—that the lambda fits, and store it on the Person class directly, making the implementation of the lambda easier (see **Listing 5**) and its use even more readable (see **Listing 6**).

Storing a Comparator<Person> instance on the Person class is a bit odd, though. It would make more sense to define a method that does the comparison, and use that instead of a Comparator instance. Fortunately, Java will allow any method to be used that satisfies the same signature as the method on Comparator, so it's equally possible to write the BY_LAST_AND_AGE Comparator as a standard instance or static method on Person (see **Listing 7**) and use it instead (see **Listing 8**).

BE ANONYMOUS

The Comparator is a prime example of the need for lambdas in the language: it's one of the dozens of places where a one-off anonymous method is useful.

Thus, even without any changes to the Collections API, lambdas are already helpful and useful. Again, if you walk away from this article right here, things are pretty good. But they're about to get a lot better.

Changes in the Collections API

With some additional APIs on the Collection classes themselves, a variety of new and more powerful approaches and techniques open up, most often leveraging techniques drawn from the world of functional programming. No knowledge of functional programming is necessary to use them, fortunately, as long you can open your mind to the idea that functions are just as valuable to manipulate and reuse as are classes and objects.

Comparisons. One of the drawbacks to the Comparator approach shown earlier is hidden inside the Comparator implementation. The code is actually doing two comparisons, one as a "dominant" comparison over the other, meaning that last names are compared first, and age is compared only if the last names are identical. If project requirements later demand that sorting be done by age first and by last names second, a new Comparator must be written—no parts of compareLastAndAge can be reused.

This is where taking a more functional approach can add some powerful benefits. If we look at that comparison as entirely separate Comparator instances, we can combine them to create the precise kind of comparison needed (see **Listing 9**).

Historically, writing the combination by hand has been less productive, because by the time you write the code to do the combination, it would be just as fast (if not faster) to write the multistage comparison by hand.

As a matter of fact, this "I want to compare these two *X* things by comparing values returned to me by a method on each *X*" approach is such a common thing, the platform gave us that functionality

LISTING 1 / LISTING 2 / LISTING 3 / LISTING 4 / LISTING 5 / LISTING 6

```
public class Person {
 public Person(String fn, String ln, int a) {
  this.firstName = fn; this.lastName = ln; this.age = a;
 }

 public String getFirstName() { return firstName; }
 public String getLastName() { return lastName; }
    public int getAge() { return age; }
}
```

Download all listings in this issue as text

out of the box. On the Comparator class, a comparing method takes a function (a lambda) that extracts a comparison key out of the object and returns a Comparator that sorts based on that. This means that **Listing 9** could be rewritten even more easily as shown in **Listing 10**.

Think for a moment about what this is doing: the Person is no longer about sorting, but just about extracting the key by which the sort should be done. This is a good thing—Person shouldn't have to think about how to sort; Person should just focus on being a Person.

It gets better, though, particularly when we want to compare based on two or more of those values.

Composition. As of Java 8, the Comparator interface comes with several methods to combine Comparator instances in various ways by stringing them together. For example, the Comparator .thenComparing() method takes a Comparator to use for comparison after the first one compares. So, re-creating the "last name then age" comparison can now be written in terms of the two Comparator instances LAST and AGE, as shown in **Listing 11**. Or, if you prefer to use methods rather than Comparator instances, use the code in **Listing 12**.

By the way, for those who didn't grow up using Collections.sort(), there's now a sort() method directly on List. This is one of the neat things about the introduction of interface default methods: where we used to have to put that kind of noninheritance-based reusable behavior in static methods, now it can be hoisted up into interfaces. (See the previous article in this series for more details.)

Similarly, if the code needs to sort the collection of Person objects by last name and then by first name, no new Comparator needs to be written, because this comparison can, again, be made of the two particular atomic comparisons shown in **Listing 13**.

This combinatory "connection" of methods, known as *functional composition*, is common in functional programming and at the heart of why functional programming is as powerful as it is.

It's important to understand that the real benefit here isn't just in the APIs that enable us to do comparisons, but the ability to pass bits of executable code (and then combine them in new and interesting ways) to create opportunities for reuse and design. Comparator is just the tip of the iceberg. Lots of things can be made more flexible and powerful, particularly when combining and composing them.

LISTING 7 LISTING 8 LISTING 9 LISTING 10 LISTING 11 LISTING 12

```
public static int compareLastAndAge(Person lhs, Person rhs) {
 if (lhs.lastName.equals(rhs.lastName))
  return lhs.age - rhs.age;
 else
  return lhs.lastName.compareTo(rhs.lastName);
}
```

Download all listings in this issue as text

Iteration. As another example of how lambdas and functional approaches change the approach to code, consider one of the fundamental operations done with collections: that of iterating over them. Java 8 will bring to collections a change via the forEach() default method defined on the Iterator and Iterable interfaces. Using it to print each of the items in the collection, for example, requires passing a lambda to the forEach method on an Iterator, as shown in **Listing 14**.

Officially, the type of lambda being passed in is a Consumer instance, defined in the java.util .function package. Unlike traditional Java interfaces, however, Consumer is one of the new functional interfaces, meaning that direct implementations will likely never happen—instead, the new way to think about it is solely in terms of its single, important method, accept, which is the method the lambda provides. The rest (such as compose and andThen) are utility methods defined in terms of the important method, and they are designed to support the important method.

For example, andThen() chains two Consumer instances together, so the first one is called first and the second is called immediately after into a single Consumer. This provides useful composition techniques that are a little outside the scope of this article.

Many of the use cases involved in walking through a collection have the purpose of finding items that fit a particular criterion—for example, determining which of the Person objects in the collection are of drinking age, because the automated code system needs to send everyone in that collection a beer. This "act upon a thing coming from a group of things" is actually far more widespread than just operating upon a col-

lection. Think about operating on each line in a file, each row from a result set, each value generated by a random-number generator, and so on. Java SE 8 generalized this concept one step further, outside collections, by lifting it into its own interface: Stream.

Stream. Like several other interfaces in the JDK, the Stream interface is a fundamental interface that is intended for use in a variety of scenarios, including the Collections API. It represents a stream of objects, and on the surface of things, it feels similar to how Iterator gives us access one object at a time through a collection.

However, unlike collections, Stream does not guarantee that the collection of objects is finite. Thus, it is a viable candidate for pulling strings from a file, for example, or other kinds of on-demand operations, particularly because it is designed not only to allow for composition of functions, but also to permit parallelization "under the hood."

Consider the earlier requirement: the code needs to filter out any Person object that is not at least 21 years of age. Once a Collection converts to a Stream (via the stream() method defined on the Collection interface), the filter method can be used to produce a new Stream through which only the filtered objects come (see **Listing 15**).

The parameter to filter is a Predicate, an interface defined as taking one genericized parameter and returning a Boolean. The intent of the Predicate is to determine whether the parameter object is included as part of the returned set.

The return from filter() is another Stream, which means that the filtered Stream is also available for further manipulation, such as to forEach() through each of the elements that come through the Stream, in this case to display the results (see **Listing 16**).

This neatly demonstrates the composability of streams—we can take streams and run them through a variety of atomic operations, each of which do one—and only one—thing to the stream. Additionally, it's important to note that filter() is lazy—it will filter only as it needs to, on demand, rather than going through the entire collection of Person objects and filtering ahead of time (which is what we're used to with the Collections API).

Predicates. It might seem odd at first that the filter() method takes only a single Predicate. After all, if a goal was to find all the Person objects whose age is greater than 21 and whose last name is Neward,

LISTING 13 LISTING 14 / LISTING 15 / LISTING 16 / LISTING 17

```
Collections.sort(people,
  Comparators.comparing(Person::getLastName)
  .thenComparing(Person::getFirstName));
```

Download all listings in this issue as text

it would seem that filter() could or should take a pair of Predicate instances. Of course, this opens a Pandora's box of possibilities. What if the goal is to find all Person objects with an age greater than 21 and less than 65, and with a first name of at least four or more characters? Infinite possibilities suddenly open up, and the filter() API would need to somehow approach all of these.

Unless, of course, a mechanism were available to somehow coalesce all of these possibilities down into a single Predicate. Fortunately, it's fairly easy to see that any combination of Predicate instances can themselves be a single Predicate. In other words, if a given filter needs to have condition A be true and condition B be true before an object can be included in the filtered stream, that is itself a Predicate (A and B), and we can combine those two together into a single Predicate by writing a Predicate that takes any two Predicate instances and returns true only if both A and B each yield true.

This "and"ing Predicate is—by virtue of the fact that it knows only about the two Predicate instances that it needs to call (and nothing about the parameters being passed in to each of those)—completely generic and can be written well ahead of time.

If the Predicate closures are stored in Predicate references (similar to how Comparator references were used earlier, as members on Person), they can be strung together using the and() method on them, as shown in **Listing 17**.

As might be expected, and(), or(), and xor() are all available. Make sure to check the Javadoc for a full introduction to all the possibilities.

map() and reduce(). Other common Stream operations include map(), which applies a function across each element present within a Stream to produce a result out of each element. So, for example, we can obtain the age of each Person in the collection by applying a simple function to retrieve the age out of each Person, as shown in **Listing 18**.

For all practical purposes, IntStream (and its cousins LongStream and DoubleStream) is a specialization of the Stream<T> interface (meaning that it creates custom versions of that interface) for those primitive types.

This, then, produces a Stream of integers out of a Collection of Person instances. This is also sometimes known as a *transformation operation*, because the code is transforming or projecting a Person into an int.

Similarly, reduce() is an operation that takes a stream of values and, through some kind of operation, reduces them into a single value. Reduction is an operation already familiar to developers, though they might not recognize it at first: the COUNT() operator from SQL is one such operation (reducing from a collection of rows to a single integer), as are the SUM(), MAX(), and MIN() operators. Each of these takes a stream of values (rows) and produces a single value (the integer) by applying some operation (for example, increment a counter, add the value to a running total, select the highest, or select the lowest) to each of the values in the stream.

BE REDUCTIONIST
Doing this bypasses an interesting opportunity to explore **one of the more powerful features of the new Java API, that of doing a reduction**—coalescing a collection of values down into a single one through some custom operations.

So, for example, you could sum the values prior to dividing by the number of elements in the stream to obtain an average age. Given the new APIs, it's easiest to just use the built-in methods, as shown in **Listing 19**.

But doing this bypasses an interesting opportunity to explore one of the more powerful features of the new Java API, that of doing a reduction—coalescing a collection of values down into a single one through some custom operation. So, let's rewrite the summation part of this using the new reduce() method:

LISTING 18 LISTING 19 / LISTING 20

```
IntStream ages =
  people.stream()
    .mapToInt((it) -> it.getAge());
```

Download all listings in this issue as text

```
.reduce(0, (l, r) -> l + r);
```

This reduction, also known in functional circles as a *fold*, starts with a seed value (0, in this case), and applies the closure to the seed and the first element in the stream, taking the result and storing it as the accumulated value that will be used as the seed for the next element in the stream.

In other words, in a list of integers such as 1, 2, 3, 4, and 5, the seed 0 is added to 1 and the result (1) is stored as the accumulated value, which then serves as the left-hand value in addition to serving as the next number in the stream (1+2). The result (3) is stored as the accumulated value and used in the next addition (3+3). The result (6) is stored and used in the next addition (6+4), and the result is used in the final addition (10+5), yielding the final result 15. And, sure enough, if we run the code in **Listing 20**, we get that result.

Note that the type of closure accepted as the second argument to reduce is an IntBinaryOperator, defined as taking two integers and returning an int result. IntBinaryOperator and IntBiFunction are examples of specialized functional interfaces—including other specialized versions for Double and Long—which take two parameters (of one or two different types) and return an int. These specialized versions were created mostly to ease the work required for using the common primitive types.

IntStream also has a couple of helper methods, including the average(), min(), and max() methods, that do some of the more common integer operations. Additionally, binary operations (such as summing two numbers) are also often defined on the primitive wrapper classes for that type (Integer::sum, Long::max, and so on).

More maps and reduction. Maps and reduction are useful in a variety of situations beyond just

simple math. After all, in any case where a collection of objects can be transformed into a different object (or value) and then collected into a single value, map and reduction operations work.

The map operation, for example, can be useful as an extraction or projection operation to take an object and extract portions of it, such as extracting the last name out of a Person object:

```
Stream<String> lastNames =
 people.stream()
   .map(Person::getLastName);
```

Once the last names have been retrieved from the Person stream, the reduction can concatenate strings together, such as transforming the last name into a data representation for XML. See **Listing 21**.

And, naturally, if different XML formats are required, different operations can be used to control the contents of each format, supplied either ad hoc, as in **Listing 21**, or from methods defined on other classes, such as from the Person class itself, as shown in **Listing 22**, which can then be used as part of the map() operation to transform the stream of Person objects into a JSON array of object elements, as shown in **Listing 23**.

The ternary operation in the middle of the reduce operation is there to avoid putting a comma in front of the first Person serialized to JSON. Some JSON parsers might accept this format, but that is not guaranteed, and it looks ugly to have it there.

It is ugly enough, in fact, to fix. The code is actually a lot easier to write if we use the built-in Collector interface and its partner Collectors, which specifically do this kind of mutable-reduction operation (see **Listing 24**). This has the added benefit of being much faster than the versions using the explicit reduce and String::concat from the earlier examples, so it's generally a better bet.

Oh, and lest we forget our old friend Comparator, note that Stream also has an operation to sort a stream in-flight, so the sorted JSON representation of the Person list looks like **Listing 25**.

BE A COLLECTOR

It is ugly enough to fix. **The code is actually a lot easier to write if we use the built-in Collector interface** and its partner Collectors, which specifically do this kind of mutable-reduction operation.

LISTING 21 LISTING 22 / LISTING 23 / LISTING 24 / LISTING 25 / LISTING 26

```
String xml =
 "<people data='lastname'>" +
 people.stream()
     .map(it -> "<person>" + it.getLastName() +
           "</person>")
     .reduce("", String::concat)
 + "</people>";
System.out.println(xml);
```

Download all listings in this issue as text

This is powerful stuff.

Parallelization. What's even more powerful is that these operations are entirely independent of the logic necessary to pull each object through the Stream and act on each one, which means that the traditional for loop will break down when attempting to iterate, map, or reduce a large collection by breaking the collection into segments that will each be processed by a separate thread.

The Stream API, however, already has that covered, making the XML or JSON map() and reduce() operations shown earlier a slightly different operation—instead of calling stream() to obtain a Stream from the collection, use parallelStream() instead, as demonstrated in **Listing 26**.

For a collection of at least a dozen items, at least on my laptop, two threads are used to process the collection: the thread named main, which is the traditional one used to invoke the main() method of a Java class, and another thread named ForkJoinPool.commonPool worker-1, which is obviously not of our creation.

Obviously, for a collection of a dozen items, this would be hideously unnecessary, but for several hundred or more, this would be the difference between "good enough" and "needs to go faster." Without these new methods and approaches, you would be staring at some significant code and algorithmic study. With them, you can write parallelized code literally by adding eight keystrokes

(nine if you count the Shift key required to capitalize the *s* in *stream*) to the previously sequential processing.

And, where necessary, a parallel Stream can be brought back to a sequential one by calling—you can probably guess—sequential() on it.

The important thing to note is that regardless of whether the processing is better done sequentially or in parallel, the same Stream interface is used for both. The sequential or parallel implementation becomes entirely an implementation detail, which is exactly where we want it to be when working on code that focuses on business needs (and value); we don't want to focus on the low-level details of firing up threads in thread pools and synchronizing across them.

BE READY

The release of Java SE 8 swiftly approaches. With it come not only the new linguistic lambda expressions (also called closures or anonymous methods)—along with some supporting language features—but also API and library enhancements that will make parts of the traditional Java core libraries easier to use.

Conclusion

Lambdas will bring a lot of change to Java, both in terms of how Java code will be written and how it will be designed. Some of these changes are already taking place within the Java SE libraries, and they will slowly make their way through many other libraries—both those owned by the Java platform and those out in "the wilds" of open source—as developers grow more comfortable with the abilities (and drawbacks) of lambdas.

Numerous other changes are present within the Java SE 8 release. But if you understand how lambdas on collections work, you will have a strong advantage when thinking about how to leverage lambdas within your own designs and code, and you can create better-decoupled code for years to come. `</article>`

LEARN MORE

- Lambda Expressions

Web applications in pure Java?

For over a decade, the web and internet have been the platform for all kinds of applications – personal and business. The biggest benefit of a web application has been the easiness to reach the audience's desktops without software installations, with a single deployment.

But at the same time we have been pushing the limits of browser technologies, most importantly HTML and JavaScript, to create even better user interfaces that meet the expectations.

The key elements in the architecture of a web based system are of course the browser and the application server.

There has been a long lasting fight over which side the applications should run on. In the browser you can have better responsiveness, interactivity and visuality for the

"Vaadin introduces a familiar user interface architecture that is based on Java classes, components and widgets."

user, but server-side environment is much easier to manage – mostly because of solid component and service oriented architectures like Java EE. But what we want to have is best of both worlds.

Vaadin is an open-source web application architecture that brings more balance to this server-client fight or dilemma. Vaadin introduces a familiar user interface architecture that is based on Java classes, components and widgets. It is very quick to learn.

The key feature is that widgets functionally span from server to client. Think of a 3D animated coverflow widget where the data is programmatically and securely bound in the server, but we have these fancy scroll effects using the latest browser technology. All this seamlessly in a single logical component.

Heavily based on object oriented Java language and dynamic loading and management of runtime components in Java EE, Vaadin brings you the reusability, expressiveness and programmatic easiness you already saw in old Java Swing applications. This time for pure web applications. In pure Java language.

LEARN MORE

- JavaOne 2013 booth #5106
- The Vaadin open source project: vaadin.com
- Live demo: vaadin.com/demo

CLOUD: CHALLENGE AND OPPORTUNITY

Cameron Purdy on cloud development

BY TIMOTHY BENEKE

ART BY I-HUA CHEN; PHOTOGRAPHY BY BOB ADLER

Oracle Vice President of Development Cameron Purdy (left) and Architect Rob Lee talk about Oracle Coherence features.

ENTERPRISE CLOUD COMPUTING

According to Cameron Purdy, vice president of development at Oracle, the advent of cloud computing has led to a new wave of software innovation comparable to the surge in web-oriented technologies 15 years ago or the rise of business applications and the Java EE platform some 10 years ago. The potential cost savings and enhanced productivity offered by cloud computing present Java developers with new challenges and opportunities.

To meet these challenges, Java developers and architects will need to change the way they think. Purdy, who has worked with Java technology since 1996, is in a unique position to address these changes. Prior to joining Oracle in 2007, he was the CEO of Tangosol, whose Coherence

Left to right: Team members Lee, Nilesh Junnarkar, and Dhiraj Mutreja discuss clustering and cloud services with Purdy.

Data Grid product offered reliable and scalable data management across the enterprise. He regularly participates in industry standards development and is a specification lead for the Java Community Process (JCP). He has five times been named a JavaOne Rock Star for the quality of his JavaOne presentations, and he was recognized in TheServerSide's "Who's Who in the Enterprise Java World."

Java Magazine *talked with Purdy to hear his latest thinking on cloud computing, Java, and Oracle's new cloud foundation products.*

Java Magazine: Why should Java developers pay attention to cloud computing?

Purdy: First, it's important to appreciate the wide scope of the term *cloud computing*. Obviously, we talk about the public cloud, including Amazon EC2 or Microsoft Azure, not to mention Oracle's own public cloud. But cloud computing is also changing the way that we manage internal data centers—what we refer to as the *private cloud*. In other words, the very same concepts, efficiencies, and capabilities that the public cloud brought us are now available inside the data center—for example, basic tenets such as self-service and being able to get a development, staging, or production environment for an application in minutes—without going through any significant paperwork or human workflow. On top of that, the elasticity of the cloud means that we're no longer constrained by a one-size-fits-all model.

Java Magazine: What do Java developers need to understand about this?

Purdy: First, applications must be built so that they can dynamically scale out and dynamically balance the load across multiple servers. This means that when servers are added while an application is running, there's no interruption of service. To accomplish this, it's good programming practice to always assume that an application or the components of an application are running in a scaled environment, which means many previous assumptions no longer apply. I'm talking about things such as file systems—which could be either local or shared and could trip up developers either way—or global Java objects on the heap, which won't actually be "global" when the application is scaled out.

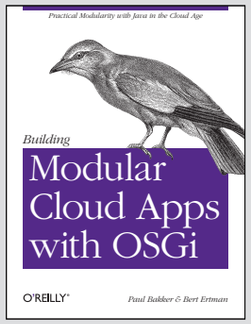
ELASTIC CLOUD

We're no longer constrained by a one-size-fits-all model. Applications must be built so that they can dynamically scale out and dynamically balance the load across multiple servers.

Developers must also understand whether the addition of resources will actually enable an application to scale out effectively. In other words, just because an application runs correctly when you add additional servers, that doesn't mean it will actually support more transactions or users. So, you need to explicitly design into the scale-out model the ability for the application to scale as close to linearly as possible. Obviously, data caching is a huge part of this, as is minimizing the amount of information that has to be shared across servers, as is minimizing contention on shared resources such as database systems.

Finally, the application needs to be understood in terms of the metrics of elasticity: when are additional resources actually needed, and when is it safe for resources to be taken away? So, you need to know *when* you need to scale out the application, which involves knowing what to monitor.

How Developers Are Using the Cloud



We talked with Bert Ertman, Java Champion and Netherlands JUG leader, about his experience as a developer with the cloud.

Java Magazine: How are you using the cloud?

Ertman: I'm using the cloud as a development and deployment platform for realizing sophisticated software-as-a-service [SaaS] offerings for my customers.

Java Magazine: What are the benefits?

Ertman: The cloud enables pay-as-you-go, both for development and testing and also for running costs. Using a stateless architecture and leveraging the advantages of a modular application architecture, we are able to achieve horizontal scalability and auto-scaling. This way, we pay only for what we truly need instead of wasting vast amounts of money on hardware and data centers.

Java Magazine: How has the cloud changed your job?

Ertman: As a developer, you have to be aware of the underlying infrastructural mechanics. From this perspective, DevOps is an interesting trend. Also, the cloud challenges a lot of the traditional development stack: relational database versus NoSQL, proprietary interfaces versus open APIs using RESTful services, stateful versus stateless, transactional versus eventual consistent, server-side web frameworks versus HTML5 and MVC JavaScript, and so on. Building cloud applications is more than just building a web app and deploying it on virtual hardware.

Java Magazine: Are you willing to share any of your hard-won knowledge about building apps for the cloud?

Ertman: Absolutely! My coworker Paul Bakker and I have just finished writing a book for O'Reilly called *Building Modular Cloud Applications with OSGi*. The book is a practical guide to applying the concepts of modularity to the tools and technologies that come as part of the cloud development stack. The book leverages what we have learned so far in the past couple of years. Right now the book is in production, and we hope to have it available just ahead of JavaOne San Francisco.

You can find Bert speaking at JavaOne San Francisco and at the J-Fall conference in the Netherlands, and online @BertErtman.

—Tori Wieldt, Senior Java Community Manager, Oracle

Java Magazine: How does Java EE 7 enable these kinds of scaling?

Purdy: Several improvements in Java EE 7 are relevant. Obviously, the new batch specification, JSR 352, is tremendously valuable, and not only for cloud computing. It's able to take significant amounts of work that might otherwise have to be done synchronously and break it down into smaller units and perform those asynchronously. In addition, the Java EE role definitions have been updated to better map to the security requirements we see in cloud environments, such as the difference between someone administering platform as a service [PaaS], on which the application is running, versus administering the application itself that is being hosted.

Software also works better in a cloud environment when we get rid of certain assumptions. One way to do this is through dependency injection, which was introduced in Java EE 5 and, as of Java EE 7, now applies across the entire Java set of enterprise specifications and is called Context and Dependency Injection [CDI]. Instead of going out and finding what you need as a component, you simply declare what you need. If you need to connect to a database, the component declares that it needs to be connected to a database and then it allows whatever environment it's running in to provide it with a connection to that database based on how that environment is configured.

Java Magazine: Could you clarify infrastructure as a service [IaaS] and PaaS with respect to Java EE?

Purdy: IaaS manages the requisite networking and server hardware, and it typically hosts virtual machines, each of which is running an operating system—all of which is below the level of Java EE. We imagine that every data center, every private cloud, and every public cloud will be providing IaaS: infrastructure services that platforms and applications can take advantage of through standardized IaaS APIs.

Generally speaking, what developers are looking for is PaaS. Developers need to be able to describe an application hosting environment: an application server or a cluster of application servers, which are often combined with a database. And they need to have their application exposed to the real world, protected by a firewall, and automatically balanced by a load balancer.

In other words, they're not interested in building or installing their own firewall, load balancer, or database. They want to be able to take an application, including necessary components and their database design, and deploy it in a way that is

Purdy chats with *Java Magazine* Editor in Chief Caroline Kvitka during the photo shoot.

accessible and secure. So, think of an application that you're deploying for a mobile platform; it's probably going to consist of a significant number of RESTful interfaces that are exposed to a mobile platform. Those RESTful interfaces will be exposed through a URL that will come to a load balancer that spreads requests across an elastically scaled application server infrastructure, providing both high availability and elastic scalability. The application servers will be running the application logic, which will be transforming information from a database or other data services into RESTful responses that are sent back to the mobile application itself. All of the complexity of provisioning, configuring, monitoring, and scaling the application infrastructure is provided and managed by PaaS.

Java Magazine: And that's something that Oracle offers?

Purdy: Yes, both with the investments we're making in the Java EE platform itself, as well as through the Oracle Java Cloud Service that's based on Oracle WebLogic Server. This service allows developers to specify what size of environment they want for their application, to automatically hook up their application with a database and define that database for their application, to deploy their application, and so on.

Java Magazine: How does PaaS affect the roles and functions in an organization, such as development, quality assurance, operational staging, production operations, and so forth?

Purdy: PaaS significantly reduces the cost of provisioning and managing those environments; the trade-off is that it tends to reduce the number of options available in terms of the flexibility of an environment. Advances such as PaaS tend to follow the same 80/20 rule that we see with IaaS: if you can cover 80 percent of your infrastructure requests with an IaaS offering, then by automating all of that, you reduce your infrastructure operational cost potentially on the order of 80 percent.

Similarly, if you can reduce the complexity of your application hosting environment by providing a PaaS offering that provides a relatively simple set of options, then perhaps 80 percent of your applications can take advantage of that. This means that you're never going to cover 100 percent, but it does free up the resources from those 80 percent to focus on the difficult 20 percent. Knowing that there's a constrained environment, you can drastically reduce the number of variables that you have to focus on. For many simple applications, PaaS significantly simplifies the application and improves the time to market.

CLOUD FOUNDATION

Oracle Cloud Application Foundation provides all the necessary building blocks for a PaaS environment, starting with intelligent and scalable load-balancing capabilities.

Java Magazine: HTML5 is a central focus of Java EE 7. How does it relate to cloud computing?

Purdy: Oracle has tried to make the Java EE 7 platform the ideal development platform for applications targeting mobile clients, smart devices,

Purdy fuels up for the day with coffee and a banana.

and tablets. And part of that is certainly its HTML5 support capability. And with many of these applications, time to market—being able to build and deploy applications quickly and at low cost, and being able to scale those applications out to handle growth—is essential. Many of the applications that we've been talking about target mobile platforms, often by providing RESTful services. And these are all areas that are targeted by Java EE 7: support for HTML5, RESTful web services, WebSockets, and JSON. These applications are typically exposing RESTful services that return JSON data, and Java EE 7 has great support for building and parsing JSON data.

Java EE 7 has much better support for providing RESTful services as well, including support for asynchronous RESTful services.

It seems that every generation of software expands its complexity to the point where it's no longer manageable. The reason the cloud is emerging as a dominant technology movement is because we need automation of management and simplification of infrastructure and platform just to manage the complexity that is inherent in today's applications. Similarly, HTML5 is needed in order to provide the level of interactivity and the richness of applications that people have been trying to handcraft and also to work around the limitations and complexity of supporting various proprietary client technologies.

Java Magazine: How does Oracle WebLogic Server 12.1.2 fit into the cloud?

Purdy: Oracle WebLogic Server 12.1.2, which is part of Oracle Cloud Application Foundation, contains a number of exciting new capabilities, such as support for RESTful services and HTML5 WebSockets. One of the features we're most excited about is what we call *dynamic clusters*. Dynamic clusters allow an application to dynamically add servers on the fly. This used to have to be manually preconfigured and now it's automated, which offers a very elastic environment for the applications we've been discussing. Also, Oracle WebLogic Server 12.1.2 supports the new Oracle Database 12*c* release, including its multitenancy option, which is a significant capability for developing cloud applications.

Java Magazine: What is *multitenancy*, and why is it important?

Purdy: Assuming that an application is providing software as a service [SaaS], multitenancy means that each organization that's purchasing that service has its own secure set of data. Typically, it would be very expensive from an infrastructure or management point of view to give each tenant its own database, but a true multitenanted database capability is actually built into Oracle Database 12*c*. Oracle WebLogic Server works very closely with the database to take advantage of that capability, including taking advantage of what's called *Database Resident Connection Pooling*

IT'S COMPLICATED

We need automation of management and simplification of infrastructure and platform just to manage the complexity that is inherent in today's applications.

These are exciting times for Java developers, says Purdy. “Java is the ideal platform for building the back-end services required by the new wave of cloud applications.”

[DRCP], which is a server-side connection-pooling capability that significantly reduces the number of physical connections necessary between a cluster of application servers and a database cluster.

Java Magazine: What is Oracle Cloud Application Foundation?

Purdy: Oracle Cloud Application Foundation provides all the necessary building blocks for a PaaS environment, starting with intelligent and scalable load-balancing capabilities—which are likely running in the demilitarized zone (DMZ) of the network—and securely routing requests to Oracle WebLogic Server and the application tier. Oracle WebLogic Server is an extremely robust and mature application server, and it is tightly integrated with the Oracle Coherence data grid and in-memory caching software. Oracle Cloud Application Foundation also includes Oracle Traffic Director, a hardware-accelerated routing and load-balancing solution.

All of these components share a common installation, configuration, provisioning, management, and lifecycle model, and a common management framework that is surfaced in a single pane of glass by Oracle Enterprise Manager. We have developer support for the latest standards and the latest version of Oracle WebLogic Server coming from Oracle JDeveloper, Oracle Application Development Framework, Oracle Enterprise Pack for Eclipse, and NetBeans.

DON’T ASSUME

Software also works better in a cloud environment when we get rid of certain assumptions. One way to do this is through dependency injection.

Java Magazine: Any closing remarks you might have for Java developers?

Purdy: It’s an exciting time to be a Java developer. Java is the ideal platform for building the back-end services required by the new wave of cloud applications, just as Java EE originally was a revolution that enabled applications for the web. There are more than a billion people in the world who now carry an application platform in their pocket—that’s something no one could have predicted 15 years ago. I don’t know exactly where we will be 15 years from today, but it will happen fast and we’d better get to work! `</article>`

Timothy Beneke is a freelance writer and editor. His interviews, which cover a wide range of topics, have appeared in *Mother Jones*, the *East Bay Express*, and the *Chicago Reader*.

LEARN MORE

- Oracle Cloud Services
- Oracle WebLogic Server
- Cameron Purdy’s blog

HARSHAD OAK

BIO

Part 1

Hands On with Oracle Java Cloud Service

An introduction to Oracle's platform-as-a-service Java offering

Java EE has been the primary software platform for enterprise and server-side development for more than a decade, and it is increasingly the platform of choice on the cloud. In this article, we will look at the Java cloud space and how you can go about choosing a Java platform-as-a-service (PaaS) provider, and then take a closer look at Oracle Java Cloud Service.

Only a few years back, when someone discussed a Java EE project, it was presumed that the project would also require setting up the requisite hardware infrastructure and having a team to manage and monitor the setup. Java EE was never great at shared hosting, but nobody seemed to care much about that, because shared hosting was almost considered below the dignity of Java EE. It used to be blasphemous for an architect to suggest that a Java EE application could be run in a shared environment.

How things have changed!

The cloud wave turned this approach on its head. Not only is a shared environment now being considered, it is actually fashionable to be talking about running an application in a shared environment on the cloud.

Java as yet has no cloud-centric specifications in place, so the Java cloud space isn't as standardized as developers might expect. Yet we find that Java is being used in all kinds of cloud deployments, especially PaaS offerings and software-as-a-service (SaaS) solutions built with Java. In this article, we will focus on Java PaaS.

IN FASHION

It is actually fashionable to be talking about running an application on the cloud.

Java PaaS

PaaS is about renting a software platform and running a custom business application on it. The promise of PaaS is to let developers focus on the business application and not have to worry about the hardware or the core software platform.

Gartner famously stated that 2011 will be the year of PaaS. However, my many unscientific surveys at conferences in 2011 and 2012 showed that while there was great interest and experimentation happening with PaaS, actual adoption was pretty low. This was due to multiple factors, such as PaaS offerings not being mature enough, developer uncertainty, and managers being unwilling to change to a new shared model.

Having said that, the various PaaS offerings have matured rapidly over the past year or so. Many now provide services comparable to what developers have been used to in on-premises Java EE environments. Now, some vendors don't just provide a deployment environment; they even provide a rich development environment on the cloud. There's healthy competition building up in this space, which should be good news for developers and customers. So while Java PaaS might not yet be the norm, adoption

looks certain to keep rising at an ever-increasing pace.

Key Considerations

There's a fair bit of overlap between the features that available cloud services offer, but the key points to consider from a software development platform point of view are as follows:

- Costs and pricing strategies vary widely across vendors. Some charge based on fine-grained usage details, while others provide duration-based subscriptions. You need to evaluate whether you would like to go with a subscription or with a pay-as-you-go model.
- Are the supported technologies and features in line with your requirements? Is your chosen framework officially supported by the cloud vendor? Which version is supported?
- Is the vendor sticking to standard technologies, or would you need to write custom, vendor-specific code?
- Considering that you are putting your precious application and data on the vendor's hardware, you want to be sure about the vendor's credentials and ability to be up and running, say, 10 years from now.
- With PaaS, you do not have access to the actual hardware setup or micro details about hardware performance. So you need to evaluate the administration dashboard carefully, because it's the primary source of information about the service and about how an application is performing.
- Is the PaaS solution integrated with your favorite Java integrated development environments (IDEs)?
- Ease of use is quite important because some cloud services can be rather confusing and, at times, even intimidating. I found this especially true with services that support not just Java but many other technologies as well.
- Most cloud vendors support at least one SQL data store and, in some cases, a NoSQL data store as well. You need to examine whether these work for you.
- Is the vendor offering a closed stack that would lock you in? Would it be possible for you to migrate to a new vendor if the need arises?
- While some vendors are focused Java cloud players, there are others that support multiple technologies. This seems to affect the features, the documentation, the ease of use, and the overall priority areas for the service. Some clouds offer Java support, but they just don't come across like they are talking about Java.
- How difficult would it be to build a team capable of developing and deploying for a particular PaaS?

Oracle Cloud

Oracle's cloud push began in 2011, and since then Oracle has launched several cloud solutions that support more than 25 million cloud users worldwide. Oracle Java Cloud Service and Oracle Database Cloud Service have been Oracle's most visible PaaS solutions so far. Oracle's other PaaS offerings are Oracle Developer Cloud Service, Oracle Storage Cloud Service, and Oracle Messaging Cloud Service.

Oracle Developer Cloud Service simplifies development with an automatically provisioned development platform that supports the complete development lifecycle. Oracle Storage Cloud Service enables businesses to store and manage digital content in the cloud. Oracle Messaging Cloud Service provides an infrastructure that enables communication between software components by sending and receiving messages via a single messaging API, establishing a dynamic, automated business workflow environment.

GO-TO PLATFORM

Java EE has been the primary software platform for enterprise and server-side development for more than a decade, and it is increasingly the platform of choice even on the cloud.

Oracle Java Cloud Service

A simplistic explanation of Oracle Java Cloud Service is that it's Oracle WebLogic Server integrated with Oracle Database. So developing and deploying on Oracle Java Cloud Service is akin to developing and deploying on Oracle WebLogic Server and using Oracle Database for persistence. Oracle Java Cloud Service runs Oracle WebLogic Server Release 10.3.6, which is the latest version in the Oracle WebLogic Server 11*g* line. Oracle Java Cloud Service drastically reduces the complexity associated with the deployment and maintenance of enterprise Java applications.

Oracle Java Cloud Service supports a mix of Java EE 5, Java EE 6, and Oracle WebLogic Server capabilities. It supports all the commonly used Java technologies

such as Servlets, JSP, JavaServer Faces (JSF), Enterprise JavaBeans (EJB) Java Persistence API (JPA), JAX-RS, JAX-WS, and more. However, while Servlet version 2.5 (Java EE 5) is supported, there's support for JSF 2 (Java EE 6). Also, note that today Oracle Java Cloud Service supports Java 6 APIs but not Java 7 APIs. So Oracle Java Cloud Service is more of a Java EE mix-and-match and not the same as having a full Java EE 5 or Java EE 6 server on the cloud.

Beyond the standard Java EE specifications, Oracle Java Cloud Service supports the deployment of applications that make use of Oracle WebLogic Server–specific extensions as well as Oracle Application Development Framework constructs.

Standards-based. One of the highlights of Oracle Java Cloud Service is that, unlike some other Java PaaS vendors, it puts great emphasis on being a standards-based solution. So Oracle Java Cloud Service does not force users to use any proprietary APIs. You can develop and deploy on Oracle Java Cloud Service while sticking purely to the relevant Java EE specification.

TIMES HAVE CHANGED

Only a few years back, **when someone discussed a Java EE project, it was presumed** that the project would also require setting up the requisite hardware infrastructure and having a team to manage and monitor the setup.

Ease of use. Oracle Java Cloud Service is easy to get started with for anyone with a Java EE background who is familiar with deploying applications on an application server. Considering that it is a Java-only cloud setup, there's also nothing that would seem strange or confusing to a Java EE developer. I also like the fact that, unlike with some vendors, Oracle Java Cloud Service does not have its own jargon for how it is priced and packaged.

SDK and IDEs. The Oracle Java Cloud Service Software Development Kit (SDK) provides tools to help you develop, deploy, and manage your applications. Note that this SDK is not meant to provide classes and libraries that you have to use in your applications. It merely consists of tools and plug-ins that you can choose to use or not use. Oracle Java Cloud Service offers rich integration with popular Java IDEs, such as NetBeans, Eclipse, and Oracle JDeveloper, all of which leverage the SDK "under the hood" to interact with Oracle Java Cloud Service. The SDK also includes Ant tasks and Maven plug-ins for interacting with your Oracle Java Cloud Service instances.

Whitelist. While Oracle Java Cloud Service runs Oracle WebLogic Server, the cloud setup is not exactly the same as a local Oracle WebLogic Server instance. For technical and security reasons, a small number of specific APIs is prevented from executing in Oracle Java Cloud Service. So there's a whitelist of technologies that fulfill the technical and security requirements of Oracle Java Cloud Service and have been approved to run on it. Using an API that's not on the whitelist triggers a whitelist violation and stops your application from being deployed.

Other Oracle PaaS services. Oracle Java Cloud Service works well with other cloud services from Oracle. Oracle Database Cloud Service is, as of today, a must-have for Oracle Java Cloud Service. Apart from the database, there's Oracle Developer Cloud Service, which provides source control management, issue tracking, continuous integration, and wiki collaboration. There's also Oracle Storage Cloud Service and Oracle Messaging Cloud Service.

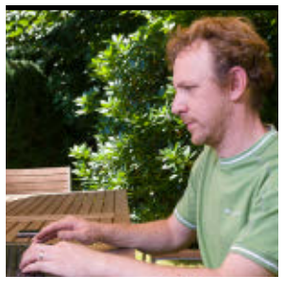
Trial. Oracle provides a free 30-day trial of Oracle Java Cloud Service. The trial comes with a feature-rich setup that supports unlimited users and applications, 1.5 GB of RAM for the Java heap, adequate file storage, and 5 GB of data transfer. The Oracle Java Cloud Service trial includes a trial of Oracle Database Cloud Service with one schema, 1 GB of Oracle Database storage, and 6 GB of data transfer.

Conclusion

In this article, we looked at the Java PaaS space today and what goes into selecting a Java PaaS service. We then took a quick look at Oracle's Java PaaS solution—Oracle Java Cloud Service—and its capabilities. In the next article, we will delve deeper into Oracle Java Cloud Service by developing and deploying a Java EE application on it. `</article>`

LEARN MORE

- Oracle Java Cloud Service documentation
- Oracle Cloud downloads



JOHAN VOS

BIO

Part 1

Building Rich Client Applications with JSR 356: Java API for WebSocket

Build a simple chat application using the JSR 356 server API and an HTML5 client.

The Java EE 7 specification adds new functionality to the Java EE specification. One capability many developers were asking for is support for WebSockets. Real-time bidirectional traffic on the internet is growing. Different actors are involved in this area, including content providers, broadcasters, software developers, and telecom operators. Due to the different characteristics of the involved actors, the success of WebSockets required standardization.

The Internet Engineering Task Force (IETF) defined a standard for the WebSocket protocol. This standard defines the low-level protocol that technologies implementing WebSockets should adhere to. For example, it defines how an HTTP connection should be upgraded to a full-duplex bidirectional WebSocket connection. The importance of this standard cannot be underestimated. Different languages and platforms are used to develop applications relying on WebSockets, and they should all rely on the same protocol in order to be interoperable.

On top of the WebSocket protocol, a number of technologies and implementations exist facilitating the use of WebSockets in a specific language or platform. For example, the W3C has a working draft describing how to leverage WebSockets from a web page, and ever since Java EE 7, a similar specification has existed for dealing with WebSockets in Java. This specification is defined in JSR 356 and was approved to be part of the Java EE 7 specification.

The Java API for WebSocket contains a server API and a client API. Containers that claim to be Java EE 7–compliant implement the server API. The only real difference between a server container and a client container is that a server container provides the infrastructure for registering WebSocket endpoints—it will listen for incoming requests on specific endpoints. The client API is, therefore, a subset of the server API.

The Reference Implementation for JSR 356, named Tyrus, is included in GlassFish 4 and contains client modules as well as server modules. The client modules can be used in any Java application and allow developers to connect to any WebSocket endpoint, as long as the endpoint adheres to the IETF 6455 standard. On top of the client modules, the server modules allow applications to register endpoints that will handle all WebSocket communications, as long as the other peer (the client) adheres to the IETF 6455 standard.

A Simple Chat Application

We will now build a very simple chat application. The server component is developed using the JSR 356 server API, and it is deployed in a GlassFish 4 container.

In this article, we create an HTML5 client. In Part 2, we will create a Java-based client, leveraging the JSR 356 client API. Both the server and the two clients use the same underlying protocol, so they work together seamlessly.

Note: The source code for the examples described in this

PHOTOGRAPH BY TON HENDRIKS

article can be downloaded here.

It should be noted that this demo application is by no means a professional chat application. The goal is to explain some of the core aspects of WebSocket support in Java, rather than to create a secure, robust, and feature-complete chat environment.

The server component. The server component of the chat application is very lightweight. A list of active chatters is maintained, but apart from that, no state is preserved. The server component listens for incoming messages on WebSockets, and sends messages to the client over the same WebSockets. The client implementations create a WebSocket connection to the server and exchange messages. The WebSocket specification is very open regarding the type of content that can be transmitted. For this demo application, we use the JSON format for encoding messages, because JSON is widely supported in Java and JavaScript.

Our demo application supports the following scenario:

- An end user visits a web page or starts a Java application.
- The web page or application opens a WebSocket connection to the server.
- The server immediately sends a list of active chatters to the client.
- Simultaneously, the end user is allowed to enter his nickname.
- The nickname is sent to the server.
- The server sends a login message with the nickname to all active clients.
- The clients process the login message and add the new user to the list of active chatters.
- The end user can then enter a new chat message, which is sent to the server.
- The server distributes the new chat message to all active clients.
- When the end user leaves the chat, the client WebSocket connection to the server is closed.
- The server detects the closed connection, and sends a logout message to the remaining clients.
- The clients process the logout message and remove the specific user from the list of active chatters.

The server component contains the following parts:

- ChatEndpoint is the class that contains the logic implementing the lifecycle methods related to the WebSocket operations.
- ChatCommand and its subclasses contain specific commands and information that are exchanged between the server and the clients.
- ChatDecoderEncoder is a utility class that translates ChatCommand instances to JSON strings and vice versa.

ChatCommand and its subclasses define the application protocol between server and clients. All ChatCommand instances have a field named command that defines the type of the command (login, logout, allusers, or message). The clients will send and receive JSON representations of these commands and can, thus, also distinguish between the different types of commands.

The ChatEndpoint class is annotated with the @ServerEndpoint annotation as shown in **Listing 1**.

We declare three elements in the @ServerEndpoint annotation:

- value="/endpoint" specifies that we want to listen for incoming WebSocket connections at the URL specified by the context root of this application plus "/endpoint". When a client connects a WebSocket to this address, an instance of the ChatEndpoint class will be able to react to lifecycle events (for example, @OnOpen, @OnMessage, @OnClose, and @OnError).
- decoders=ChatDecoderEncoder.class specifies that the ChatDecoderEncoder utility class will be used for decoding incoming text messages into ChatCommand instances.
- encoders=ChatDecoderEncoder.class specifies that the ChatDecoderEncoder utility class will be used for encoding ChatCommand instances into (JSON) text messages.

We want to maintain a list of active users. For this simple application, we will use the static Map shown in **Listing 2**. Each entry in the sessionUsers map contains a WebSocket session and the name of the user associated with the specific WebSocket session.

When a client establishes a WebSocket connection to the URL specified in the value element of the @ServerEndpoint annota-

LISTING 1 LISTING 2

```
@ServerEndpoint(value="/endpoint", decoders=ChatDecoderEncoder.
class, encoders=ChatDecoderEncoder.class)
public class ChatEndpoint {
...
}
```

Download all listings in this issue as text

tion, the method annotated with @OnOpen will be called. This method might contain a number of parameters: the Session instance, an EndpointConfig parameter, and a number of PathParam parameters corresponding to path information. In our example, we need to know only the Session instance, which is created by the underlying implementation and which uniquely identifies the associated WebSocket (including the client peer that initiated the connection).

Our implementation of the @OnOpen method has two goals:

- Send the list of active chatters to the client.
- Register the new WebSocket session with our application. Until we know the name of the user who will chat, we associate an anonymous user with this session.

These goals are achieved using the code shown in **Listing 3**.

The list of active chatters is obtained by taking the values of the sessionUsers map. We create an instance of the AllUsersCommand class and populate it with the list of active chatters. The instance is then sent to the remote peer using the Session.getBasicRemote() .sendObject(...) method.

Because the ServerEndpoint is configured with the ChatDecoderEncoder class as the encoder for messages, the AllUsersCommand instance is converted to a String object. The WebSocket implementation will, therefore, call ChatDecoderEncoder .encode(ChatCommand object), which returns a String object. The resulting String object is then sent over the open WebSocket to the client, where it can be parsed.

The conversion between the instances of ChatCommand and JSON strings is achieved using JSR 353, the JSON API for Java EE.

The implementation of this method is beyond the scope of this article, and can be found in **Listing 4**.

When a user wants to participate in the chat, the client sends a login message containing the nickname of the user. We assume that the client sends a JSON string containing the required information.

When our ServerEndpoint receives a message, the method annotated with @OnMessage is called. In our demo application, this method has the signature shown in **Listing 5**. The method annotated with @OnMessage can have a number of optional parameters, as explained in the Javadoc for @OnMessage.

In our case, the method will be called with an instance of ChatCommand and the Session parameter that uniquely defines the WebSocket session. The WebSocket implementation will create the ChatCommand instance by calling the ChatEncoderDecoder .encode(String) method, converting the JSON string sent by the client peer of the WebSocket into an instance of ChatCommand.

This is a very convenient approach, because it allows a clean separation between the content of the message (contained in the ChatCommand class) and the transport protocol (JSON). The ChatEndpoint instance does not need to be aware of the transport protocol. By changing the encoders

LISTING 3 LISTING 4 / LISTING 5

```
@OnOpen
  public void onOpen (Session s) {
    AllUsersCommand auc = new AllUsersCommand();
    auc.setUids(sessionUsers.values());
    try {
      s.getBasicRemote().sendObject(auc);
    } catch (IOException | EncodeException ex) {
      Logger.getLogger(ChatEndpoint.class.getName()).log(Level.SE-
VERE, null, ex);
    }
    sessionUsers.put(s, "anonymous");
  }
```

Download all listings in this issue as text

element in the @ServerEndpoint annotation, a different transport protocol can be used.

The implementation of our method annotated with @OnMessage will do the following:

- In case the supplied ChatCommand is a LoginCommand, the nickname of the user will replace the "anonymous" value that initially was associated with the WebSocket session.
- The message will be multicasted to all active chatters.

These goals are achieved with the code shown in **Listing 6**.

Note that we send the message to all active chatters by iterating over the keySet of the sessionUsers map. This requires that we keep track of connections that are broken and users who are leaving.

We assume that the clients will close the WebSocket connection when the active user is leaving (either by closing the application or by pressing a logout button). Good enough; the Java WebSocket API has the @OnClose annotation, which can be used to indicate that a certain method should be called when a WebSocket session is closing. The underlying WebSocket implementation will make sure that the method annotated with @OnClose will be called if a WebSocket connection is closed (either intentionally or because of network issues).

STAY TUNED

In this article, we create an HTML5 client. In Part 2, we will create a Java-based client, leveraging the JSR 356 client API. The server and the two clients use the same underlying protocol, so they work together seamlessly.

In our case, the @OnClose annotated method will send a logout command to all other sessions and subsequently remove this session from the list of active chatters. This behavior is obtained using the code in **Listing 7**.

Note: We could also use the convenient session.getOpenSessions() method to obtain a list of active sessions.

HTML5 client. A very simple version of an HTML5 client that connects to the server component described above is shown in the file in the source code called client/html5/chat.html.

When a user enters the web page, a WebSocket to the server URL will be created. We define a body onLoad handler that contains the required functionality to create a WebSocket:

```
<body onLoad="openSocket();">
...
</body>
```

The openSocket() function is defined in **Listing 8**.

The WebSocket interface in **Listing 8** is defined as part of the W3C draft of the WebSocket API, and it is available in most modern browsers. The WebSocket interface allows developers to provide the following lifecycle callback functions:

- onopen
- onmessage
- onerror
- onclose

Note that these functions are conceptually similar to the lifecycle annotations in the Java API for WebSocket (@OnOpen, @OnMessage, @OnError, and

LISTING 6 LISTING 7 / LISTING 8

```
@OnMessage
public void onMessage (ChatCommand cmd, Session s) throws IOException, EncodeException {
  if (cmd instanceof LoginCommand) {
    String uid = ((LoginCommand)cmd).getUid();
    sessionUsers.put(s, uid); // this will overwrite the "anonymous" user
  }
  for (Session session : sessionUsers.keySet()) {
    session.getBasicRemote().sendObject(cmd);
  }
}
```

Download all listings in this issue as text

@OnClose). This makes the development of an end-to-end, bidirectional WebSocket-based application much easier.

Because our server component will immediately send a list of active chatters upon opening a WebSocket connection, the .onmessage handler on the connection object will be called. In this handler, we receive the message coming from the server in JSON format, which can easily be processed in JavaScript.

The code in **Listing 9** first detects the type of the incoming message. This can be achieved by inspecting the command field of the incoming JSON data. The value of this field corresponds to the value of the ChatCommand .getCommand() method in the server component.

If the incoming message is an allusers command, the function will populate an HTML element with the nicknames of the active users (see **Listing 10**).

The AllUsersCommand contains a field named uids, which contains the list of active chatters. We can easily obtain this list in JavaScript (allusers = x.uids) and iterate over the active users. For each user, we create a new HTML paragraph and we set the innerHTML of that paragraph to the user ID (nickname) of the user.

We also add a unique identifier to the element, because we should be able to retrieve and remove the element when that specific user leaves the chat. In that case, the client will receive a logout command from the server. The incoming message is processed as shown in **Listing 11**.

As a consequence, the name of the user who left is no longer shown. Note that in a more powerful chat application, users who have left the chat could be shown in a different style.

When a new user enters the chat while our client is active, we will be notified via a login command sent by the server. In that case, we retrieve the nickname (or user ID) for the user, and add it to the chatters element using the code in **Listing 12**.

WEBSOCKET SPEC

The W3C has a working draft describing **how to leverage WebSockets from a web page**, and ever since Java EE 7, a similar specification has existed for dealing with WebSockets in Java. This specification is defined in JSR 356.

LISTING 9 LISTING 10 / LISTING 11 / LISTING 12 / LISTING 13 / LISTING 14

```
var x = JSON.parse(evt.data);
command = x.command;
if (command == "allusers") {
        ...
}
if (command == "login") {
        ...
}if (command == "logout") {
        ...
}if (command == "message") {
        ...
}
```

Download all listings in this issue as text

The web page contains a text field for entering the user's nickname and a button for joining the chat. Clicking the button will call the loginUser() function, which will send a login message to the server. Also, the list of active chatters (which was already retrieved when starting the WebSocket connection) will be shown, as well as a chat box containing the messages. All of this is achieved with the function shown in **Listing 13**.

The code in **Listing 13** shows how easy it is to send a message over a WebSocket connection to a server component. Using the JSON.stringify function, a JSON message is created and sent to the server, where the JSON message is decoded into a Java object (an instance of ChatCommand).

When the user enters a chat message, the talk function is called, and the content of the message (contained in a text field named msg) is sent to the server, as shown in **Listing 14**.

When a user sends a message, all clients will receive a message command. Our client will add the content of that message to the messages HTML element. **Listing 15** shows how this is achieved.

While this example is far from complete, it shows how an HTML5 client can easily communicate with a Java back end and exchange

LISTING 15

```
if (command == "message") {
    otheruid = x.uid;
    mytext = x.text;
    var chld = document.createElement("p");
    chld.innerHTML = mytext;
    var messages = document.getElementById("messages");
    messages.appendChild(chld);
}
```

Download all listings in this issue as text

messages over a WebSocket. Both the HTML5 client and the Java back end are based on the same protocol, and they share the same lifecycle concepts.

Conclusion

In this article, we built a very simple chat application, making use of an HTML5 client. The Java WebSocket API and the HTML5 WebSocket specification are very much aligned with each other. This makes it easy to build decoupled WebSocket applications using different technologies (Java or JavaScript) on the server and on the client. The lifecycle events are clearly defined, and allow for a wide range of application-specific protocols (such as our chat protocol). This is yet more proof that Java EE is the best server-side technology for managing HTML5-based applications.

Stay tuned for Part 2, in which we will create a Java-based client, leveraging the JSR 356 client API.

`</article>`

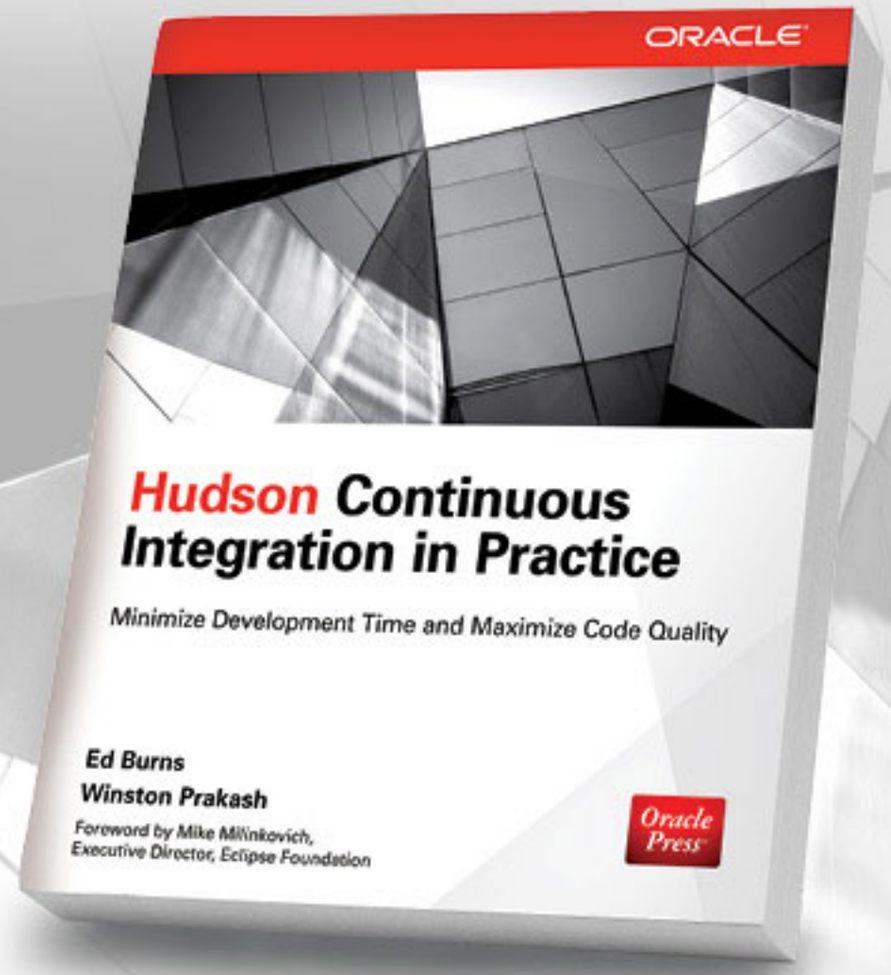
LEARN MORE

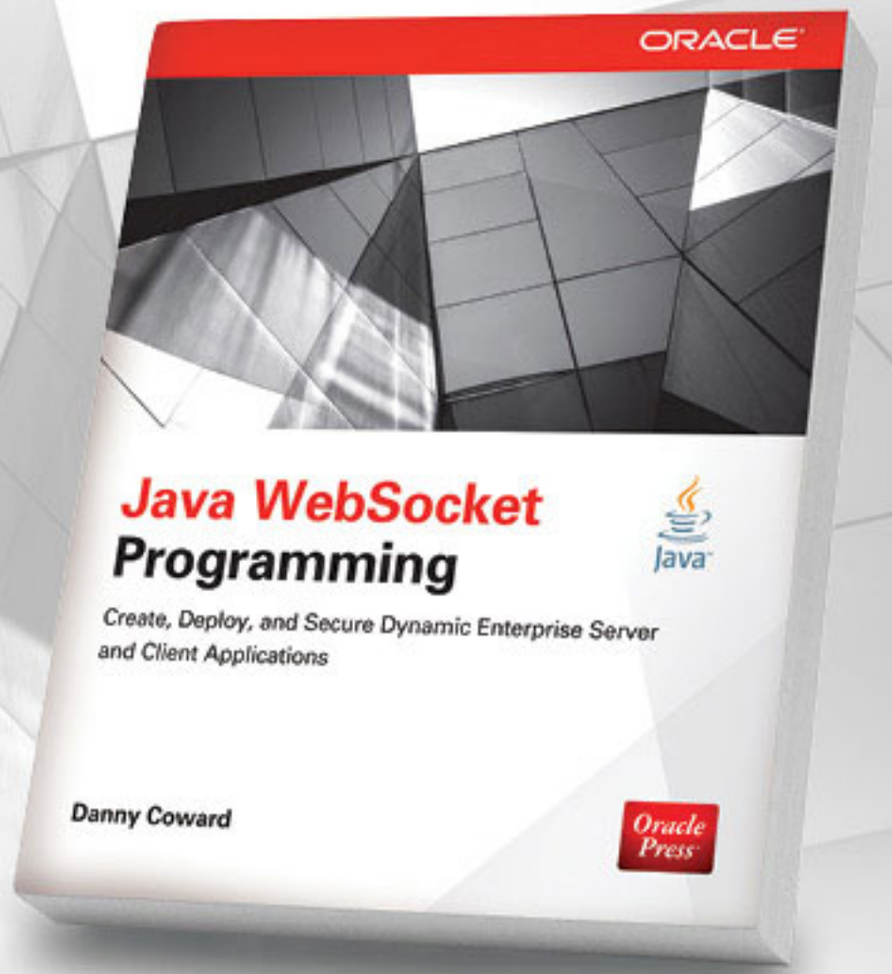
- JSR 356, Java API for WebSocket
- Johan Vos' blog

Oracle Press™

Your Destination for Java Expertise

Written by leading technology professionals, Oracle Press books offer the most definitive, complete, and up-to-date coverage of Java available.





Hudson Continuous Integration in Practice
Ed Burns and Winston Prakash

Streamline each process in your development lifecycle to optimize productivity while reducing risk and complexity.

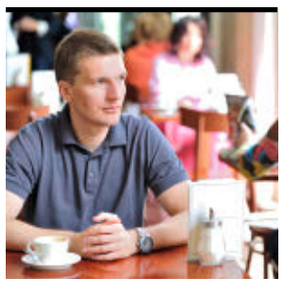
Java WebSocket Programming
Danny Coward

The top expert on Java WebSocket programming shows how to build dynamic enterprise Web applications that fully leverage state-of-the-art communication technologies.

Oracle Press™

Available in print and eBook formats

Visit us at OraclePressBooks.com • Follow us @OraclePress and Like us OraclePress



ADAM BIEN

BIO

Part 2

JavaFX Data Binding for Enterprise Applications

Learn about the data flow in the JavaFX API.

User interface (UI) components expose the state of the business logic to the user. A user's input changes the state of the UI components first, but at some point, the state of the UI components needs to be synchronized with the underlying business logic, which is often implemented as plain old Java domain objects. This article describes the data flow between the presentation and business logic realized with the JavaFX API.

Note: The previous article in this series described how to integrate the JavaFX Scene Builder output into multiview enterprise applications.

What Is Data Binding?

Data binding is the process of automatic data synchronization between objects. Data binding can be declarative (for example, the value binding in JavaServer Faces [JSF] is declarative) or programmatic, as discussed in this article.

JavaFX comes with different data binding granularities. A high-level, coarse-grained, fluent API allows chaining of calls, which results in complex expressions. You can implement string concatenations as well as complex computations, and you can evaluate the expression in lazy or eager fashion. The lower-level API directly connects two properties and synchronizes the state between them without any further processing.

No Properties, No Binding

Java SE does not support data binding at the language level. You cannot just connect two fields of a class and expect a magical synchronization behind the scenes. JavaFX comes with classes, comparable to JavaBeans, that provide such magic. You can even use the JavaFX properties without launching a UI, as shown in **Listing 1**.

Binding also works bidirectionally; changes to one part of the property will be reflected in the other parts (see **Listing 2**).

Direct data binding works only between a javafx.beans.property.Property and a javafx.beans.value.ObservableValue. To connect a JavaFX property with a JavaBean, you will have to translate the state changes into a series of method invocations. This can be easily achieved with a ChangeListener, as shown in **Listing 3**.

Any custom implementation of ObservableValue could be used as a binding target. To demonstrate this, we will use a simple timer called Pomodoro (see **Listing 4**), which fires at a fixed schedule and passes the current time to the listeners. The Pomodoro technique is sometimes used by programmers to

FAST FACT

With data binding, you can either directly connect disparate parts of the application or just listen to the changes and translate the events into higher-level method calls.

PHOTOGRAPH BY THOMAS EINBERGER/ GETTY IMAGES

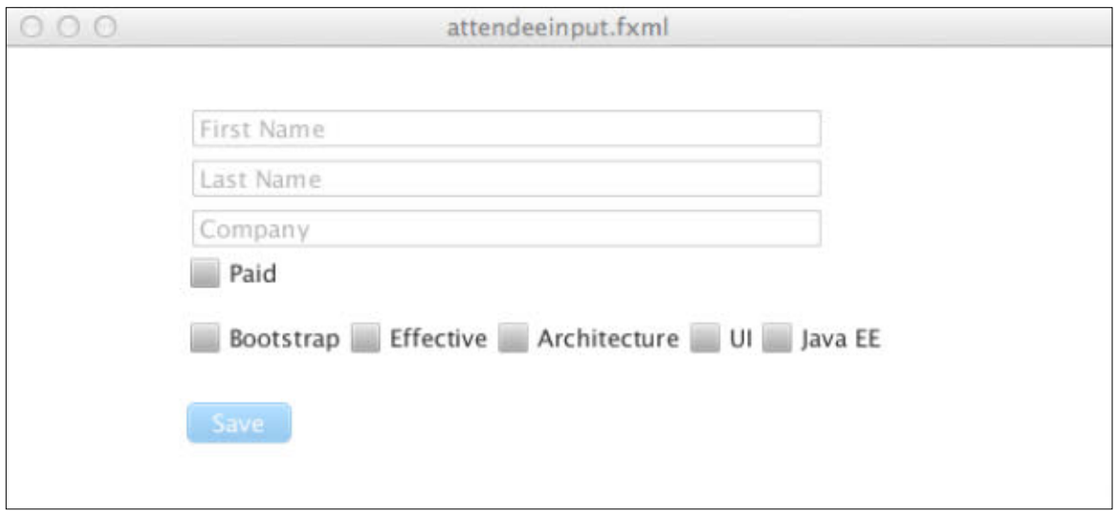


Figure 1

sustain long documentation writing periods.

Pomodoro extends the abstract ObjectBinding class, which significantly simplifies the implementation. Only the method computeValue needs to be implemented. All the ChangeListener bookkeeping is already implemented by the ObjectBinding.

Every second, the internal timer fires (in the real world, the length of the period is 25 minutes with a five-minute break) and invokes the invalidate method. Behind the scenes, all registered listeners and observers are notified and synchronized. The code in **Listing 5** demonstrates the usage.

All JavaFX UI components also expose a set of bindable properties, which can be used for direct synchronization with presenters and models.

A Binding DSL

JavaFX offers a higher-level binding API, which allows you to perform computations on the fly. Instead of just synchronizing the plain values of two bound objects, additional computations can be performed in real time, as shown in **Listing 6**.

The higher-level API supports not only numerical operations but also String and Boolean properties, as shown in **Listing 7**. String binding is particularly useful for the implementation of bindable "live templates" within the application, such as status bars or labels for UI components.

Validating the Input

The binding DSL is particularly useful for input validation. Imagine an input view that allows you to save only coherent data (see **Figure 1**). The First Name, Last Name, and Company fields need to be filled in and one of the workshop name checkboxes has to be selected in order to make the Save button active. Traditionally, such validation was implemented with a series of if-else statements, sometimes factored out into a Mediator. **Listings 8a** and **8b** show

LISTING 1 | LISTING 2 | LISTING 3 | LISTING 4 | LISTING 5 | LISTING 6

```
import javafx.beans.property.SimpleStringProperty;
import javafx.beans.property.StringProperty;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.is;
import org.junit.Test;

  @Test
  public void bindString() {
    StringProperty input = new SimpleStringProperty();
    StringProperty output = new SimpleStringProperty();
    output.bind(input);
    input.set("duke");
    assertThat(output.get(), is("duke"));
  }
```

Download all listings in this issue as text

an example of how to do the input validation using the binding DSL.

The relevant UI elements are injected first by the javafx.fxml.FXMLLoader. In the method initialize, the underlying StringProperty and BooleanProperty instances are connected together to compute the activation state of the Save button. The validation comprises two parts: the BooleanProperty nameEntered reflects the state of the name input completion, and the BooleanProperty dayChosen reflects a logical OR of the checkboxes.

Finally, both properties are chained together to eventually compute the activation state of the Save button. The code is almost fluently readable: the Save button is not disabled when a day is chosen and a name is entered. Plain Java code would result in far more plumbing: you would have to register a "focus-lost listener" to each of the UI components and perform the validation logic with a series of if-else statements.

JPA Integration with Adapter Binding

Java Persistence API (JPA) entities are POJOs with a few JPA annotations containing additional metadata. However, JPA is not aware of JavaFX properties and uses simple Java types as backing properties. The JPA runtime (the class javax.persistence.EntityManager) is going to either call the fields directly or use getters and setters to synchronize the state between the entity and the database, as shown in **Listings 9a** and **9b**.

Instead of directly exposing the fields as persistent properties, you can use JavaBean accessors as an interface to the JPA persistence. With JavaBean accessors as persistent properties, the state can be internally maintained in JavaFX properties and exposed as plain Java properties. With this approach, the internal state is exposed as ordinary accessors and JavaFX properties at the same time.

Direct binding to JPA entities is an attractive way to store offline data. Each state change of the UI elements is automatically synchronized with the JPA entity. Because the entity is managed by the EntityManager, you only have to commit the transaction in order to flush the changes to the database. With direct binding, you also get undo-like functionality for free. To update the UI, you only have to call the EntityManager::refresh method and pass the entity. The state of the JPA entity will be overridden with the database data. The EntityManager will use the ordinary accessors to pass the data, which

LISTING 7 LISTING 8a / LISTING 8b / LISTING 9a / LISTING 9b

```
@Test
  public void stringOperations() {
    final String message = "hey, duke";
    StringProperty content = new SimpleStringProperty();
    StringProperty label = new SimpleStringProperty(
"Message: ");
    StringExpression result = label.concat(content);
    content.set(message);
    assertThat(result.get(), is(label.get() + message));
  }
```

Download all listings in this issue as text

will change the internal state of the JavaFX properties. Any other bound UI component will be immediately updated.

WHAT A SURPRISE

Data binding **has surprising effects** on application design. With JavaFX presenters, capabilities can be cleanly exposed as properties.

Binding for Narrow Interfaces

Data binding has surprising effects on application design. Before the advent of JavaFX properties, view and presenter capabilities were exposed in the best-case scenario as an interface. Sometimes the view and the controller were connected with a series of arbitrary references. With JavaFX presenters, capabilities can be cleanly exposed as properties without any additional plumbing.

The application-level presenter AirhacksPresenter is interested in the currently selected Attendee domain object in an overview panel. This dependency manifests as a single selectedAttendee Property method, as shown in **Listings 10a** and **10b**.

The ObjectProperty selected Attendee is changed by some other presenter and contains the currently active domain object. The AttendeeInputPresenter subscribes itself as a listener to this property in the initialize method and reacts to the changes.

On the other side, the newAttendeeProperty is the outbound communication channel. Every time a new attendee is created by the user, the declaratively bound save method is executed and the state of the newAttendeeProperty is changed. Any other interested presenter or view can react to the changes without any further coupling or knowledge.

In fact, the application-level presenter, AirhacksPresenter, wires all the independent presenters together in the initialization phase, as shown in **Listing 11**.

The state of the selectedAttendee Property did not require any further transformations or filtering, so it was directly bound. Additional filtering is performed before passing the value in the anonymous ChangeListener implementation, so the AttendeeInputPresenter and the WorkshopPresenter were glued together with a ChangeListener. Without the null check, you could also directly bind both properties.

Similarly, the WorkshopPresenter consumes the selectedAttendee Property and offers the new AttendeeProperty to others at the same time. The WorkshopPresenter also acts as a dispatcher and coordinates subordinate DayPresenter instances with corresponding views. There is still no direct coupling required between the WorkshopPresenter and the AttendeeInputPresenter (see **Listings 12a** and **12b**).

LISTING 10a / LISTING 10b / LISTING 11 / LISTING 12a / LISTING 12b

```
package com.airhacks.control.presentation.attendeeinput;44444
import
com.airhacks.control.business.registrations.entity.Attendee;
import javafx.beans.property.SimpleBooleanProperty;
import javafx.beans.property.SimpleObjectProperty;
import javafx.beans.property.ObjectProperty;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.fxml.Initializable;

public class AttendeeInputPresenter implements Initializable {
  private ObjectProperty<Attendee> selectedAttendee;
  private ObjectProperty<Attendee> newAttendee;

  @Override
  public void initialize(URL url, ResourceBundle rb) {
    this.selectedAttendee = new SimpleObjectProperty<>();
    this.newAttendee = new SimpleObjectProperty<>();
    this.selectedAttendee.addListener(
new ChangeListener<Attendee>() {
      @Override
      public void changed(
ObservableValue<? extends Attendee> observable,
      Attendee oldValue, Attendee newValue) {
                        //
      }
    });
  }
```

Download all listings in this issue as text

Bindable Collections + Tables = Good Friends

JavaFX tables are directly bindable to a javafx.collections.ObservableList containing domain objects. Any state changes of the backing list are directly reflected in the view. The ObservableList is the actual model of the TableView. Each entry is a row; the columns are represented by individual fields of an underlying domain object, as shown in **Listings 13a**, **13b**, and **13c**.

A presenter doesn't have to directly interact with a TableView—it suffices to manipulate the underlying ObservableList. This gives us a true decoupling; the TableView becomes replaceable with any UI component accepting an ObservableList as a model. Also, the same list can be exposed to other presenters for communication purposes. Such pragmatic exposure of an underlying model is perfectly viable for simple views exposing their UI components to the presenter.

For an interpresenter communication, a more specific interface prevents the exposure of unintended implementation details. In **Listings 13a**, **13b**, and **13c**, only the specific actions, such as deletedAttendeeProperty, selectedAttendeeProperty, and editingStarted, are exposed as ObjectProperty or BooleanProperty instances to other presenters. At any time, the internal ObservableList together with the corresponding TableView UI component can be easily replaced without affecting collaborating presenters and views.

Also the DayPresenter plays a classic presenter role and translates generic events into higher-level "created," "deleted," and "edited" events and makes the interface more usable and maintainable at the same time.

Binding for Unit Testing

The presenters communicate with each other only through JavaFX properties; hence, there is no direct coupling to concrete presenter implementations. Also, all UI components are never exposed to other presenters. This fact simplifies unit testing. You have to mock out only the input and output properties and register test-specific listeners to validate the output for more-complex presentation logic. Because the UI elements are

FAST FACT

JavaFX properties streamline communication between components and make additional plumbing superfluous.

LISTING 13a LISTING 13b / LISTING 13c

```
public class DayPresenter implements Initializable {
  @FXML
  private TableView<Attendee> attendeesTable;
  private ObservableList<Attendee> attendees;
  private BooleanProperty editingStarted;
  private ObjectProperty<Attendee> deletedAttendee;

  public void prepareTable() {
    this.attendeesTable.setEditable(true);
    ObservableList columns = attendeesTable.getColumns();
    final TableColumn firstNameColumn =
createTextColumn("firstName", "First Name");
    firstNameColumn.setOnEditCommit(
new EventHandler<CellEditEvent<Attendee, String>>() {
      @Override
      public void handle(
CellEditEvent<Attendee, String> t) {
        getEditedAttendee(t).setFirstName(
t.getNewValue());
        editingStarted.set(false);
      }
    });

    columns.add(firstNameColumn);
    attendeesTable.setItems(this.attendees);
        //...
  }
```

Download all listings in this issue as text

hidden, you don't have to open the UI, which allows the use of Continuous Integration (CI) environments, such as Jenkins.

Conclusion

At first glance, JavaFX is "only" the next generation of the Swing toolkit. However, Inversion of Control (IoC), Dependency Injection (DI), and data binding have a huge impact on an application's architecture. With them, WYSIWYG editors, such as the JavaFX Scene Builder, become truly usable without any further workarounds.

JavaFX properties streamline communication between components and make additional plumbing, such as the introduction of artificial interfaces, superfluous. With data binding, you can either directly connect disparate parts of the application or just listen to the changes and translate the events into higher-level method calls.

Not covered in this article is the also-supported lazy evaluation of binding expressions. Lazy evaluation makes binding more efficient by evaluating the whole expression on demand, not upon each change of the bound components. In the context of the sample application described in this article, you could implement the workshop price computation eagerly or lazily using JavaFX binding on the fly.

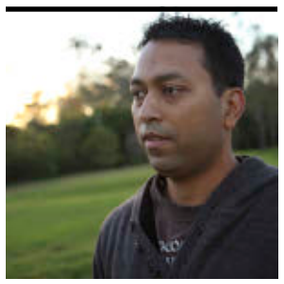
JavaFX is shipped with JDK 1.8, so JavaFX properties can be considered a part of the JDK now. This opens the door for JavaFX properties to the server side. `</article>`

IN REAL TIME

JavaFX offers a higher-level binding API, which allows you to perform computations on the fly. Instead of just synchronizing the plain values of two bound objects, additional computations can be performed in real time.

LEARN MORE

- Mediator pattern
- AirHacks Control sample application
- Airpad sample application
- Follome.fx



Enter a Brave New World—Embedded Java at Your Fingertips

Learn the basics about embedded Java.

VIKRAM GOYAL

BIO

*Listen to author **Vikram Goyal** describe the topics covered in this article on embedded Java.*

So you're a regular Java developer. You have done years of service developing apps on the server side and get your kicks with any number of server- and client-side APIs. You might have even developed some MIDlets for the Java ME environment. You think you've done it all.

But suddenly, everyone seems to be talking about a new technology: embedded Java. Can Java really go so small? You thought that the era of resource-constrained programming was over. With the advances in the architecture of Android, iPhone, and Java ME devices, lack of memory was no longer an issue. Suddenly, lack of resources is a reason to celebrate.

As you might have guessed, we are not talking about consumer devices (at least, not direct-to-consumer devices). We are talking about devices such as the Raspberry Pi and other microcontrollers with which you can manipulate circuit boards and small resource systems. Such microdevices allow you to manipulate and work directly with the onboard circuitry.

What Is Embedded Java?

Embedded versions of Java use the same Java technology that you work with now—except the embedded versions are bite size. For example, Oracle Java ME Embedded has a footprint smaller than that of Java ME, and it is targeted at devices that power set-top boxes, vending machines, sensors, or, well, microcontrollers.

Java can be defined and adapted to different devices using either Oracle Java ME Embedded or Oracle Java SE Embedded. In this article, we look at how to do that and we look at how Oracle Java ME Embedded technologies are adapted to the embedded environment.

BITE-SIZE JAVA

Embedded versions of Java use the same Java technology that you work with now—except the embedded versions are bite size.

Oracle Java ME Embedded is defined by the Information Module Profile-Next Generation (IMP-NG) specification (JSR 228). (There is a separate specification for Oracle Java SE Embedded.) As you might have guessed, this JSR is an extension of the really old JSR 195, which was—not surprisingly—called the IMP specification. That JSR never got off the ground, but adding the next-generation bit has done wonders. Or it could be that the time is right for the new specification.

An Information Module Profile is a strict subset of the Mobile Information Device Profile (MIDP), which you are probably well acquainted with. So, if we were creating MIDlets using MIDP, we must define a new name for the applications that we create with IMP-NG.

IMlets = Baby MIDlets

Anything that you create using IMP-NG is called an *IMlet*. In a way, an IMlet is

PHOTOGRAPH BY JONATHAN WOOD/ GETTY IMAGES

really a MIDlet that doesn't have access to everything that a MIDlet has access to. It still extends the javax.microedition.midlet.MIDlet class, and it still has the same life-cycle that a MIDlet has. So you still implement the startApp(), pauseApp(), and destroyApp() methods.

Similar to the Java technology for the Java apps you have been making, embedded Java technology based on Oracle Java ME Embedded requires you to create your IMlet within a development environment such as NetBeans. The close connection between MIDlets and IMlets is emphasized by the fact that the Oracle Java ME Embedded environment is distributed with the Java ME SDK. As of this writing, Java ME SDK 3.3 preview edition is available, and it can be downloaded here.

GO BIG OR SMALL

Java is well positioned to be in the thick of the Internet of Things. The opportunities are now endless with embedded Java appearing in all devices, big or small.

Install Oracle Java ME Embedded as you would install a Java ME SDK in your favorite development environment, and you are ready to start developing your IMlets. The process is so similar to developing a MIDlet that I won't even bother showing you how to create your Hello World IMlet.

After creating your Hello World IMlet, you need to install it in an actual device to see it in action. This is where things get a little tricky and slightly different from the MIDlet environment.

Unlike MIDlets, IMlets are targeted at a plethora of devices, not just at mobile phones. You can develop an IMlet for a vending machine, a set-top box, a microcontroller, a network card, a router, a tracking device—anything that meets the following minimum hardware specifications:

- Minimum of 128 KB of nonvolatile memory (for IMP), 8 KB for persistent data, and 128 KB for the Java runtime.
- Two-way, wireless networking with limited bandwidth.
- Optional UI and sound capabilities; a UI, if present, must *not* be the MIDP UI.

To test your IMlet on a real device, you need a device that supports a kernel, a mechanism for installing the Java runtime (unless the device comes preloaded with the Java runtime), and a way to provide persistent storage. Further, this device must be able to manage the application lifecycle. A good entry-level device that allows you to do all this is called the Raspberry Pi, and it is as cheap as chips (US$25). You can order one here.

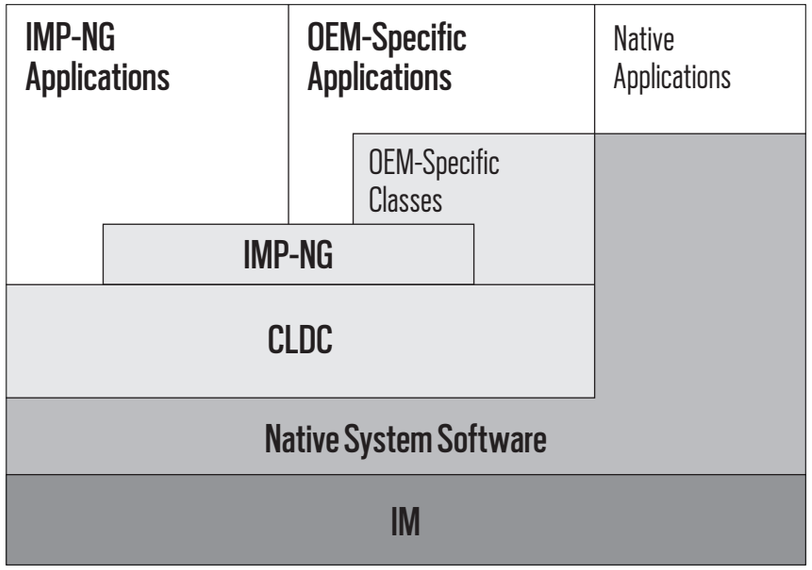


Figure 1

Once you have a device, download the Java runtime for it. At the moment, Oracle provides the Java runtime for two devices: Raspberry Pi and ARM Cortex-M3/M4. The runtime for both devices can be downloaded here. (See Angela Caicedo's blog for a great tutorial on coding for the ARM Cortex-M3/M4.)

Finally, with your software and hardware in place, you can follow these steps to create useful IMlets.

Different device manufacturers might decide to provide their own versions of the runtime, which you're free to use. The beauty of an embedded Java environment is that you can create your binary for one platform and port it to another without any modifications. This is truly portable code—write once, run everywhere.

How Does It All Fit Together?

Figure 1 is a high-level architectural diagram borrowed from the IMP-NG specification. The Information Module (IM) in the above figure represents the hardware. Your IMlet sits in the top left corner. It talks to the IMP-NG profile, which in turn sits on top of the Connected Limited Device Configuration (CLDC) for that environment. This is where the Java abstraction ends and the native system software (or the kernel) environment takes over.

Of course, microcontrollers and

other embedded devices might not rely on Java as the sole development environment. This is made clear within the architecture. IMlets can coexist with other embedded platforms, provided there is space for that. It is highly unlikely that there will be multiple runtime environments on the same device due to the constraints.

Programming Specifications

The programming specifications are minimal. If you go with the strict IMlet API specification (IMP-NG), then you are just dealing with a subset of the MIDlet API.

The API packages are javax.microedition.midlet, javax.microedition.rms, javax.microedition.io, javax.microedition.pki, javax.microedition.media, javax.microedition.media.control, java.lang, and java.util.

In addition to these packages, each runtime that is provided includes targeted APIs for the device in question. These are called the optional APIs, and if you program with these, your IMlet will probably run only on a particular device. Doing this increases capability but reduces portability, which might or might not be a good thing.

YOU'RE READY

Embedded Java—through both Oracle Java ME Embedded and Oracle Java SE Embedded—is a technology that average Java developers can use to create sophisticated systems using their existing knowledge.

Conclusion

Java is well positioned to be in the thick of the Internet of Things. The opportunities are now endless with embedded Java appearing in all devices, big or small. Machine-to-machine systems and automated systems can now be programmed to a sophisticated degree with portable code. Embedded Java—through both Oracle Java ME Embedded and Oracle Java SE Embedded—is a technology that average Java developers can use to create sophisticated systems using their existing knowledge. `</article>`

LEARN MORE

- Oracle Java ME Embedded resource page
- Terrence Barr's blog

//fix this /

In the July/August issue, Simon Ritter gave us a code challenge around JavaFX. He presented us with a JavaFX application that tries to place a button on each side of the divider in a SplitPane. The correct answer is #3. Each button needs to be placed in its own Pane (StackPane, FlowPane, or any other subclasses would also work).

Now when the application is run, the divider is in the center of the window, with the buttons positioned in the top left corner of each half of the SplitPane.

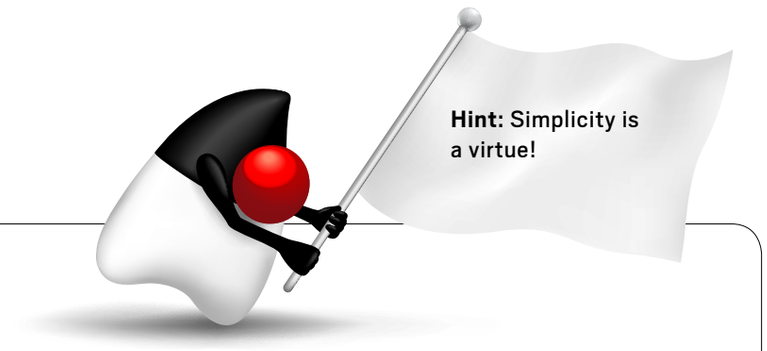
This issue's challenge comes from Attila Balazs, a polyglot developer from Cluj-Napoca, Romania, who presents us with a class initialization challenge.

1 THE PROBLEM

A developer in your group struggles with the following code. Most of the time it deadlocks, and sometimes it throws a NullPointerException but it never runs to the end correctly. How can you fix the problem?

2 THE CODE

```
final class App {
 public static final Integer RETRIES = 3;
 private static final App instance = new App(Controller
.getInstance());
 private Controller ctrl;

 public App(Controller controller) { this.ctrl = controller; }
 public void run() { ctrl.run(); }
 public static App getInstance() { return instance; }
}

final class Controller {
 private static final Controller instance = new Controller(App
.RETRIES);
 private final Integer retries;

 private Controller(Integer retries) { this.retries = retries; }
 public static Controller getInstance() { return instance; }
 public void run() { }
 public Integer getRetries() { return retries; }
}

//...

// some thread starts running the app
new Thread(new Runnable() {
 public void run() { App.getInstance().run(); }
}, "AppThread").start();

// meanwhile, on the main thread
Controller.getInstance().getRetries();
```



3 WHAT'S THE FIX?

1) Use int instead of Integer for RETRIES.
2) Use jstack to find the deadlock.
3) Move RETRIES into Controller.
4) Use -XX:CompileThreshold=0 to inline App.RETRIES

GOT THE ANSWER?

Look for the answer in the next issue. Or submit your own code challenge!

ART BY I-HUA CHEN