//NOVEMBER/DECEMBER 2013 /

Java™ magazine

By and for the Java community

SMARTEST HOUSE ON THE STREET

The Internet of Things comes home with Java-based openHAB

+ Development Best Practices

31 DIABOLICAL PERFORMANCE TUNING

40 DEVOPS FOR DEVELOPERS

ORACLE.COM/JAVAMAGAZINE

ORACLE®

//table of contents /

COVER ART BY WES ROWELL; PHOTOGRAPHY BY TON HENDRIKS

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, Arianna Pucherelli
Production Designers
Sheila Brennan, Kathy Cygnarowicz

PUBLISHING
Vice President
Jeff Spicer
Publisher
Jennifer Hamilton +1.650.506.3794
Associate Publisher and Audience Development Director
Karin Kinnear +1.650.506.1985

ADVERTISING SALES
President, Sprocket Media
Kyle Walkenhorst +1.323.340.8585
Western and Central US, LAD, and Canada, Sprocket Media
Tom Cometa +1.510.339.2403
Eastern US and EMEA/APAC, Sprocket Media
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 (US/Canada)
Oracle Services
+1.888.283.0591 (US)
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 GTxcel

Futuristic, automated homes are becoming a reality. I grew up watching *The Jetsons*, a cartoon about a space-age family living in 2062 with robotic housekeepers, wacky inventions, and personal spacecraft. Now, 50 years since it first aired, some of the show's predictions are looking closer to reality.

If you attended JavaOne in San Francisco, you couldn't miss the buzz about the Internet of Things (IoT)—with predictions of a world where billions of devices will be connected to the internet. It's here, and Java is playing a big part. You were also sure to see robots at the event, whether they were NAO robots, robotic fish, or Lego bots. There was also much buzz about the Raspberry Pi, which had its own developer challenge.

Home automation is a prime example of IoT. In this issue's "Smartest House on the Street," we take you inside a modern home where devices and applications are integrated throughout, using a Java-based software environment called openHAB (a 2013 Duke's Choice Award winner).

Need to get up to speed on Java and IoT? Read Eric Bruno's "Internet of Things 101" to see why Java, which was born on an embedded device, is in a great position to become the standard for IoT development.

No matter what type of project you're working on, we know that you need to be the best developer you can be. So in this issue we bring you a special section focused on development best practices. Look for the icon on the left to find articles covering agile development, performance tuning, concurrency development, DevOps, and team dynamics.

What futuristic projects are you dreaming up? 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

MAKE THE FUTURE JAVA

Java™

FIND YOUR JUG HERE

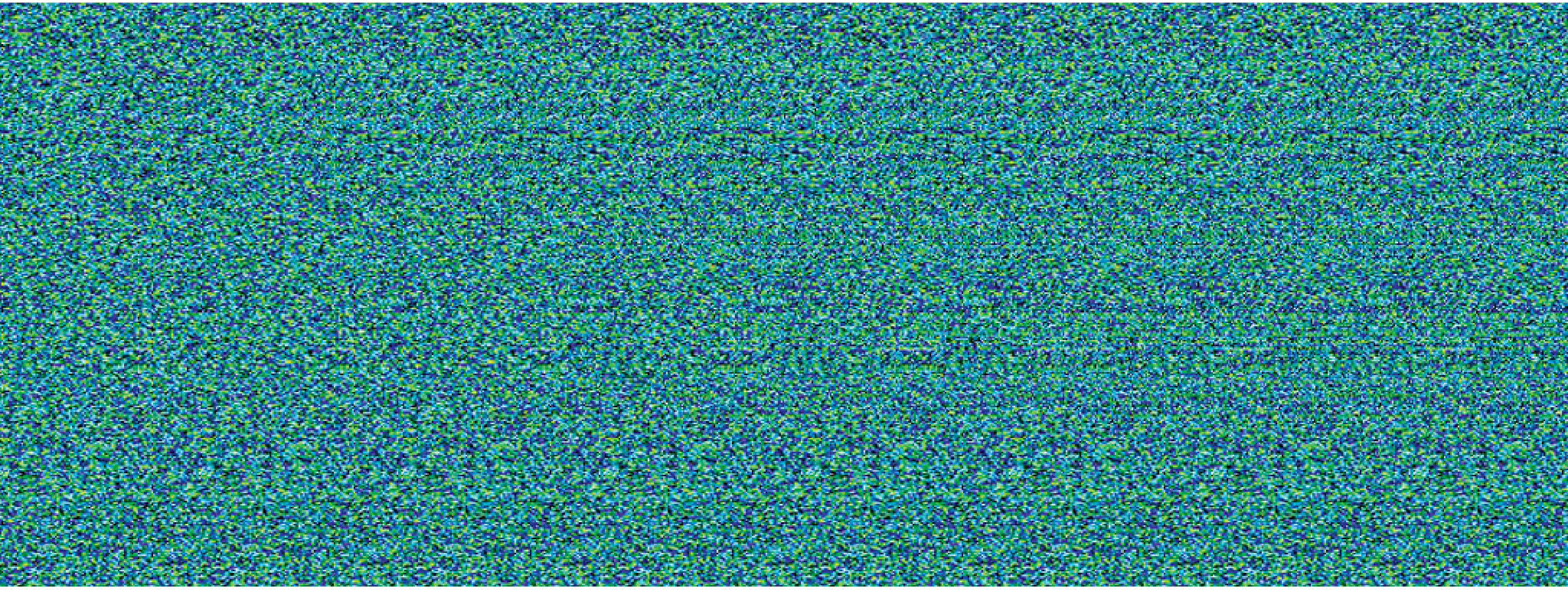
My local and global JUGs are great places to network both for knowledge and work. My global JUG introduces me to Java developers all over the world.

Régina ten Bruggencate
JDuchess

LEARN MORE

ORACLE

The answer is right in front of you



Java Image Enabling SDKs that Help You See the Big Picture

At first glance it may seem difficult, but it's really quite simple. Atalasoft's JoltImage product is a proven SDK for image enabling your Java-based web applications, easily. Image enabling helps to add dimension to your data, so you can uncover insights such as correlations and causations hidden inside your 2-dimensional documents. Our SDK does the heavy lifting for you, saving time, money, and the headaches of figuring it out yourself. Backed by our highly knowledgeable & caffeinated support engineers, JoltImage will enable your success and make the big picture so much easier to see.

Atalasoft JoltImage

Click for tips on viewing the stereogram

Image enabling experts & bacon connoisseurs. Visit us online to see our full line of SDK products for .NET and Java

Atalasoft A Kofax Company
www.atalasoft.com

Clockwise from top: A festive atmosphere at the Taylor Street Café; Peter Utzschneider talks about the Internet of Things; Nandini Ramani discusses convergence.

PHOTOGRAPHS BY ORANGE PHOTOGRAPHY

The 18th JAVAONE

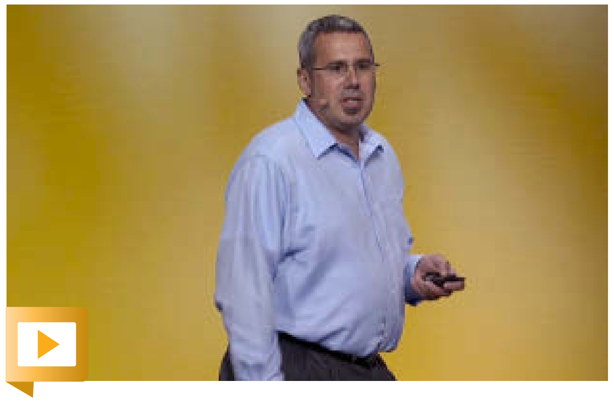
Attendees from more than 92 countries attended JavaOne 2013, where they chose from more than 400 sessions; hung out in a Codegarten; participated in a Raspberry Pi coding challenge; and attended educational and networking events. What's more, the America's Cup sailing race added to the festive atmosphere of JavaOne—and culminated with a historic comeback win by ORACLE TEAM USA.

The conference kicked off at San Francisco's Moscone Center with the Strategy keynote by Oracle's **Peter Utzschneider** and **Nandini Ramani**. The theme, "Make the Future Java," was unchanged from last year's theme, for good reason, Utzschneider said. "There is a lot going on in the industry, with massive shifts and innovation happening that pose huge challenges and opportunities for Java." The goal is to make Java better, stronger, more robust, and relevant for decades to come.

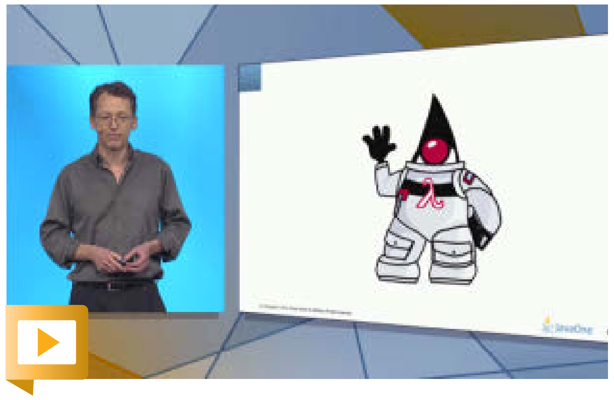
The combination of mobile and social is creating an enormous amount of data in many forms, with growing volume and velocity. Utzschneider said that between 10 and 50 billion nonhuman-driven devices will be coming on the internet in the next two years. "This is about the Internet of Things [IoT]," he said. "It will be a major game changer for Java developers and the larger community."

Ramani discussed unifying the Java platform. "With Java SE 8, we will release the Compact Profile and will replace Connected Device Configuration, so we will have one less implementation. We are also increasing commonality both from an API and a language perspective."

Ramani also touched on the IoT. "Everyone believes that there is a need for an open standard platform for the Internet of Things space that is coming—Java is the logical choice to address this market," she said.



Watch Strategy keynote highlights.



Watch Technical keynote highlights.

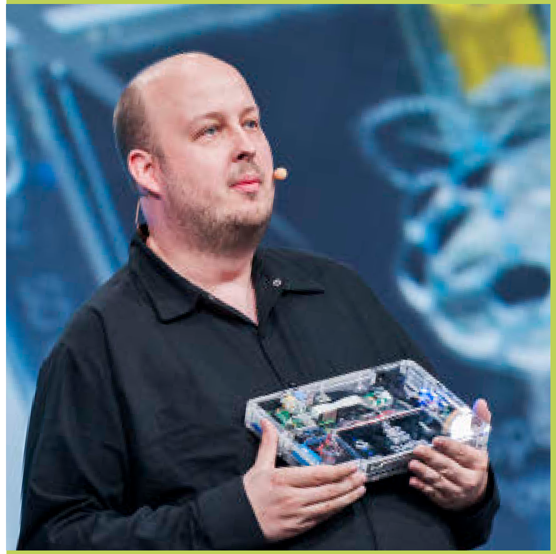
TECHNICAL KEYNOTE

Oracle's **Mark Reinhold** kicked off the JavaOne Technical keynote by quoting **James Gosling**. For Java to thrive, it must maintain what Gosling called the "feel of Java" and retain the key values of readability, simplicity, and universality.

"If we keep those," said Reinhold, "then Java will remain not just productive but fun. It is not enough to simply add popular features each year." Reinhold went on to focus on many innovations in Java. Here are some highlights:

Lambda expressions. Reinhold called lambda expressions the single largest upgrade to the programming model ever. "This is the first time we have done a carefully coordinated coevolution of the JVM [Java Virtual Machine], the language, and the libraries all together—and the results still feel like Java," he said. Oracle's **Brian Goetz** came on stage and said that lambda expressions will "change the way we all program in Java every day. Java has always given us good tools for abstracting over datatypes. I wanted to do better in abstracting over patterns of behavior—that's where lambda comes in."

DukePad. Oracle's **Jasper Potts** and **Richard Bair** demoed the DukePad, a do-it-yourself tablet based on the Raspberry Pi and Oracle Java SE Embedded 8. The DukePad uses Raspbian Linux as the OS and an OSGi-based JavaFX environment.



Top to bottom: Richard Bair plays chess; Jasper Potts shows off the DukePad; Mark Reinhold talks about lambdas.

PHOTOGRAPHS BY ORANGE PHOTOGRAPHY

FREESCALE AND THE IoT

Geoff Lees presents an Internet of Things vision.

Geoff Lees of Freescale Semiconductor kicked off the Java Community keynote to a standing-room-only crowd. He presented a vision of how the Internet of Things (IoT) might become a reality.

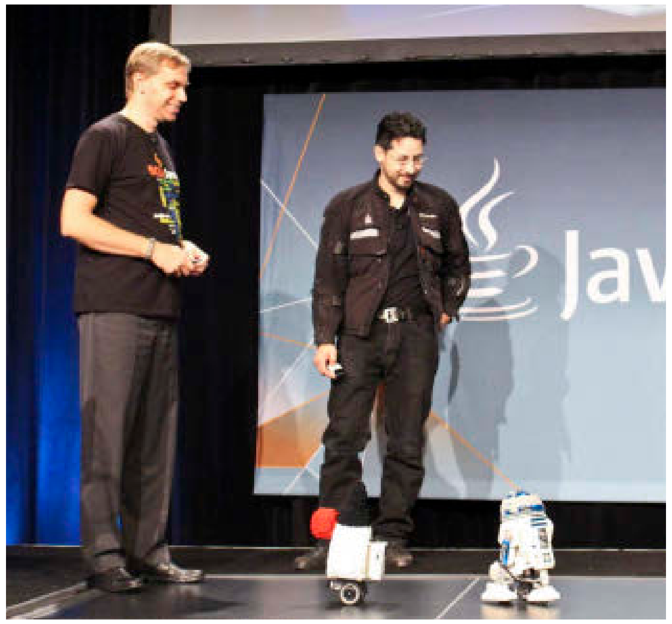
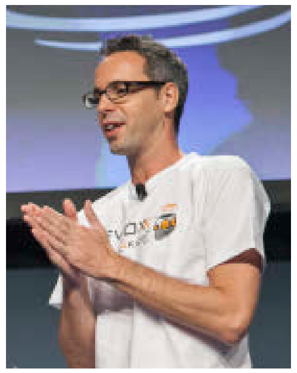
"The microcontroller community is rapidly moving to adopt Java, and we need your help," said Lees.

The IoT is changing the way the semiconductor industry is thinking about technology, Lee said, in terms of processing node transitions, greater utilization of advanced sensor technologies, integration of those technologies, and rapid adoption of low-power technologies both from processing and design techniques. Advances in signal analog integration and the IoT are bringing these things closer.

"Instead of the next few years, we're thinking about how to do all of this in the next few months," said Lees. The key to creating a secure IoT lies with Java developers, he said. Java-based edge nodes offer the potential to have secure encryption and authentication services throughout the network. Developing those in other environments will be locally difficult, will not be global, and will not reach the tipping point required for the IoT to develop.

Lees said that Freescale and Oracle are collaborating to develop a platform for software and hardware models for both edge nodes and a wide variety of gateway solutions. "We're working on optimizing Java together and bringing Java functionality further into the network."

PHOTOGRAPHS BY ORANGE PHOTOGRAPHY AND PETER PILGRIM

Community Keynote

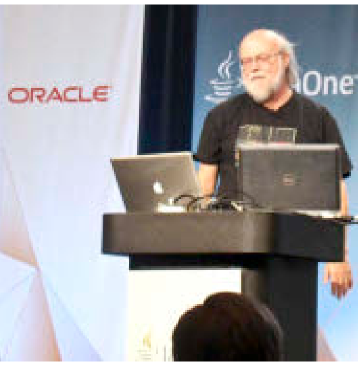
Oracle's **Donald Smith** followed Freescale Semiconductor's **Geoff Lees** at the Community keynote with a look back at recent JavaOne conferences. JavaOne 2011 was about moving Java forward and rebooting the infrastructure after the Java SE 7 launch. JavaOne 2012 focused on innovation and showing Java's role in major tech segments such as the cloud, big data, the Internet of Things, and open source.

"This year," Smith said, "we take one step beyond all that, and celebrate the end user and application developers. We want to show some inspiring applications being built thanks to the hard work of the Java ecosystem."

Over the course of the keynote, a number of innovators joined Oracle's **Henrik Stahl** on stage.

Java Champion **Stephan Janssen** talked about 2013 Duke's Choice Award winner Devoxx4Kids, a program that

Clockwise from left: Henrik Stahl and Stephen Chin play with Lego bots; Stephan Janssen talks about Devoxx4Kids; Aditya Gupta shares his Minecraft programming skills.



teaches computer programming to children between 10 and 14 years old. He offered the Devoxx4Kids teaching materials to any Java user group that would like to hold an event for children.

Oracle Academy's **Alison Derbenwick Miller** described the academy's outreach efforts that affect 2.5 million students in 102 countries.

Aditya Gupta, a 10-year-old Minecraft hacker, entertained the audience by showing off enhancements he made to the Minecraft source code (making pigs fly, creating never-ending cascading explosions, and otherwise having fun with the code).

Two Duke Segway robots appeared on stage and strutted around under the guidance of Java Champion **Stephen Chin**.

Java Champion **Paul Perrone** (via video) showed off Java-powered cars.

Drew Hylbert of Opower shared how Java technology is used to enable consumers to save energy.

Mike Marzo, a technology fellow at Goldman Sachs, discussed the value of the 100 million lines of Java code that Goldman's developers have written over the years.

Finally, **James Gosling** (top left), the "father of Java" and chief software architect at Liquid Robotics, appeared and remarked that Aditya Gupta made him feel that he, too, should be a Minecraft hacker. Gosling showed the view of Hawaii from one of Liquid Robotics' Wave Glider bots in the ocean and explained in detail how it all worked.

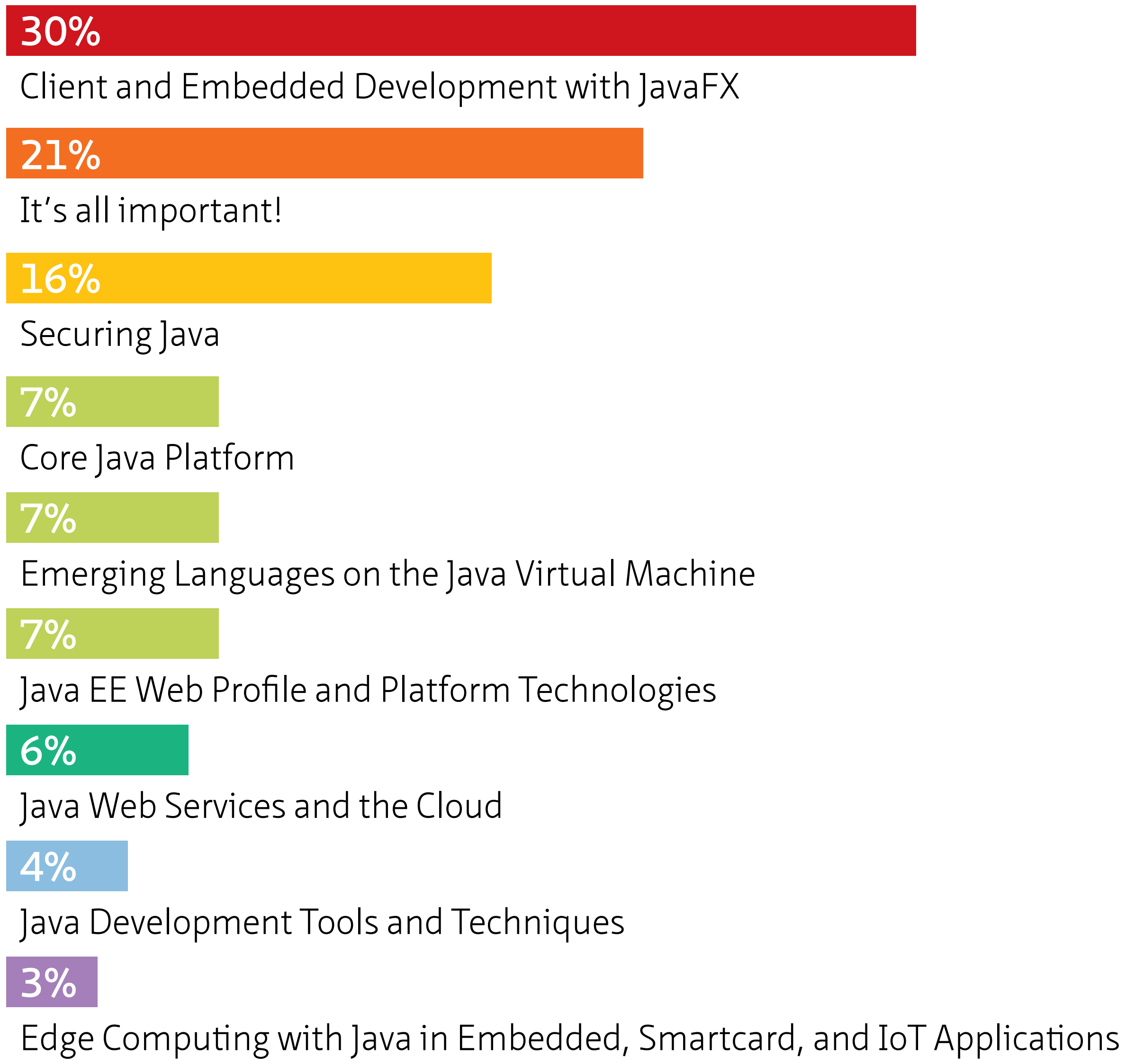
The Community keynote offered a strong sense of renewal and pride in what Java has accomplished and where it's headed.

PHOTOGRAPH BY PETER PILGRIM

JAVA.NET POLL

MOST IMPORTANT JAVAONE TRACK

In the weeks preceding JavaOne, a Java.net poll asked the Java community to consider the eight different technology tracks at JavaOne and select the most important one. The poll ran for two weeks, during which 304 votes were cast in response to the question: "The most important track at JavaOne 2013 will be . . . ." Here are the results:

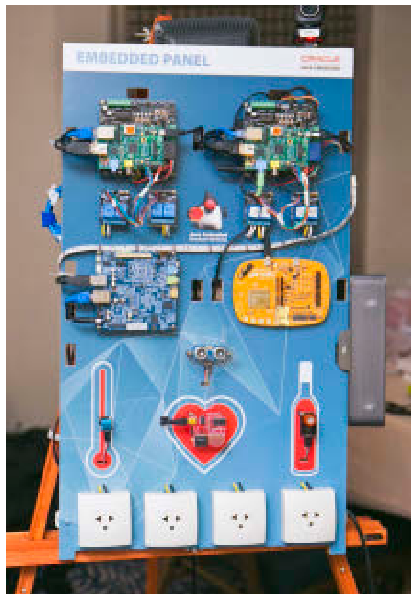


Ride Goes On

Despite heavy rains on the Saturday before JavaOne, the Geek Bike Ride went on, with 12 determined riders making the trip from Fisherman's Wharf over the Golden Gate Bridge to Sausalito. Talk about Java persistence!

PHOTOGRAPHS BY ARUN GUPTA AND YOLANDE POIRIER

JAVA EMBEDDED CHALLENGE FOR RASPBERRY PI



Clockwise from left: Vinicius Senger describes the Raspberry Pi's configuration; sensors on Raspberry Pi, Gemalto, and Beagle boards; Yara Senger discusses the goals of the challenge.

The Java Embedded Challenge for Raspberry Pi at JavaOne provided an opportunity for conference attendees to build embedded projects. The two-day challenge kicked off with a series of lectures that introduced the Raspberry Pi and projects that use it.

Globalcode's **Vinicius Senger**, who ran the event with Globalcode's **Yara Senger**, described the Raspberry Pi's layout and its configuration with other boards. He also showed off an embedded panel that he had built, which included Arduino, Beagle, and other boards. He added sensors to monitor alcohol in breath, heart rate, distance, and sound.

Over three days, attendees built Oracle Java SE Embedded applications integrated with sensors, Raspberry Pi, Arduino, and other boards. Experts and mentors gave presentations and coached the teams while they developed their applications.

"Our main goals were achieved," said **Yara Senger**. "People had fun coding until 11 p.m. despite the great parties [going on], and developed amazing projects in just a couple of days."

The teams developed seven remote-controlled applications: a heart monitor application using Google Glass, a radio-controlled car application, a home automation platform, a sobriety field tester, a remote control using any phone, a load-balancing cloud application, and a burglar alarm with voice/text alerts.

THE POWER OF COMMUNITY

Timon Veenstra

Badr El Houari

The importance of community was vividly illustrated throughout JavaOne 2013. The conference opened on Sunday, September 22, with NetBeans Day and Java user group (JUG) forums that covered everything from starting and maintaining a JUG to GlassFish to the Java Community Process (JCP) and the Adopt-a-JSR and Adopt-OpenJDK efforts. The last day of the conference began with the Community keynote address (see page 7). In between there were plenty of community-related sessions, panel discussions, and evening Birds-of-a-Feather (BOF) sessions.

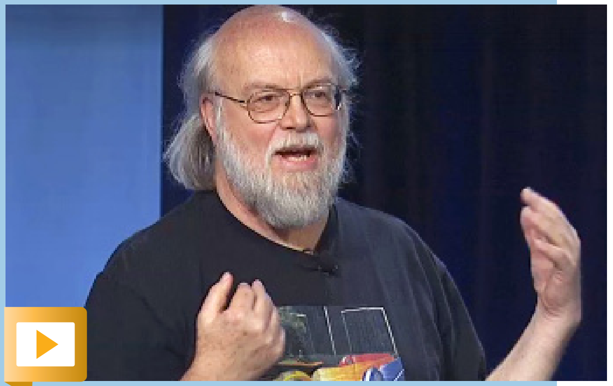
The fact that community is a fundamental aspect of Java's reach into people's lives was illustrated in a Sunday JUG forum presented by Morocco JUG leader **Badr El Houari** and JUG-Africa leader **Max Bonbhel**. El Houari successfully launched the first JMaghreb Conference in November 2012, with 30 sessions, 18 speakers, and 850 attendees. Bonbhel had just returned from the third JCertif Conference, which was attended by more than 1,800 developers from 20 different countries.

The NetBeans community was also fully engaged at JavaOne. Four Duke's Choice Awards went to applications built on top of the NetBeans platform. **Sean Phillips**, of a.i. solutions, spoke at the Community keynote about the NetBeans-based GEONS ground support system his team developed.

Meanwhile, **Timon Veenstra**, lead developer for AgroSense (a 2012 Duke's Choice Award winner), held a session where an airplane controlled by a NetBeans application flew in the session hall.

Another Duke's Choice Award winner was JFrog's Bintray, a community centered on redistributable software binaries. Bintray provides developers with the knowledge of how others are consuming their software, and also provides developers with a searchable resource for finding software for specific needs.

There are many different programming languages and toolkits, but in its community aspect Java is unique.


James Gosling at NetBeans Day

TIMON VEENSTRA PHOTOGRAPH BY TON HENDRICKS

Product News

The Java SE 8 Specification and JDK 8, its official Reference Implementation, are expected to be available in March 2014. The key features of Java SE 8 and JDK 8 are Project Lambda (JSR 335), the Nashorn JavaScript engine, a new Date and Time API (JSR 310), a set of Compact Profiles, and the removal of the "permanent generation" from the Java HotSpot VM.

Java ME 8 is expected to be available in March 2014, in conjunction with the Java SE 8 Specification. New features in Java ME 8 include Java language and API alignment with Java SE 8; support for modern web protocols; a comprehensive application model that will enable both simple, single-use devices and more-complex deployments; advanced security; standard APIs for power management; and interaction with a broad set of standard peripherals. Oracle Java ME Embedded 8 will be the Oracle implementation of the Java ME 8 standard. Oracle Java ME Embedded 8 Early Access is now available as a binary runtime for Raspberry Pi Model B (ARM11) and ST Microelectronics STM32F4DISCOVERY (ARM Cortex-M4).

PARTNER NEWS

Oracle introduced the Oracle Java Platform Integrator program, which gives partners the ability to customize Oracle Java ME Embedded and Oracle Java SE Embedded to reach different device types and market segments.

Gemalto is working with Oracle and V2COM, a developer of smart grid systems, to deliver a flexible smart-energy solution. This platform combines Gemalto's Cinterion modules, Oracle Java ME Embedded, Oracle Java SE Embedded, the Oracle Utilities Meter Data Management solution, and V2COM's Intelligenceware Suite.

Qualcomm Technologies and Oracle have collaborated to bring Oracle Java ME Embedded to key chipsets in Qualcomm Technologies' Internet of Everything portfolio. Oracle Java ME Embedded support is currently available on QSC6270-Turbo, and Oracle and Qualcomm are working together on expanding this to MDM6x00 and MDM9x15, as well as other chipsets.

Freescale Semiconductor has joined the OpenJDK community and will collaborate with Oracle and others to help evolve the Java platform and optimize Java for Freescale i.MX ARM-based applications processors. Freescale has also joined the Java Community Process (JCP) and intends to work with Oracle and other JCP members on future Java specifications for small and large devices.

Linaro has also joined the OpenJDK community and is already contributing to porting and optimizing Java for 64-bit ARM processors.

Square has also joined the OpenJDK community and is actively collaborating with Oracle and others in the community to enhance the Java programming language, Java Virtual Machine, and core libraries.

The Raspberry Pi will now ship with JDK 7. Future Raspbian images will ship with Java by default, and existing Raspberry Pi users can install the new Java support by typing the following:

```
sudo apt-get update && sudo apt-get install oracle-java7-jdk
```

From left: Mohamed Taman, Gil Tene (with Heather VanCura), Brian Goetz

Java Community Process Awards

The 11th annual Java Community Process (JCP) Program Award recipients were honored during JavaOne. The categories and winners were

JCP Member/Participant of the Year: Gil Tene, Azul Systems

According to the JCP, "[Tene] has worked diligently to provide clear advice on matters of software patents, IP, and licensing that seeks to benefit both nonprofits/individuals as well as organizations with vested commercial interests in Java."

Outstanding Spec Lead: Brian Goetz, Oracle

The JCP recognized Goetz for "tirelessly working away at an incredibly complex JSR: JSR 335, Lambda Expressions for the Java Programming Language."

Most Significant JSR: JSR 335, Lambda Expressions for the Java Programming Language

According to the JCP, JSR 335 "brings Java kicking and screaming into the modern programming language age . . . ."

Outstanding Adopt-a-JSR Participant: Mohamed Taman and Faissal Boutaounte, Morocco JUG and EGJUG

Taman and Boutaounte were praised "for adopting JSR 339, JAX-RS 2.0 specification, along with many other JSRs. One JIRA issue filed by Morocco JUG on JSR 339 was classified as a 'release-stopper.'"

Java Advent Calendar

The Transylvania Java User Group is seeking contributions for the second edition of the Java Advent Calendar, a series of 24 Java-related technical articles to be published daily during the first 24 days of December.

The Java Advent Calendar is based on a special Christmas calendar that is popular with children. Each day the child selects a new window to open and discovers the surprise—typically a small toy or chocolate—hiding behind it. Help make the Java Advent Calendar a gift to Java developers by contributing an article to the calendar!

JAVA CHAMPION PROFILE

BEN EVANS

PHOTOGRAPH BY JOHN BLYTHE

***Ben Evans** is a leader of the London Java Community (LJC), a member of the Java Community Process (JCP), an author and speaker, and a founder of startup jClarity. He was named a Java Champion in February 2013.*

Java Magazine: Where did you grow up?

Evans: I spent my childhood in Cornwall, England, and studied at Cambridge University, but I'm one of those technologists who believe that an open and playful mind is essential for proper creativity. So, I would tend to have concerns about some of the connotations that usually go along with the phrase *grow up*.

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

Evans: My parents felt very strongly that computers and programming were going to be an important part of the future, and gave me my own machine for my eighth birthday, in 1984.

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

Evans: ZX Spectrum 48K with BASIC in onboard ROM, soon followed by Z80 assembler.

Java Magazine: What was your first professional programming job?

Evans: Despite my early start, it took me 12 years to figure out that people would pay me to program. My first programming job was building a website for a surf shop/hotel/nightclub. I got paid in wetsuits, room and board, and a bar tab.

Java Magazine: What do you enjoy for fun and relaxation?

Evans: I like to go hiking, surfing in the summer, and snowboarding in the winter. I love to travel and usually don't have much trouble finding ways to have fun and relax.

Java Magazine: What happens on your typical day off?

Evans: I don't have many days when I don't do some work, but if I'm in London, I'll try to enjoy the side of the city that most people don't normally see because they're at work during the day.

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

Evans: I love to travel, so the opportunity to visit friends and colleagues all over the world and talk about my work is probably the best side effect.

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

Evans: I really like the jacket, and it's become my most frequently worn article of clothing that isn't black.

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

Evans: To pick just one technology, I'd say the JSON support. However, if we think about Java EE 6, it took a while for the real benefits to become well known across the industry—and I think the same will happen with Java EE 7.

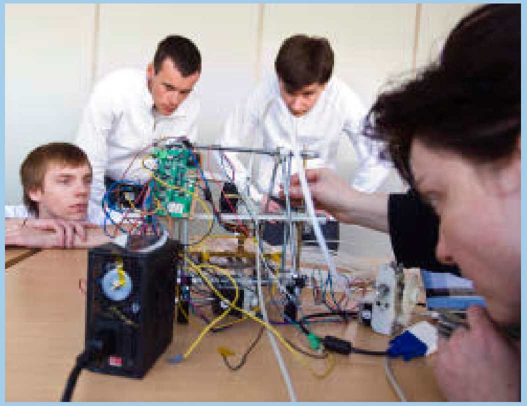
Java Magazine: What are you looking forward to in the coming years?

Evans: The unexpected things.

Follow Evans on Twitter and read his blog.

FEATURED JAVA USER GROUP

LYON JAVA USER GROUP


(Left) A traditional Lyon JUG meeting; a programming workshop

The Lyon Java User Group was formed in 2009 after discussions that took place between **Alexis Hassler**, **Laurent Gayet**, **Julien Ripault**, and **Cédric Exbrayat** on the French forum Developpez.com. It wasn't too long before the group contacted **Agnès Crepet**, who soon became a very active member of the Java user group (JUG). Exbrayat notes, "Agnès is a passionate developer. We were very pleased to have her join us, and when she did she also formed our Duchess group."

"When we founded the JUG, there were no user groups around whatsoever," says Exbrayat. "So we did it to meet other people, and we had talks about Java of course, but also about the web and mobile. Now there are a lot of small communities around, on every language, and I like the idea that our energy has contributed to that."

Lyon JUG normally holds one meeting each month. In addition, the JUG sometimes holds programming workshops on topics related to that month's meeting. Crepet says, "I think we are a fairly traditional JUG with monthly events, but we do encourage beginner speakers to do a lightning talk during each event."

"We also have a bigger event each year, Mix-IT, curated with the local agile group, where we host a conference for two days with international speakers and about 500 attendees," Exbrayat notes. "We have more than half of the talks about agile methodology, innovation, and mind-blowing new things (robotics and 3-D printing)."

The monthly meetings provide a great way to meet other developers. "It's sad that so many people are missing this opportunity," Hassler adds.

Learning about other technologies is also important to the JUG. "Java and its huge ecosystem are an amazing value in the lives of developers, but I'm really convinced that we have to know other languages and tools," says Crepet. "That's why we speak about Git or NoSQL in our JUG sessions."

PHOTOGRAPHS COURTESY LYON JUG

Smart Home Workshop at JCertif 2013

Oracle Technology Network sponsored a Smart Home Workshop at JCertif 2013, held September 9–15 in Brazzaville, Republic of the Congo.

Firas Gabsi (above left), a passionate, young Java EE and mobile engineer and a professor of engineering at the Engineering Institute in Tunisia, led the workshop.

Attendees learned to integrate Oracle Java SE Embedded with Raspberry Pi and other equipment for home automation systems.

"I didn't know much about the Java embedded technology, but now I have acquired the basic knowledge that I will keep improving," said Java engineer **Yanhick Keny**, who was one of 80 participants.

Give it a try! Download the project from GitHub.

Gabsi demos one of the projects.

EVENTS

Jfokus *FEBRUARY 3–5*
STOCKHOLM, SWEDEN

Jfokus, one of the premier European Java developer conferences, is back for its eighth year. Held in Stockholm, Sweden, Jfokus consists of six tracks including a subconference on embedded technologies. Conference topics include Java SE and Java EE, front end and web, mobile, continuous delivery and DevOps, the Internet of Things, cloud and big data, future trends, alternative languages—such as Scala and Clojure—on the Java Virtual Machine, and agile development. Speakers include Dr. Venkat Subramaniam of Agile Developer, Jim Manico of WhiteHat Security, Martin Thompson of Real Logic, and many more.

PHOTOGRAPH BY GETTY IMAGES

jDays
NOVEMBER 26–27
GOTHENBURG, SWEDEN
This Java developers conference offers sessions on Java SE, Java EE, frameworks and servers, front end, web and mobile, trends and future, solutions, case studies and real-world experiences, and methodologies and tools.

Groovy & Grails eXchange
DECEMBER 12–13
LONDON, ENGLAND
Industry-leading experts and developers from around the world gather at this conference to learn and share everything about the Groovy and Grails ecosystem.

Take Off
JANUARY 30–31
LILLE, FRANCE
This conference for web developers and designers offers a range of sessions on topics including web servers, front end, frameworks, and development techniques.

DevNexus 2014
FEBRUARY 24–25
ATLANTA, GEORGIA
The Atlanta Java User Group (AJUG) has organized this Java conference for the past eight years. It covers Java topics including web technologies, architecture, big data, enterprise software, mobile, Java SE, testing tools, and methodologies.

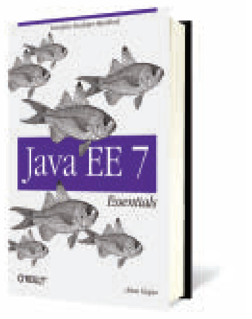
8th Annual IndicThreads Pune Conference
EARLY 2014
PUNE, INDIA
Web technologies, server side, big data, and mobile software development are the focus of this conference.

Java Spotlight Podcast

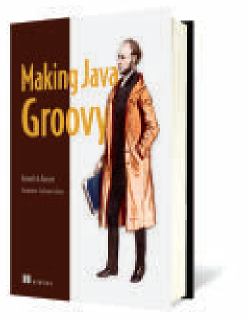
Listen to the Java Spotlight podcast for interviews, news, and insight for and from Java developers. Hosted by **Roger Brinkley**, this weekly show includes a rotating panel of all-star Java developers.

JAVA BOOKS

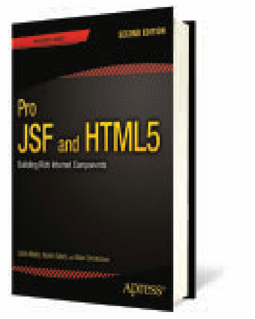
JAVA WEBSOCKET PROGRAMMING
By Danny Coward
Oracle Press (August 2013)
Learn how to build dynamic enterprise web applications that fully leverage state-of-the-art communication technologies. Written by the leading expert on Java WebSocket programming, this book offers practical development strategies and detailed example applications. *Java WebSocket Programming* explains how to design client/server applications, incorporate full-duplex messaging, establish connections, create endpoints, handle path mapping, and secure data. You'll also learn how to encrypt web transmissions and enrich legacy applications with Java WebSocket.



JAVA EE 7 ESSENTIALS
By Arun Gupta
O'Reilly (August 2013)
Get up to speed on the principal technologies in Java EE 7, and learn how the latest version embraces HTML5, focuses on higher productivity, and provides functionality to meet enterprise demands. Written by Arun Gupta, a member of the Java EE team, this book provides a chapter-by-chapter survey of several Java EE 7 specifications, including WebSocket, Batch Processing, RESTful Web Services, and Java Message Service.



MAKING JAVA GROOVY
By Kenneth A. Kousen
Manning Publications (September 2013)
Making Java Groovy is a practical handbook for developers who want to blend Groovy into their day-to-day work with Java. It starts by introducing the key differences between Java and Groovy—and how you can use them to your advantage. Then, it guides you step-by-step through realistic development challenges, from web applications to web services to desktop applications, and shows how Groovy makes them easier to put into production.



PRO JSF AND HTML5
By Zubin Wadia, Hazem Saleh, and Allan Lykke Christensen
Apress (November 2013)
Pro JSF and HTML5 shows you how to leverage the full potential of JavaServer Faces (JSF) and Ajax. This is not an entry-level tutorial, but a book about building Ajax-enabled JSF components for sophisticated, enterprise-level rich internet applications. Written by JSF experts and verified by established community figures, this book provides reliable and groundbreaking JSF components to help you exploit the power of JSF in your Java web applications.

JCP Executive Series

Charting the Future—JCP.next

Oracle's Patrick Curran discusses the JCP's evolutionary path toward greater transparency and participation, and the embrace of open source processes. **BY STEVE MELOAN**

PHOTOGRAPHY BY BOB ADLER

Patrick Curran is chair of the Java Community Process (JCP) organization. In this role, he oversees the activities of the JCP's Program Management Office (PMO), including evolving the process and the organization, managing its membership, guiding Specification (Spec) Leads and experts, chairing Executive Committee meetings, and managing JCP.org.

Curran has worked in the software industry for more than 25 years. His experience at Sun Microsystems, and then Oracle, spans 20 years. Before joining the JCP, he led the Java Conformance Engineering team in Sun's Client Software Group. He was also chair of Sun's Conformance Council, which was responsible for defining Sun's policies and strategies around Java conformance and compatibility.

He has participated actively in several consortia and communities including the W3C (as a member of the Quality Assurance Working Group and co-chair of the Quality Assurance Interest Group) and OASIS (as co-chair of the Test Assertions Guidelines Technical Committee).

Curran (center) confers with Thomas Lampart of Gemalto M2M (left) and Calinel Pasteanu of Oracle at a JCP Executive Committee meeting prior to JavaOne.

In this interview, Curran discusses how the JSR process is being used to modify itself toward greater transparency, participation, and open source compatibility.

Java Magazine: What are the most important changes implemented by JCP.next?

Curran: JSRs 348, 355, and 358 are collectively referred to as JCP.next. These JSRs address a number of important issues related to openness, agility, and governance.

There used to be a perception that Expert Groups operated secretly, working behind closed doors for months or even years, after which a new specification would mysteriously appear. JSR 348, which is now in effect, changes that paradigm substantially. It requires Expert Groups to conduct their business transparently via public mailing lists and a public Issue Tracker. The JSR explicitly states that all JCP members, and members of the public, must have the opportunity to view, comment on, and participate in the process.

Transparency and participation are the keys to running an effective standards-development process in a world where open source practices are becoming increasingly ubiquitous. JSR 348 facilitates these practices.

JSR 355, also complete, was fairly simple. We previously had two separate Executive Committees, one for Java ME and one for Java SE/EE. Now those have been merged into one. With Java ME and Java SE/EE convergence on the horizon, it no longer makes sense to maintain two Executive Committees. This change will streamline operations considerably.

JSR 358 is still in progress. It will take transparency and participation to the next level, mandating the use of open source development processes and open source licenses for virtually all JSRs. This is a complex task, because it involves changes to the Java Specification Participation Agreement (JSPA), the legal contract that JCP members sign when they join the organization. It is difficult to modify the JSPA because it contains complex legal language dealing with issues such as intellectual property and licensing models. We have to be very careful when changing it, because the repercussions can be far-reaching. So, JSR 358 is going to take some time. We've been working on it for a year, and it will probably take another year to complete.

Java Magazine: The JSRs you described make great strides in transparency. Where is there still room for improvement? And what roles will the Executive Committee and the PMO play?

Curran: All Expert Groups are complying with the "letter of the law." Now we must ensure that they comply with the "spirit." There's no point in having a public mailing list if it receives little traffic, or if comments from outside the Expert Group are ignored. And there's little value in having a public Issue Tracker if very few issues are logged, or if those that are logged are not acted upon. We must make sure that people who wish to participate know how to participate, and that their participation is effective. The JSR's home page must provide the necessary information to put all that in motion.

We're working on a set of reporting requirements that will track the activities of Expert Groups. They will have to gather and publish this information, and make it available for everyone. By doing this, we believe engagement will be significantly improved.

Also, the PMO is working to provide information that will allow us to judge how well an Expert Group is meeting its transparency and participation obligations. We hope that the combination of public disclosure of information, public pressure, and the Executive Committee taking this information into account when voting on JSRs, will motivate the Expert Groups to fully meet their obligations.

OUT IN THE OPEN
"Transparency and participation are the keys to running an effective standards-development process."

Java Magazine: In what ways will JCP.next bring more individuals, Java user groups [JUGs], and other entities into JCP processes?

Curran: We expect that enabling people to participate through the use of public mailing lists and open source development processes will make JCP participation much more attractive to the average Java developer. We're actively recruiting JUGs through a program we call Adopt-a-JSR, whereby developers get together through the JUGs to assist with JSR efforts that interest them. This might involve activities such as critiquing the spec, testing the implementation, or being involved with documentation. We now have more than 40 JUGs as members, collectively representing tens of thousands of developers. In JSR 358, we're working to create a new class of membership tailored for individuals. This will involve a membership agreement that is much simpler than the current complex legal document and will not require an employer's signature, which has sometimes been a stumbling block.

We've been very successful in recruiting individuals and JUGs, but we still have work to do to increase participation by commercial entities. Since the primary motivation for companies to join the JCP is an active and vibrant JSR development process, the key to greater involvement is to have a lot of JSRs in progress. This is not something the Executive Committee or the PMO can directly influence, since JSRs are initiated by our members. However, we hope that the introduction of open source development processes and open source licensing via JSR 358 will provide a significant boost in commercial participation.

Java Magazine: How will JCP.next use both positive and negative reinforcement to keep innovation on track?

Curran: We use a number of mechanisms to encourage Spec Leads to move their work through the process in a timely manner. The main positive reinforcement is the Star Spec Lead

Curran talks with Executive Committee member Bruno Souza of SouJava.

Left to right: Mike Marzo of Goldman Sachs, Curran, and Jack Chung of Aplix listen to a presentation during a JCP Executive Committee meeting.

program, which publicly recognizes exceptional Spec Leads through the annual JCP Awards. The awards also recognize outstanding JSRs and members who have contributed significantly during the previous year.

As for negative reinforcement, JSR 348 introduced the notion of time-outs, whereby JSRs that do not reach defined process stages within specified time periods are subject to a Renewal Ballot. The Executive Committee may then require that the JSR be withdrawn if the Spec Lead cannot justify the delay and convince the Executive Committee that work is still constructively proceeding.

Java Magazine: What methodologies will JCP .next use to ensure that a specification, its Reference Implementation [RI], and its Technology Compatibility Kit [TCK] are completed simultaneously?

Curran: The PMO checks to be sure that the specification, RI, and TCK are all available before initiating a Final Approval Ballot. The software licenses are also required. If all these elements are not in place, the ballot cannot move ahead. We've also added some language to the Process Document stating that if the materials are not formally posted to jcp.org within 14 days, or if links to these materials are later broken, another Executive Committee ballot can be initiated to label the JSR as withdrawn on the grounds that the Spec Lead has abandoned it. So we hope this combination of checks will ensure that all the necessary elements are in place when the process is complete, and that they will continue to be available to implementers.

Java Magazine: How will the maintenance process be affected by JCP.next?

Curran: Previously there was no formal ballot before the Maintenance Release. There was nothing in the Process Document that required published updates to the RI, the TCK, or even the Spec. Spec Leads would sometimes simply publish a change log and leave it up to implementers to figure out what they needed to do to adopt the new version. That didn't make sense. Now we have maintenance reviews similar to the Final Review, whereby materials are made available, the Executive Committee will study them, and the public will be granted access.

But we're aware that more work lies ahead. We expect to make further changes in JSR 358 in order to make the maintenance process more compatible with the type of continuous development and release processes adopted by open source projects.

Java Magazine: Will you explore in greater detail the merging of the Executive Committees for Java SE/EE and Java ME? How will this benefit both camps and the process as a whole?

Curran: Java ME began as a subset of Java SE, but over the years they've diverged. Now many of the new features of Java SE, particularly the language features, are being incorporated into Java ME, making them more similar and compatible. So, the merging of Executive Committees made sense, and was a simple change that we introduced in JSR 355. We wanted to empha-

Curran takes a walk with Executive Committee member David Britto of TOTVS.

size that Java is one platform, and since we expect that Java SE/EE and Java ME will become more aligned over time, it didn't make sense to maintain two separate Executive Committees. The two committees seldom worked or met separately anyway. With the filing of JSR 360 (Connected Limited Device Configuration 8) and JSR 361 (Java ME Embedded Profile), the Java SE/EE/ME convergence is in motion. We expect increased synergy between the Java SE/EE and Java ME platforms and a more streamlined Executive Committee with fewer members.

Java Magazine: What is the current balance of power between Oracle and others within the JCP?

Curran: This is one of the most fundamental areas of concern as we revise our processes via JCP.next. Obviously Oracle has a special role as the steward of Java, as the Spec Lead for the platforms, and as the most significant investor in the development of Java technologies. However, if Java were proprietary it would not have achieved such broad and ubiquitous success. There are approximately 9 million Java developers worldwide. It is, therefore, critical that processes be open and inclusive, and that others, particularly Oracle's competitors, have the opportunity to collaborate and to participate.

A significant majority of JSRs are now led by Oracle, which skews the Spec Lead role in Oracle's favor. Others certainly participate, since they are active members of the Expert Groups, but it would be healthier if there were more JSRs led from outside Oracle. JSR 358 will embrace open source development processes and open source licensing, which we hope will increase the participation of others in Oracle-led JSRs, and also encourage the creation of more JSRs from outside the Oracle domain. It's a delicate balance between Oracle's legitimate business interests, as the steward of Java, and the need for open and collaborative processes.

Java Magazine: Transparency and participation are obviously enhanced by JCP.next. In conclusion, could you expand further on how JCP.next promotes best practices?

Curran: Facilitating transparency and enabling participation are essential to maintaining an effective standards-development organization in the current era.

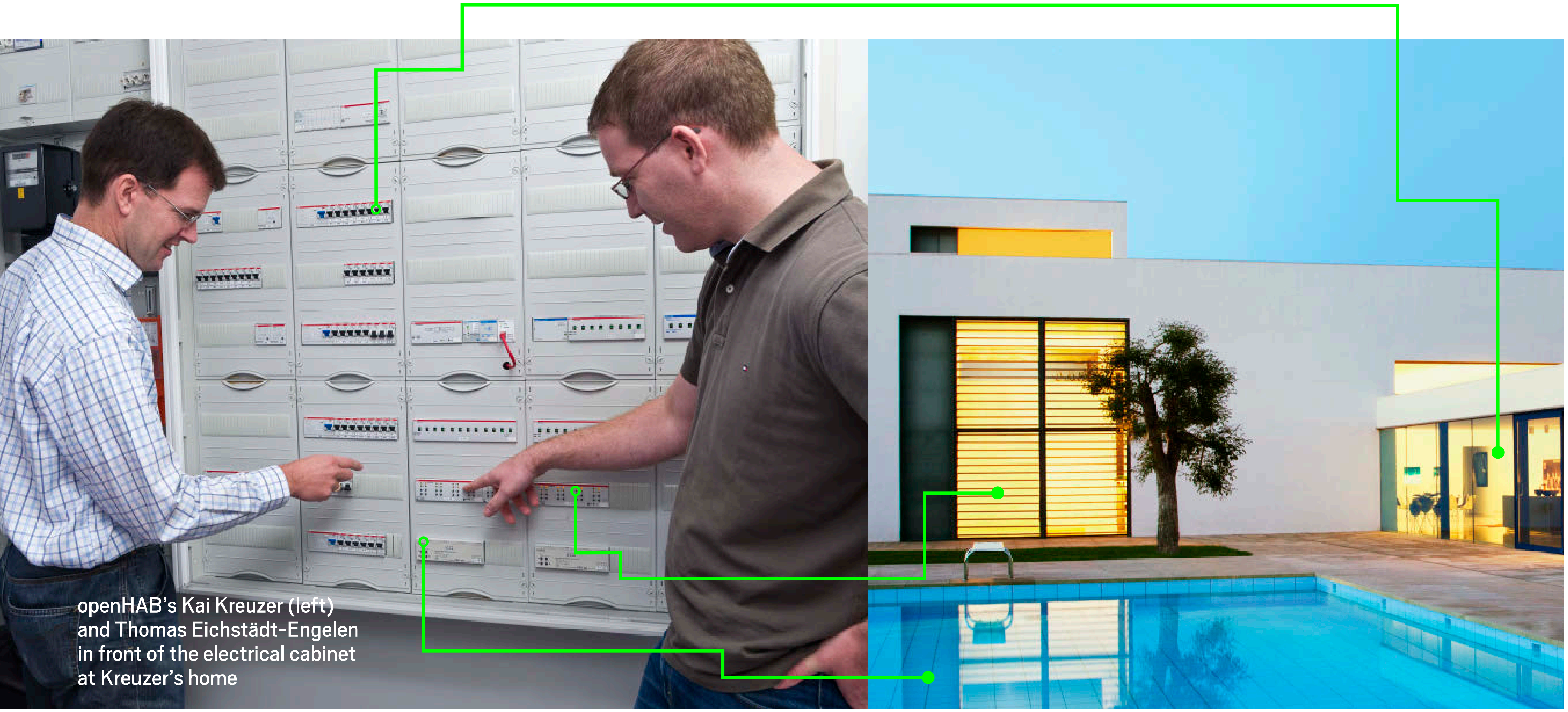
People will participate only if our governance and processes are open. That's critical. These best practices increase the quality of the standards we develop. With more active participants, we'll have greater diversity of viewpoints and more problems being fixed. The resulting technologies benefit across the board.

JCP.next is a continuum that began with some basic transparency and participation changes, and is moving forward to embrace open source development processes and open source licensing. These modifications will help to ensure the continuing strength and relevance of Java as we move forward in a very dynamic and competitive landscape. </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*.

LEARN MORE

- Read Patrick Curran's blog
- JCP.next


openHAB's Kai Kreuzer (left) and Thomas Eichstädt-Engelen in front of the electrical cabinet at Kreuzer's home

SMARTEST HOUSE ON THE STREET

The Internet of Things meets home automation with openHAB, a Java-based software environment that integrates devices and applications into a cohesive network. **BY DAVID BAUM**

ART BY WES ROWELL; PHOTOGRAPHY BY TON HENDRIKS

Kai Kreuzer is not an easy man to sneak up on. As you approach the front door of his highly automated house, you will trigger a sensor that activates a webcam mounted above the front door. Ring the doorbell and he will be alerted to your presence, via either loudspeakers throughout the property or a video display on his iPhone. He may choose to let you in by remotely unlatching the door. With a few more taps on his

Kreuzer controls his home entertainment system through HABDroid, one of the native clients for openHAB.

SNAPSHOT

openHAB
openhab.org

Headquarters:
Darmstadt, Germany

Industry:
Home automation, open source software

Contributors:
37

Java technology used:
Java SE 1.7

phone he can adjust the lights, turn on the music, and adjust the temperature of the home's central heating system. If he is in the garden, you may hear his voice coming over a PA system, directing you outside.

What's unique about this scenario is not the individual systems that automate every aspect of Kreuzer's modern home, but the way these systems work together to improve the convenience, security, and efficiency of dozens of routine tasks.

Home automation is a prime example of the *Internet of Things*, a phenomenon that is exploding across the domains of healthcare, manufacturing, transportation, and communications as billions of intelligent devices flood into our personal and professional lives. Java plays a starring role as a central integration point in this machine-to-machine world by making our cars more efficient and dependable, our homes more comfortable and secure, and our healthcare less costly and more patient friendly—to name a few prominent examples.

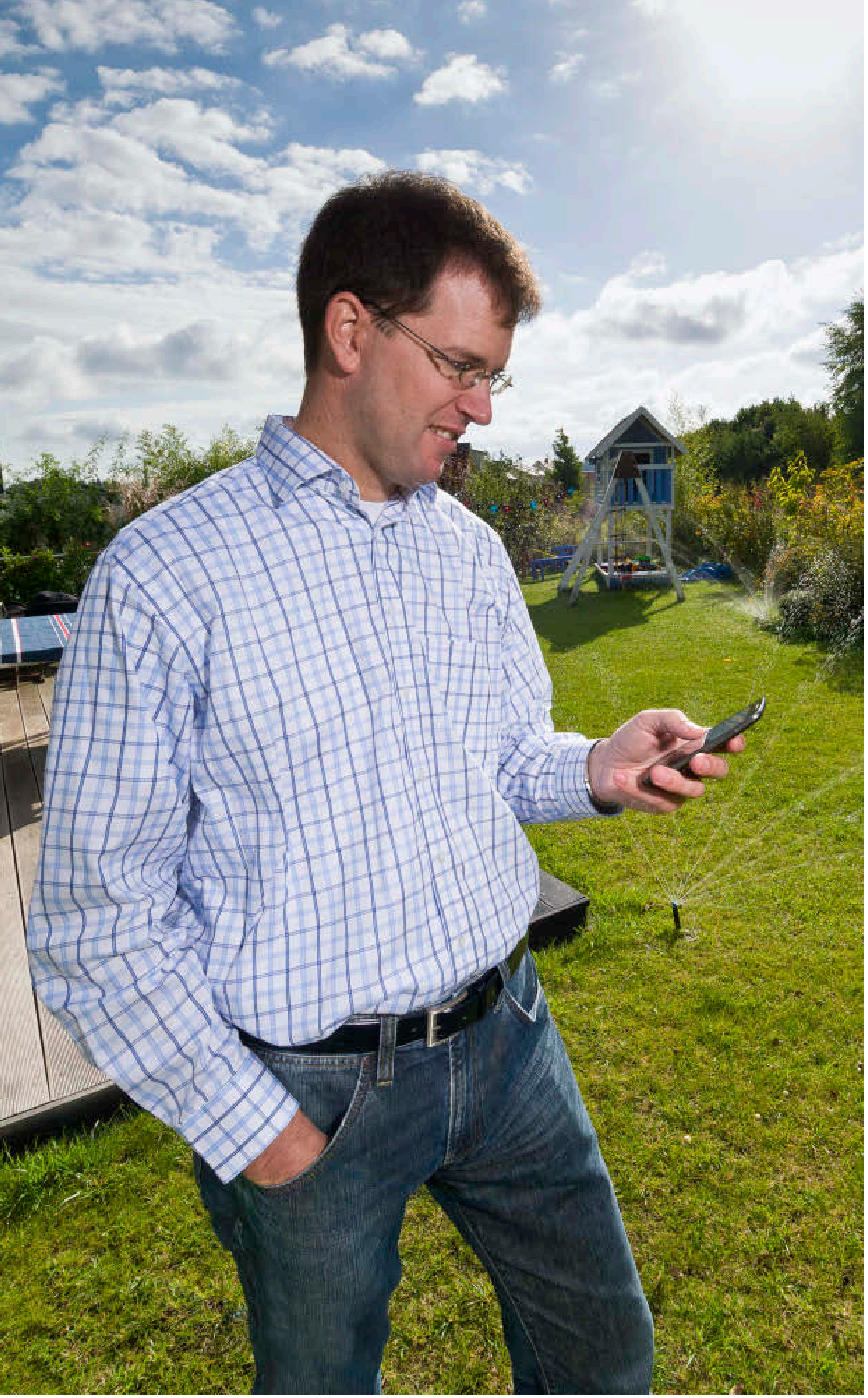
Kreuzer studied mathematics and computer science at the Technical University of Darmstadt, where he took a particular interest in Java. He has used the language extensively in his professional life as well as in his home automation systems. This lifelong hobby culminated with the founding of open Home Automation Bus (openHAB) in 2010, a Java-based software environment that integrates devices and applications throughout his home.

With openHAB as the controller, all of a home's comfort systems, security systems, and energy systems work in concert and can be triggered in unison—through a smartphone interface, via Google Calendar events, or through many other hardware and software interfaces.

Kreuzer's brainchild quickly gained traction in the open source community. Today there are 37 contributors and between 2,000 and 3,000 openHAB installations worldwide.

COMMUNITY ROLE
Contributors help to expand the openHAB ecosystem by creating connections to various devices, applications, and interfaces.

Thomas Eichstädt-Engelen is a fellow home automation enthusiast who now serves as a project leader at openHAB. With a degree in computer science from the University of Hagen and full-time work in the IT field, he is well versed in the software industry



Kreuzer uses his smartphone to manually trigger his sprinkler system.

and has an affinity for the open source movement. He joined Kreuzer a few months after the openHAB environment was launched.

"My own home automation project began with the physical infrastructure and soon progressed to software," Eichstädt-Engelen recalls. "It is not enough to have the hardware in your flat. You have to have something to control it. I liked openHAB because it was free and open source. It was a perfect fit to my skills because it is based on Java and OSGi."

Eichstädt-Engelen has used openHAB at home to connect many different devices and applications into a cohesive fabric. "In every room I have a loudspeaker, centrally operated by openHAB, along with the lights, the appliances, and the heating and ventilation systems," he explains. "All of the doors and windows have electric contacts that signal the security system. Most of the electric appliances can be controlled remotely, and the various systems work in unison. For example, the heating system will shut off if the windows are left open. The door won't lock behind you if you go out on the balcony to get some fresh air."

DESIGN GOALS: BETTING ON JAVA AND OPEN SOURCE

Kreuzer and Eichstädt-Engelen both have day jobs in the IT field and pursue openHAB in their spare time. Eichstädt-Engelen is a software developer at innoQ, a midsize software consulting company. Kreuzer works at Deutsche Telecom as an IT engineer and architect, with previous jobs in the media and banking industries. "Java was always my language of choice, personally and professionally," he says.

In 2008 Kreuzer built a house from scratch and decided to include advanced automation systems that could be interconnected. He worked closely with an electrical engineer to plan and wire the house. Like Eichstädt-Engelen, he was leery of proprietary solutions that would tie his automation systems to a vendor that

Top: A weather station in Kreuzer's garden detects brightness, rain, wind, and temperature, and sensors control window blinds, sprinklers, and the HVAC system. Right: A GIRA system monitors windows and lights, shows missed calls or visitors, and communicates via text-to-speech.

WORKING TOGETHER
OpenHAB enables discrete automation with centralized control: individual systems can be automated, yet each one is aware of the others through a common controller.

might not exist in 10 or 20 years.

"Since I planned to live in this home indefinitely, I didn't want to bet on a horse that wouldn't win in the long run," he says. "I knew that if I invested in an open source solution I could always extend it as technology evolved. It could be developed and maintained by myself and by others in the open source community. As a professional Java developer, I was not really satisfied with existing open source solutions so I decided to create something completely from scratch and build it for my own use, yet make it flexible and extensible so that it would also be useful for others."

Fast-forward five years and you can see the results of Kreuzer's suburban automation efforts. In addition to a webcam near the front door, he installed a Near Fields Communications (NFC) reader that can control the lock based on signals from inexpensive RFID tags that he issues to his guests. This method is particularly useful for service people, who he may want to let in on a one-time or periodic basis. "Rather than giving a key to my cleaning service, I can issue an RFID tag. This lets me control when the lock will open for them," he says.

A weather station in Kreuzer's garden detects brightness, rain, wind, and temperature, along with moisture sensors in the soil. Because the sprinkler system is attached to openHAB, the watering system comes on only when the plants need it. A pump is activated with a float switch, which also registers its activities, so there is always enough water in the tank for the garden. This level of integration is what makes openHAB so useful in comparison to standalone or "siloed" automation solutions.

"If you buy a security solution from one vendor and a comfort solution from another, it is generally very difficult to combine them into one smart house," Eichstädt-Engelen explains. "In addition, many of today's home automation systems were designed for very high-end homes.

Kreuzer and Eichstädt-Engelen discuss openHAB with *Java Magazine*'s Caroline Kvitka during JavaOne 2013.

With openHAB, we are bringing this same level of sophistication to average homes and apartments."

Home automation use cases can be divided into three basic categories: comfort, security, and energy management. Some features of a home, such as draperies and shutters, span multiple categories: automatically opening them in the morning and closing them in the evening keeps the house comfortable; saves energy; and improves security, because it makes it appear as if the resident is home even when the house is empty. These activities can occur automatically based on the time of day or can be triggered by sensors that detect when people are home. Lighting, heating, cooling systems, and appliances can also be activated automatically or can trigger other systems. For example, when Kreuzer's washing machine finishes a load it broadcasts its status over the loudspeaker.

WORKING WITH THE JAVA COMMUNITY

The first binary build of openHAB was available for download in the fall of 2010. Since then, Eichstädt-Engelen and Kreuzer have presented the solution at Java user groups and Java conferences such as EclipseCon, Devoxx, JAXconf, and GeeCON. Their growing community now includes a loyal base of contributors that help to expand the openHAB ecosystem by creating *bindings*, or connections to various devices, applications, and interfaces.

"Bindings are the pieces of code that allow us to connect openHAB to other systems and also to integrate it all together," says Eichstädt-Engelen.

There are currently about 50 bindings to commercial automation systems including Z-Wave, Plugwise, SONOS, Bluetooth, Modbus, EnOcean, and KNX. openHAB also provides consoles such as XMPP, OSGi, and Google Calendar. Thanks to these consoles and bindings, along with the steady evolution of home automation technologies, homeowners no longer need physical buttons to switch on lights and other electric devices. Pointing a smartphone at an NFC tag hidden behind the wallpaper will do the job—or they can program openHAB so that the phone will signal these devices automatically whenever their owner walks into the room.

"Soon, presence detection technology will allow openHAB to identify you as you move around the home," Kreuzer says. "This will allow openHAB to adjust the lights or shutters or music to your preferences, based on the time of day or any other variables you choose."

End users simply download openHAB and unpack it in a Java runtime environment on the target system. "It's more or less a one-click installation," Eichstädt-Engelen notes. "Within five minutes, you can have a running openHAB system."

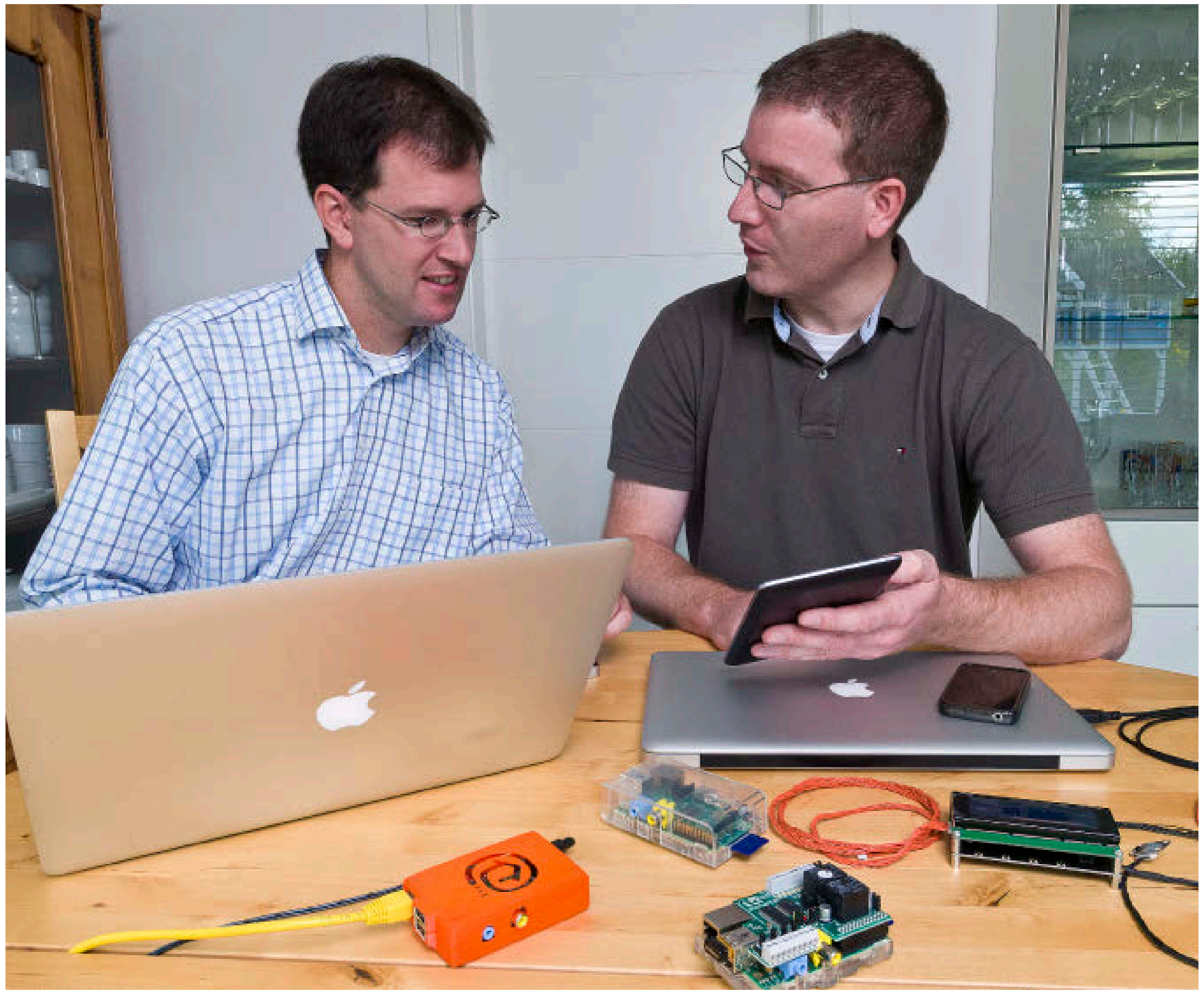
The project includes the openHAB Designer, an Eclipse Rich Client Platform application for configuring the openHAB runtime. It comes with editors for the openHAB configuration files, with full integrated development environment (IDE) support such as syntax checking, autocompletion, highlighting, and content assistance. Kreuzer claims that this mature development environment makes it easier to implement and deploy rules for automatic actions.

NO LOCK-IN
One of the fundamental design goals is modularity: it is easy to replace one technology with another so that homeowners aren't locked in to particular types of applications and devices.

WHAT JAVA BRINGS

"Java is very useful because it has a huge ecosystem of libraries, great debugging tools, and lots of support available on the web."

—Kai Kreuzer, Founder, openHAB



Kreuzer and Eichstädt-Engelen program openHAB and test it on a Raspberry Pi with a tablet.

openHAB also includes a scripting language so that developers can easily define new types of automation logic.

"OSGi is an important aspect for openHAB because it brings modularity to our systems," Kreuzer explains. "This makes it easy for contributors to build modules and bindings independently of the core development. Users can pick relevant modules and add them to their openHAB system at runtime."

"Java is very useful because it has a huge ecosystem of libraries, great debugging tools, and lots of support available on the web," Kreuzer continues. "It's easy to find a solution for a specific problem. Java is also platform independent, so you can run it on Linux, Windows, Mac, or an embedded platform."

Eichstädt-Engelen agrees. "The best JVM [Java Virtual Machine] available for embedded systems is Oracle Java SE Embedded."

Using Java also makes it easier to collaborate with other developers. "There would be fewer contributors if we had chosen another language," Kreuzer adds. "We are part of a big ecosystem, so we can almost always find the solutions we need."

Neither Eichstädt-Engelen nor Kreuzer plans to commercialize openHAB. However, they recently proposed the Eclipse SmartHome project, which will make central parts of openHAB available under the Eclipse license. This move paves the way for the integration into commercial products and ensures sustainability.

But no matter where the openHAB technology goes and how the open source community evolves, the two partners plan to keep the system closely aligned with Java.

"Java is future proof and platform independent," Kreuzer sums up. "The Java APIs are very stable and consistent from one version to another. Java's diversity makes it easy for other contributors to get involved, all over the world." `</article>`

Based in Santa Barbara, California, **David Baum** writes about innovative businesses, emerging technologies, and compelling lifestyles.

BEST DEVELOPMENT PRACTICES

AGILE ADJUSTMENT

Java Champion **Venkat Subramaniam** explores the subtleties of agile development. **BY TIMOTHY BENEKE**

PHOTOGRAPHY BY BOB ADLER

Since its inception in 2001, agile development—defined on Wikipedia as "a group of software development methodologies based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams"—has had an impressive history, with many organizations adapting its principles. In recent years however, a number of questions have arisen: What kinds of organizations make a good fit for agile? Is it being deployed in dysfunctional ways? Does agile development require developers to have a certain kind of skill set to be effective? What goes wrong when agile fails?

To address these and other issues, we met up with Dr. Venkat Subramaniam, the founder of Agile Developer, who has trained and mentored thousands of software developers in the US, Canada, Europe, and Asia. Known for his contagious enthusiasm, he helps developers succeed with agile practices on their software projects, and he speaks frequently at international conferences and user groups. He is the author of many books, including, most

Venkat Subramaniam catches up on e-mail during a break between sessions at JavaOne 2013 in San Francisco, California.

recently, Functional Programming in Java: Harnessing the Power of Java 8 Lambda Expressions.

Prior to starting his own company, Subramaniam worked in various positions, from programmer analyst to systems architect, at organizations such as Halliburton, Raytheon, and Invensys. He holds a PhD in computer science from the University of Houston, where he is an adjunct professor, and was named a Java Champion in 2013.

Java Magazine: Why do people talk about the demise of agile?
Subramaniam: It is a little like the parable of the four blind men who visit an elephant at the zoo. They cannot see the elephant, so each touches a different part and draws a different conclusion about it.

I recall a time when there was suspicion about whether object-oriented development would ever work for the enterprise. Some companies succeeded while others struggled as the world ventured into the then-new paradigm. Today, it's a foregone conclusion and object-oriented programming is widely adopted. Organizations are going through a similar phase with agile, a learning curve, to figure out what it is and how to use it.
Java Magazine: How are organizations using agile?
Subramaniam: I see two ways in which organizations have taken up agile. There are organizations in which the management team has pushed agile, and we see things such as Scrum being predominant. Then there are organizations in which the push is from the bottom up, and we see more-technical or extreme programming practices being predominant. Neither of these situations is ideal, because there is a part of the organization that is not aligned with, supportive of, and responsive to the need for change. The results of such efforts are often not very positive unless the rest of the organization joins the effort to foster change.

AGILE'S ESSENCE
Collaboratively and skillfully adjusting to feedback cycles through iterative, incremental development is the essence of agile.

Organizations have to be agile about being agile. They have to try out a set of practices, but quickly make adjustments to find out what actually works. It is critical to apply practices that are prudent based on the specific context of the organization, the team, and the environment in order to succeed with agile development.
Java Magazine: To what degree are problems applying agile tied to the overall culture of companies?
Subramaniam: Culture makes a big difference, but there is often not one, company-wide culture in large organizations. I've walked across a large floor at a company and come across multiple different cultures.

Separate teams within the same company might have different ways of responding to change, criticism, and feedback. Some of them might be able to adapt to agile development more easily than others.
Java Magazine: Are there certain kinds of developers who are not well suited to agile development?
Subramaniam: Agile is not well suited if the team would rather work in silos. The Dreyfus model of skills acquisition rates people on a scale of one to five, with *five* being people who glow in the dark—these are folks who get things done with no guidance or

The Agile Manifesto

In February 2001, 17 developers gathered at a resort in Utah to explore how best to produce software. The result, the "Manifesto for Agile Software Development," stated, in its entirety:

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more."

In addition, they offered 12 basic agile principles that ranged from the importance of change and of working software, to the value of simplicity, sustainable development, and face-to-face communication.

intervention. *One* consists of novices; leave them alone with a task and they might struggle to get moving. Each of us varies on this scale for various activities we perform. So there is no level-one person or level-five person—we're at various levels for different activities. The study says the average level in any organization is level two, which is not sufficient for excelling. We can raise this level for a team through active collaboration and open communication.

Some developers are naturally open to communication and constructive criticism. Some developers simply are not, and that can be a problem. There's a cautious balance we have to strike between personal pride and team spirit. Ego is like cholesterol—there are good parts and bad parts. We all want to take pride in our individual work, while at the same time, we have to place the success of the team ahead of that. It comes down to openness and accepting trade-offs. The agile methods work better in teams in which developers put forth their ideas, not as a fort to defend, but as something that should evolve and be improved upon by their team.

Java Magazine: One of the original principles of agile emphasized that face-to-face conversation is the best form of communication. This method emphasizes both personal interaction and conversation, which implies relationships of relative equality.

Subramaniam: Yes, it does. I have had many very productive relationships working with people who I have never met in person, or only met years later, so face-to-face conversations are by no means absolutely necessary for good communication. It is a matter of the attitudes that people bring to the interaction. Two people, no matter how far away they are, will produce better results if they're keen on working together and value the collective success. More than on face-to-face communication, we need to focus on the trust we have built.

If I whine and complain about someone to you, you're going to wonder what I say about you to others. This erodes trust. Having a face-to-face conversation is not going to make this any better. A quick check on our attitudes and levels of trust is the first step. Given all things equal, I do believe that face-to-face conversation brings better results. But all things are rarely equal.

Java Magazine: Your most recent book, *Functional Programming in Java: Harnessing the Power of Java 8 Lambda Expressions* (Pragmatic Programmers, 2013), points to a different way of doing Java programming. How might this relate to agile development?

Subramaniam: Lambda expressions in Java 8 open up the possibility of functional programming in Java. This style of programming, if done well, can lead to fewer errors, more-elegant and more-concise code, and code that is far easier to parallelize. Fewer moving parts in functional-style code means it's easier to write automated tests and easier to evolve the highly expressive code. This can lead to code that's ready for feedback more rapidly than in the past. Agile development, in essence, is feedback-driven development. Collaboratively and skillfully adjusting to feedback cycles through iterative, incremental development is the essence of agile. `</article>`

Timothy Beneke is a freelance writer and editor, best known for his books on gender.

LEARN MORE

• Venkat Subramaniam's blog

Diabolical Java Performance Tuning

Lessons in what *not* to do.

BEN EVANS AND
MARTIJN VERBURG

BIO

PHOTOGRAPHS BY
JOHN BLYTHE AND BOB ADLER

In this article, we'll discuss how to approach performance tuning and wrestle with the #1 problem that all applications need to be concerned with.

Never mind about reliability, supportability, elegance, architectural flexibility, ease of understanding, or even correctness. All that matters is getting a result out quickly. Who cares if you're right tomorrow? All that matters is getting the result (any result) as quickly as possible—if you're faster than everyone else, you can define what's "right." Smart developers focus on performance to the exclusion of all else.

Using Benchmarks

Always believe the benchmarks you find online—what could go wrong? Following that, it's also important to understand that benchmarks always correlate really well with real-world application performance (even Wikipedia agrees).

Microbenchmarks are the best type of benchmark. They're fun to do, and once you understand the low-level detail of your application's behavior, it's easy to just extrapolate up from that and deduce how the rest of the stack will behave.

The best kind of microbenchmark is where you can prove that some very low-level aspect of the Java Virtual Machine (JVM) has some tiny performance differences—these types of results are always significant when aggregated up into a whole system. For example, it stands to reason that dispatch of an interface method absolutely must be slower than regular virtual dispatch, right?

Handling Data and Results

The Don't Repeat Yourself (DRY) principle is an important part of modern software development. In a performance context, it means that you should never run any performance test more than once. With so many other development tasks, it's important not to waste time on the dull work of repeating a performance test. After all, you're a perfectly competent professional, so you're bound to have set up the test perfectly and gotten all the bugs out of the run the first time around.

Ignore anyone who says things such as "statistical significance" or "shape of the distribution." They're probably just math nerds who can't cope with the awesomeness required to be a real performance rock star ninja. Do the test once and just measure the averages (and by *averages*, we mean the mean), and you'll be done handling those annoying results. Then you can get back to the real work of a performance engineer—squeezing every last glistening drop of performance out of the algorithms in your application.

Measuring things is for people who don't have your keen insights; you built the app, so you know exactly where the problems are. Algorithm optimization is hard, so that's where the real performance professionals spend their time; never mind collecting and analyzing data.

Optimizing Algorithms

The bottleneck is almost certainly where you think it is. Trust in your amazing analytical skills—they will lead you to the right culprit, which is usually any code that wasn't written by you.

Fortunately, optimizations of other people's code are easy to do. First, find any code with a simple, unoptimized algorithm. A good candidate is anything that has a long-running loop with a conditional check in the middle of it. Now, remember that general-case algorithms are stupid. You understand your data set better than anyone else, so you should spend as much time as you need to design a specialized algorithm that fits your data. Someone of your talent and intellect can surely do better than mediocre, compromised textbook examples.

Don't worry if this takes quite a lot of time; this is the most important part of performance tuning, and doing it correctly is likely to require someone of your abilities.

Optimizing the JVM

Don't bother testing with recent releases of Java and tracking the minor version numbers. Nothing much changes internally between major releases of Java.

Instead, you can often get better performance by fiddling with the switches you pass to Java. Don't bother being systematic about this—just locate a set of switches that you think have some bearing on the problem, and play around with them until you find a combination that works for you and will give you extra, free performance with no downsides.

There are more than 100 performance tuning switches for the JVM. Use as many as you can. Don't worry that some of the switches seem to do conflicting or contradictory things. A true professional (such as you) will also use the undocumented ones found in the Java HotSpot VM source code.

Understanding the Hardware

Understanding the CPU is a waste of time—it's the kind of thing that is best left to hardware engineers. A software engineering genius shouldn't have to care about L3 caches or how the data is actually laid out in memory.

Solid-state drives are faster in all use cases, so use them all the time. Don't bother to measure the actual throughput or other observables. Remember, more capacity is better, so buy loads of RAM and really push up the size of your heap.

Using Performance Advice

Performance advice is like fine wine: it just gets better with age, and it certainly never goes out of date. Performance advice from the 1990s is still absolutely accurate today. You should build on the deep insights of the amazingly smart engineers who came up with the tips in the first place. They were probably a lot like you.

Working with the Team

Always remember to guard your performance test harnesses and results carefully. Otherwise, some other people might steal your work and claim credit for it. Never let them independently recheck your work. They'll just get it wrong and detract from your obviously correct conclusions.

Better yet, let others do the boring work for you. If someone else has already done something vaguely similar and written it up on a blog or the Stack Overflow site (especially if they have a high reputation score), just accept their conclusions—that's close enough. After all, how different can the behavior of two applications be? It's all just Java (or all just the JVM), right?

Once you've reached your conclusions about how to improve performance, whether that involves using an optimized algorithm or a new bit of ultrafast shiny tech, it's a better use of your time to browbeat your colleagues into accepting your solution than to waste time having them independently check your conclusions. Always show them who's boss, and never document or explain your reasoning—the last thing you want is to have them messing around with your carefully tuned system.

Scrutinizing Technology

When thinking about the application architecture as a whole, always remember to scrutinize the newest technology very closely. If you don't fully understand it (bearing in mind that you're a genius), there's bound to be something not quite right about it.

Conclusion

Performance tuning can be boiled down to a few very simple rules:

- Performance is the most important aspect of any application.
- You are awesome, which is why you're doing the tuning—it's the most important job of all.
- Performance analysis is all about staring at source code and optimizing algorithms.
- Other people will likely mess up your good work, so guard it from them jealously.
- Measuring is boring and unnecessary.

If you follow these basic rules, your application will be on the front page of major industry publications in record time.

Needless to say, the diabolical advice in this article should not be followed, but instead should be referenced as an anti-patterns guide. </article>

LEARN MORE

- "Java Performance Tuning"
- "Nine Fallacies of Java Performance"

JAVA CONCURRENT ANIMATED

Java Champion **Victor Grazi** on writing better concurrency applications **BY JANICE J. HEISS**

BEST DEVELOPMENT PRACTICES

In 2009, Java developer Victor Grazi introduced Java Concurrent Animated, a much-praised set of animations that serves as a tutorial for concurrency development. Recently inducted as a Java Champion for his contributions to the Java community, Grazi is a vice president at JP Morgan Chase's Securities Lending division. He is a frequent presenter at technical conferences where he speaks about his first love, Java concurrency, and other Java-related topics.

Java Magazine: How many people have now tried out your Java Concurrent Animated app?

Grazi: The app was introduced in July 2009; since then, we've had about 20,000 downloads. Given that there are perhaps 10 million Java developers out there, we are only scratching the surface. Top downloads by country are the US (23 percent), India (14 percent), and China (7 percent).

You can download the self-executing JAR file. Then just double-click the JAR file to start it. It is menu-driven, or you can use the up and down arrow keys

PHOTOGRAPHY BY CHRISTOPHER LANE/GETTY IMAGES

to navigate between the various image and animation slides. It works on all platforms—Windows, Mac, Linux, and so on. Java SE 6 or higher is now required.

Java Magazine: What are the typical responses to it?

Grazi: People tell me that they find it very helpful. Many really get excited about it, especially teachers and team leads who are trying to impart proper concurrency skills to teams. Java was one of the first languages to introduce concurrency into its core library. It was a powerful feature, but we suddenly found some very good programmers writing some very bad code. Proper concurrent programming is difficult to impossible to get right, but if people would take the time to understand some of the frameworks out there, it would make coding for concurrency much less error prone.

Take, for example, the Java memory model. Developers often ignore it and code away blissfully ignorant that their code is broken, because the Java Virtual Machine (JVM) and servers might not leverage the optimizations provided by the Java memory model. But as cores increase in speed and number, manufacturers are expected to take advantage of these efficiencies, and suddenly code that has been working like a charm will start experiencing sporadic failures due to incorrect concurrency management.

MORE THAN IT SEEMS

Java Concurrent Animated is not just some Flash animations; it is a set of interactive Java programs, such that each animation is actually utilizing the underlying concurrency component it is illustrating.

Java Magazine: Would you say that your app does some of our imagining for us in ways that enable developers to intuitively grasp Java concurrency principles and processes more quickly?

Grazi: That's an interesting way to put it. You see, Java Concurrent Animated is not just some Flash animations—it is a set of interactive Java programs, such that each animation is actually utilizing the underlying concurrency component it is illustrating. There is a snippet panel on the right side of the screen, and as an animation proceeds, executing code snippets are dynamically highlighted and restored as they execute.

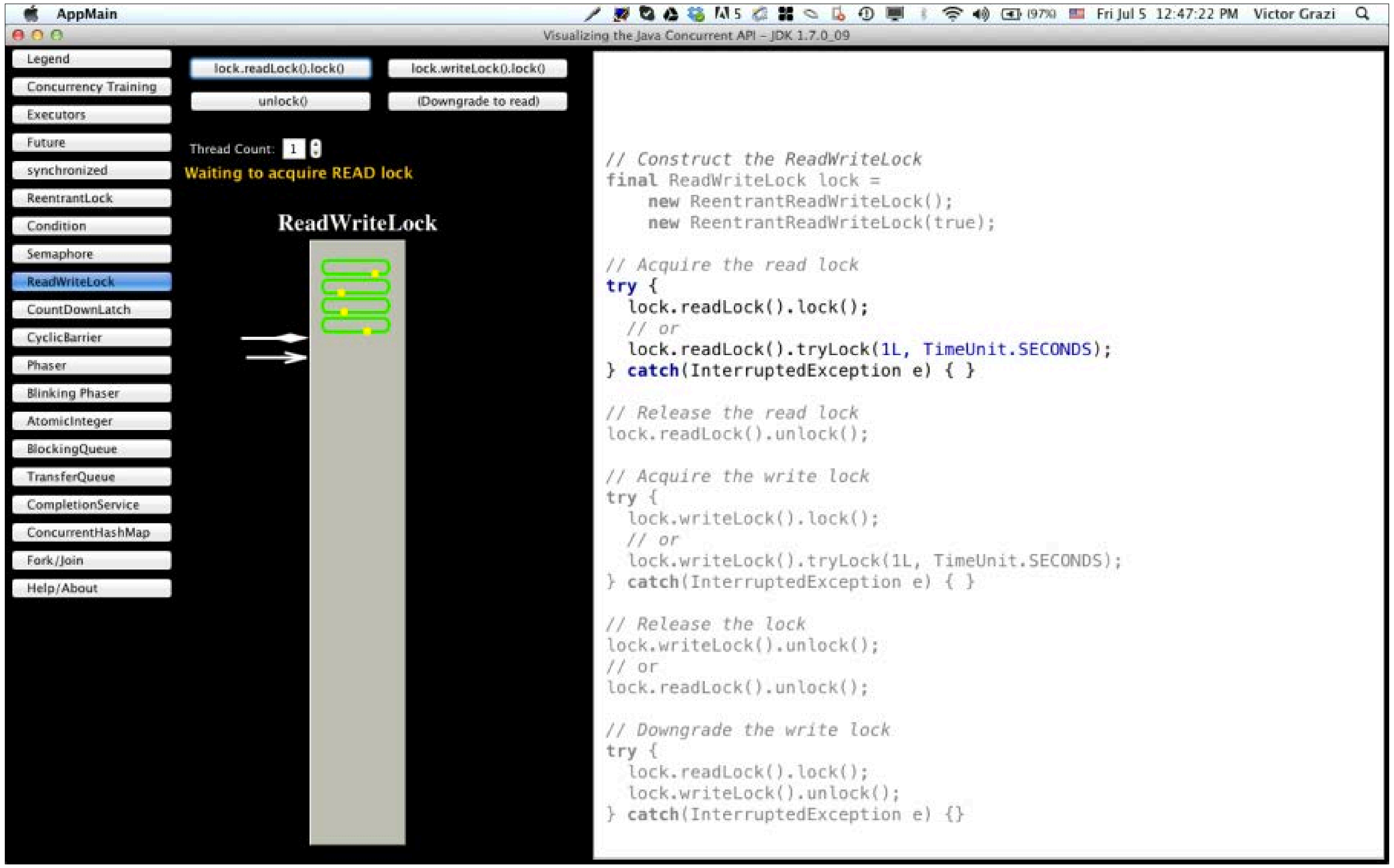
Let me give you an example that happened with the ReadWriteLock animation. A ReadWriteLock is used to ensure data consistency. It allows an unlimited number of reader threads to acquire a read lock and operate concurrently. But a writer thread must wait for all readers to complete before it can acquire the lock. Once a writer thread acquires the lock, no other reader or writer can acquire it.

Suppose a writer is waiting for some working readers to release the read lock, and suddenly a new reader comes along. Who should get preference? Should the reader move ahead of the writers—after all, why have a new reader wait around for a waiting writer if other readers are already holding the lock? So that was how it worked in Java 5. But once I started running the animation in Java 6, I noticed the behavior changed, and subsequent readers would wait for all waiting writers to release the lock before acquiring the lock.

Available Animations

Available Java Concurrent Animated animations include Executors [FixedThreadPoolExecutor, SingleThreadExecutor, CachedThreadPoolExector, and RejectedExecutionHandler(s)]; Future; synchronized; ReentrantLock; Condition; Semaphore; ReadWriteLock; CountDownLatch; CyclicBarrier; Phaser; AtomicInteger; BlockingQueue; TransferQueue; CompletionService; ConcurrentHashMap; and ForkJoinPool.

In addition, Grazi has an animation for the Java memory model and new Java 8 constructs in the works.



Waiting to acquire READ lock

```
// Construct the ReadWriteLock
final ReadWriteLock lock =
    new ReentrantReadWriteLock();
    new ReentrantReadWriteLock(true);

// Acquire the read lock
try {
  lock.readLock().lock();
  // or
  lock.readLock().tryLock(1L, TimeUnit.SECONDS);
} catch(InterruptedException e) { }

// Release the read lock
lock.readLock().unlock();

// Acquire the write lock
try {
  lock.writeLock().lock();
  // or
  lock.writeLock().tryLock(1L, TimeUnit.SECONDS);
} catch(InterruptedException e) { }

// Release the lock
lock.writeLock().unlock();
// or
lock.readLock().unlock();

// Downgrade the write lock
try {
  lock.readLock().lock();
  lock.writeLock().unlock();
} catch(InterruptedException e) {}
```

Figure 1

I thought this new behavior was a bug and mentioned it to concurrency expert Dr. Heinz Kabutz, who explained that it was not a bug but a feature. If new readers were allowed to jump ahead of waiting writers, there is a high risk of producing a thread-starvation condition because there is a high probability that no writers would ever get in, and they would accumulate forever. That's an example of how the animations alter their behavior depending on the JVM runtime version.

Java Magazine: What is distinctive about Java concurrency programming that makes an animated tutorial of particular value?

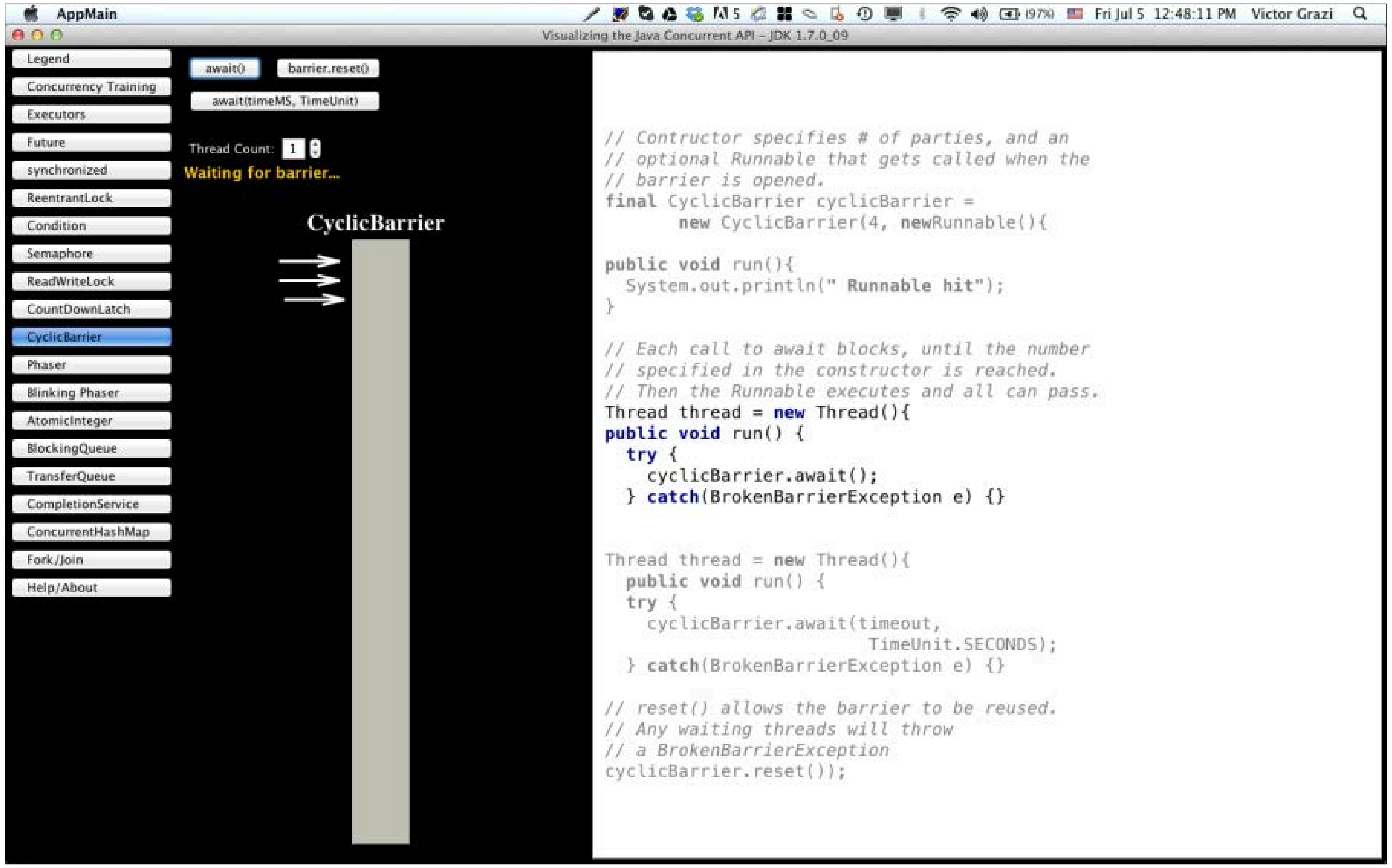
Grazi: Miller's Law teaches that there is a limit to the number of concepts our brains can hold at one time. The human brain tends to process sequentially, so even if we overcome our physical limitations and get things right, it is difficult to come back later and try to reconstruct our prior thought processes. Certainly, if a different developer dives in later, it is that much harder to recapture the cognitive circuits from the original effort. So, suddenly, brittle code starts breaking.

By using a framework, we are not only delegating the concurrency to the smart people who built and maintain the framework, but we are also employing a lexicon for communicating our design. So, I can say, "The following code is operating as a CyclicBarrier," and people will understand what that means. By introducing interactive animations for all the java.util.concurrent components, developers can click buttons to easily visualize the functionality they are exploring, making it much easier to viscerally assimilate the algorithms.

Java Magazine: You were operating on some intuitions about what would make it easier to learn concurrent programming—intuitions that appear to be valid, judging from the response of developers. Can you make those intuitions explicit?

Grazi: Well, yes. For example, above I explained the basic workings of a ReadWriteLock. Readers might have understood it or not. Now take a look at the animation for that, which is shown in **Figure 1**.

The green threads are the readers, the top white thread (with the diamond-head) is a writer, and the



```
// Contructor specifies # of parties, and an
// optional Runnable that gets called when the
// barrier is opened.
final CyclicBarrier cyclicBarrier =
      new CyclicBarrier(4, newRunnable(){

public void run(){
  System.out.println(" Runnable hit");
}

// Each call to await blocks, until the number
// specified in the constructor is reached.
// Then the Runnable executes and all can pass.
Thread thread = new Thread(){
public void run() {
  try {
    cyclicBarrier.await();
  } catch(BrokenBarrierException e) {}


Thread thread = new Thread(){
  public void run() {
  try {
    cyclicBarrier.await(timeout,
                    TimeUnit.SECONDS);
  } catch(BrokenBarrierException e) {}

// reset() allows the barrier to be reused.
// Any waiting threads will throw
// a BrokenBarrierException
cyclicBarrier.reset());
```

Figure 2

white thread beneath it is a new reader, which must wait for the readers and writer to finish before acquiring the lock. If you click buttons and see the animations, it becomes much easier to understand than parsing through some turgid explanation.

Java Magazine: Heinz Kabutz has remarked that Java was built to be able to do many things at once, which is what concurrency is all about. How does your learning system improve the skills of programmers so that they have less risk of concurrency bugs?

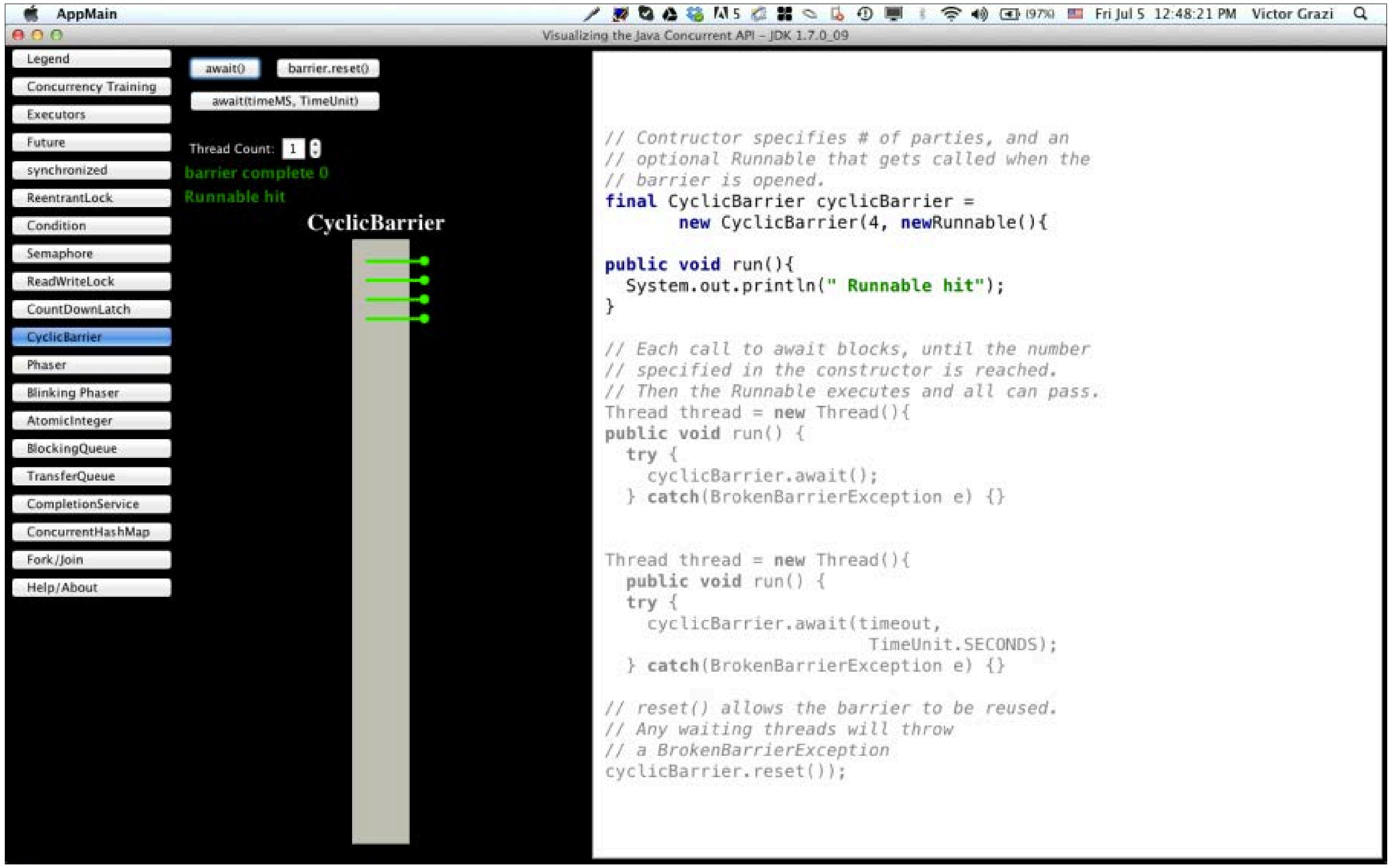
Grazi: Very often when I am grappling with a concurrency problem, I know that the solution lies in some design pattern, but which one? Java Concurrent Animated provides a catalog for developers in their quest for the right concurrency solution; it acts as a muse that inspires the correct solution.

Java Magazine: Java Concurrent Animated had its origins when you were managing a development team that was using Java concurrency and you wanted to understand it better and have your team understand it better. Can you describe the process by which this led you to the animations?

Grazi: My training is in server-side Java at investment institutions, where concurrency is a common concern. Traders require low latency to ensure that they won't lose to a competitor in a 1 millisecond window of opportunity for grabbing a trade. Batches need to be completed quickly, and so on. So I started seeing horrific write-only constructs that confirmed that people were struggling with concurrency. I have been there myself.

One afternoon, while I was sitting in an airport heading to Chicago to make a concurrency presentation for my team, I was putting the finishing touches on a PowerPoint presentation that had a sequence of slides dedicated to each of the important constructs. I had written a little functioning state machine that displayed simple text messages that I would refer to in order to guide me through the important states of each concurrency construct in java.util.concurrent. In a previous lifetime, I had worked at an interactive gaming dot-com startup, so I knew a lot about animation. It occurred to me that I could replace the PowerPoint slides with a series of interactive animations that would be much more intuitive.

I worked on an initial animation



```
// Contructor specifies # of parties, and an
// optional Runnable that gets called when the
// barrier is opened.
final CyclicBarrier cyclicBarrier =
       new CyclicBarrier(4, newRunnable(){

public void run(){
  System.out.println(" Runnable hit");
}

// Each call to await blocks, until the number
// specified in the constructor is reached.
// Then the Runnable executes and all can pass.
Thread thread = new Thread(){
public void run() {
  try {
    cyclicBarrier.await();
  } catch(BrokenBarrierException e) {}


Thread thread = new Thread(){
  public void run() {
  try {
    cyclicBarrier.await(timeout,
                        TimeUnit.SECONDS);
  } catch(BrokenBarrierException e) {}

// reset() allows the barrier to be reused.
// Any waiting threads will throw
// a BrokenBarrierException
cyclicBarrier.reset());
```

Figure 3

engine while waiting for that flight, and then I hooked in my state machine. By morning I had a functioning prototype. Over the years, I have tightened the framework, incorporating suggestions from experts. I sent an early version to Brian Goetz and was surprised to receive suggestions for each animation. I incorporated all of his suggestions. Kirk Pepperdine joined me at my first presentation at JavaOne and suggested adding some true PowerPoint explanations to the animations so that presenters have a trigger to remember what to discuss. I added that, and it is a very helpful feature—not just for presenters but even for end users. Heinz Kabutz also joined the presentation and suggested some changes to the original animations that would make them more intuitive.

At another presentation, a passionate software consultant named Oliver Zeigermann pointed out that an animation for ConcurrentHashMap was conspicuously absent. I asked him if he would be interested in contributing it and he made that valuable addition.

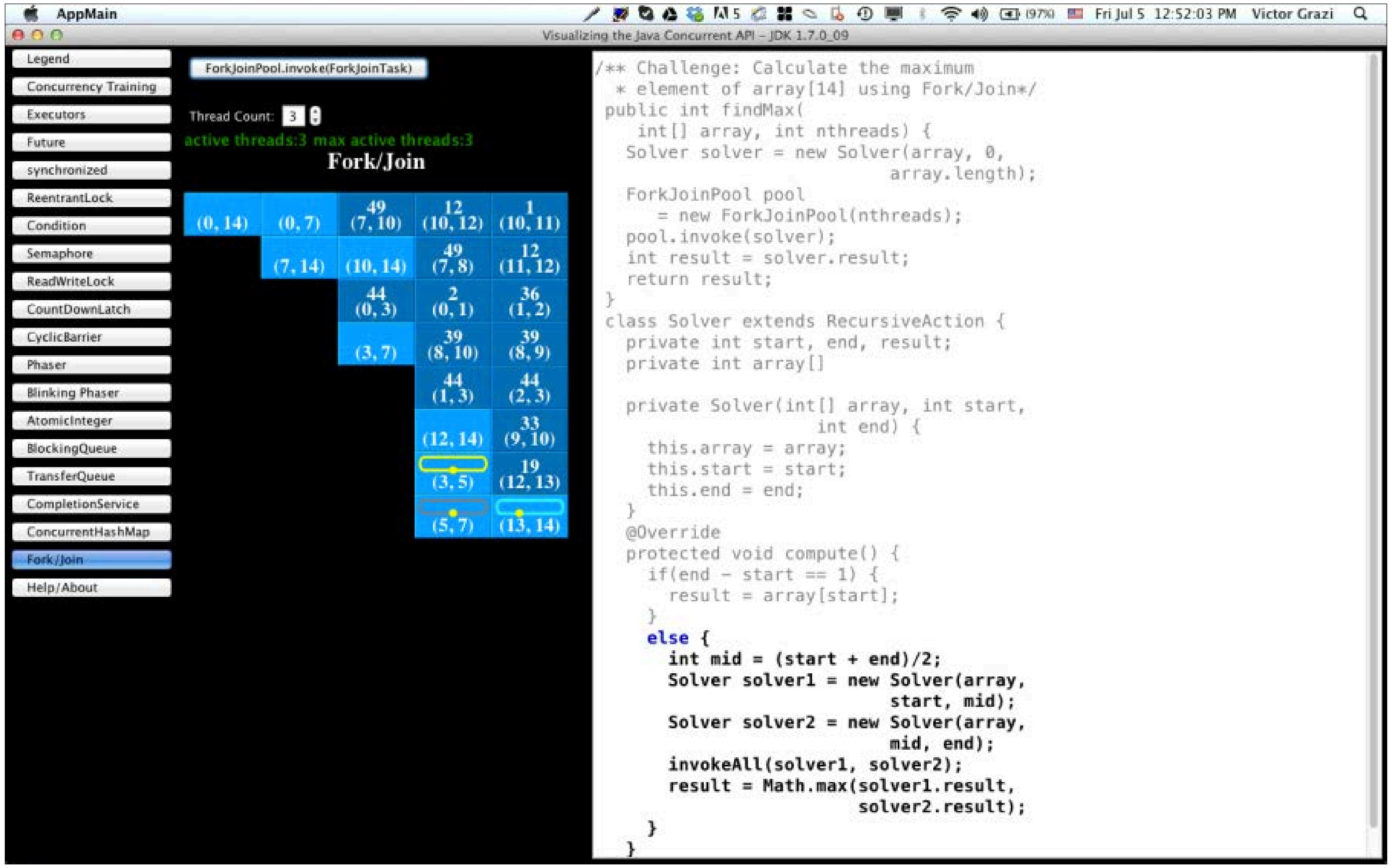
Java Magazine: Could you take us through some of the classes of Java concurrency and explain how animation makes it easier for developers to get insight into each one?

Grazi: Well, it would be difficult to do that without an animation, but let's take a look at the CyclicBarrier, which has two important states, as shown in **Figure 2** and **Figure 3**, which illustrate a barrier with four parties. In **Figure 2**, we can see that only three parties have arrived, so they are prevented from proceeding. **Figure 3** shows that once the fourth party arrives, everyone moves forward.

So this illustrates the barrier concept, which is that each party must wait until all parties have arrived. As the constructs increase in complexity—for example, the fork/join animation or the animation demonstrating the native wait and notify mechanism—the benefits are more pronounced.

Java Magazine: Tell us about some of the challenges you have faced in creating the animations.

Grazi: There have been several. Originally, the animations represented threads as arrows, which worked for most of the components. But then we had to provide a visualization not for threads but for objects being queued



```
/** Challenge: Calculate the maximum
 * element of array[14] using Fork/Join*/
public int findMax(
   int[] array, int nthreads) {
  Solver solver = new Solver(array, 0,
                         array.length);
  ForkJoinPool pool
     = new ForkJoinPool(nthreads);
  pool.invoke(solver);
  int result = solver.result;
  return result;
}
class Solver extends RecursiveAction {
  private int start, end, result;
  private int array[]

  private Solver(int[] array, int start,
                      int end) {
    this.array = array;
    this.start = start;
    this.end = end;
  }
  @Override
  protected void compute() {
    if(end - start == 1) {
      result = array[start];
    }
    else {
      int mid = (start + end)/2;
      Solver solver1 = new Solver(array,
                         start, mid);
      Solver solver2 = new Solver(array,
                         mid, end);
      invokeAll(solver1, solver2);
      result = Math.max(solver1.result,
                      solver2.result);
    }
  }
```

Figure 4

in a BlockingQueue. So, I had to introduce a concept of "sprite-type," and then we had an arrow sprite-type and a new "object" sprite-type that is rendered as an oval, not an arrow. Then ConcurrentHashMap and AtomicInteger required a new sprite-type, because we are trying to visualize their compare and swap activities.

Then fork/join came, and the challenge was how to represent something completely different using the existing framework. There was an additional challenge there. The fork/join animation needed to solve a real problem, but which problem should the animation solve?

I originally had the animations solve for the nth Fibonacci number, but that was not working. I wrestled with that for a couple of days until I realized that the recursive definition of the Fibonacci sequence (Fn+1 = Fn + Fn-1) cannot be efficiently parallelized, because each value depends on the prior values. Therefore, it is inherently a sequential calculation no matter how you try to parallelize it. So, I switched to a different problem—that of finding the greatest element in an array, which worked nicely. During the animation, you can see the actual problem being solved using random array values (see **Figure 4**).

Java Magazine: Where has your animation been presented?

Grazi: I presented it at JavaOne a few times and also at many other conferences—such as JavaZone in Oslo, Jazoon in Zurich, QCon in New York—and at many SIGs [special interest groups] and JUGs [Java user groups]. I love presenting and I love traveling around the world, so Java Concurrent Animated has provided me with a great opportunity to do both. It always gets high ratings.

The Java Concurrent Animated presentations provide awareness, and they also show the attending developers the value of downloading the app and how much easier learning concurrency can be if you have a framework and an inspiration. </article>

Janice J. Heiss is the Java acquisitions editor for *Java Magazine* and Oracle Technology Network.

LEARN MORE

• Java Concurrent Animated

Database DevOps with MySQL, Hudson, Gradle, Maven, and Git

DevOps will help developers produce higher-quality software.

MICHAEL HÜTTERMANN

BIO

The term *DevOps* describes the improved collaboration between development and operations teams. In software engineering, databases are often on a critical path. This article describes DevOps and explains what database DevOps might look like using concrete concepts and tools.

So first, let's start with conflicts that occur during software engineering.

Conflicts Due to Different Goals, Processes, and Tools

Often conflicts exist between development and operations teams, primarily for the following reasons:

- Different goals: For developers, more changes in a short time; for operations staff, fewer changes in production and, thus, more stability
- Different processes and concepts: For developers, more-pragmatic approaches; for operations staff, more focus on reproducibility
- Different tools: For developers, development tools; for operations staff, more production-like approaches

These differences often lead to gaps or silos between departments, which I'll cover next.

Different Teams

In traditional settings, the term *development* describes mainly the programmers on the development team (see **Figure 1**). Testers and QA personnel are also part of the team, but they often have dedicated project roles, and their activities start after the programmers have finished their work.

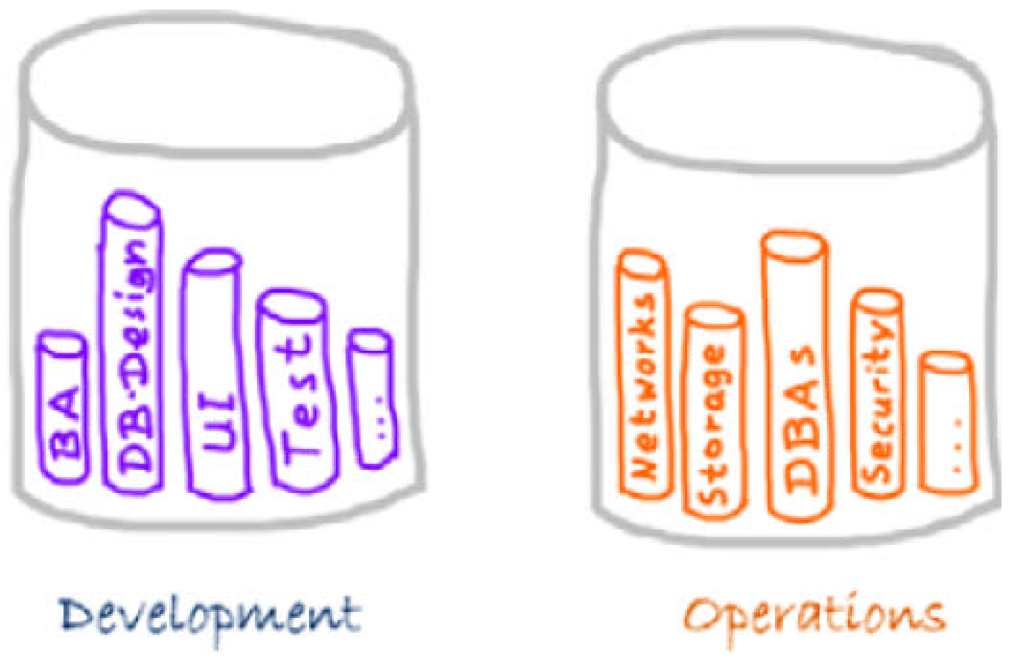


Figure 1

The term *operations* refers to a team that comprises database administrators, system administrators, network administrators, and other types of administrators. These are the experts who put the software into production and manage the production infrastructure (for example, setting up and maintaining the servers and systems). The operations group essentially accompanies and accounts for the "last mile" in the delivery process.

In a barrier-rich setting, both groups form silos, in which they have locally optimized goals and their own processes and tools.

Developing software is hard; developing databases is harder.

Database on the Critical Path

Let me give some concrete examples of what daily struggles might look like for database development. Application developers apply continuous integration, including frequent check-ins and automated testing, and they often apply continuous deployment of the business application to the target environment. Database developers, on the other hand, often lack the basics for real database version control and continuous deployment.

This gap is created by the major differences between application development/deployment and database development/deployment. Traditionally, application development is based on local files, with local changes being published only upon check-in. Developers can change and debug code locally, without interfering with the work being done by other team members. Deployment is performed by automatically copying deliverables from the build server to respective environments.

Database development, on the other hand, is often based on a central resource. In many cases, though, local developer databases or individual schemas in central databases are used to provide isolated and productive working environments.

In addition, database deployment is not a straightforward process of copying and replacing. For example, a database table cannot be simply dropped and afterward recreated with the new structure. With database deployments, often no two deployments are exactly the same, since either the source or the target has been altered or updated by previous deployments or new developments.

DevOps can help to streamline software development, including managing changes on databases, in a holistic way, spanning different departments. Now let's examine in more detail what DevOps is all about.

DevOps in a Nutshell

DevOps describes practices that streamline the software delivery process, emphasizing learning by streaming feedback from production to development and improving the cycle time (that is, the time from inception to delivery). DevOps will not only empower you to deliver software more quickly, but it will also help you to produce higher-quality software that is more aligned with individual requirements and basic conditions.

DevOps seeks to align goals, concepts, and tools with each other for both development and operations. DevOps is about improving the collaboration of development and operations (the why) through shared goals, concepts, and tools (the how).

With DevOps, organizational barriers are minimized. With the "one team approach," the usage of *agile* practices is expanded to operations. Experts from development and operations are both now "developers," meaning that they work together closely and help to "develop" the solution.

DevOps targets different activities and aspects.

Numerous Activities and Aspects

DevOps encompasses numerous activities and aspects, such as the following:

- *Culture*. This concept emphasizes people over processes and tools. Software is made by and for people.
- *Automation*. Automation is essential for DevOps to gain quick feedback.
- *Measurement*. DevOps finds a specific path to measurement. Quality and shared (or at least aligned) incentives are critical.
- *Sharing*. Sharing creates a collaboration platform for exchanging ideas, knowledge, and experience.

WHY THE CONFLICT?

Often conflicts exist between development and operations teams—primarily due to different goals, different processes, and different tools—which lead to gaps or silos between departments.

In defining and rolling out database DevOps, it can be helpful to distinguish among four different areas. **Figure 2** shows the DevOps area matrix approach. Area 1 is about extending development to operations. A common use case for this, in a database context, is to put conversion scripts into the version control system and use the same database migration tool in development and operations, such as Flyway, which we'll discuss a bit later.

Area 2 is about extending operations to development. For database DevOps, this means providing visibility for traffic on production systems, including locked rows, blocking queries, and resource contention.

Area 3 embeds development into

operations. Examples include setting up constraints and shared goals for nonfunctional requirements. Examples of shared goals are that 80 percent of database searches will return results to the screen in less than two seconds (a shared performance goal); the system shall not make use of any technology that would make it difficult to port to another Linux distribution (a shared portability goal); the database will be capable of storing 20 million members on the specified hardware while still meeting performance objectives (a shared capacity goal); or automated tests must exist for all components including infrastructure code (a shared maintainability goal).

Area 4 embeds operations into development. This can be done to enhance collaboration by providing access to information to development without the involvement of database administrators (DBAs), thus preventing DBAs from being gatekeepers.

Database DevOps

A robust database change-management solution is the most effective way to overcome daily challenges. With features such as version control, continuous integration, and automation, database change management enables DBAs and developers to better communicate and collaborate with each other and to avoid the potential pitfalls—accidental overrides, conflicts, and so on—that can arise when they work in silos. That, in turn, will lead to greater returns from DevOps strategies. The following patterns might help to foster DevOps, particularly database DevOps.

WHY SILOS?

In a barrier-rich setting, both groups form silos, in which they have locally optimized goals and their own processes and tools.

Use database update scripts. With DevOps, database elements should be released automatically, with update scripts. It is a good idea to distinguish update scripts between database *expansion scripts* and *contraction scripts*.

Expansion scripts involve database changes that can be applied without breaking the database's backward compatibility with the existing version of the application (for example, adding elements, such as new tables or columns). These scripts can run at any point before upgrading the corresponding application.

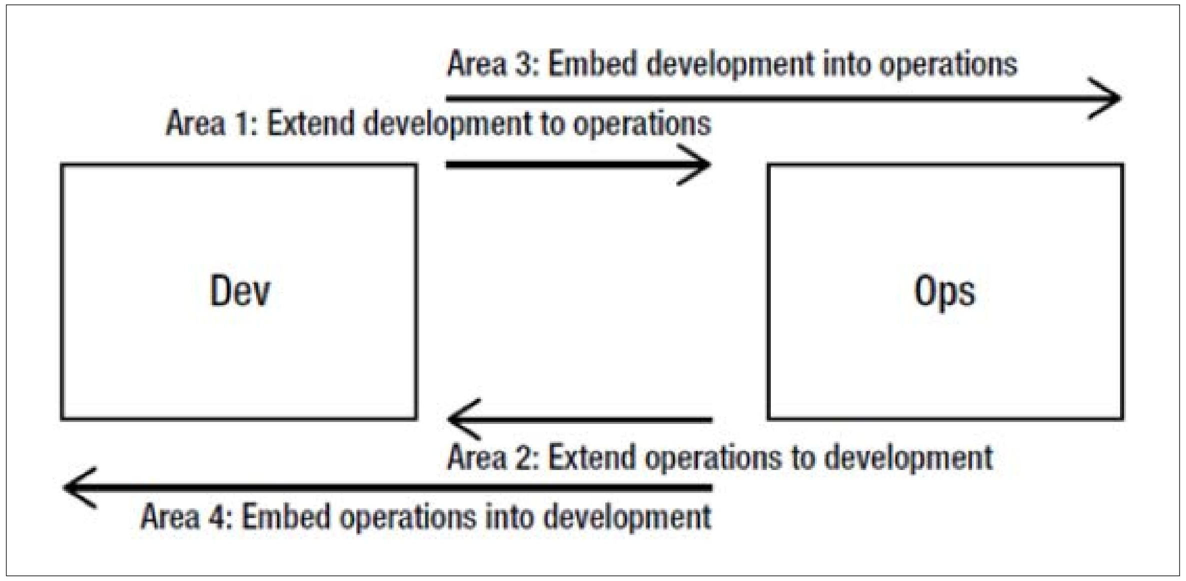

Area 3: Embed development into operations
Area 1: Extend development to operations
Dev
Ops
Area 2: Extend operations to development
Area 4: Embed operations into development

Figure 2

Contraction scripts migrate the database and break backward compatibility (for example, removing structure).

Using expansion and contraction scripts conveniently decouples database migrations from application deployments.

Release databases automatically. One of the more-advanced challenges in automatically releasing databases is to link the database in its current version (that is, the current set of structural elements, such as tables and columns, and their data)—or, in other words, in its current state—with the current version of the rest of what makes up the complete release. By having database elements already under version control, you can create tags (often called labels) and add all of your configuration items to a defined baseline.

Automatically deploying database changes results in the need for a process that supports applying database changes incrementally while preserving current structure and content. Many approaches exist for updating an existing database, and all have the following activities in common:

- Put all code and database elements (all change sets) into version control.
- Create SQL scripts that have to be applied to roll forward to the next version and to roll backward to the previous version of the database. These scripts are grouped into single change sets.
- Investigate whether a roll backward mechanism is needed at all. Increased complexity can be avoided by accelerating the

development flow and is a good strategy for applying hot fixes.

- Create one file for each change set to hold the respective change set. Give the file a unique name including a numbering scheme. Change sets move the database forward (or backward). Change sets are applied based on a baseline. Thus, the concrete content of a change set might contradict a previous change set. In other words, each change set can consist of multiple SQL statements. For managing one specific change set, favor one file over multiple files, to foster task-based development.
- Create baselines in which you freeze all the configuration items of your application, including the database elements.
- Retrieve the baseline for deployment. In cases of a full installation, apply again the initial baseline of database elements and the sum of incremental change sets. In cases of incremental installation, check the current state (the version) of the specific database and apply all new change sets (that have not been applied before) to it.
- Ensure that the deployment process picks up a baseline from version control. Place the database change sets into the dedicated working directory.
- Store the database version. One approach is to use a database table that holds the meta-information, especially the version of the database scheme. Additionally, you can create columns for SQL scripts that have to be applied to roll forward and to roll backward. Shell scripts can now roll backward or forward using the information held in the database.
- Consider monitoring to minimize mean time to repair (MTTR), mean time to detect (MTTD), and smoke testing.

You can develop a solution yourself, or you can use a framework such as Flyway. In the upcoming sections, we'll go through an example of using Flyway in an integrative build chain, integrating Maven, Gradle, Git, and Hudson.

Flyway, the Database Migration Tool

Flyway is a database migration tool that supports the concepts mentioned earlier. It can be used by both development and operations. Actually, development has to write the change sets and has to test the deployment on test machines. Flyway can migrate from any version (including an empty database) to the latest version of the schema. It's based on plain old SQL; thus, it's a good discussion base for both development and operations, as well as being a solution that is light and straightforward.

Flyway only has a few dependencies; you need Java 5 or higher and a JDBC driver. It applies the pattern of convention over configuration, for example, to automatically discover SQL migrations through classpath scanning. It supports many databases, including Oracle Database 10g and higher (all editions, including Oracle Database, Express Edition) and MySQL 5.1 and higher. Data definition language (DDL) files exported by Oracle Database or MySQL can be used unchanged in a Flyway migration.

Any Oracle Database and MySQL script executed by Flyway can be executed by SQL*Plus and other Oracle-compatible tools (after the placeholders have been replaced). Flyway can be executed in many ways, including via a Maven plug-in, Gradle, or Ant. It can also be executed from the command line. Now let's focus on the Flyway support for MySQL.

AVOIDING CONFLICT
With features such as database version control, continuous integration, and automation, database change management enables DBAs and developers to better communicate and collaborate with each other and to avoid the potential pitfalls—accidental overrides, conflicts, and so on—that can arise when they work in silos.

Flyway and MySQL. Flyway supports the standard SQL syntax with statement delimiters, including delimiter changes for stored procedures using DELIMITER statements. You can use comment directives generated by mysqldump (/!.../;) and MySQL-style single-line comments (# Comment).

DDL exported by mysqldump can be used unchanged in a Flyway migration. Any MySQL SQL script executed by Flyway can be executed by the MySQL command-line tool and other MySQL-compatible tools (after any placeholders have been replaced, which allow parameterized calls and configuration).

A Concrete Example

Now for an example. Let's start by looking at the folder where the migration scripts are located. They are placed in the resources folder in our project source tree (see **Listing 1**).

At the moment, we have two files of plain SQL. The first file is V1__Create_person_table.sql. It is a DDL file and has the content shown in **Listing 2**.

The second file is a data manipulation language (DML) file. The file V2__Insert_persons.sql consists of three statements for inserting three rows into the previously created table (see **Listing 3**).

Now let's look at the Maven project object model (POM), the metainformation file of the Maven build tool for our project. This is the place to define how Flyway will be used and to tell the system about the database it should connect to. The logic is placed inside a Maven profile. **Listing 4** shows the relevant snippet.

ROLE OF DEVOPS
DevOps seeks to align goals, concepts, and tools for both development and operations.

Now let's trigger Flyway via Maven. We've introduced a dedicated profile, which we'll have to activate. In our simple example, the migration itself is started after a Maven installation. Keep in mind that Flyway uses classpath scanning: the migration scripts must be copied to the target (that is, Maven's target folder) to be applied successfully. The Maven call can look like this:

```
clean install flyway:migrate
-Pdb
```

To make the call even easier and to gain from other benefits, such as nice visualization, we will also use Hudson, the continuous integration engine, for database migrations. We can trigger the build manually, or let Hudson observe changes in version control and perform builds automatically, by calling the build scripts. The call results in some console output, as shown in **Listing 5**.

In our case, the database was empty and no migrations were run before. Thus, Flyway does some bootstrapping (creating metainformation, particularly the version number of the schema) and triggers our two migrations' scripts (because we've placed two SQL files into our directory).

Examining our database, we now have two new tables, both created by Flyway:

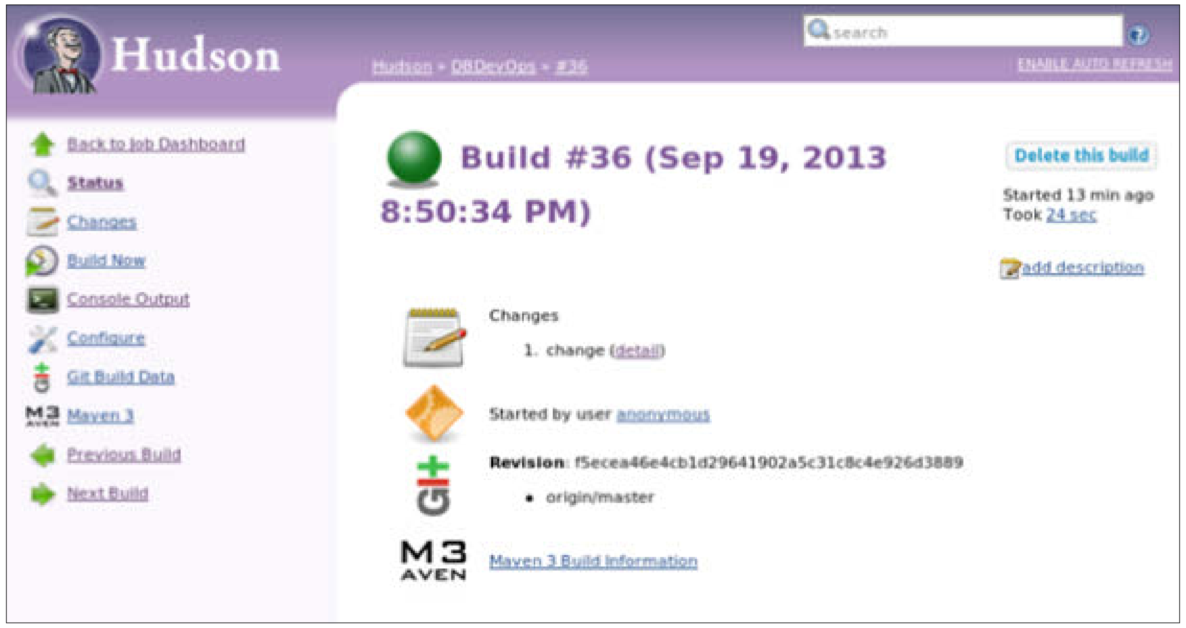
```
mysql> show tables;
+----------------+
|Tables_in_mydb|
+----------------+
|PERSON          |
|schema_version|
+----------------+
2 rows in set (0.01 sec)
```

One table is created by our migration scripts. The other table contains metainformation about the database, and migrations on it—including the information about which version the database is currently in—in order to derive which statements must be applied or not, in a specific environment, with a specific run.

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

```
michael@michael-VirtualBox:~/talk/project/devops/src/main/re-
sources/db/migration$
ls -la
total 16
drwxrwxr-x 2 michael michael 4096 Sep 22 11:16 .
drwxrwxr-x 3 michael michael 4096 Sep 22 11:16 ..
-rw-rw-r-- 1 michael michael  112
Sep 18 07:59 V1__Create_person_table.sql
-rw-rw-r-- 1 michael michael  149
Sep 18 07:28 V2__Insert_persons.sql
```

Download all listings in this issue as text



Build #36 (Sep 19, 2013 8:50:34 PM)

Started 13 min ago
Took 24 sec

Changes
1. change (detail)

Started by user anonymous

Revision: f5ecea46e4cb1d29641902a5c31c8c4e926d3889
- origin/master

Maven 3 Build Information

Figure 3

Now let's have a quick look into the table PERSON. The table has three rows (because the two migration scripts created the table PERSON and inserted three entries):

```
mysql> select * from PERSON;
+—+————————+
| ID | NAME        |
+—+————————+
| 1  | Peter Meyer |
| 2  | Peter Bonnd |
| 3  | Klara Korn  |
+—+————————+
3 rows in set (0.00 sec)
```

By executing Flyway's reporting features (on the command line or via Hudson), we can also get information about the current state of migration by triggering a flywayInfo command. Triggering Flyway with the build tool Gradle, using gradle flywayInfo, delivers the output shown in **Listing 6**.

How does Gradle work? **Listing 7** shows the Gradle build file that, from a functional perspective, pretty much is comparable to our Maven POM. With Gradle, project and build information are written with the programming language Groovy, which is a first-class citizen on the Java Virtual Machine. The file is also part of the source code tree in version control, as is the case with the Maven POM.

With Flyway, we can choose the build tool, either Maven or Gradle, that best fits our requirements.

Now let's add another migration to the process. Another file, our third migration script, which

LISTING 6 / LISTING 7 / LISTING 8 / LISTING 9

```
+———————+——————————————————+———————————————————+——————————+
| Version | Description       | Installed on        | State    |
+———————+——————————————————+———————————————————+——————————+
| 1       | Create person table | 2013-09-19 20:52:32 | Success |
| 2       | Insert persons      | 2013-09-19 20:52:32 | Success |
| 3       | InsertUpdate persons| 2013-09-19 20:55:52 | Success |
+———————+——————————————————+———————————————————+——————————+
```

Download all listings in this issue as text

is named V3__InsertUpdate_persons.sql, includes an update to an existing row and the creation of a stored procedure, ending with a call to that stored procedure to insert a new row into the existing table, as shown in **Listing 8**.

We can contribute the new artifact by committing and pushing to our version control system Git, as shown in **Listing 9**.

Hudson detects changes in Git, and rebuilds the project according to the build scripts, which are also part of the code tree in version control (see **Figure 3**).

The migration framework detects the current version of the

LISTING 10

```
[INFO]
[INFO] --- flyway-maven-plugin:2.1.1:migrate (default-cli) @
devops ---
[INFO] Current version of schema 'mydb': 2
[INFO] Migrating schema 'mydb' to version 3
[INFO] Successfully applied 1 migration to schema 'mydb'
(execution time
00:00.069s).
```

Download all listings in this issue as text

database, that is (2), and derives the information to apply the newly available migration number (3), resulting in the output shown in **Listing 10**.

We now have a fourth row in our PERSON table and an updated row, which obviously is a fix of a previously introduced typo. The new row was inserted by calling the freshly created stored procedure.

```
mysql> select * from PERSON;
+—+————————+
| ID | NAME        |
+—+————————+
| 1 | Peter Meyer  |
| 2 | Peter Bond   |
| 3 | Klara Korn   |
| 4 | Donald Luck  |
+—+————————+
4 rows in set (0.00 sec)
```

Crisp, isn't it?

Conclusion

This article introduced DevOps, a modern way of fostering collaboration between development and operations. Through shared goals, shared processes, and shared tools, barriers between different organizational departments are minimized.

The article also provided a round-trip through a concrete example that showed processes and tools—such as Hudson, Gradle, Maven, and Git—that might shape the daily collaboration between the different stakeholders and help to deliver software faster and with better quality. `</article>`

LEARN MORE

- *Agile ALM: Lightweight Tools and Agile Strategies* (Manning, 2011)
- *DevOps for Developers* (Apress, 2012)

Adding a Member to Your Development Team

The best and worst ways to manage the addition of a new team member

BEST DEVELOPMENT PRACTICES

T. LAMINE BA

BIO

I stare out the window. The sun is setting above the hills. Yet the team of a dozen Java developers and I, as the project technical lead, know that it is going to be another long day at work. We are behind schedule, and the customer will not settle for less than expected. So I try my best to keep a calm and unemotional demeanor in order to lead my staff with confidence, while my gut twists itself just enough to remind me that I should be worried. Excessive overtime has already exhausted my developers and, according to my calculations, productivity is starting to fall as a result. Furthermore, cost variance analysis of my schedule reveals that the best option at this time is to add more resources to the project.

Adding a new member to a team is a seemingly trivial step to take when your software development project is stalling. Yet managers commonly find themselves in a more difficult position upon the arrival of the new member. The reason? The integration of a newcomer into your team is a sensitive and risky event and should be handled by all leads with great attention and care.

Definition of a Team

Michael Levin, a Java Champion and a founder of multiple Java user groups around the world, suggests the following about a team: "Human resources come in various stages of sharpness. They all will work, potentially. The trick is to test their cutting ability before you start." This implies that finding the right resource for the right job is not a trivial task. So before solving the equation at hand, we should first understand what constitutes a team.

By definition, and according to the Project Management Institute, a team includes all stakeholders of a project. In this context, however, we are only concerned with the development team. That said, as social animals, teaming is a vital and inherent part of our existence and survival. To better understand its characteristics, we have to refer to Bruce Tuckman, who proposed the "Forming-Storming-Norming-Performing" model of group development in 1965. He argues that team formation usually follows four predictable and consecutive stages. During the *forming* stage, members are not clear about the objectives of the group. It is up to the group leader to provide direction. During the *storming* stage, multiple ideas compete with one another. Next, during the *norming* stage, the goal has been acknowledged by all and a common plan is in place. Finally, during the *performing* stage, the team is able to reorganize itself and adapt rapidly to changes caused by environmental factors in order to reach high

NOT SO TRIVIAL

Adding a new member to a team is a **seemingly trivial step** to take when your software development project is stalling.

performance levels and achieve specific objectives.

Impact of a New Member

As discussed earlier, a new team member can potentially disturb a whole organization. In order to prevent such a predicament, the technical lead has to be aware at all times of the stage of the team. The leader must assume that when there is change, teams tend to revert back to the storming stage in order to adapt. Hence, it is important to note that some stages are more desirable than others when introducing a new member into the team.

Here are a few suggestions on how to deal with the situation during each stage.

The team is in forming stage. In most cases, this is the best time to add a new member, considering that the goal of the project is not clear to most stakeholders. The new member can easily fit into the group and rapidly contribute.

The team is in storming stage. At this time, the different team players are discussing and proposing many ideas. Subgroups are created according to experiences, affinities, or other factors. Through self-organization, each subgroup elects its leaders, who will be responsible for distributing the right information across given channels. Tensions and adversity should be expected, and soft skills, such as good negotiation abilities, are required from the lead. During this phase, a new member might be easily welcomed into the team depending, for the most part, on the manager's competency. Hence, the manager ought to be sensitive about the new, informally created hierarchy within the team, which matters a lot to the members. Furthermore, it is advisable for the manager to fully disclose the role of the new member to the team and allow the team to actively participate in the reorganization that follows.

The team is in norming stage. The team has a clear plan. All members understand it and have agreed to it. Now, the goal of the team supersedes the goal of each member and progress has become the dashboard of all stakeholders. At this time, it is very risky to add a new member; the stakeholders might be resistant to any changes because they have already worked hard to reach a consensus. As a result, it is necessary for the technical lead to guide and monitor closely the new member through the team integration process. It is an ideal time to present performance charts, what-if scenarios, and prediction models to all stakeholders so the team comprehends and appreciates the need for more resources. The lead should also prepare for the natural tendency of the team to revert back to the storming phase. Then, great negotiation skills may move the team swiftly back to norming mode.

The team is in performing stage. During the performing stage, the team's work is very efficient. The group is able to satisfy all deliverables under budget and within schedule. There usually is a high level of satisfaction with the work conditions among most members. This level of productivity is not easily nor frequently reached. It is also usually short-lived, as the team may eventually revert back to storming. This is why it is the worst time to add a new member. It is preferable to "ride the wave" as long as possible before initiating any reorganization. However, if the project absolutely requires a new resource immediately, then proceeding with great caution and in concert with the team is advised. In fact, under such circumstances, many important decisions may be anticipated and proposed by the developers themselves. In other words, there may already be a strong advocacy by the team to bring in a new resource when required to help meet the goals of the group. It is, then, up to the manager to monitor closely all indicators suggesting that the new member is not "a good fit" and to take appropriate measures accordingly as early as possible.

Politics and Culture

Politics and culture are also important facets to consider when adding a new member. As a technical lead, it is essential to monitor and understand the way the team interacts and to share this information with the new member in order to facilitate his or her integration into the group. For example, the team may favor a very quiet working environment, or prefer to meet at a specific time of the day (such as mornings), or choose visuals rather than text in their presentations. Ultimately, most trends (from a corporate culture standpoint) should be controlled and to some extent driven by the manager. Also, the manager should not expect new members to learn all the rituals of the team on their own.

TEAM CULTURE

It is essential that the team lead monitor and understand **the way the team interacts** and share this information with the new member.

Taking the time to inform and prepare new members will reduce the learning curve.

Benefits of Good Object-Oriented Design Practices

According to Murphy's Law, "Anything that can go wrong, will go wrong." Following good object-oriented design practices becomes helpful when making changes to your organization. For instance, inheritance in Java provides the ability for a class to extend or override another class. Taking advantage of inheritance may facilitate the integration of a new programmer by allowing him or her to modify only a portion of the codebase. The coder can also reuse other tested and approved modules to experiment with his or her application. Furthermore, there are many benefits in using the method known as *polymorphism*. Polymorphism creates the ability for an object to contain other objects, including itself. This approach allows a programmer to implement various behaviors of the same object. As an example, the team could create an object animal that may have the ability to "quack" if it is a duck or "bark" if it is a dog. Such organization of the code makes it easier for new members to be functional, contribute, and fulfill their role while altering and learning about only a small part of the system. In the earlier example, assuming that the new member is responsible for the "quacking" behavior of the class animal, he or she may be assigned only the associated set of classes corresponding to the implementation of this behavior.

Conclusion

It is early morning, and I am the first person in the office. I walk up to the window. I see the most gorgeous sky slowly fading from a dark orange to a bright yellow. I stare and enjoy the beauty of Mother Nature, while reviewing in my head my plan of attack for the day: I will greet this new recruit, introduce him to the team, and prepare for the worst while expecting the best. Full of confidence, I exhale. Then, I feel a timid touch on my right shoulder, followed by a soft "Good morning." I turn around to find a young man full of energy, yet slightly disoriented as it is obviously his first day. He has his hand out, expecting a shake. I shake his hand, and with a greeting smile, I tell him, "It is a good morning, indeed. Please, follow me." </article>

LEARN MORE

- Project Management Institute

3 Billion Devices Run Java

Computers, Printers, Routers, BlackBerry Smartphones, Cell Phones, Kindle E-Readers, Parking Meters, Vehicle Diagnostic Systems, On-Board Computer Systems, Smart Grid Meters, Lottery Systems, Airplane Systems, ATMs, Government IDs, Public Transportation Passes, Credit Cards, VoIP Phones, Livescribe Smartpens, MRIs, CT Scanners, Robots, Home Security Systems, TVs, Cable Boxes, PlayStation Consoles, Blu-ray Disc Players...

Java™ | #1 Development Platform

ORACLE®

oracle.com/goto/java
or call 1.800.ORACLE.1

Concurrency Utilities for Java EE

Learn how to best execute tasks using application server concurrency services.

JOSH JUNEAU

BIO

*Listen to **Josh Juneau** introduce the topic of concurrency utilities in Java EE 7.*

Poor performance can be detrimental to an application's success. Performance can be measured in many ways. Whether application users can initiate tasks and then move onto others without waiting, or whether a database needs to be queried and return results without blocking a user interface, the bottom line is that waiting for tasks to be completed can become a user nightmare, making a poorly performing application difficult to use.

In Java SE, the idea of *threading* is a standard technique that is used to help prevent bad user experiences, because it allows for long-running tasks to be spawned into a separate thread and executed in the background without waits occurring.

Java EE 7 includes the concurrency utilities for Java EE, which standardize a solution for using application components and Java EE services in an asynchronous manner. The API provides an easy path for those familiar with Java SE concurrency to use it for enterprise applications.

Before Java EE 6, it was more difficult to perform concurrent tasks with server-side applications, because application server containers typically ran application component code on a thread that was managed by the container, and container-supplied object access occurred within the same thread. It is unwise to spawn new threads in an application server environment using java.lang.Thread or java.util.Timer, because that can lead to unreliable results.

TASK MASTER

Concurrency utilities in Java EE 7 **provide a solution** for using application components and Java EE services in an asynchronous manner.

The situation improved when asynchronous Enterprise JavaBeans (EJB) was introduced with Java EE 6, which allowed processing to occur asynchronously on a thread other than the request handling thread. The concurrency utilities for Java EE build upon asynchronous EJB, and introduce a set of application server concurrency services that assist in the asynchronous execution of tasks in an effort to provide a more complete concurrency solution for Java enterprise applications. In this article, we will cover each of these services and provide usage scenarios.

Server Resources

The following application server container resources are used to facilitate the use of concurrency utilities for Java EE: ManagedExecutorService, ManagedScheduledExecutorService, ContextService, and ManagedThreadFactory. These four managed service resources reside within the application server for asynchronous execution of tasks, obtaining container context, and the creation of managed threads.

The javax.enterprise.concurrent.ManagedExecutorService is used to execute submitted tasks in an asynchronous manner. It is an interface that extends its Java SE counterpart, the java.util.ExecutorService interface. The tasks submitted to a ManagedExecutorService are executed by threads that are controlled by the application server.

GlassFish 4.x contains a default ManagedExecutorService instance that can be accessed using the JNDI name of concurrent/_defaultManagedExecutorService or the standard JNDI name of java:comp/DefaultManagedExecutorService, as defined in the Java EE 7 Platform specification. Typically, tasks that make use of the ManagedExecutorService incorporate long-running processes that need to be executed in an asynchronous manner so that they do not cause other processes to become locked during execution.

The javax.enterprise.concurrent.ManagedScheduledExecutorService interface is used to execute tasks that are submitted on a scheduled basis in an asynchronous manner, very much like the ManagedExecutorService. This interface extends the java.util.concurrent.ScheduledExecutorService and javax.enterprise.concurrent.ManagedExecutorService interfaces. Again, tasks managed by this service are executed by threads that are controlled by the application server container. The difference between the two resources is that this one has the ability to assume scheduled execution using delay and periodic task execution capabilities that are made available by the ScheduledExecutorService interface.

GlassFish 4.x contains a default ManagedScheduledExecutorService with a JNDI name of concurrent/_defaultManagedScheduledExecutorService or the standard JNDI name of java:comp/DefaultManagedScheduledExecutorService.

The javax.enterprise.concurrent.ContextService is used to create contextual objects without using a managed executor, by utilizing the java.lang.reflect proxy capabilities to associate component container context with object instances. This allows objects to become contextual, allowing them to execute with the thread context of the associated component instance. GlassFish 4.x contains a default ContextService with a JNDI name of concurrent/_defaultContextService or the standard JNDI name of java:comp/DefaultContextService.

Lastly, a ManagedThreadFactory resource can be for creating managed container threads that can be utilized to perform tasks in the background while other processes are executing. GlassFish 4.x contains a default ManagedThreadFactory with a JNDI name of concurrent/_defaultManagedThreadFactory or the standard JNDI name of java:comp/DefaultManagedThreadFactory.

Note: While an installation of GlassFish 4.x contains a default server resource for each of the different concurrency utilities, it is possible to generate your own, if desired. To do so, you can use either the GlassFish administration console or the asadmin command-line utility. There are several attributes you can configure to customize the concurrency resources, including Deployment Order, Thread Priority, and more. For more information on creating such resources, refer to the Java EE 7 Tutorial or the Concurrency Utilities 1.0 specification.

Accessing Application Resources

In order to submit tasks to the server-side concurrency resources for processing, an application must include a reference for the server-side resources, either defined within an application's deployment descriptors or via the @Resource annotation. For instance, within a web application, the resources can be defined within the web.xml file.

To register references to the resources within web.xml, entries for any or each of the four interfaces can be made inside of a <resource-env-ref> element by assigning a <resource-env-ref-name> and a <resource-env-ref-type> for each entry. **Listing 1** demonstrates how to register a ManagedExecutorService within the web.xml file.

The @Resource annotation can also be used to inject the resource into the code, as needed, by passing the name of the server-side resource instance to the annotation. For instance, to obtain an instance of the ManagedExecutorService by injection, you can use the following lines of code:

```
@Resource(name=
  "concurrent/myMES")
ManagedExecutorService mes;
```

The examples demonstrated in the rest of this article will focus on using the @Resource annotation.

Example Scenario

We will demonstrate the use of each different concurrency utility using a real-life example application, a basic vacation planner for the Acme World theme park. It allows guests to book reservations for park tickets and restaurant meals. In the examples, the application will be used under different scenarios for each concurrent task demonstration.

Managed Tasks

The javax.enterprise.concurrent.ManagedExecutorService interface can be used to create server-side resources that have the ability to execute tasks in an asynchronous manner. Classes that contain task implementations to be run asynchronously can be submitted to a ManagedExecutorService for processing. Any class that is to be submitted to the ManagedExecutorService must implement either the java.lang.Runnable or the java.util.concurrent.Callable interface.

Task classes that implement java.lang.Runnable must include a run() method, which is invoked by the ManagedExecutorService. Such tasks are useful for running reports or long-running queries that are executed in the background. Classes that implement java.util.concurrent.Callable must also implement the ManagedTask interface, allowing the class to supply an identifiable name via the IDENTITY_NAME property. These classes also must implement the call() method, which performs the work for the task at hand. Such task classes are used for issuing multiple tasks to be processed asynchronously.

To demonstrate the submission of a long-running task, let's suppose that the directors at Acme World wish to run a report that will provide them with a summary of each park reservation. To initiate the report, they click a button that will send them the report via e-mail. This button is bound to a managed bean action, which will invoke a potentially long-running task to query the database and retrieve reservation information, including information regarding the cost of each reservation. Because this is a long-running task, we'd like the user to receive an immediate response from the server, indicating that the information will be e-mailed shortly. Therefore, the task is handed off to a ManagedExecutorService for processing, and then control is returned to the user. The class that contains the logic for the task is named AcmeReservationReport, and the code is shown in **Listings 2a** and **2b**.

To make use of the AcmeReservationReport, it must be submitted to a ManagedExecutorService resource on the server. This is typically done by creating an instance of the task class and submitting it to the ManagedExecutorService using its submit() method. A Future object will be returned, which can be used to determine the status of the long-running task. **Listing 3** demonstrates how to submit the task class for processing.

In some cases, an application requires processing of more than one long-running task asynchronously; then the results of each can be processed once they

LISTING 1 / LISTING 2a / LISTING 2b / LISTING 3

```
<resource-env-ref>
    <description>
     Reference to ManagedExecutorService
    </description>
    <resource-env-ref-name>
      concurrent/__defaultManagedExecutorSerice
    </resource-env-ref-name>
    <resource-env-ref-type>
javax.enterprise.concurrent.ManagedExecutorService
    </resource-env-ref-type>
  </resource-env-ref>
```

Download all listings in this issue as text

become available. Tasks that implement java.lang.Callable are appropriate for such cases.

To demonstrate such a task, let's suppose that a weekly e-mail for guests that have a reservation to Acme World is constructed by combining the results of queries made to both the park database tables and the restaurant reservations tables. To do so, two tasks are submitted, one to query the park reservation and the other to query the restaurant reservations. Once the tasks have completed, they are processed together to formulate the content for the e-mail report.

Listing 4 contains the source code for the AcmeParkReservation task; **Listing 5** contains the source code for the AcmeRestaurant Reservation task. Each of these tasks implements Callable and the ManagedTask interfaces. The constructors accept the arguments that will be needed to execute the task, and the call() method performs the long-running implementation. In this case, the database is queried within each of the tasks, and information is then returned for further processing.

The AcmeSummaryEmail Controller JavaServer Faces (JSF) managed bean controller is used to process each of the two tasks and then combine the results to formulate the e-mail report. To build the report, each of the tasks is sent to the ManagedExecutorService separately for processing, and in turn, a Future<Object> is returned for each. This Future object can be used to determine whether the task has been completed by invoking the object's isDone() method, and once the task has finished, it can be used to obtain information from the object that is returned by invoking the object's get() method. **Listings 6a** and **6b** contain the sources for the AcmeSummaryEmailController JSF controller.

A ManagedExecutorService instance can be used by multiple application components at a time, and each task becomes contextual, meaning that it will retain the context of the submitting component. The task is then executed within the context of the submitting component. A ManagedExecutorService instance can be terminated or suspended when the server itself is shut down, or when applications or components are disabled or removed.

Scheduled Tasks

The javax.enterprise.concurrent .ManagedScheduledExecutorService can be used to execute tasks at specified times. Interfacing with the ManagedScheduledExecutor

LISTING 4 / LISTING 5 / LISTING 6a / LISTING 6b

Note: The following listing has been excerpted, as noted by the . . . symbol. The full code listing is available by downloading the code listings for this issue.

```
public class AcmeParkReservation implements Callable<
ParkReservation>, ManagedTask {

  String reportId;
  BigDecimal parkReservationId;
  Map<String, String> executionProperties;
  private ParkReservationFacade parkReservationFacade;

  public AcmeParkReservation(String reportId,
BigDecimal parkReservationId,
ParkReservationFacade parkReservationFacade){
      this.reportId = reportId;
      this.parkReservationId = parkReservationId;
      this.parkReservationFacade = parkReservationFacade;

      executionProperties = new HashMap<>();
      executionProperties.put(
ManagedTask.IDENTITY_NAME, getIdentityName());
  }

  public String getIdentityName() {
      return "AcmeParkReservation:  reportId=" +
reportId + ", parkReservationId=" + parkReservationId;
  }

  @Override
  public ParkReservation call() throws Exception {
     // Perform long-running task
     return parkReservationFacade.findById(parkReservationId);
  }
. . .
}
```

Download all listings in this issue as text

Service is very similar to working with the ManagedExecutorService in that any task classes must implement either the java.lang.Runnable or the java.util.concurrent.Callable interface. The only significant difference between these two types of concurrency resources is the scheduling ability that is specified when submitting a scheduled task.

Listing 7 shows a task class, entitled AcmeReservationCount, which is used to periodically poll the Acme World reservation database to see how many new reservations have been made. Note that the implementation does not contain any scheduling information, but only provides a task implementation.

As mentioned previously, scheduling occurs at task submission time, and **Listing 8** demonstrates how to submit the AcmeReservationCount class to the ManagedScheduledExecutorService for processing. In this case, the task will be executed every 60 minutes, and it will log a count of new reservations that have been made within the past 60 minutes. This scheduling is done by calling one of the ManagedScheduledExecutorService methods, which are inherited from the java.util.concurrency interfaces.

In this example, the schedule AtFixedRate method is invoked, which accepts the Runnable, the initialDelay, the period of time, and the time unit. In the example, the time unit is TimeUnit.MINUTES. The scheduled managed task is implemented within a servlet context listener named AcmeWorldServletContextListener. At application startup, the context is initialized and the task is scheduled to be executed once an hour.

A ManagedScheduledExecutorService instance is capable of being utilized by multiple applications and components at a time. Each task that is submitted to the service is contextual, and the context of the thread running the task is changed to match that of the task instance while it is being executed. The context is restored once the task is completed.

Creating Contextual Objects

The javax.enterprise.concurrent.ContextService interface can be used for creating contextual objects without the use of a managed executor. This is achieved by using the dynamic proxy capabilities of the java.lang.reflect package, associating the application component context with the object instances that are created. This service can be used to develop custom concurrent applications and services, allowing for the

LISTING 7 LISTING 8

Note: The following listing has been excerpted, as noted by the . . . symbol. The full code listing is available by downloading the code listings for this issue.

```
public class AcmeReservationCount implements Runnable,
Serializable {
   ParkReservationFacade parkReservationFacade =
lookupParkReservationFacadeBean();
   String reportName;
. . .

   public void run() {
         runCountReport();
   }

   /**
    * Prints a count of reservations.
    */
   public void runCountReport() {
      System.out.println("Park Reservation Count for Today");
      System.out.println("==============================");
      Long reservationCount =
parkReservationFacade.findCount();
      System.out.println(reservationCount);
      // Email in production application
   }

   private ParkReservationFacade
lookupParkReservationFacadeBean() {
      try {
         Context c = new InitialContext();
         return (ParkReservationFacade) c.lookup(
"java:global/JavaMagazineEE7/ParkReservationFacade");
      } catch (NamingException ne) {
         . . .
      }
   }
}
```

Download all listings in this issue as text

creation of custom Java EE components, such as ExecutorService implementations.

Contextual object proxy instances are created by calling upon a ContextService instance createContextualProxy() method. Task instances must implement the Callable interface, and the call() method contains the task implementation. The example in this article demonstrates how to generate a custom Singleton EJB that can be used for generating a managed thread pool executor. **Listing 9** contains the code for ExecutorAccessor, the Singleton EJB that is used for creating a managed thread pool executor, which will be used to submit the contextual task instance.

The task in this example is implemented in a class named AcmeSingleReservation (**Listing 10**), which accepts a reservation ID number and queries the database to return the matching reservation record object. The call() method in the class creates a database connection, queries the specified reservation, and returns the object.

The task is invoked via a stateless session EJB that passes a given reservation ID number to the AcmeSingleReservation task class, returning the reservation object. It does so by using different threads to retrieve each of the reservations. The ContextService is used to create a contextual proxy for each AcmeSingleReservation call, and that proxy is then submitted to the thread pool that is retrieved from the ExecutorAccessor Singleton EJB. The reservation objects are then returned by calling the thread pool take().get() method chain. **Listing 11** contains the complete implementation for ReservationChecker, the stateless session bean that contains this implementation.

Server Threads

Server-side thread instances can be created using the javax.enterprise.concurrent.ManagedThreadFactory interface. Spawning server-side threads can be beneficial when there is a requirement to execute a long-running task without producing a bottleneck within an application. New threads can be created by calling the java.util.concurrent.ThreadFactory interface's newThread(Runnable r) method, passing the Runnable class to be executed. Once created, the newly created thread's start() method can be invoked to start the thread within the context of the application component instance.

For example, let's assume that the Acme World application is going to poll the

LISTING 9 / LISTING 10 / LISTING 11

```
@Singleton
public class ExecutorAccessor {

  private ExecutorAccessor executorAccessor;
  private ThreadPoolExecutor tpe;
  @Resource(name = "concurrent/__defaultManagedThreadFactory")
  ManagedThreadFactory threadFactory;

  @PostConstruct
  public void postConstruct() {
    tpe = new ThreadPoolExecutor(
        5, 10, 5, TimeUnit.SECONDS,
        new ArrayBlockingQueue<Runnable>(10),
threadFactory);
  }

  public ExecutorService getThreadPool() {
    return tpe;
  }
}
```

Download all listings in this issue as text

RestaurantReservation entity periodically to see if a new reservation has been made, and then send alerts accordingly. A Runnable class can be created to house the procedure for polling the database and creating the alerts. **Listing 12** shows the code for the Runnable that will be used to create the managed thread in this example, a class named ReservationAlerter. That class can then be made into a thread by calling a ManagedThreadFactory's newThread() method, as shown in **Listing 13**.

Listing 14 shows the complete code for obtaining a reference to the ManagedThreadFactory via @Resource injection, creating the new Thread instance, and then starting it.

Utilizing Transactions

The concurrency resources support user-managed global transaction demarcation via the javax.transaction.UserTransaction interface. This allows transactions to begin via the begin() method, commit via the commit() method, and roll back via the rollback() method, if needed. Any managed task instances are run outside of the submitted thread's transaction scope. This means that the executor is responsible for coordinating transactional actions, and any transaction that is active in an executing thread will be suspended. **Listing 15** illustrates how a task can interact with transaction-capable resources in a single transaction.

Conclusion

The days of developing custom solutions to deal with multithreaded enterprise applications are over. The concurrency utilities for Java EE provide a standard solution for concurrency in enterprise applications, which supports simple and advanced concurrency patterns. Those who are familiar with using concurrency APIs under Java SE will have no problem learning how to utilize the concurrency utilities for Java EE, because both APIs follow similar guidelines. For beginners, the API is straightforward and intuitive.

To see the examples used in this article in action, download and create a NetBeans project using the article sources. Then create the database using Apache Derby, which comes packaged with GlassFish. Finally, deploy the application project to your GlassFish server. </article>

LEARN MORE

- The Java EE 7 Tutorial
- JSR 236: Concurrency Utilities for Java EE

LISTING 12 / LISTING 13 / LISTING 14 / LISTING 15

Note: The following listing has been excerpted, as noted by the . . . symbol. The full code listing is available by downloading the code listings for this issue.

```
public class ReservationAlerter implements Runnable {

  @Override
  public void run() {
    while (!Thread.interrupted()) {
      reviewReservations();
      try {
        Thread.sleep(100000);
      } catch (InterruptedException ex) {
        // Log error
      }
    }
  }

  public Collection reviewReservations() {
    Connection conn = null;
    Properties connectionProps = new Properties();
    connectionProps.put("user", "user");
    connectionProps.put("password", "password");
    Collection reservations = null;
    try {
      // Obtain connection and retrieve reservations
      conn = DriverManager.getConnection(
        "jdbc:derby:acme;create=false",
        connectionProps);
   // Use the connection to query the database for reservations
    } catch (SQLException ex){
      System.out.println("Exception: " + ex);
    } finally {
     . . .}
    return reservations;
  }
}
```

Download all listings in this issue as text

ERIC J. BRUNO
BIO

Internet of Things 101

Explore concepts and issues relevant to the Internet of Things by creating an embedded application.

Many companies are promising a future where all the smart computing devices around us will communicate with one another—and with the enterprise applications we use most—in what some call a machine-to-machine (M2M) architecture. Oracle has a plan here as well, focused on the Internet of Things (IoT). Although achieving this promise might sound as simple as enabling Bluetooth or Wi-Fi on a mobile device, making the most of the connected future requires much deeper integration.

What Is a Device?

The first step when planning an IoT architecture and strategy is to define what a device is. Most people initially define a device as a smartphone or a tablet, but smaller, less-powerful phones also qualify. Sensors, electrical controllers, RFID readers, and modern home appliances play a role in the world of M2M and IoT as well. Basically any electronic endpoint capable of being uniquely identified, with pertinent resource or location data, qualifies as a usable device in an IoT architecture.

The amount of data and the utility each type of device provides might vary, but when combined in large quantities, tremendous value can come from even the simplest data. The key is identifying and connecting all the devices together so that they can communicate and even collaborate; this is where the internet plays a role.

The Internet of Things

In 1999, Kevin Ashton—the cofounder of the Auto-ID Center at MIT—coined the term *Internet of Things* to define a system in which the internet is connected to the real world via ubiquitous sensors. Ashton's vision was to create a global system of computers, devices, and sensors that feed on their own data, instead of being dependent solely on humans for data input. As a result, the system can have an understanding of where everything is and what's taking place at any location and point in time. This would lead to a world of connected systems that could greatly reduce waste, lower costs, and eliminate loss (however you define it) for just about any human-machine activity.

As researchers are scrambling to resolve issues—such as addressability, communication standards, battery power, and data collection—enterprises are rushing in to fill the gaps, define standard platforms, and create an ecosystem in which all parties stand to gain. The IoT is in its very early stages, and it's an exciting time to be part of its formation.

THE RIGHT STUFF
Perhaps **no other technology today** is better positioned to power an Internet of Things strategy than Java.

Java and IoT: A Platform for Hyperconnectivity

Java was born on an embedded device. Initially part of the Green Project to power the Star7 handheld device, Java went on to see enormous success in both the web and embedded domains. Perhaps no other technology today is better positioned to power an IoT strategy than Java.

Given its ability to run on a wide range of devices—from feature phones with limited CPU and memory to rackmounted servers with immense power and capacity—Java is *meant* to power a world of compute resources with ubiquitous connectivity, otherwise known as *hyperconnectivity*. Java-powered devices in this ecosystem will communicate with each other, with users, and with the real world via sensors and relays to gather knowledge—and, ultimately, understanding—about the world around us in real time.

Imagine the rich information that can be discovered and reported in the areas of healthcare, automotive (aka telematics), building and industrial automation, smart metering, transportation, logistics, and so on. Here are just a few examples:

- Potentially fatal mistakes can be prevented when automated devices crosscheck a patient's allergies against medication in bottles that identify their contents.
- Traffic conditions can be improved when telematics systems inside vehicles automatically alert and reroute drivers before congestion occurs.
- Electricity generation can be made more efficient if gateway devices within homes and office buildings help to more accurately report and predict peaks and valleys in demand over time.

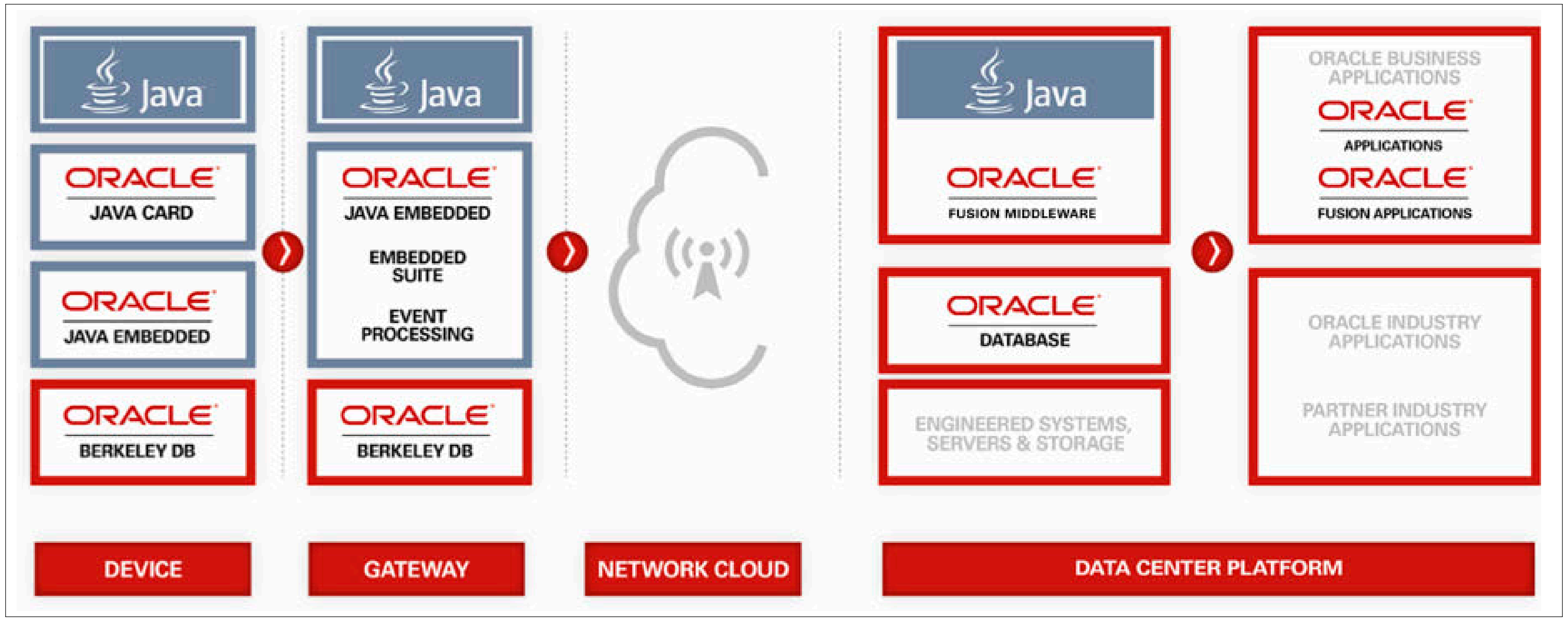


Figure 1

Oracle is helping to define a standard IoT platform based on Java, with Java-based middleware, data storage, and analytics—all with the appropriate hardware—so that developers can focus on innovation and solutions (see **Figure 1**). Using Oracle Java ME Embedded in sensors helps ensure systems can be built efficiently, using the same language throughout the end-to-end solution with device portability and ongoing support. Oracle Java SE Embedded and Oracle Java Embedded Suite help to further provide the capabilities needed to deploy fast, low-cost M2M gateways with data capture, event processing, and analytics performed end to end.

However, a complete IoT architecture goes beyond Java and simply identifying a software suite. It requires a managed solution in which the key benefits are end-to-end security and the ability to expand and scale to future needs, meet industry requirements for standards and security, reduce costs in terms of IT infrastructure, integrate with existing systems, and perform well enough at every layer. Let's look at Oracle's IoT platform now in more detail.

Provisioning and management. Devices, data, and users need to be identified, authenticated, and authorized at each point of an IoT solution. For example, before making decisions, it's critical to know the reliability and true identity of sensors in your network as they report information. Additionally, due to government regulations and ethics in real-world scenarios, you need to ensure data not only stays out of the wrong hands, but that only those authorized to view the data are allowed to do so (for example, a doctor making a medical diagnosis for a specific patient).

Provisioning new devices, identifying where data originates in your network, and giving permission to the right users at the right time are all crucial functions required in an IoT platform. Having the ability to efficiently (and centrally) manage expanding networks of small

IoT at JavaOne 2013

Throughout JavaOne and Oracle OpenWorld, the movements of attendees were tracked by IoT in Motion, an Internet of Things collaboration between Oracle, Eurotech, Hitachi Communication Technologies America (Hitachi CTA), and Hitachi Consulting, that counted and tracked conference attendees in various locations.

Oracle's Jennifer Douglas provided a technology overview: "Hitachi Consulting helped build the actual application that is running the data, using an Oracle Exalytics box and the Oracle Business Intelligence dashboard. Hitachi CTA has their SuperJ running in conjunction with Oracle Java SE Embedded through a gateway to the Eurotech Everyware Cloud, which collects the raw data. Then the Exalytics box compiles the data and converts it into something we can actually utilize." All of the technology ran on Java.

While IoT in Motion can distinguish a dog or a vehicle from a person, the stereoscopic camera merely registers and counts people going in and out of the spaces without monitoring any features of individual people.

—Janice J. Heiss

sensors and distributed compute nodes is also required.

Security. As mentioned above, user and device identity is part of application security. Oracle's IoT architecture defines entry points to customers' LDAP systems for user identity management, network equipment and industry-standard protocols for secure communication, and safe offsite data storage for scenarios in which this is a requirement. End-to-end security must be well defined and enforced, yet be flexible enough to allow system expansion and usefulness.

Connectivity. Device connectivity begins with basic protocols such as Ethernet, but standards of communication for sensors and sensitive data must also be supported. For instance, there are industry protocols such as DASH7 for wireless sensor networks, ZIGBEE for electrical controllers, 6LoWPAN for building automation and smart grid applications, COSEM for smart metering, and Bluetooth for a wide range of uses such as telematics systems.

Beyond protocols, an IoT platform needs to account for devices that might lose and gain connectivity repeatedly while being used, and ensure that data is stored and later forwarded as needed.

Event processing. As data is collected from the hundreds or thousands of endpoints in your network, you need to gain actionable insight into your data in real time. Complex event processing software built into your IoT platform provides the ability to filter or make decisions on data before sending all of it to your data center. Doing so helps reduce bandwidth and processing needs (and costs) in your back-end systems. The result is a more-responsive system, since automated decisions are made closer to their datasources and the systems that need to act on them, as opposed to making decisions centrally.

Storage. Data, especially from sensors in the real world, comes in many shapes and sizes. From structured relational data to unstructured, random data, all the information your IoT system captures must be collected, stored, retrieved, and analyzed to mine its full value. This data must be stored properly and securely at each layer, whether it's in memory within Oracle Berkeley DB on a local device or gateway, in a NoSQL database in a public cloud or the middle tier of your architecture, or in a relational database in your data center or private cloud.

An IoT platform helps store data in the right locations based on structure and retrieval needs. Balancing the need to quickly retrieve data without sacrificing security and data integrity is the job of the IoT platform, not an application concern.

GOING DEEP

Although achieving the promise of the "Internet of Things" might sound as simple as enabling Bluetooth or Wi-Fi on a mobile device, **making the most of the connected future requires much deeper integration.**

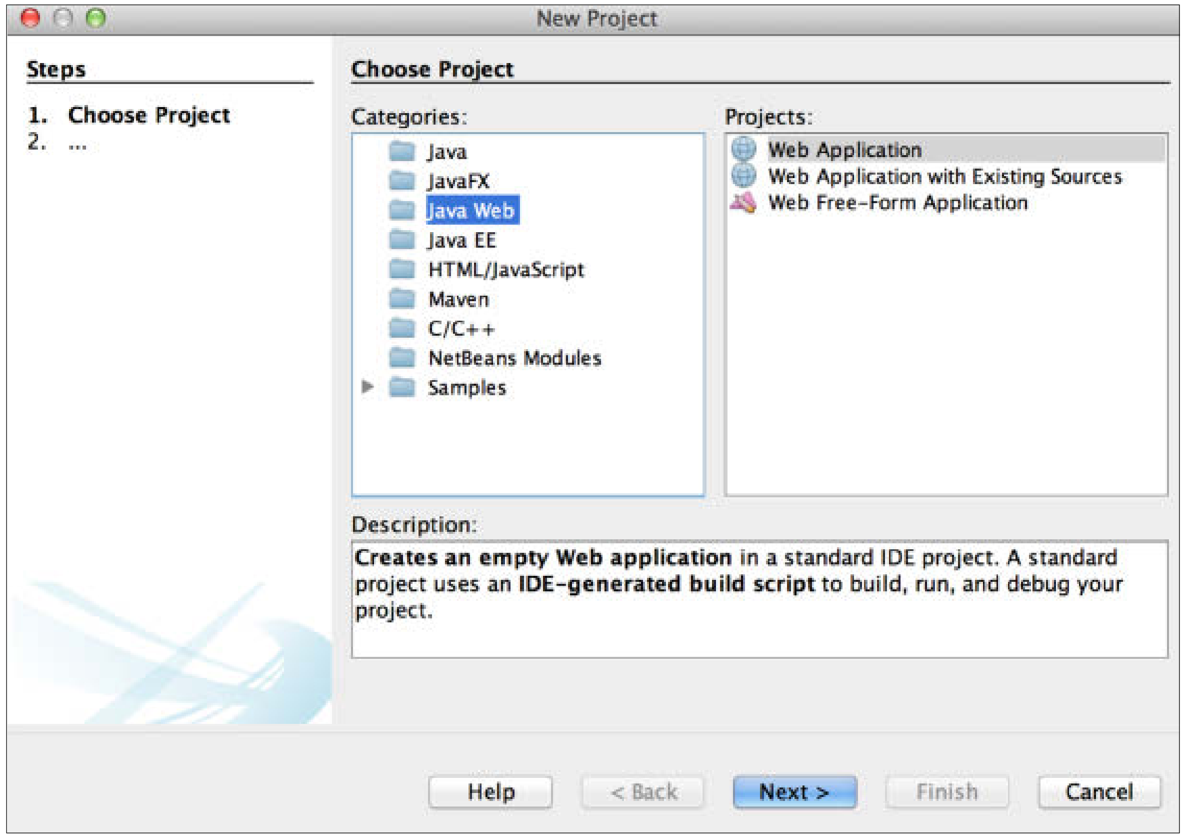
Big data and analytics. Analyzing the potentially huge amounts of data captured from your devices can reveal a plethora of hidden value. There are generally two types of analytics: real-time analytics and historical analytics.

With real-time analytics, data is analyzed as it's collected from devices and potentially combined with other data to make dynamic decisions. This tends to work best when it's performed as close to the devices as possible. Historical analytics involves data captured over time, perhaps even years, to look for correlations between applications and user-specific trends.

Historical data sets can grow to enormous sizes over time and, therefore, the analytics are usually run within the data center as close as possible to the data stores.

Big data is a term used to describe a collection of data sets so large and complex that it's difficult to process using traditional database management tools and algorithms. Analytics solutions meet the challenge of storing, indexing, searching, visualizing, and drawing inferences and conclusions from these large data sets. IoT platforms need to integrate structured and unstructured data, perform map-reduce operations on the data, and support various file systems such as the Hadoop Distributed File System (HDFS).

As mentioned above regarding event processing, performing edge analytics at locations physically close to data collection and curation might yield a more hierarchical system capable of reduced response times.



Creates an empty Web application in a standard IDE project. A standard project uses an **IDE-generated build script** to build, run, and debug your project.

Figure 2

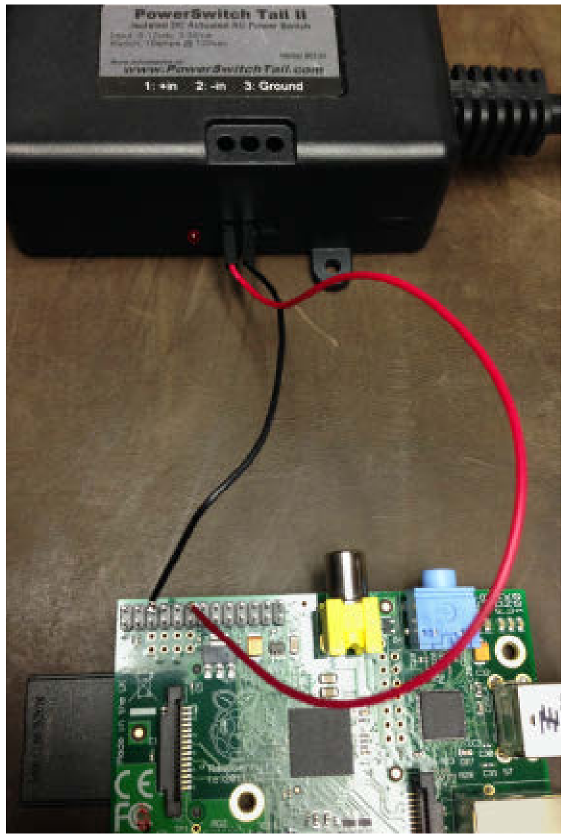


Figure 3

A Working IoT Example

As an example of Java in an embedded, IoT scenario, let's build the foundation of a home gateway server using a 512 MB Raspberry Pi. You can envision this gateway being used by a homeowner to remotely control an electrical device or to program appliances, such as a dishwasher, to turn on only when the price per watt-hour for electricity drops below a certain threshold. Integrating the gateway with a smart meter, which is itself a gateway for the power company, a true IoT solution can be built that offers benefits to everyone: the consumer enjoys lower electricity costs, and the power company manages demand more effectively.

To make the solution more realistic, we'll deploy the application as a Java servlet so that it can be accessed and controlled from a browser. This will allow the homeowner to access the home gateway from a desktop, laptop, or smart device no matter where the homeowner is—security concerns aside.

Note: The source code for the example described in this article can be downloaded here.

Coffee and dessert. In an embedded context, you have a choice between Java ME and Java SE. The target device capabilities, application, and other factors will affect your decision—for example, Java ME requires fewer resources, such as RAM—but for our example, we'll choose Oracle Java SE Embedded. Having the ability to run with as little as 32 MB of memory, Oracle Java SE Embedded strikes a balance between the requirements of standard Java and embedded development. Additionally, it fits our requirement to run an embedded web server to power our Java servlet.

To begin, download Oracle Java SE Embedded, and then install it by unpacking the .tar file in your home directory on your Raspberry Pi using the following command:

```
~ $ tar xvf <filename>.tar
```

You can test your installation by running ~/ejre1.7.0_40/bin/java -version, replacing the directory name with the name of the installation directory.

Next, to run our servlet application, you can use Apache Tomcat, IBM's WebSphere Liberty Profile, Oracle Java Embedded Suite, or any others designed to run in an embedded context. Being based on the standard Servlet specification, our servlet code should work equally well within each server, with only slight differences in how it gets deployed. For this article, we'll focus on the code itself.

We begin with NetBeans by creating a new "Java Web" type of application (as shown in **Figure 2**). In my case, I chose GlassFish as the target host, and I used Oracle Java Embedded Suite on my Raspberry Pi to host the servlet. This configuration allowed me to easily do local development and debugging with a similar deployment environment on my Raspberry Pi.

My servlet is named Gateway, and it uses the Pi4J library to enable easy access to the Raspberry Pi's general-purpose input/output (GPIO) pins from Java. The code shown in **Listing 1** defines a variable to represent Pin 1 on the Raspberry Pi.

Setting Pin 1 to "high" sends 3.3 volts of power to a connected device. In our case, this pin will connect to a PowerSwitch Tail device, which accepts a low-power connection to turn an internal AC relay switch on and off, thereby controlling any attached electrical components, such as a light (see **Figure 3**).

The PowerSwitch Tail plugs into AC power on one end and controls any AC component plugged into the other end. The remaining code in the servlet simply turns Pin 1—and any component connected—on or off via a URL (see **Listing 2**).

To test the servlet, I deployed it to the servlet engine running on my Raspberry Pi, and from a browser I entered the following URL to set Pin 1 to "high": http://<IP address of Pi>/SmartGrid/Gateway?power=on.

Doing this after properly hooking up the PowerSwitch Tail to the Raspberry Pi, plugging the PowerSwitch Tail into AC power, and plugging a component—such as a desk lamp—into the other end of the PowerSwitch Tail will remotely turn on the component. To turn it off, simply change the value of the power parameter to off: http://<IP address of Pi>/SmartGrid/Gateway?power=off.

Conclusion

Thanks to the power and versatility of Java, its ability to run on small embedded devices such as the Raspberry Pi, and the availability of a wide range of tools and libraries, you can enter the world of IoT without learning a brand-new skill set. Gone are the days of building your own tool chain with cross compilers and other specialized tools, requiring you to be proficient in multiple programming languages. Java is truly prepared to help you succeed in your new IoT projects now and in the future. One set of tools, one language, and a set of integrated and high-performance Java Virtual Machines (JVMs) are available to enable you today. </article>

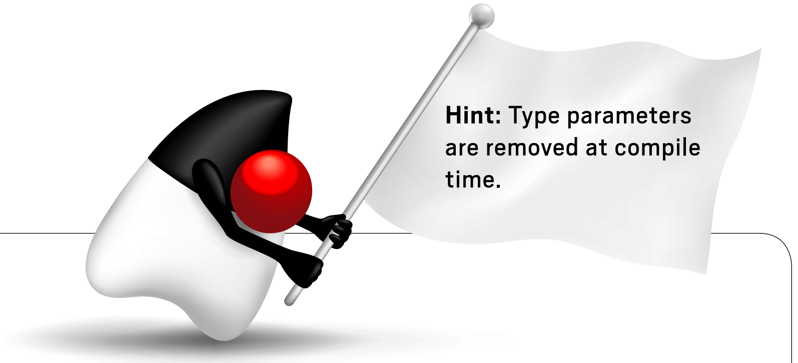
LISTING 1 LISTING 2

```
import com.pi4j.io.gpio.GpioController;
...

@WebServlet(urlPatterns = {"/Gateway"})
public class Gateway extends HttpServlet {
  private static final long serialVersionUID = 1L;

  final GpioController gpio =
      GpioFactory.getInstance();

  final GpioPinDigitalOutput pin =
      gpio.provisionDigitalOutputPin( RaspiPin.GPIO_01,
                              "SmartGrid",
                              PinState.LOW);
  ...
```

Download all listings in this issue as text

LEARN MORE

- Oracle Java Embedded



In the September/October 2013 issue, polyglot developer Attila Balazs gave us a class initialization code challenge. He presented us with code that deadlocks or throws a NullPointerException but never runs to the end correctly and asked us for a fix. The correct answer: both #1 and #3. The root cause of the problem is that we manage to call the classloader from two separate threads and deadlock it:

- From AppThread we reference App, which means that the class must be loaded and initialized. During the initialization we need to load (and initialize) Controller, however.
- From main we are already loading it, thus the main thread waits after AppThread to load/initialize the App class while AppThread waits after main to initialize the Controller class.

If we change Integer to int, the compiler would see RETRIES as a constant and inline it, eliminating the dependencies between the two classes. Also, if we move RETRIES into Controller (where it belongs, because it is referenced only from that class) the dependency would be broken again.

This issue's challenge comes from Oracle ACE and NuBean Consultant Deepak Vohra.

1 THE PROBLEM

Generics are a relatively new addition to Java (introduced in Java SE 5), and it is not always clear how they can be used.

2 THE CODE

What needs to be fixed in the class Sample?

```
import java.util.ArrayList;
import java.util.HashMap;

public class Sample{

    public <T> void setArrayList(ArrayList<T> arrayList, T t){
    }

    public <T> void setArrayList(ArrayList arrayList, Object obj){
    }

    public <T extends HashMap<K,V>> void setArrayList(ArrayList<T>
arrayList){
    }

}
```

3 WHAT'S THE FIX?

1) You can't have three methods named setArrayList. Remove the last method named setArrayList.
2) Type parameter T cannot extend HashMap<K,V>.
3) If some of the methods are generic, the class must be generic.
4) Remove one of the first two methods and add the type parameters K and V to the third method.

ART BY I-HUA CHEN

GOT THE ANSWER?
Look for the answer in the next issue. Or submit your own code challenge!