The Java™ platform is stable and its community vibrant, but that doesn't mean Java technology is done evolving. Oracle's acquisition of Sun Microsystems is only the latest in a series of fundamental changes that are redefining the Java industry. Rather than sit around and wonder, Java technology zone editor, Jenni Aloi, brought together a developerWorks roundtable — nine leading thinkers in the Java community — for an honest inquiry into the current and emerging state of Java technology, culture, and industry.

Since Oracle announced its intention last year to acquire Sun Microsystems, the Java developer community has been debating what the change in stewardship will mean in practical terms, and how it will affect the future of the Java platform.

In fact, the evolution of Java technology has been a question of concern and debate for many years now, with some being so bold as to declare the technology "dead" before its fifteenth birthday. Therefore, many have been surprised by the surging revival of interest in the Java platform and libraries, as enterprise developers began to address, in earnest, issues of reliability and scale.

With Oracle's acquisition finalized and Java 7 due for formal release later this year, the time was ripe to pull together a roundtable of Java community leaders — developers, architects, and analysts (corporate and independent, alike) — and get
their perspectives on the current and emerging state of Java technology, culture, and industry.

Responses varied from expected to surprising, and were reflective of the panelists’ expertise and extensive work with the Java platform. Sincere thanks to roundtable participants: Dan Allen, Neal Ford, Andrew Glover, Richard Hightower, Mik Kersten, Christopher J. Mackie, Alex Miller, Ted Neward, J. Steven Perry. Learn more about our contributors’ extensive backgrounds, including their published works on developerWorks and elsewhere.

Note: The views and comments of the panelists do not represent the views of IBM. For more information about IBM’s current activities around the Java platform, see Chris Bailey’s "Java technology, IBM style: A new era in Java technology," which introduces some of the major enhancements arriving through the Oracle Java 7 specification, as well as the additional value-add enhancements from IBM. Trent Gray-Donald, technical lead on the IBM SDK for Java 7, also speaks about the upcoming changes in a timely and informative podcast.

Java technology in 2010

To launch the discussion, we asked panelists what they saw as the most important challenge facing Java-based developers today, and its greatest opportunity. While responses varied, the most common theme was complexity: the complexity of the Java platform and associated Web application frameworks, as it bumps up against the inherent challenges of software development today.

"A typical Java Web app uses 15 or 20 MB of frameworks before you can even start writing code!" said Alex Miller. "Getting started with this mess of stuff is daunting to me, and I have a decade of Java development experience."

Mik Kersten agreed: "The size of the systems we work on continues to grow, while the capacity of our brains does not."

Meanwhile, according to Dan Allen, the pressure to deliver is intense. "Developers simply cannot afford to spend time developing lines of code that do not contribute to an application’s business logic, especially if the code solves problems that have already been solved," he said.

Most panelists agreed that the way out of complexity — and indeed the survival of the Java platform — hinges on innovation in Java-based frameworks, tools, and languages.

What do you think?
We asked our esteemed panelists some big questions for this
"For Java to remain viable it must be seen as a platform that enables rapid deployment of working applications," said Andrew Glover. He cited Grails as an example of a simplified enterprise stack built on top of the Java platform.

"With frameworks like Grails, you can build working Web applications extremely fast, just as you could with Rails," he said.

Other technologies that support the evolution of the Java platform, said panelists, include the Spring framework, Eclipse, Java-based cloud and virtualization technologies, key/value store databases, and Java EE 6 itself.

Technology won't solve all of the problems facing Java developers; however, Java's programming culture is also under pressure to change.

"Java developers stereotypically seem to fetishize complexity," said Neal Ford. "We as developers must remain diligent to balance solving problems without creating self-imposed new ones."

Steve Perry agreed. "Innovation is great, and I love change, but software development really needs to grow up," he said. "The world is relying more and more on software, and if the way we develop it doesn't improve, we're going to go through some real pains."

According to Chris Mackie, the movement of software development activities out of enterprises is driving the evolution of both Java culture and technology. "Outsourcing vendors, where the jobs are growing, are different types of firms with different organizational cultures and different coding needs," he said, even "highly skilled and experienced Java developers may find the transition to the new environments challenging."

Oracle and Java technology

After months of litigation, Oracle now has an ownership stake in the future of Java. We asked panelists what they thought was most important about the new ownership, and what they'd like to see change, or stay the same, under Oracle's stewardship.

Most, like Rick Hightower, didn't anticipate much change on the technology front. "Oracle is not going to go out of its way to make things hard for Java developers," he said.
Several panelists expressed concern about the JCP and the hope that it would continue, or perhaps become more robust, under Oracle. "I'd love to see Oracle eliminate the field-of-use restrictions that Apache has been objecting to for years and break the JCP logjam holding up the Java 7 JSR," said Alex Miller.

"My hope is that Oracle's promises to keep the JCP alive and make it more participatory aren't just PR platitudes," added Steve Perry. "Is it going to get more expensive for companies to participate? What about individual participation?"

Speaking about IDEs, Mik Kersten called the choice of JDeveloper as first among equals problematic, "given Eclipse's lead in terms of Java features, extensibility, and available integrations."

"I hope that we see Oracle recognize the importance of Eclipse to Java's success," he added.

"Longer term, it will be interesting to see if Oracle steers Java closer to enterprises, or whether it nudges Java out into the cloud," said Chris Mackie. "I think that a stronger cloud focus would be beneficial to the community, but wonder if Oracle sees it as a priority."

"Leveraging the cloud also makes a hardware move less exciting, which puts Oracle in an interesting position as a hardware company (with Solaris)," said Andrew Glover.

On the question of what's more important to Java technology — the change in corporate ownership or the rise of open source alternatives — the panel was split.

"For me, the momentum around Java isn't related to ownership, but to the community at large and its contributions to open source," said Glover. But, said Dan Allen, it does matter "that the company in charge of Java finally knows what to do with it."

"We now have new people with different intentions in charge and that can only change how Java and the JCP (via Oracle's controlling stake) evolve," said Alex Miller. "I think Oracle will mix things up in the Sun Java products and do a better job of converting that into cash."

It seems clear that Oracle's profit-driven business model will transform some aspects of Java culture.

"A lot of Java developers are going to howl like the proverbial wolfman when they start seeing the things that used to be free disappear behind a tollbooth," said Ted Neward. "And frankly, I think this change has been a long time coming."
Java 7: Closures and concurrency

More than three years in progress, Java 7 is currently slated for release in late 2010. We asked whether closures are still worth arguing about, and whether JDK 7 goes far enough in meeting challenges like concurrency.

Some questioned the relevance of Java 7 for on-the-ground application development. "I still deal with clients who are stuck on Java 1.3 because of contracts with commercial companies providing application servers," said Andrew Glover. "Business realities like these make the adoption of Java 7 seem far off for some portion of the Java world."

Java 4 and Java 5 brought exciting features to the field, said Steve Perry, "But with Java 7 not so much." However, working with established and stable technologies serves his interests as a consultant, he said.

Some of the most interesting work on concurrency is happening independent of Java 7, noted Glover, such as with the open-source, Java-based framework Kilim and functional languages like Scala.

"JSR 203 is a great addition for NIO. JSR 166 has a bunch of nice concurrency updates, and I'm holding on to a hope that JSR 310 will still make it in for a date and time overhaul," said Alex Miller.

Miller also saw good additions to the JVM, with "great work going on for invokedynamic and what looks to be a pretty hot new garbage collector, G1."

As for closures, said Miller, they're "still worth arguing about, and I'm glad they're back in the mix."
Among other things, closures would provide "a common standard bytecode format for communication" among the various JVM languages, said Neal Ford. "Now, Scala uses a different internal representation than JRuby, Clojure, and Groovy; they all have their own bespoke format." Having one internal format would enable all of the Java-based languages to resolve to the same internal representation, he said.

"The success of Groovy is a pretty damn good sign that people want closures in Java," said Dan Allen. He also expressed disappointment with the lack of properties and direct field/method references in Java, which he said leads to waste and/or violation of type-safety.

"A lot of language information is lost at compile-time that would otherwise be tremendously useful for framework developers," said Allen. "One example is the name of method parameters and various generics type information, which get dropped moving from Java to bytecode."

"While annotations are brilliant, they only scratch the surface of the metadata that needs to be supported," he said.

Ultimately, however, "the Java language designers should just freeze Java at version 7," said Ford. "Java the language will become like the assembly language of the JVM: a place to go for low-level access, because you are writing a low-level library, and for maintenance."

Even on the platform front, said Miller, "Java is not at a point in its lifecycle where revolutionary change can or will happen — it's mostly going to look like it does and do what you expect. And for the enterprise that's a good thing."

**Project Jigsaw**
The argument that the JDK ought to be modularized goes back to the mid '90s, but interest in Java modularity — or perhaps hope for a realizable solution — has exploded in the past few years. Project Jigsaw, part of the OpenJDK project, is Sun's effort to modularize the JDK in the Java 7 release.

We asked panelists what they thought of Project Jigsaw in relation to OSGi, and whether it could meet the need for a modularized Java platform and programming model.

"I suspect that there is no single 'right modularity solution' for Java," said Chris Mackie. "It's easy to discern very different use cases lurking behind various positions regarding the 'right' sort of modularity."

In referring to the various JSRs on modularization, Alex Miller noted, "I think the entire process around JSR 294, 277, and Project Jigsaw has been a mess from the beginning." He continued, "The people involved are universally smart and well-intentioned, yet I have little hope that the final outcome will be something that I want to use."

"I think OSGi is a great technology but I don't think it's the one and only possible answer either," he said.

What we need is a "facility that allows for version tracking, some compile-time and runtime support for version dependency checking, and an open metadata system for components," said Ted Neward. Instead, Project Jigsaw has "turned into yet another engineers-gone-wild love-child of OSGi and Maven," he said.

"It's a shame because I think the vision that Mark Reinhold and Alex Buckley have been showcasing around modularity is actually very compelling," added Miller. "The notion of integrating module definition, dependency management, and deployment in a cohesive way could enable systems like CPAN, Ruby Gems, or Groovy Grape, and kickstart a whole better way to build and deploy Java software."

Still, modularizing the Java platform has its downsides and detractors.

"I just can't get that worked up about Project Jigsaw," said Steve Perry. "What they've proposed solves use cases I hardly ever care about."

"The fact that modularity solutions are only now maturing suggests that a very large portion of existing Java apps, while they might benefit from modularity, have no essential need for it," said Mackie. For many applications "modularity is likely to bring more immediate pain than gain."

Modularity could, however, be critical in Java development for mobile environments, he said. "It's hard to see how, for example, Java thrives on the smartphone without it."
Mobile applications "simply don't need all the baggage that the JVM brings," said Dan Allen. "The Android and Java ME split may not have happened had Sun been open to modularizing the JDK" sooner.

Java and the multi-language VM

Does the JVM need "one true language"?
Like the Java community itself, panelists were split on the need for a core JVM language around which other, lesser languages revolve.

According to Neal Ford, "Java is dead. Period." But the future of polyglot programming looks bright, he said. Taking Swing development as an example, "you can use a DSL in JRuby (Swiby) or Groovy (SwingBuilder) to build the user interface, build the critical thread-safe parts in a state-of-the-art functional language like Clojure, and keep all your Java libraries you've accreted over the years for talking to mainframes."

To Steve Perry, however, that scenario looked like more work, without a tangible business benefit. "As a consultant, I try to stay aloof from the churn of technology," he said, "and only jump in when something really catches my attention."

Dan Allen, meanwhile, called the JVM a perfect sandbox. "Once a stellar language is discovered," he said, "developers can actually migrate to it without dumping everything they know and have invested in." The success of Groovy and Scala prove that the sandbox approach is viable, he said, but neither language is "the one." Both "make compromises that make me believe there is still something better out there."

Has the age of the "one true JVM language" passed, or is it that we just haven't found the right one yet? Let us know what you think!

For some time now, certain Java community pundits (several of whom write for developerWorks) have debated whether the Java language is dead, or dead like COBOL (see Resources). The deeper question today is whether the death or life of Java the language matters, given the plethora of languages now residing on the JVM.

The multi-language VM "makes the 'Java is dead' argument less and less interesting," said Andrew Glover. "Java is a platform! If the Java language lacks some feature I need (or see in another language) chances are some alternate JVM language has it."

"The real genius in Java was the clean separation of the language from the virtual machine," said Neal Ford. The robustness of the JVM allows developers to "mix and match languages, using each to their strengths," he said.
The multi-language VM (as realized by the OpenJDK Da Vinci Machine project) will only create opportunities for Java developers, said Alex Miller, "driving more work into language interoperability, metaobject protocols, and other good stuff."

Miller and others agreed that we'll continue to see heavy investment in Java from big enterprise software companies.

"Java is a very easy language to understand and is quite good for the corporate developer," said Dan Allen. The challenge is to evolve the language, and do so aggressively. "New language features need to come in, old language features need to be deprecated, and deprecations need to be eliminated! Nobody using Java 5 annotation is still using deprecated code from Java 1.1."

Ted Neward also called on the Java language designers to "look at the other languages in their rough niche — C++, C#, and Visual Basic to start — and figure out a set of new features that could make Java exciting and interesting again."

"Java gives us a great base in terms of an object-oriented programming language," said Mik Kersten. "It's done, it works, and its ongoing evolution will take incremental steps. But as a community we do need to ensure that additional innovation can happen on the JVM."

What's next for Spring?

One of Java's most robust and popular development platforms changed hands in 2009, when VMWare bought SpringSource. We asked panelists how they saw the Spring Portfolio evolving under VMWare.

There seemed a general consensus that Spring, under VMWare, will move into cloud computing. Spring will "become the platform for doing cloud development," said Ted Neward, "and incidentally continue to support the enterprise development it's been supporting up until now."

Mik Kersten concurred. "SpringSource's CloudFoundry is a glimpse of the future, where developers deploy, debug and manage their apps in the cloud, all from within their IDE," he said.

From a business application developer's perspective, Spring might be trying to do too much, said Steve Perry, and "approaching the 800-pound gorilla-size box." Developers shouldn't have to upgrade their Spring implementation every three months, he said.

Rick Hightower also saw the Spring Portfolio "growing in leaps and bounds in every direction." "Spring Roo and Spring STS are great examples of doing things the 'Spring way.'"
Emerging Spring Portfolio products may be getting back to Spring's roots in aspect-oriented programming, he added. "If you have not done so already, you may want to knock the dust off of your AspectJ books."

OpenJDK and Apache Harmony

The decision to open source the Java platform was a welcome one and the result, at least in part, of enormous and ongoing pressure from the Java developer community. Apache Harmony arose first, but OpenJDK had Sun's weight behind it. With the two implementations underway, we asked panelists whether the Java ecosystem benefits from two open source Java platform implementations.

"Different implementations provide a greater chance for finding an optimum solution to the problem, or a solution that caters to a specialized need," said Dan Allen. "For instance, one JVM might be faster to start up, while the other may maintain lower memory consumption over the long run."

"There's a sense in which we have a perfect niche strategy," allowed Chris Mackie. But, over time "it becomes harder to justify partly competing projects — not to their creators, but to the growing numbers of users of both projects who don't necessarily share the creators' passions or self-interest."

What users do share, he said, is "an interest in maximizing return-on-investment."

A lot of Java developers are going to howl like the proverbial wolfman when they start seeing the things that used to be free disappear behind a tollbooth. And frankly, I think this change has been a long time coming.

—Ted Neward

For some, Apache Harmony and OpenJDK epitomize what works, and what doesn't, about open source.

"It's a nice idea, to have the Java source available," said Ted Neward, "and I'm
personally very glad they went to the effort of doing so. But let's not kid ourselves — the two percent of the Java developer community who care about it aren't doing much with it in an open-source manner."

"Open source is a beautiful thing," said Andrew Glover, "especially when you are paid to support and innovate around it. Projects that have a commercial backer for the most part seem to thrive and grow at a faster rate than those that rely on true volunteers."

As for the technology, said Rick Hightower, "OpenJDK and Apache Harmony are like safety parachutes. I am glad they both exist; I hope I never have to use either of them. I hope Oracle is a good steward of the Java platform and that we don't have to rely on the backup plan."

Cloud computing

With influential companies focusing substantial computing and marketing resources on the cloud, we asked panelists for their on-the-ground perspective on the state of cloud computing, and what would be a realistic approach to leveraging cloud infrastructure in the enterprise today.

"Cloud in 2010 is the same as virtualization was for 2008, as grid was for 2007, and as Web services were for 2005," said Ted Neward. "It’s wispy, hard to hold on to, and despite a seemingly solid appearance, has no substance to it."

"'Cloud computing' is a buzzword in the hype cycle," agreed Alex Miller, "but the related technology is real and useful now. Practically every developer I know is either using or looking at cloud infrastructure in either private or public clouds," he said.
"The obvious right-now approach is to leverage something like Amazon's EC2 for testing," said Andrew Glover. "There is no need to spend cash up front for test infrastructure hardware when you can pay as you go."

Miller agreed: "Most groups seem to be getting their feet wet with scalability and performance testing or deploying internal apps as they get familiar with the pros and cons," he said.

"Provisioning a server is a huge overhead and maintaining it is a burden," said Dan Allen. "If all of that can be abstracted, and controlled from the IDE [...] the developer can focus on the business logic."

Some expressed concern that developers, and organizations, may underestimate the work involved in porting applications to a new cloud infrastructure.

"If I have a customer who wants a cloud, I'll get some books, read up on it, and build 'em one," said Steve Perry. "But until that happens, I'm staying out of the cloud."

"We don't want to force developers to hack their platforms, frameworks, and tools just so they run in the cloud," said Allen.

Chris Mackie agreed. "Sticking most current enterprise apps up on EC2 is easier said than done," he said. Which means, of course, that cloud savvy developers "are likely to find themselves in near-term demand."

Java technology in new industries

**IBM and other corporations are investing heavily in the idea that better technology is key to building a smarter planet. It's a simple premise, but**
software developers know better than most that new initiatives also bring new challenges. We asked panelists about the heavy lifting that would be required to bring smarter technology into new industries, and also what industries would most benefit from an infusion of intelligent software solutions.

"Two industries come to mind immediately: health care and finance, particularly banking," said Steve Perry. "I believe tighter regulations will mean we need better software methods" in both industries, he said.

Dan Allen also cited the lack of technology infrastructure in health care. "People from halfway around the world can share what they are eating for breakfast (thanks to Twitter), yet after a month my wife's doctor's office still can't locate the X-rays from the lab," he said. "The key here is simple, lightweight Web services like REST."

Both Allen and Andrew Glover saw room for expansion in mobile technology. "Mobile is the way to go," said Glover. "A smarter planet will leverage mobile computing for greater information sharing and efficiencies."

For Java developers, "embracing and bolstering Android will be key," said Allen. "But that means addressing and standardizing the problem of multimedia in Java."

Chris Mackie pointed out another area where Java technology could make a difference. "One of the most distinguishing features of the nonprofit sector today is that it is the only remaining, major economic sector worldwide that has not fully realized the productivity gains of current-generation technologies," he said. "Just bringing today's technology to nonprofits is arguably the biggest potential win that's sitting on the table right now."

Some cautioned against expanding into new industries before "fixing the basic flaws in our own house," as Ted Neward put it.

Engineering quality software takes time and attention to craft, neither of which the software industry currently supports, said Steve Perry. "We want high productivity, and that, in my opinion, almost always results in low maintainability."

For software to be of real use outside the enterprise, not only quality but usability will need to improve, said Neward. Most software today "requires a fairly hefty up-front commitment — either in installing other stuff that it needs, or in building it from source, or in installing the necessary bits required to install the bits that are required, ad infinitum — before it starts to pay off," he said.

"Being able to build, change, and deploy technology faster than your competitor is crucial," said Alex Miller, leaving most industries open to impact. But, he said, "managing complexity is critical to making Java approachable."
Open source: Where's the money?

Few Java developers can imagine a world without the plethora of free and open source libraries, frameworks, and tools that are essential to many Java applications built today. But the financial conundrum of open source development sometimes causes pundits to question its long-term viability. We asked panelists about the future of open source, and about what works, and doesn’t, for developers and organizations seeking to make a living from open source, and not just give away software for free.

Most panelists agreed that open source software will continue to be viable for the foreseeable future. Vendors will continue to make money primarily from support and training around their products, along with packaged enterprise versions of tools also available for free.

"IBM Rational® has succeeded at making Eclipse the open source tool platform and continues to sell innovative products on top of that platform," Mik Kersten said.

Open source projects that want to succeed must have a "sustainable momentum of growth," said Andrew Glover. Not all open source business models succeed. "The underlying product needs a diverse and large community to support a commercial entity offering support," said Glover.

Maturity is also key, said Chris Mackie. "Open source has such a compelling value proposition for mature, vertical software markets that there is almost no chance that it will disappear." In fact, Mackie noted the tendency of initial, proprietary innovations to "give way over time to durable, open source alternatives."

Proprietary software vendors can be more agile and can concentrate capital faster
and more efficiently, he said. But open source has the advantage of human capital, which it leverages "in quantities vastly greater than any proprietary vertical market vendor can match."

"If you want to make money with open source, figure out how you’re going to make money, then figure out how open source can help you," said Ted Neward. "Going the other way hasn't worked yet."

He cited Oracle’s acquisition of Sun as proof that "companies who took a slower adoption path to open source — namely, Microsoft and Oracle — have had better success at figuring out where and how open source will help their business rather than hurt it."

"IBM has made billions on open source software, and invested a lot into open source as well," said Rick Hightower. It's "the poster child for how to make money on open source software and how to give back to the community."

"I hope we'll see more companies like Oracle and VMWare shepherding [open source] technology," added Steve Perry."

Conclusion

Java technology is almost 20 years old, and clearly on the cusp of a new era in its evolution. Oracle’s stewardship has an important role in shaping the future of Java technology, but so do the will and creativity of the Java development and open source communities.

The Java Community Process has emerged as a topic of real concern with many developers wondering whether — and exactly how — Oracle will keep its promise to both maintain and revitalize the JCP. Many expect the model of free and open source software to change under Oracle — whether subtly or dramatically — and those changes will impact how Java developers relate to the technology and their own contributions to it.

As Java technology enters its maturity, the culture of Java programming is also maturing. The drive for innovation isn't gone, but it's balanced by a growing recognition — both in the programming trenches and among CTOs and business managers — that code must be written with maintainability in mind.

Another force shaping the future of the Java platform is industry making new demands on technology while remaining, in some ways, mired in outdated thinking and ways of doing things. That must change for the Java platform to evolve and keep pace with industry needs.

If there's one message from this roundtable, it's that the future of Java technology is
exciting — and also challenging. For Java developers who are able to excel in the space where rapidly evolving technology (such as cloud and mobile computing) meets a specific industry (such as health care and finance), the future looks bright and very busy.

*It's your turn:* Add your input to this roundtable discussion on forces shaping the future of the Java industry.
Resources

Learn

• "Podcast with Trent Gray-Donald (developerWorks, April 2010): The technical lead on the IBM SDK for Java 7 talks with developerWorks about Java 7 and the value-add in the IBM SDK.

• "Java technology, IBM style: A new era in Java technology" (Chris Bailey, developerWorks, April 2010): Learn how IBM will add even more punch to the Oracle Java 7 release due in late 2010.

• "OTN Tech Cast LIVE: Justin Kestelyn Interviews Mark Reinhold" (Editor's Daily Blog, Java.net, February 2010): A synopsis of the complete Tech Cast interview with Java SE Principal Engineer Mark Reinhold, on the topic of what's coming in Java 7.

• "Oracle and the Future of the JCP" (Mitchell Pronschinske, JavaLobby, February 2010): Members of the JCP Executive Committee share their thoughts about Oracle's announced intention to make the JCP more participatory.

• Pure Danger Tech: Java 7: Alex Miller’s exhaustive listing of proposed features for Java 7 and information about them.

• "Dead like COBOL?" (Ted Neward, developerWorks, May 2008): Ted Neward takes on the question of whether the Java platform is dying; and if so, what's replacing it.

• Java development 2.0 (Andrew Glover, developerWorks): An article series focusing on technologies redefining Java development, including cloud computing, polyglot programming, and new approaches to concurrency (like Kilim).

• Language, expressiveness, and design, Part 1" (Neal Ford, developerWorks, July 2009): Neal Ford recasts four classic Gang of Four patterns using Groovy, a dynamic language for the JVM. Part of Ford's developerWorks series, Evolutionary architecture and emergent design.

• "The case for Java modularity" (Jeff Hanson, JavaWorld.com, August 2008): A Java engineer explains why modularity is an important evolutionary step for the Java platform.

• Java Specification Requests: A complete listing of JSRs, including ones slated for release in Java 7.

• Java platform roundtable contributors: The roundtable was enriched by the quality of its contributors, all of them Java community thought leaders and active participants in the evolution of Java technology, culture, and industry.

• Browse the technology bookstore for books on these and other technical topics.
• developerWorks Java technology zone: Find hundreds of articles about every aspect of Java programming.

• developerWorks Cloud computing zone: A collection of resources from developerWorks to guide you at every stage of cloud development, helping you learn key skills and solve problems.

Discuss

• Get involved in the My developerWorks community.

About the authors

Jenni Aloi

Jenni Aloi has managed the developerWorks Java zone since November 2000. Prior to that she served as managing editor of IDG’s Web Publishing Inc.; senior editor of JavaWorld.com; editor at Coriolis Group Books; and editor at ComputerPrep, developing training manuals. Though she often complains (about nearly everything), she is happy to have spent close to 20 years doing what she does best: herding cats.

Athen O’Shea

Athen O’Shea is a high technology editor and writer, and former editor of JavaWorld.com. As a freelancer, he has plied his trade for developerWorks since 2001.

Trademarks

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.