Mark Little, vice president of engineering at Red Hat, overlooks a clean room cooling and air filtration system in the Newcastle University building where the Red Hat offices are located in northeast England.

JCP Executive Series

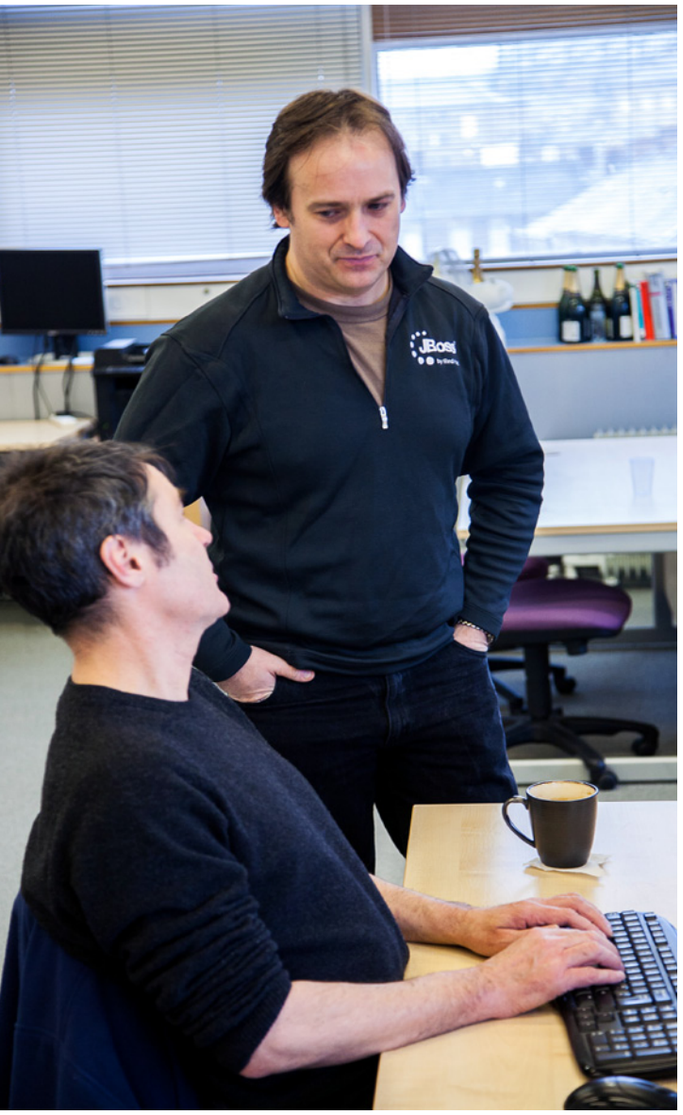
A Conversation with Mark Little

Red Hat's vice president of engineering discusses Java EE 7 and the JCP.

BY STEVE MELOAN

Continuing our series of interviews with distinguished members of the Executive Committee of the Java Community Process (JCP), we turn to Mark Little, vice president of engineering at Red Hat. Initially known for its Enterprise Linux OS, Red Hat acquired JBoss in 2006 and added enterprise middleware to its roster of technology offerings. Red Hat participated heavily in the development of both the Java EE 6 specification and the just-released Java EE 7 specification.

PHOTOGRAPHY BY JOHN BLYTHE



LIttle confers with a colleague.

Java Magazine: Red Hat has been very involved in the evolution of Java EE 7. Can you comment on leading the specifications for CDI [Contexts and Dependency Injection] 1.1, and Bean Validation, and participating in the development of a number of other JSRs?

Little: We led the CDI and Bean Validation updates, because we were already leading those in Java EE 6. And as you indicated, we've been very active on a number of other JSRs—including JTA [Java Transaction API], JCA [Java EE Connector Architecture], and JMS [Java Message Service] updates.

We've tried to be actively involved, in one way or another, with all of the JSRs that have been updated in Java EE 7. Even if we're not leading them, it's still important that we bring the perspective of Red Hat customers and our wider open source community to how Java EE is being adopted.

Java Magazine: Can you comment on the importance of these JSRs within Java EE 7, and your experiences during their refinement and evolution?

Little: We think that CDI is probably the most important addition to the whole Java EE architecture that we've seen in a number of years. And I don't just say that because Red Hat led the effort, because it was really a group effort. I say it because we were hearing from many vendors that it was hard at times to develop applications using the standard Java EE stack. CDI has tried to greatly simplify that. So annotations are a great addition to the language, and we're now seeing them being used in lots of different areas.

Since Java EE 6 was released, we've seen the momentum around it building. We're seeing a lot more people who didn't consider Java EE 6 on their radar screen but who are now really taking a look at it as a means of simplifying the development of enterprise applications. And they mention CDI time and time again. So it was pretty obvious to us that we wanted to lead the update to CDI in Java EE 7, because there were some things that we couldn't do in the version that went into Java EE 6. And there was also feedback that we'd gotten from users when Java EE 6 was finalized that we wanted to take into account.

In terms of the process, we've done pretty much the same this time around as we did with Java EE 6. All of our processes are open—so we have an open mailing list, all the participants see what everybody else is talking about, and we have open issue tracking. The drafts go through a very wide revision process.

Java Magazine: How has the Web Profile of Java EE 6 affected Red Hat technologies, and how will that further change for Java EE 7—in terms of new APIs like WebSocket and JSON-P?

Little: The growth and complexity of the Java EE stack is really not unique. If you look at CORBA and DCE and other standards, you see a similar phenomenon. Java EE 6 recognized that fact and introduced the concept of a Web Profile, which is essentially a stripped-down version of the full profile.

In terms of the impact that it had on Red Hat, we'd been delivering our own version of profiles to our communities for some time. There was no standard that we could present, but we could offer the ability to streamline the stack. If a customer didn't want Web services, for instance, we could provide that.

So I think the Web Profile was a really good thing to introduce as a standard. And the feedback that we've gotten has been extremely positive. We see a lot of people who might not have considered Java EE 6, who are now looking at the Web Profile, and then eventually upgrading to the full profile because some of the things that they want aren't in the Web Profile. So I think it's a really good way to on-board more users, and from a company perspective, to gain new customers.

We saw a lot of new APIs in Java EE 6, including JAX-RS and CDI, which have received a lot of attention for the benefits they've brought to developers. With

BIG SPEC

"We think that CDI is probably the most important addition to the whole Java EE architecture that we've seen in a number of years."

Java EE 7, we're seeing the APIs extend yet again with additions such as JSON-P and WebSocket, both of which are vital for keeping the enterprise Java stack at the forefront of waves such as cloud and mobile.

Java Magazine: A variety of cloud-related features originally intended for Java EE 7 have been deferred to Java EE 8. Can you comment on this change, how Java EE is used in cloud environments today, and how the technology will evolve in post–Java EE 7 releases?

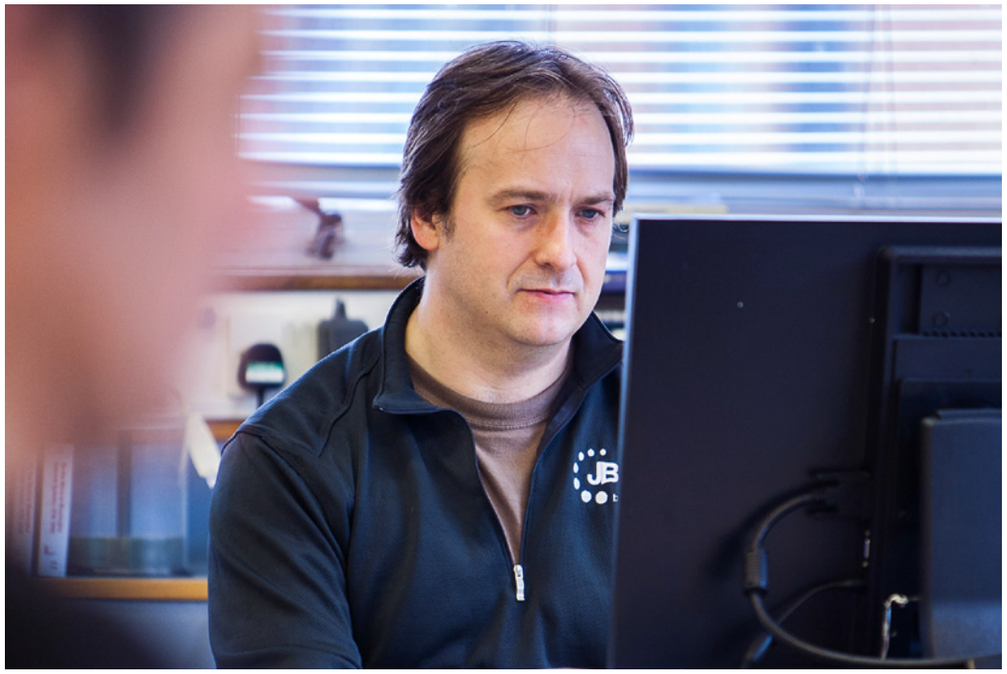
Little: From discussions with our communities and our customers, what they wanted was the ability to offload applications from their own infrastructure onto somebody else's, but without having to reimplement. And the obvious way to do that is to ensure that the platform you've been running with your own hardware is on the cloud. From very early on, we've been working to ensure that our implementations will run on an infrastructure as a service, and therefore form a platform as a service.

When Java EE 6 came along, it actually offered us an easier route to do that, because with the profiles introduced in Java EE 6, it became easier for us to offer a standard Web Profile and a standard full-profile platform to customers who want to run their applications in the cloud. We announced our own platform as a service back in 2011, which was initially based on a pre-release of our Java EE 6–compliant application server. And then we released EAP6, which is our full implementation, where we made the announcement of OpenShift, our platform-as-a-service offering.

When we were originally working on Java EE 7, there were quite a lot of new JSRs that were going to be focused on making Java EE more cloud-aware. But as I said, Java EE 6 is pretty darn good in the cloud today. There are certain areas where it can be improved—in terms of modularity and multitenancy. But I agree with the deferment move on these features, so that we could get Java EE 7 out on schedule. Those features will be in the next release, and therefore Java EE 8 will be even better for evolving clouds.

Java Magazine: How will the JBoss Developer Studio IDE reflect/utilize the new offerings found in Java EE 7?

Little: We try to keep JDBS at the vanguard of getting things in front of the actual developers, so we can determine pretty quickly where the problems are. If a change in CDI isn't really right, for example, then we'll get a lot of feedback from developers through JDBS. So it's important to us that we get these features into the IDE as quickly as we possibly can, so we can get people to kick the tires.



Little says that open source communities "drive everything we do at Red Hat."

Java Magazine: Java EE 7 has pruned a number of older features (JSR 77, JSR 88, JSR 93, JSR 101, and so on). How will this be addressed in Red Hat's Java EE 7 offerings?

Little: This isn't the first time that JSRs have been pruned. What we tend to do, and what I expect we will do with these JSRs, is if they're no longer in the Java EE 7 spec, then we will remove them from our Java EE 7–compliant implementation.

But we've got the Java EE 6 implementation—AS7 [JBoss Application Server 7] is the community version, and EAP6 [JBoss Enterprise Application 6] is the product version. And EAP6 is supported for seven years. So custom-

INNOVATION'S ROLE

"One of the things that we're trying to do in the Executive Committee is to encourage innovation in upstream, open source communities, around things that we might want to see in Java EE 8 and Java EE 9."

ers who really need those features can have them in a supported way for another five or six years. For JSRs that become optional, we will typically support them in one way or another. We won't remove them until they're officially removed from the Java EE stack.

Java Magazine: In the big-picture sense of JCP involvement, how do you resolve and reconcile diverse interests when working with other member companies?

Little: Everyone understands that Java and Java EE are very important to a huge part of our industry. So we have a duty to work together in its evolution. It's really no different than working in OASIS or W3C or any of the other standards groups. Issues may come up, and you won't always get everything that you want, but you have to compromise.

Java Magazine: What are the issues and balances between preserving current standards and promoting future innovation?

Little: A big danger is standardizing too early—because all of the good standards are based on experience. If you standardize too early, you end up with APIs or data formats that are not fit for the purpose, and then nobody uses them.

But it's a trade-off—if you wait too long to standardize, then you can end up with vendor lock-in, or with multiple different ways of doing the same thing. One of the things that we're trying to do in the Executive Committee is to encourage innovation in upstream, open source communities, around things that we might want to see in Java EE 8 and Java EE 9.

Java Magazine: How do you think the JCP can better serve the Java community, and how would you suggest promoting greater participation?

Little: The work that the various Java user groups have been doing, like Adopt-a-JSR, is great. Everything we do at Red Hat is driven by open source communities—we get a lot of feedback on things that we're going to propose in the various JSRs long before we might actually include it in a document. So we try and bring our communities into the Executive Committee in that way.

Java Magazine: At last year's JavaOne, you and Cameron Purdy gave a session exploring non-Java languages running on the JVM [Java Virtual Machine]—how a Ruby developer, for example, can leverage the Java EE stack, utilizing features such as JMS or EJBs [Enterprise JavaBeans]. Can you discuss this trend, in terms of an expanding Java community/ecosystem?

Little: Over the last several years, we've seen a growing adoption of languages like JRuby and Clojure and Scala that run on the JVM. And we're developing our own language, called Ceylon. What we've been trying to do with these communities is encourage developers who might have needs for high-performance messaging, or transactions, or databases not to reinvent the wheel when there's already a perfectly good set of solutions in terms of the Java EE stack.

So we've tried to find out what they expect from a given API, say for messaging, write something with the wider developer community, and then layer our stack, or at least part of our stack, underneath that. So Ruby developers are defining the API, and we're providing a binding of that API to our JMS implementation.

The last thing we want with these new language communities is for them to reinvent the same bugs that we had 20 years ago. It's far better for them to use what we've already got and improve upon it. And we have some very successful projects—like the Ruby project, called TorqueBox. Ultimately, the developers don't even need to know that they're running on top of an application server, or that they're using a Java EE stack beneath the covers. `</article>`

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

LEARN MORE

- Java Community Process
- Red Hat