



55

PHOTOGRAPHY BY PHIL SALTONSTALL



JAVA EE 7 ARRIVES

Anil Gaur on the evolution of Java EE **BY STEVE MELOAN**



As the industry standard for enterprise Java computing, Java Platform, Enterprise Edition, has constantly evolved to keep pace with

the emerging usages, patterns, frameworks, and technologies of the enterprise space. Java EE 7 is now here, with essential support for modern-era Web applications—including HTML5, WebSocket, Java API for RESTful Web Services (JAX-RS), JSON Processing (JSON-P), and more.

Java Magazine sat down with Anil Gaur, vice president of software development at Oracle, to explore the new features and technologies of the just-released Java EE 7

Anil Gaur (center), vice president of software development at Oracle, chats with Java EE 7 Platform Specification Leads Bill Shannon and Linda DeMichiel.











Gaur discusses Java EE 7 capabilities with Arun Gupta (dark shirt), Java evangelist at Oracle, and Shreedhar Ganapathy, engineering manager at Oracle.

platform and examine what it means to enterprise developers and customers. **Java Magazine:** What's new and notable in Java EE 7?

Gaur: The Java EE platform is at a critical point right now, in that we have to rapidly move forward to keep pace with emerging Java usages and patterns. These changes are being greatly driven by modern Web applications. Over the past few years, many other frameworks have evolved, and we also wanted to align the platform with those changes.

So the main theme of Java EE 7 is productivity and HTML5 support. The HTML5 enhancement comes from WebSocket, Java Servlet 3.1, the NIO [non-blocking I/O] feature, Server-Sent Events [SSE], JAX-RS, and JSON-P.

In addition, several enhancements have been made throughout the platform to improve productivity, because that's another pain point, and it directly reflects on the cost of development and maintaining applications. With Java EE 7, developers will have

to write less boilerplate code, and the platform will also offer richer functionality through various new specifications, such as <u>batch processing</u> and <u>concurrency</u>.

What's more, the JMS [Java Message Service] spec has gone through major enhancements in JMS 2.0, where we've introduced some of the concepts already available in the Servlet and EJB [Enterprise JavaBeans] specs. And the Expression Language specification has also been enhanced in version 3.0, which will benefit many other technologies within the platform.

The various specs that have gone through changes are JPA 2.1 [Java Persistence API], Java Servlet 3.1, EJB 3.2, Bean Validation 1.1, and CDI [Contexts and Dependency Injection for the Java EE platform], which was added in Java EE 6 and was one of the most popular and widely used specs there. Java Magazine: How will Java EE 7 provide developer simplification while also offering richer application possibilities?

Gaur: We've particularly focused on simplification in terms of introducing the concept of profiles. We've heard many times that the platform has been growing into a monolithic bundle that was sometimes difficult for very specific, targeted usages. And we took that very seriously. So we invested heavily in the Java EE 6 platform, where we introduced the Web Profile, a runtime stack of standard APIs specifically targeted to Web application development. And that profile has allowed us to put in place a basic framework that we can use toward defining future profiles. Developer simplification also occurs by providing more defaults and by using CDI, which reduces the use of XML coding.

We've seen great demand for the new specs that we've added in Java EE 7. People were already using technologies such as WebSocket and JSON, but prior to Java EE 7, there were no standard APIs. Now, developers no longer need to package their own JSON libraries in the applications. The

support is in the base platform. And they can invoke WebSocket endpoints directly from Java clients. So that also makes for significant development simplification, while helping to minimize the size of the applications that are deployed.

Batch processing is another addition to the platform, which has been regularly requested by customers. This provides a standardized programming model to implement batch applications. The JSR defines the domain area, batch job, application executor, and job manager. And in addition to the support in Java EE, some of the features will also be included in Java SE. But Java EE is a superset of Java SE, so it inherently provides a richer feature set. JTA [Java Transaction API], for example, is only found in Java EE.

We've also added concurrency to the platform, which had similarly been requested by developers. This allows application-level work to be performed concurrently, using different threads of execution. Prior to this release, various application servers had their own proprietary implementations, but that reduced application portability. Their inclusion in Java EE 7 provides a standard API available to all EE vendors, customers, and developers.

And, as I indicated earlier, the JMS 2.0 spec has been revised and simplified in a variety of ways that had been requested by the community. JMS 2.0 will make greater use of annotations,

as did the EJB and Servlet specs in prior versions of the platform. JMS connection factory will also leverage Dependency Injection [DI]; in addition, the API is being cleaned up so that interim objects are not required, and then there are AutoCloseable resources. These changes should be very attractive to developers and allow them to start using JMS more extensively.

RESPONDING TO FEEDBACK

We've heard many times that the platform has been growing into a monolithic bundle that was sometimes difficult for very specific, targeted usages. And we took that very seriously.

So, yes, we've added more APIs to the platform, but most of the additions have been based on community requests and what was needed to solve the issues we've seen for developing enterprise applications.

Java Magazine: What's been pruned from the release, what does that achieve, and how does it potentially affect application backward compatibility?

Gaur: While we've added many new features to Java EE 7, we have also made certain APIs optional. So the

platform has grown in terms of functionality, but there has also been some cleanup of older features—which will not only help reduce the platform footprint, but free developers from having to deal with obsolete specifications.

This process began in Java EE 6, where we marked several technologies as optional, and targeted them for eventual pruning—Java EE Management [JSR 77]; Application Deployment [JSR 88]; JAXR, for interfacing with UDDI registries [JSR 93]; JAX-RPC, for XML-based RPC [JSR 101]; and EJB 2.x Container Managed Persistence, which is effectively replaced by the Java Persistence API [JSR 317].

Since these were marked for pruning in the Java EE 6 release, it makes it possible for us to remove them in this release. But that's optional, of course. It's up to each vendor to make that decision for the release. If they're seeing continued demand, if their customer base is still using these specs, they can keep them in the platform. But we will remove them in the next release []ava EE 8]. And I expect most vendors will follow that same strategy. Java Magazine: What does the Web Profile stack for Java EE 7 bring to the release and to Web application developers?

Gaur: The Web Profile was introduced in Java EE 6 and became very popular, because it offered a subset of the full platform, providing standard APIs that are specifically targeted to meet the





Gaur and Java EE 7 team leads share a lighthearted moment. needs of the majority of Web applications. Not every developer needs all the Java EE APIs, so this offers a reduced runtime memory footprint, simplified installation, and enhanced responsiveness.

So as with Java EE 6, the Java EE 7 release includes a Web Profile. But there are new specs being added— WebSocket and JSON-P. And the JAX-RS 2.0 enhancement, the client APIs, is another notable addition. These specifications are, of course, not restricted to just the Web Profile.

But more and more developers are using JavaScript and HTML5 in their applications, and WebSocket and JSON-P are two specs very well aligned with that programming model. So that makes it very attractive and suitable for next-generation Web application development.

Java Magazine: Can you discuss the decision to defer some cloud-related features originally intended for Java EE 7 to Java EE 8 and the features involved? Gaur: This was a tough decision, but I think we made the right call. And we consulted with our community members, Expert Groups, and vendors before making the final decision.

This decision occurred partly due to lack of maturity in the cloud space—specifically, in provisioning, multitenancy, and elasticity. We came to realize that fact when we were already well into the implementation. Ultimately, we wanted to avoid adding APIs or functionalities that were either not fully realized or not ideally suited to the intended goal.

So we chose a more conservative approach—to make sure that we reach our goal in a way that can maximally benefit the industry and developers. After consulting with community members, and conducting a survey, we decided to shift the focus of Java EE 7 toward HTML5 and productivity. This shift in scope allowed us to better retain our focus on enhancements, simplification, and usability—and to still deliver the release on schedule. In the end, we didn't want to hold up features that were ready to go for something that's not fully realized.

Java Magazine: What specific JSRs were deferred?

Gaur: There were no specific JSRs, as such. The cloud requirements were

cross-cutting, impacting many different specs. So no actual specs were dropped in the process. But features within several specs, including the Java EE 7 platform spec, were affected. Java Magazine: In the interim period, what facilities exist for running Java EE applications in the cloud?

Gaur: If you look at many of the vendors out there, in most cases, their cloud platforms are built on Java, which is great news for us. And Java EE 6 is already compatible with those environments. So the container-based model that we have in Java EE, along with security isolation, makes it very suitable for writing PaaS [platform as a service]—enabled applications. And JAX-RS, the API for RESTful Web Services, is also well tailored for cloud applications. So given all this, it's still a good platform for building cloud solutions.

Java Magazine: How has the Java community been involved in the evolution and refinement of Java EE 7, particularly with the recent cloud-related decisions?

Gaur: We have been developing the Java EE Reference Implementation as an open source project since Java EE 5, under Project GlassFish. And with Java EE 7, we decided to go even further, such that all the Java EE specs are now developed with participation from a broader community in addition to the Expert Groups and Java EE vendors. So over the last two or three years, the spec development process has



become more and more transparent. And that's not by accident. Oracle and the community had a strong desire to engage more community members—and earlier in the process—so that the community could review and provide feedback as we developed the specifications.

We filed JSR 348 to define the rules by which a JSR needs to be run—in a transparent manner. All the Oracle-led JSRs are now following that process. The Expert Groups have open e-mail aliases for technical discussions, the interim drafts of specifications are visible on the project page, and we use IIRA, which contains a list of all the features that we target for each of the JSRs. So everyone who is interested can now file a JIRA request if they want certain features to be added or if they have feedback on features that are already planned. Each JSR has a dashboard for JIRA requests.

This has allowed a much broader community to provide their ongoing feedback. And, as a result, we can deliver a superior technology at a faster pace. In addition, we engaged 19 Java user groups [JUGs] who participated in a program called Adopt-a-JSR, in which developers can volunteer their time to review what's being proposed and how it might impact technologies in the platform. JUGs have also developed several small applications to try out what's being proposed. The feedback provided by these JUGs



Gaur talks with team members Rajiv Mordani (left) and Nazrul Islam at Oracle's Santa Clara, California, offices.

helped iron out issues long before the platform was finalized.

Java Magazine: What features are now planned for Java EE 8, and what are the target dates?

Gaur: After the release of Java EE 7, Oracle and other vendors will begin working on Java EE 8. The focus of the release will be a standards-based cloud programming model—deferred from Java EE 7—in which we will add support for multi-tenancy and elasticity.

You can also expect to see modularity based on Jigsaw in Java EE 8. There is also strong interest in some sort of NoSQL support in the Java EE platform, so we will evaluate that as well. But in the meantime, the platform will continue to evolve. It's not as if, after Java EE 7, the only time you'll get to see something new is in Java EE 8. We'd like

to have new JSRs, such as JCache and JSON-B [JSON Binding], finalized well before Java EE 8. So there will be incremental progress in the platform, where our customers will be able to see and experiment with those new APIs before Java EE 8 is ultimately released.

In terms of a formal release date for

Java EE 8, the community will help define the scope and the features of the release, and that will ultimately determine the dates. The Expert Groups will play a major role in coming up with a final feature set and the release dates. Java Magazine: At JavaOne 2012, Hasan Rizvi pointed out that the ongoing success of Java is based on a triad of technology innovation, community participation, and Oracle's stewardship. Do you have any thoughts on further enhancing the partnership between

Oracle and the Java community?

Gaur: I'm pleased to say that the relationship with the lava community has strengthened quite significantly under Oracle's stewardship. And that has been demonstrated in several different ways. The community is very actively engaged

in defining the next-generation platform, and I expect that to continue. And we will continue to run the Expert Groups in an open fashion, which will enable more people to participate without requiring them to join the JCP or adhere to any heavyweight process. We will also continue to develop Reference Implementations as open source, and that will help us gather feedback quickly and, ultimately, develop a platform that is appealing to both enterprise customers and developers.

As evidence of this open approach, the Java EE 6 release was adopted more quickly than any previous release. I attribute that to the fact that we've been engaging the community at earlier stages, and I expect that we'll continue on that path with similar positive results.

Java Magazine: How can developers access Java EE 7?

LISTENING TO THE COMMUNITY We've added more APIs to the platform, but most of the additions have been based on community requests and what was needed to solve the issues we've seen for developing enterprise applications.

Gaur: The Java EE SDK is our most popular bundle, which developers can use to learn the new technologies in the platform. That is now available on Oracle Technology Network. The SDK bundle consists of GlassFish Server Open Source Edition 4.0, which is also the Reference

Implementation for Java EE 7. It contains a comprehensive tutorial that covers the new APIs, and there are many sample programs that will help developers understand each of the specifications and get up to speed faster.

I should also touch upon IDE support. The NetBeans IDE will be the first IDE to support the Java EE 7 APIs. We worked very closely with the NetBeans folks as we developed the platform, so that enables us to have a much tighter integration between the NetBeans IDE and GlassFish and a superior outof-the-box experience. And I expect Eclipse and Intelli] will be next in line with support for the Java EE 7 platform. **Java Magazine:** What can the GlassFish community expect with GlassFish 4.0, which is being released in conjunction with Java EE 7?

Gaur: As with Java EE 5 and Java EE 6, the Reference Implementation of Java EE 7 is derived from Project GlassFish.

Being the Reference Implementation, GlassFish is always up to date with the latest Java EE specifications. So when we ship the Java EE 7 platform, the community will have GlassFish Server Open Source Edition 4.0. It will offer support for all the Java EE 7 APIs, but the clustering and high availability features are there for evaluation purposes only. Those production deploymentcentric features will be made available in the subsequent release, 4.1.

GlassFish 4.0 will be available for download in two different flavors. The first distribution contains support for all the APIs in Java EE 7, while the second contains a subset of APIs defined by the Java EE 7 Web Profile specification. So developers will have a choice—they can start with the SDK bundle that contains the open source bits of GlassFish 4.0, or they can download the GlassFish bits separately from Java.net. </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

- Download Java EE 7
- GlassFish 4.0
- NetBeans IDE