Java EE und Java EE 8: eine Bestandsaufnahme

Peter Doschkinow, Wolfgang Weigend
ORACLE Deutschland B.V. & Co. KG
München

Schlüsselworte:
Java EE, WebLogic, Application Platform

Zusammenfassung

Java EE 7 ist der letzte Meilenstein der Java EE Plattform, die die effiziente Bereitstellung von kritischen Unternehmensdaten und ihre Nutzung in mobilen und Web-basierten HTML5-Anwendungen erheblich vereinfacht. JAX-RS 2 führt eine standardisierte Client API für den Zugriff auf RESTful-Endpoints. Die JSON-P API unterstützt bei der Konvertierung zwischen Java Objekten und ihrer JSON-Darstellung. WebSocket ermöglicht eine neue Qualität von Real-Time Kommunikation, die über die Möglichkeiten von HTTP weit hinausgeht. Viele der Java EE API bekommen eine asynchrone Alternative, um eine Thread-Blockierung bei der Request-Verarbeitung zu vermeiden und die Skalierbarkeit der Plattform zu erhöhen.

Schon vor mehreren Monaten wurde der Java Specification Request JSR-366 für Java EE 8 einstimmig angenommen. Nach detaillierten Umfragen und mehr als 4500 Antworten aus der Java Community wurden bereits die Hauptinhalte der in Arbeit befindlichen Java EE Spezifikation festgelegt. Zum weiteren Ausbau der Unterstützung von HTML5-Anwendungen gehören der Ausbau der Servlet API zur Unterstützung des nagelneuen HTTP/2 Protokolls, die Standardisierung von Server-Sent Events und JSON-Binding, sowie die Einführung eines neuen Action-basierten MVC-Framework. Die Entwickler-Produktivität wird durch neue Security-Interceptors, eine einfache CDI-basierte Alternative zu den Message Driven Beans und diverse CDI-Verbesserungen gesteigert. Der Betrieb der Java EE Plattform in Cloud-Umgebungen wird durch neue REST-basierte API für Management und Deployment und neue Security API vereinfacht. Es wird untersucht wie neue Architekturen und Programmiermodelle wie Microservices und Reactive Programming unterstützt werden können.

Der Vortrag richtet sich an alle, die die Vorteile der neuen Java EE Technologien in ihren nächsten Enterprise-Anwendungen auf einer kompatiblen, hochverfügbaren und performanten Anwendungsplattform wie WebLogic nutzen möchten. Es werden auch die Themen besprochen, die die Eckpunkte von Java EE 8 bilden werden.

Schwerpunkte von Java EE 7

Das Hauptthema in Java EE 7 ist die Server-seitige Unterstützung von HTML5-Anwendungen, die auf Grund ihrer Mächtigkeit, gute Portabilität und Eignung sowohl für Desktops als auch für mobile Endgeräte zunehmend an Bedeutung gewinnen. Die Erweiterung des Funktionsumfangs, um den stetig steigenden Unternehmensanforderungen zu entsprechen und die weitere Steigerung der Entwickler-Produktivität sind die zwei anderen Schwerpunkte, auf die sich die Java EE 7 Plattform konzentriert:

Abb. 1: Java EE 7 Schwerpunkte

Mehrere Application Server haben bereits Java EE 7 implementiert. Dazu gehören: GlassFish 4.x, WildFly 8.x, TMAX JEUS 8, Hitachi Cosminexus 10.0, IBM WebSphere Liberty 8.5.5.6. Der aktuelle Stand ist unter http://www.oracle.com/technetwork/java/javaee/overview/compatibility-jsp-136984.html zu finden.

Schwerpunkte von Java EE 8

- **Support der letzten Web-Standards zur Vertiefung der Unterstützung von HTML5-Anwendungen**

  In der neuen Servlet-Spezifikation (JSR-369) wird der neue HTTP/2 Standard unterstützt. Die Servlet 4.0 API wird den Zugang zum HTTP/2 Request/Response Multiplexing ermöglichen, indem der Servlet-Request als eine HTTP/2 Nachricht exponiert wird. Eine neue Klasse „Priority“ wird für die Implementierung von Stream-Priorisierung benötigt. Weiterhin wird der HTTP/2 Server-Push und das Protokoll-Upgrade von HTTP 1.1 unterstützt.

  Eine neue Client- und Server-seitige Java API für die Nutzung von Server-Sent Events wird im Rahmen von JAX-RS 2.1 (JSR-370) formuliert. Sie ist sehr ähnlich dem entsprechenden JavaScript Standard.

  Auch im Rahmen von JAX-RS 2.1 wird ein neues Action-basiertes MVC Framework mit dem JSR-371 eingeführt. Zum Unterschied von Komponenten-basierten MVC Frameworks wie JSF und Wicket, muss beim Action-basiertes MVC mehr vom Entwickler manuell gemacht werden (Request-Parameter Verarbeitung, Input Konvertierung und Validierung etc.). Dafür bekommt er aber vollständige Kontrolle über die HTML5-Programmierung (HTML, JavaScript, CSS). In JSR-371 werden vermutlich folgende Java EE Standards zum Einsatz kommen: CDI, Bean Validation und JPA für das Model, Facelets und JSP für das View und JAX-RS für die Implementierung des Controllers.

  JSON-P 1.1 ist ein Update der in Java EE 7 eingeführten JSON-P API, um die neuen JSON-Pointer(IETF RFC 6901) und JSON-Patch(IETF RFC 6902) Standards in Java umzusetzen. Es geht dabei darum, ähnlich wie mit XPATH für XML standadisiert Felder von JSON-Objekten zu referenzieren und JSON-Dokumente analog zu HTTP PATCH für Ressourcen (RFC 5789) zu

modifizieren.

JSON-Binding (JSR-367) wird die Konvertierung zwischen Java-Objekten und ihrer JSON-Darstellung auf einer höheren Ebene als mit JSON-P standardisieren, ähnlich wie JAXB im Vergleich zu JAXP.

- **Vereinfachung der Entwicklung**

Die neue CDI 2.0 Spezifikation (JSR-365) wird in drei Modulen aufgeteilt – CDI full mit Java EE Support, CDI full (auf Java SE) und CDI light (auf Java SE, aber keine Unterstützung für Kontexte und AOP). Container-Dienste, die momentan nur für bestimmte Komponenten wie z.B. EJB nutzbar sind, werden mit den Mitteln von CDI für alle Managed Beans verfügbar gemacht. Es werden neue asynchrone Events eingeführt (bisher gab es nur synchrone). Der Startup-Prozess (Bootstarpping) von CDI auf der Java SE wird standardisiert. CDI 2.0 umfasst noch diverse Verbesserungen und Vereinfachungen für die Nutzung von Interceptoren, Dekoratoren und der Extension SPI. Durch die Einführung von Priority-Annotations wird die Anordnung von Observern ermöglicht. Die Abstimmung mit und die Nutzung von CDI wird generell in vielen anderen Teilspezifikationen überprüft.

Eine neue API zum Empfang von asynchronen Nachrichten wird eine einfache Alternative zu Message-Driven Beans anbieten, so dass sie jede entsprechend annotierte CDI Bean empfangen kann, auch ohne ein MessageListener Interface implementieren zu müssen.

Die EJB 2.0 Client-View API und die Unterstützung für CORBA IIOP-Interoperabilität werden auf ihre Relevanz geprüft und gegebenenfalls als optional erklärt (Pruning).

- **Ausbau der Infrastruktur für den Betrieb in Cloud-Umgebungen**

In den letzten Java EE Versionen wurde eine rudimentäre Unterstützung für die Provisionierung in Cloud-Umgebungen eingeführt – beispielsweise durch die generierung von DB-Schemas in JPA 2.1 und verschiedene Resource-Definitionen.

Das Ziel von Java EE 8 Security 1.0 (JSR-375) ist eine einfache API zu definieren, die Java EE Anwendungen in die Lage versetzt, ihre Sicherheitsaspekte und Konfiguration selbst zu verwalten, um dadurch portabel in verschiedene Cloud- oder On-Premis Umgebungen transportiert zu werden, unabhängig von den proprietären Security-Mechanismen der jeweiligen Application Server. Dazu gehört eine standardisierte Schnittstelle für User- und Role-Management, sowie für Interaktionen mit Security Provider, hinter denen sich oft User- und Role-Repositories, die auf RDBMS-, NoSQL- oder LDAP-Server implementiert sind, verbergen. Die Einführung von Password-Aliasing ermöglicht die Referenzierung von Passwörtern, die in einem sicheren, mit der Anwendung gebündelten Archiv verschlüsselt sind und somit für Angreifer nicht zugänglich sind. Neue CDI Authorization-Interceptoren werden die existierenden Autorisierungsmechanismen (role based access control) erweitern und feingranularer gestalten.

REST-basierte API für Monitoring und Management werden die Entwicklung von modernen Verwaltungstools für Application Server ermöglichen, die für alle, die mit dem Standard kompatible sind, funktionieren werden.

- **Anpassungen zur Nutzung von Java SE 8**

Java EE 8 baut auf Java SE 8. Alle Java EE 8 Teil-Spezifikationen werden überprüft, inwieweit sie die neuen Java SE 8 Features wie Repeating Annotations, Lambda Expressions, Date/Time API,

Type Annotations, Completable Futures, etc. nutzen können.

Java EE 8 Status und Roadmap

Folgende Bilder veranschaulichen den aktuellen Status der Java EE 8 API und Roadmap:

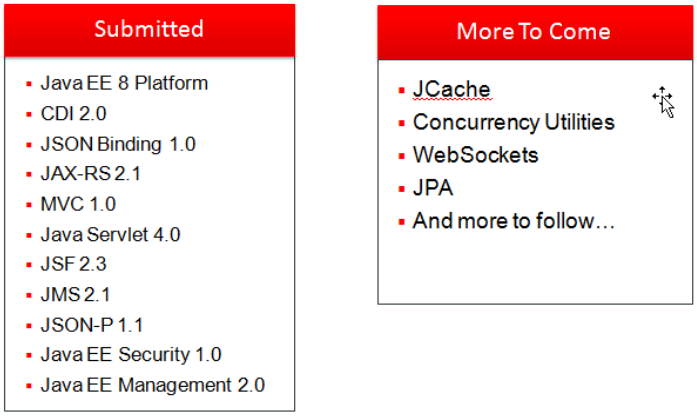

Abb. 2: Java EE 8 Status

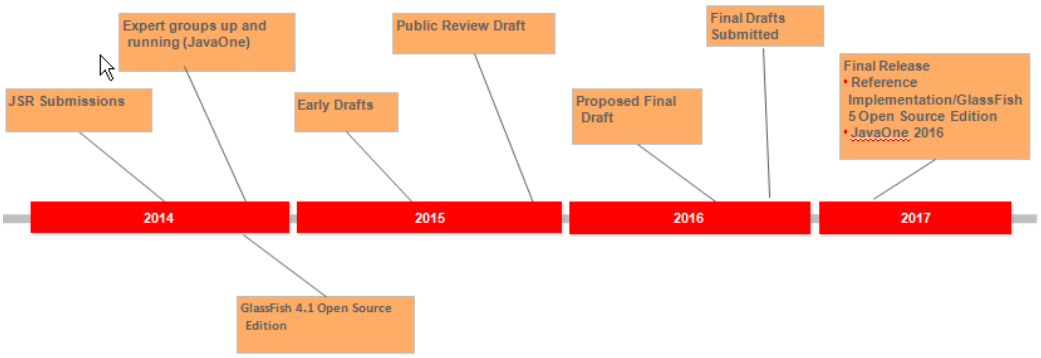

Abb. 3: Java EE 8 Roadmap

Inzwischen ist der Java Standardisierungsprozeß sehr transparent geworden und die Beteiligung der Java Community daran hat zugenommen. Deshalb wird es leichter sein, in den nächsten Monaten mitzubekommen, wohin die Java EE 8 Reise geht, um rechtzeitig Einfluss darauf zu nehmen.

Fazit

Die Java EE Plattform ermöglicht heute die Entwicklung und Bereitstellung von unternehmens-kritischen Anwendungen mit höchster Flexibilität und Effizienz und ermöglicht die standardisierte Nutzung neuester Technologien und Frameworks. Das Hauptziel von Java EE ist es vom Anfang an gewesen, allgemeine Infrastrukturaufgaben über ein Container-Modell auszublenden und den Zugriff auf Ressourcen zu abstrahieren, so dass sich der Entwickler auf die Business-Logik seiner Anwendung konzentrieren kann. Die kontinuierliche Vereinfachung der API für Zugriff auf Container-Services und die Erweiterung des Umfangs dieses Services hat dazu geführt, dass die Entwicklung von Enterprise-Anwendungen sehr einfach geworden ist. Dieser Trend wird mit Java EE 8 fortgesetzt.

Kontaktadresse:
Peter Doschkinow, Wolfgang Weigend
ORACLE Deutschland B.V. & Co. KG
Riesstr. 25
D-80992 München
Telefon: +49 (0) 1802672253
Fax: +49 (0) 1802672329
E-Mail peter.doschkinow@oracle.com, wolfgang.weigend@oracle.com
Internet: www.oracle.com