Java EE – die nächste Generation

Peter Doschkinow, ORACLE Deutschland B.V. & Co. KG

Die Fertigstellung von Java EE 8 und die Ankündigung von Oracle, die Java-EE-Spezifikationen, Test-Compatibility-Kits (TCK) und Referenz-Implementierungen vollständig zu öffnen und an die Eclipse Foundation zu übergeben, wurden in der Java Community als ein Befreiungsschlag empfunden.

Diese Schritte sowie die positive Stimmung auf der JavaOne 2017 haben nach monatelangen Diskussionen und Verunsicherungen über den Zustand, das Management und die Zukunft der Java-EE-Plattform App-Server-Hersteller sowie Entwickler neu mobilisiert und eine neue, erfolgreiche Zukunft für Enterprise Java mit mehr Community-Beteiligung und effizienteren Prozessen in Aussicht gestellt.

Java EE 8

Java EE 8 und die Referenz-Implementierung GlassFish 5 wurden am 21. September 2017 zeitgleich mit der Java SE 9 freigegeben. Im Vergleich zur vorherigen Version Java EE 7 ist der Umfang der neuen Features geringer ausgefallen; einige der vorgesehenen Teilspezifikationen wie MVC, JMS 2.1 und Management API 2.0 wurden ausgelassen. Ein Grund dafür ist, wie Java-EE-8-Spec-Lead Linda DeMitchiel in ihrer JavaOne-Session erklärte, dass führende Mitarbeiter zeitweise mit anderen, höher priorisierten Aufgaben beschäftigt waren. Auch der Umzug der Java-EE-/GlassFish-Infrastruktur von *„java.net"* in die moderne GitHub-Umgebung unter *„https://github.com/javaee"* während der Entwicklung hat viel Zeit gekostet. Er war jedoch notwendig, um die Plattform für interessierte Entwickler zugänglicher zu machen und ihre Zusammenarbeit besser zu unterstützen.

Die Java Community freut sich nun auf die neue Version. Java EE wurde weiter modernisiert und ihre Nutzung in automatisierten CI/CD-Pipelines („continuous integration and delivery") ist einfacher geworden. Neue APIs standardisieren und erleichtern den Umgang mit modernen Web-Technologien. Zu den wichtigsten Neuerungen gehören:

- Java Servlet 4.0 API mit HTTP/2-Support: Die Unterstützung für das HTTP/2-Protokoll mit Request/Response Multiplexing, Server Push und Protokoll-Upgrade von HTTP/1.1 ermöglicht dem Web-Client (unter Beibehaltung der HTTP/1.1-Semantik) Nebenläufigkeit, ohne mehrere TCP-Verbindungen zum Server aufzubauen, und verbessert somit seine Latenz-Zeiten.
- Erweiterter JSON-Support einschließlich eines neuen JSON-Binding-API vereinfacht den Umgang sowie die Modifikation von JSON-Strukturen und kann über Java-Annotations ein anpassbares Mapping zwischen diesen und Java-Objekten definieren.
- Ein neues REST-Reactive-Client-API, motiviert durch Java-Bibliotheken wie RxJava und das Reactive-API von Jersey, die Referenz-Implementierung von JAX-RS sowie ein Reactive-Client-API zeigen ihre Stärken am besten im Use-Case Microservice-Orchestrierungen. Wenn die Ergebnisse eines Aggregations-Service von mehreren Services abhängen, die aus Performance-Gründen asynchron und parallel abgefragt werden müssen, so werden der Code und die Fehlerbehandlung schnell komplex und unübersichtlich. Das Reactive-Client-API ist ein Fluent-API für asynchrone Data-Streams, das das Observer-Muster implementiert und für diesen Fall eine einfachere, besser lesbare und zuverlässigere Implementierung ermöglicht.
- Asynchrone CDI-Events verbessern die Skalierbarkeit und die Zuverlässigkeit bei der Zustellung von Events zwischen Produzenten und Konsumenten, weil sie in verschiedenen Threads stattfindet.
- Ein neues, portierbares Security-API vereinfacht die Konfiguration der Anwendungs-Authentifizierung (BASIC, DIGEST, FORM, oder CERT). Es standardisiert den Umgang mit dem benötigten Repository für User-Credentials, Gruppen-Zugehörigkeit und Autorisierung, egal ob eingebettet oder auf einer Datenbank beziehungsweise LDAP-Server basierend. Das API definiert auch einen Plattform-übergreifenden Security Context, der den Zugriff auf Ressourcen deklarativ und programmatisch einheitlich regeln kann (bisher zersplittert in Servlet, JAX-RS, CDI, JAX-WS, EJB, JSF und WebSocket API).
- Server-Sent Events Support (Client- und Server-seitig) standardisiert ein übliches Kommunikations-Muster, bei dem der Server dem Web-Client mehrere Updates über dieselbe HTTP-Verbindung schickt.
- Unterstützung für die neuen Java-SE-8-Features (wie Data & Time API, Streams API, Annotations).

Wie in der Vergangenheit üblich, steht neben GlassFish 5 auch ein Java-EE-8-Software-Developer-Kit zum Download bereit, in dem zusätzlich Code-Beispiele, die API-Dokumentation sowie ein Tutorial enthalten sind. *Abbildung 1* zeigt die neuen und modifizierten Java EE 8 Specs.

Eclipse Enterprise for Java (EE4J) – die neue Heimat von Java EE

Im August 2017 hat Oracle damit begonnen, verschiedene Alternativen zu untersuchen, um Java-EE-Technologien an eine Open-Source-Organisation zu übergeben. Ziel war, die Weiterführung der Java-EE-Standards agiler, flexibler und offener zu gestalten. Unter Einbeziehung der Sichten von IBM, Red Hat und der Community hat sich Oracle schließlich für die Eclipse Foundation entschieden, die sich als effizientes Zuhause für viele Open-Source-Projekte wie die IBM-J9-Java-Implementierung und EclipseLink erwiesen hat. Die Java-EE-Plattform, bestehend aus Spezifikationen, Referenz-Implementierungen, TCK und Dokumentationen, wird nun unter dem Dach des neuen Eclipse-Top-Level-Projekts EE4J weiterentwickelt. Im ersten Schritt sollen der Java-EE-8-Kern, GlassFish 5 und die von Oracle angeführten Java-EE-8-Technologien, inklusive Code und Build-Scripts, dorthin gebracht werden.

Der Nachfolger von Java EE 8 bekommt demnächst einen neuen Namen (Java EE ist ein Markenname von Oracle). Die Lizenzierung wird flexibilisiert und erfolgt zunächst nach Eclipse-Public-Lizenz (EPL) 2.0 mit der sekundären Lizenz GPL 2 mit Classpath Exception. Das bedeutet, dass EE4-Ergebnisse GPL-2-kompatibel sind und, falls mit GP-2-Artefakten kombiniert, zusammen unter der GPL-2-Lizenz bereitgestellt werden können.

Das EE4J-Open-Source-Projekt hat sich folgende Ziele gesetzt:

- Erstellung von Standard-APIs, Referenz-Implementierungen und TCK für Java-Runtimes für Entwicklung und Betrieb von Server-Side- und Cloud-Native-Anwendungen auf der Grundlage von Java EE 8
- Definition eines agilen und offenen Prozesses, flexible Lizenzierung und offenere Verwaltung für die Evolution der Plattform
- Kompatibilität zwischen Java EE 8 und EE4J-Folgeversionen
- Förderung der gemeinsamen Nutzung und leichte Integration der einzelnen Technologien sowie ihre Stand-alone-Nutzung auf Java SE, um leichtgewichtige und Microservices-Architekturen besser zu unterstützen

EE4J ist offen für die Integration von Innovationen, die in anderen Open-Source-Projekten stattfinden. Ein anderes spannendes und Java-EE-verwandtes Eclipse-Projekt ist zum Beispiel MicroProfile, das sich mit der Optimierung von Java-Enterprise für Microservice-Architekturen beschäftigt und sich als Innovations-Lab für Java-EE-Technologien versteht. EE4J wird untersuchen, wie eine effiziente Zusammenarbeit und Integration von MicroProfile-Ergebnissen gestaltet werden kann.

Oracle wird nach wie vor Support für bestehende Java-EE-Lizenznehmer und solche, die nach Java EE 8 migrieren, anbieten. Oracle beabsichtigt, auch existierende WebLogic-Versionen und Java EE 8 in einer künftigen WebLogic-Version zu unterstützen. Somit möchte Oracle einen fließenden Übergang von existierenden Java-EE-Standards in eine offenere Umgebung, in der sie erfolgreicher weitergeführt werden können.

Fazit

Java EE bildet seit vielen Jahren erfolgreich das Fundament für viele unternehmenskritische Anwendungen, weil es komplexe und wichtige Dienste für Entwickler über standardisierte und leicht zu nutzende Schnittstellen zur Verfügung stellt. Java EE 8 ist der letzte Meilenstein in Richtung Vereinfachung, Modernisierung und Anpassung der Plattform an aktuelle Web-Technologien und Trends. Die Übergabe von Java EE an die Eclipse Foundation hat Begeisterung und breite Zustimmung

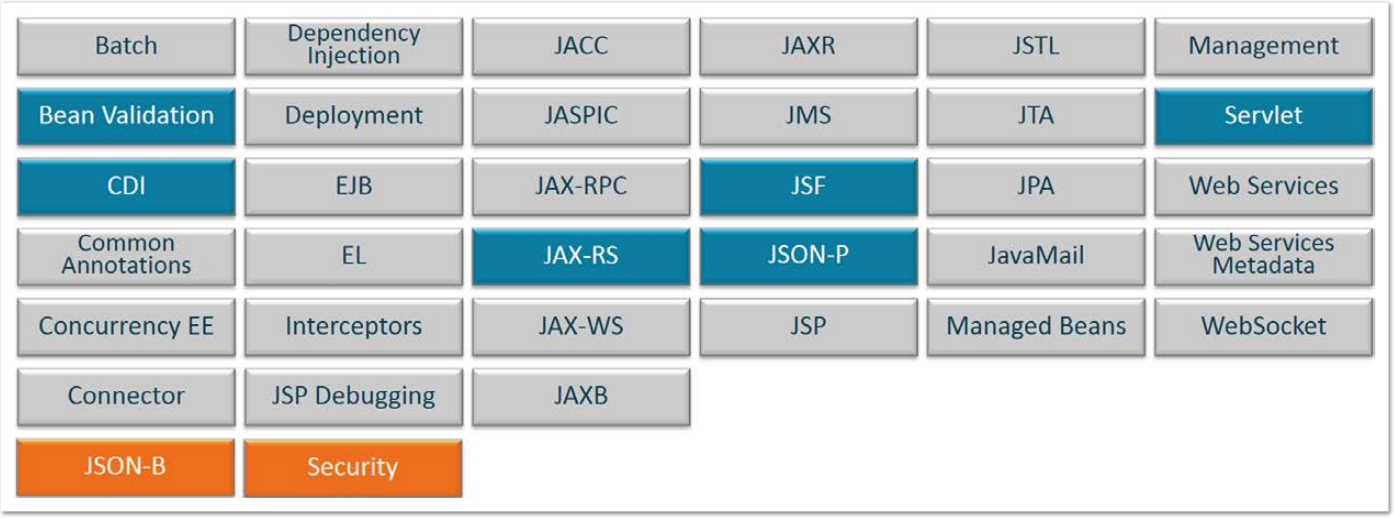

Abbildung 1: Die neuen Java EE 8 Specs in orange, die stark modifizierten in blau

in der Java-EE-Community ausgelöst sowie die berechtigte Hoffnung, dass effizientere Prozesse, Management und stärkeres Community-Engagement im Rahmen von EE4J-Enterprise Java schnell nach vorne bringen. Wie James Governor auf der JavaOne angemerkt hat, „haben sich in der Java-Welt die Dinge in den letzten drei Wochen wahrscheinlich mehr geändert als in den vergangenen dreizehn Jahren".

Links

Java EE: *http://www.oracle.com/technetwork/java/javaee/overview*
GlassFish: https://github.com/javaee/glassfish
Java EE Freigabe: *https://blogs.oracle.com/theaquarium/opening-up-ee-update*
EE4J: *https://projects.eclipse.org/projects/ee4j/charter*
The Future of Java EE (Mark Little, Red Hat): *http://middlewareblog.redhat.com/2017/09/11/the-future-of-java-ee*

Peter Doschkinow
peter.doschkinow@oracle.com

Peter Doschkinow arbeitet als Senior Java Architekt bei Oracle Deutschland. Er beschäftigt sich mit serverseitigen Java-Technologien und Frameworks, Web Services und Business Integration, die er in verschiedenen Kundenprojekten erfolgreich eingesetzt hat. Vor seiner Tätigkeit bei Oracle hat er wertvolle Erfahrungen als Java Architect and Consultant bei Sun Microsystems gesammelt.

Java EE 8 und danach – mit Cloud und Community

Werner Keil, JCP Executive-Committee-Mitglied, Java EE 8 EG Mitglied

Nachdem die Fertigstellung von Java EE 8 bereits maßgeblich durch Mitglieder der Java Community erfolgte, wirkt die geplante Übergabe an die Eclipse Foundation als Open-Source-Projekt wie ein logischer nächster Schritt. Insbesondere, da Oracle mit dem Fn Project seine eigenen Visionen und Angebote für die Cloud in einem „Serverless"-Umfeld eben erst auf der JavaOne 2017 und Oracle Code präsentiert hatte.

Man wendet sich damit bei Oracle doch etwas von Java ab, da zwar eine gute Unterstützung für Java-„Functions" in der Fn Cloud versprochen wurde, deren Infrastruktur aber komplett auf Google Go, Docker oder Amazon AWS und Lambda basiert. Dies erlaubt eine Vielzahl von Sprachen (von JavaScript über Python und Go bis Java), wodurch Java nur eine von vielen ist. Es bleibt abzuwarten, wie sehr sich Java hier gegen die Konkurrenz bewährt.