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.

## Java EE 8

Am 21. September 2017 wurden Java EE 8 *[1]* und die Referenz-Implementation GlassFish 5 *[2]* veröffentlicht. Anders als bei den JSRs, die in Java EE 8 einflossen, sind die Committer des GlassFish-Projekts fast ausschließlich Oracle-Mitarbeiter. Nach dem Umzug von *„java.net"* in die GitHub-Organisation *„https://github.com/javaee"* ist die Transparenz ein großer Vorteil von GitHub. Hier sind alle, die in ein Java-EE-Repository gepusht haben, klar ersichtlich. Dies kommt nicht zuletzt auch dem Transparency-Auftrag von „JCP.next" entgegen. Nachdem die wichtigsten Neuerungen in Java EE 8 von Oracle bereits hinlänglich beleuchtet wurden, möchte ich den Blick auf den Anteil der Community in den jeweiligen JSRs richten, die darin neu oder aktualisiert eingeflossen sind:

**Java Servlet 4.0 API mit HTTP/2-Support:** Dies wurde fast ausschließlich von Oracle als Spec Lead getragen, wenngleich die Mitglieder der Expertengruppe und die erweiterte Community durchaus gefragt wurden.

**Java Server Faces:** Dieser JSR unter Leitung von Ed Burns (ebenso wie Servlet 4.0) gehört traditionell zu den offensten und transparentesten. Selbst wenn Details der Implementierung oder das TCK bisher nicht offen entwickelt werden, ist hier ein starker Anteil der Java-Community und verschiedener JSF-Implementationen eingeflossen.

**JSON-Binding 1.0 API:** Wurde zwar zu erheblichen Teilen durch Spec Lead Dmitry Kornilov und seine Kollegen in Prag getragen, aber auch in dieser Expertengruppe sind Open-Source-Projekte und Einzelpersonen vertreten, die ihre Wünsche und Ideen einbrachten. So wurden auch das Logo und die Website durch Mitglieder der Community gestaltet. Da die Anfänge bei EclipseLink (JAX-B) lagen, die RI später allerdings in ein unabhängiges Eclipse-Projekt „Yasson" übergeführt wurde, kann man JSON-B als Paradebeispiel und Vorreiter dessen betrachten, was die gesamte Plattform unter EE4J noch vor sich hat. Dmitry wurde dafür auch zu Recht mit dem JCP-Award 2016 als bester Spec Lead ausgezeichnet.

**JSON-P 1.1:** Diese Verbesserung und Anpassung des JSON-JSR an neue JSON-Standards wurde von der Community, also einigen engagierten Mitgliedern der Expertengruppe, gerettet, als Oracle zwischen 2015 und 2016 begann, die Prioritäten zugunsten seiner Cloud-Produkte oder Projekte wie Fn zu verschieben. Dadurch waren viele Spec Leads gezwungen, die Arbeit an JSRs praktisch ganz einzustellen, und andere dazu, das Unternehmen früher oder später zu verlassen. Der ursprüngliche Spec Lead von JSR-374 war so ein Fall: Ich war als einziges EG-Mitglied bereits im Vorläufer JSON-P 1.0 auch in der Expertengruppe und damit bei allen Änderungen und Verbesserungen besonders an einer Kontinuität und Rückwärtskompatibilität interessiert. Gemeinsam mit anderen Mitgliedern der EG und Vertretern der Community, die als Contributor gewürdigt wurden, gelang es, diesen Spagat aus Erneuerung und Kontinuität relativ gut zu meistern. Weil JSON-P 1.1 auch als Grundlage für den neuen Standard JSON-B 1.0 dient, erklärte sich Dmitry Kornilov um die JavaOne 2016 bereit, diesen auch zu übernehmen. Beide gehören zu guten Beispielen dafür, wie der Spec-Lead-Vertreter bei Oracle sich der Community öffnete und sie einbezog, nicht zuletzt auch, da er neben seinen Aufgaben in der Cloud gleich 2 JSRs zu leiten und fertigzustellen hatte; wie übrigens auch Ed Burns – ein weiterer der „Super Spec Leads" bei Oracle, die wohl zu den ganz wenigen Vertretern im EE4J-Projekt gehören und dort die Zukunft von Enterprise Java weiter mitbestimmen dürfen. Dmitry ist wohl auch durch seine Rolle als Eclipse-Projektleiter für Yasson inzwischen dem EE4J PMC beigetreten.

**JAX-RS 2.1:** Dieser JSR wurde relativ stark von Oracle getragen und umgesetzt. Bereits im Vorfeld war ein Red-Hat-Vertreter aus der Expertengruppe ausgestiegen, offenbar nach Konflikten oder Meinungsverschiedenheiten mit dem Spec Lead, die anders nicht beilegbar schienen. Einige andere EG-Mitglieder repräsentieren aber sehr wohl die Java Community. Ein paar davon sind auch in den von Oracle aus Java EE 8 abgegebenen MVC-Standard involviert, da dieser ja JAX-RS massiv als Basis nutzt. Es bleibt abzuwarten, wie die Zukunft von JAX-RS im Rahmen eines neuen EE4J-Umfelds aussehen wird. Andererseits wurden viele Neuerungen wie REST-Reactive-Client-API bereits umgesetzt; also könnte sich der Änderungsbedarf hier auch vorerst in Grenzen halten.

**CDI 2.0:** Hier standen asynchrone Events im Vordergrund, ebenso die bessere Unterstützung von CDI in einem Java-SE-Umfeld, Stichwort „Serverless" oder „Microservices". Obwohl mit wenigen Ausnahmen fast alle Commits auch hier durch Vertreter der Spec-Lead-Firma Red Hat erfolgten, ist die Expertengruppe von einer starken Diskussionskultur getragen, bei der auch selbstständige, unabhängige EG-Mitglieder wie ich und andere ihre Meinung einbringen konnten und praktisch alle Entscheidungen sehr demokratisch nach dem Mehrheitsprinzip erfolgten, statt vom Spec Lead auf eigene Faust durchgeboxt zu werden. Auch hier wurde Antoine Sabot-Durand dafür sowohl mit einer Star-Spec-Lead-Auszeichnung gewürdigt, als auch Gewinner eines JCP-Awards 2017 in der Spec-Lead-Kategorie. CDI war für JSRs – beinahe exotisch – immer schon einer der offensten Standards. Mit API, RI, TCK und sogar der Spezifikation unter einer (Apache-2.0)-Open-Source-Lizenz. Beim Spec-Dokument bilden CDI und Bean Validation die seltene Ausnahme. Alle anderen JSRs halten sich hier ausschließlich an die Oracle-Spec-Lizenz, während Red Hat deren Gültigkeit nur auf von „jcp.org" heruntergeladene PDF-Dokumente beschränkt. Dass Red Hat zu den treibenden Kräften sowohl hinter Eclipse MicroProfile als auch hinter EE4J gehört, verwundert daher wenig. Wie JSRs inklusive CDI 2.1 oder 3.0 in Zukunft aussehen sollen und ob die Package-Struktur diese dann noch rückwärtskompatibel macht, ist dagegen noch Inhalt langwieriger Verhandlungen, die jenen über die „Jamaika"-Koalition in Deutschland nach der Bundestagswahl 2017 in nichts nachstehen.

**Bean Validation 2.0:** Die Expertengruppe, wie CDI von Red Hat geleitet, umfasst zahlreiche Mitglieder, viele von ihnen Einzelpersonen. Bean Validation 2.0 bezog nicht zuletzt Contributions aus der Community ein, etwa die JSR-354-Unterstützung (JavaMoney) *[3]*, die Zalando zuvor auf GitHub entwickelt hatte. Hier wurden also gerade bei der Referenz-Implementation Hibernate Validator viele Einflüsse aus der Gemeinschaft genutzt, während API oder TCK – offen und transparent – dennoch primär von Red Hat entwickelt wurden.

**Java EE Security-API 1.0:** Dieser neue Standard wurde noch stärker als JSON-B 1.0 massiv durch die Community getragen und gemeinsam aufgebaut. Es fing mit einer sehr demokratischen Abstimmung und Bewertung der User Stories und Requirements an, beinahe wie ein verteilter Agile-Planning-Poker. Auch hier musste der Oracle

Spec Lead leider sehr bald anderen Aufgaben in der Cloud gehorchen, und die EG war weitgehend auf sich allein gestellt. Dank der besonders aufopfernden Leistung von Arjan Tijms und einer Handvoll anderer (zumeist selbstständiger beziehungsweise unabhängiger) EG-Mitglieder wurde dieser JSR dennoch ein Erfolg und rechtzeitig für Java EE 8 fertiggestellt.

**Unterstützung für neue Java-SE-8-Features:** Diese wurden großteils als Maintenance Release (MR) und nicht als eigene JSRs für Java-EE-8-Komponenten wie JPA umgesetzt; naturgemäß durch den Spec Lead, da formell eine Expertengruppe für ein MR gar nicht mehr existiert und allein der Maintenance Lead dafür verantwortlich ist. Dennoch kann ich speziell aus jenen Expertengruppen wie JPA, in denen ich schon seit Java EE 6 oder 7 dabei bin, sagen, dass hier die Mitglieder über Entscheidungen befragt wurden und es eine teils auch recht rege Diskussion darüber gab.

## Java EE Guardians

Der erwähnte Arjan Tijms gehört wie fast alle Einzelpersonen beziehungsweise Mitarbeiter von Software-Herstellern, die halfen, JSR-375 und viele andere Java EE 8 JSRs erfolgreich fertigzustellen, zu den „Java EE Guardians" *[4]*, einer Grassroot-Bewegung, die der frühere Java-EE-Evangelist bei Oracle, Reza Rahman, ins Leben rief, kurz nachdem er selbst Oracle verließ, als Antwort auf die Kürzungen bei der Standardisierung zugunsten meist kommerzieller Cloud-Angebote. Die Liste wird von so illustren Persönlichkeiten angeführt wie Java-Begründer James Gosling, der übrigens inzwischen auch mit Amazon AWS bei einem führenden Cloud-Anbieter arbeitet, also einerseits mit Oracle konkurriert, andererseits aber (siehe Fn Project) die Basis eigener Cloud-Angebote darstellt.

Die Java EE Guardians wurden anfänglich belächelt (auch ob des an Marvel angelehnten Namens), erwiesen sich jedoch durchaus wie eben jene sehr unterschiedlichen und manchmal chaotisch anmutenden Marvel-Helden als Retter, wenn nicht der Galaxis, so doch zumindest von Java EE 8. Ohne Vertreter der Java EE Guardians wären etliche vor allem der neueren JSRs in den Anfängen steckengeblieben oder hätten wie JCache (JSR-107) inzwischen gleich zum zweiten Mal den Anschluss an den Java-EE-8-Release-Train schlicht verpasst.

Die für Java SE und speziell OpenJDK sehr erfolgreiche Adopt-a-JSR-Bewegung fand davor an Java EE leider nie wirklich Gefallen. Erst die Java EE Guardians brachten selbst jene, die vielleicht nicht auf dieser Liste zu finden sind, zum Umdenken und zur Unterstützung von Java EE oder damit verwandten Projekten wie MicroProfile beziehungsweise EE4J. Heute findet man auch die „üblichen Verdächtigen" unter großen Java-User-Groups wie LJC oder SouJava, die nun Adopt-ähnliche Aktionen auch für Java EE anbieten oder MicroProfile und in naher Zukunft EE4J unterstützen. Ohne die Java EE Guardians als Zündfunke und Fackelträger dieser Bewegung wäre das vielleicht nie geschehen oder hätte deutlich länger gedauert.

## Eclipse Enterprise for Java [5]

Auf Druck durch verschiedene Seiten, etwa Spring und seine „Mix & Match"-Politik bei der Nutzung von Standards oder auch die Formierung von Alternativen wie Eclipse MicroProfile *[6]*, sowie nicht zuletzt durch die eigene Ausrichtung auf die Cloud und Lösungen wie das Fn Project *[7]* sah sich Oracle praktisch gezwungen, das Zepter über die Java-Enterprise-Plattform um die JavaOne 2017 abzugeben. Eclipse erwies sich durch eine konsequente Release-Politik mit jährlichen Zyklen sowie etlichen JSRs von JPA bis JSON-B als die beste Alternative. Auch einige von Oracle nicht mehr umsetzbare Standards zur Unterstützung von Microservices wie Konfiguration, Circuit Breaker, Health Check etc., die seit Anfang 2017 bei Eclipse MicroProfile eine Art Rapid Incubator fanden, waren ein weiterer Grund sich für Eclipse zu entscheiden.

Die Eclipse Foundation ist bekanntermaßen sehr transparent und alles geschieht sehr offen – ohne dass eine einzelne Firma oder ein Mitglied die alleinige Kontrolle hätte, was an Oracle und davor an Sun im JCP gerne kritisiert wird. Da sich die Eclipse Foundation anders als Oracle praktisch nur auf Mitgliedsbeiträge oder Spenden ihrer Mitglieder stützt und auch in einigen Bereichen wie bei Juristen, Lizenzexperten etc. nicht eine Armee von Anwälten besitzt – wie Oracle und andere Großkonzerne –, ist allerdings etwas Vorsicht bei den Erwartungen angesagt, insbesondere was den Übergang zu Eclipse anbelangt.

Oracle kündigte auf der JavaOne 2016 an, NetBeans der Apache Foundation überantworten zu wollen. Eclipse hätte dafür bekanntlich keinen echten Sinn ergeben. Erst vor wenigen Wochen, am 10. September 2017, begann laut GitHub Mirror das Repository bei Apache zu existieren. Es hatte also rund ein Jahr für NetBeans gedauert. Die Code-Basis aller Java-EE-JSRs, die nicht bereits bei Eclipse liegen (JSON-B, JPA) dürfte um einiges größer sein. Neben EE4J haben die Eclipse-Lizenz- und -Patent-Experten ja auch noch den üblichen Annual Release Train (Oxygen+1) zu bewältigen sowie mehrere Sub-Communities wie IoT, LocationTech, PolarSys oder ambitionierte und Release-freudige Projekte wie MicroProfile. Als Spec Lead von JSR 363 habe ich selbst erlebt, dass dem Wunsch der Deutschen Telekom und des SmartHome-Projekts, unser JSR nutzen zu dürfen, erst vier bis sechs Monate nach dem Antrag durch SmartHome nachgekommen wurde. Es hieß damals, die Telekom sei zwar ein Solution-Member, aber kein so großer Beitragszahler wie vermutlich Red Hat oder IBM. Daher dauerte etwas länger, selbst mit Schwergewichten, die höhere Beiträge leisten. Hinter EE4J könnte es also viele Monate dauern, bis die einzelnen JSRs dem Umbrella folgen, der ja nicht viel mehr als ein „Feature" im Eclipse-Sprachgebrauch ist oder ein Parent POM.

Das Rotationsprinzip im EE4J PMC, das den Vorsitz alle drei Monate wechselt, ist eine gute Sache. Es ähnelt etwa dem Amt des Schweizer Bundespräsidenten *[8]*, das einmal im Jahr neu besetzt wird, und bei dem nur selten dieselbe Partei zweimal den Präsidenten stellt. Außerdem ist der Vizepräsident immer von einer anderen Partei. Den unabhängigen Vertreter Ivar Grimstad als ersten Vorsitzenden zu ernennen, zeugt auch vom Wunsch, der Community eine Stimme zu geben – auch wenn er nicht in der Java-EE-8-Plattform-EG dabei war und dort, anders als im initialen EE4J PMC, deutlich mehr kleinere Unternehmen oder Einzelpersonen vertreten sind. Die Einladung an Firmen wie Ivars Arbeitgeber (eine große Beraterfirma aus Skandinavien), doch der Eclipse Foundation beizutreten, statt ihn jetzt als „Individual" kostenlos vorzuschicken, wurde auch recht offen und unverhohlen kommuniziert, was angesichts der Spendenfinanzierten Eclipse Foundation auch nicht unverständlich ist. Daher ist es jedoch möglich, dass im Gegensatz zum zuletzt auch für Firmen kostenlosen JCP dort Vertreter größerer Unternehmen, Beratungsfirmen oder IT-Anbieter selbst als Individual Eclipse Committer bevorzugt werden, weil man sich dort weitaus höhere Beiträge erhofft, wenn das Unternehmen sich engagiert und selbst beitritt.

## Fazit

Es ist viel geschehen im Bereich „Enterprise Java". Auch wenn der 20. Geburtstag der Java-EE-Plattform (die im Jahr 1998 erstmals präsentiert wurde) gleichzeitig ihr Todestag sein mag, so ist das meiste positiv zu sehen. Allerdings wird es wegen der vornehmlich Beitrags-finanzierten Eclipse Foundation und ihrer teils begrenzten Ressourcen in den Bereichen „Legal", „IP" oder „Patentwesen" trotz des Überschwangs auf der JavaOne 2017 über die „letzten drei Wochen vs. 13 Jahre" sicher gut 13 Monate dauern, bis die ersten Java-EE-JSRs bei EE4J landen werden. Im Extremfall könnten es sogar zwei bis drei Jahre werden, doch muss der Ehrlichkeit halber hinzugefügt werden, dies kommt von jemandem, der schon viel Vergleichbares in eigenen Projekten erlebt hat, ebenso in der Java-EE-Umbrella-Plattform seit Version 6 und in unzähligen JSRs.

## Links


[1] *https://javaee.github.io*
[2] *https://github.com/javaee/glassfish*
[3] *http://javamoney.org*
[4] *https://javaee-guardians.io*
[5] *https://projects.eclipse.org/projects/ee4j/charter*
[6] *https://microprofile.io*
[7] *http://fnproject.io*
[8] *https://de.wikipedia.org/wiki/Liste_der_Schweizer_Bundespr%C3%A4sidenten*



**Werner Keil**
werner@catmedia.us


Werner Keil ist zurzeit als externer Java-EE-Consultant, Microservice-Architekt und Test-Automation-Experte im Automotive/Embedded-Bereich tätig sowie als Agile Coach, Java-Embedded- und Eclipse-RCP-Experte bei einem Anbieter von Embedded und Realtime-Systemen. Er hilft Global-500-Unternehmen aus Branchen wie „Mobil/Telekom", „Web 2.0", „Finanzen", „Tourismus/Logistik", „Autobau", „Gesundheit", „Umwelt & Öffentliche Hand" sowie IT-Anbietern, darunter Oracle oder IBM. Für einen Medien-Giganten entwarf er Mikro-Formate für die Suchmaschine eines Online-Musik-Shops und für eine Privat-Universität ein Portal, das heute gemeinhin als soziales Netzwerk gilt. Werner Keil entwickelt Enterprise-Systeme mithilfe von Java, Java EE, Oracle, IBM oder Microsoft und betreibt Web-Entwicklung mit Adobe, Ajax/JavaScript sowie dynamischen oder funktionalen Sprachen.



```
public abstract class NewJob {
   private final static int JUNIOR = 1;
   private Set<Expertise> wantedExpertise;
   public NewJob() {
      wantedExpertise = new HashSet(Arrays.asList(Expertise.values()));
   }
   public void apply(short jobLevel, Expertise javaExpertise) {
      if (jobLevel >= JUNIOR && wantedExpertise.contains(javaExpertise)) {
         applyToZertificon();
         workAtZertificon();
      } else {
         tellYourFriends();
      }
   }
   public void applyToZertificon() {
      Browser.open("https://www.zertificon.com/jobs");
      MailClient.sendMail("Bewerbung als Java-Entwickler", "jobs@zertificon.com");
   }
   private Benefits workAtZertificon() {
      final String city = "Berlin";
      final Boolean permanentPosition = true;
      final String[] products = {"E-Mail-Verschlüsselung", "Dateitransfer", "Zertifikatsmanagement"};
      return new Benefits("attraktive Konditionen", "flexible Arbeitszeiten", "nettes Team");
   }
   public enum Expertise {
      JAVA_SE, JAVA_EE, Spring
   }
   public abstract void tellYourFriends();
}
```