Java EE: eine Zwischenbilanz

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

Schlüsselworte:
Java EE, Java EE 7, Application Platform

Zusammenfassung

Ist die Java EE Plattform noch relevant? Java EE 7 ist ihr letzter Meilenstein, der die Bereitstellung von kritischen Unternehmensdaten für ihre Nutzung in mobilen und Web-basierten HTML5-Anwendungen wie noch nie vereinfacht. JAX-RS 2 führt
eine standardisierte Client API für den Zugriff auf RESTful-Endpoints. Die JSON-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 hinausgeht. JMS 2 wurde radikal überarbeitet und die Komplexität in ihrer Nutzung eliminiert. Die Aufnahme der lang ersehnten Batch-API im Zusammenspiel mit den Concurrency-Utilities für Java EE ermöglicht den Betrieb einer ganzen Klasse von Anwendungen, der bislang hauptsächlich auf Mainframes eingeschränkt war. Diese und viele andere Verbesserungen in der Java EE Plattform wurden bereits in GlassFish 4.0 implementiert. Viele Beispiele, die die einzelnen Technologien beleuchten, werden als Bestandteil vom Java EE 7 SDK ausgeliefert. Der Vortrag richtet sich an alle, die die Vorteile dieser neuen und spannenden Standard-Technologien in ihren nächsten Enterprise-Anwendungen nutzen möchten. Es werden auch die Themen besprochen, die die Eckpunkte von Java EE 8 bilden werden.

Einleitung

Java EE ist eine standard Anwendungsplattform, die auf die Entwicklung, Deployment und Management von hoch-skalierbaren, multi-tier und server-zentrischen Web-Aplikationen fokussiert ist. Am Anfang ihrer Entwicklung lag der Schwerpunkt auf die Adressierung der Hauptanforderungen von Enterprise-Anwendungen wie Robustheit, Skalierbarkeit und Anbindung von Unternehmens-Ressourcen über Web Services und andere Standard-Protokolle. Java EE 5 war ein Wendepunkt, bei dem die Verbesserung der Entwicklungsproduktivität zum ersten mal in Vordergrund gerückt ist. Dies wurde in erster Linie durch die Einführung von Annotationen, die Vereinfachung der EJB Spezifikation und das neue und leichter zu benutzende Persistenz-Modell von JPA erreicht. Die Vereinfachung des Java EE Programmiermodells wurde mit Java EE 6 fortgesetzt und maßgeblich durch die Einführung der APIs für und Context- und Dependency Injection und Java-basierte RESTful Web Services, sowie durch eine bessere Integration von Open Source Web-Frameworks, geprägt. Ein neues Thema in Java EE 6 war die Flexibilisierung der Plattform. Die Einführung vom Web Profile, ein Subset der Gesamtspezifikation, hat die Erstellung und Deployment von leicht-gewichtigen Web-Anwendungen ermöglicht und die Java EE Einstiegshürde für kleinere Application Server Hersteller gesenkt. Die Java EE 6 ist die bisher erfolgreichste Java EE Ausgabe, als Plattform der ersten Wahl für Enterprise-Anwendungen anerkannt und wurde in kürzester Zeit von 19 konformen Application Server implementiert. Der letzte Meilenstein von Java EE ist Java EE 7, freigegeben am 12 Juni dieses Jahres. Java EE 7 führt neue API zur Unterstützung von HTML5-basierten Anwendungen ein, adressiert besser neue und alte Unternehmensanforderungen und erhöht weiter die Produktivität in der Anwendungsentwicklung. Noch nie war die Beteiligung der Java Community bei einer Java EE Spezifikation so hoch. 19 Java User Groups in der ganzen Welt haben im Rahmen des Adopt-a-JSR Programms einen Beitrag bei der Durchsicht, beim Testen und der Erstellung von Beispielen für die

einzelnen Teilspezifikationen geleistet. Das folgende Bild veranschaulicht die Evolution der Java EE Plattform:

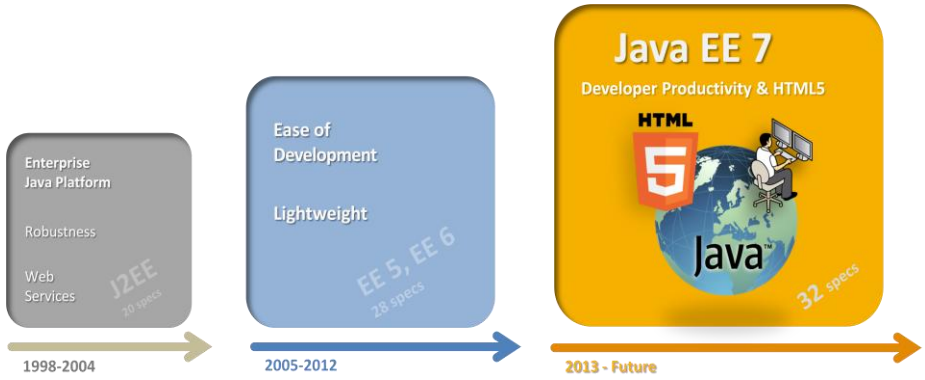

Abb. 1: Die Entwicklung der Java EE Plattform

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. 2: Java EE 7 Schwerpunkte

- **HTML5 Unterstützung**

Eine typische HTML5-Anwendung setzt sich aus HTML5-, JavaScript-, CSS3-Code und Server-Ressourcen zusammen. Die einfache und effiziente Bereitstellung von Server-Ressourcen ist genau die Stärke der neuen bzw. aktualisierten WebSocket, JSON, JAX-RS und Servlet API.

Die WebSocket API ermöglicht die Nutzung von WebSockets, die ein Teil der HTML5-Spezifikationen sind und die Erstellung von einer neuen, dynamischeren Klasse von Web-Anwendungen mit Server-Push Fähigkeiten vereinfacht. Sie ist ein neuer Grundbaustein in Java EE, ähnlich wie die Servlet-API, nur statt auf das HTTP-Protokoll auf die bidirektionale Kommunikation des WebSocket-Protokolls ausgerichtet. Mit WebSocket-Annotations lässt sich ein POJO leicht in

einen WebSocket-Endpoint umwandeln:

```
@ServerEndpoint("/chat")
public class ChatBean {
   static Set<Session> peers = Collections.synchronizedSet(…);
   @OnOpen
   public void onOpen(Session peer) {
      peers.add(peer);
   }
      . . .
   @OnMessage
   public void message(String message) {
      for (Session peer : peers) {
         peer.getBasicRemote().sendText(message);
      }
   }
}
```

Durch die neue JSON-API werden Abhängigkeiten von Third-Party JSON-Bibliotheken in Java EE Anwendungen überflüssig. Mit ihr lassen sich Java Objekte auf ihre JSON-Darstellung abbilden, was für ihre Serialisierung/De-Serialisierung bei der Client-Server Kommunikation notwendig ist, besonders wenn auf der Client-Seite JavaScript verwendet wird. Die JSON-API besteht aus einer Low-Level Streaming API, ähnlich zur StAX-API in der XML-Welt, und einer einfacher zu verwendenden Object-Model API, die der bekannten DOM-API für XML-Dokumente entspricht. Die JSON-API wird sehr häufig in Web-Anwendungen genutzt, die ihre Services über WebSockets oder JAX-RS exponieren.
JAX-RS ist ein Annotation-basiertes Framework, das im wesentlichen HTTP-Requests auf Java Methoden-Aufrufe abbildet und als eine Art DSL(Domain Specific Language) für den Umgang mit dem HTTP-Protokoll angesehen werden kann. Mit JAX-RS ist es sehr einfach durch die Verwendung von wenigen Annotations in Java implementierte Anwendungsdienste als REST Web Services zu exponieren. In JAX-RS 2.0, die Bestandteil von Java EE 7 ist, wird die Client-API standardisiert, so dass Java-Client und Java EE Anwendungen ohne Third-Party Abhängigkeiten auf REST-basierte Backend-Dienste zugreifen können:

```
Client client = ClientFactory.newClient();
String balance = client.target("http://.../atm/{cardId}/balance")
                       .pathParam("cardId", "123456")
                       .queryParam("pin", "3711")
                       .request("text/plain")
                       .get(String.class);
```

Zu den neuen Features von JAX-RS 2.0 gehören noch die Unterstützung von asynchroner Verarbeitung, der verbesserte Umgang mit Hypermedia, ein Filter- und Interceptor Framework und Validierung der Request/Response Daten.

- **Unternehmensanforderungen**

Eine der neuen und lang ersehnten Features von Java EE 7 ist die Unterstützung von Batch-Anwendungen. Im Rahmen der neuen Java API für Batch-Anwendungen(JSR-352) wird die Entwicklung und Betrieb von klassischen Batch-Aplikationen auf der Java Plattform vereinfacht und standardisiert. Batch-Anwendungen haben gemeinsame Anforderungen, wie z.B. Unterstützung von Logging, Checkpoints und Parallelisierung. JSR-352 definiert ein Job Specification Language als

Domänen-spezifische Sprache, ein Java-Programmiermodell und eine Laufzeitumgebung für Batch-Anwendungen für Java SE und Java EE, die solchen Anforderungen gerecht werden.
Die Java EE hat schon lange Unterstützung für die Ausführung von asynchronen Hintergrund-Aufgaben z.B. durch asynchrone EJB-, JAX-RS- oder Servlet-Aufrufe angeboten. Mit der neuen Concurrency Utilities für Java EE API gibt es zum ersten mal die Möglichkeit, auf Anwendungsebene die Nebenläufigkeit zu steuern, ohne die Stabilität des Containers zu beeinträchtigen. Sowohl geläufige, als auch fortgeschrittene Concurrency-Muster werden unterstützt. Die Concurrency Utilities für Java EE API erweitert im wesentlichen die Concurrency Utilities API vom java.util.concurrent Package der Java SE indem eine vom Container verwaltete Version der zentralen Klassen zur Verfügung gestellt wird – z.B. ManagedExecutorService für ManagedService und ManagedThreadFactory für ThreadFactory. Die Verwendung des selben Programmiermodels erleichtert somit den Einstieg für die Nutzung dieser neuen Funktionalität.
Neue Features in JMS 2.0 kommen den stendig steigenden Unternehmensanforderungen entgegen. Die Zulassung von mehreren gleichzeitigen Consumer der selben Topic-Subscription führt zu einer besseren Skalierbarkeit. Die Empfehlung, dass der JMS-Provider einen JCA-konformen Resource-Adapter mit sich bringt, wird die Integration von Third-Party JMS-Implementierungen in Java EE 7 kompatiblen Application Server deutlich erleichtern. Die Robustheit von Anwendungen wird durch die Verwendung der neuen Möglichkeiten der deklarativen Validierung von Methoden-Parameter und ihren Rückgabewerten in BeanValidation 1.1 erhöht.

- **Produktivitätssteigerung**

Java EE 7 bringt mit sich deutliche Produktivitätsverbesserungen. Sie manifestieren sich hauptsächlich in drei Bereichen: Reduktion des notwendigen Codes, um die Business-Logik zu schreiben, Nutzung von mehr Annotationen, um durch Konventionen die umständlichen Konfigurationen in XML zu reduzieren und eine bessere Integration der beteiligten Technologien untereinander.

Default Resources ist eine neue Feature, die vom Plattform-Provider die Bereitstellung von Managed Resources wie DataSource, JMSConnectionFactory und ManagedExecutorService an wohldefinierten Stellen im JNDI-Baum standardmäßig verlangt und somit dem Entwickler ihre Definition vor der Nutzung erspart. In JMS 2.0 kann man am deutlichsten sehen, wie durch die Einführung vom JMSContext und die Verwendung vom Default JMSConnectionFactory der notwendige Code zur Sendung einer Nachricht drastisch reduziert werden kann:

```
@Inject
JMSContext context;
@Resource(lookup = "java:global/jms/demoQueue”)
Queue demoQueue;
public void sendMessage(String payload) {
   context.createProducer().send(demoQueue, payload);
}
```

Die Einführung von Annotations in Java EE hat den notwendigen Konfigurationsaufwand deutlich verringert. Ihr Ausbau in Java EE 7 hat das POJO basierte Programmiermodell verstärkt und weiterentwickelt. Wie bereits gezeigt verwandeln WebSocket-Annotationen ein POJO in ein WebSocket Endpoint. JAX-RS 2.0 Annotationen wie @Provider transformieren POJOs in JAX-RS Filter, Interceptoren oder MessageBodyReader/Writer, ohne notwendige Konfiguration. Die Resource-Definitionen werden um @JMSDestinationDefinition und @MailSessionDefinition erweitert, was die Angabe von Metadaten im Source-Code statt im Deployment-Deskriptoren ermöglicht. Die Konvention, dass CDI standardmäßig aktiviert ist, macht das beans.xml File optional und verbessert somit weiter die Erfahrung des Entwicklers im Umgang mit der Java EE Plattform.

ManagedBeans wurden in Java EE 6 eingeführt und waren ein erster Schritt in Richtung eines Abgleichs von EJBs, JSF Managed Beans und CDI Beans. In Java EE 7 wird dieser Trend fortgesetzt, indem der Einsatz von CDI Beans in allen JSF-Artefakten ermöglicht und bevorzugt wird, so dass im Laufe der Zeit JSF Managed Beans verschwinden. Die deklarative und einfache Nutzung von Transaktionen im EJB Container kann durch eine generelle Lösung, die auf CDI Interceptor basiert, auf andere Java EE Komponenten ausgedehnt werden: Methoden, die mit @Transactional annotiert wurden, werden automatisch in einem transaktionalen Kontext ausgeführt. Weiterhin wurde die Bean Validierung auf Methoden-Parameter und Rückgabe-Werte ausgedehnt und vollständig in JAX-RS integriert.

Java EE 7 SDK

Der schnellste Einstieg in die Welt der neuen Java EE Technologien wird durch das Java EE 7 SDK geboten. Es beinhaltet den GlassFish 4.0 Server Open Source Edition (die Referenzimplementierung) und diverse Beispiele in der Form von Maven-Projekten, die entweder von einem IDE oder von der Kommando-Zeile gestartet werden können. Eine Gruppe von Beispielen ist unter dem <javaee7-sdk>/samples-Verzeichnis untergebracht und umfasst 41 exemplarische Anwendungen, die die neuen Java EE 7 API auf sehr einfache Weise beleuchten. Sie sind alle gleich strukturiert und haben eine Beschreibung, Key Features, Anweisungen zur Kompilierung, Installation und Ausführung und eine Troubleshooting-Sektion. Die andere Gruppe ist im Tutorial unter dem Verzeichnis <javaee7-sdk>/docs/javaee-tutorial zusammengefasst. Diese Beispiele sind etwas tiefgehender und demonstrieren auch die älteren API, die zu der Java EE 7 Plattform gehören. Ein End-to-End Beispiel mit Schwerpunkt auf dem Zusammenspiel der einzeinen Technologien liegt unter <javaee7-sdk>/docs/firstcup zum Ausprobieren bereit.

Entwicklungswerkzeuge

Die Open Source Entwicklungsumgebung NetBeans bietet seit der Version 7.3.1 umfangreiche Unterstützung für Java EE 7. Es gibt neue Java EE 7 Projekt-Typen auf der Basis von Maven oder Ant, Editor Code-Completion bei Nutzung der Java EE API und verschiedene Wizards, die z.B. die Erstellung von RESTful WebServices Clients, WebSocket Endpoints oder Datenbank-Scripts von Entity-Klassen vereinfachen. GlassFish 4.0 ist entweder mit gebündelt oder kann leicht angebunden werden, so dass der komplette Zyklus von Anwendungsentwicklung, Installation, Ausführung und Debugging unterstützt wird. Viele der Beispiele vom Java EE 7 SDK können auch in NetBeans über den Wizard für neue Projekte, Unterkategorie Samples → Java EE, ausprobiert werden.

Der Support für Java EE 7 ist auch eines der Fokus-Themen von Eclipse Kepler 4.3. Neue Java EE Projekte können auf Maven-Basis erstellt werden. Es gibt neue Fassetten für JPA 2.1, JSF 2.2, Servlet 3.1, JAX-RS 2.0, EJB 3.2 sowie Unterstützung für NamedStoredProcedureQuery (JPA 2.1), Konfiguration der Schema-Generierung in persistence.xml (JPA 2.1) und vieles mehr. GlassFish 4.0 kann über das GlassFish-Plugin für Eclipse leicht integriert werden. Eine alternative zu Eclipse Kepler 4.3 bietet noch das Oracle Enterprise Pack for Eclipse(OEPE) 12.1.2, in dem out-of-the-box neben Java EE 7 und GlassFish 4.0 auch andere Oracle Middleware Produkte wie WebLogic, Coherence und ADF unterstützt werden.

Java EE Next

Kaum wurde die Version 7 fertiggestellt, hat die Java EE Community angefangen, sich Gedanken über die nächste zu machen. Natürlich steht die Standardisierung für eine PaaS im Raum. Der Grundstein dafür wurde bereits in Java EE 7 durch die Einführung von mehreren Default-Ressourcen und den umfangreichen Metadaten für Resource-Definitionen gelegt. Die Mandantenfähigkeit und die Unterstützung für SaaS wurden zunächst im Kontext von Java EE 7 diskutiert, während erste Schritte

in diese Richtung schon in EclipseLink implementiert wurden. Der nächste logische Schritt für einen besseren JSON-Support wäre eine JSON-Binding API. Es stellt sich die Frage, ob die Integration von NoSQL Datenbanken durch eine Anpassung von JPA oder durch eine neue spezielle API adressiert werden sollte. Kann etwas in Bezug auf Modularität getan werden, als Vorbereitung für die in Java SE 9 vorgesehene Modularisierung der Java Plattform? Das sind momentan einige der Fragen, die die Java EE Community beschäftigen. Es gibt aber auch viele Stimmen, die in erster Linie den Stand von Java EE 7 konsolidieren und polieren möchten, bevor von neuen Features die Rede ist.

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 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 so einfach geworden ist, wie noch nie.

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