Kontext und Dependency Management in Java EE – Einführung in JSR-299

Björn Konrad
ORDIX AG
Wiesbaden

Schlüsselworte:

Dependency Injection, JSR-299, JEE6, JSF, JSP, EJB

Einleitung

Als Standard für Kontext und Dependency Injection ist der JSR-299 die Basis von JEE6 Umgebungen. Durch Dependency Management sollen sämtliche Schichten miteinander gekoppelt werden. Insbesondere lassen sich Zugriffe von Präsentation (JSF/JSP) auf Business Tier (EJB) erheblich vereinfachen.

Standardisierung von Managed Beans

Vor Java EE 6 gab es keine einheitliche Definition für den Begriff „Bean“ oder „Managed Bean“ in der Java Plattform. Eine Bean konnte beispielsweise eine JSF Managed Bean oder eine Enterprise Java Bean sein. Fähigkeiten einer Bean waren spezifikationsbezogen und mussten von Java EE Containern getrennt unterstützt werden.

Die Managed Bean Spezifikation, die nun Bestandteil der Java EE 6 Spezifikation geworden ist, standardisiert den Begriff „Managed Bean“ für die Java Plattform. Somit ist eine Bean ein vom Container verwaltetes Java Objekt, dessen Implementierung möglichst minimale Anforderungen hat - auch bekannt unter dem Akronym „Plain Old Java Object“ (POJO). Somit ist es beispielsweise möglich, eine Enterprise Java Bean in einer JSF Seite verwenden zu können. Weiterhin sollten Managed Beans generelle Basis-Services wie Resource Injection, Lifecycle Callbacks und Interceptoren unterstützen - Funktionalitäten, die in allen Schichten einer Java EE Anwendung sinnvolle Verwendung finden und helfen, den Applikationscode besser strukturieren und entkoppeln zu können.

Der JSR-299 als Standard für Managed Bean Services

Der JSR-299 „Kontext und Dependency Injection für die Java EE Plattform“ (kurz „CDI“) ist ein Standard, der diese Basis-Services spezifiziert. Die CDI wird somit zu einem wesentlichen Bestandteil der Java EE Spezifikation. Wie in Abbildung 1 dargestellt ist CDI die Grundlage weiterer Technologien der Java Plattform, die auf CDI und Managed Bean Services aufsetzen.

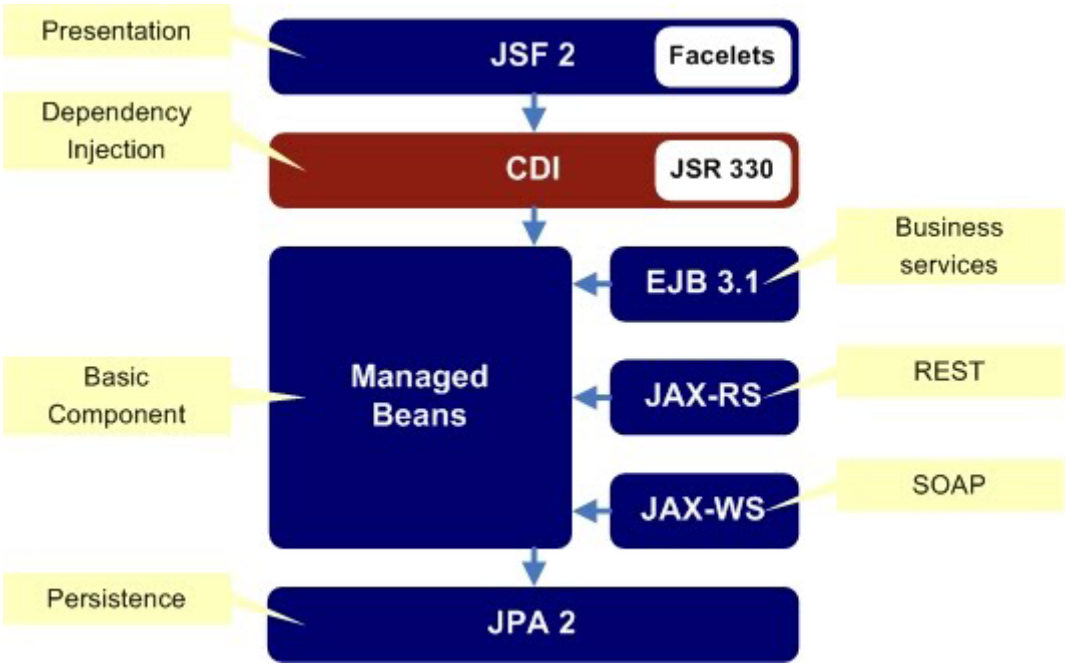

Abb. 1: CDI in JEE6

Der JSR verfolgt zwei wesentliche Ziele: Einerseits soll durch CDI eine möglichst lose Kopplung zwischen Komponenten des Applikationscodes erreicht werden. Auf der anderen Seite soll es der Standard ermöglichen, weitere Technologien und Frameworks viel einfacher in die Java Plattform integrieren zu können. Im Rahmen dieses Vortrags wird auf die wichtigsten Services und Funktionalitäten der CDI Spezifikation eingegangen. Dazu zählen Dependency Injection, Bean Lifecycle Kontext, Interceptoren, Dekoratoren, Events, Integration von CDI in die Java EE Umgebung sowie Portable Extensions.

Dependency Injection

Der Vortrag behandelt zunächst die Dependency Injection Mechanismen des JSR-299. Es werden ganz allgemein die Grundprinzipien und Vorteile von Dependency Injection erläutert. Anschießend geht der Vortrag auf das Programmiermodell des JSR-299 Standards ein. Es wird gezeigt, wie per Annotation das gesamte Objektmodell einer Applikation zusammengesteckt werden kann. Besonders wird darauf eingegangen, wie dies im Gegensatz zu anderen bekannten Dependency Injection Frameworks (beispielsweise Spring) auf typsichere Art und Weise (d.h. Compiler-geprüft) erfolgt.

Der Vortrag erläutert die verschiedenen Szenarien, Java Komponenten durch Dependency Injection lose koppeln zu können: Dazu gehören einerseits Qualifier Annotationen, die zur Compile-Zeit verwendet werden, um spezifische Komponenten-Implementierungen selektieren zu können (siehe nachfolgendes Beispiel).

```
@Stateless
public class BidService {

@Inject @JdbcDao
```

```
private BidDao bidDao;
...
}

@JdbcDao
public class LegacyBidDao implements BidDao {
@Resource(name="jdbc/ActionBazaarDB")
private DataSource dataSource;
...
}
```

Andererseits wird gezeigt, wie auch zur Deployment Zeit alternative Implementierungen einer Java Komponente injizierbar sind (beispielsweise Mock Implementierungen). Weiterhin geht der Vortrag auf Producer Methoden ein, die dafür sorgen, dass zur Laufzeit erzeugte Objekte ebenfalls per Dependency Injection kontrollierbar sind.

Bean Lifecycle Kontext

Der Bean Lifecycle Kontext ist ein weiterer Schwerpunkt dieses Vortrages. Hierdurch kann der Lebenszyklus zustandsbehafteter Objekte durch einfache Scope Type Annotationen gemanaged werden. Nachfolgendes Beispiel zeigt, wie eine Managed Bean per Annotation an den Request Lifecycle gebunden wird:

```
@Named
@RequestScoped
public class BidManager {
@Inject
private BidService bidService;
...
}
```

Es wird auf die grundsätzliche Vorgehensweise eingegangen und zeigt anhand von JSF/EJB Beispielen, wie die Lifecycle Annotationen der CDI Spezifikation eingesetzt werden können.

Interceptoren, Dekoratoren und Event Notifications

Das Themengebiet Interceptoren, Dekoratoren und Event Notifications wird näher erläutert. Diese Deklarationen ergänzen die Spezifikation um weitere Funktionalitäten, die zur losen Kopplung des Anwendungscodes führen. Während Interceptoren dafür verwendet werden, Geschäftslogik und technische Belange voneinander zu trennen, sorgen Dekoratoren dafür, Geschäftslogik noch weiter voneinander entkoppeln zu können. Durch Event Notifications wurde das Observer/Observable Pattern in die CDI Spezifikation aufgenommen. Im Rahmen des Vortrags wird anhand von einfachen Beispielen auf das Programmiermodell von Interceptoren, Dekoratoren und Event Notifications eingegangen.

Integration

Die Integration von CDI in die Java EE Umgebung ist ein weiterer Bestandteil des Standards. Der Vortrag geht darauf ein, welche Java EE Resourcen (beispielsweise ein EntityManager) in eine Bean injiziert werden können und welche Built-In Beans automatisch per Dependency Injection zur Verfügung stehen. Die Portable Extensions Funktionalität der CDI Spezifikation schließt den Vortrag ab. Portable Extensions bieten erhebliche Vereinfachungen zur Integration weiterer Technologien, Frameworks oder Erweiterungen in die Java Plattform. CDI ermöglicht dies durch die Bereitstellung verschiedener Service Provider Interfaces. Der Vortrag erläutert, wie sich hierdurch Business Process Management Engines oder Third-party Frameworks wie Spring, Seam, GWT oder Wicket einfach in die Java EE Umgebung integrieren lassen.

Kontaktadresse:

Björn Konrad
ORDIX AG
Kreuzberger Ring 13
D-65205 Wiesbaden

Telefon: +49 (0) 611- 778 4000
Fax: +49 (0) 180- 167 3490
E-Mail info@ordix.de
Internet: www.ordix.de