CDI

Zunächst gibt es 3 Möglichkeiten Objekte zu erstellen.

- new
- Factory
- Dependency Injection

Dependency Injection funktioniert nach dem Hollywood Prinzip (Don't call us we call you). Das bedeutet, dass man sich nicht selbst um die Erstellung eines Objektes kümmern muss sondern, dass der CDI Container das gewünschte Objekt zur Verfügung stellt.

Beans

Die durch CDI miteinander verknüpften Klassen werden in der CDI-Spezifikation Managed Beans genannt. CDI Beans können sowohl injizierte Objekte darstellen als auch als Injektionsziel dienen.

Die Anforderungen an CDI Bean sind denkbar gering: Nahezu jede konkrete Java-Klasse ist dazu geeignet. Benötigt wird nur ein Konstruktor ohne Parameter (wir werden später sehen, dass auch Klassen mit anderen Konstruktoren CDI Beans sein können).

CDI beachtet allerdings nicht alle Klassen im Classpath. Zusätzlich zu den Klassen selbst ist eine Datei namens beans.xml notwendig, die komplett leer sein darf.

Scopes

Scopes sind verantwortlich für den Lebenszyklus von beans, da sich der CDI Container um die Erstellung und die Vernichtung kümmert. Es sollte also festgelegt werden, wann beans erstellt bzw. gelöscht werden sollen.

CDI hat einige vordefinierte Scopes:

- Application Scope
  - Der Application Scope (@ApplicationScoped) ist im Grunde genommen ein Singleton welcher den gesamten Lebenszyklus einer Applikation überlebt.
- Session Scope
  - Der Session Scope (@SessionScoped) hält eine Instanz speziell für einen Benutzer. Das kann beispielsweise sinnvoll sein, wenn bei einer Web-Applikation der Benutzername oben rechts in der Ecke angezeigt werden soll.
- Request Scope
  - Der Lebenszyklus des Request Scopes (@RequestScoped) hält beans im Gegensatz zum Session Scope nur während einzelnen http-Requests aufrecht.
- Conversation Scope
  - Der Conversation Scope (@ConversationScoped) ist vorgesehen für Interaktionen welche sich über eine Reihe von Requests strecken. Im Gegensatz zum Request Scope besitzt der Conversation Scope keinen vordefinierten Lebenszyklus, stattdessen kann die Applikation Einfluss darauf nehmen wann eine neue Konversation erstellt wird. Bei einem

Bestellvorgang kann dieser Scope sinnvoll sein, wenn von Benutzer die Adresse, Zahlungsinformationen usw. verlangt werden.

Interceptors

Ein Interceptor ist eine Klasse, die zum Einfügen in Methodenaufrufe oder Lebenszyklusereignisse verwendet wird. Der Interceptor führt eine Trennung Aktivitäten durch, indem er Aufgaben wie Logging oder Überwachung ausführt, die nicht mit der Geschäftslogik der Anwendung zusammenhängen und die häufig innerhalb einer Anwendung wiederholt werden. Zb. Alle DAO Methoden!

Für einen Interceptor muss ein pointcut definiert werden. Ein pointcut ist eine Methode die mittels Annotation angibt wann sie aufgerufen wird.

Decorators

Mit Decorators werden Methoden hinzugefügt, welche nicht in der eigentlichen Klasse verfügbar sind.

Events

Durch Events ist es einfach Observer Methoden zu definieren, die automatisch als Reaktion auf ein oder mehrere Ereignisse aufgerufen werden.

Beispiele