Relationale Datenbanken

Eine Einführung

JavaEE

Framework aufbauend auf Java

Bietet folgende Funktionalitäten:

- Datenbankenzugriff (Transactions)
- InjectionFramework (Beans, CDI)
- Webprotokolle (REST, WebSocket, etc)
- Skalierbarkeit
- Concurrency
- Management der Komponenten

Muss in einen ApplicationServer ausgeführt werden

- Glassfish
- Wildfly
- Tomcat

Apache Maven

Build Automatisierung

Bietet folgende Funktionalitäten:

- Wie soll die Software gebaut werden
- Welche Abhängigkeiten sind dafür notwendig und wo bekomme ich die her?

Beispiel

- Guava maven (einfach in die Suchmaschine eingeben)

Konfiguration

- Ein Maven Projekt wird über die „pom.xml“ konfiguriert, die im Hauptverzeichnis des Projektes liegt

Apache Maven – JavaEE Integration

Folgende Pakete werden benötigt:

```
<dependency>
    <groupId>javax</groupId>
    <artifactId>javaee-api</artifactId>
    <version>7.0</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.enterprise</groupId>
    <artifactId>cdi-api</artifactId>
    <scope>provided</scope>
    <version>2.0</version>
</dependency>
<dependency>
    <groupId>org.jboss.spec.javax.annotation</groupId>
    <artifactId>jboss-annotations-api_1.2_spec</artifactId>
    <scope>provided</scope>
    <version>1.0.0.Final</version>
</dependency>
```

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.3.6.Final</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.jboss.spec.javax.ws.rs</groupId>
    <artifactId>jboss-jaxrs-api_2.0_spec</artifactId>
    <scope>provided</scope>
    <version>1.0.0.Final</version>
</dependency>
```

JavaEE Datenbankenobjecte

Wie erstelle ich ein Datenbankenobject einfach in Java?

- Java Klasse anlegen
- Felder anlegen, welche in der Datenbankentabelle auftauchen sollen definieren
- Getter(!) und Setter(!) für diese Felder bereitstellen
- Auf Klassenlevel @Entity Annotation hinzufügen
- Es muss ein Feld geben mit @Id Annotation und @GeneratedValue

Beispiel

```
@Entity
public class Entry {


    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;

    @Column
    private String name;

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
} 24.04.2020
```

Getter und Setter

Getter und Setter werden von Hibernate und JPA benutzt um über Reflections die Felder setzen bzw. auslesen zu können.

Im Normalfall, werden Getter bzw Setter verwendet um eine Kapselung der Membervariablen einer Klasse vorzunehmen und so nur eine Manipulation von Außen über vordefinierte Schnittstellen zu ermöglichen.

Das heißt, nur Felder die von außen manipuliert werden können sollen, bieten überhaupt Setter und nur Felder die gelesen werden dürfen, bieten Getter Zusätzlich lässt sich so auch noch überprüfen, ob beim Setzen eines Wertes, dieser einen sinnvollen Wert hat.

JavaEE Beans

Werden vom Container (z.B. Wildfly) verwaltet

- @Stateless – hat keinen Zustand
- @Statefull – speichert sich Zustände, z.B. Membervariablen
- @Singleton – es existiert in der gesamten Anwendung nur eine Instanz davon

Ermöglichen Injektion von anderen Beans und den Zugriff auf die Datenbank über den EntityManager

JavaEE EntityManager

Verwaltet den Zugriff auf die Datenbank und ist überall involivert wo die DB manipuliert wird

- Nur verfügbar in „Bean“ Klassen (@Stateless, @Stateful, @Singleton)
- Wird unter anderen über @PersistentContext geladen
- Transaction am Anfang eines Bean Aufrufes geöffnet, am Ende geschlossen
- Find, createQuery,CreateNamedQuery häufigst genutze Funktionen

JavaEE JPA

Java Persistence API

- Schnittstelle die Spezifikationen liefert
- Hibernate als mögliche Implementation
- Ermöglicht direktes Arbeiten auf den Objekten

JavaEE JPQL

Java Persistence Query Language

- Vereinfachung und Abstraktion zu SQL

Beispiel

Select u from User u where u.login = „Sven“

Liefert direkt das Objekt „User“ zurück bzw alle Objekte die, die Bedingung erfüllen