Workshop: JPA Schnelleinstieg
Übungen

Ablauf

Dieser Workshop gliedert sich in mehrere Vortragsteile, die den Teilnehmern die Thematik JPA näherbringen. Im Anschluss daran sind jeweils einige Übungsaufgaben von den Teilnehmern – idealerweise in Gruppenarbeit – am Rechner zu lösen.

Voraussetzungen

1) Aktuelles JDK 11, idealerweise auch JDK 14/15, installiert
2) Aktuelles Eclipse installiert (Alternativ: NetBeans oder IntelliJ IDEA)

Teilnehmer

- Entwickler und Architekten mit Java-Erfahrung, die ihre Kenntnisse zu Hibernate und JPA vertiefen möchten

Kursleitung und Kontakt

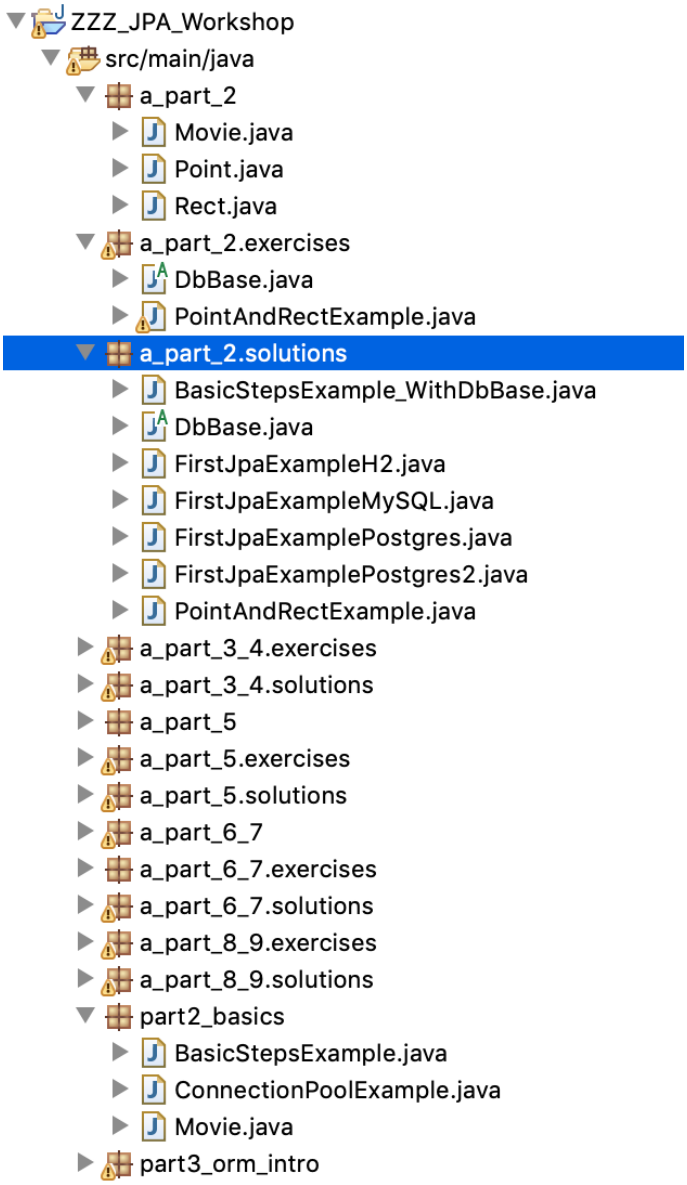
Michael Inden
Freiberuflicher Buchautor, Trainer und Konferenz-Speaker
E-Mail: michael.inden@hotmail.com
Blog: https://jaxenter.de/author/minden

Weitere Kurse (Java, Unit Testing, Design Patterns, Refactorings, ...) biete ich gerne auf Anfrage als Inhouse-Schulung an.

PART 2: Basisbausteine

Set Up 15 min

1. **Download + Entzippen**
2. **Gradle / Maven Build**
   a. `gradle clean assemble`
   b. `gradle cleanEcl ecl`
   c. `mvn clean install`
3. **Projektimport (Eclipse / IntelliJ)**



4. **HSQLDB starten**

```
java –classpath ../lib/hsqldb.jar org.hsqldb.server.Server
```

Aufgabe 1: Persistence Unit Entities 15 min

Aufgabe 1 a: Im Projekt sind die zwei Klassen **`Point.java`** und **`Rect.java`** vorgegeben sowie eine Datei **`persistence.xml`**, in der verschiedene Persistence Units definiert sind. Ergänzen Sie in der PU `java-profi-PU-PART-2-EXERCISES` die notwendigen Informationen zu den zu verwaltenden Klassen. Persistieren Sie in der Methode **`executeStatements()`** ein paar Instanzen der beiden Klassen **`Point.java`** und **`Rect.java`**. Prüfen Sie dies durch eine Query von beiden Typen und den Start des Programms **`PointAndRectExample.java`**.

Aufgabe 1 b: Leider sehen wir momentan noch keine Ausgaben zu den von Hibernate generierten SQL-Anweisungen. Aktivieren Sie die Protokollierung und führen Sie das Programm erneut aus.

Aufgabe 2: Grundgerüst für Aktionen 30 min

Aufgabe 2 a: Im Projekt ist bereits eine einfache Entity **`Movie.java`** vorgegeben genauso wie eine Klasse **`BasicStepsExample.java`**, um erste Gehversuche ausführen zu können. Erstellen Sie aus der bereits vorgegeben Klasse **`BasicStepsExample.java`** eine Klasse **`DbBase.java`**, so dass diese für weitere Aktionen parat ist.

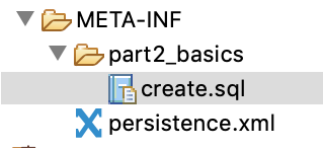
Aufgabe 2 b: Erzeugen Sie die Klasse **`BasicStepsExampleWithDbBase.java`** basierend auf **`BasicStepsExample.java`** durch den Einsatz der gerade erstellten Basisklasse **`DbBase.java`**.

Aufgabe 2 c: Die Aktionen sollen nun eingebettet in JUnit 4 / 5 erfolgen können. Erstellen Sie einen passenden Test **`MovieDbTest.java`** unter **`src/test/java`**.

Aufgabe 2 d: Erstrebenswert ist es, die Aktionen ähnlich wie in der Klasse **`DbBase.java`** auch für Tests bündeln zu können. Was erschwert es diesen zu verallgemeinern? Erstellen Sie eine Testklasse wie folgt: `public class MovieDbBaseTest extends AbstractDbBaseTest`

Aufgabe 3: Persistence Unit 15 min

Experimentieren Sie mit einem Create-Skript. Binden Sie dieses geeignet ein.



Aufgabe 4: Verschiedene Datenbanken 15 min

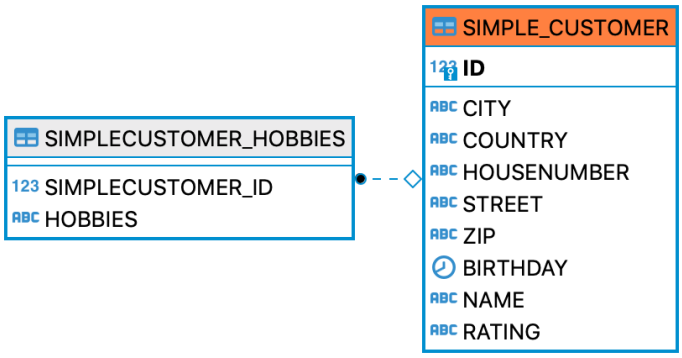
Basierend auf der Klasse **`BasicStepsExample.java`** sollen Sie Verbindung zu verschiedenen Datenbanken herstellen. Erstellen Sie dazu etwa folgende Persistence Units und Klassen **`JpaExampleH2/MySql/`**... Welchen Vorteil bring hierbei schon die Utility- Klasse **`DbBase.java`**?

- `java-profi-PU-H2`
- `java-profi-PU-MYSQL`
- `java-profi-PU-POSTGRES`

PART 3 / 4: Einstieg ORM / More ORM

Aufgabe 1: Entity SimpleCustomer 20 min

Erstellen Sie eine einfache Entity **SimpleCustomer.java**, die Attribute für Name, Geburtstag, eine Kundenbewertung (Gold, Silber, Bronze), eine Adresse und eine Liste von Hobbies besitzt. Nutzen Sie einige der vorgestellten Techniken wie Embeddable, Verwaltung von Enums und Element Collections. Prüfen Sie Ihre Implementierungen mit der Hauptklasse **SimpleCustomerExample.java**.

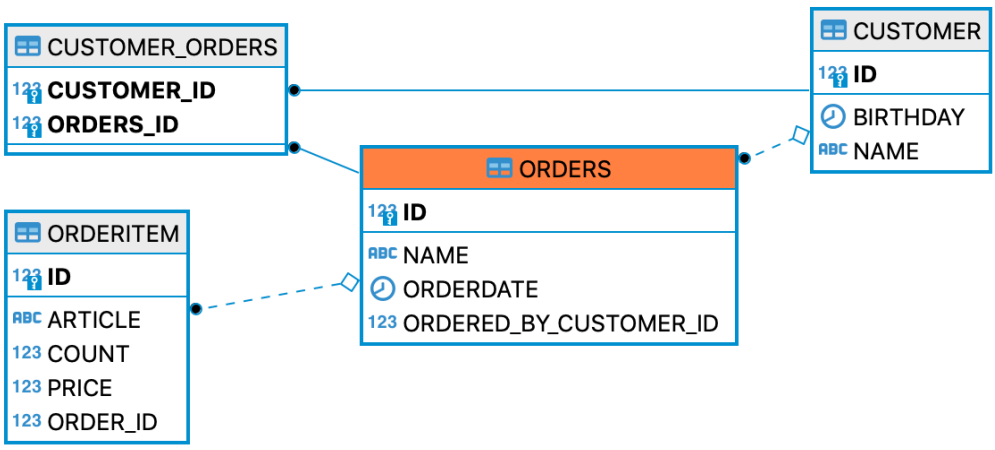


Aufgabe 2: Entity Order und OrderItem 30 min

Modellieren Sie eine Bestellung als Entity **Order.java**, die einem Kunden zugewiesen ist und verschiedene Bestellpositionen in Form der Entity **OrderItem.java** besitzen kann. Erweitern Sie bei Bedarf die Modellierung des Kunden und nutzen dazu eine Entity **Customer.java**. Welche Art von Beziehung haben **Order** und **OrderItem** und **Order** und **Customer**. Bedenken Sie folgende Fragestellungen:

- Wie wirkt sich das auf die Modellierung aus?
- Wie muss die Kaskadierung aussehen?
- Weitere Diskussionspunkte zur Modellierung: Wie sieht es mit den eingebundenen Adressen aus? Zuvor haben wir genau diese als sinnvolle Erweiterung gesehen? Gilt das auch hier?

Prüfen Sie Ihre Implementierungen mit der Hauptklasse **CustomerExample.java**.



Aufgabe 3: Entity `TreeItem` 30 min

Modellieren Sie eine Baumstruktur mithilfe der Entity **`TreeItem.java`**, die wiederum weitere Einträge vom Typ **`TreeItem`** verwaltet.

Tipp: Hilfreich kann folgender Artikel sein:
https://www.codejava.net/frameworks/hibernate/hibernate-parent-child-mapping-example

Bonus: Erzeugen Sie eine schöne Darstellung des Baums!

```
Root                                  Root
|-- Analysis                          |-- Analysis
--- --- Object Modelling     =>       |   \-- Object Modelling
|-- Implementation                    \-- Implementation
--- --- Coding                            |-- Coding
--- --- Testing                           \-- Testing
```

Bonus II: Füge weitere Phasen hinzu und noch eine weitere Ebene.

Aufgabe 4: Vererbung 15 min

Rekapitulieren Sie die Vor- und Nachteile der verschiedenen Varianten Vererbung mit JPA abzubilden. Was fühlt sich natürlich an? Was ist bezüglich der Performance zu bedenken? Welche Einschränkungen existieren bei den verschiedenen Varianten?

Aufgabe 5: Vererbung und Non-Entity Base Class 15 min

Erstellen Sie eine Basisklasse **`UuidBase.java`** für Entities, die selbst keine Entity ist. Wozu kann das nützlich sein? Was gibt es hierbei für **`equals()`** und **`hashCode()`** zu bedenken?

Aufgabe 6: EntityGraph 30 min

Vollziehen Sie die Beispiele für Entity Graphs unter

- https://turkogluc.com/understanding-jpa-entity-graphs/
- https://turkogluc.com/understanding-the-effective-data-fetching-with-jpa-entity-graphs-part-2/

nach.

PART 5: JPQL

Aufgabe 1: JPQL Queries 15 min

Schreiben Sie ein paar Queries:

a) Alle Haustiere geboren vor / nach gegebenem Datum
b) Alle Haustiere, deren Mutter unbekannt ist.
c) Alle Haustiere, deren Mutter einen bestimmten Namen hat.

Aufgabe 2: JOIN FETCH 15 min

Gegeben ist die Klasse **`MovieRentalJoinFetchExample.java`**. Diese verdeutlicht die Auswirkungen von N + 1 Select. Vollziehen Sie dies nach und beheben Sie die Probleme durch JOIN FETCH.

PART 6 / 7: Transaktionen und DAOs

Aufgabe 1: Transaktionsgröße 30 min

In den Folien wurde eine Chunked-Verarbeitung gezeigt. Vollziehen Sie das Beispiel der Klasse **`Point.java`** und der Alles-Oder-Nichts bzw. Chunked- Verarbeitung nach.

Aufgabe 2: DAO einführen 30 min

Gegeben sei eine einfache CRUD-Applikation **`PersonDbAccesssWithoutDaoExample.java`** mit direkten Zugriffen auf die DB per `EntityManager`. Führen Sie als Abstraktion einen DAO als Klasse **`PersonDAO.java`** ein und nutzen Sie diesen in einer Klasse **`PersonDaoExample.java`**.

HINWEIS: Vollziehen Sie die Fallstricke rund um `refresh()` nach.

BONUS: Reichern Sie diesen ggf. mit ein paar Hilfsmethoden an. Ist es noch ein DAO oder ein Repository? Und: Ist das wirklich von Relevanz?

Aufgabe 3: Design-Diskussion 30 min

Mit dem Wissen um DAOs und REST-Applikationen, wie sollte ein Design aussehen. Skizzieren Sie einen REST-Controller, Service und Datenzugriff mit DAO oder Repository. Wie handhabt man die Konvertierung von Entity in DTO. Benötigt man diese noch? Wie erfolgt das Mapping? Schauen Sie sich einmal das Tool MapStruct an: https://mapstruct.org/

PART 8 / 9: Entity Listener / Validation / Caching

Aufgabe 1: EntityListener erweitern 20 min

Korrigieren/ergänzen Sie die gegebene `EntityListener`-Implementierung in der Klasse **`Employee.java`**, sodass alle Callback-Meldungen des `EntityListener` ausgegeben werden. Belegen Sie das Attribut `greeting` mit einem Gruß, beispielsweise «Hallo Peter» oder «Grüezi Ueli». Erweitern Sie auch die Klasse **`AuditingEntityListenerExample.java`** passend.

Aufgabe 2: Validation 20 min

Gegeben sei eine Klasse **`Customer.java`**, bislang ohne Validierung. Diese soll nun hinzugefügt und ausgewertet werden. Folgende Bedingungen sollen gelten:

- Der Vorname ist nicht leer und zwischen 2 und 100 Zeichen lang
- Der Nachname darf nicht leer sein.
- Das Alter muss positiv oder 0 sein.
- Das Geburtsdatum muss in der Vergangenheit liegen.
- Der Start des nächsten Urlaubs muss heute oder in der Zukunft liegen.
- Schließlich soll die Kontakt-Email ein gültiges Format besitzen.

Erweitern und führen Sie zur Prüfung die Klasse **`CustomerValidationExample.java`** aus.

Aufgabe 3: Caching 30 min

Experimentieren Sie mit verschiedenen Caching-Strategien. Welche Entities werden dann gecacht und warum? Eliminieren Sie Entities aus dem Cache und prüfen Sie nach, dass die Anzahl passend reduziert wurde.