

# Hibernate

Eine kurze Einführung von Jannik Adam

28.11.2018

# Fragen

- Was ist Hibernate?
- Wann ist es sinnvoll Hibernate zu verwenden?
- Wie ist Hibernate aufgebaut?
- Wie nutze ich Hibernate?

# Das Problem das Hibernate lösen soll

- ▶ Zusammenarbeit von:
  - ▶ **objektorientierter Software** möchte Daten mit einer
  - ▶ **relationalen Datenbank(en)** speichern
- ▶ Problem
  - ▶ **Synchronisierung** zweier verschiedener Darstellungen von Daten
- ▶ Lösung:
  - ▶ Objekt/Relational Mapping (**ORM**)
  - ▶ Hibernate ist eine ORM Software

# Was genau ist das Problem?

„Object-relational impedance mismatch“:

1. Feinheit
2. Vererbung
3. Identität
4. Assoziationen
5. Navigation

# Macht Hibernate für unser Projekt Sinn?

WIR BENUTZEN:

**JAVA**

**EINE RELATIONALE DATENBANK**

**EIN STATISCHES OO DOMAIN MODEL**

**GEEIGNET FÜR CRUD-OPERATIONEN UND  
WENIGE KOMPLEXE DB-ANFRAGEN**

# Teile einer Anwendung mit Hibernate

- ▶ Hibernate Bibliothek
  - ▶ <http://hibernate.org/orm/releases/standalone/maven/>
- ▶ Hibernate Konfiguration mit Datenbank
- ▶ Hibernate Mapping der Klasse(n)
- ▶ POJO Klasse(n)
- ▶ Logik/ Main-Klasse

# POJO – Plain Old Java Objects

```
Public class PojoClass {  
    int id;  
    String wert2;  
  
    public int getid() {  
        return this.id;  
    }  
  
    public void setid(int id) {  
        this.id = id;  
    }  
  
    ....
```

# 6 Schnittstellen von Hibernate

- ▶ Konfiguration
- ▶ SessionFactory
- ▶ Session
- ▶ Transaction
- ▶ Query + Criteria API



# Ausdruck einer typischen Transaktion

```
SessionFactory factory = new Configuration().configure().buildSessionFactory();
```

```
...
```

```
Session session = factory.openSession();
```

```
Transaction tx = null;
```

```
try {
```

```
    tx = session.beginTransaction();
```

```
    // Arbeit ...
```

```
    tx.commit();
```

```
} catch (Exception e) {
```

```
    if (tx != null)
```

```
        tx.rollback();
```

```
    e.printStackTrace();
```

```
} finally {
```

```
    session.close();
```

```
}
```

# Konfiguration

- ▶ Konfiguration für Hibernate
  - ▶ Konfiguration meistens in einer Datei (hier: hibernate.cfg.xml)
- ▶ Wird der SessionFactory mitgegeben

# SessionFactory

- ▶ Generiert Session-Instanzen
- ▶ Konfiguriert Hibernate für die Verwendung der Datenbank
- ▶ Eine Session Factory für gesamte Anwendung pro Datenbank

# Session

- ▶ Primäre Schnittstelle der Hibernate-Anwendung
- ▶ Beinhaltet Persistenz-bezogene Operationen
- ▶ Behält Objekte unter eigener Führung, bevor es in die Datenbank übernommen wird

# Transaktionen

- ▶ Ein Arbeitsschritt mit der Datenbank
- ▶ Werden abgewickelt vom unterliegenden Transaktionen Manager (Bsp.: JDBC)
- ▶ optional

# Query & Criteria API

- ▶ Query:
  - ▶ SQL oder HQL
- ▶ Criteria:
  - ▶ Alternative bzw. Erweiterung zur Query Schnittstelle
  - ▶ Filterregeln und logische Bedingungen
  - ▶ Projektionen/ Aggregationen
  - ▶ Sortierung
  - ▶ Pagination

# Nutzung von Hibernate

- ▶ Hibernate entweder durch Standalone oder Maven einbinden
- ▶ Erstellen von:
  - ▶ hibernate.cfg.xml (Konfiguration)
  - ▶ <Klassenname>.hbm.xml (Mapping)
  - ▶ <Klassenname>.java (POJO)
  - ▶ <Manager\_Von\_<Klassenname>>.java (Anwendung)

# Beispiel

- ▶ Okay, lasst es uns angehen!
- ▶ [https://www.tutorialspoint.com/hibernate/hibernate\\_many\\_to\\_one\\_mapping.htm](https://www.tutorialspoint.com/hibernate/hibernate_many_to_one_mapping.htm)



# Mapping

- ▶ Von Hand oder
- ▶ Generierung durch Tool:
  - ▶ XDoclet
  - ▶ Middlegen
  - ▶ AndroMDA
  - ▶ Hibernate selbst bietet Editor an

# Sachen zum Nachschlagen

- ▶ Annotations
- ▶ Criteria API
- ▶ Hibernate OGM (für NoSQL-Lösungen)
- ▶ Batch Processing
- ▶ Mapping Types
- ▶ Und noch viel mehr ...

<http://hibernate.org/orm/>

<https://de.slideshare.net/amit.himani/intro-to-hibernate>

<https://www.tutorialspoint.com/hibernate/index.htm>



# Danke für eure Aufmerksamkeit!

- ▶ Meine persönliche Empfehlung:
  - ▶ <https://bit.ly/2FFWmJV>