Datenzugriffskomponenten mit JPA 2.1

Nils Grünewald

Aktuelle Technologien zur Entwicklung verteilter Java-Anwendungen

Agenda

1. Einleitung
2. Schichtenmodell
3. Einbindung & Konfigurataion
4. EntityManager
5. JPQL
6. Criteria-API

1. Einleitung

- JPA = Java Persistence API
- Release v1.0 am 11.05.2006
- Aktuell: Java EE 7, 06.12.2013
- Verwaltung der Datenbank
- Zugriff auf Datenbank über
  - Standardfunktionen
  - JPQL
  - Criteria API

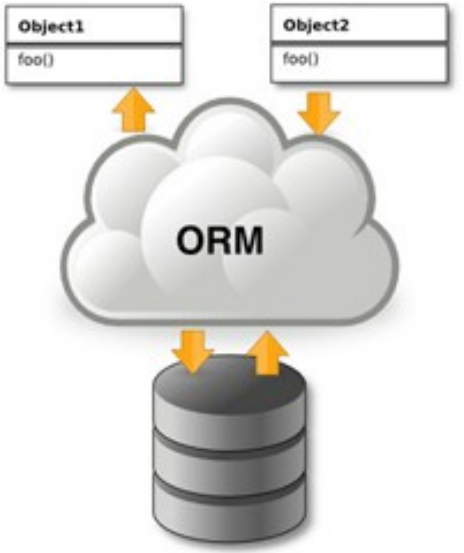


Abb 1

Frameworks

- EclipseLink (www.eclipse.org)
- Hibernate (www.hibernate.org)

2. Schichtenmodell

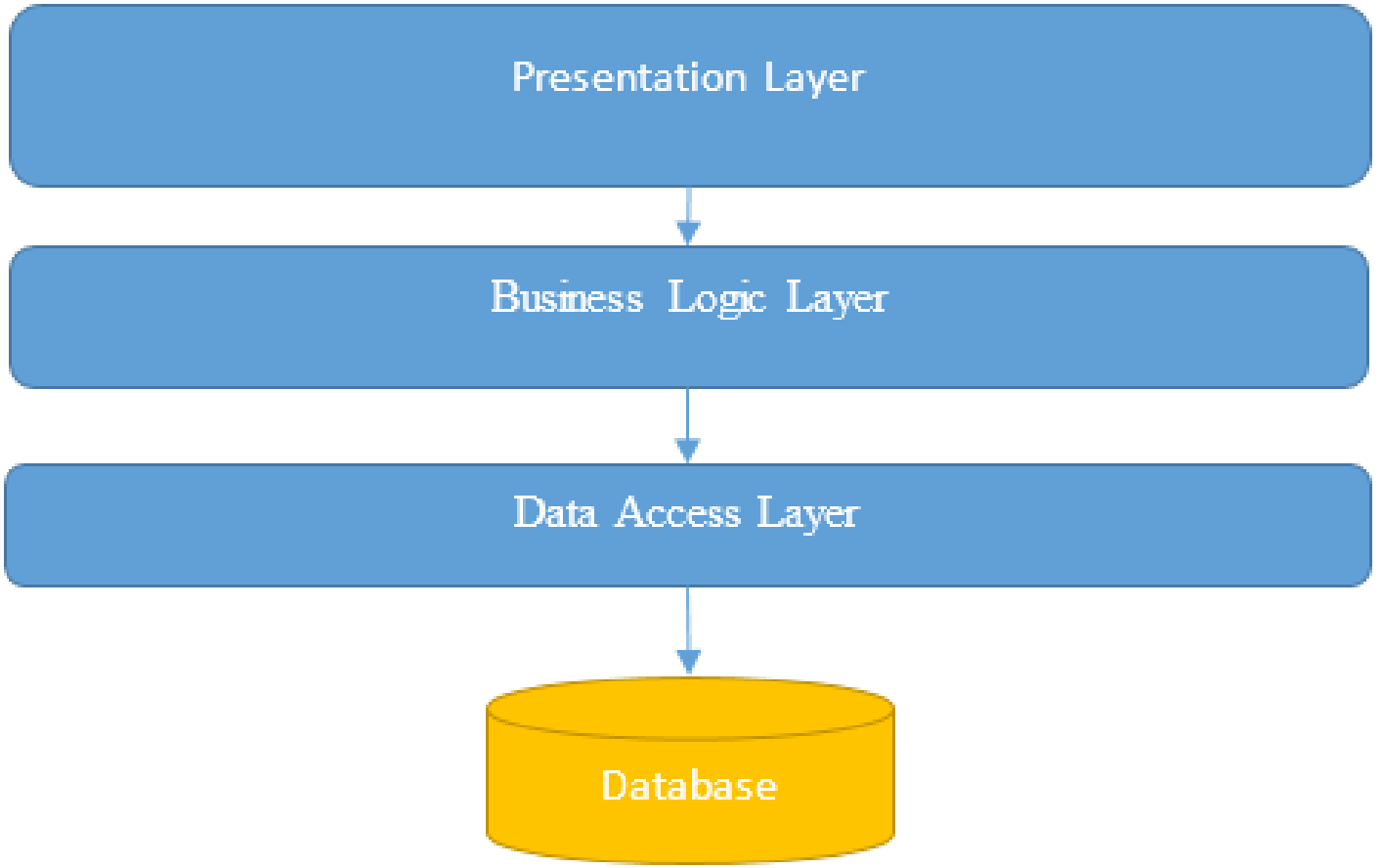


Abb 2

3. Einbindung

```
<dependencies>
	<dependency>
		<groupId>org.eclipse.persistence</groupId>
		<artifactId>eclipselink</artifactId>
		<version>2.5.0-RC1</version>
	</dependency>
	<dependency>
		<groupId>mysql</groupId>
		<artifactId>mysql-connector-java</artifactId>
		<version>6.0.2</version>
	</dependency>
</dependencies>
```

3. Konfiguration

```
<persistence-unit name="jpa-demo">

<class>model.Book</class>

<properties>
	<property name="javax.persistence.jdbc.url"
value="jdbc:mysql://localhost:3306/jpadb?serverTimezone=UTC" />
	<property name="javax.persistence.jdbc.user" value="root" />
	<property name="javax.persistence.jdbc.password"
value="wA_20*ä" />
	<property name="javax.persistence.jdbc.driver"
value="com.mysql.cj.jdbc.Driver" />
	<property name="eclipselink.logging.level" value="FINE" />
	<property name="eclipselink.ddl-generation" value="create-
tables" />
</properties>

</persistence-unit>
```

3. DDL-Generation

Value	Beschreibung
none	Dies ist als Standardwert vorbelegt. Hier können keine Tabellen erzeugt werden
create-tables	Für jede Entity wird eine Tabelle erzeugt. Sollte es die zugehörige Tabelle schon geben, wird diese im Normalfall weiterhin verwendet.
create-or-extend-tables	Das selbe wie „create-tables“ mit der Möglichkeit zur Veränderung der Tabelle.
drop-and-create-tables	Für Entwicklung gedacht, nicht in produktiv: Zu jeder Entity werden alle eventuell existierende Tabellen gelöscht und anschließend neu erzeugt.

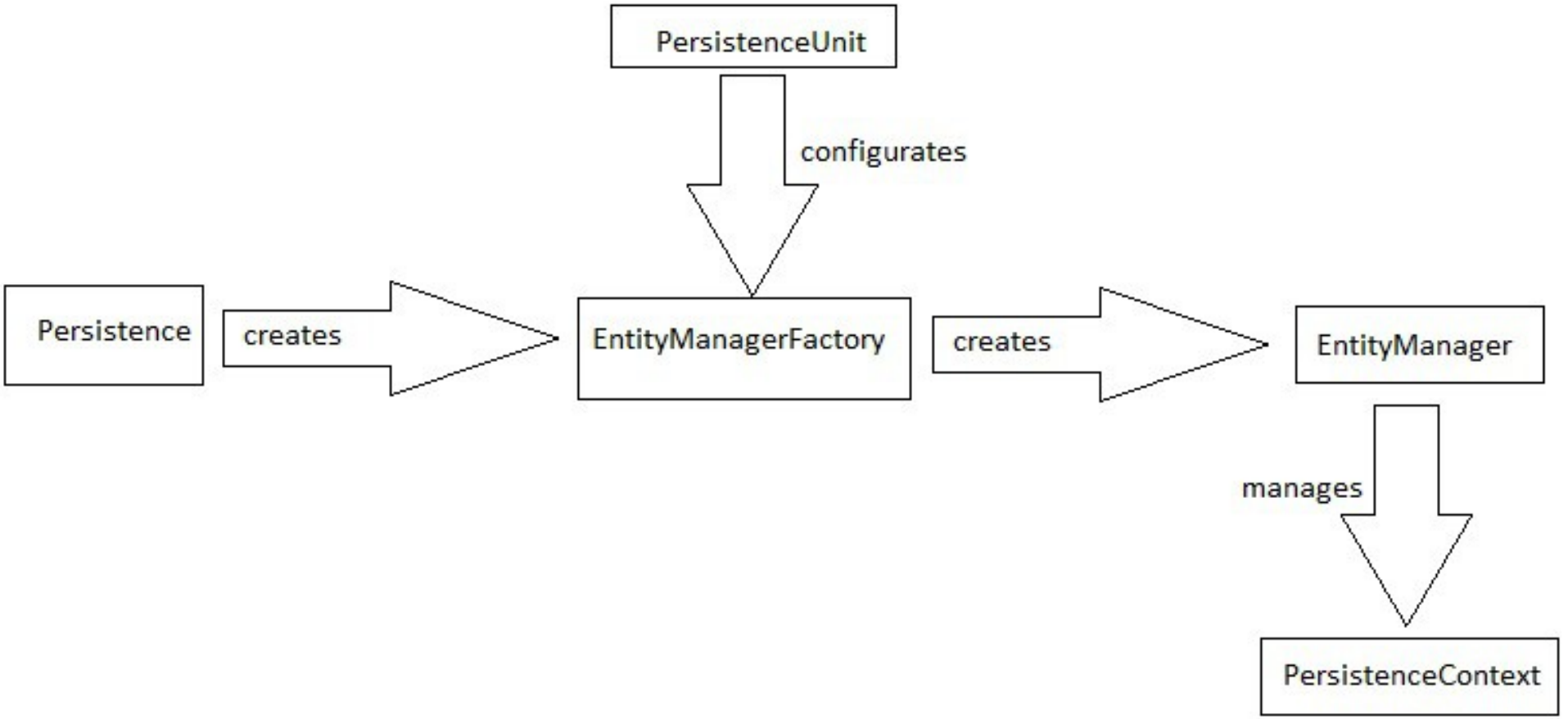
Entitäten

- Klasse
  - Nach außen sichtbar
  - Deklaration als Entity
  - Definition des Primärschlüssels
- Konstruktor
  - Default-Konstruktor benötigt
  - Public/Protected
- Beziehungen
  - Über Annotationen
  - 1:1, 1:n, n:1, n:m
- Datenbanktabelle
  - Spalten entsprechend Java-Klasse
  - Primärschlüssel

EntityManager

- Zentrale Steuereinheit
- Zwischenspeicher: Persistence Context
- Verwaltet JPA-Entities
- Erzeugt über:
  - Persistence.createEntityManagerFactory()
  - EntityManagerFactory.createEntityManager()
- Factory konfiguriert über PersistenceUnit

EntityManager



5. JPQL

- Java Persistence Query Language
- SQL-ähnlicher Code
- Anweisungen
  - SELECT, UPDATE, DELETE
- Komplexe Anweisungen
- Erstellen von Entitäten wie bisher
- Typisierte Queries

5. Query

- EntityManager.createQuery()
- Syntax
  - [SELECT Identifikationsvariable.Attribut]
    FROM Entity [AS] Identifikationsvariable
  - Entity = Klassenname

6. Criteria-API

- Komplexe Abfragen wie JPQL
- Kein SQL-Code → Java-Methoden
- → typsicher
- SQL-Code generiert
- Anweisungen
  - CriteriaQuery, CriteriaUpdate, CriteriaDelete

6. Criteria-API

1. CriteriaBuilder beschaffen über EntityManager
2. CriteriaQuery erzeugen über Builder,
   1. CreateQuery des Builders
   2. Mitgabe des Klasennamens
3. CreateQuery()
4. Ergebnis über getResultList() → TypedQuery

```
CriteriaBuilder criteriaBuilder = entitymanager.getCriteriaBuilder();
CriteriaQuery<Book> criteriaQuery =
      criteriaBuilder.createQuery(Book.class);
Query query = entitymanager.createQuery(criteriaQuery);
```

Criteria-API

- Komplexe Abfragen über CriteriaQuery
  - Select(), from(), where()
- Über Schnittstellen
  - Root
  - Path
  - Predicate

```
CriteriaBuilder criteriaBuilder = entitymanager.getCriteriaBuilder();
CriteriaQuery<Book> criteriaQuery = criteriaBuilder.createQuery(Book.class);

Root<Book> root = criteriaQuery.from(Book.class);
Path<Object> b_author = root.get("b_author");
Predicate predicate = criteriaBuilder.equal(b_author, "Nils Grünewald");

criteriaQuery.select(root).where(predicate);
Query query = entitymanager.createQuery(criteriaQuery);
```

Vielen Dank für Ihre Aufmerksamkeit

Quellen

Abb 1
https://github.com/amgohan/jackcess-orm

Abb 2
http://www.codeproject.com/Articles/853950/Data-transformation-Cornerstone-of-SOLID-architect