Die Spielregeln 1. Mappings 2. Das API 3. Abfragen

Frank Schwarz, buschmais GbR

Java Forum Stuttgart

Java Persistence Puzzlers

buschmais
Beratung . Technologie . Innovation

Ich frage.

Sie antworten.

```
package application;

public class AbstractDevice {

	protected int state = 0;

	void start() {
		state = 1;
	}

}
```

```
package application.devices;

import application.AbstractDevice;

public class Phone extends AbstractDevice {

	void start() {
		state = 2;
	}

}
```

```
package application;

import application.devices.Phone;

public class Main {

	public static void main(String[] args) {
		AbstractDevice device = new Phone();
		device.start();

		System.out.println(device.state);
	}

}
```

Wie verhält sich das Programm?

(a) Es gibt „1“ aus.
(b) Es gibt „2“ aus.
(c) Compile-Fehler in „Main“
(d) Compile-Fehler in „Phone“

Wovon handelt der Vortrag?

buschmais
Beratung . Technologie . Innovation

EJB 3 - Persistence "JPA 1"

JSR220, verabschiedet am 11. May 2006

JPA 2.0

JSR317, erscheint demnächst

Mappings

API

Abfragen

Spielregeln

Welche Frameworks werden betrachtet?

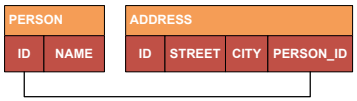
buschmais
Beratung . Technologie . Innovation

	OpenJPA	Hibernate	EclipseLink
Version	1.2.1	3.3.2 (Core) 3.4.0 (EM)	1.1.2
Standard	JPA 1	JPA 1	JPA 1
Hauptsponsor	IBM	RedHat	Oracle
SVN-Aktivität (1)			
Commits	1206	864 73	906 (3)
Änderungen	15482	9241 341	6928
Commiters (2)	6 (14)	4 (17) 3 (7)	8 (17)

(1) betrachtet über die letzten 52 Wochen mit Bezug zu JPA
(2) aktivste Commiters mit 80% aller Commits (alle Commiters)
(3) Trunk/JPA + Branches/1.1.0/JPA + Branches/1.0/JPA

http://svn.apache.org/repos/asf/openjpa
http://anonsvn.jboss.org/repos/hibernate
svn://dev.eclipse.org/svnroot/rt/org.eclipse.persistence

Welche Bedeutung hat die Gerichtetheit einer Beziehung?




```java
@Entity
public class Person {

	@Id
	private long id;
	private String name;

	@OneToMany
	@JoinColumn(name = "PERSON_ID")
	private Set<Address> addresses = new HashSet();

	// getters & setters

}
```

```java
@Entity
public class Address {

	@Id
	private long id;
	private String street;
	private String city;

	// getters & setters

}
```

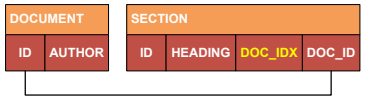
Wie viele der O/R-Mapper unterstützen dieses Mapping?

(a) Keiner, es ist nicht JSR220 konform.
(b) Einer.
(c) Zwei.
(d) Alle drei.

Sind Mengen auch Listen?

```java
@Entity
public class Document {

    @Id
    private long id;
    private String author;

    @OneToMany(mappedBy = "document")
    @o.h.annotations.IndexColumn(name = "DOC_IDX")
    private List<Section> sections = new ArrayList<Section>();

    // getters & setters
}
```

```java
@Entity
public class Section {

    @Id
    private long id;
    private String heading;

    @ManyToOne
    @JoinColumn(name = "DOC_ID")
    private Document document;

    // getters & setters
}
```



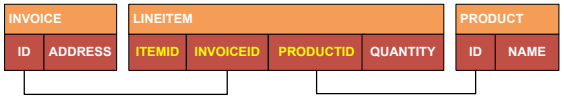
Funktioniert dieses Mapping?

(a) Ja.
(b) Ja, aber wegen des List-Typs redundant.
(c) Nein, das Mapping ist fehlerhaft.

"Additional mapping annotations (e.g., column and table mapping annotations) may be specified to override or further refine the default mappings […]. Such schema-level mapping annotations must be specified on the owning side of the relationship."

JSR220-Persistence, S. 25

Derived Identities bei JPA 2.0




```java
@Entity
@IdClass(LineItemId.class)
public class LineItem {

      @Id
      private long itemId;

      @Id
      @ManyToOne
      private Product product;

      @Id
      @ManyToOne
      private Invoice invoice;

      private long quantity;
}
```

```java
@Entity
public class Invoice {

      @Id
      private long id;

      private String address;

      @OneToMany(mappedBy="invoice")
      @OrderBy("id.itemId")
      private List<LineItem> lineItems;

      // getters & setters
}
```

```java
@Entity
public class Product {

      @Id
      private long id;

      private String name;

      // getters & setters
}
```

```java
@Entity
public class LineItem {

      @EmbeddedId
      private LineItemId id;

      //@Id?
      @ManyToOne
      private Product product;

      //@Id?
      @ManyToOne
      private Invoice invoice;

      private long quantity;

      // getters & setters
}
```

```java
@Embeddable
public class LineItemId {

      private long itemId;

      private long productId;

      private long invoiceId;

      // getters & setters
      // equals & hashCode
}
```

Wie heißt bei JPA 2 die Annotation, die die Referenz auf den Primärschlüssel herstellt?

(a)
```java
@Id(mappedTo="productId")
@ManyToOne
private Product product;
```

(b)
```java
@MappedById("productId")
@ManyToOne
private Product product;
```

(c)
```java
@EmbeddedIdAttribute("productId")
@ManyToOne
private Product product;
```

(d)
```java
@DerivedId("productId")
@ManyToOne
private Product product;
```

buschmais
Beratung . Technologie . Innovation

Garbage Collection a la JPA: Entfernung verwaister Objekte

Cascade-Optionen (`javax.persistence.CascadeType`):

- MERGE
- PERSIST
- REFRESH
- REMOVE
- ALL

```
@Entity
public class Person {

      // ...

      @OneToMany(
            cascade = {CascadeType.PERSIST, CascadeType.REFRESH},
            orphanRemoval = true)
      private Set<Address> addresses = new HashSet();

      // ...

}
```

Welche Bedeutung hat „`orphanRemoval=true`"?

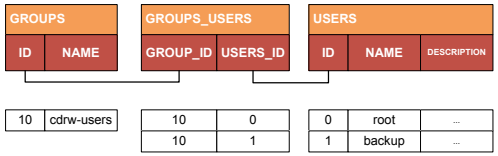
(a) Es wird ein kaskadierendes Löschen vorgenommen.

(b) Wie (a), zusätzlich werden Adress-Objekte, die aus der Collection per #remove(Object) entfernt werden, automatisch gelöscht.

(c) Wie (b), zusätzlich ist es nicht gestattet, Adress-Objekte aus einer Collection zu entnehmen und in eine andere einzufügen.

Mapper	Annotation	Bedeutung
EclipseLink	`org.eclipse.persistence.annotations.PrivateOwned`	(c)
Hibernate	`org.hibernate.annotations.Cascade( org.hibernate.annotations.CascadeType.DELETE_ORPHAN)`	(c)
OpenJPA	`org.apache.openjpa.persistence.Dependent`	(b)

Identitätsklau unter Objekten




```java
@Entity
@Table(name = "GROUPS")
public class Group {

      @Id
      private long id;

      private String name;

      @OneToMany(cascade = CascadeType.ALL)
      private Set<User> users;

      // getters & setters
}
```

```java
@Entity
@Table(name = "USERS")
public class User {

      @Id
      private long id;

      private String name;

      private String description;

      // getters & setter
}
```

```java
entityManager.getTransaction().begin();
Group detachedGroup = new Group();
detachedGroup.setId(10);
Group attachedGroup = entityManager.merge(detachedGroup);
entityManager.getTransaction().commit();
```

Welchen Effekt besitzt dieser Code-Schnipsel?

(a) Es tritt eine Exception auf, da #merge(Object) auf neuen Objekten nicht statthaft ist.

(b) Es tritt eine Constraint-Verletzung in der Datenbank auf, da eine neue Gruppe mit der Id=10 erstellt wird.

(c) Es passiert nichts.

(d) Der Inhalt von Group(10) wird zurückgesetzt, ebenso werden die Assoziationen zu den User-Objekten gelöscht. Die User-Objekte bleiben erhalten.

The semantics of the merge operation applied to an entity X are as follows:

- If X is a detached entity, the state of X is copied onto a pre-existing managed entity instance X' of the same identity or a new managed copy X' of X is created.
- If X is a new entity instance, a new managed entity instance X' is created and the state of X is copied into the new managed entity instance X'.
- [...]

The persistence provider must not merge fields marked LAZY that have not been fetched: it must ignore such fields when merging.

JSR220-Persistence, S. 51 f

Lesesperren auf Objekten

```
@Entity
public class Person {

      @Id
      private long id;

      @Version
      private long version;

      private String name;

      // getters & setters

}
```

```
entityManagerTS1.getTransaction().begin();
entityManagerTS2.getTransaction().begin();

Person personTS1 = entityManagerTS1.find(Person.class, new Long(1));
Person personTS2 = entityManagerTS2.find(Person.class, new Long(1));

entityManagerTS1.lock(personTS1, LockModeType.READ);

personTS2.setName("Torsten");

entityManagerTS1.getTransaction().commit();
entityManagerTS2.getTransaction().commit();
```

Tritt eine OptimisticLockException auf?

(a) Ja.

(b) Nein, es tritt aber eine andere Exception auf, die zum Zurückrollen einer der beiden Transaktionen führt.

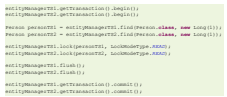
(c) Beide Transaktionen werden erfolgreich festgeschrieben.

(d) Das Verhalten hängt von der Datenbank ab.

Wenn eine Transaktion lock(entity, LockModeType.READ) auf einem versionierten Objekt aufruft, muss der EntityManager sicherstellen, dass keines der beiden Phänomene auftreten kann:

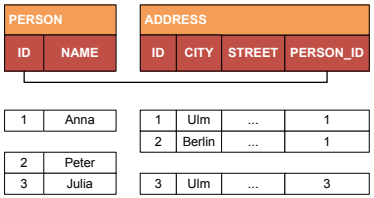
- P1 (Dirty read): Transaktion T1 ändert eine Zeile. Eine andere Transaktion T2 liest diese Zeile und erhält den modifizierten Wert, bevor T1 festgeschrieben oder zurückgerollt wird. T2 wird schließlich erfolgreich festgeschrieben.
- P2 (Non-repeatable read): Transaktion T1 liest eine Zeile. Eine andere Transaktion modifiziert oder löscht diese Zeile, bevor T1 festgeschrieben wird. Beide Transaktionen werden schließlich erfolgreich festgeschrieben.

JSR220-Persistence, S. 56



buschmais
Beratung . Technologie . Innovation

und wenn ja, wie viele?




```
Query query = entityManager.createQuery(
      "SELECT p FROM Person p, IN(p.addresses) a"+
      "   WHERE a.city = 'Ulm' OR a.city = 'Berlin'"
      );
Collection<Person> collection = query.getResultList();
System.out.println(collection.size());
```

Was gibt dieser Programm-Schnipsel aus?

(a) 2

(b) 3

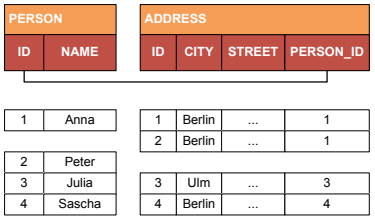
(c) 6

(d) Es hängt vom eingesetzten O/R-Mapper ab.

```
@Entity
public class Person {

      @Id
      private long id;
      private String name;

      @OneToMany(mappedBy = "person")
      private Set<Address> addresses = new HashSet();

      // getters & setters
}
```

```
@Entity
public class Address {

      @Id
      private long id;
      private String street;
      private String city;

      @ManyToOne
      private Person person;

      // getters & setters
}
```

Abfragen mit ALL-Quantor




```
Query query = entityManager.createQuery(
      "SELECT p FROM Person p WHERE" +
      " 'Berlin' = ALL (SELECT a.city FROM Address a WHERE a.person = p)"
      );
Collection<Person> collection = (Collection<Person>) query.getResultList();
System.out.println(collection.size());
```

Was gibt dieser Programm-Schnipsel aus?

(a) 2

(b) 3

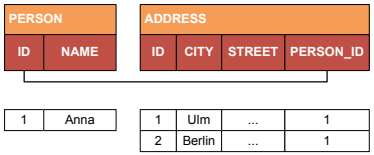
(c) 6

(d) Es hängt vom eingesetzten O/R-Mapper ab.

```
@Entity
public class Person {

      @Id
      private long id;
      private String name;

      @OneToMany(mappedBy = "person")
      private Set<Address> addresses = new HashSet();

      // getters & setters
}
```

```
@Entity
public class Address {

      @Id
      private long id;
      private String street;
      private String city;

      @ManyToOne
      private Person person;

      // getters & setters
}
```

Fetch-Joins




```
Query query = entityManager.createQuery(
      "SELECT DISTINCT p FROM Person p "+
      " JOIN p.addresses a "+
      " LEFT JOIN FETCH p.addresses"+
      " WHERE a.city = 'Berlin'"
      );
Collection<Person> collection = (Collection<Person>) query.getResultList();
Person person = collection.iterator().next();
System.out.println(person.getAddresses().size());
```

```
@Entity
public class Person {

      @Id
      private long id;
      private String name;

      @OneToMany(mappedBy = "person")
      private Set<Address> addresses = new HashSet();

      // getters & setters
}
```

```
@Entity
public class Address {

      @Id
      private long id;
      private String street;
      private String city;

      @ManyToOne
      private Person person;

      // getters & setters
}
```

Was gibt dieser Programm-Schnipsel aus?

(a) 0

(b) 1

(c) 2

(d) Es hängt vom eingesetzten O/R-Mapper ab.

buschmais

Beratung . Technologie . Innovation

Fragen & Antworten

buschmais

Beratung . Technologie . Innovation

http://code.google.com/p/oopex/

buschmais

Beratung . Technologie . Innovation

http://www.buschmais.de