Advanced JPA

Lecturer: Ondrej Mihályi

Lazy loading

- Simple types are loaded from DB at the time when entity is loaded (EAGER)
- Collections are loaded only when a method on a collection is executed (LAZY)
- Only reference to a proxy collection is injected to the entity
- This behavior can be changed by FetchType attribute on collection annotations
- When lazy collection is accessed when entity was disconnected from transaction, **LazyLoadingException** is thrown

Lazy loading behavior

- Default fetch behavior can be changed by FetchType attribute on collection annotations
- Lazy references can be loaded eagerly in JPQL using FETCH in joins

```
SELECT p from Person p JOIN FETCH p.children
```

Lazy loading problems

- lazy collection is accessed when entity was disconnected from transaction, LazyLoadingException is thrown
- There are various strategies to avoid this:
  - Data Transfer Objects, Eager queries, Entity Graphs …

Id generation for primary key

- an attribute of type long or String is marked with @Id
- Strategies to generate Id values:
  - IDENTITY – id generation by the DB
  - SEQUENCE – generation of id from a sequence
  - TABLE – emulates SEQUENCE using a table
  - AUTO – JPA provider will choose according to what is supported by the DB

Data integrity - problem

1) Adam’s transaction reads data X
2) Barbara’s transaction reads data X
3) Adam’s transaction modifies data X, and changes it to XA
4) Adam’s transaction **writes data XA**
5) Barbara’s transaction modifies data X and changes it to XB
6) Barbara’s transaction **writes data XB**

Optimistic locking

- Technique to prevent conflicts without using DB locking mechanism
  - DB locks are often too restrictive, slow down the DB, may cause deadlocks
- Each row of a table stores its version number
- Rollback when JPA finds out that the current version is older than the version in DB – already modified

Optimistick locking – solution

1) Adam’s transaction reads data X
2) Barbara’s transaction reads data X
3) Adam’s transaction modifies data X to XA
4) Adam’s transaction **writes data XA**
5) Barbara’s transaction modifies data X to XB
6) Barbara’s transaction tries to write data XB, but **receives and error**

Optimistick locking – solution

7) Barbara needs to read data XA (or start a completely new transaction)
8) Barbara’s transaction modifies data XA and changes it to XAB
9) Barbara’s transaction **writes data XAB**

Optimistic locking – version attribute

- @Version marks entity property, which is mapped to the version column
- Version number is updated and used by JPA automatically

JPA locks

- EntityManager methods
  - `find(Class entityClass, Object primaryKey, LockModeType lockMode)`
  - `lock(Object entity, LockModeType lockMode)`
  - other that retrieve data: `merge()` or `refresh()`
- Query: setLockMode `(LockModeType lockMode)`

Customization of DB mapping

- JPA listeners
  - @PrePersist, @PreRemove, @PostPersist, @PostRemove, @PreUpdate, @PostUpdate, @PostLoad

Customization of DB mapping

- Converters – new in Java EE 7
- @Convert to specify converter class
  - Can be on a field or on a class
- Converter class annotated with @Converter and implements AttributeConverter