Understanding the use of @Version annotations in JPA

Ask Question

I would like to use entity versioning in my application. I know a little how it works, but not quite. I know that an `@Version` annotation column should be created in the entity and the object will have the version `1` value when it is stored in the database. Each substitution of any field in this entity will result in the entity being incremented by 1.

The entity has the first version at first. I obtained this entity and replaced one field, but at the same time another user also obtained this entity, also replaced the field and quickly saved the object to the database incrementing the version to 2. So while I want to save this obsolete object, how should I compare whether the entity version agrees with my object

Or different situation as in this project. The entity has a column with the version https://github.com/Netflix/genie/blob/master/genie-core/src/main/java/com/netflix/genie/core/jpa/entities/AuditEntity.java. The user can update the entity by sending a DTO object that has a version field

```
private void updateEntityWithDtoContents(
        final ApplicationEntity entity,
        final Application dto
    ) throws GenieException {
        entity.setVersion(dto.getVersion());
}
```

The version in the object is set https://github.com/Netflix/genie/blob/master/genie-core/src/main/java/com/netflix/genie/core/jpa/entities/BaseEntity.java. However, here the version is just a field where the version is manually set and only writes itself to the database. So how to check if the Application is the current version with the base.

How can I check if the `Application` object is the same version to `ApplicationEntity` from the database?

java spring hibernate jpa spring-boot

asked Dec 18 '17 at 20:14
afsdgz
18 2

Very briefly: with @version annotation, you don't need to check the values on the database by yourself, Hibernate will do that for you, and it'll raise an `OptimisticLockException` if you are trying to save an outdated entity.

You can read more here

answered Dec 18 '17 at 21:14

gtosto
772 5 11

Just to increase, @Version is about concurrency updates in entity and this not duplicate the databse row with a new number, just increase that row number version. And you can't edit this number version mannualy, unless you perform a bulk update. – Henrique Fernandes Cipriano Dec 19 '17 at 20:01

When it comes to locking there are two ways that you can implement a lock in JPA, **Optimistic Locking** & **Pessimistic Locking**. To get a clear idea about the difference between these two, refer here.

1. Optimistic Locking: This can be implemented by introducing a **version** column to your table. You just need to put the **@version** annotation in JPA. Hibernate will handle all the heavy work. A sample code snippet is as follows

```
@Entity
@Table(name = "orders")
public class Order {
   @Id
   private long id;
```

```
    private String status;

    // ... mutators
}
```

Note that the version column may have different data types according to the use case, Please refer this for more info.

2. Pessimistic Locking: This can be implemented by using the **@Lock** annotation. A sample code is as follows

```
interface WidgetRepository extends Repository<Widget, Long> {

   @Lock(LockModeType.PESSIMISTIC_WRITE)
   Widget findOne(Long id);
}
```

answered Dec 23 '17 at 21:03

rnavagamuwa
96 9