Hibernate Tip: Share Primary Key in a One-to-One Association

Question:

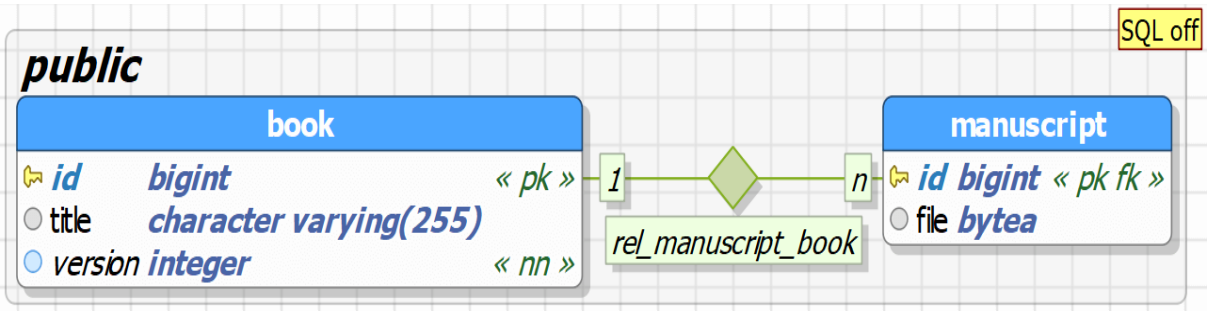
I need to map a one-to-one association in which the primary key value of one entity is also used as the primary key value of the other entity. How can I do that with JPA or Hibernate?

Solution:

You can use JPA's *@MapsId* annotation to tell Hibernate that it shall use the foreign key of an associated entity as the primary key.

Let's take a look at a simple example.

Each *Book* has a *Manuscript*, and each *Manuscript* belongs to 1 *Book*. The foreign key of the *Book* is also the primary key of the *Manuscript*.



Mapping the *Book* entity:

There is nothing special about the mapping of the *Book* entity. It defines the primary key attribute *id* and tells Hibernate to use a sequence to generate the primary key values. It also specifies the *title* attribute as a simple *String* and a one-to-one association to the *Manuscript* entity.

```
@Entity
public class Book {

    @Id
    @GeneratedValue(strategy =
GenerationType.SEQUENCE)
    @SequenceGenerator(name = "book_seq")
    private Long id;

    private String title;

    @OneToOne(mappedBy = "book")
    private Manuscript manuscript;

    ...
}
```

Mapping the *Manuscript* entity

The mapping of the *Manuscript* entity is more complex but also not very complicated. It defines an *id* attribute as the primary key and a *file* attribute of type *byte[]*.

The important part is the *book* attribute which defines the association between the *Book* and the *Manuscript* entity. The *@OneToOne* and the *@JoinColumn* annotations specify the association. The *@MapsId* annotation tells Hibernate to use the primary key value of the *Book* entity as the primary key value of the *Manuscript* entity.

```
@Entity
public class Manuscript {

    @Id
    private Long id;
```

```
private byte[] file;

    @OneToOne
    @JoinColumn(name = "id")
    @MapsId
    private Book book;

    ...
}
```

Persisting a new *Manuscript*

Let's give the mapping a try and persist a *Manuscript* for an existing *Book* entity.

```
Book b = em.find(Book.class, 1L);

Manuscript m = new Manuscript();
m.setBook(b);

b.setManuscript(m);

em.persist(m);
```

As you can see in the log output, Hibernate writes a new record to the *Manuscript* table.

```
06:45:12,563 DEBUG [org.hibernate.SQL] -
    select
        book0_.id as id1_0_0_,
        book0_.title as title2_0_0_,
        book0_.version as version3_0_0_,
        manuscript1_.id as id1_1_1_,
        manuscript1_.file as file2_1_1_
```

```
from
        Book book0_
    left outer join
        Manuscript manuscript1_
            on book0_.id=manuscript1_.id
    where
        book0_.id=?
06:45:12,645 DEBUG [org.hibernate.SQL] -
    insert
    into
        Manuscript
        (file, id)
    values
        (?, ?)
```

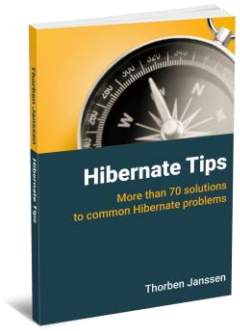
Learn more

JPA and Hibernate also support several other association mappings. I explain them in more details in Ultimate Guide – Association Mappings with JPA and Hibernate.

And if you're already familiar with the basic association mappings, you might be interested in the following posts:

- Best Practices for Many-To-One and One-To-Many Association Mappings
- How to map an association as a java.util.Map
- Why you should avoid CascadeType.REMOVE for to-many associations and what to do instead
- Hibernate Tips: How to model an association that doesn't reference primary key columns

Hibernate Tips Book



Get more recipes like this one in my book Hibernate Tips: More than 70 solutions to common Hibernate problems.

It gives you more than 70 ready-to-use recipes for topics like basic and advanced mappings, logging, Java 8 support, caching and statically and dynamically defined queries.