Hibernate Tip: Map an association to a Map

Question:

I need the elements of a mapped association as a *java.util.Map*. Do I have to implement a helper method for that or can Hibernate do it itself?

Solution:

If you want to use an entity attribute as the map key, you can use JPA's *@MapKey* annotation to map the association to a *java.util.Map*. It is supported by Hibernate and all other JPA implementations.

As you can see in the following code snippet, you can define this mapping in almost the same way as the mapping to a *java.util.List.*

```
@Entity
public class Author {

    @ManyToMany
    @JoinTable(
        name="AuthorBookGroup",
        joinColumns={@JoinColumn(name="fk_author",
            referencedColumnName="id")},
        inverseJoinColumns={@JoinColumn(name="fk_group",
            referencedColumnName="id")})
    @MapKey(name = "title")
    private Map<String, Book> books =
        new HashMap<String, Book>();
    ...
}
```

The main difference is that you need to define the key of the map with a *@MapKey* annotation. In the previous code snippet, I use the *title* attribute of the *Book* entity as the key. And you, of course, also need to use a *java.util.Map* as the attribute type instead of a *java.util.List*.

You can also use a database column which is not mapped to an entity attribute as the map key. I explained that in more detail in this post.

After you've defined the mapping, you can use the mapped association in the same way as you use any other *Map*.

```
Author a = new Author();
a.setFirstName("Thorben");
a.setLastName("Janssen");
em.persist(a);

Book b = new Book();
b.setTitle("Hibernate Tips");
b.setFormat(Format.PAPERBACK);
b.getAuthors().add(a);
em.persist(b);

a.getBooks().put(b.getTitle(), b);
```

And please keep in mind, that a *Map* replaces the existing value when you add a new value with the same key. And unfortunately, JPA and Hibernate don't support a *Collection* as the value of the *Map*.

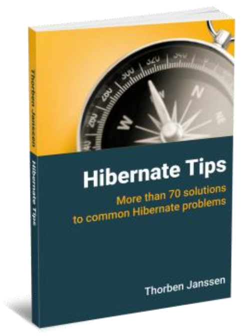
So, better make sure that the attribute, which you reference in the *@MapKey* annotation, doesn't contain any duplicate values for a given relationship. That means for this example that an Author isn't allowed to write 2 books with the same title.

Learn more

If you want to learn more about association mappings with JPA and Hibernate, you should also take a look at the following posts:

- I get into more details about mapping an associations to a *java.util.Map* and I also show you how to map multiple entities to the same map key in How to map an association as a java.util.Map
- And I wrote a broader introduction to JPA's relationship mappings 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 this post about general best practices for mapping one-to-many and many-to-one assocations.

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.