Hibernate Tip: Filter entities from a mapped association

Question:

I want to exclude some records from an association mapping. How can I filter the elements of a mapped entity association?

Solution:

You can use Hibernate’s *@Where* annotation to define an SQL clause which filters the elements of a mapped association.

Let’s take a look at an example. Books can be published in different formats, e.g. as an ebook or as a paperback. You can model that with a *Book* entity and a *Format* enum.

```
@Entity
public class Book {

	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	@Column(name = "id", updatable = false, nullable =
false)
	private Long id;

	@Enumerated(EnumType.STRING)
	private Format format;

	@ManyToMany
	@JoinTable(name = "book_author",
                joinColumns = {@JoinColumn(name = "fk_book")},
                inverseJoinColumns = {
                        @JoinColumn(name = "fk_author")})
        private List<Author> authors = new ArrayList<Author>();
        ...
}
```

```
public enum Format {
        PAPERBACK, EBOOK;
}
```

Each *Book* was written by one or more authors. So, you need a many-to-many association between the *Book* and the *Author* entity. As you can see in the code snippet, I modeled that as a typical many-to-many association on the *Book* entity.

You could, of course, do the same on the *Author* entity. But let's say you want to use different associations for ebooks and physical books. In that case, you need to define 2 associations on the *Author* entity and filter them accordingly.

As you can see in the following code snippet, you can easily do that with a *@Where* annotation. You just need to provide an SQL expression which Hibernate adds to the *WHERE* clause of the SQL statement.

Hibernate Tip: Filter entities from a mapped association

```
@Entity
public class Author {

        @Id
        @GeneratedValue(strategy = GenerationType.AUTO)
        @Column(name = "id", updatable = false,
                        nullable = false)
        private Long id;

        @ManyToMany(mappedBy = "authors")
        @Where(clause = "format = 'EBOOK'")
        private List<Book> ebooks = new ArrayList<Book>();

        @ManyToMany(mappedBy = "authors")
        @Where(clause = "format = 'PAPERBACK'")
        private List<Book> printBooks = new ArrayList<Book>();
        ...
}
```

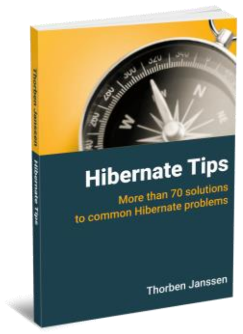
When you now load an *Author* entity and initialize the associations, Hibernate executes independent SQL statements to get the elements of the *ebooks* and *printBooks* association and uses the provided SQL snippet in the *WHERE* clause.

```
14:02:09,070 DEBUG [org.hibernate.SQL] -
    select
        author0_.id as id1_0_0_,
        author0_.firstName as firstNam2_0_0_,
        author0_.lastName as lastName3_0_0_,
        author0_.version as version4_0_0_
    from
        Author author0_
    where
        author0_.id=?
14:02:09,109 DEBUG [org.hibernate.SQL] -
    select
        ebooks0_.fk_author as fk_autho2_2_0_,
        ebooks0_.fk_book as fk_book1_2_0_,
        book1_.id as id1_1_1_,
        book1_.format as format2_1_1_,
book1_.title as title3_1_1_,
        book1_.version as version4_1_1_
    from
        book_author ebooks0_
    inner join
        Book book1_
            on ebooks0_.fk_book=book1_.id
            and (
                book1_.format = 'EBOOK'
            )
    where
        ebooks0_.fk_author=?
14:02:09,117 DEBUG [org.hibernate.SQL] -
    select
        printbooks0_.fk_author as fk_autho2_2_0_,
        printbooks0_.fk_book as fk_book1_2_0_,
        book1_.id as id1_1_1_,
        book1_.format as format2_1_1_,
book1_.title as title3_1_1_,
        book1_.version as version4_1_1_
    from
        book_author printbooks0_
    inner join
        Book book1_
            on printbooks0_.fk_book=book1_.id
            and (
                book1_.format = 'PAPERBACK'
            )
    where
printbooks0_.fk_author=?
```

Learn more

Another common use case for Hibernate’s *@Where* annotation is the implementation of a soft delete. I explain that in more detail in How to implement a soft delete with Hibernate.

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.