Hibernate Tip: Persist a List as an ElementCollection

Question:

I need to persist a simple *List* of *String*s. Do I really need to create an extra entity for it?

Solution:

Since JPA 2.0, you can use an element collection to persist a *Collection* of value types. You just need to annotate the attribute with *@ElementCollection* and the persistence provider will persist the elements of the *Collection* in an additional database table.

```
@Entity
public class Author {

	@ElementCollection
	private List<String> phoneNumbers =
                                        new ArrayList<String>();

	...
}
```

The element collection might seem easier to use than an entity with a one-to-many association. But it has one major drawback: The elements of the collection have no id and Hibernate can't address them individually.

When you add a new *Object* to the *List* or remove an existing one, Hibernate deletes all elements and inserts a new record for each item in the *List*.

Let's take a quick look at an example. The following code snippet selects an *Author* entity and adds a second *phoneNumber* to the element collection.

```
EntityManager em = emf.createEntityManager();
em.getTransaction().begin();

Author a = em.find(Author.class, 1L);
a.getPhoneNumbers().add("42424242");

em.getTransaction().commit();
em.close();
```

When you execute this code, Hibernate performs 2 SELECT statements to read the *Author* entity and the associated *phoneNumbers*. Then I add the second *phoneNumber* to the element collection. And Hibernate updates the *Author* entity, drops all associated *phoneNumbers* and inserts a new record for each *phoneNumber*.

```
14:53:13,371 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=?
14:53:13,427 DEBUG [org.hibernate.SQL] -
    select
        phonenumbe0_.Author_id as Author_i1_1_0_,
        phonenumbe0_.phoneNumbers as phoneNum2_1_0_
from
        Author_phoneNumbers phonenumbe0_
    where
        phonenumbe0_.Author_id=?
```

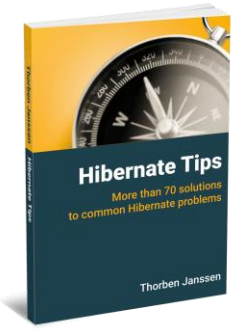
```
14:53:13,480 DEBUG [org.hibernate.SQL] -
    update
        Author
    set
        firstName=?,
        lastName=?,
        version=?
    where
        id=?
        and version=?
14:53:13,491 DEBUG [org.hibernate.SQL] -
    delete
    from
        Author_phoneNumbers
    where
        Author_id=?
14:53:13,494 DEBUG [org.hibernate.SQL] -
    insert
    into
        Author_phoneNumbers
        (Author_id, phoneNumbers)
    values
        (?, ?)
14:53:13,496 DEBUG [org.hibernate.SQL] -
    insert
    into
        Author_phoneNumbers
        (Author_id, phoneNumbers)
    values
        (?, ?)
```

As you can see, an element collection is an easy but not the most efficient option to store a list of value types in the database. You should, therefore, only use it for very small collections so that Hibernate doesn't perform too many SQL statements. In all other cases, a one-to-many association is the better approach.

Learn more

Before you use an element collection, you should take a look at the mapping of many-to-one associations. JPA and Hibernate make them very easy to use and Hibernate can handle them more efficiently.

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.