Hibernate Tip: How to fetch associations in batches

Question:

I need to initialize the associations of multiple entities. Is there a way to tell Hibernate to fetch the associated entities for multiple entities?

Solution:

The best way to load associated entities is to use a JOIN FETCH clause or a *@NamedEntityGraph.* These tells Hibernate to fetch the associated entities with the initial query. So, Hibernate joins the mapped database tables and loads all the data with only 1 query.

If you can't use these options or if the created cartesian product gets too big, you can also annotate the association with Hibernate's *@BatchSize* annotation. It tells Hibernate to initialize the relationships of multiple *Author* entities in a batch.

```
@Entity
public class Author {

    @ManyToMany(mappedBy="authors")
    @BatchSize(size = 5)
    private List<Book> books = new ArrayList<Book>();

    ...
}
```

When you use this mapping in the following test case, Hibernate will initialize the *books* association for up to 5 *Author* entities in one batch.

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

List<Author> authors = em.createQuery("SELECT a FROM Author a", Author.class).getResultList();
for (Author a : authors) {
    log.info("Author with id "+a.getId());
    a.getBooks().size();
}

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

It executes 1 query to load the 6 *Author* entities that exist in my test database. Then the test case starts to iterate through the *List* of *Author* entities. When it calls the *getBooks*() method of the first *Author* entity, Hibernate initializes the association of the first 5 *Author* entities returned by the previous query. As you can see in the log output, it uses an *IN* clause to efficiently reference the primary keys of the 5 entities. And when the test case calls the getBooks() method of the 6th and last *Author* entity, Hibernate performs another query to fetch the associated *Book* entities.

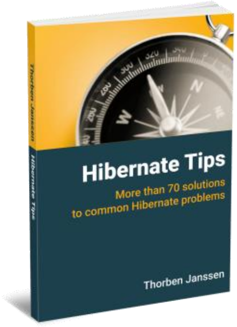
```
18:23:34,974 DEBUG [org.hibernate.SQL] -
    select
        author0_.id as id1_0_,
        author0_.firstName as firstNam2_0_,
        author0_.lastName as lastName3_0_,
        author0_.version as version4_0_
    from
        Author author0_
18:23:35,019 INFO
[org.thoughts.on.java.model.TestBatchSize] - Author with id 1
18:23:35,024 DEBUG [org.hibernate.SQL] –
    select
        books0_.authorId as authorId2_2_2_,
        books0_.bookId as bookId1_2_2_,
        book1_.id as id1_1_0_,
        book1_.publisherid as publishe5_1_0_,
        book1_.publishingDate as publishi2_1_0_,
        book1_.title as title3_1_0_,
        book1_.version as version4_1_0_,
        publisher2_.id as id1_3_1_,
        publisher2_.name as name2_3_1_,
        publisher2_.version as version3_3_1_
    from
        BookAuthor books0_
```

```
inner join
        Book book1_
            on books0_.bookId=book1_.id
    left outer join
        Publisher publisher2_
            on book1_.publisherid=publisher2_.id
    where
        books0_.authorId in (
            ?, ?, ?, ?, ?
        )
18:23:35,040 INFO
[org.thoughts.on.java.model.TestBatchSize] - Author with id 2
18:23:35,041 INFO
[org.thoughts.on.java.model.TestBatchSize] - Author with id 3
18:23:35,041 INFO
[org.thoughts.on.java.model.TestBatchSize] - Author with id 4
18:23:35,041 INFO
[org.thoughts.on.java.model.TestBatchSize] - Author with id 5
18:23:35,041 INFO
[org.thoughts.on.java.model.TestBatchSize] - Author with id 6
18:23:35,041 DEBUG [org.hibernate.SQL] –
select
        books0_.authorId as authorId2_2_2_,
        books0_.bookId as bookId1_2_2_,
        book1_.id as id1_1_0_,
    book1_.publisherid as publishe5_1_0_,
    book1_.publishingDate as publishi2_1_0_,
    book1_.title as title3_1_0_,
    book1_.version as version4_1_0_,
    publisher2_.id as id1_3_1_,
    publisher2_.name as name2_3_1_,
    publisher2_.version as version3_3_1_
from
    BookAuthor books0_
inner join
    Book book1_
        on books0_.bookId=book1_.id
left outer join
    Publisher publisher2_
        on book1_.publisherid=publisher2_.id
where
    books0_.authorId=?
```

Learn more

Fetching the associated entities in batches is a lot better than loading them one by one. But in most situations, it would be even better to initialize the association with the query that loads the entity. JPA and Hibernate provide different options to do that. I show you 5 of them in 5 ways to initialize lazy associations and when to use them.

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.