## Advantages of the *stream()* method

In the beginning, it looks like a small improvement that makes your code a little less clunky. You already can take the List of query results and call its *stream()* method to get a Stream representation.

But that is not the most efficient approach. Hibernate will get all the selected entities from the database, store them in memory and put them into a List. You then call the *stream()* method and process the results one by one.

But if you're working on a huge result set, you better scroll through the result records and fetch them in smaller chunks. You're already familiar with that approach if you've used JDBC result sets or Hibernate's *ScrollableResult.*

The Hibernate team used the existing *scroll()* method and the *ScrollableResult* to implement implement the new *stream()* method.

## How to use the *stream()* method

The *stream()* method is part of the Query interface and you can, therefore, use it with all kinds of queries and projections.

### Entities

Entities are the most common projection with Hibernate, and you can use them in a Stream in any way you like.

```
Stream<Book> books = session.createQuery(
        "SELECT b FROM Book b", Book.class).stream();
books.map(b -> b.getTitle() + " was published on " +
                b.getPublishingDate())
        .forEach(m -> log.info(m));
```

## Scalar Values

Up to now, scalar values were not a very popular projection because it returns a *List* of *Object[]*. You then have to implement a loop to go through all *Object[]*s and cast its elements to their specific types. That gets a lot easier with Streams.

```
Stream<Object[]> books = session.createNativeQuery(
        "SELECT b.title, b.publishingDate FROM book b").stream();
books.map(b -> new BookValue((String)b[0], (Date)b[1]))
        .map(b -> b.getTitle() + " was published on " +
                b.getPublishingDate())
```

## POJOs

POJOs or similar projections can be easily created with a constructor expression, as you can see in the following code snippet.

Unfortunately, there seems to be a bug (HHH-11029) in Hibernate 5.2.2 so that these projections don't work with Streams. Instead of mapping the *BookValues* to *String*s and writing them to the log file, the following code snippet throws a *ClassCastException*.

```
Stream<BookValue> books = session.createQuery(
        "SELECT new org.thoughts.on.java.model.BookValue(b.title,
                b.publishingDate) FROM Book b",
        BookValue.class).stream();
books.map(b -> b.getTitle() + " was published on " +
                b.getPublishingDate()).forEach(m -> log.info(m));
```