Ordering vs Sorting - What should you use?

The difference between Sorting and Ordering

Sorting and ordering seem to do the same. They bring the associated entities into a defined order. But they differ in the approach.

When use sorting, Hibernate will load the associated Book entities from the database and use a Java Comparator to sort them in memory. That is not a good approach for huge Sets of entities.

Ordering uses an ORDER BY clause in the SQL statement to retrieve the entities in the defined order. Database handles these kinds of operations very efficiently. So this approach is a lot better for huge associations.

Sorting

Sorting requires you to either use a *SortedSet* or a *SortedMap* as an attribute type. Both require that all elements have to implement Comparable and stores them in a defined order.

You can choose between 2 approaches to sort the elements of your association. You can either use natural, or you can provide a custom *Comparator*.

Natural Sorting

You can tell Hibernate to use natural sorting by annotating the association attribute with *@SortNatural*. This approach uses the *Comparable* implementation of the related entities.

Ordering vs Sorting - What should you use?

```
@Entity
public class Author implements Comparable {
        …

        @ManyToMany(mappedBy = “authors”)
        @SortNatural
        private SortedSet books = new TreeSet();


        …
}
```

```
@Entity
public class Book implements Comparable {
        …

        private String title;


        …


        @Override
        public int compareTo(Book o) {
                return title.compareTo(o.getTitle());
        }
}
```

Custom *Comparator*

When you don't want to use the *Comparable* implementation of your entities for your sorting, you can use the *@SortComparator* annotation to provide a custom *Comparator*.

```
@Entity
public class Author implements Comparable {
        …

        @ManyToMany(mappedBy = “authors”)
        @SortComparator(SortById.class)
        private SortedSet<Book> books = new
TreeSet<Book>();


        …
}
```

```
public class SortById implements Comparator {

        @Override
        public int compare(Book o1, Book o2) {
                return o1.getId().compareTo(o2.getId());
        }
}
```

Ordering

The ordering feature is defined by the JPA specification and does what most developers expect. It uses an ORDER BY clause in the SQL query to retrieve the associated entities in a specific order.

In contrast to the sorting feature, you can use ordering on *List*s and your entities don't have to implement the *Comparable* interface. You shouldn't use it with *SortedSet*s to avoid an additional sorting of the already ordered entities.

You can define the ORDER BY clause with JPA's *@OrderBy* annotation. If you don't provide any additional information, Hibernate will order the associated entities by their primary key. If you want to retrieve them in a different order, you can define an *orderby_list* with one or more *orderby_items* as follows:

*orderby_list::= orderby_item [,orderby_item]**

orderby_item::= [property_or_field_name] [ASC | DESC]

```
@Entity
public class Author {
        …

        @ManyToMany(mappedBy = “authors”)
        @OrderBy(“title ASC”)
        private List books = new ArrayList();

        …
}
```

Sorting or Ordering? What should you use?

After you’ve read this post, I hope it became obvious that you should order your entities and not sort them. The 2 main reasons are performance and portability:

The ordering of result set records is a typical task for relational databases. They are highly optimized for it, and you can easily benefit from that by annotating your entity association with an @OrderBy annotation.

The ordering feature is defined by the JPA specification so you can use it with every compliant JPA implementation, like Hibernate, EclipseLink or OpenJPA.