Sorting Query Results with Spring Data

Last modified: November 27, 2018

by [baeldung](https://www.baeldung.com/author/baeldung/)

* [**Spring Data**](https://www.baeldung.com/category/persistence/spring-persistence/spring-data/)

**I just announced the new *Learn Spring*course, focused on the fundamentals of Spring 5 and Spring Boot 2:**

[**>> CHECK OUT THE COURSE**](https://www.baeldung.com/ls-course-start)

**1. Introduction**

In this tutorial, we’re going to **learn how to sort query results** with [Spring Data](https://www.baeldung.com/spring-data).

First, we’ll take a look at the schema of the data that want to query and sort.

And then, we’ll dive right into how to achieve that Spring Data.

Let’s get started!

**2. The Test Data**

Below we have some example data. Although we have represented it here as a table we could use any one of the databases supported by Spring Data to persist it.

The question we want to answer is, “Who is occupying which seat on the airline?” but to make that more user-friendly we want to sort by seat number.

|  |  |  |
| --- | --- | --- |
| **First Name** | **Last Name** | **Seat Number** |
| Jill | Smith | 50 |
| Eve | Jackson | 94 |
| Fred | Bloggs | 22 |
| Ricki | Bobbie | 36 |
| Siya | Kolisi | 85 |

**3. Domain**

To create a [Spring Data Repository](https://docs.spring.io/spring-data/data-commons/docs/current/reference/html/#repositories.core-concepts) we need to provide a domain class as well as an id type.

Here we’ve modeled our passenger as a JPA entity, but we could just as easily have modeled it as a MongoDB document or any other model abstraction:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | @Entity  class Passenger {        @Id      @GeneratedValue      @Column(nullable = false)      private Long id;        @Basic(optional = false)      @Column(nullable = false)      private String firstName;        @Basic(optional = false)      @Column(nullable = false)      private String lastName;        @Basic(optional = false)      @Column(nullable = false)      private int seatNumber;        // constructor, getters etc.  } |

**4. Sorting with Spring Data**

We have a few different options at our disposal for sorting with Spring Data.

**4.1. Sorting with the *OrderBy* Method Keyword**

One option would be to use Spring Data’s method derivation whereby the query is generated from the method name and signature.

**All we need to do here to sort our data is include the keyword *OrderBy* in our method name** along with the property name(s) and direction (Asc or Dsc) by which we want to sort.

We can use this convention to create a query that returns our passengers in ascending order by seat number:

|  |  |
| --- | --- |
| 1  2  3  4 | interface PassengerRepository extends JpaRepository<Passenger, Long> {        List<Passenger> findByOrderBySeatNumberAsc();  } |

We can also combine this keyword with all the standard Spring Data method names.

Let’s see an example of a method that finds passengers by last name *and* orders by seat number:

|  |  |
| --- | --- |
| 1 | List<Passenger> findByLastNameOrderBySeatNumberAsc(String lastName); |

**4.2. Sorting with a *Sort* Parameter**

**Our second option is to include a**[***Sort***](https://docs.spring.io/spring-data/commons/docs/current/api/org/springframework/data/domain/Sort.html)**parameter** specifying the property name(s) and direction by which we want to sort:

|  |  |
| --- | --- |
| 1 | List<Passenger> passengers = repository.findAll(Sort.by(Sort.Direction.ASC, "seatNumber")); |

In this case, we’re using the *findAll()* method and adding the *Sort* option when calling it.

We can also add this parameter to a new method definition:

|  |  |
| --- | --- |
| 1 | List<Passenger> findByLastName(String lastName, Sort sort); |

Finally, if perhaps we are paging, we can specify our sort in a [*Pageable*](https://docs.spring.io/spring-data/commons/docs/current/api/org/springframework/data/domain/Pageable.html) object:

|  |  |
| --- | --- |
| 1 | Page<Passenger> page = repository.findAll(PageRequest.of(0, 1, Sort.by(Sort.Direction.ASC, "seatNumber"))); |

**5. Conclusion**

We have two easy options for sorting data with Spring Data through method derivation using the *OrderBy*keyword or using the [*Sort*](https://docs.spring.io/spring-data/commons/docs/current/api/org/springframework/data/domain/Sort.html) object as a method parameter.

As always, you can find the code [over on GitHub](https://github.com/eugenp/tutorials/tree/master/persistence-modules/spring-data-jpa).