Spring Data JPA Pagination and Sorting

By Ramesh Fadatare (Java Guides)

Pagination and Sorting Overview

As you know, pagination allows the users to see a small portion of data at a time (a page), and sorting allows the users to view the data in a more organized way.

Paging and sorting is mostly required when we are displaying domain data in tabular format in UI.

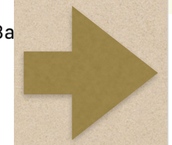
Pagination consist of two fields – page size and page number.

Sorting consist of two fields - sortBy (single or multiple fields) and sortDir (sort direction can be ASC or DESC)

Spring Data JPA Pagination and Sorting

```java
public interface ProductRepository extends JpaRepository<Product, Long> {

}
```

```java
@NoRepositoryBean
public interface JpaRepository<T, ID> extends PagingAndSortingRepository<
    List<T> findAll();

    List<T> findAll(Sort sort);

    List<T> findAllById(Iterable<ID> ids);

    <S extends T> List<S> saveAll(Iterable<S> entities);

    void flush();

    <S extends T> S saveAndFlush(S entity);

    <S extends T> List<S> saveAllAndFlush(Iterable<S> entities);

    Deprecated
    @Deprecated
    default void deleteInBatch(Iterable<T> entities) { this.deleteAllInBa

    void deleteAllInBatch(Iterable<T> entities);

    void deleteAllByIdInBatch(Iterable<ID> ids);

    void deleteAllInBatch();

    Deprecated
    @Deprecated
    T getOne(ID id);

    T getById(ID id);

    <S extends T> List<S> findAll(Example<S> example);

    <S extends T> List<S> findAll(Example<S> example, Sort sort);
}
```



```java
@NoRepositoryBean
public interface PagingAndSortingRepository<T, ID> extends CrudRepository<T, ID> {
    Iterable<T> findAll(Sort sort);

    Page<T> findAll(Pageable pageable);
}
```

Spring Data JPA Pagination

To apply only pagination in result set, we need to create Pageable object without any Sort information and pass Pageable object to findAll() method:

```
int pageNo = 0;
int pageSize = 3;
Pageable pageable = PageRequest.of(pageNo, pageSize);
Page<Product> page = productRepository.findAll(pageable);

List<Product> productList = page.getContent();
productList.forEach((p) ->{
    System.out.println(p);
});

int totalPages = page.getTotalPages();
long totalItems = page.getTotalElements();
int size = page.getSize();
int numberOfElements = page.getNumberOfElements();
boolean isLast = page.isLast();
boolean isfirst = page.isFirst();
```

Spring Data JPA Sorting

To apply only sorting in result set, we need create Sort object and pass it to findAll() method

```
String sortBy = "name";
String sortDir = "desc";
Sort sort = sortDir.equalsIgnoreCase(Sort.Direction.ASC.name()) ? Sort.by(sortBy).ascending()
        : Sort.by(sortBy).descending();

List<Product> sortedProducts = productRepository.findAll(sort);
```

Spring Data JPA Sorting By Multiple Fields

If we wish to apply sorting on multiple columns or group by sort, then that is also possible by creating Sort using simple builder pattern steps.

```
String sortBy = "name";
String sortByDesc = "description";
// default sorting ascending order: DEFAULT_DIRECTION = Sort.Direction.ASC;
String sortDir = "desc";
Sort sortName = sortDir.equalsIgnoreCase(Sort.Direction.ASC.name()) ? Sort.by(sortBy).ascending()
        : Sort.by(sortBy).descending();

Sort sortDesc = sortDir.equalsIgnoreCase(Sort.Direction.ASC.name()) ? Sort.by(sortBy).ascending()
        : Sort.by(sortBy).descending();

// sorting on multiple columns or group by sort,
Sort groupBySort = sortName.and(sortDesc);

List<Product> sortedProducts = productRepository.findAll(groupBySort);
```

Spring Data JPA
Pagination and Sorting Together

```
String sortBy = "name";
// default sorting ascending order: DEFAULT_DIRECTION = Sort.Direction.ASC;
String sortDir = "asc";
int pageNo = 0;
int pageSize = 3;

Sort sort = sortDir.equalsIgnoreCase(Sort.Direction.ASC.name()) ? Sort.by(sortBy).ascending()
        : Sort.by(sortBy).descending();

Pageable pageable1 = PageRequest.of(pageNo, pageSize, sort);

Page<Product> listOfProducts = productRepository.findAll(pageable1);
```