Grupo 29

# Item 4. Paginated repositories

Manual de instrucciones

Khawla Al Asfar

Juan Carlos Utrilla Martín

Juan Rodríguez Dueñas

Yassine Taziny

José Manuel Lara Morilla

# Contenido

# 1. Introducción

Existen diferentes formas de diseñar un sistema de paginación desde la perspectiva de una experiencia de usuario: podemos tener links del tipo: *first*/*previous*/*next*/*last*, y mostrar un conjunto limitado de páginas en la página actual.

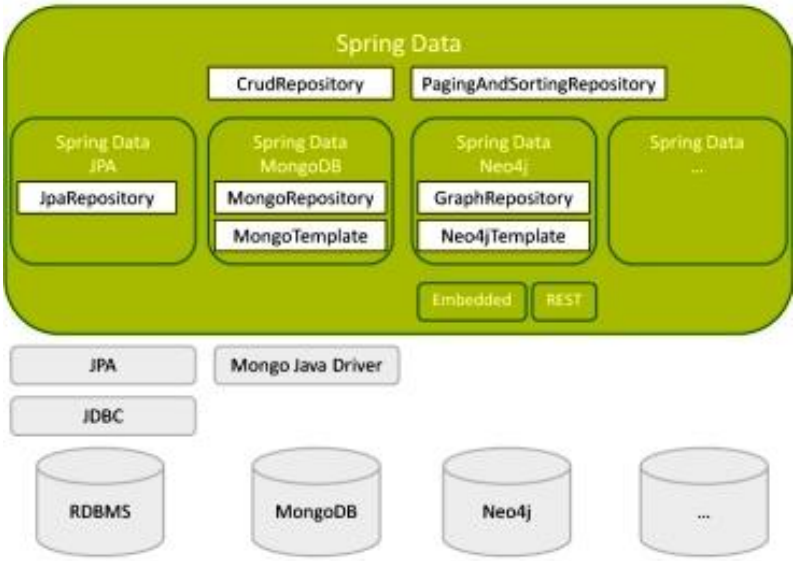
Hasta lo que llevamos realizado de curso en la asignatura, cada vez que realizamos un listado de alguna de las entidades de nuestro modelo de dominio, obtenemos la lista completa de objetos de la base de datos y luego el componente los filtra seleccionando la página apropiada y descartando el resto. Esta es una solución muy ineficiente.

Para construir repositorios eficientes en este ítem, implementaremos un repositorio paginado o *Paginated Repository* que nos permitirá elegir qué páginas mostrar y cuantos objetos queremos visualizar en cada página.

# 2. Entorno tecnológico

**Spring Data** es un proyecto de Spring cuyo objetivo es proveer un framework de la capa de capa de acceso a datos (DAO en inglés) que unifica y facilita el acceso a distintos tipos de persistencia, tanto a bases de datos relacionales como las de tipo NoSQL y simplifica el trabajo a la hora de realizar implementaciones concretas.

Proporciona una definición para implementar repositorios compatibles bajo la especificación JPA mediante interfaces genéricas para estos aspectos (CrudRepository, PagingAndSortingRepository) e implementaciones específicas para cada tipo de persistencia.



# 3. Configuración del proyecto

Antes de comenzar con la implementación del código, debemos incluir las dependencias delas librerías con las que vamos a trabajar utilizando Maven. Para ello, añadiremos en el pom.xml las siguientes dependencias:

```
<!-- Spring Framework - Data -->
	<dependency>
		<groupId>org.springframework.data</groupId>
		<artifactId>spring-data-jpa</artifactId>
		<version>1.4.3.RELEASE</version>
	</dependency>

<!-- Spring Framework -->
	<dependency>
		<groupId>org.springframework</groupId>
		<artifactId>spring-orm</artifactId>
		<version>4.0.0.RELEASE</version>
	</dependency>
```

# 4. Implementación

Para la construcción de nuestro repositorio paginado implementaremos código en repositorios y servicios del dominio de *Trip*.

## 4.1. Repositorio

La interfaz a la que tenemos que extender nuestro repositorio de *Trip* sería *PaginatedAndSortingRepostory*, interfaz que nos permitirá realizar paginación y ordenación sobre una colección de *Trip*.

Aquí podríamos realizar dos soluciones:

1. Crear un nuevo repositorio paginado para *Trip* (*PaginatedTripRepository*).
2. Duplicar aquellos métodos que devuelven una colección de *Trip* en nuestro repositorio *TripRespository*.

Finalmente nos hemos decantado por la segunda opción debido a que la interfaz *JpaRespository* ya extiende de la interfaz *PaginatedAndSortingRepository*, por lo tanto, no haría falta implementar un nuevo repositorio que extienda de dicha interfaz.

Después de esta explicación procederemos a mostrar la implementación realizada sobre la query *listTripPerManager*, la cual permitirá recuperar los viajes asociados/creados por un manager.

```
@Query("select t from Trip t where t.manager.id = ?1")
public Collection<Trip> listTripPerManager(int managerId);

@Query("select t from Trip t where t.manager.id = ?1")
public Page<Trip> listTripPerManager(int managerId, Pageable page);
```

Como dijimos anteriormente, hemos optado por duplicar el método *listTripPerManager*, realizando unas modificaciones necesarias para que esta duplicación funcione correctamente:

- Se le añade como parámetro un tipo *Pageable*, el cual es una interfaz abstracta para la paginación de la información recuperada de una base de datos.
- Se ha modificado el tipo que devuelve el método a un tipo *Page<Trip>*, que es una sub-lista de una lista de objetos. Esta sub-lista permite almacenar información sobre su posición dentro de la lista completa.

## 4.2. Servicio

Continuando con la implementación del servicio de *Trip* (*TripService*), hemos optado por realizar el mismo proceso explicado en *TripRespository*:

```
public Collection<Trip> listTripPerManager(final int managerId) {

    Collection<Trip> res = null;
    res = this.tripRepository.listTripPerManager(managerId);
    return res;
}

public Collection<Trip> listTripPerManager(final int managerId, final int pageNumber, final int numberTripsPerPage) {

    Collection<Trip> res = null;
    final Pageable page = new PageRequest(pageNumber, numberTripsPerPage);
    final Page<Trip> pageResult = this.tripRepository.listTripPerManager(managerId, page);
    res = pageResult.getContent();
    return res;

}
```

Hemos procedido a duplicar el método *listTripPerManager* donde el primero llama el método del repositorio *listTripPerManager(managerId)* y el segundo llama al método del repositorio *listTripPerManager(managerId, page)*.

Para el segundo método se le incluye como parámetros *pageNumber* y *numberTripsPerPage*. El primero especifica los elementos a mostrar de que página (0 representa la primera página, 1 representa la segunda…), y la segunda especifica cuantos elementos de *Trip* deben de aparecer en cada página.

Una vez especificados los parámetros necesarios para la paginación continuamos la explicación sobre cómo trabaja el método:

- Creamos una variable de tipo *Pageable* a través del constructor de *PageRequest*, pasándole como parámetros los parámetros mencionado anteriormente, *pageNumber* y *numberTripsPerPage*.
- Llamamos al método *listPerManager(managerId, page)* del repositorio pasándole como parámetros managerId y la variable de tipo *Pageable* creado en el paso anterior, *page*. Esto devolverá un tipo *Page<Trip>*.
- Se transforma este tipo a *Collection<Trip>* a través del método *getContent()* aplicado sobre la variable *pageResult* de tipo *Page<Trip>*.
- Devolver esta *Collection<Trip>*.

## 5. Pruebas

Para probar este método hemos realizado una modificación sobre el método *listTripPerManager()* de la clase *TripServiceTest*.

```
@Test
public void listTripPerManager() {

    super.authenticate("manager1");

    Collection<Trip> trips = null;

    trips = new ArrayList<Trip>();
    int managerId;
    final int pageNumber = 0;
    final int numberTripsPerPage = 4;

    managerId = this.mangerService.findByPrincipal().getId();
    trips = this.tripService.listTripPerManager(managerId, pageNumber, numberTripsPerPage);

    System.out.println("*********** Retrieved list of trips from page " + pageNumber + " with " + numberTripsPerPage + " trips each page ***********");
    for (final Trip trip : trips)
        System.out.println("Trip title = " + trip.getTitle());

    super.authenticate("manager1");
}
```

- Añadimos dos variables que nos permitirán definir el número de la página y el número de viajes por página a mostrar: *pageNumber* y *numberTripsPerPage*.
- Llamamos al método *listTripPerManager(managerId, pageNumber, numberTripsPerPage)*, el cual nos devolverá una colección de longitud *numberTripsPerPage* o menor (si dicho manager no posee tantos viajes para mostrarlo en la página especificada).
- Mostramos a través de la consola los viajes recuperados de la base de datos.

Si ejecutamos el ejemplo de la imagen superior obtendremos lo siguiente a través de la consola:

```
*********** Retrieved list of trips from page 0 with 4 trips each page ***********
Trip title = Trip 1
Trip title = Trip 2
Trip title = Trip 3
Trip title = Trip 4
```

Si modificamos el número de la página a la 2 (*pageNumber* = 1) obtendremos únicamente 3 viajes debido a que únicamente el *manager1* posee asociados los viajes del 1 al 7, y por lo tanto no es capaz de recuperar los 4 siguientes viajes:

```
*********** Retrieved list of trips from page 1 with 4 trips each page ***********
Trip title = Trip 5
Trip title = Trip 6
Trip title = Trip 7
```

# 6. Referencias


- **Pagination and Sorting With Spring Data JPA:** http://springinpractice.com/2012/05/11/pagination-and-sorting-with-spring-data-jpa
- **Interface Pageable:** https://docs.spring.io/spring-data/commons/docs/current/api/org/springframework/data/domain/Pageable.html
- **Interface Page<T>:** https://docs.spring.io/spring-data/commons/docs/current/api/org/springframework/data/domain/Page.html
- **Working with Spring Data Repositories:** https://docs.spring.io/spring-data/data-commons/docs/1.6.1.RELEASE/reference/html/repositories.html