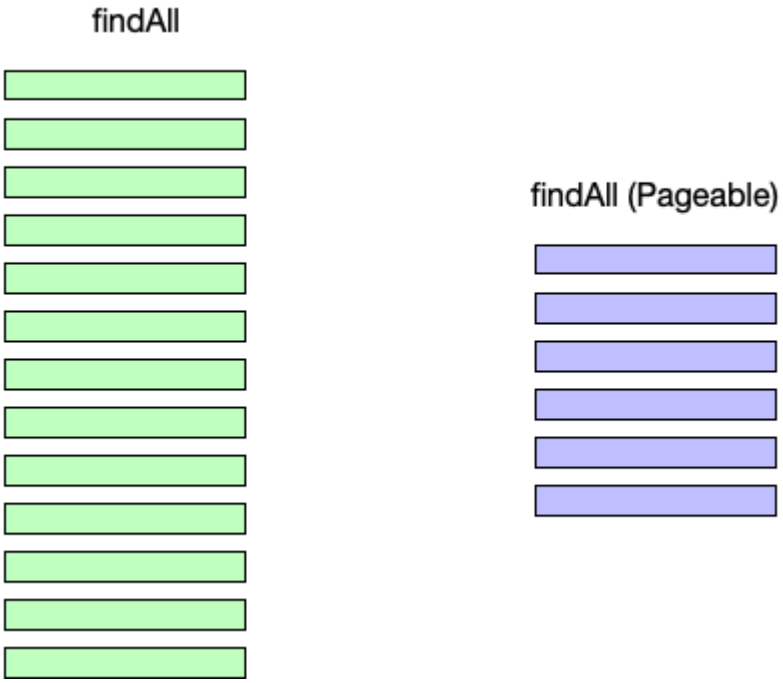
La Spring Data Paginación es una técnica muy útil cuando trabajamos con aplicaciones Java que consultan muchos registros en una base de datos. En lugar de traer todas las entidades de golpe, podemos dividir los resultados en páginas más pequeñas. En este artículo veremos cómo aplicar paginación en repositorios usando una entidad `Persona`, centrándonos en lo que pueden devolver los métodos: `Page`, `List` y `Slice`.

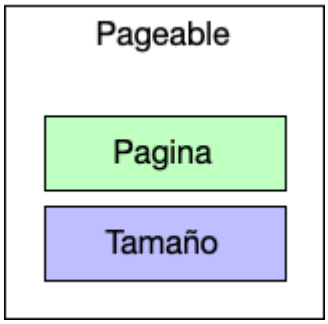


Spring Data Paginación en repositorios Persona

La Spring Data Paginación se aplica principalmente en la capa de repositorios, que es donde definimos cómo acceder a los datos. Imaginemos que tenemos una entidad llamada `Persona`, con campos como `id`, `nombre`, `apellido` y `email`. Si nuestra tabla tiene miles de personas, consultar todos los registros con un simple `findAll()` puede afectar al rendimiento de la aplicación.

Con Spring Data JPA, podemos usar la interfaz `Pageable` para indicar qué página queremos consultar, cuántos elementos debe tener y cómo queremos ordenar los resultados. Esto hace que la consulta sea más eficiente, porque la base de datos solo devuelve una parte concreta

de los registros. Además, Spring Data genera automáticamente muchas consultas sin necesidad de escribir SQL manualmente.



Un repositorio típico para trabajar con la entidad `Persona` puede extender de `JpaRepository`. A partir de ahí, podemos crear métodos que reciban un `Pageable` y devuelvan un objeto `Page`, `Slice` o incluso una `List`, dependiendo de las necesidades del caso.

```
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;

public interface PersonaRepository extends JpaRepository {

    Page findByApellido(String apellido, Pageable pageable);

}
```

En este ejemplo, el método `findByApellido` busca personas por apellido y devuelve un `Page`. Esto significa que no solo obtenemos la lista de personas, sino también información adicional como el número total de páginas, el total de elementos y si existe una página siguiente.

Page, List y Slice al consultar entidades Persona

Cuando hablamos de Spring Data Paginación, uno de los puntos más importantes es entender la diferencia entre `Page`, `List` y `Slice`. Aunque los tres pueden representar resultados de una consulta, no ofrecen la misma información. Elegir bien el tipo de retorno ayuda a mejorar el rendimiento y a mantener el código más claro.

`Page` es la opción más completa. Devuelve los datos de la página actual y también información global de la consulta, como el total de registros encontrados. Es ideal cuando queremos construir una interfaz con paginación clásica, por ejemplo: “Página 1 de 10”. Sin embargo, para calcular el total, Spring Data suele ejecutar una consulta adicional de conteo.

```
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Sort;

public class PersonaService {

    private final PersonaRepository personaRepository;

    public PersonaService(PersonaRepository personaRepository) {
        this.personaRepository = personaRepository;
    }

    public Page buscarPersonasPorApellido(String apellido) {
        PageRequest pageable = PageRequest.of(
            0,
            10,
            Sort.by("nombre").ascending()
        );
```

```
        return personaRepository.findByApellido(apellido, pageable);
    }
}
```

Por otro lado, `Slice` es útil cuando solo necesitamos saber si existe una siguiente página, pero no nos interesa conocer el total de registros. Esto puede ser más eficiente en listados grandes o scroll infinito. También podemos usar `List` junto con `Pageable` si únicamente queremos los datos, sin metadatos de paginación.

```
import org.springframework.data.domain.Slice;
import org.springframework.data.domain.Pageable;
import java.util.List;

public interface PersonaRepository extends JpaRepository {

    Slice findByNombreContaining(String nombre, Pageable pageable);

    List findByEmailContaining(String email, Pageable pageable);

}
```

En resumen, si necesitamos una paginación completa, usamos `Page`. Si queremos una navegación más ligera, usamos `Slice`. Y si solo nos interesa limitar resultados sin información extra, usamos `List`. En todos los casos, la Spring Data Paginación permite que las consultas sobre entidades `Persona` sean más ordenadas, eficientes y fáciles de mantener.

La Spring Data Paginación es una herramienta esencial para trabajar correctamente con grandes cantidades de datos en aplicaciones Java. Usando repositorios con la entidad `Persona`, podemos devolver `Page`, `Slice` o `List` según el nivel de información que necesitemos. Aplicar paginación desde la capa de repositorios mejora el rendimiento, facilita el desarrollo y permite construir APIs más limpias y escalables.

- Java Stream Sum y Business Objects
- ¿Que es el Principio KISS?
- Java EE 6 uso de filtros dinámicos
- React State un concepto clave de React
- JPA Polymorphic Query