



DGTIC UNAM
DIRECCIÓN GENERAL DE CÓMPUTO Y
DE TECNOLOGÍAS DE INFORMACIÓN
Y COMUNICACIÓN



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
DIRECCIÓN GENERAL DE CÓMPUTO Y DE TECNOLOGÍAS
DE INFORMACIÓN Y COMUNICACIÓN



DIPLOMADO

Desarrollo de sistemas con tecnología Java

Módulo 8

Persistencia con Spring Data

Dr. Omar Mendoza González

omarmendoza564@aragon.unam.mx

Paging And Sorting



Paginaciones y ordenaciones

AlumnoPagingAndSortingRepository

AlumnoRepository

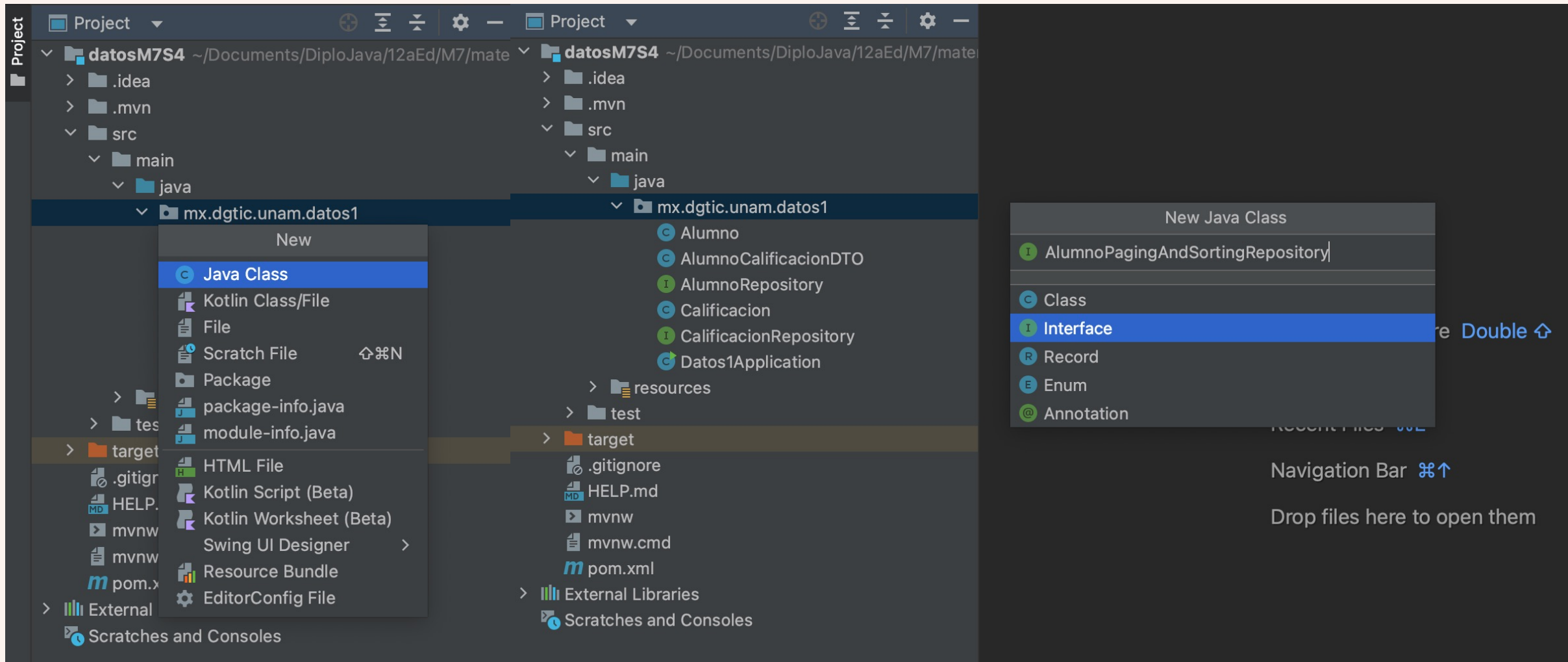
findAll(Sort ordenacion)

AlumnoPagingAndSortingRepository

```
public interface AlumnoPagingAndSortingRepository  
extends
```

```
PagingAndSortingRepository<Alumno, String> {  
}
```

AlumnoRepository



The screenshot displays the IntelliJ IDEA IDE interface. The left sidebar shows the project structure for 'datosM7S4'. The 'src/main/java' directory is expanded, showing the package 'mx.dgtic.unam.datos1'. A context menu is open over this package, listing options like 'Java Class', 'Kotlin Class/File', 'File', 'Scratch File', 'Package', 'package-info.java', 'module-info.java', 'HTML File', 'Kotlin Script (Beta)', 'Kotlin Worksheet (Beta)', 'Swing UI Designer', 'Resource Bundle', and 'EditorConfig File'. The 'New Java Class' dialog is also visible, showing a list of class types: 'Alumno', 'AlumnoCalificacionDTO', 'AlumnoRepository', 'Calificacion', 'CalificacionRepository', 'Datos1Application', 'Class', 'Interface', 'Record', 'Enum', and 'Annotation'. The 'Interface' option is selected. The right sidebar shows the 'Navigation Bar' with a search icon and a 'Drop files here to open them' message.

Project Structure:

- datosM7S4
 - .idea
 - .mvn
 - src
 - main
 - java
 - mx.dgtic.unam.datos1
 - Alumno
 - AlumnoCalificacionDTO
 - AlumnoRepository
 - Calificacion
 - CalificacionRepository
 - Datos1Application

New Java Class Dialog:

- AlumnoPagingAndSortingRepository
- Class
- Interface
- Record
- Enum
- Annotation

Navigation Bar: Search icon, Double click icon

Drop files here to open them

AlumnoPagingAndSortingRepository

```
1 package mx.dgtic.unam.datos1;
2
3 import ...
4
5
6
7
8
9
10 @SpringBootTest
11 public class Datos410ApplicationPagingAndSorting {
12     1 usage
13     private static final String ALUMNO = "OMAR MENDOZA GONZALEZ";
14     1 usage
15     @Autowired
16     AlumnoPagingAndSortingRepository repositorioAlumno;
17
18     @Test
19     void buscarTodosPageableTest() {
20         System.out.println(ALUMNO);
21         System.out.println("findAlumnoPageable");
22         Pageable pagina = PageRequest.of( page: 0, size: 5, Sort.by( ...properties: "nombre").descending());
23         repositorioAlumno.fin
24     }
25 }
```

findAllByEstatura(double estatura, Pageable pag... List<Alumno>
findByOrderByNombre() List<Alumno>
findAll(Sort sort) Iterable<Alumno>
findByOrderByNombreDescPaternoDesc() List<Alumno>
findAll(Pageable pageable) Page<Alumno>

Press ^ to choose the selected (or first) suggestion and insert a dot afterwards Next Tip

Ordenamiento

Sort

By

nombre

Ascending

Ordenamiento anidado

Sort

By

nombre

Ascending

And

Sort

By

paterno

Descending

Ordenamiento anidado

```
@Test
void buscarTodosOrdenadosParametroTest() {
    System.out.println(ALUMNO);
    System.out.println("Sort.By.nombre");
    repositorioAlumno.findAll(Sort.by(...properties: "nombre")).forEach(System.out::println);

    System.out.println("Sort.By.nombre.descending()");
    repositorioAlumno.findAll(Sort.by(...properties: "nombre").descending()).forEach(System.out::println);

    System.out.println("Sort.By.estatura.descending()");
    repositorioAlumno.findAll(Sort.by(...properties: "estatura").descending()).forEach(System.out::println);

    System.out.println("Sort.By.nombre.ascending().paterno.descending()");
    repositorioAlumno.findAll(
        Sort.by(...properties: "nombre").ascending()
            .and(Sort.by(...properties: "paterno").descending())
    )
        .forEach(System.out::println);
}
```

PagingAndSortingRepository

Es una interfaz en **Spring Data JPA** que proporciona funcionalidades adicionales para

Paginación

Dividir grandes conjuntos de resultados en páginas manejables.

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

Ordenamiento

Ordenar los resultados según uno o varios criterios.

```
Iterable<T> findAll(Sort  
sort)
```

PagingAndSortingRepository

- Metodos
 - **findAll(Pageable pageable)**
 - Obtiene una página de resultados
 - **Pageable**: Contiene información sobre la página y el tamaño
 - **findAll(Sort sort)**
 - Ordena los resultados según los criterios especificados
 - **Sort**: Define los campos por los cuales se debe ordenar.

PagingAndSortingRepository

- **findAll(Pageable pageable)**
 - **getTotalElements():** Devuelve el número total de elementos en todas las páginas.
 - **getTotalPages():** Devuelve el número total de páginas.
 - **getNumber():** Devuelve el número de la página actual (empieza en 0).
 - **getNumberOfElements():** Devuelve el número de elementos en la página actual.
 - **getSize():** Devuelve el tamaño de la página (número de elementos por página).
 - **isFirst():** Devuelve true si la página actual es la primera.
 - **isLast():** Devuelve true si la página actual es la última.
 - **hasNext():** Devuelve true si hay una página siguiente.
 - **hasPrevious():** Devuelve true si hay una página anterior.
 - **getContent():** Devuelve la lista de elementos (Alumno) en la página actual.

Ordenamiento de resultados

- Puede aplicarse una ordenación estática usando la cláusula ***OrderBy*** después de la propiedad de búsqueda en el nombre del método de consulta que hace referencia a una propiedad y proporcionando una dirección de ordenación (Asc o Desc).
- `public List<Alumno> findByOrderByNombre();`
- `public List<Alumno> findByOrderByNombreAscPaternoDesc();`

AlumnoRepository

AlumnoRepository

`findByOrderByNombre()`

`findByOrderByNombreAscPaternoDesc()`

TypedSort

- Para una forma más segura de tipos para definir expresiones de ordenación, utilizar la entidad para la cual se define la expresión de ordenación y usar referencias de métodos para definir las propiedades en las que ordenar.

```
TypedSort<Alumno> ordenar= Sort.sort(Alumno.class);  
Sort ordenFinal = ordenar.by(Alumno::getNombre).ascending()  
.and(ordenar.by(Alumno::getPaterno)).ascending();
```


TypedSort

TypeSort

<Alumno>

alumnoOrden

Sort.sort

Alumno.class

Sort

By

Alumno::nombre

Ascending

And

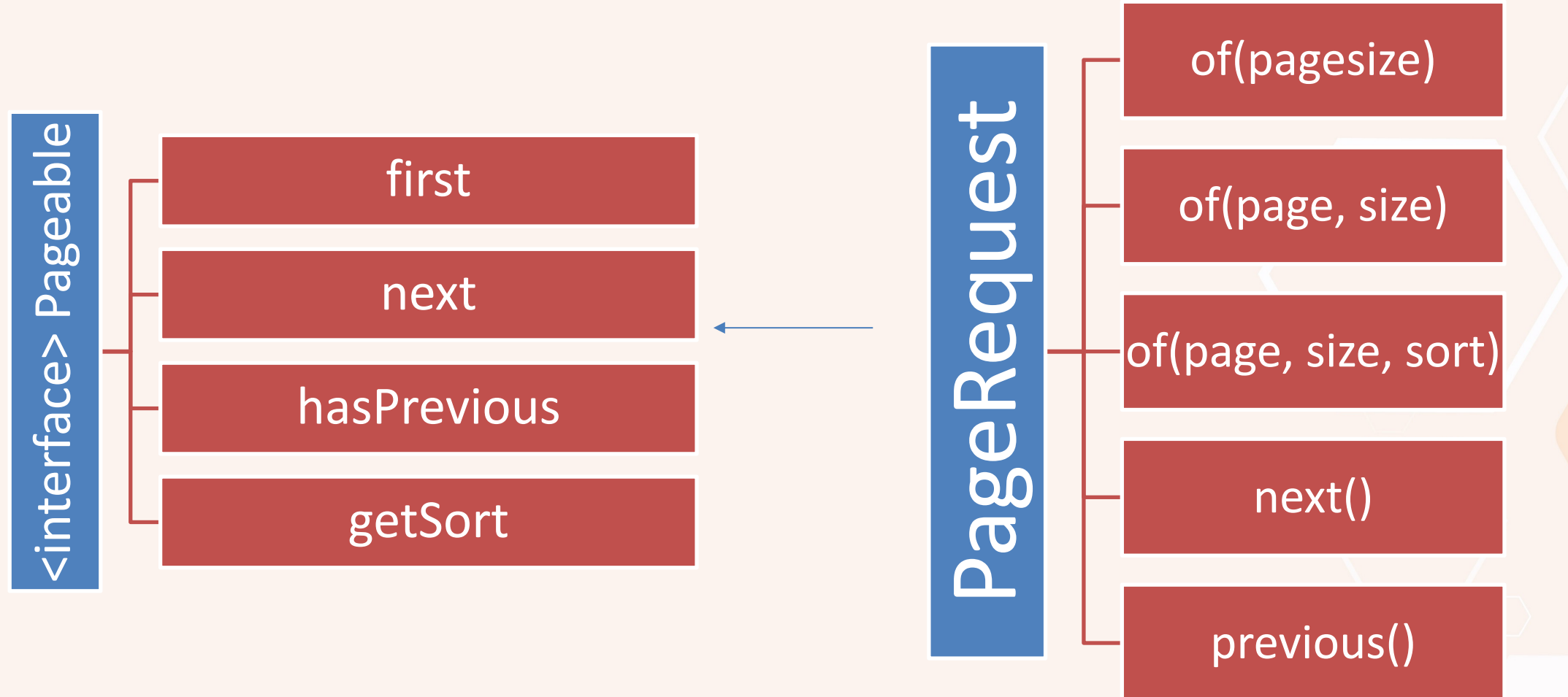
Sort

By

Alumno::paterno

Descending

Paginación



AlumnoRepository

AlumnoRepository

`findAll(Pageable)`

`findAll(Sort)`

Paginación

Page

anterior

Page

Page

siguiente

Contacto

Dr. Omar Mendoza González

omarmendoza564@aragon.unam.mx