11ª Emisión

DIPLOMADO

Desarrollo de Sistemas con Tecnología Java

Módulo 7

Persistencia con Spring Data

Dr. Omar Mendoza González

omarmendoza564@aragon.unam.mx

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Dirección General de Cómputo y de Tecnologías de información y Comunicación
Dirección de Docencia en TIC

DGTIC

50 AÑOS Educación Continua 1971 - 2021

Named Queries

- Son uno de los conceptos centrales en JPA.
- Permiten declarar una consulta en la capa de persistencia y hacer referencia a ella en la capa de negocio.
- Facilita la reutilización de una consulta existente.
- Se pueden definir usando la anotación ***@NamedQuery*** en una clase de entidad o usando un elemento <named-query /> en su asignación XML.
- Cuando define una NamedQuery, puede proporcionar una consulta ***JPQL*** o una consulta **SQL** nativa de formas muy similares.

Named Queries

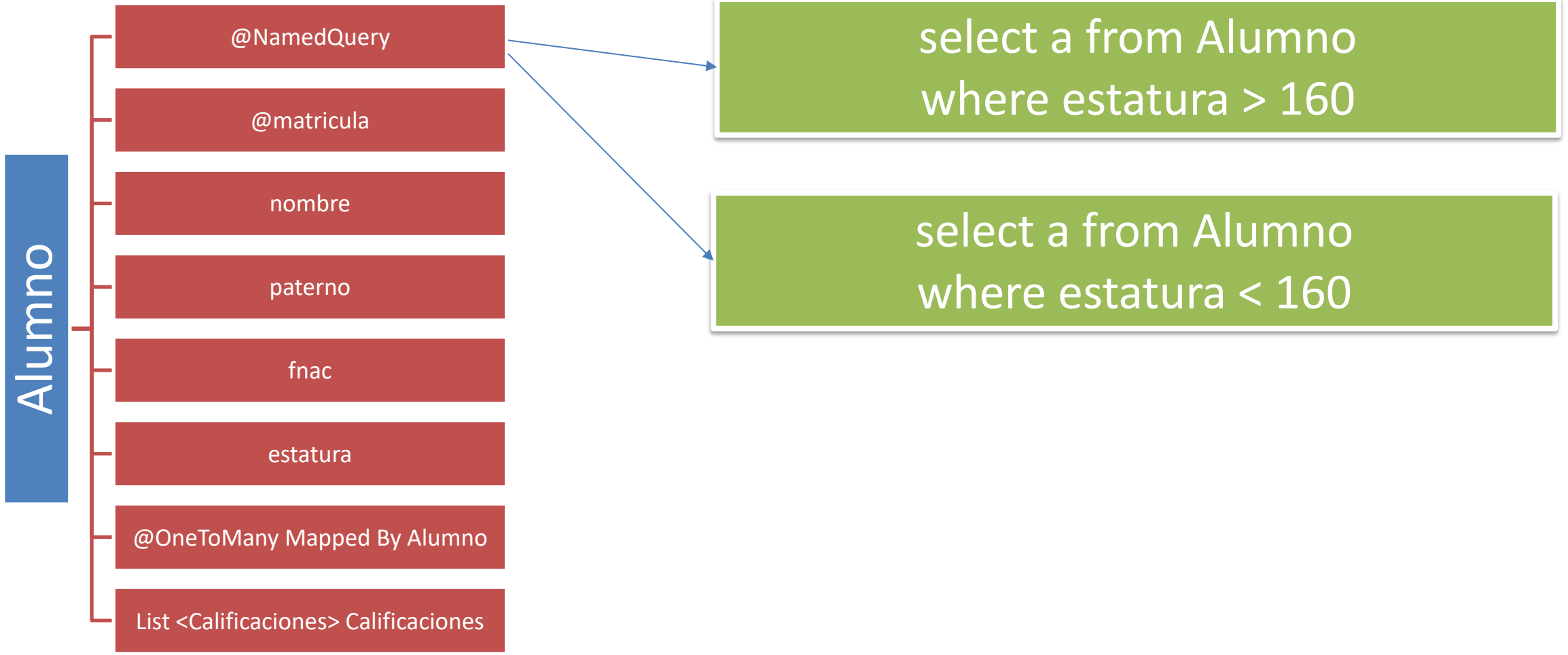
- JPQL y su sintaxis es similar a SQL. Pero existen diferencias esenciales entre ambos:
  - Se define una consulta JPQL en función de su modelo de entidad.
  - Cuando se ejecuta, el proveedor de persistencia genera una consulta SQL basada en las asignaciones de entidades y la declaración JPQL proporcionada.
  - Esto permite definir consultas independientes de la base de datos, pero se limita a las funciones admitidas por su proveedor de persistencia.
  - JPQL admite solo un pequeño subconjunto del estándar SQL y casi ninguna característica específica de la base de datos.

Named Queries

- La definición de una consulta JPQL con nombre es bastante simple.
- Solo tiene que anotar una de sus clases de entidad con ***@NamedQuery*** y proporcionar 2 String s para el nombre y los atributos de consulta .

  – El nombre de su consulta debe ser único dentro de su contexto de persistencia
  – El valor del atributo de consulta debe ser una cadena que contenga una declaración JPQL válida.
  – Si la consulta devuelve una entidad, puede definir su proyección implícitamente como puede ver en la consulta Alumno.findByNombre .
  – La consulta Author.findByNombreAndPaterno contiene una cláusula SELECT para definir la proyección de forma explícita.

Named Query y Entidades



Named Query y Parametros

Named Queries

```
@Entity
@NamedQuery(name = "Alumno.buscarPorNombre", query = "FROM
Alumno WHERE nombre = ?1")
@NamedQuery(name = "Alumno.buscarPorNombreAndPaterno",
query = "SELECT a FROM Alumno a WHERE a.nombre = ?1 AND
a.paterno = ?2")
public class Alumno { ... }
```

Named Queries

```
//NamedQuery
public List<Alumno> findAltos();
public List<Alumno> findAltosConFecha(Date fecha);
public List<Alumno> findAllWithCalificaciones();

public List<Alumno> buscarPorNombre(String nombre);
public List<Alumno> buscarPorNombreAndPaterno(String nombre, String paterno);
```

@Query

- Permite agregar una consulta dinámica dentro de la propia clase de repositorio

```
@Query ("select avg(a.estatura) from Alumno a")
public double findEstaturaPromedioAlumnos();
```

AlumnoRepository

Native Query

- Las consultas SQL nativas son más potentes y flexibles que las consultas JPQL.
- El proveedor de persistencia no analiza estas consultas y las envía directamente a la base de datos.
- Esto permite utilizar todas las funciones de SQL admitidas por la base de datos.
- Pero también debe manejar los diferentes dialectos de la base de datos si necesita admitir múltiples DBMS.

AlumnoRepository

Native Query

```
//@Query
@Query ("select avg(a.estatura) from Alumno a")
public double findEstaturaPromedioAlumnos();

@Query (value="select * from alumnos",nativeQuery=true)
public List<Alumno> findAllAlumnosNative();


@Query (value = "SELECT * FROM alumnos WHERE nombre = ? AND paterno = ?",
        nativeQuery=true)
public List<Alumno> buscarNativePorNombreAndPaterno(String nombre,
        String paterno);
```

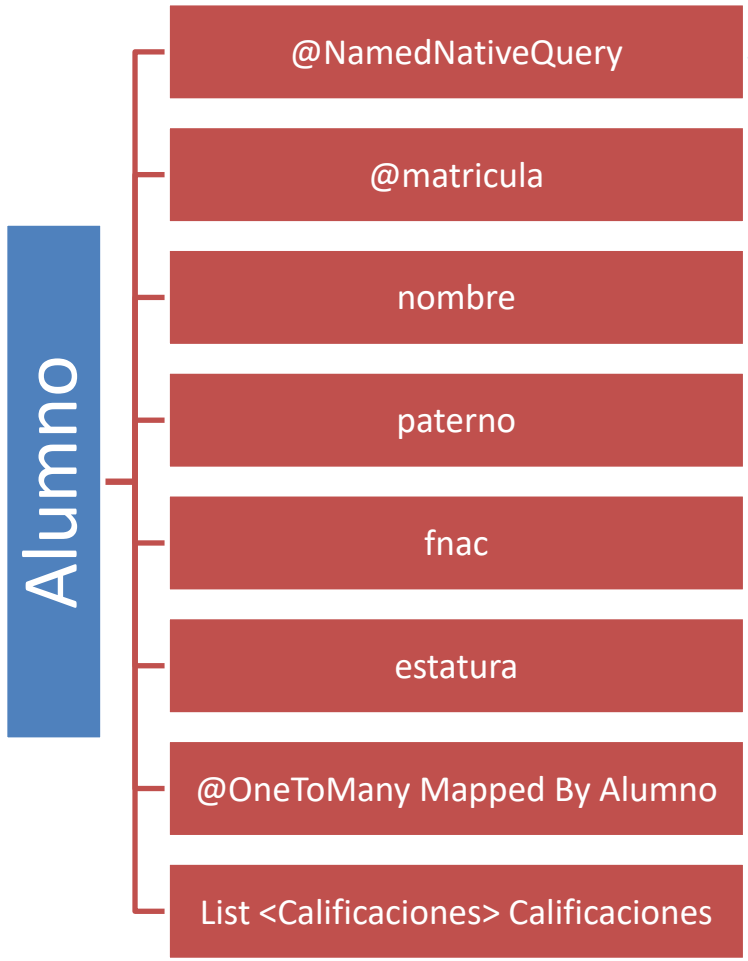
Named Native Query

- Puede definir una consulta nativa con nombre casi de la misma manera que especifica una consulta JPQL con nombre.
  - Debe usar una *anotación* **@NamedNativeQuery**
  - El valor del atributo de consulta debe ser una declaración SQL en lugar de una declaración JPQL.
  - Puede definir una clase de entidad o una referencia a un **@SqlResultSetMapping** que se usará para mapear el resultado de su consulta.

Named Native Query

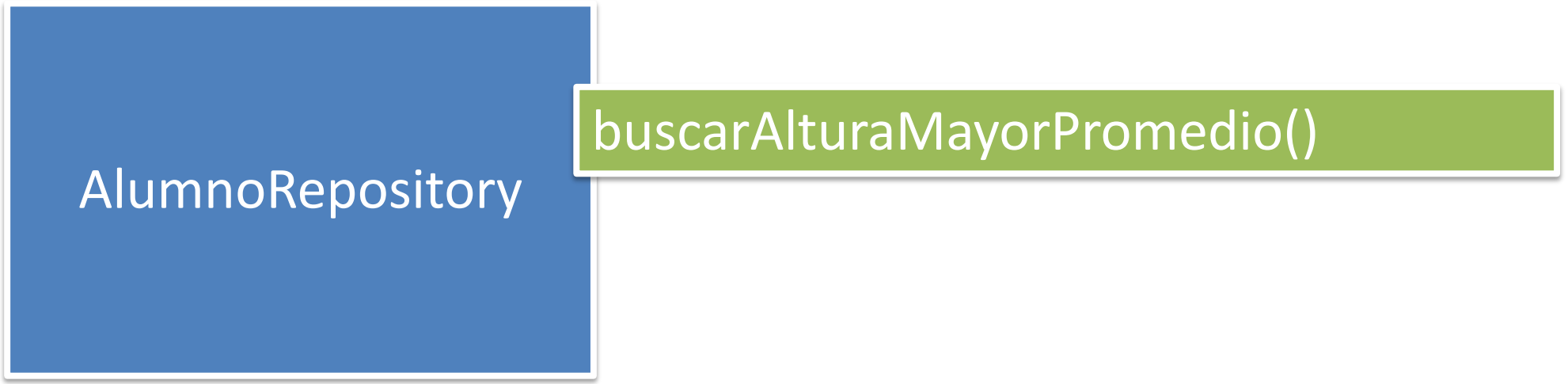
```
@Entity
@NamedNativeQuery(
      name = "Alumno.buscarAlturaMayorPromedio",
      query = "select * from alumnos where estatura >
      (select avg(estatura) from alumnos)",
      resultClass = Alumno.class)

public class Alumno { ... }
```

Named Named Query




```
select * from alumnos where estatura >
(select avg(estatura) from alumnos)
```

AlumnoRepository



Join Fetch

- **Join Fetch** es una de las opciones de las que dispone el estándar de JPA a la hora de **reducir el número de consultas** que se generan contra la base de datos, Algo que es bastante habitual y que **degrada el rendimiento**.

- Permite inicializar asociaciones o colecciones de valores junto con sus objetos principales mediante una única selección

- select distinct a from Alumno a join a.calificaciones

- select distinct a from Alumno a ***join fetch*** a.calificaciones

Join Fetch

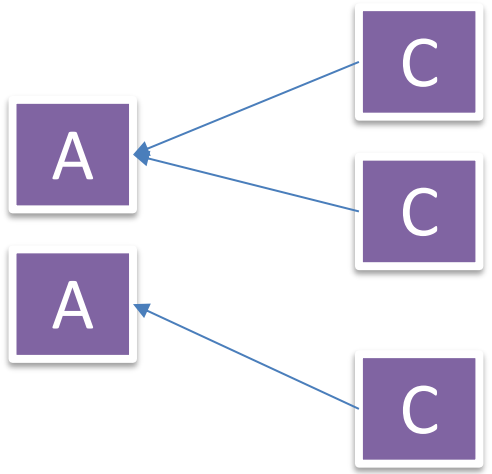
- Las dos consultas, está utilizando JOIN para consultar a todos los alumnos que tienen al menos una calificacion asociado.
- La diferencia es
  - En la primera consulta, solo devuelve los Alumnos.
  - En la segunda consulta, está devolviendo los Alumnos **y** todas las Calificaciones asociadas.
- Si usa la segunda consulta, no se necesitará hacer una nueva consulta para acceder a la BD nuevamente para ver las Calificaciones de cada Alumno.

Join Fetch

```
@NamedQuery select distinct a from Alumno a join fetch a.calificaciones
```

AlumnoRepository

findAllWithCalificaciones()



Join Fetch

```java
@Test
void buscarTodosConCalificacion() {

    List<Alumno> lista = repositorioAlumno.findAllWithCalificaciones();
    System.out.println("findAllWithCalificaciones");

    for (Alumno a: lista) {

        System.out.println(a.getNombre());

        List<Calificacion> calificaciones = a.getCalificaciones();
        //primer prueba imprimimos todo el objeto calificacion
        //calificaciones.forEach(System.out::println);
        for(Calificacion c : calificaciones) {
            System.out.println(c.getMateria() + " " + c.getCalificacion());

        }

    }
    //lista.forEach(System.out::println);
    assertThat(lista.size(), greaterThan(0));
}
```

Contacto

Dr. Omar Mendoza González

omarmendoza564@aragon.unam.mx