Tema 2 – Spring Boot extendiendo el acceso a datos

Grado en Ingeniería Informática en Tecnologías de la Información

Departamento de Ingeniería de Sistemas Informáticos y Telemáticos

Área de Lenguajes y Sistemas Informáticos

Dr. Luis V. Calderita

Objetivos

- Modificar la aplicación web creada con Spring Boot que nos permitía acceder a una base de datos relacional H2 para:
  - Ampliar las operaciones de consulta del @repository con patrones de nombres específicos
  - Señalar que es posible definir métodos específicos mediante sentencias SQL

Métodos basados en patrón de nombres

- Definir métodos de consulta de acuerdo a un patrón de nombres en Inglés.
- Spring Data es capaz de parsear la declaración de los métodos y crear la consulta asociada
- Spring Data usa el concepto de Sujeto y Predicado para parsear los métodos.
- Spring Data permite cláusulas adicionales para ajustar más los métodos

Query Patterns admitidos

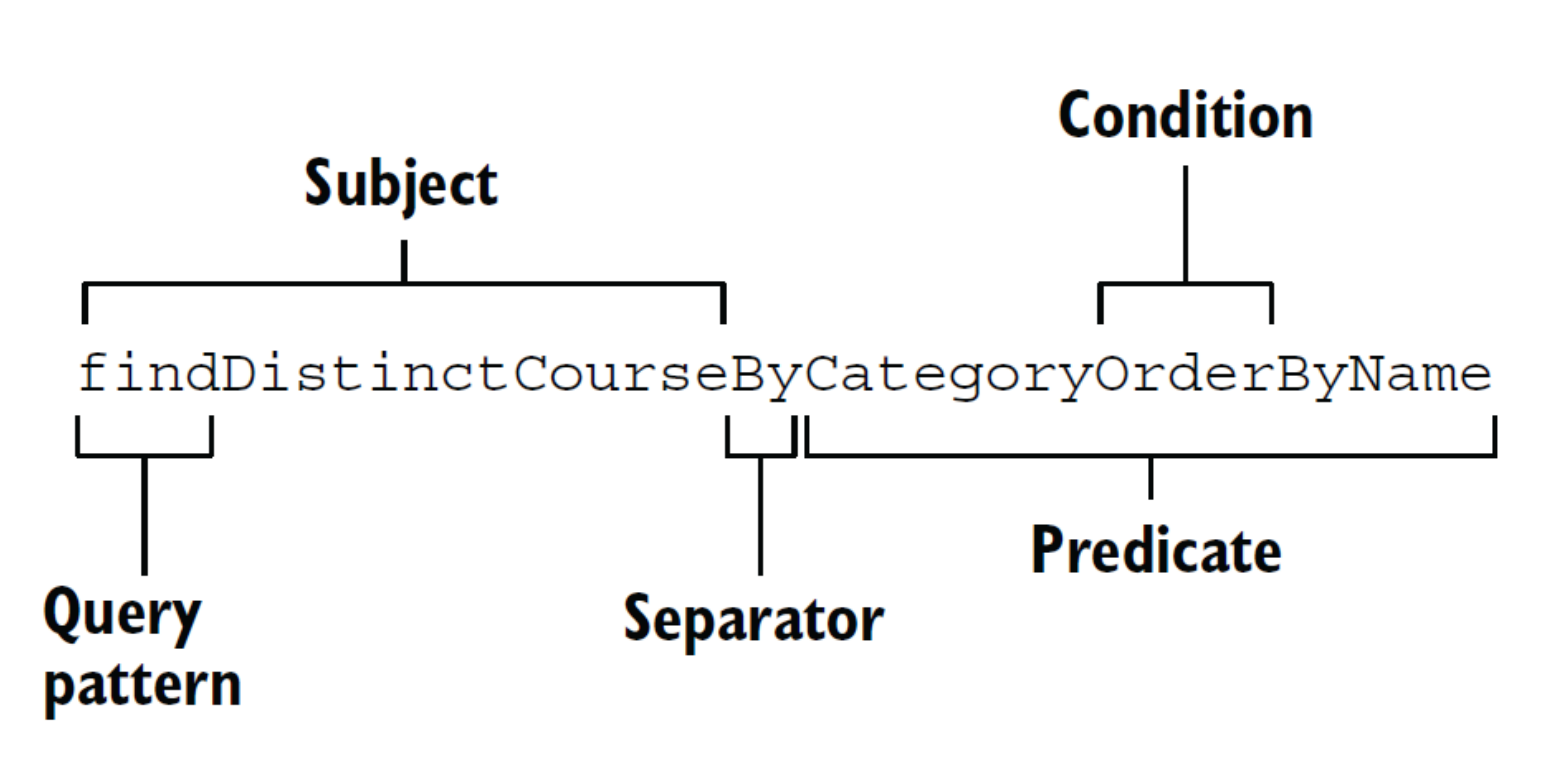
- ***Query***, para consultar entidades
  - find..By, read..By, get..By,query..By, search..By
- ***Count***, para contar entidaes
  - count..By()
- ***Exists***, para definir métodos que comprueban la existencia de una entidad
  - exists..By()
- ***Delete****, para borrar entidades*
  - delete..By() y remove..By()

Patrones admitidos: Modificadores y Condiciones

- A la declaración del método se le pueden añadir los modificadores y condiciones habituales, como por ejemplo*:*
  - *Distinct, And, Or, LessThan, StartsWith, OrderBy…*
- Todos las opciones soportadas:
  - *https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#repository-query-keywords*

Estructura del patrón de nombres

- `findDistinctCourseByCategoryOrderByName()`



Ejemplos, JPA Named Queries

Returns the count of courses for the supplied category. Count queries can return an integer or long type.

Finds all courses by category and orders the entities by name

Checks if a course with the supplied name exists. Returns true if course exists and false otherwise. Exists queries return the Boolean type.

Finds all courses by category. A find query returns an Iterable type.

```
@Repository
public interface CourseRepository extends CrudRepository<Course, Long> {

    Iterable<Course> findAllByCategory(String category);
    Iterable<Course> findAllByCategoryOrderByName(String category);
    boolean existsByName(String name);
    long countByCategory(String category);
    Iterable<Course> findByNameOrCategory(String name, String category);
    Iterable<Course> findByNameStartsWith(String name);
```

Finds all courses that start with the supplied course name string

Finds all courses that match the supplied course name or the course category

Especificando consultas

- Las consultas basadas en nombres son una buena opción en muchas ocasiones.
- Para aquellas situaciones en las que sea necesario escribir una sentencia Spring Data JPA incluye también mecanismos
- Las sentencias se escriben en lenguaje Jakarta Persistence Query Language (JPQL)

Specifying query using @NamedQuery

- La anotación @NamedQuery se define en una **entidad** o en su superclase
- Los parámetros obligatorios de la anotación son name y query
  - Name, identifica el nombre del método en la interfaz Repository
  - Query, contiene la consulta en formato JPQL
- El método y la consulta se definen en la @Entity. El método también se declara en el repository.

Ejemplo, @NamedQuery

```
@Entity

@NamedQuery(name = "Course.findAllByCategoryAndRating",
➥ query = "select c from Course c where c.category=?1
➥ and c.rating=?2")
public class Course {
```

The @NamedQuery annotation lets you specify the query for the repository method in JPQL format.

```
@Repository
public interface CourseRepository extends CrudRepository<Course, Long> {

    Iterable<Course> findAllByCategoryAndRating(String category, int rating);
}
```

Cuando llamemos al método desde una instancia del repositorio, se ejecutará la consulta

@NamedQueries

- @NamedQueries, permite crear más de una consulta:

```
@Entity

@NamedQueries({
    @NamedQuery(name = "Course.findAllByRating",
➥ query = "select c from Course c where c.rating=?1"),
    @NamedQuery(name = "Course.findAllByCategoryAndRating",
➥ query = "select c from Course c
➥ where c.category=?1 and c.rating=?2"),
})
public class Course {
```

@Query

- @Query permite declarar la consulta directamente en la declaración del método en la interfaz del repositorio

```
import org.springframework.data.jpa.repository.Query;
@Repository
public interface UsuarioRepository extends CrudRepository<Usuario, Long> {

    @Query ("select u from Usuario u where u.email =?1")
    List<Usuario> encontrarPorEmail(String email);
```

Practicando

- Sugerencias:
  - Extiende los métodos de la interfaz del repositorio usando la definición de métodos mediante patrones de nombres
  - Reimplementa los métodos CRUD mediante @Query. Usa nombres en castellano para diferenciarlos

Recursos

- Spring Initializr:
  - https://start.spring.io
- JPQL
  - https://eclipse-ee4j.github.io/jakartaee-tutorial/#the-jakarta-persistence-query-language
- JPA Named Queries:
  - https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods