01-2025

code URJC

Desarrollo Web

Tecnologías de servidor web

Tema 2.2: Bases de Datos SQL en Spring

Universidad Rey Juan Carlos

Micael Gallego, Francisco Gortázar, Michel Maes, Óscar Soto e Iván Chicano

code
URJC

Micael Gallego, Francisco Gortázar, Michel Maes, Óscar Soto, Iván Chicano

Bases de datos SQL en Spring

code URJC

- El proyecto **Spring Data** ofrece mecanismos para el acceso a **Bases de datos relacionales y no relacionales**
  - **Creación del esquema** partiendo de las clases del código Java (o viceversa)
  - **Conversión automática** entre objetos Java y el formato propio de la base de datos
  - **Creación de consultas** en base a métodos en interfaces

http://projects.spring.io/spring-data/
http://projects.spring.io/spring-data-jpa/

Bases de datos SQL en Spring

code URJC

ejem1

- **Objeto de la base de datos (entidad)**

Anotaciones usadas para generar el esquema y para hacer la conversión entre objetos y filas

El atributo anotado con **@Id** es la clave primaria de la tabla. Lo habitual es que se genere de forma automática

```
@Entity
public class Customer {

  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private long id;

  private String firstName;
  private String lastName;

  // Constructor necesario para la carga desde BBDD
  protected Customer() {}

  public Customer(String firstName, String lastName) {
    this.firstName = firstName;
    this.lastName = lastName;
  }

  // Getter, Setters and toString
  ...
}
```

Bases de datos SQL en Spring

code URJC

ejem1

- **Repositorio**

El interfaz padre **JpaRepository** dispone de métodos para **consultar, guardar, borrar** y **modificar** objetos de la base de datos.

```
public interface CustomerRepository extends JpaRepository<Customer, Long> {

   List<Customer> findByLastName(String lastName);

   List<Customer> findByFirstName(String firstName);

}
```

Métodos que se traducen automáticamente en **consultas a la BBDD** en base al nombre y los parámetros.

Si no se necesita ningún tipo de consulta especial, el **interfaz puede quedar vacío**

Clase de la **entidad** y clase de su identificador (generalmente **Integer** o **Long**)

Bases de datos SQL en Spring

code URJC

ejem1

- **Uso de la base de datos**
  - Un componente **inyecta el repositorio** con @Autowired
  - Métodos disponibles:
    - Consultar todos (**findAll**)
    - Consulta por id (**findById**)
    - Guardar un nuevo objeto o actualizarlo (**save**)
    - Borrar un objeto (**delete**)
    - Consultas por cualquier criterio (métodos añadidos al interfaz)

Bases de datos SQL en Spring

code URJC

ejem1

- Uso de la BBDD

```
@Controller
public class DataBaseUsage implements CommandLineRunner {

  @Autowired
  private CustomerRepository repository;

  @Override
  public void run(String... args) throws Exception {

     repository.save(new Customer("Jack", "Bauer"));
     repository.save(new Customer("Chloe", "O'Brian"));

     List<Customer> bauers = repository.findByLastName("Bauer");
     for (Customer bauer : bauers) {
          System.out.println(bauer);
     }

     repository.delete(bauers.get(0));
  }
}
```

Controlador especial que se ejecuta cuando se **inicia la aplicación**. Puede leer los parámetros de la **línea de comandos**

Código de ejemplo que usa el **repositorio** para guardar, consultar y borrar datos de la **BBDD**

Bases de datos SQL en Spring

ejem1

- pom.xml

```
<groupId>es.codeurjc</groupId>
<artifactId>bbdd_ejem1</artifactId>
<version>0.1.0</version>

<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>3.4.1</version>
</parent>
...
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
 <dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
  </dependency>
</dependencies>
```

Dependencia a **Spring Data** para BBDD **relacionales**

Dependencia a la **BBDD en memoria H2**. En las bases de datos en memoria, el esquema se genera de forma **automática** al iniciar la app

Bases de datos SQL en Spring

code URJC

ejem2

- Una aplicación web puede gestionar la información en una BBDD
  - Se añaden las dependencias de JPA y la BBDD en el **pom.xml**
  - En vez de usar una estructura en memoria se usa un **repositorio**
- Como ejemplo vamos a transformar la aplicación de gestión de posts

Bases de datos SQL en Spring

code URJC

ejem2

- pom.xml

```
...
<dependencies>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
  </dependency>

</dependencies>
```

Dependencias **Web**

Dependencia a la **BBDD H2**

Bases de datos SQL en Spring

code URJC

ejem2

- Objeto de la base de datos (entidad)

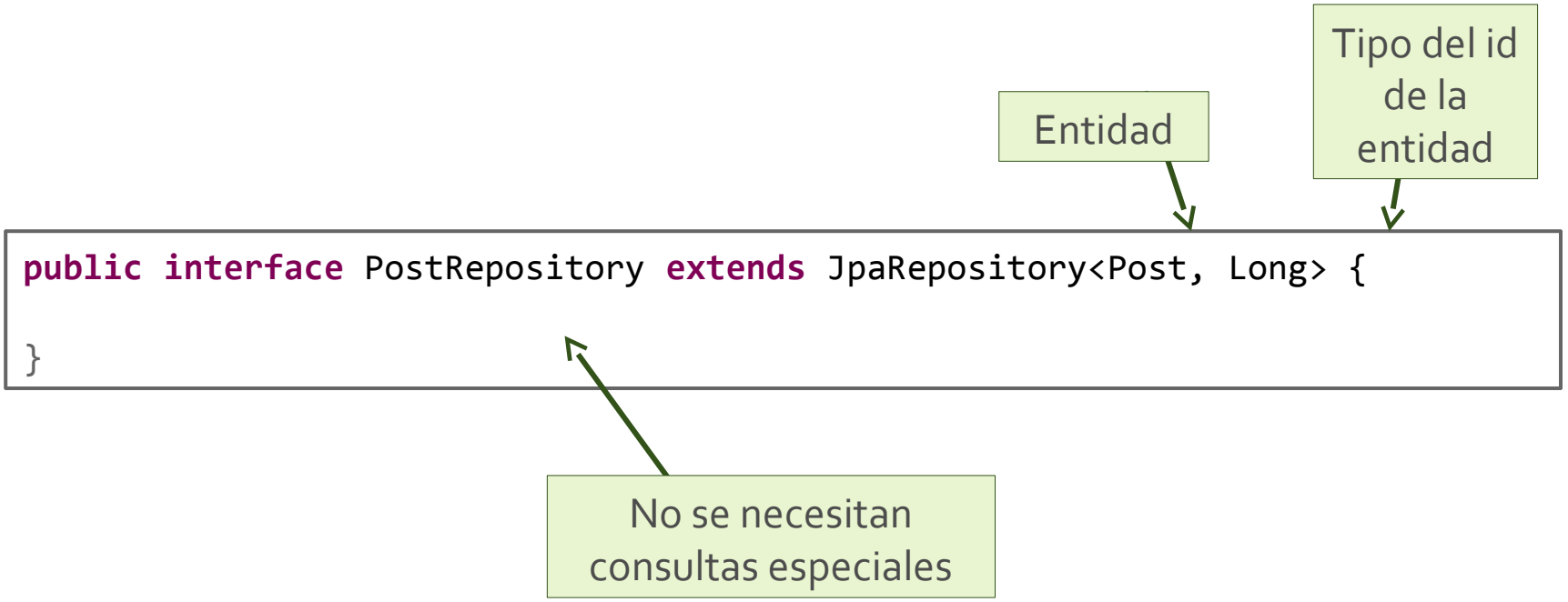
```
@Entity
public class Post {

	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private Long id;

	private String username;
	private String title;
	private String text;

	public Post() { }

	public Post(String username, String title, String text) {
		super();
		this.username = username;
		this.title = title;
		this.text = text;
	}
}
```

Bases de datos SQL en Spring

code URJC

ejem2

- Repositorio



Bases de datos SQL en Spring

code URJC

ejem2

- Controlador web

Se pueden inyectar tantos repositorios como sea necesario

Un método **@PostConstruct** se ejecutará después de haber inyectado las dependencias

```
@Controller
public class PostController {

  @Autowired
  private PostRepository posts;

  @PostConstruct
  public void init() {
    posts.save(new Post("Pepe", "Vendo moto", "Barata, barata"));
    posts.save(new Post("Juan", "Compro coche", "Pago bien"));
  }

  @GetMapping("/")
  public String showPosts(Model model) {
      model.addAttribute("posts", posts.findAll());
      return "index";
  }

...
```

Desde los métodos se usa el repositorio

Bases de datos SQL en Spring

code URJC

ejem2

- **Consulta de un recurso**
  - Gestión de *Optional* con *findById*

```
@GetMapping("/post/{id}")
public String showPost(Model model, @PathVariable long id) {

    Optional<Post> op = posts.findById(id);

    if (op.isPresent()) {
        Post post = op.get();
        model.addAttribute("post", post);
        return "show_post";
    } else {
        return "post_not_found";
    }
}
```

Bases de datos SQL en Spring

code URJC

- **Manipulación de Optional**
  - Históricamente en Java cuando un método de consulta indica la **ausencia de valor** devolviendo **null**
  - Usar null es propenso a errores y puede provocar **NullPointerException**
  - **Optional** representa en el sistema de tipos que el valor es opcional y obliga al desarrollador a considerar cómo quiere gestionar la ausencia de valor

https://blogs.oracle.com/javamagazine/12-recipes-for-using-the-optional-class-as-its-meant-to-be-used

Bases de datos SQL en Spring

code URJC

ejem2

- **Consulta de un recurso**
  - Gestión de *Optional* con *findById*

```
@GetMapping("/post/{id}")
public String showPost(Model model, @PathVariable long id) {

    Optional<Post> op = posts.findById(id);

    if (op.isPresent()) {
        Post post = op.get();
        model.addAttribute("post", post);
        return "show_post";
    } else {
        return "post_not_found";
    }
```

Si hay valor (**isPresent**) lo obtenemos (**get**) para devolverlo

En un repositorio el método **findById** devuelve *Optional<Post>*

Bases de datos SQL en Spring

code URJC

ejem2

- Creación de recurso

```
@PostMapping("/post/new")
public String newPost(Model model, Post post) {

    posts.save(post);

    return "saved_post";
}
```

Bases de datos SQL en Spring

code URJC

ejem2

- Reemplazo de recurso

```
@PostMapping("/editpost")
public String editBookProcess(Model model, Post editedPost) {

    Optional<Post> op = posts.findById(editedPost.getId());
    if (op.isPresent()) {
        posts.save(editedPost);
        model.addAttribute("post", editedPost);
        return "edited_post";
    } else {
        return "post_not_found";
    }
}
```

Bases de datos SQL en Spring

code URJC

ejem2

- Borrado de recurso

```
@PostMapping("/post/{id}/delete")
public String deletePost(Model model, @PathVariable long id) {

    Optional<Post> post = posts.findById(id);

    if (post.isPresent()) {
        posts.deleteById(id);
        return "deleted_post";
    } else {
        return "post_not_found";
    }
}
```

Bases de datos SQL en Spring

code URJC

ejem2

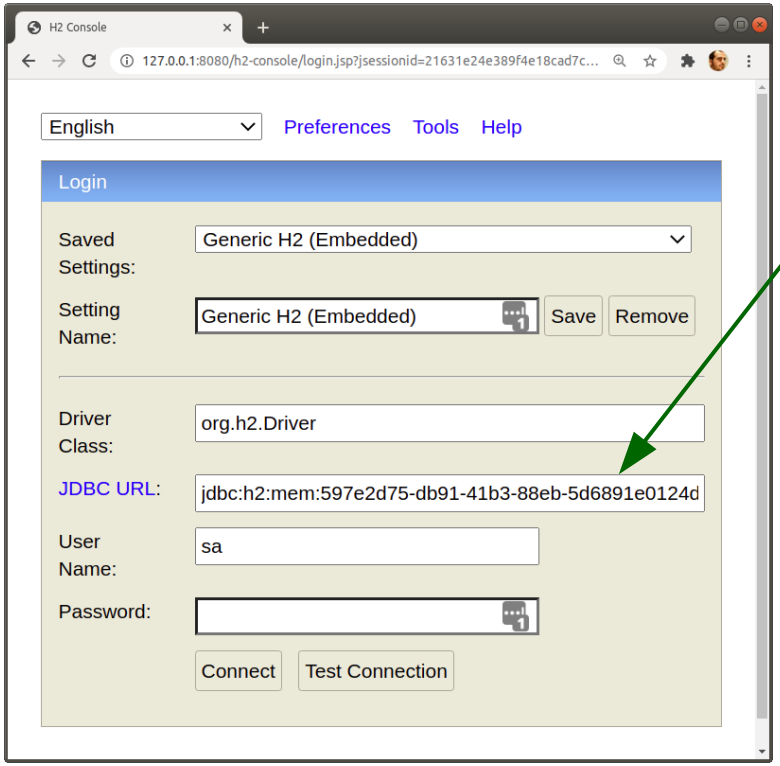
- **Consola web de base de datos H2**
  - Si desarrollamos una **aplicación web** que use una **base de datos H2** podemos acceder a una consola web de la propia BBDD
  - La consola es accesible en
    http://127.0.0.1:8080/h2-console
  - Para activar esa consola:
    - Usar las **devtools** en ese proyecto
    - O especificar en el fichero **application.properties**

```
spring.h2.console.enabled=true
```

Bases de datos SQL en Spring

ejem2

- Consola web de base de datos H2



Hay que poner como JDBC URL la ruta que aparece en el log

```
H2 console available at '/h2-
console'. Database available at
'jdbc:h2:mem:597e2d75-db91-41b3-
88eb-5d6891e0124d'
```

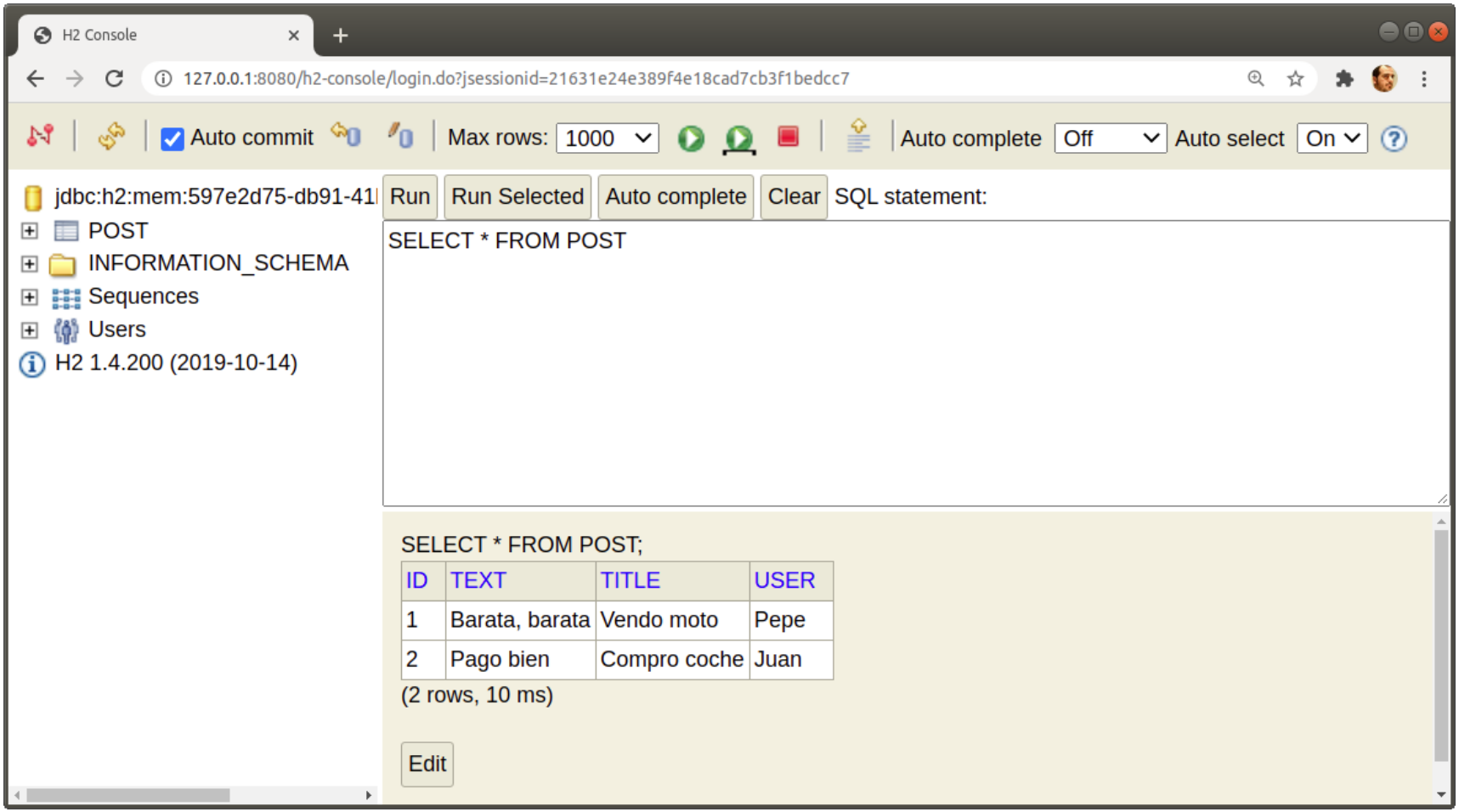
Para simplificar podemos configurar una URL fija:

```
spring.datasource.url=jdbc:h2:mem:testdb
```

Bases de datos SQL en Spring

code URJC

ejem2



Bases de datos SQL en Spring

code URJC

- Las apps de base de datos en Spring usan dos librerías a la vez:
  - **JPA (*Java Persistence API*)**
    - Estándar de Java EE para acceso a base de datos relacionales
  - **Spring Data**
    - Facilidades propias de Spring que facilitan el acceso a base de datos con JPA

Bases de datos SQL en Spring

code URJC

- **Spring Data**
  - Permite la creación de **consultas partiendo de métodos** en los repositorios
  - Se puede usar con **cualquier** base de datos (relacional o NoSQL)
  - Los métodos pueden tener varios parámetros que forman **expresiones lógicas**: Y, O, No…
  - Los métodos se pueden anotar para definir la consulta de forma más precisa con **JPQL o SQL**

https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#repositories

Bases de datos SQL en Spring

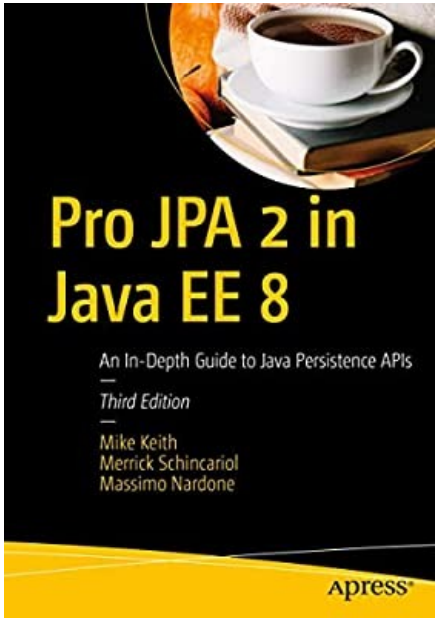
code URJC

- JPA (*Java Persistence API*)
  - Acceso a Bases de datos relacionales desde Java
  - **JPA** forma parte del estándar **Java EE**
  - **Funcionalidades**
    - Generación del **esquema** de la base de datos partiendo de las entidades (clases Java)
    - **Conversión automática** entre objetos y filas de tablas
    - Lenguaje de consulta de alto nivel **JPQL**

http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html

Bases de datos SQL en Spring

- JPA (*Java Persistence API*)



- **Pro JPA 2 in Java EE 8**
- By Mike Keith , Merrick Schincariol, Massimo Nardone
- Apress

https://learning.oreilly.com/library/view/pro-jpa-2/9781484234204/

https://www.javacodegeeks.com/2015/02/jpa-tutorial.html

Bases de datos SQL en Spring

code URJC

- **JPA (*Java Persistence API*)**
  - Para usar JPA es necesario usar una **librería** concreta que lo implemente
  - En Spring por defecto se usa la librería **Hibernate**

http://www.hibernate.org

Bases de datos SQL en Spring

code URJC

- **JPA (*Java Persistence API*)**
  - La técnica para convertir entre el modelo de objetos y el modelo relacional se conoce como "**mapeo objeto relacional**" (*object relational mapping*) u ORM
  - Un ORM realiza las **conversiones** pertinentes entre **objetos/clases** y **filas/tablas**
  - Modos de funcionamiento:
    - **Generación de tablas partiendo de clases***
    - Generación de clases partiendo de tablas

Bases de datos SQL en Spring

code URJC

ejem2

- **Mostrar las sentencias SQL**
  - Para aprender JPA y depurar los posibles errores es muy útil poder ver qué sentencias se ejecutan en la BBDD

application.properties

```
spring.jpa.properties.hibernate.format_sql=true
logging.level.org.hibernate.SQL=DEBUG
logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE
```

Bases de datos SQL en Spring

- Spring Data
  - http://projects.spring.io/spring-data/
- Spring Data Commons
  - http://docs.spring.io/spring-data/commons/docs/current/reference/html/
- Spring Data JPA
  - http://docs.spring.io/spring-data/jpa/docs/current/reference/html/
- Bases de datos SQL en Spring Boot
  - http://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-sql.html
- Inicialización de bases de datos SQL en Spring Boot
  - http://docs.spring.io/spring-boot/docs/current/reference/html/howto-database-initialization.html
- Tutorial Spring Data JPA
  - https://spring.io/guides/gs/accessing-data-jpa/