

# Clase 11

SpringBoot + Hibernate  
+ WEB + REST

# JPA (Java Persistence API)

- Especificación en JAVA (interfaz).
- Especificación para almacenar y recuperar datos en bases de datos.
- Define un marco de trabajo.
- Define anotaciones y métodos para mapear objetos a tablas.
- Operaciones CRUD.
- SQL complejo a través de JPQL (Java Persistence Query Language)

# ORM / Hibernate

- ORM (Object-Relational Mapping)
  - Capa de abstracción entre el código Java y las tablas de la BD
  - Mapeo entre registros de la BD y objetos Java
- Hibernate
  - Mapeo BD  $\longleftrightarrow$  Objetos
  - Soporte a múltiples motores de base de datos
  - Consultas con HQL y JPQL
  - Cache
  - Transacciones y sesiones
  - Relaciones entre objetos (OneToOne, ManyToOne, OneToMany)
  - Herencia y polimorfismo

# Spring Initializr



## Project

☐ Gradle - Groovy

☐ Gradle - Kotlin

☒ Maven

## Language

☒ Java

☐ Kotlin

☐ Groovy

## Spring Boot

☐ 3.1.1 (SNAPSHOT)

☒ 3.1.0

☐ 3.0.8 (SNAPSHOT)

☐ 3.0.7

☐ 2.7.13 (SNAPSHOT)

☐ 2.7.12

## Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☒ Jar ☐ War

Java

☐ 20

☒ 17

☐ 11

☐ 8

## Dependencies

ADD DEPENDENCIES... ⌘ + B

### Lombok

DEVELOPER TOOLS

Java annotation library which helps to reduce boilerplate code.

### Spring Web

WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

### Spring HATEOAS

WEB

Eases the creation of RESTful APIs that follow the HATEOAS principle when working with Spring / Spring MVC.

# Spring Initializr



## Project

☐ Gradle - Groovy

☐ Gradle - Kotlin ☒ Maven

## Language

☒ Java

☐ Kotlin

☐ Groovy

## Spring Boot

☐ 3.2.0 (SNAPSHOT) ☐ 3.2.0 (M1) ☐ 3.1.3 (SNAPSHOT) ☒ 3.1.2

☐ 3.0.10 (SNAPSHOT) ☐ 3.0.9 ☐ 2.7.15 (SNAPSHOT) ☐ 2.7.14

## Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☒ Jar ☐ War

Java ☐ 20 ☒ 17 ☐ 11 ☐ 8

## Dependencies

ADD DEPENDENCIES... ⌘ + B

### Spring Web

WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

### Spring Data JPA

SQL

Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

### Spring Boot DevTools

DEVELOPER TOOLS

Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

### H2 Database

SQL

Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.

# pom.xml

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>3.1.2</version>
  <relativePath/> <!-- lookup parent from repository -->
</parent>
<groupId>ar.edu.um.programacion2.anio2023</groupId>
<artifactId>clasell</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>clasell</name>
<description>Demo project for Spring Boot</description>
<properties>
  <java.version>17</java.version>
</properties>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
  </dependency>
```

# Proyecto

```
External Libraries
> < zulu-17 > /Users/Villano/.sdkman/candidates/java/17.0.2-zulu
> Maven: ch.qos.logback:logback-classic:1.4.8
> Maven: ch.qos.logback:logback-core:1.4.8
> Maven: com.fasterxml.jackson.core:jackson-annotations:2.15.2
> Maven: com.fasterxml.jackson.core:jackson-core:2.15.2
> Maven: com.fasterxml.jackson.core:jackson-databind:2.15.2
> Maven: com.fasterxml.jackson.datatype:jackson-datatype-jdk8:2.15.2
> Maven: com.fasterxml.jackson.datatype:jackson-datatype-jsr310:2.15.2
> Maven: com.fasterxml.jackson.module:jackson-module-parameter-names:2.15.2
> Maven: com.fasterxml:classmate:1.5.1
> Maven: com.h2database:h2:2.1.214
> Maven: com.jayway.jsonpath:json-path:2.8.0
> Maven: com.sun.istack:istack-commons-runtime:4.1.2
> Maven: com.vaadin.external.google:android-json:0.0.20131108.vaadin1
> Maven: com.zaxxer:HikariCP:5.0.1
> Maven: io.micrometer:micrometer-commons:1.11.2
> Maven: io.micrometer:micrometer-observation:1.11.2
> Maven: io.smallrye:jandex:3.0.5
> Maven: jakarta.activation:jakarta.activation-api:2.1.2
> Maven: jakarta.annotation:jakarta.annotation-api:2.1.1
> Maven: jakarta.inject:jakarta.inject-api:2.0.1
> Maven: jakarta.persistence:jakarta.persistence-api:3.1.0
> Maven: jakarta.transaction:jakarta.transaction-api:2.0.1
> Maven: jakarta.xml.bind:jakarta.xml.bind-api:4.0.0
> Maven: net.bytebuddy:byte-buddy:1.14.5
> Maven: net.bytebuddy:byte-buddy-agent:1.14.5
> Maven: net.minidev:accessors-smart:2.4.11
> Maven: net.minidev:json-smart:2.4.11
> Maven: organtlr:antlr4-runtime:4.10.1
> Maven: org.apache.logging.log4j:log4j-api:2.20.0
> Maven: org.apache.logging.log4j:log4j-to-slf4j:2.20.0
> Maven: org.apache.tomcat.embed:tomcat-embed-core:10.1.11
> Maven: org.apache.tomcat.embed:tomcat-embed-el:10.1.11
> Maven: org.apache.tomcat.embed:tomcat-embed-websocket:10.1.11
> Maven: org.apiguardian:apiguardian-api:1.1.2
> Maven: org.aspectj:aspectjweaver:1.9.19
> Maven: org.assertj:assertj-core:3.24.2
```

```
> Maven: org.eclipse.angus:angus-activation:2.0.1
> Maven: org.glassfish.jaxb:jaxb-core:4.0.3
> Maven: org.glassfish.jaxb:jaxb-runtime:4.0.3
> Maven: org.glassfish.jaxb:txw2:4.0.3
> Maven: org.hamcrest:hamcrest:2.2
> Maven: org.hibernate.common:hibernate-commons-annotations:6.0.6.Final
> Maven: org.hibernate.orm:hibernate-core:6.2.6.Final
> Maven: org.jboss.logging:jboss-logging:3.5.3.Final
> Maven: org.junit.jupiter:junit-jupiter:5.9.3
> Maven: org.junit.jupiter:junit-jupiter-api:5.9.3
> Maven: org.junit.jupiter:junit-jupiter-engine:5.9.3
> Maven: org.junit.jupiter:junit-jupiter-params:5.9.3
> Maven: org.junit.platform:junit-platform-commons:1.9.3
> Maven: org.junit.platform:junit-platform-engine:1.9.3
> Maven: org.mockito:mockito-core:5.3.1
> Maven: org.mockito:mockito-junit-jupiter:5.3.1
> Maven: org.objenesis:objenesis:3.3
> Maven: org.opentest4j:opentest4j:1.2.0
> Maven: org.ow2.asm:asm:9.3
> Maven: org.projectlombok:lombok:1.18.28
> Maven: org.skyscreamer:Project Library:1.5.1
> Maven: org.slf4j:jul-to-slf4j:2.0.7
> Maven: org.slf4j:slf4j-api:2.0.7
> Maven: org.springframework.boot:spring-boot:3.1.2
> Maven: org.springframework.boot:spring-boot-autoconfigure:3.1.2
> Maven: org.springframework.boot:spring-boot-devtools:3.1.2
> Maven: org.springframework.boot:spring-boot-starter:3.1.2
> Maven: org.springframework.boot:spring-boot-starter-aop:3.1.2
> Maven: org.springframework.boot:spring-boot-starter-data-jpa:3.1.2
> Maven: org.springframework.boot:spring-boot-starter-jdbc:3.1.2
> Maven: org.springframework.boot:spring-boot-starter-json:3.1.2
> Maven: org.springframework.boot:spring-boot-starter-logging:3.1.2
> Maven: org.springframework.boot:spring-boot-starter-test:3.1.2
> Maven: org.springframework.boot:spring-boot-starter-tomcat:3.1.2
> Maven: org.springframework.boot:spring-boot-starter-web:3.1.2
> Maven: org.springframework.boot:spring-boot-test:3.1.2
> Maven: org.springframework.boot:spring-boot-test-autoconfigure:3.1.2
> Maven: org.springframework.data:spring-data-commons:3.1.2
> Maven: org.springframework.data:spring-data-jpa:3.1.2
```

# Servicio - Interfaz

```
public interface PersonaService {  
    public Persona get(long id);  
    public void add(Persona persona);  
    public void remove(long id);  
    public void remove(Persona p);  
    public List<Persona> getAll();  
}
```



# Servicio

```
@Service
@Qualifier("memoria")
public class PersonaMemoriaService implements PersonaService {
    protected Map<Long, Persona> personas;

    public PersonaMemoriaService() {
        this.personas = new HashMap<>();
    }

    @Override
    public Persona get(long id) {
        Persona p = this.personas.get(id);
        return p;
    }

    @Override
    public void add(Persona p) {
        this.personas.put(p.getId(), p);
    }
}
```

# Servicio

```
@Override
public void remove(long id) {
    this.personas.remove(id);
}

@Override
public void remove(Persona p) {
    this.personas.remove(p.getId());
}

@Override
public List<Persona> getAll() {
    List<Persona> personas;
    personas =
this.personas.values().stream().collect(Collectors.toCollection(ArrayList::new));
    return personas;
}
}
```

# Controlador REST

```
@RestController
@RequestMapping ("/persona/")
public class PersonaController {

    @Autowired
    @Qualifier ("memoria")
    PersonaService personas;

    @GetMapping ("/{id}")
    public Persona get (@PathVariable Long id) {
        Persona p = this.personas.get (id);
        return p;
    }

    @GetMapping
    public List<Persona> getAll () {
        List<Persona> personas = this.personas.getAll ();
        return personas;
    }

    @PostMapping
    public Persona post (@RequestBody Persona persona) {
        this.personas.add (persona);
        return persona;
    }
}
```

# Persona

```
@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
public class Persona {
    @Id
    @GeneratedValue
    protected long id;

    protected String nombre;
    protected String apellido;
    protected int codigo;
}
```

# Repository

```
public interface PersonaRepository extends JpaRepository<Persona, Long> {  
  
}
```

```
this.personaRepository.  
}  
  
@Override  
public void remove(long id) {  
    Optional<Persona> persona =  
        this.personaRepository.findById(id);  
    if (persona.isPresent()) {  
        persona.get().delete();  
    }  
}  
  
@Override  
public void remove(Persona persona) {  
    this.personaRepository.delete(persona);  
}  
  
1 usage  
  
@Override  
public List<Persona> getPersonas() {  
    List<Persona> personas =  
        this.personaRepository.findAll();  
    return personas;  
}
```

Method	Return Type
findAll()	List<Persona>
delete(Persona entity)	void
save(S entity)	S
findById(Long id)	Optional<Persona>
findAll(Sort sort)	List<Persona>
findAll(Pageable pageable)	Page<Persona>
findAll(Example<S> example)	
findAll(Example<S> example, Sort sort)	
findAll(Example<S> example, Pageable pageable)	
findBy(Example<S> example, Function<S, Boolean> predicate)	
equals(Object obj)	
hashCode()	
toString()	
count()	
count(Example<S> example)	
deleteAll()	
deleteAll(Iterable<? extends Persona> entities)	
deleteAllById(Iterable<? extends Long> ids)	
deleteAllByIdInBatch(Iterable<Long> ids)	
deleteAllInBatch()	
deleteAllInBatch(Iterable<Persona> entities)	

```
this.personaRepository.  
}  
  
@Override  
public void remove(long id) {  
    Optional<Persona> persona =  
        this.personaRepository.findById(id);  
    if (persona.isPresent()) {  
        persona.get().delete();  
    }  
}  
  
@Override  
public void remove(Persona persona) {  
    this.personaRepository.delete(persona);  
}  
  
1 usage  
  
@Override  
public List<Persona> getPersonas() {  
    List<Persona> personas =  
        this.personaRepository.findAll();  
    return personas;  
}
```

Method	Return Type
deleteById(Long id)	void
exists(Example<S> example)	boolean
existsById(Long id)	boolean
findAllById(Iterable<Long> ids)	List<Persona>
findOne(Example<S> example)	Optional<S>
flush()	void
getReferenceById(Long id)	Persona
saveAll(Iterable<S> entities)	List<S>
saveAllAndFlush(Iterable<S> entities)	List<S>
saveAndFlush(S entity)	S
getClass()	Class<? extends PersonaRepository>
notify()	void
notifyAll()	void
wait()	void
wait(long timeoutMillis)	void
wait(long timeoutMillis, int nanos)	void
deleteInBatch(Iterable<Persona> entities)	void
findById(Long id)	Optional<Persona>
findOne(Long id)	Optional<Persona>
± arg	functionCall(expr)
± cast	((SomeType) expr)

# Servicio

```
@Service
@Qualifier("db")
public class PersonaDBService implements PersonaService {
    @Autowired
    protected PersonaRepository personaRepository;
    @Override
    public Persona get(long id) {
        Optional<Persona> persona = this.personaRepository.findById(id);
        return persona.get();
    }

    @Override
    public void add(Persona persona) {
        this.personaRepository.save(persona);
    }
}
```

# Servicio

```
@Override
public void remove(long id) {
    Optional<Persona> borrar = this.personaRepository.findById(id);
    this.personaRepository.delete(borrar.get());
}
```

```
@Override
public void remove(Persona p) {
    this.personaRepository.delete(p);
}
```

```
@Override
public List<Persona> getAll() {
    List<Persona> personas = this.personaRepository.findAll();
    return personas;
}
```

```
}
```

# Controlador REST

```
@RestController
@RequestMapping ("/persona/")
public class PersonaController {

    @Autowired
    @Qualifier ("db")
    PersonaService personas;

    @GetMapping ("/{id}")
    public Persona get (@PathVariable Long id) {
        Persona p = this.personas.get (id);
        return p;
    }

    @GetMapping
    public List<Persona> getAll () {
        List<Persona> personas = this.personas.getAll ();
        return personas;
    }

    @PostMapping
    public Persona post (@RequestBody Persona persona) {
        this.personas.add (persona);
        return persona;
    }
}
```



# application.properties

```
# Enabling H2 Console
```

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

```
spring.data.jpa.repositories.bootstrap-mode=default
```

```
spring.jpa.defer-datasource-initialization=true
```

```
# H2 en memoria
```

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

# application.properties

```
# Enabling H2 Console
```

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

```
spring.data.jpa.repositories.bootstrap-mode=default
```

```
spring.jpa.defer-datasource-initialization=true
```

```
# H2 en disco
```

```
spring.datasource.url=jdbc:h2:file:./testdb;DB_CLOSE_ON_EXIT=FALSE;DB_CLOSE_
DELAY=-1;
```

```
spring.datasource.username=fer
```

```
spring.datasource.password=fer
```

```
spring.datasource.initialize=true
```

```
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
```

```
spring.jpa.hibernate.ddl-auto=update
```

# Repository - Consultas

```
public interface PersonaRepository extends JpaRepository<Persona, Long> {  
    //Creación de consultas  
    // #1 Métodos con nombres de campos- Derived Query  
    List<Persona> findByNombre(String nombre);  
  
    // #2 Usando JPQL  
    @Query("SELECT p from Persona p WHERE p.nombre = :nombre")  
    List<Persona> buscarPorNombre(@Param("nombre") String nombre);  
  
    // #3 Usando SQL Nativo  
    @Query(value = "SELECT * from persona WHERE nombre = :nombre", nativeQuery = true)  
    List<Persona> buscarPorNombreSQL(@Param("nombre") String nombre);  
  
    // Actualizando datos  
    @Modifying  
    @Transactional  
    @Query("UPDATE Persona p set p.apellido = :apellido WHERE p.id = :id")  
    int actualizarPersona(@Param("apellido") String apellido, @Param("id") Long id);  
}
```

# Repository - Derived Query

## Consultas usando campos String

```
List<Persona> findByNombre(String nombre);  
List<Persona> findByNombreIs(String nombre);  
List<Persona> findByNombreEquals(String nombre);  
List<Persona> findByNombreIsNot(String nombre);  
  
List<Persona> findByNombreIsNull();  
List<Persona> findByNombreIsNotNull();  
  
List<Persona> findByNombreStartingWith(String nombre);  
List<Persona> findByNombreEndingWith(String nombre);  
List<Persona> findByNombreContaining(String nombre);  
List<Persona> findByNombreLike(String nombre);  
List<Persona> findByNombreIn(Collection<String> nombres);
```

# Repository - Derived Query

## Consultas usando campos Boolean

```
List<Persona> findByBooleanoTrue();
```

```
List<Persona> findByBooleanoFalse();
```

## Consultas usando campos Boolean

```
List<Persona> findByFechaAfter(ZonedDateTime fecha);
```

```
List<Persona> findByFechaBefore(ZonedDateTime fecha);
```

# Repository - Derived Query

## Consultas usando campos Numéricos

```
List<Persona> findByNumeroLessThan(Integer numero);
```

```
List<Persona> findByNumeroLessThanEquals(Integer numero);
```

```
List<Persona> findByNumeroGreaterThan(Integer numero);
```

```
List<Persona> findByNumeroGreaterThanEquals(Integer numero);
```

```
List<Persona> findByNumeroBetween(Integer menor, Integer mayor);
```

```
List<Persona> findByNumeroIn(Collection<Integer> numeros);
```

# Repository - Derived Query

## Consultas combinando campos

```
List<Persona> findByNombreAndApellido(String nombre, String apellido);
```

```
List<Persona> findByNombreOrApellido(String nombre, String apellido);
```

```
List<Persona> findByNombreAndNumero(String nombre, Integer numero);
```

```
List<Persona> findByNombreAndApellidoAndNumeroBetween(String nombre, String Apellido,  
Integer menor, Integer mayor);
```

```
List<Persona> findByNombreOrApellidoAndNumeroBetweenAndBooleano(String nombre, String  
Apellido, Integer menor, Integer mayor, Boolean booleano);
```

# Repository - Derived Query

## Ordenando resultado

```
List<Persona> findByNombre(String nombre);  
List<Persona> findByNombreOrderByNombre(String nombre);  
List<Persona> findByNombreOrderByNombreAsc(String nombre);  
List<Persona> findByNombreOrderByNombreDesc(String nombre);
```

```
List<Persona> findByNumeroLessThan(Integer numero);  
List<Persona> findByNumeroLessThanOrderByNumero(Integer numero);  
List<Persona> findByNumeroLessThanOrderByNombre(Integer numero);
```

```
List<Persona> findByFechaAfter(ZonedDateTime fecha);  
List<Persona> findByFechaAfterOrderByFechaDesc(ZonedDateTime fecha);  
List<Persona> findByFechaAfterOrderByNombre(ZonedDateTime fecha);
```

```
List<Persona> findByNombreOrApellidoAndNumeroBetweenAndBooleano(String nombre, String  
Apellido, Integer menor, Integer mayor, Boolean booleano);  
List<Persona>  
findByNombreOrApellidoAndNumeroBetweenAndBooleanoOrderByNombreDesc(String nombre,  
String Apellido, Integer menor, Integer mayor, Boolean booleano);
```



# Relaciones - OneToMany

## Clase Automovil (muchos)

```
@Entity
public class Automovil {
    @Id
    @GeneratedValue
    protected long id;

    protected String marca;
    protected String modelo;
    protected Integer anio;
}

public interface AutomovilRepository
extends JpaRepository<Automovil, Long> {
}
```

## Clase Chofer (uno)

```
@Entity
public class Chofer {
    @Id
    @GeneratedValue
    protected Long id;

    protected String nombre;
    protected String apellido;
    protected Integer edad;

    @OneToMany
    protected List<Automovil> automoviles;
}

public interface ChoferRepository extends
JpaRepository<Chofer, Long> {
}
```

# Controller - Automovil

```
@RestController
@RequestMapping ("/automovil/")
public class AutomovilController {
    @Autowired
    AutomovilRepository automovilRepository;

    @GetMapping
    public List<Automovil> getAll () {
        List<Automovil> automoviles = this.automovilRepository.findAll ();
        return automoviles;
    }

    @GetMapping ("/{id}")
    public Automovil getPorId (@PathVariable Long id) {
        Optional<Automovil> automovil = this.automovilRepository.findById (id);
        return automovil.get ();
    }

    @PostMapping
    public Automovil post (@RequestBody Automovil automovil) {
        this.automovilRepository.save (automovil);
        return automovil;
    }
}
```

# Controller - Chofer

```
@RestController
@RequestMapping("/chofer/")
public class ChoferController {
    @Autowired
    ChoferRepository choferRepository;

    @GetMapping
    public List<Chofer> getAll() {
        List<Chofer> choferes = this.choferRepository.findAll();
        return choferes;
    }

    @GetMapping("/{id}")
    public Chofer getPorId(@PathVariable Long id) {
        Optional<Chofer> chofer = this.choferRepository.findById(id);
        return chofer.get();
    }
}
```

# Controller - Chofer

```
@Autowired
ChoferRepository choferRepository;

@Autowired
AutomovilRepository automovilRepository;

// Automovil
@PostMapping
public Automovil post(@RequestBody Automovil automovil) {
    this.automovilRepository.save(automovil);
    return automovil;
}

@PostMapping
public Chofer post(@RequestBody Chofer chofer) {
    this.automovilRepository.saveAll(chofer.getAutomoviles());
    this.choferRepository.save(chofer);
    return chofer;
}
```

# Repository - Derived Query

## Repository

```
public interface ChoferRepository extends JpaRepository<Chofer, Long> {  
    List<Chofer> findByAutomoviles(Automovil automovil);  
    List<Chofer> findByAutomovilesMarca(String marca);  
}
```

## Controlador

```
@GetMapping("/automovil/{id}")  
public List<Chofer> getChoferPorAutomovil(@PathVariable Long id) {  
    Optional<Automovil> auto = this.automovilRepository.findById(id);  
    Automovil auto2 = this.automovilRepository.findById(id).get();  
    List<Chofer> choferes = this.choferRepository.findByAutomoviles(auto2);  
    List<Chofer> choferes2 = this.choferRepository.findByAutomovilesMarca(auto2.getMarca());  
    return choferes;  
}
```