



# **Programación Orientada A Objetos II**

**Jaime**

**UAZ**

Examen 3  
Crud con Spring Boot

Jesús Adrián Arias Delgado

Para empezar se crea una estructura de proyecto utilizando la página <https://start.spring.io/>

Aquí podemos definir las dependencias a utilizar y elegimos la versión de java que utilizaremos.

La dependencia de JPA es para poder utilizar las anotaciones de este formato.

Spring web nos facilita la creación del formato web.

Thymeleaf nos permite utilizar archivos HTML para la creación del sitio web.

The image shows the Spring Boot project generator form. It is divided into two main sections: Project Configuration and Dependencies.

**Project Configuration:**

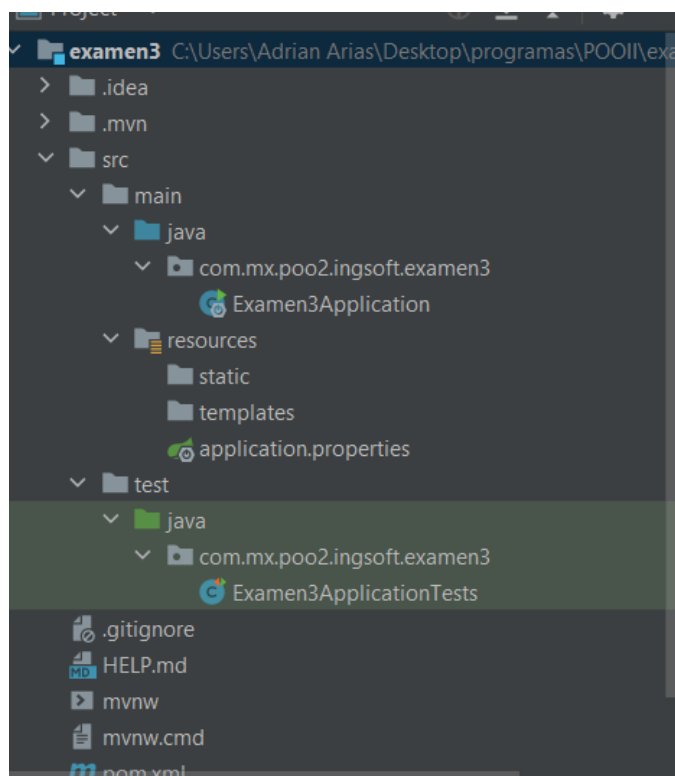
- Project:** Radio buttons for Gradle - Groovy, Gradle - Kotlin, and Maven (selected).
- Language:** Radio buttons for Java (selected), Kotlin, and Groovy.
- Spring Boot:** Radio buttons for 3.1.1 (SNAPSHOT), 3.1.0 (selected), 3.0.8 (SNAPSHOT), and 3.0.7.
- Spring Boot:** Radio buttons for 2.7.13 (SNAPSHOT) and 2.7.12.
- Project Metadata:**
  - Group: com.mx.poo2.ingsoft
  - Artifact: Examen3
  - Name: Examen3
  - Description: Demo project for Spring Boot
  - Package name: com.mx.poo2.ingsoft.Examen3
- Packaging:** Radio buttons for Jar (selected) and War.
- Java:** Radio buttons for 20, 17 (selected), 11, and 8.

**Dependencies:**

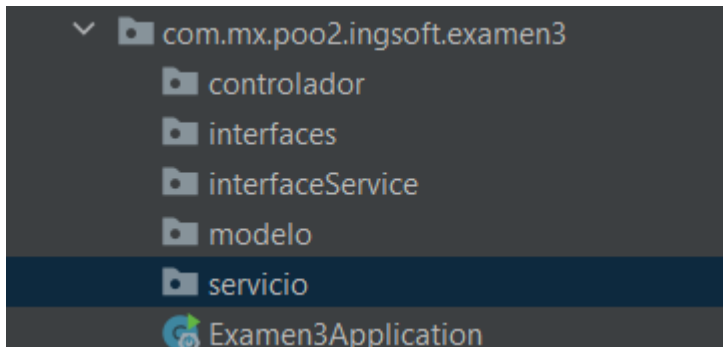
- Spring Data JPA:** SQL. Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.
- Spring Web:** WEB. Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
- Thymeleaf:** TEMPLATE ENGINES. A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.
- MySQL Driver:** SQL. MySQL JDBC driver.

Buttons: ADD DEPENDENCIES... CTRL + B

Lo siguiente es abrir el proyecto con nuestro IDE de preferencia, en mi caso utilizaré IntelliJ IDEA.



Utilizaremos el patrón MVC, por lo que deberemos de crear la capa modelo, vista y controlador.



Se creará la clase Persona para ver los datos ya que es el ejemplo más común. Y se ponen las anotaciones de JPA.

```
public class Persona {  
    3 usages  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private int id;  
    3 usages  
    @Column  
    private String nombre;  
    no usages  
    public Persona(){  
    }  
    no usages  
    public Persona(int id, String nombre){  
        super();  
        this.id = id;  
        this.nombre = nombre;  
    }  
    no usages  
    public int getId() {  
        return id;  
    }  
    no usages  
    public void setId(int id) {  
        this.id = id;  
    }  
    no usages  
    public String getNombre() {
```

Después crearemos el servicio para la persona que contiene los métodos del CRUD. La crearemos con su respectiva interfaz.

```
public class PersonaService implements IPersonaService {

    no usages
    @Override
    public List<Persona> listar() {
        return null;
    }

    no usages
    @Override
    public Optional<Persona> listarId(int id) {
        return Optional.empty();
    }

    no usages
    @Override
    public int save(Persona p) {
        return 0;
    }

    no usages
    @Override
    public void delete() {
    }

}

package com.mx.poo2.ingsoft.examen3.interfaceService;

import com.mx.poo2.ingsoft.examen3.modelo.Persona;

import java.util.List;
import java.util.Optional;

public interface IPersonaService {

    public List<Persona> listar();

    public Optional<Persona> listarId(int id);

    public int save(Persona p);

    public void delete();

}
```

Agregamos las anotaciones necesarias de SpringBoot

```
@Service
public class PersonaService implements IPersonaService {

    1 usage
    @Autowired
    private IPersona dato;

    no usages
    @Override
    public List<Persona> listar() {
        return (List<Persona>) dato.findAll();
    }

}
```

Luego creamos el controlador y agregamos sus respectivas anotaciones de Spring Boot. Esto nos permitirá utilizar una dirección para realizar una operación.

```
@Controller
@RequestMapping
public class Controlador {

    1 usage
    @Autowired
    private IPersonaService service;

    no usages
    @GetMapping("/listar")
    public String listar(Model modelo){
        List<Persona> personas = service.listar();
        modelo.addAttribute("personas", personas);
        return "index";
    }

}
```

Creamos nuestra base de datos

```
mysql> CREATE TABLE PERSONA(  
  -> id INT PRIMARY KEY,  
  -> nombre varchar(45));  
Query OK, 0 rows affected (0.03 sec)  
  
mysql>
```

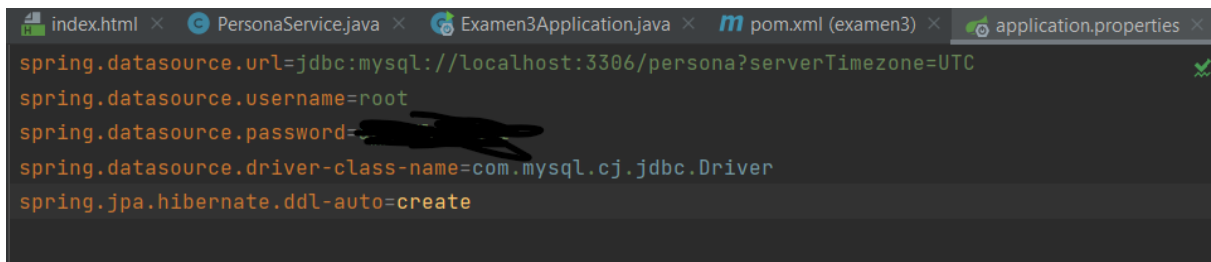
También la llenamos

```
mysql> INSERT INTO persona VALUES(1,'Jaime');  
Query OK, 1 row affected (0.01 sec)  
  
mysql> INSERT INTO persona VALUES(2,'juan');  
Query OK, 1 row affected (0.00 sec)  
  
mysql> INSERT INTO persona VALUES(3,'Pablo');  
Query OK, 1 row affected (0.00 sec)
```

Y luego modificamos el archivo HTML que nos permite modificar la vista, agregando el thymeleaf para poder utilizar los datos de persona.

```
<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"  
      integrity="sha384-9ndCyUaIbzAi2FUVXJi0CjMCapSm07SnPjef0486qhlNuZ2cdeRh002iuk6FUUVM" crossorigin="anonymous">  
<head>  
  <meta charset="UTF-8">  
  <title>Title</title>  
</head>  
<body>  
  <div class="container mt-4">  
    <table class="table">  
      <thead class="table-dark"><tr><th>ID</th></tr></thead>  
      <thead class="table-dark"><tr><th>NOMBRE</th></tr></thead>  
      <thead class="table-dark"><tr><th>TELEFONO</th></tr></thead>  
      <tbody>  
        <tr th:each="persona: ${personas}">  
          <td th:text="${persona.id}">${persona.id}</td>  
          <td th:text="${persona.nombre}">${persona.nombre}</td>  
          <td th:text="${persona.telefono}">${persona.telefono}</td>  
        </tr>  
      </tbody>  
    </table>  
  </div>  
</body>  
</html>
```

Modificamos las propiedades de la aplicación



The screenshot shows an IDE window with several tabs: index.html, PersonaService.java, Examen3Application.java, pom.xml (examen3), and application.properties. The application.properties file is active, displaying the following configuration:

```
spring.datasource.url=jdbc:mysql://localhost:3306/persona?serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=[REDACTED]
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.hibernate.ddl-auto=create
```

Y ya está listo el proyecto.