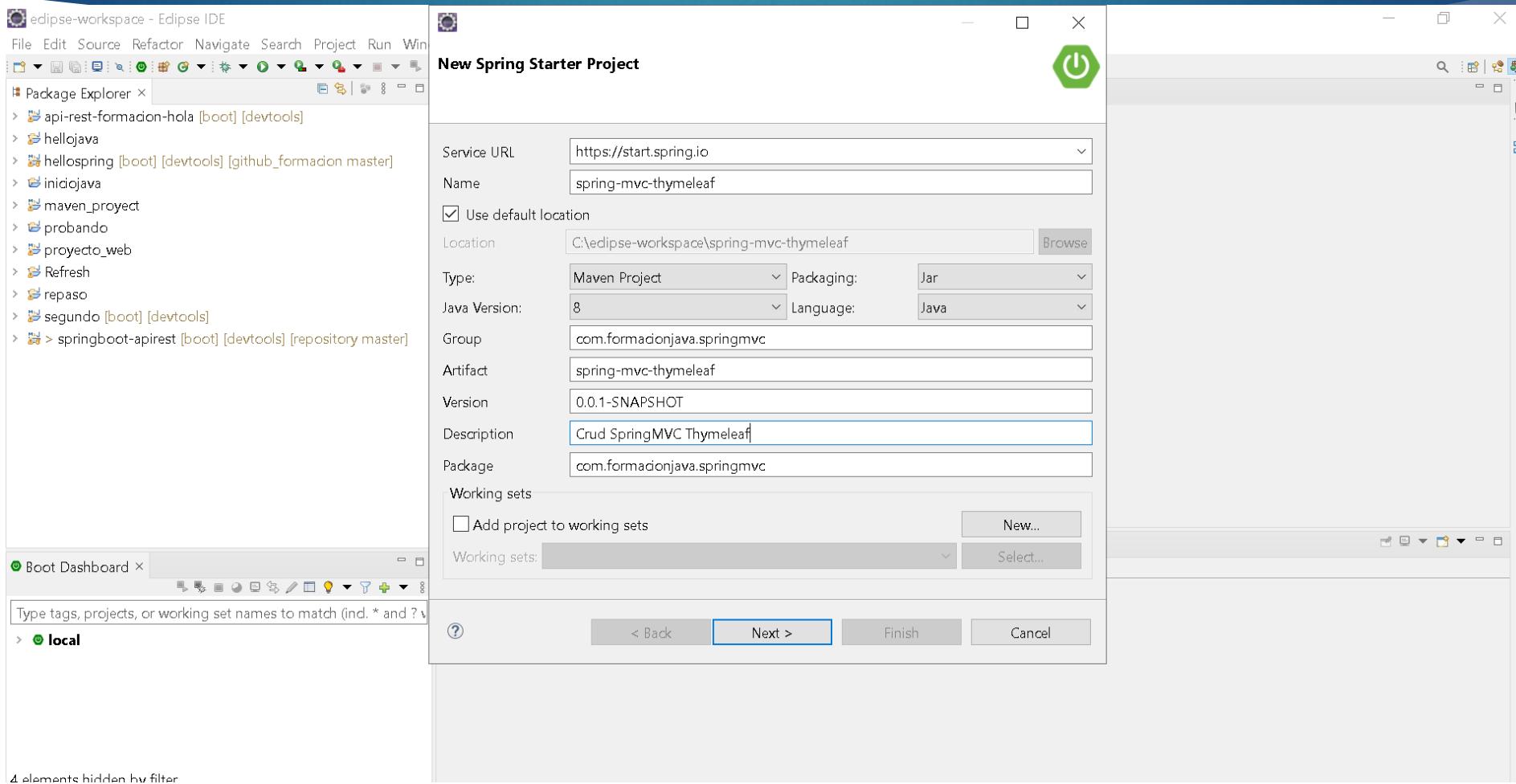


Thymeleaf

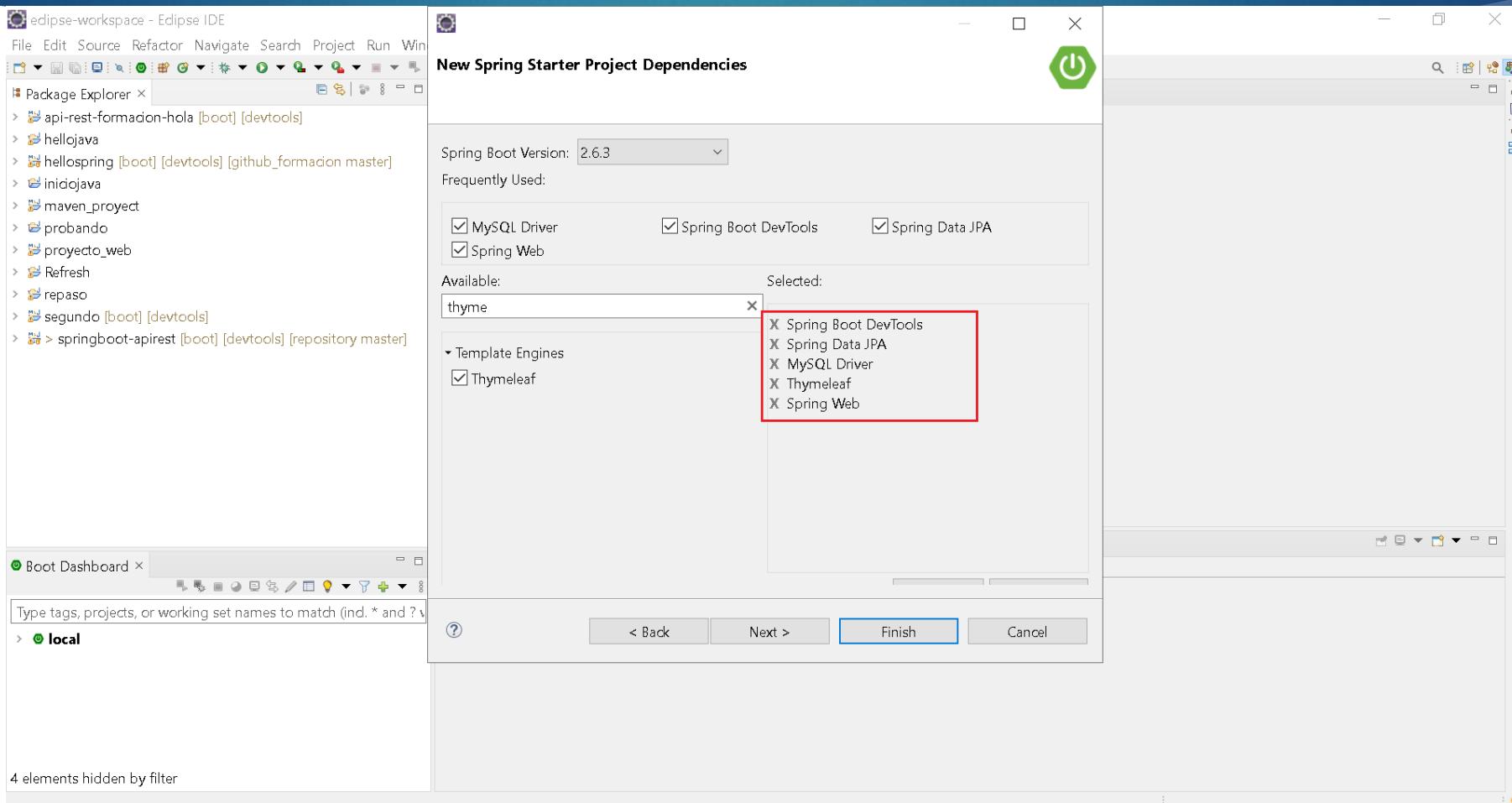
- ▶ **Thymeleaf** es un motor de plantillas de Java para procesar y crear HTML, XML, JavaScript, CSS y texto.

- ▶ <https://www.thymeleaf.org/>

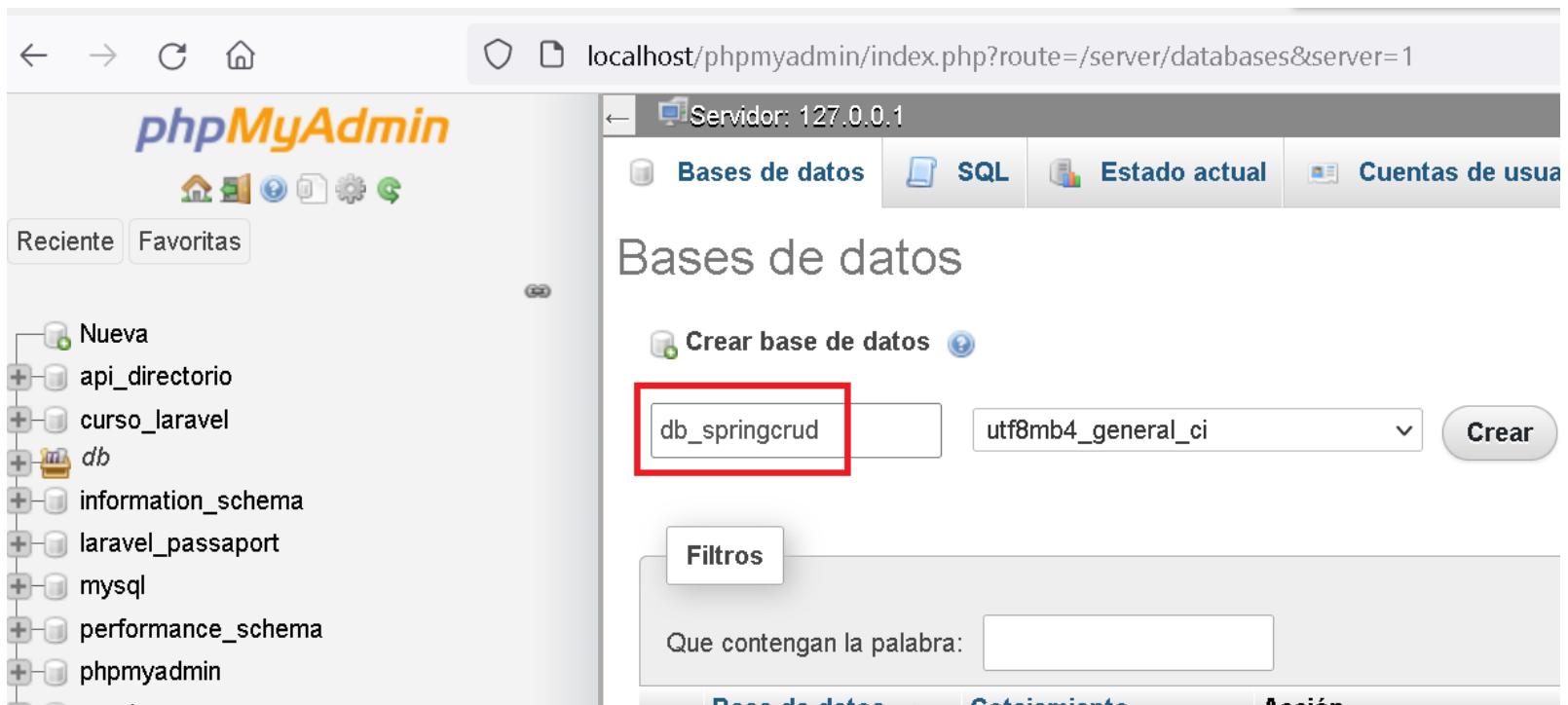
Creamos un nuevo proyecto



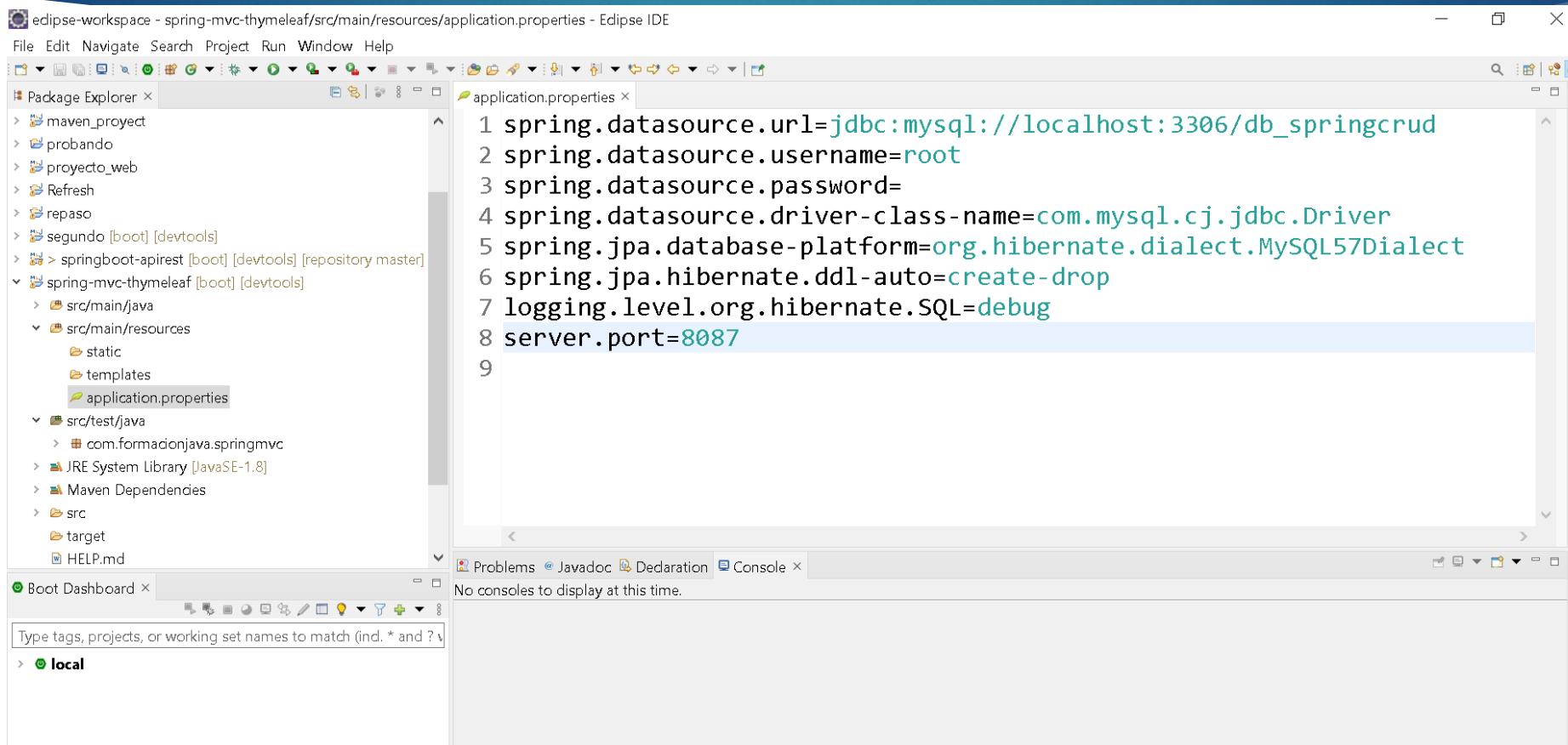
Agregamos las siguientes dependencias



Creamos la base de datos



Configuramos el application.properties



The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - spring-mvc-thymeleaf/src/main/resources/application.properties - Edipse IDE". The menu bar includes File, Edit, Navigate, Search, Project, Run, Window, Help. The left sidebar is the Package Explorer showing projects like maven_proyect, probando, proyecto_web, Refresh, repaso, segundo [boot] [devtools], springboot-apirest [boot] [devtools] [repository master], and spring-mvc-thymeleaf [boot] [devtools]. The main editor window displays the application.properties file with the following content:

```
1 spring.datasource.url=jdbc:mysql://localhost:3306/db_springcrud
2 spring.datasource.username=root
3 spring.datasource.password=
4 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
5 spring.jpa.database-platform=org.hibernate.dialect.MySQL57Dialect
6 spring.jpa.hibernate.ddl-auto=create-drop
7 logging.level.org.hibernate.SQL=debug
8 server.port=8087
9
```

The bottom status bar says "No consoles to display at this time."

Creamos nuestra entidad

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** edipse-workspace - spring-mvc-thymeleaf/src/main/java/com/formacionjava/springmvc/entity/Trabajador.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar with various icons for file operations.
- Package Explorer:** Shows the project structure:
 - mvnw
 - mvnw.cmd
 - pom.xml
 - spring-mvc-thymeleaf [boot] [devtools]
 - src/main/java
 - com.formacionjava.springmvc
 - com.formacionjava.springmvc.dao
 - com.formacionjava.springmvc.entity
 - Trabajador.java
 - src/main/resources
 - src/test/java
 - JRE System Library [JavaSE-1.8]
 - Maven Dependencies
 - src
 - target
 - HELP.md
 - mvnw
 - mvnw.cmd
 - pom.xml
- Editor:** The main editor window displays the code for `Trabajador.java`. The code defines a class `Trabajador` that implements `Serializable`. It includes annotations for `@Entity`, `@Table(name="empleado")`, `@Id`, `@GeneratedValue(strategy = GenerationType.IDENTITY)`, and three `@Column` annotations for `nombre`, `apellido`, and `email`. A private attribute `telefono` is also declared.

```
2
3+import java.io.Serializable;■
11
12 @Entity
13 @Table(name="empleado")
14 public class Trabajador implements Serializable {
15
16@ Id
17 @GeneratedValue(strategy = GenerationType.IDENTITY)
18 private Long id;
19
20@ Column(nullable= false, length= 50)
21 private String nombre;
22
23@ Column(nullable= false, length= 50)
24 private String apellido;
25
26@ Column(nullable= false, length= 50,unique = true)
27 private String email;
28
29 private int telefono;
30
31
32
```
- Bottom Bar:** Shows tabs for Problems, Javadoc, Declaration, and Console. The Console tab is selected, displaying the message "No consoles to display at this time."

eclipse-workspace - spring-mvc-thymeleaf/src/main/java/com/formacionjava/springmvc/entity/Trabajador.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer × Trabajador.java ×

```
44
45  public Long getId() {
46      return id;
47  }
48  public void setId(Long id) {
49      this.id = id;
50  }
51  public String getNombre() {
52      return nombre;
53  }
54  public void setNombre(String nombre) {
55      this.nombre = nombre;
56  }
57  public String getApellido() {
58      return apellido;
59  }
60  public void setApellido(String apellido) {
61      this.apellido = apellido;
62  }
63  public String getEmail() {
64      return email;
65  }
66  public void setEmail(String email) {
67      this.email = email;
```

Boot Dashboard ×

Type tags, projects, or working set names to match (ind)

> local

Problems Javadoc Declaration Console ×

No consoles to display at this time.

4 elements hidden by filter

edipse-workspace - spring-mvc-thymeleaf/src/main/java/com/formacionjava/springmvc/entity/Trabajador.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help



Package Explorer ×

```
mvnw  
mvnw.cmd  
pom.xml  
spring-mvc-thymeleaf [boot] [devtools]  
src/main/java  
com.formacionjava.springmvc  
com.formacionjava.springmvc.dao  
com.formacionjava.springmvcentity  
Trabajador.java  
src/main/resources  
src/test/java  
JRE System Library [JavaSE-1.8]  
Maven Dependencies  
src  
target  
HELP.md  
mvnw  
mvnw.cmd  
pom.xml
```

Boot Dashboard ×

Type tags, projects, or working set names to match (in)

> local

```
*Trabajador.java ×  
60  public void setApellido(String apellido) {  
61      this.apellido = apellido;  
62  }  
63  public String getEmail() {  
64      return email;  
65  }  
66  public void setEmail(String email) {  
67      this.email = email;  
68  }  
69  public int getTelefono() {  
70      return telefono;  
71  }  
72  public void setTelefono(int telefono) {  
73      this.telefono = telefono;  
74  }  
75  
76  /**  
80  *  
81  */  
82  private static final long serialVersionUID = 1L;  
83
```

Problems Javadoc Declaration Console ×

No consoles to display at this time.

Creamos un repositorio dao

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - spring-mvc-thymeleaf/src/main/java/com/formacionjava/springmvc/dao/TrabajadorRepositorio.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar icons.
- Package Explorer:** Shows the project structure:
 - repaso
 - segundo [boot] [devtools]
 - springboot-apirest [boot] [devtools] [repository]
 - spring-mvc-thymeleaf [boot] [devtools]
 - src/main/java
 - com.formacionjava.springmvc
 - com.formacionjava.springmvc.dao
 - TrabajadorRepositorio.java
 - com.formacionjava.springmvc.entity
 - src/main/resources
 - src/test/java
 - JRE System Library [JavaSE-1.8]
 - Maven Dependencies
 - src
 - target
 - HELP.md
 - mvnw
 - mvnw.cmd
 - pom.xml
- Editor:** The file TrabajadorRepositorio.java is open, showing the following code:

```
1 package com.formacionjava.springmvc.dao;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import org.springframework.stereotype.Repository;
5
6 import com.formacionjava.springmvc.entity.Trabajador;
7
8 @Repository
9 public interface TrabajadorRepositorio extends JpaRepository<Trabajador, Long> {
10
11 }
```
- Bottom Bar:** Problems, Javadoc, Declaration, Console. The Console tab shows: "No consoles to display at this time."

Ahora creamos los servicios

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - spring-mvc-thymeleaf/src/main/java/com/formacionjava/springmvc/service/TrabajadorServices.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar icons.
- Package Explorer:** Shows the project structure:
 - iniciojava
 - maven_proyect
 - probando
 - projeto_web
 - Refresh
 - repaso
 - segundo [boot] [devtools]
 - springboot-apirest [boot] [devtools] [repository mas]
 - spring-mvc-thymeleaf [boot] [devtools]
 - src/main/java
 - com.formacionjava.springmvc
 - com.formacionjava.springmvc.controllers
 - com.formacionjava.springmvc.dao
 - com.formacionjava.springmvc.entity
 - com.formacionjava.springmvc.service
 - TrabajadorServices.java (highlighted with a red box)
 - TrabajadorServiceImpl.java
 - src/main/resources
 - static
 - templates
 - application.properties
 - import.sql
 - src/test/java
 - JRE System Library [JavaSE-1.8]
 - Maven Dependencies
 - Editor:** Displays the code for `TrabajadorServices.java`:

```
1 package com.formacionjava.springmvc.service;
2
3 import java.util.List;
4
5 public interface TrabajadorServices {
6
7     public List<Trabajador> listarTodosLosTrabajadores();
8
9 }
10
11
12 }
```

eclipse-workspace - spring-mvc-thymeleaf/src/main/java/com/formacionjava/springmvc/service/TrabajadorServicesImpl.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer TrabajadorServicesImpl.java

```
1 package com.formacionjava.springmvc.service;
2
3 import java.util.List;
4
5 @Service
6 public class TrabajadorServicesImpl implements TrabajadorServices {
7
8     @Autowired
9     private TrabajadorRepository repositorio;
10
11     @Override
12     public List<Trabajador> listarTodosLosTrabajadores() {
13         return repositorio.findAll();
14     }
15 }
```

Problems Javadoc Declaration Console

<terminated> spring-mvc-thymeleaf - SpringMvcThymeleafApplication [Spring Boot App] C:\Users\dell\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - spring-mvc-thymeleaf/src/main/java/com/formacionjava/springmvc/service/TrabajadorServicesImpl.java - Eclipse IDE
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Standard Eclipse toolbar icons.
- Package Explorer:** Shows the project structure. A red box highlights the package `com.formacionjava.springmvc.service` and its file `TrabajadorServicesImpl.java`.
- Editor:** Displays the Java code for `TrabajadorServicesImpl.java`. The code implements the `TrabajadorServices` interface, uses `@Service` and `@Autowired` annotations, and overrides the `listarTodosLosTrabajadores` method to return all `Trabajador` objects from the `repositorio` repository.
- Bottom Status Bar:** Shows the status message: <terminated> spring-mvc-thymeleaf - SpringMvcThymeleafApplication [Spring Boot App] C:\Users\dell\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64

Creamos el controlador

The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - spring-mvc-thymeleaf/src/main/java/com/formacionjava/springmvc/controllers/TrabajadorController.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations. The left side features the Package Explorer view, which lists several projects and their contents. A red box highlights the "src/main/java/com.formacionjava.springmvc.controllers.TrabajadorController.java" entry. The right side is the code editor window, also with a red box around its title bar. The code itself is:

```
1 package com.formacionjava.springmvc.controllers;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.stereotype.Controller;
5 import org.springframework.ui.Model;
6 import org.springframework.web.bind.annotation.GetMapping;
7
8 import com.formacionjava.springmvc.service.TrabajadorServices;
9
10
11 @Controller
12 public class TrabajadorController {
13
14     @Autowired
15     private TrabajadorServices servicio;
16
17     @GetMapping={"/trabajadores","/"})
18     public String listarTrabajadores(Model modelo) {
19         modelo.addAttribute("trabajador",servicio.listarTodosLosTrabajadores());
20         return "trabajador";
21     }
22
23
24 }
```

Modelo

- ▶ El modelo puede proporcionar atributos utilizados para representar vistas

Creamos los datos de pruebas

The screenshot shows a Java IDE interface with the following components:

- Package Explorer:** On the left, it lists several projects and their contents:
 - iniciojava
 - maven_proyect
 - probando
 - proyecto_web
 - Refresh
 - repaso
 - segundo [boot] [devtools]
 - > springboot-apirest [boot] [devtools] [repository mas]
 - > spring-mvc-thymeleaf [boot] [devtools]
 - src/main/java
 - > com.formacionjava.springmvc
 - > com.formacionjava.springmvc.controllers
 - > TrabajadorController.java
 - > com.formacionjava.springmvc.dao
 - > com.formacionjava.springmvc.entity
 - > com.formacionjava.springmvc.service
 - src/main/resources
 - static
 - templates
 - application.properties
 - import.sql
 - > src/test/java
 - > JRE System Library [JavaSE-1.8]
- Connection profile:** A panel titled "Connection profile" with fields for Type, Name, Database, and Status (Disconnected, Auto).
- import.sql:** An open SQL file containing four `INSERT INTO` statements to insert data into the "empleado" table.

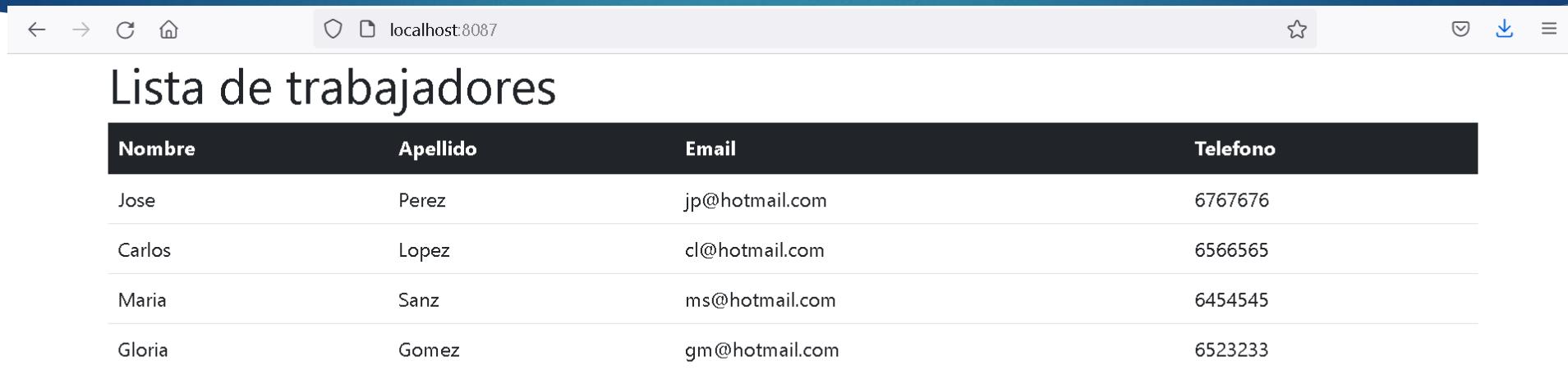
```
1 INSERT INTO empleado (nombre,apellido,email,telefono) VALUES('Jose','Perez','jp@hotmail.com',6767676);
2 INSERT INTO empleado (nombre,apellido,email,telefono) VALUES('Carlos','Lopez','cl@hotmail.com',6566565);
3 INSERT INTO empleado (nombre,apellido,email,telefono) VALUES('Maria','Sanz','ms@hotmail.com',6454545);
4 INSERT INTO empleado (nombre,apellido,email,telefono) VALUES('Gloria','Gomez','gm@hotmail.com',6523233);
```

Ahora agregamos la plantilla

The screenshot shows the Eclipse IDE interface. The title bar reads "eclipse-workspace - spring-mvc-thymeleaf/src/main/resources/templates/trabajador.html - Eclipse IDE". The menu bar includes File, Edit, Source, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations. The left side features the Package Explorer with a tree view of project files. A red box highlights the "templates" folder under "src/main/resources" which contains "trabajador.html". The main editor window displays the content of "trabajador.html". A red box highlights the head section of the HTML code, specifically the link to Bootstrap CSS. The code uses Thymeleaf syntax to display a list of workers.

```
<!DOCTYPE html>
<html xmlns:th="https://www.thymeleaf.org">
<head>
<meta charset="utf-8">
<title>Listado de trabajadores</title>
<!-- CSS only -->
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">
</head>
<body>
    <div class="container">
        <div class="row">
            <h1>Lista de trabajadores</h1>
        </div>
        <table class="table">
            <thead class="table-dark">
                <tr>
                    <th>Nombre</th>
                    <th>Apellido</th>
                    <th>Email</th>
                    <th>Telefono</th>
                </tr>
            </thead>
            <tbody>
                <tr th:each="trabajador: ${trabajador}">
                    <td th:text="${trabajador.nombre}>Nombre</td>
                    <td th:text="${trabajador.apellido}>Apellido</td>
                    <td th:text="${trabajador.email}>Email</td>
                    <td th:text="${trabajador.telefono}>Telefono</td>
                </tr>
            </tbody>
        </table>
    </div>
</body>
</html>
```

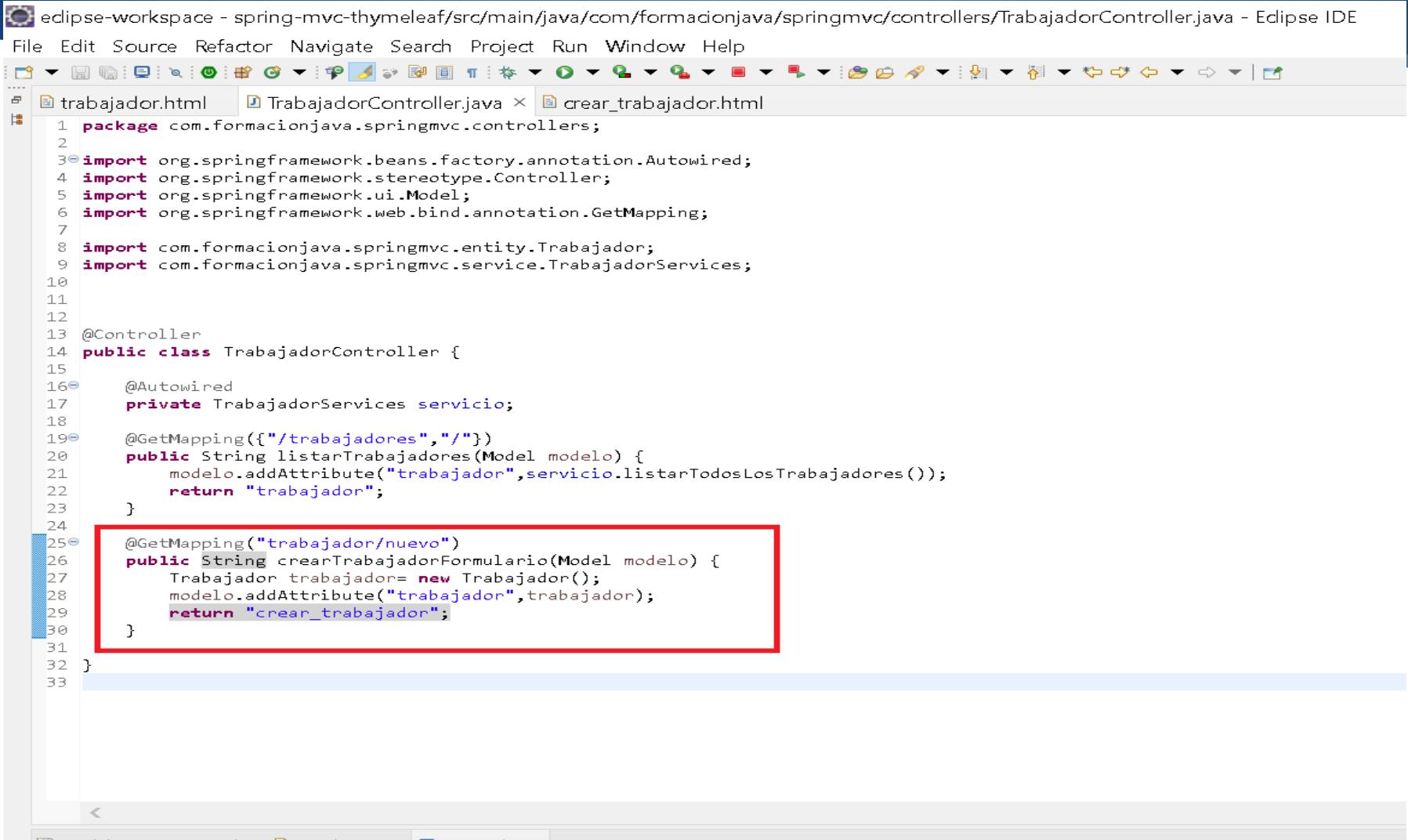
Ejecutamos y probamos todo hasta aquí



A screenshot of a web browser window displaying a table of workers. The browser's address bar shows "localhost:8087". The table has four columns: Nombre, Apellido, Email, and Telefono. The data is as follows:

Nombre	Apellido	Email	Telefono
Jose	Perez	jp@hotmail.com	6767676
Carlos	Lopez	cl@hotmail.com	6566565
Maria	Sanz	ms@hotmail.com	6454545
Gloria	Gomez	gm@hotmail.com	6523233

Ahora me creo un nuevo controlador



The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - spring-mvc-thymeleaf/src/main/java/com/formacionjava/springmvc/controllers/TrabajadorController.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar below has various icons for file operations. The editor tab bar shows "trabajador.html", "TrabajadorController.java", and "crear_trabajador.html". The code editor displays Java code for a controller:

```
1 package com.formacionjava.springmvc.controllers;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.stereotype.Controller;
5 import org.springframework.ui.Model;
6 import org.springframework.web.bind.annotation.GetMapping;
7
8 import com.formacionjava.springmvc.entity.Trabajador;
9 import com.formacionjava.springmvc.service.TrabajadorServices;
10
11
12
13 @Controller
14 public class TrabajadorController {
15
16     @Autowired
17     private TrabajadorServices servicio;
18
19     @GetMapping={"/trabajadores","/"})
20     public String listarTrabajadores(Model modelo) {
21         modelo.addAttribute("trabajador",servicio.listarTodosLosTrabajadores());
22         return "trabajador";
23     }
24
25     @GetMapping("trabajador/nuevo")
26     public String crearTrabajadorFormulario(Model modelo) {
27         Trabajador trabajador= new Trabajador();
28         modelo.addAttribute("trabajador",trabajador);
29         return "crear_trabajador";
30     }
31
32 }
```

A red rectangular box highlights the code block from line 25 to 30, which defines a new method for creating a new employee form.

Agrego un navbar a mi html

edipse-workspace - spring-mvc-thymeleaf/src/main/resources/templates/trabajador.html - Eclipse IDE

File Edit Source Navigate Project Run Window Help

trabajador.html x TrabajadorController.java crear_trabajador.html

```
1 <!DOCTYPE html>
2 @html xmlns:th="https://www.thymeleaf.org/">
3 <head>
4   <meta charset="utf-8">
5   <title>Listado de trabajadores</title>
6   <!-- CSS only -->
7   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EqzqF4YVdCiI4qQz0WEfXKjzD0Qw0u8ZPj5q/KO24P8vSIo8gI4JdQXp9tZJWZ" crossorigin="anonymous">
8 
9 
10 </head>
11 <body>
12 
13   <nav class="navbar navbar-expand-md navbar-light bg-light">
14     <div class="container-fluid">
15       <a class="navbar-brand">Trabajadores</a>
16       <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
17         <span class="navbar-toggler-icon"></span>
18       </button>
19       <div class="collapse navbar-collapse" id="navbarNav">
20         <ul class="navbar-nav">
21           <li class="nav-item">
22             <a class="nav-link active" aria-current="page" th:href="@{/trabajadores}">Trabajadores</a>
23           </li>
24           <li class="nav-item">
25             <a class="nav-link" th:href="@{/trabajador/nuevo}">Nuevo Trabajador</a>
26           </li>
27 
28         </ul>
29       </div>
30     </div>
31   </nav>
32 
33 
34 
35   <div class="container">
36     <div class="row">
37       <h1>Lista de trabajadores</h1>
38     </div>
39     <table class="table table-striped table-bordered">
40       <thead>
41         <tr>
42           <th>Nombre</th>
43           <th>Apellido</th>
44           <th>Edad</th>
45           <th>Sexo</th>
46           <th>Acciones</th>
47         </tr>
48       </thead>
49       <tbody>
50       </tbody>
51     </table>
52   </div>
```

Creo un nuevo html crear_trabajador

The screenshot shows the Eclipse IDE interface with the title bar "edipse-workspace - spring-mvc-thymeleaf/src/main/resources/templates/crear_trabajador.html - Edipse IDE". The menu bar includes File, Edit, Source, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations. The left sidebar shows project files: "trabajador.html", "TrabajadorController.java", and "crear_trabajador.html" (which is selected and highlighted with a red border). The main editor area displays the content of "crear_trabajador.html":

```
1 <!DOCTYPE html >
2 <html xmlns:th="https://www.thymeleaf.org/">
3 <head>
4 <meta charset="utf-8">
5 <title>Listado de trabajadores</title>
6 <!-- CSS only -->
7 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
8 rel="stylesheet" integrity="sha384-1BmE4kWBq78iYhFLdvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
9 crossorigin="anonymous">
10 </head>
11 <body>
12
13
14 <nav class="navbar navbar-expand-md navbar-light bg-light">
15   <div class="container-fluid">
16     <a class="navbar-brand" href="#">Trabajadores</a>
17     <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
18       <span class="navbar-toggler-icon"></span>
19     </button>
20     <div class="collapse navbar-collapse" id="navbarNav">
21       <ul class="navbar-nav">
22         <li class="nav-item">
23           <a class="nav-link active" href="#" th:href="@{/trabajadores}">Trabajadores</a>
24         </li>
25         <li class="nav-item">
26           <a class="nav-link" href="#">Nuevo Trabajador</a>
27         </li>
28       </ul>
29     </div>
30   </div>
31 </nav>
32
33
34
35
36 <div class="container">
37   <div class="row">
38     <div class="col-lg-6 col-md-6 col-sm-6 container justify-content-center card">
```

The status bar at the bottom shows "Problems @ Javadoc Declaration Console" and "spring-mvc-thymeleaf - SpringMvcThymeleafApplication [Spring Boot App] C:\Users\dell\p2\pool\plugins\org.eclipse.jdt.openjdk.hotspot.jre.full.win32.x86_64_16.0.2.v20210721-1149\jre\bin\javaw.exe (2 feb 2022 7:15:21)".

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** On the left, under the package `spring-mvc-thymeleaf`, the `src/main/resources/templates` folder is expanded, showing files `crear_trabajador.html` and `trabajador.html`. The file `crear_trabajador.html` is currently selected and highlighted with a red box.
- Code Editor:** The main window displays the content of the `crear_trabajador.html` file. The code is a Thymeleaf template for creating a new employee. It includes a container div, a form for inputting employee details (Nombre, Apellido, Email, Telefono), and a footer with a "Guardar" button.

```
<div class="container">
    <div class="row">
        <div class="col-lg-6 col-md-6 col-sm-6 justify-content-center card">
            <h1 class="text-center" >Registro de trabajador</h1>
            <div class="card-body">
                <form th:action="@{/trabajador}" th:object="${trabajador}" method="POST" >
                    <div class="form-group">
                        <label>Nombre:</label>
                        <input type="text" name="nombre" th:field="*{nombre}" class="form-control" placeholder="Ingrese su nombre" required>
                    </div>
                    <div class="form-group">
                        <label>Apellido:</label>
                        <input type="text" name="apellido" th:field="*{apellido}" class="form-control" placeholder="Ingrese su apellido" required>
                    </div>
                    <div class="form-group">
                        <label>Email:</label>
                        <input type="email" name="email" th:field="*{email}" class="form-control" placeholder="Ingrese su email" required>
                    </div>
                    <div class="form-group">
                        <label>Telefono:</label>
                        <input type="number" name="telefono" th:field="*{telefono}" class="form-control" placeholder="Ingrese su telefono" required>
                    </div>
                    <br>
                    <div class="box-footer">
                        <button class="btn btn-success">Guardar</button>
                    </div>
                </form>
            </div>
        </div>
    </div>
</div>
<body>
```

Probamos la url y la vista

The screenshot shows a web browser window with the following details:

- Address Bar:** localhost:8087/trabajador/nuevo
- Page Title:** Registro de trabajador
- Form Fields:**
 - Nombre: Ingrese su nombre
 - Apellido: Ingrese su apellido
 - Email: Ingrese su email
 - Telefono: Ingrese su telefono
- Buttons:** A green "Guardar" button at the bottom left.

Agregamos al servicio el método para realizar registro

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** edipse-workspace - spring-mvc-thymeleaf/src/main/java/com/formationjava/springmvc/service/TrabajadorServices.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar with icons for file operations.
- Package Explorer:** Shows the project structure:
 - api-rest-formacion-hola [boot] [devtools]
 - hellojava
 - hellospring [boot] [devtools] [github_formacion]
 - iniciojava
 - maven_proyect
 - probando
 - projeto_web
 - Refresh
 - repasso
 - segundo [boot] [devtools]
 - springboot-apirest [boot] [devtools] [repository]
 - spring-mvc-thymeleaf [boot] [devtools]**
 - src/main/java
 - com.formationjava.springmvc
 - com.formationjava.springmvc.controllers
 - com.formationjava.springmvc.dao
 - com.formationjava.springmvc.entity
 - com.formationjava.springmvc.service**
 - TrabajadorServices.java**
 - TrabajadorServicesImpl.java
 - src/main/resources
 - src/test/java
 - JRE System Library [JavaSE-1.8]
 - Maven Dependencies
 - src
 - target
 - HELP.md
 - mvnw
 - mvnw.cmd
 - pom.xml
- Editor:** Displays the content of TrabajadorServices.java:

```
1 package com.formationjava.springmvc.service;
2
3 import java.util.List;
4
5 public interface TrabajadorServices {
6
7     public List<Trabajador> listarTodosLosTrabajadores();
8
9     public Trabajador guardarTrabajador(Trabajador trabajador);
10
11 }
12
13
```

The method `public Trabajador guardarTrabajador(Trabajador trabajador);` is highlighted with a red box.
- Bottom Status Bar:** Problems, Javadoc, Declaration, Console
- Bottom Footer:** spring-mvc-thymeleaf - SpringMvcThymeleafApplication [Spring Boot App] C:\Users\dell\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full

edipse-workspace - spring-mvc-thymeleaf/src/main/java/com/formacionjava/springmvc/service/TrabajadorServicesImpl.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer TrabajadorServices.java TrabajadorServicesImpl.java

```
1 package com.formacionjava.springmvc.service;
2
3 import java.util.List;
4
5 @Service
6 public class TrabajadorServicesImpl implements TrabajadorServices {
7
8     @Autowired
9     private TrabajadorRepositorio repositorio;
10
11    @Override
12    public List<Trabajador> listarTodosLosTrabajadores() {
13        return repositorio.findAll();
14    }
15
16    @Override
17    public Trabajador guardarTrabajador(Trabajador trabajador) {
18        return repositorio.save(trabajador);
19    }
20
21}
22
23
24
25
26
27}
28
```

Problems Javadoc Declaration Console

spring-mvc-thymeleaf - SpringMvcThymeleafApplication [Spring Boot App] C:\Users\dell\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.2.v20210721-1149

Writable Smart Insert 21 : 1 : 556

Agregamos el método de registro al controlador

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - spring-mvc-thymeleaf/src/main/java/com/formacionjava/springmvc/controllers/TrabajadorController.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar icons.
- Package Explorer:** Shows the project structure:
 - iniciojava
 - maven_proyect
 - probando
 - projeto_web
 - Refresh
 - repaso
 - segundo [boot] [devtools]
 - springboot-apirest [boot] [devtools] [repo]
 - spring-mvc-thymeleaf [boot] [devtools]
 - src/main/java
 - com.formacionjava.springmvc
 - com.formacionjava.springmvc.controllers
 - TrabajadorController.java
 - com.formacionjava.springmvc.dao
 - com.formacionjava.springmvc.entity
 - com.formacionjava.springmvc.service
 - src/main/resources
 - static
 - templates
 - crear_trabajador.html
 - trabajador.html
 - application.properties
 - import.sql
 - src/test/java
 - JRE System Library [JavaSE-1.8]
 - Maven Dependencies
 - src
- TrabajadorController.java Tab:** The current file being edited.
- Code Editor:** The code for the TrabajadorController class. A red box highlights the last section of the code, which contains the new registration method:

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.servlet.ModelAndView;
import com.formacionjava.springmvc.TrabajadorServices;
import com.formacionjava.springmvc.entity.Trabajador;

@Controller
public class TrabajadorController {

    @Autowired
    private TrabajadorServices servicio;

    @GetMapping({"/trabajadores", "/"})
    public ModelAndView listarTrabajadores(Model modelo) {
        modelo.addAttribute("trabajador", servicio.listarTodosLosTrabajadores());
        return "trabajador";
    }

    @GetMapping("trabajador/nuevo")
    public ModelAndView crearTrabajadorFormulario(Model modelo) {
        Trabajador trabajador= new Trabajador();
        modelo.addAttribute("trabajador", trabajador);
        return "crear_trabajador";
    }

    @PostMapping("/trabajador")
    public String guardarTrabajador(@ModelAttribute("trabajador") Trabajador trabajador) {
        servicio.guardarTrabajador(trabajador);
        return "redirect:/trabajadores";
    }
}
```

@ModelAttribute

- ▶ Es una anotación que vincula un parámetro de método o un valor de retorno de método a un atributo de modelo con nombre y luego lo expone a una vista web

Probamos el registro

The screenshot shows a web browser window with the following details:

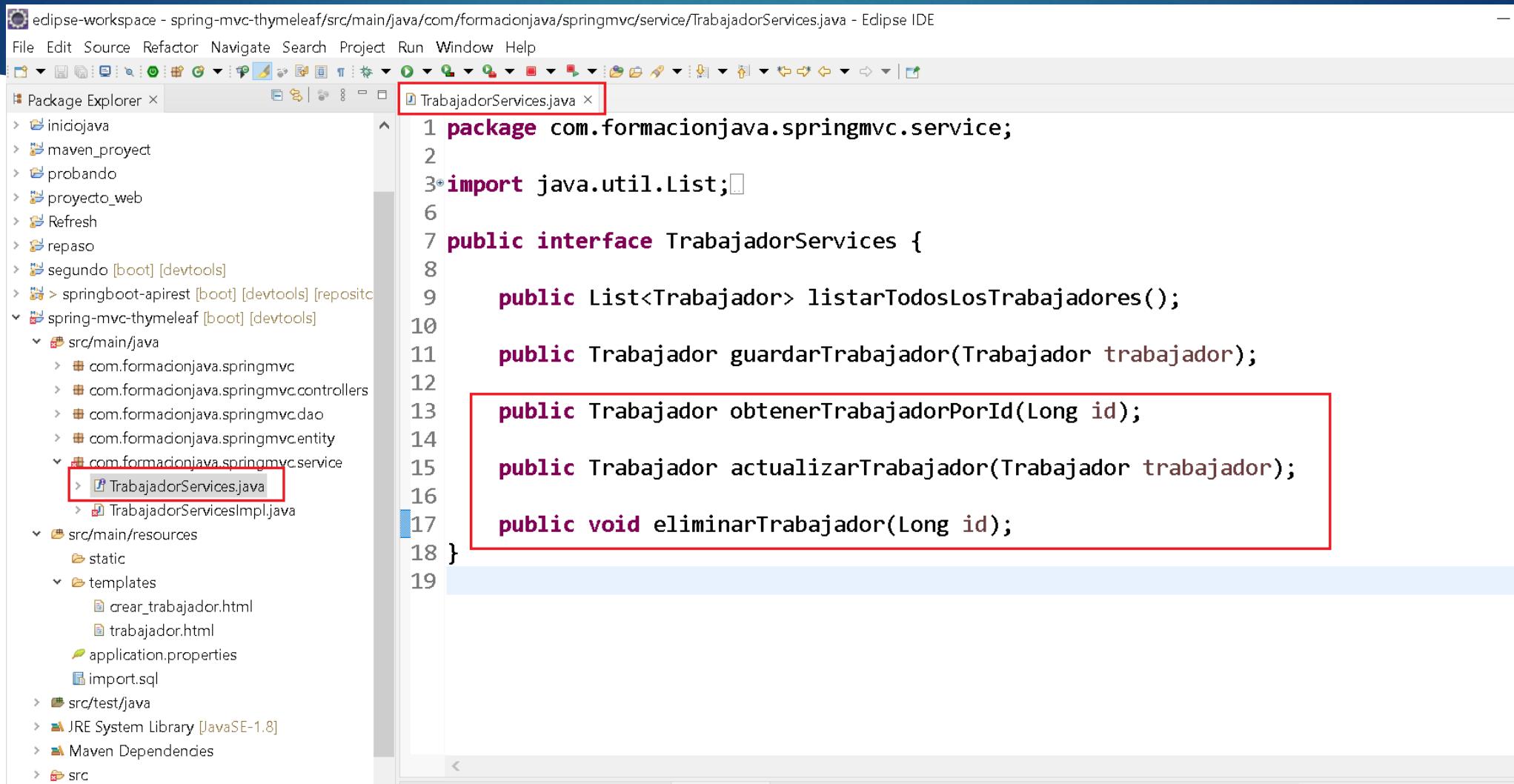
- Address Bar:** localhost:8087/trabajador/nuevo
- Navigation:** Back, Forward, Stop, Home, Refresh, Star, Lock, and More.
- Breadcrumbs:** Trabajadores > Trabajadores > Nuevo Trabajador
- Section Title:** Registro de trabajador
- Fields:**
 - Nombre: Manuel
 - Apellido: Molas
 - Email: manuel@email.com
 - Telefono: 323232
- Buttons:** Guardar (Save)

Trabajadores Trabajadores Nuevo Trabajador

Lista de trabajadores

Nombre	Apellido	Email	Telefono
Jose	Perez	jp@hotmail.com	6767676
Carlos	Lopez	cl@hotmail.com	6566565
Maria	Sanz	ms@hotmail.com	6454545
Gloria	Gomez	gm@hotmail.com	6523233
Leonel	Messi	leo@messi.com	322323
Manuel	Molas	manuel@email.com	323232

Agrego a mi servicio los métodos faltantes para el crud completo



The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - spring-mvc-thymeleaf/src/main/java/com/formacionjava/springmvc/service/TrabajadorServices.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations. The left sidebar is the Package Explorer showing project structure. The main editor window displays the TrabajadorServices.java code:

```
1 package com.formacionjava.springmvc.service;
2
3 import java.util.List;
4
5 public interface TrabajadorServices {
6
7     public List<Trabajador> listarTodosLosTrabajadores();
8
9     public Trabajador guardarTrabajador(Trabajador trabajador);
10
11    public Trabajador obtenerTrabajadorPorId(Long id);
12
13    public Trabajador actualizarTrabajador(Trabajador trabajador);
14
15    public void eliminarTrabajador(Long id);
16
17 }
18
19 }
```

The methods from line 13 to 18 are highlighted with a red box. The file path "TrabajadorServices.java" is also highlighted with a red box in the Package Explorer.

```
17  @Override
18  public List<Trabajador> listarTodosLosTrabajadores() {
19      return repositorio.findAll();
20  }
21
22  @Override
23  public Trabajador guardarTrabajador(Trabajador trabajador) {
24      return repositorio.save(trabajador);
25  }
26
27  @Override
28  public Trabajador obtenerTrabajadorPorId(Long id) {
29      return repositorio.findById(id).get();
30  }
31
32  @Override
33  public Trabajador actualizarTrabajador(Trabajador trabajador) {
34      return repositorio.save(trabajador);
35  }
36
37  @Override
38  public void eliminarTrabajador(Long id) {
39      repositorio.deleteById(id);
40  }
41
42
43 }
```

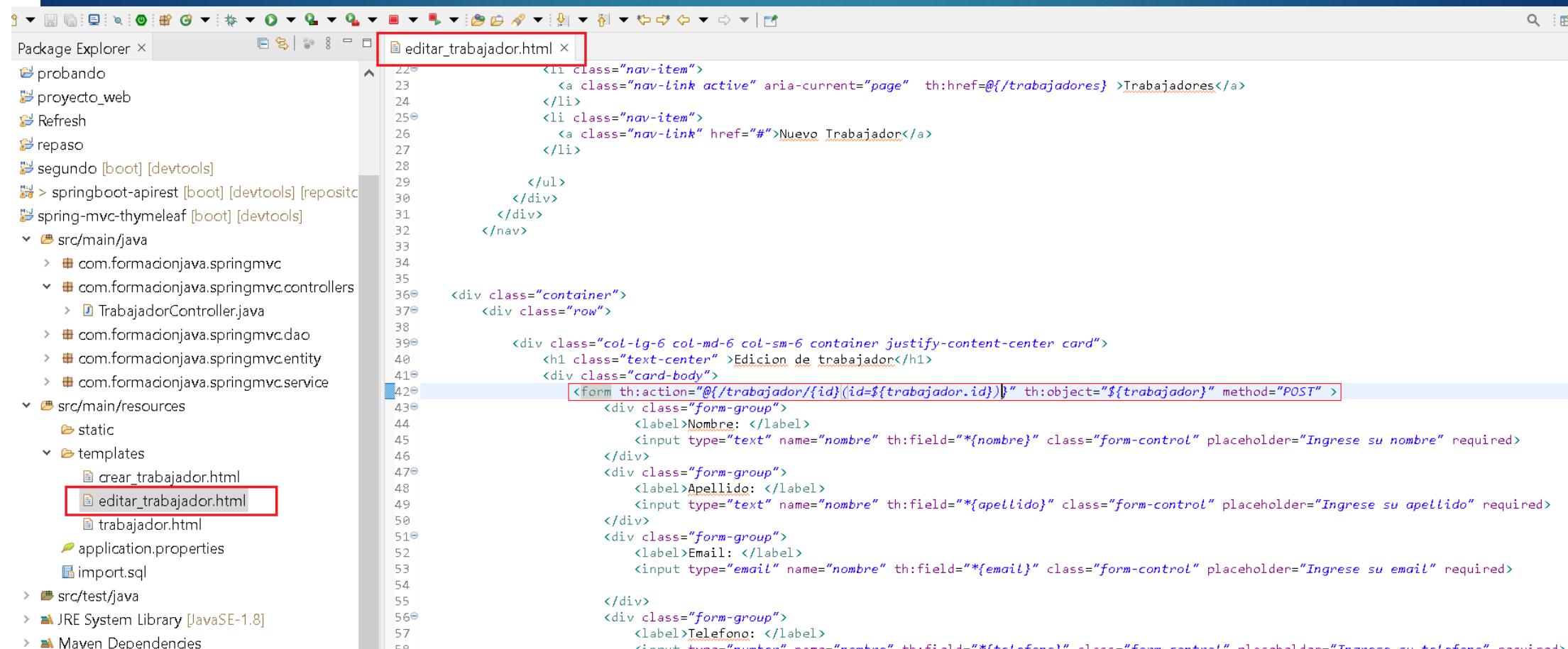
Agregamos los métodos al controlador

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - spring-mvc-thymeleaf/src/main/java/com/formacionjava/springmvc/controllers/TrabajadorController.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar icons.
- Package Explorer:** Shows the project structure:
 - probando
 - projeto_web
 - Refresh
 - repaso
 - segundo [boot] [devtools]
 - springboot-apirest [boot] [devtools] [repository]
 - spring-mvc-thymeleaf [boot] [devtools]
 - src/main/java
 - com.formacionjava.springmvc
 - com.formacionjava.springmvc.controllers
 - TrabajadorController.java (highlighted with a red box)
 - com.formacionjava.springmvc.dao
 - com.formacionjava.springmvc.entity
 - com.formacionjava.springmvc.service
 - src/main/resources
 - static
 - templates
 - crear_trabajador.html
 - editar_trabajador.html
 - trabajador.html
 - application.properties
 - import.sql
 - src/test/java
 - JRE System Library [JavaSE-1.8]
 - Maven Dependencies
 - src
 - target
 - HELP.md
 - mvnw
 - Editor:** Displays the code for TrabajadorController.java:

```
40
41     @GetMapping("/trabajador/editar/{id}")
42     public String mostrarFormularioEditar(@PathVariable Long id, Model modelo) {
43         modelo.addAttribute("trabajador", servicio.obtenerTrabajadorPorId(id));
44         return "editar_trabajador";
45     }
46
47     @PostMapping("/trabajador/{id}")
48     public String actualizarTrabajador(@PathVariable Long id, @ModelAttribute("trabajador") Trabajador trabajador,
49                                         Model modelo) {
50
51         Trabajador trabajadorExistente = servicio.obtenerTrabajadorPorId(id);
52
53         trabajadorExistente.setId(id);
54         trabajadorExistente.setNombre(trabajador.getNombre());
55         trabajadorExistente.setApellido(trabajador.getApellido());
56         trabajadorExistente.setEmail(trabajador.getEmail());
57         trabajadorExistente.setTelefono(trabajador.getTelefono());
58
59         servicio.actualizarTrabajador(trabajadorExistente);
60
61         return "redirect:/trabajadores";
62     }
63
64     @GetMapping("/trabajador/{id}")
65     public String eliminarTrabajador(@PathVariable Long id) {
66         servicio.eliminarTrabajador(id);
67         return "redirect:/trabajadores";
68     }
69
70
71 }
72 }
```
 - Bottom Bar:** Problems, Javadoc, Declaration, Console.

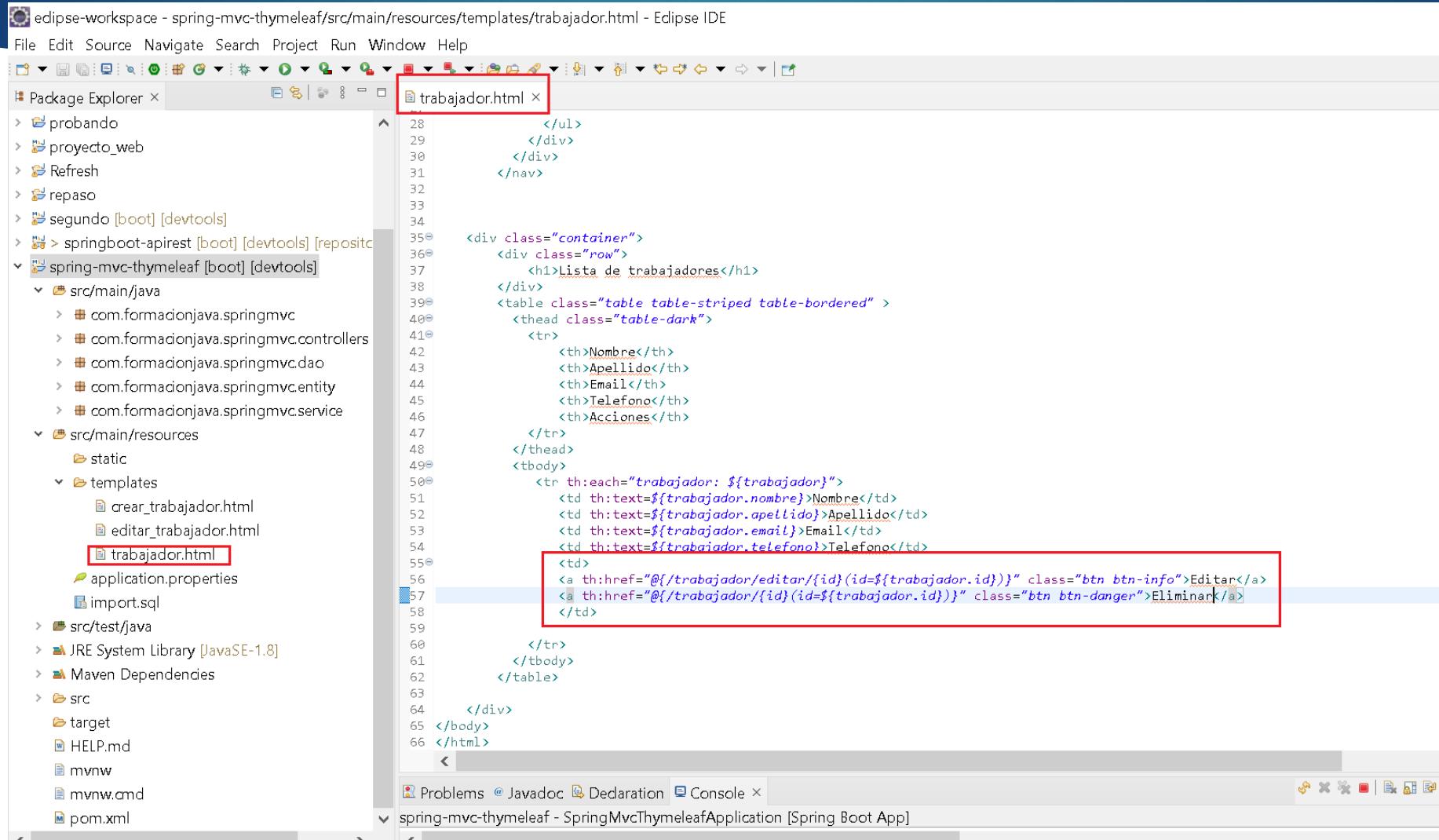
Creamos una copia de crear_trabajador y la llamamos editar_trabajador y solo editamos la línea marcada aquí



```
22<li class="nav-item">
23    <a class="nav-link active" aria-current="page" th:href="@{/trabajadores}">Trabajadores</a>
24</li>
25<li class="nav-item">
26    <a class="nav-link" href="#">Nuevo Trabajador</a>
27</li>
28
29</ul>
30</div>
31</div>
32</nav>
33
34
35
36<div class="container">
37    <div class="row">
38
39        <div class="col-lg-6 col-md-6 col-sm-6 container justify-content-center card">
40            <h1 class="text-center">Edición de trabajador</h1>
41            <div class="card-body">
42                <form th:action="@{/trabajador/{id}(id=${trabajador.id})}" th:object="${trabajador}" method="POST">
43                    <div class="form-group">
44                        <label>Nombre:</label>
45                        <input type="text" name="nombre" th:field="*{nombre}" class="form-control" placeholder="Ingrese su nombre" required>
46                    </div>
47                    <div class="form-group">
48                        <label>Apellido:</label>
49                        <input type="text" name="apellido" th:field="*{apellido}" class="form-control" placeholder="Ingrese su apellido" required>
50                    </div>
51                    <div class="form-group">
52                        <label>Email:</label>
53                        <input type="email" name="email" th:field="*{email}" class="form-control" placeholder="Ingrese su email" required>
54                    </div>
55                    <div class="form-group">
56                        <label>Teléfono:</label>
57                        <input type="number" name="telefono" th:field="*{telefono}" class="form-control" placeholder="Ingrese su teléfono" required>
58                </form>

```

En la plantilla trabajador.html agrego lo siguiente



The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - spring-mvc-thymeleaf/src/main/resources/templates/trabajador.html - Eclipse IDE
- Menu Bar:** File Edit Source Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar icons.
- Package Explorer:** Shows the project structure:
 - probando
 - projeto_web
 - Refresh
 - repasso
 - segundo [boot] [devtools]
 - springboot-apirest [boot] [devtools] [repository]
 - spring-mvc-thymeleaf [boot] [devtools]
 - src/main/java
 - com.formacionjava.springmvc
 - com.formacionjava.springmvc.controllers
 - com.formacionjava.springmvc.dao
 - com.formacionjava.springmvc.entity
 - com.formacionjava.springmvc.service
 - src/main/resources
 - static
 - templates
 - crear_trabajador.html
 - editar_trabajador.html
 - trabajador.html
 - application.properties
 - import.sql
 - src/test/java
 - JRE System Library [JavaSE-1.8]
 - Maven Dependencies
 - src
 - target
 - HELP.md
 - mvnw
 - mvnw.cmd
 - pom.xml
- Editor Area:** Displays the content of the 'trabajador.html' template file. A red box highlights the section of code being added:

```
<td>
<a th:href="@{/trabajador/editar/{id}(id=${trabajador.id})}" class="btn btn-info">Editar</a>
<a th:href="@{/trabajador/{id}(id=${trabajador.id})}" class="btn btn-danger">Eliminar</a>
</td>
```
- Bottom Status Bar:** Shows the current workspace and project information: Problems Javadoc Declaration Console
- Bottom Title Bar:** spring-mvc-thymeleaf - SpringMvcThymeleafApplication [Spring Boot App]

Probamos el CRUD completo

The screenshot shows a web browser window with the title "Listado de trabajadores". The address bar displays "localhost:8087/trabajadores". The page content is titled "Lista de trabajadores" and contains a table with four rows of employee data. Each row includes edit and delete buttons.

Nombre	Apellido	Email	Telefono	Acciones
Jose	Perez	jp@hotmail.com	6767676	<button>Editar</button> <button>Eliminar</button>
Carlos	Lopez	cl@hotmail.com	6566565	<button>Editar</button> <button>Eliminar</button>
Maria	Sanz	ms@hotmail.com	6454545	<button>Editar</button> <button>Eliminar</button>
Leonel	Messi	leo@messi.com	54578	<button>Editar</button> <button>Eliminar</button>

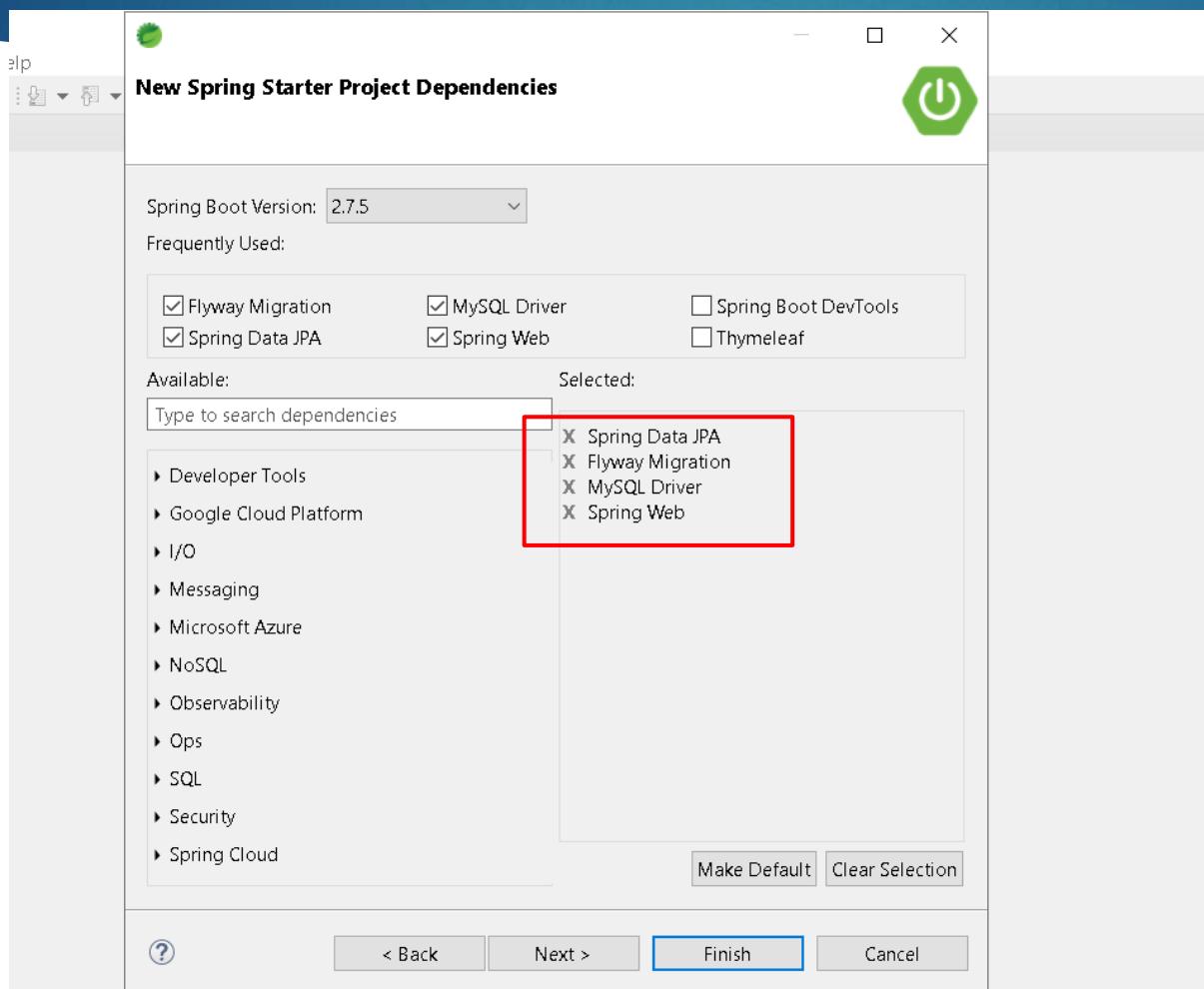
¿Qué es Flyway?

- ▶ Es un sistema robusto de **migración de base de datos**, que funciona con diferentes bases de datos y hace cualquier migración fácil y sencilla. Tiene compatibilidad con Java y también puede realizar migraciones a través de plugins de maven y gradle

Como funciona Flyway

- ▶ Flyway nos va a ayudar a seguir **una trazabilidad de que scripts que ya se han migrado, cuando y quién lo ha realizado.** Esta nueva tabla que Flyway crea en nuestro schema, va añade metada con un **checksum** en el que se indica si la migración tuvo éxito.
- ▶ Cuando se ejecuta flyway para realizar una migración de sql, se realiza las siguiente comprobaciones:
 - Se crea la tabla en la que guardar la información.
 - Se analiza los ficheros de la migración.
 - Se comparan los ficheros con las migraciones pasadas si ha habido cambios en los ficheros de versiones anteriores fallará la migración.
 - Los nuevos scripts de migración serán ejecutados y se añadirá la información de los nuevos scripts.

Creamos un ejemplo



Creamos una entidad user

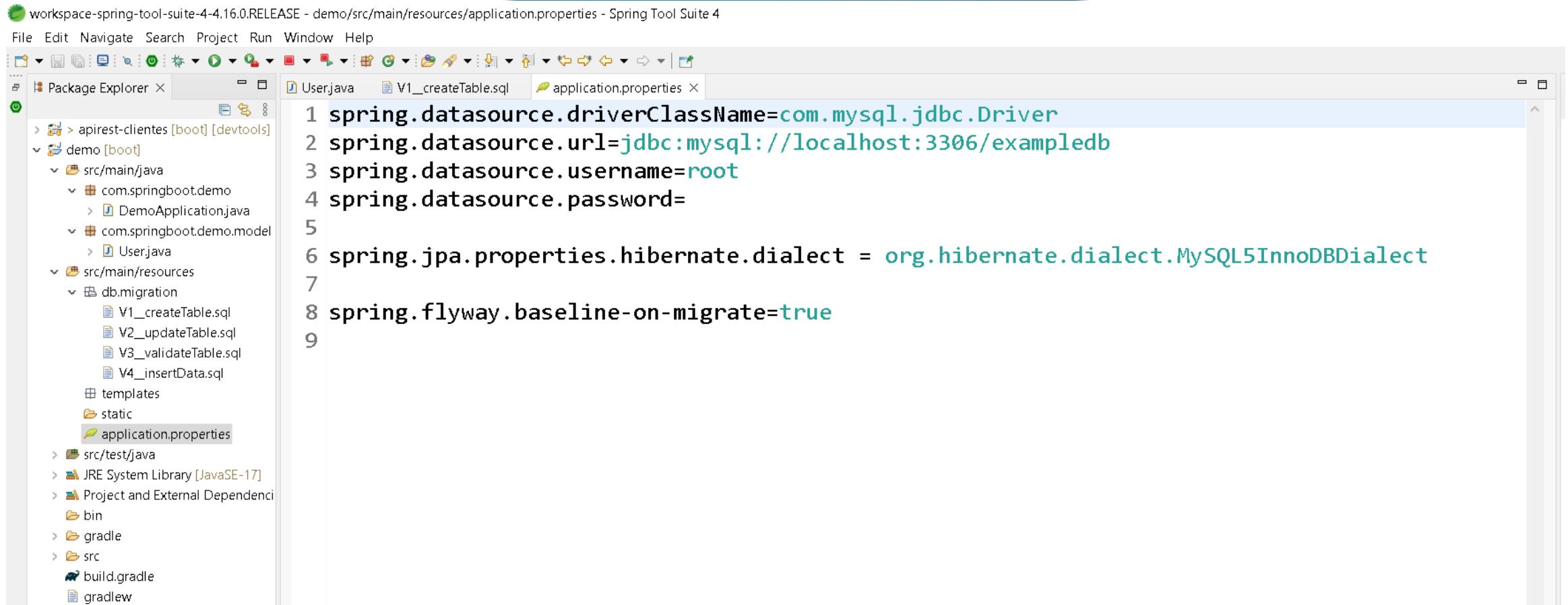
The screenshot shows the Spring Tool Suite 4 interface. On the left is the Package Explorer view, which lists several Spring Boot projects: apirest-clientes, demo (the current project selected), hola-spring, spring-boot-flyway-master, springboot-hola2, and spring-mvc-thymeleaf. The demo project's structure is visible, including src/main/java/com.springboot.demo.model where the User.java file is located. The central area is the User.java code editor, displaying the following Java code:

```
1 package com.springboot.demo.model;
2
3 import javax.persistence.Entity;
4
5 @Entity
6 @Table(name = "usuarios")
7 public class User {
8     @Id
9     @GeneratedValue
10    private int id;
11    private String username;
12    private String first_name;
13    private String last_name;
14    private String email;
15    private String mobile;
16    public int getId() {
17        return id;
18    }
19    public void setId(int id) {
20        this.id = id;
21    }
22}
```

At the bottom of the screen, the Eclipse IDE status bar shows the path: demo - DemoApplication [Spring Boot App] C:\Program Files\springtools\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32\jre\bin\java, followed by two log entries from the application:

```
2022-10-27 22:38:42.576  WARN 15312 --- [           main
2022-10-27 22:38:43.048  INFO 15312 --- [           main
```

Configuramos el application.properties



The screenshot shows the Spring Tool Suite 4 interface with the following details:

- Title Bar:** workspace-spring-tool-suite-4-4.16.0.RELEASE - demo/src/main/resources/application.properties - Spring Tool Suite 4
- Menu Bar:** File Edit Navigate Search Project Run Window Help
- Toolbar:** Includes icons for New, Open, Save, Cut, Copy, Paste, Find, Replace, and others.
- Package Explorer:** Shows the project structure:
 - apirest-clientes [boot] [devtools]
 - demo [boot]
 - src/main/java
 - com.springboot.demo
 - DemoApplication.java
 - com.springboot.demo.model
 - User.java
 - src/main/resources
 - db.migration
 - V1_createTable.sql
 - V2_updateTable.sql
 - V3_validateTable.sql
 - V4_insertData.sql
 - templates
 - static
 - application.properties
 - src/test/java
 - JRE System Library [JavaSE-17]
 - Project and External Dependencies
 - bin
 - gradle
 - src
 - build.gradle
 - gradlew
- Editor Area:** Displays the contents of the application.properties file:

```
1 spring.datasource.driverClassName=com.mysql.jdbc.Driver
2 spring.datasource.url=jdbc:mysql://localhost:3306/exampledb
3 spring.datasource.username=root
4 spring.datasource.password=
5
6 spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect
7
8 spring.flyway.baseline-on-migrate=true
9
```

workspace-spring-tool-suite-4-4.16.0.RELEASE - demo/src/main/resources/db/migration/V1_createTable.sql - Spring Tool Suite 4

File Edit Navigate Search Project Run Window Help

User.java V1_createTable.sql application.properties V2_updateTable.sql V3_validateTable.sql V4_insertData.sql build.gradle

```
1 CREATE TABLE USUARIOS (
2     id bigint(20) NOT NULL AUTO_INCREMENT,
3     username varchar(100) NOT NULL,
4     first_name varchar(50) NOT NULL,
5     last_name varchar(50) DEFAULT NULL,
6     email varchar(50) NOT NULL,
7     PRIMARY KEY (id),
8     UNIQUE KEY UK_username (username)
9 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

10

Package Explorer X

apirest-clientes [boot] [devtools]

demo [boot]

- src/main/java
 - com.springboot.demo
 - DemoApplication.java
 - com.springboot.demo.model
 - User.java
- src/main/resources
 - db.migration
 - V1_createTable.sql
 - V2_updateTable.sql
 - V3_validateTable.sql
 - V4_insertData.sql
 - templates
 - static
 - application.properties
- src/test/java
- JRE System Library [JavaSE-17]
- Project and External Dependencies
- bin
- gradle
- src
 - build.gradle
 - gradlew
 - gradlew.bat
 - HELP.md
 - settings.gradle
- hola-spring [boot] [devtools]
- spring-boot-flyway-master [boot]
- springboot-hola2 [boot] [devtools]
- spring-mvc-thymeleaf [boot] [devtools]

Problems @ Javadoc Declaration Console X

workspace-spring-tool-suite-4-4.16.0.RELEASE - demo/src/main/resources/db/migration/V2_updateTable.sql - Spring Tool Suite 4

File Edit Navigate Search Project Run Window Help

Package Explorer X User.java V1_createTable.sql application.properties V2_updateTable.sql X

```
1 ALTER TABLE USUARIOS
2 ADD COLUMN mobile VARCHAR(15) AFTER last_name;
```

apirest-clientes [boot] [devtools]

demo [boot]

- src/main/java
 - com.springboot.demo
 - DemoApplication.java
 - com.springboot.demo.model
 - User.java
- src/main/resources
 - db.migration
 - V1_createTable.sql
 - V2_updateTable.sql
 - V3_validateTable.sql
 - V4_insertData.sql
 - templates
 - static
 - application.properties
- src/test/java
- JRE System Library [JavaSE-17]
- Project and External Dependencies
 - bin
- gradle
- src
 - build.gradle
 - gradlew
 - gradlew.bat

workspace-spring-tool-suite-4-4.16.0.RELEASE - demo/src/main/resources/db/migration/V3_validateTable.sql - Spring Tool Suite 4

File Edit Navigate Search Project Run Window Help

Package Explorer X User.java V1_createTable.sql application.properties V2_updateTable.sql V3_validateTable.sql X

```
1ALTER TABLE USUARIOS MODIFY mobile VARCHAR(11) ;
```

apirest-clients [boot] [devtools]

demo [boot]

- src/main/java
 - com.springboot.demo
 - DemoApplication.java
 - com.springboot.demo.model
 - User.java
- src/main/resources
 - db.migration
 - V1_createTable.sql
 - V2_updateTable.sql
 - V3_validateTable.sql
 - V4_insertData.sql
 - templates
 - static
 - application.properties
- src/test/java
- JRE System Library [JavaSE-17]
- Project and External Dependencies
- bin
- gradle
- src
 - build.gradle
 - gradlew
 - gradlew.bat
 - HELP.md
 - settings.gradle
- hola-spring [boot] [devtools]

workspace-spring-tool-suite-4-4.16.0.RELEASE - demo/src/main/resources/db/migration/V4_insertData.sql - Spring Tool Suite 4

File Edit Navigate Search Project Run Window Help

The screenshot shows the Spring Tool Suite 4 interface with a dark blue header and a light blue sidebar. The main window displays a code editor for a database migration script named V4_insertData.sql. The script contains two SQL INSERT statements:

```
1 INSERT INTO USUARIOS(username, first_name, last_name,email,mobile) VALUES('javatechie', 'java', 'techie','a@gmail.com','9910121314');
2 INSERT INTO USUARIOS(username, first_name, last_name,email,mobile) VALUES('bh', 'basant', 'hota','b@gmail.com','7735888844');
```

Unit Test

- Son códigos diseñados para comprobar que el código principal este funcionando de la forma que esperamos.

Tipos de pruebas

- ▶ Pruebas Unitarias (Unit Test)
- ▶ Pruebas de Integración
- ▶ Pruebas de funcionamiento del sistema
- ▶ Pruebas de Aceptación
- ▶ Pruebas de estrés

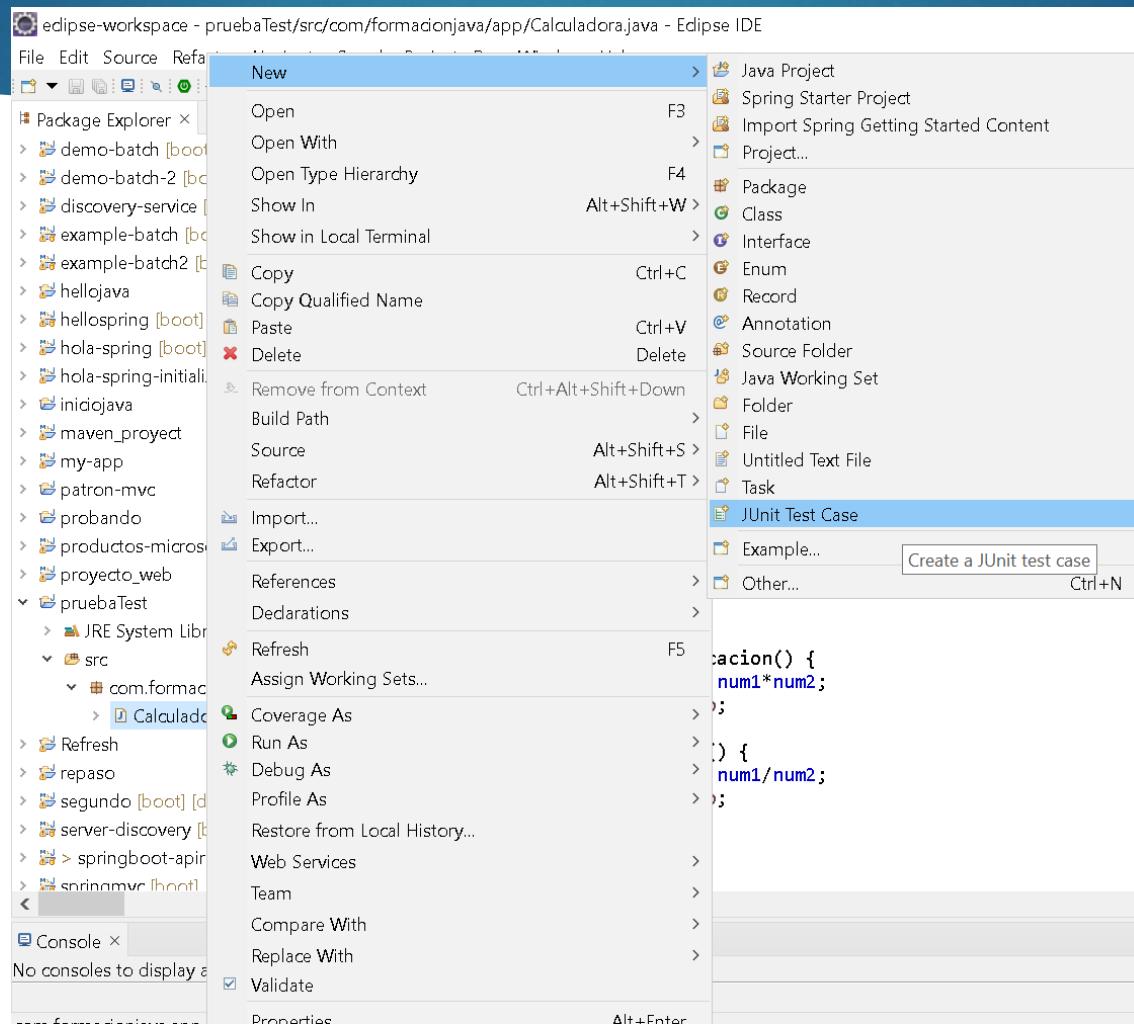
JUnit

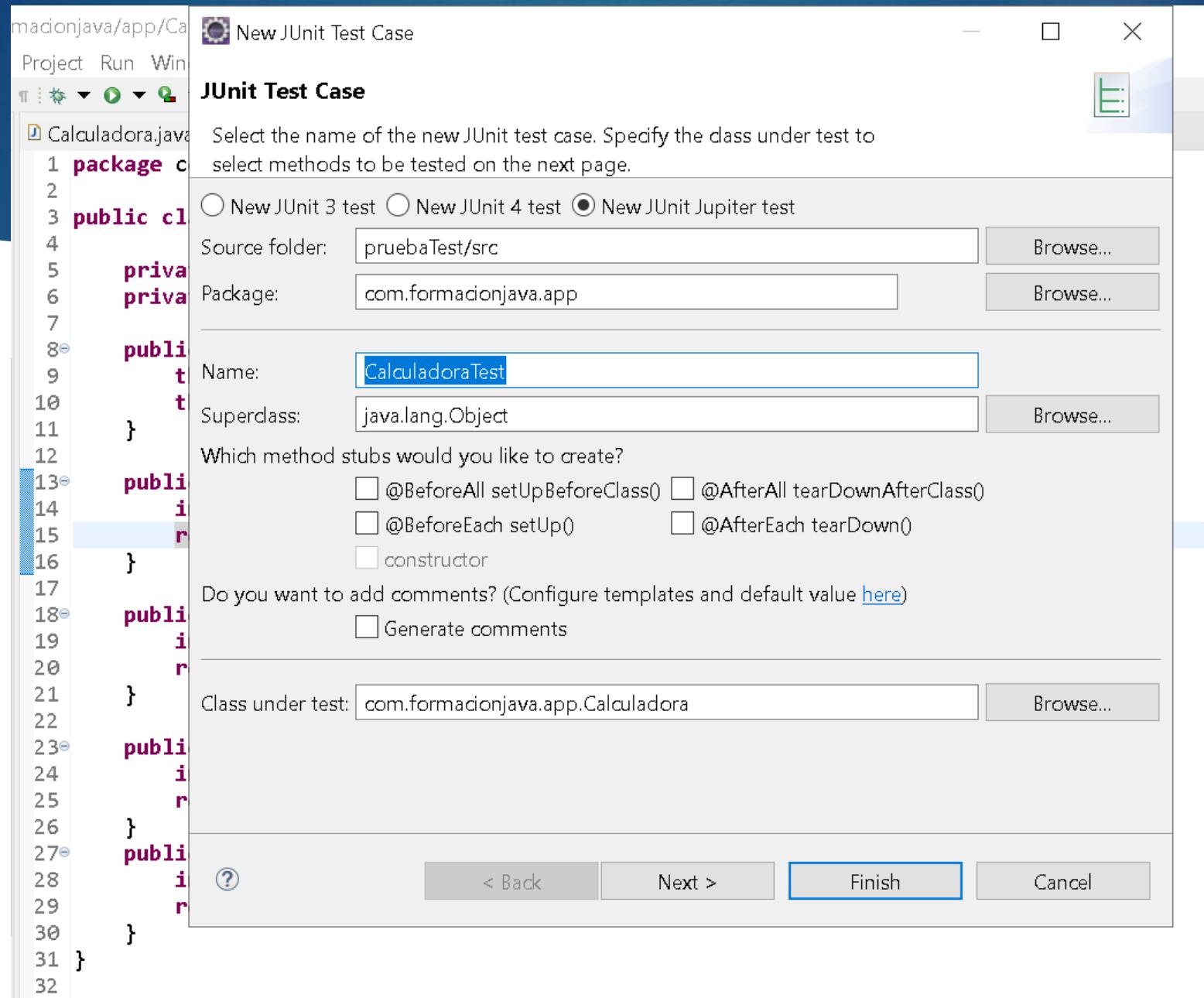
- Es una librería java para escribir y ejecutar pruebas unitarias

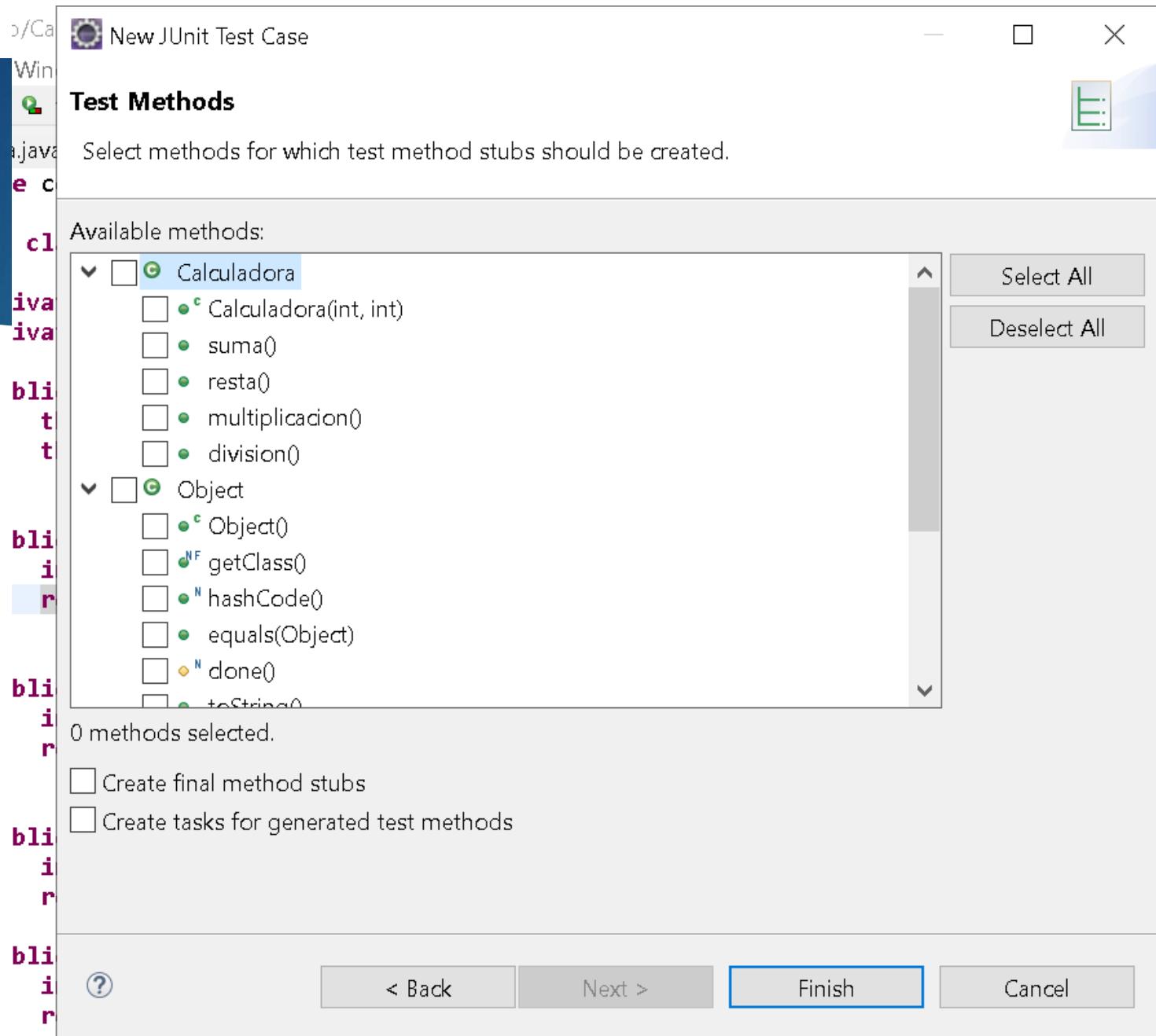
Realizamos una prueba

```
1
2
3 public class Calculadora {
4
5     private int num1;
6     private int num2;
7
8     public Calculadora(int a,int b) {
9         this.num1 = a;
10        this.num2 = b;
11    }
12
13     public int suma() {
14         int resultado = num1+num2;
15         return resultado;
16     }
17
18     public int resta() {
19         int resultado = num1-num2;
20         return resultado;
21     }
22
23     public int multiplicacion() {
24         int resultado = num1*num2;
25         return resultado;
26     }
27     public int division() {
28         int resultado = num1/num2;
29         return resultado;
30     }
31 }
32
```

Creamos la clase de unitTest

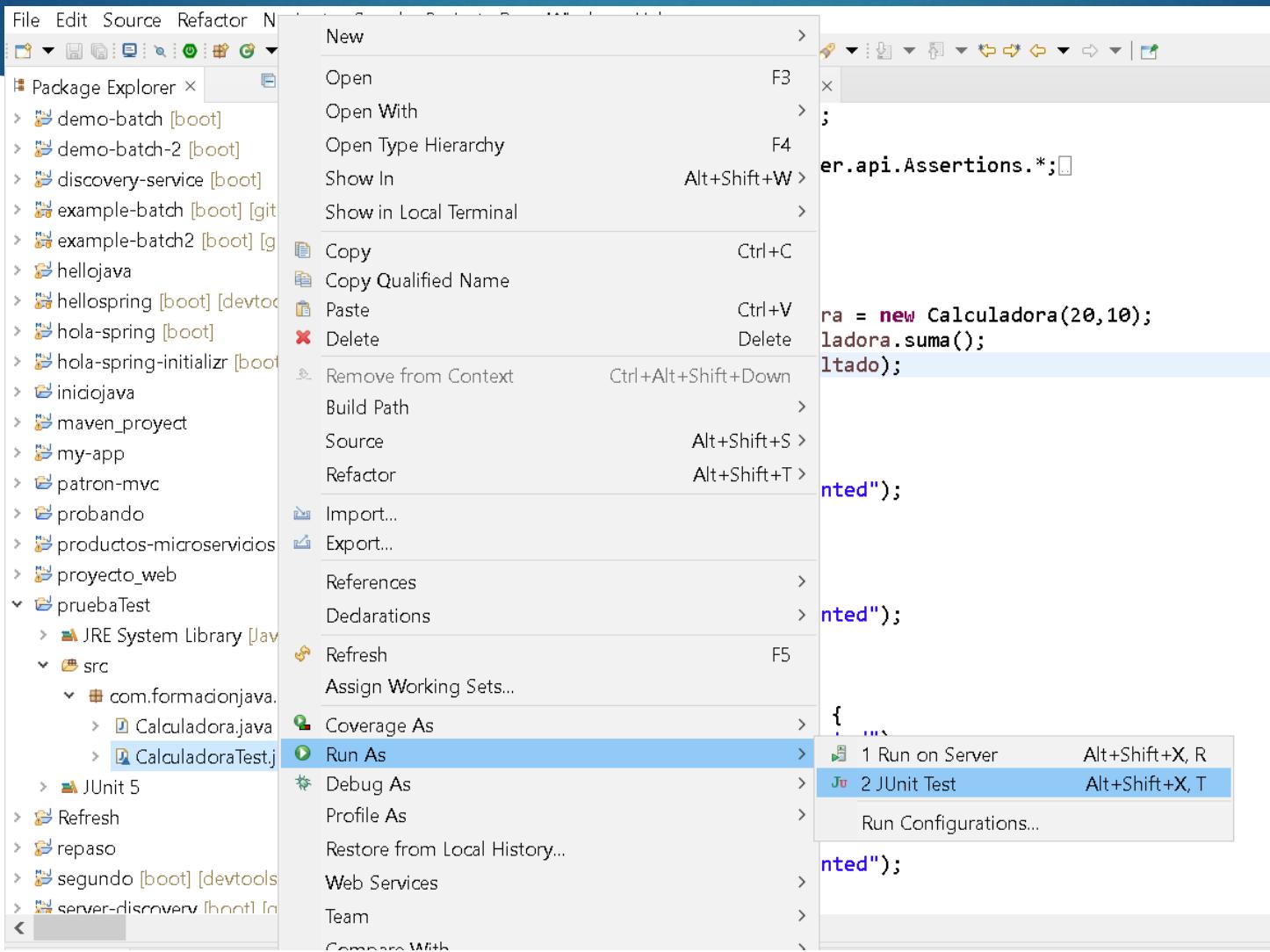


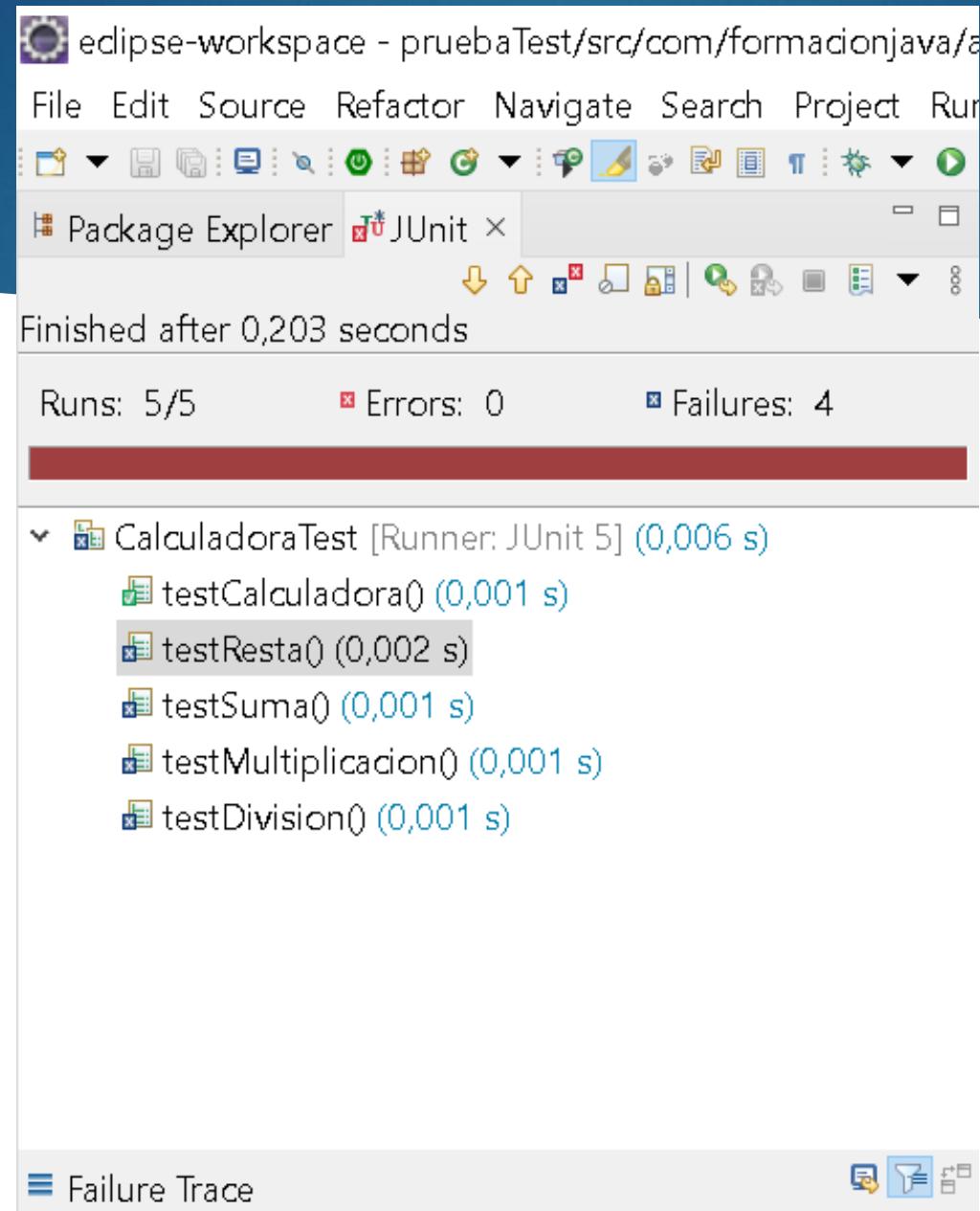




```
Calculadora.java  CalculadoraTest.java x
1 package com.formacionjava.app;
2
3+import static org.junit.jupiter.api.Assertions.*;
4
5 class CalculadoraTest {
6
7     @Test
8     void testCalculadora() {
9         fail("Not yet implemented");
10    }
11
12
13
14     @Test
15     void testSuma() {
16         fail("Not yet implemented");
17    }
18
19     @Test
20     void testResta() {
21         fail("Not yet implemented");
22    }
23
24     @Test
25     void testMultiplicacion() {
26         fail("Not yet implemented");
27    }
28
29     @Test
30     void testDivision() {
31         fail("Not yet implemented");
32    }
33
34 }
```

Ejecutamos el UnitTest





eclipse-workspace - pruebaTest/src/com/formacionjava/app/CalculadoraTest.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer JUnit ×

Finished after 0,224 seconds

Runs: 4/4 Errors: 0 Failures: 0

CalculadoraTest [Runner: JUnit 5] (0,003 s)

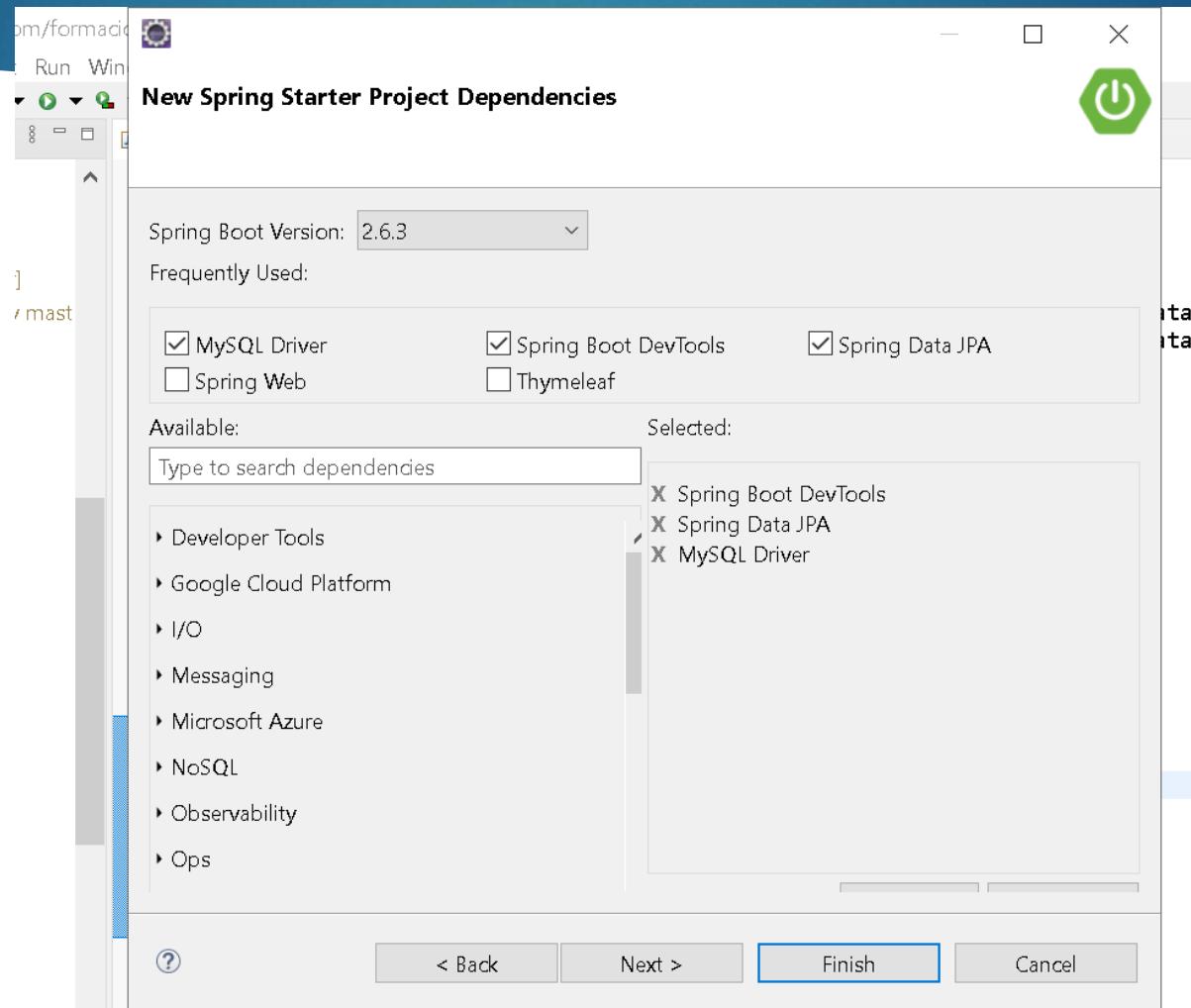
- testResta() (0,000 s)
- testSuma() (0,000 s)
- testMultiplicacion() (0,000 s)
- testDivision() (0,001 s)

Failure Trace

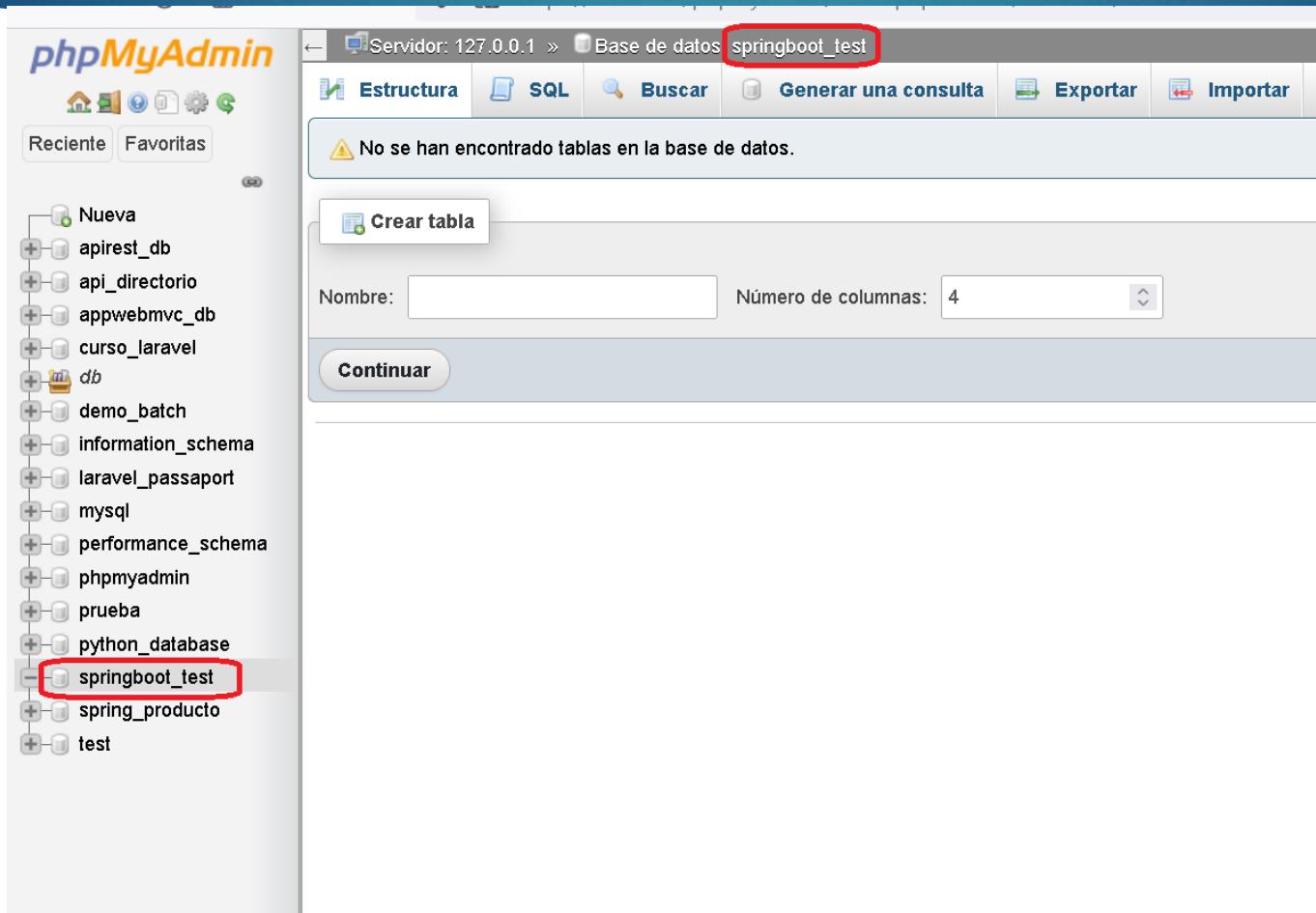
Calculadora.java CalculadoraTest.java ×

```
3+import static org.junit.jupiter.api.Assertions.*;  
6  
7 class CalculadoraTest {  
8  
9     /*@Test  
10     void testCalculadora() {  
11         */  
12     }  
13  
14     @Test  
15     void testSuma() {  
16         Calculadora calculadora = new Calculadora(20,10);  
17         int resultado = calculadora.suma();  
18         assertEquals(30, resultado);  
19     }  
20  
21     @Test  
22     void testResta() {  
23         Calculadora calculadora = new Calculadora(20,10);  
24         int resultado = calculadora.resta();  
25         assertEquals(10, resultado);  
26     }  
27  
28     @Test  
29     void testMultiplicacion() {  
30         Calculadora calculadora = new Calculadora(3,2);  
31         int resultado = calculadora.multiplicacion();  
32         assertEquals(6, resultado);  
33     }  
34  
35     @Test  
36     void testDivision() {  
37         Calculadora calculadora = new Calculadora(20,10);  
38         int resultado = calculadora.division();  
39         assertEquals(2, resultado);  
40     }  
41  
42 }
```

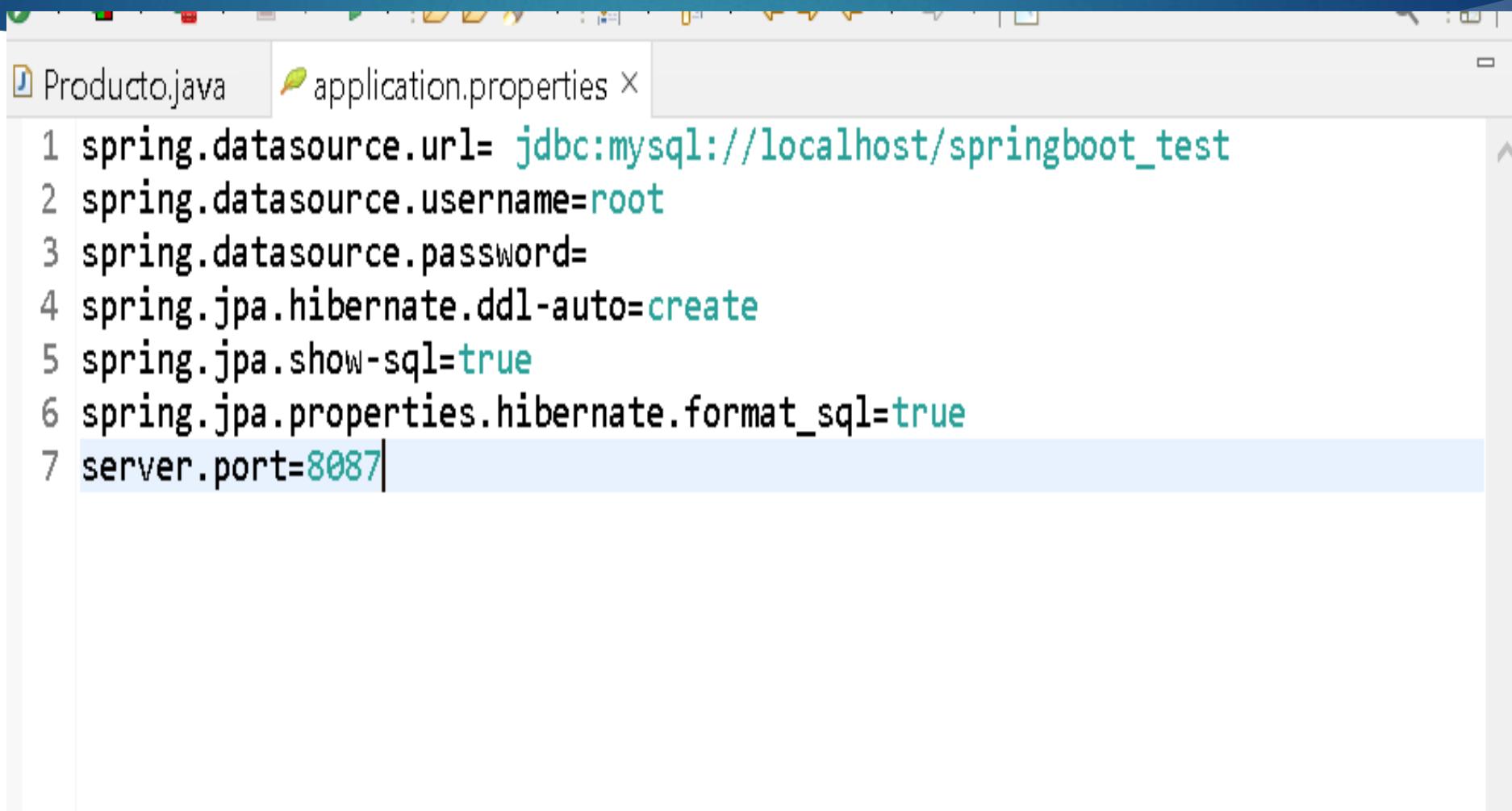
Utilizando pruebas unitarias en SpringBoot, creamos un proyecto con las siguientes dependencias



Creamos una base de datos



En nuestro application.properties



The screenshot shows a Java IDE interface with the application.properties file open. The tab bar at the top has 'Producto.java' and 'application.properties'. The code editor displays the following configuration properties:

```
1 spring.datasource.url= jdbc:mysql://localhost/springboot_test
2 spring.datasource.username=root
3 spring.datasource.password=
4 spring.jpa.hibernate.ddl-auto=create
5 spring.jpa.show-sql=true
6 spring.jpa.properties.hibernate.format_sql=true
7 server.port=8087
```

Creamos la entidad Producto

```
1 package com.formacionspringboot.app.entity;
2
3+import javax.persistence.Entity;□
8
9 @Entity
10 @Table(name="productos")
11 public class Producto {
12
13@  @Id
14  @GeneratedValue(strategy = GenerationType.IDENTITY)
15  private Long id;
16  private String nombre;
17  private float precio;
18
19  public Producto() {
20
21  }
22  public Producto(String nombre, float precio) {
23      this.nombre = nombre;
24      this.precio = precio;
25  }
26  public Long getId() {
27      return id;
28  }
29  public void setId(Long id) {
30      this.id = id;
31  }
32  public String getNombre() {
33      return nombre;
34  }
35  public void setNombre(String nombre) {
36      this.nombre = nombre;
37  }
```

Creamos un repositorio ProductoDao

The screenshot shows the Eclipse IDE interface with a blue header bar and a green vertical bar on the right. The main window displays a Java code editor and a Package Explorer.

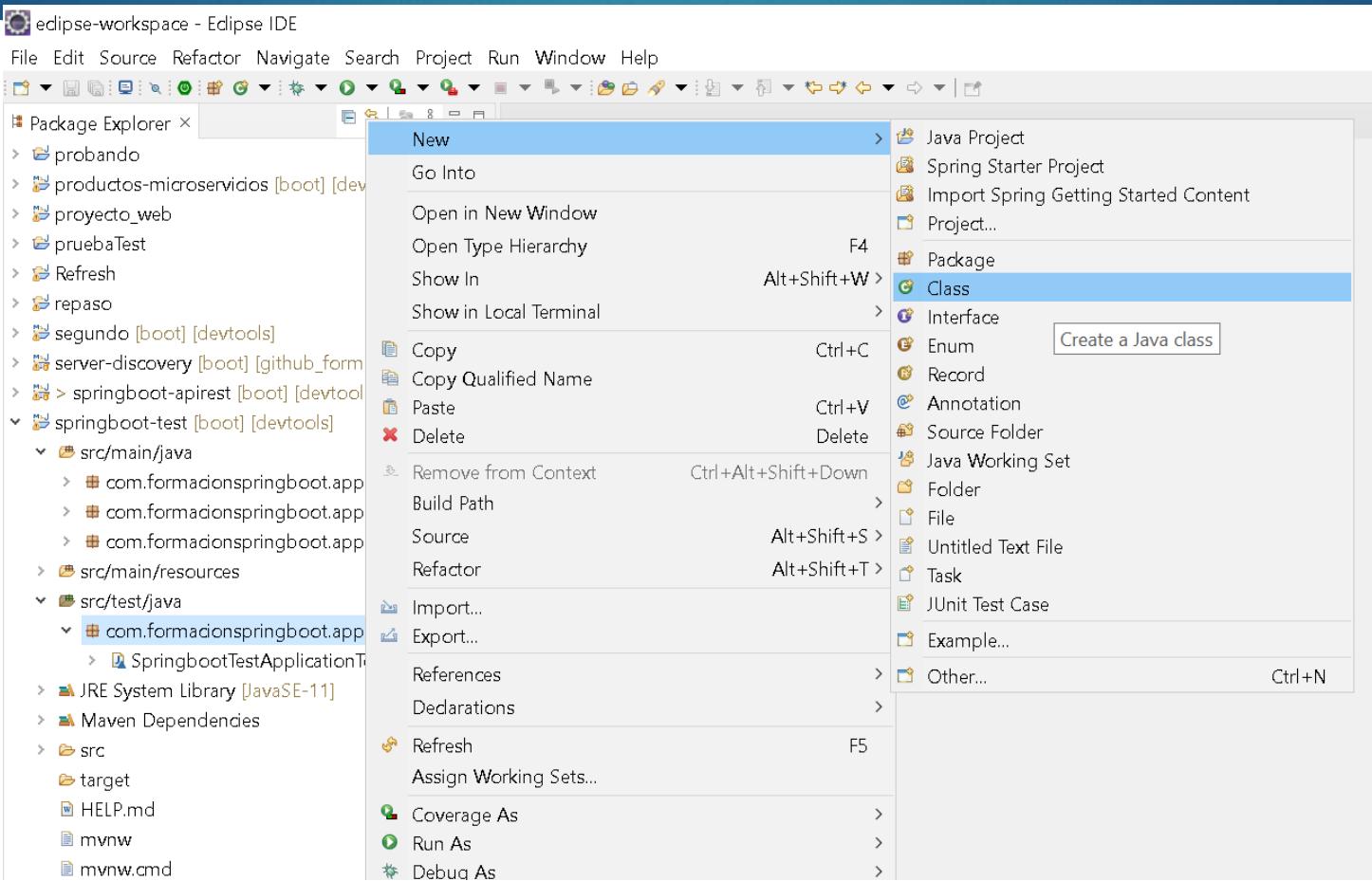
Package Explorer:

- probando
- productos-microservicios [boot] [devtools]
- proyecto_web
- pruebaTest
- Refresh
- repaso
- segundo [boot] [devtools]
- server-discovery [boot] [github_formacion master]
- > springboot-apirest [boot] [devtools] [repository]
- springboot-test [boot] [devtools]**
 - src/main/java
 - com.formacionspringboot.app
 - com.formacionspringboot.app.dao
 - ProductoDao.java**
 - com.formacionspringboot.app.entity
 - src/main/resources
 - src/test/java
 - JRE System Library [JavaSE-11]
 - Maven Dependencies
 - src
 - target
 - HELP.md
 - mvnw

Code Editor (ProductoDao.java):

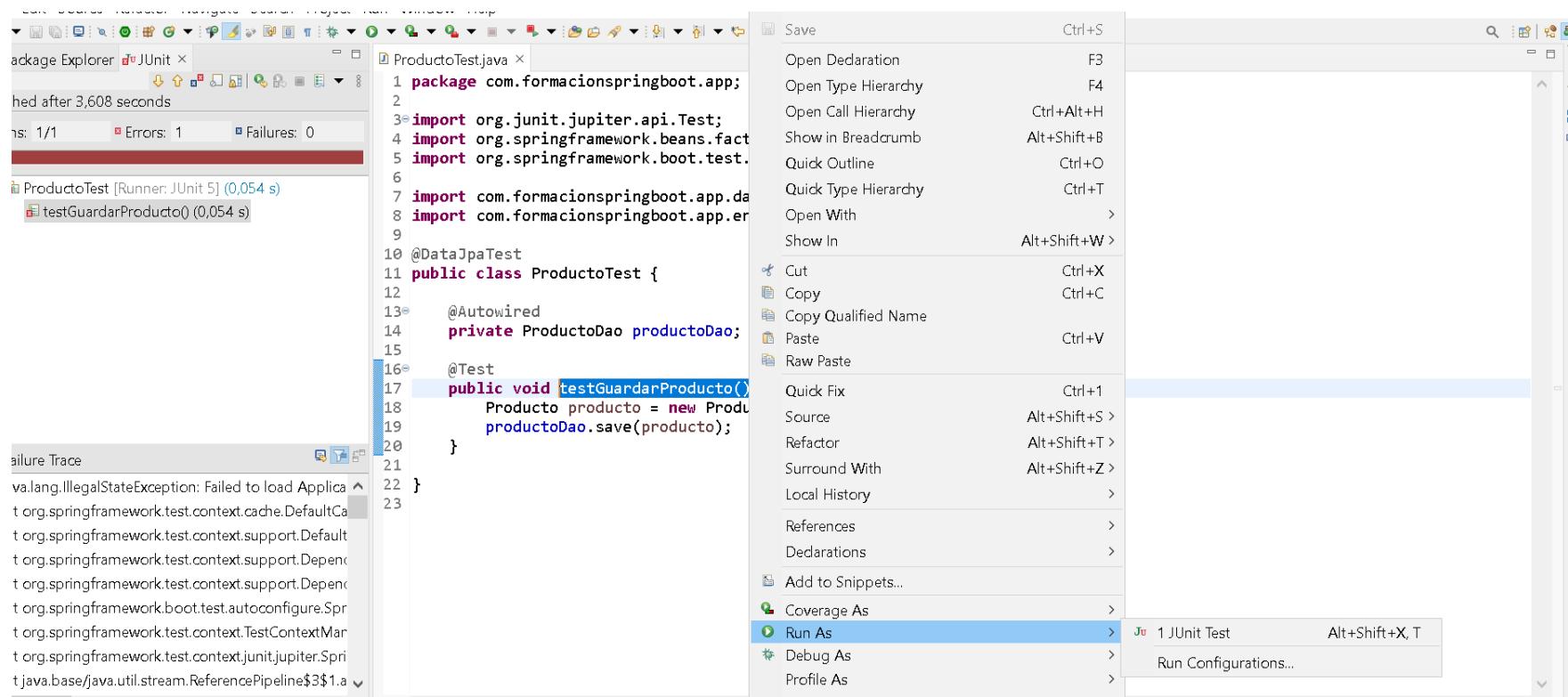
```
1 package com.formacionspringboot.app.dao;
2
3 import org.springframework.data.repository.CrudRepository;
4 import com.formacionspringboot.app.entity.Producto;
5
6 public interface ProductoDao extends CrudRepository<Producto, Long>{
7
8 }
```

Dentro del paquete principal de src/test/java creamos una clase ProductoTest



```
1 package com.formacionspringboot.app;
2
3 import org.junit.jupiter.api.Test;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;
6
7 import com.formacionspringboot.app.dao.ProductoDao;
8 import com.formacionspringboot.app.entity.Producto;
9
10 @DataJpaTest
11 public class ProductoTest {
12
13     @Autowired
14     private ProductoDao productoDao;
15
16     @Test
17     public void testGuardarProducto() {
18         Producto producto = new Producto("TV Samsung HD",4000);
19         productoDao.save(producto);
20     }
21
22 }
```

Lo ejecutamos



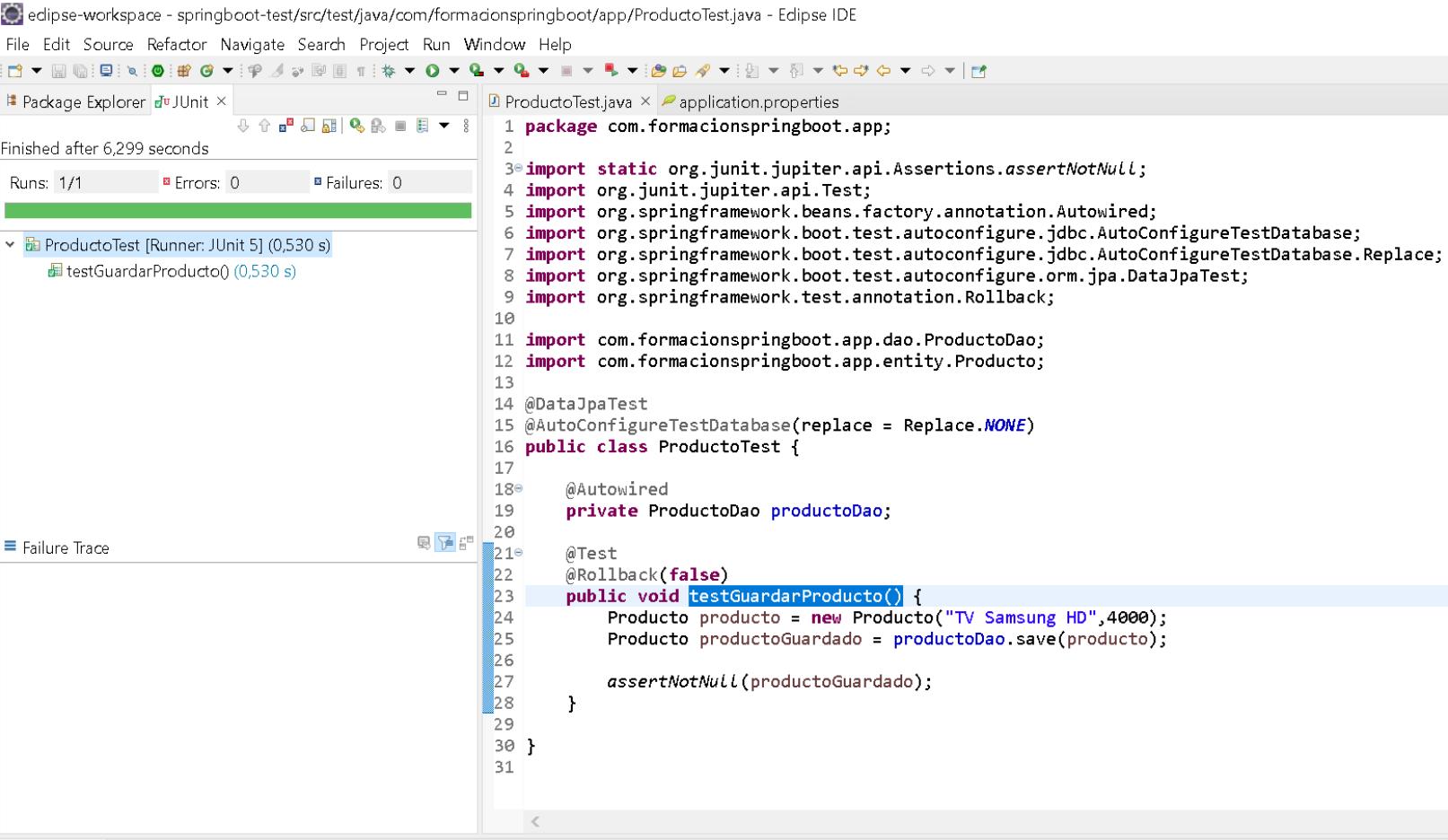
Agregamos las notaciones y código para necesarias para lanzar los test y registrarlos en la base de datos

formacionspringboot/app/ProductoTest.java - Eclipse IDE

Window Help

```
1 package com.formacionspringboot.app;
2
3 import static org.junit.jupiter.api.Assertions.assertNotNull;
4 import org.junit.jupiter.api.Test;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase;
7 import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase.Replace;
8 import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;
9 import org.springframework.test.annotation.Rollback;
10
11 import com.formacionspringboot.app.dao.ProductoDao;
12 import com.formacionspringboot.app.entity.Producto;
13
14 @DataJpaTest
15 @AutoConfigureTestDatabase(replace = Replace.NONE)
16 public class ProductoTest {
17
18     @Autowired
19     private ProductoDao productoDao;
20
21     @Test
22     @Rollback(false)
23     public void testGuardarProducto() {
24         Producto producto = new Producto("TV Samsung HD",4000);
25         Producto productoGuardado = productoDao.save(producto);
26
27         assertNotNull(productoGuardado);
28     }
29
30 }
```

Ejecutamos el test



The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - springboot-test/src/test/java/com/formacionspringboot/app/ProductoTest.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar with various icons for file operations.
- Package Explorer:** Shows the project structure with "ProductoTest" selected.
- JUnit View:** Displays the test results: "Finished after 6,299 seconds", "Runs: 1/1", "Errors: 0", and "Failures: 0".
- Code Editor:** The "ProductoTest.java" file is open, showing Java code for testing a Product entity. The code includes imports for JUnit Jupiter and Spring Boot Test annotations, and a test method named "testGuardarProducto()".
- Status Bar:** Shows "Failure Trace" and other standard Eclipse status bar information.

```
1 package com.formacionspringboot.app;
2
3 import static org.junit.jupiter.api.Assertions.assertNotNull;
4 import org.junit.jupiter.api.Test;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase;
7 import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase.Replace;
8 import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;
9 import org.springframework.test.annotation.Rollback;
10
11 import com.formacionspringboot.app.dao.ProductoDao;
12 import com.formacionspringboot.app.entity.Producto;
13
14 @DataJpaTest
15 @AutoConfigureTestDatabase(replace = Replace.NONE)
16 public class ProductoTest {
17
18     @Autowired
19     private ProductoDao productoDao;
20
21     @Test
22     @Rollback(false)
23     public void testGuardarProducto() {
24         Producto producto = new Producto("TV Samsung HD",4000);
25         Producto productoGuardado = productoDao.save(producto);
26
27         assertNotNull(productoGuardado);
28     }
29
30 }
31
```

Verificamos el registro en la base de datos

The screenshot shows the MySQL Workbench interface. The left sidebar lists databases: pirest_db, pi_directorio, Nueva, directorios, failed_jobs, migrations, password_resets, users, ppwebmvc_db, urso_laravel, /b, memo_batch, information_schema, laravel_passaport, mysql, performance_schema, hpmysql, rueba, python_database, springboot_test, Nueva, productos, and spring_producto. The main window title is "Servidor: 127.0.0.1 » Base de datos: springboot_test » Tabla: productos". The toolbar includes Examinar, Estructura, SQL, Buscar, Insertar, Exportar, Importar, Privilegios, and Operaciones. A message bar at the top says "Mostrando filas 0 - 0 (total de 1, La consulta tardó 0,0076 segundos.)". Below it is a SQL query: "SELECT * FROM `productos`". The table view shows one row: id: 1, nombre: TV Samsung HD, precio: 4000. There are buttons for Editar, Copiar, and Borrar. The bottom section contains buttons for Imprimir, Copiar al portapapeles, Exportar, Mostrar gráfico, and Crear vista.

	id	nombre	precio
	1	TV Samsung HD	4000

Agregamos este método al repositorio

formacionspringboot/app/dao/ProductoDao.java - Eclipse IDE

n Window Help

File Edit View Search Project Properties Run Debug Tools Window Help

src

1 package com.formacionspringboot.app.dao;

2

3 import org.springframework.data.repository.CrudRepository;

4

5

6 public interface ProductoDao extends CrudRepository<Producto, Long>{

7

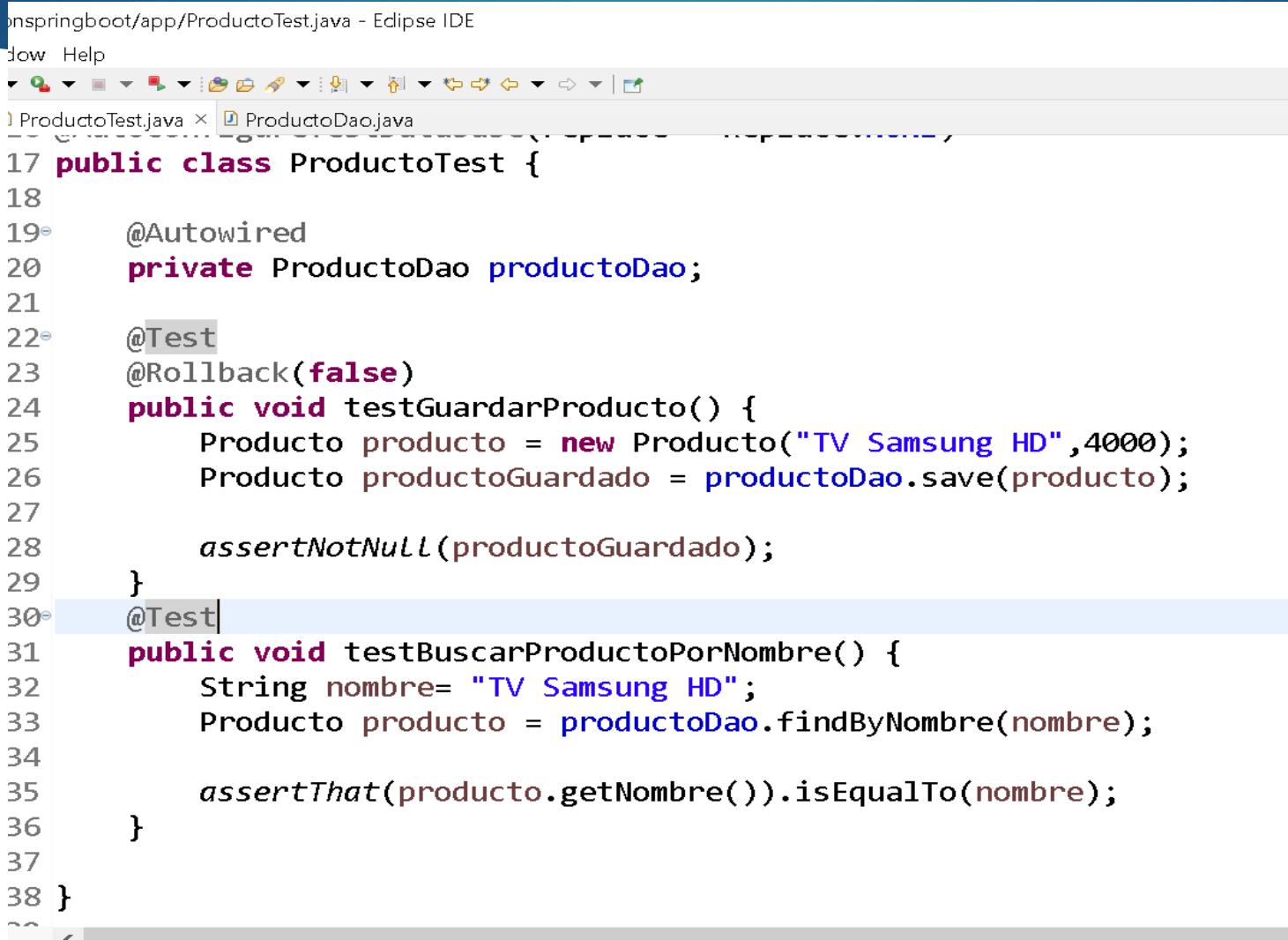
8 public Producto findByNombre(String nombre);

9

10}

11

Agregamos el siguiente método para el test



The screenshot shows the Eclipse IDE interface with the title bar "Eclipse IDE" and a toolbar above the editor. The editor window displays a Java file named "ProductoTest.java". The code is annotated with line numbers (17 to 38) and includes imports for "JUnit4" and "Mockito". It defines a class "ProductoTest" with two test methods: "testGuardarProducto" and "testBuscarProductoPorNombre". The "testGuardarProducto" method creates a new "Producto" object and saves it using the "productoDao" dependency. It then asserts that the saved product is not null. The "testBuscarProductoPorNombre" method searches for a product by name and asserts that the retrieved product's name equals the search term.

```
17 public class ProductoTest {  
18  
19     @Autowired  
20     private ProductoDao productoDao;  
21  
22     @Test  
23     @Rollback(false)  
24     public void testGuardarProducto() {  
25         Producto producto = new Producto("TV Samsung HD", 4000);  
26         Producto productoGuardado = productoDao.save(producto);  
27  
28         assertNotNull(productoGuardado);  
29     }  
30     @Test  
31     public void testBuscarProductoPorNombre() {  
32         String nombre= "TV Samsung HD";  
33         Producto producto = productoDao.findByNombre(nombre);  
34  
35         assertThat(producto.getNombre()).isEqualTo(nombre);  
36     }  
37  
38 }
```

Ejecutamos

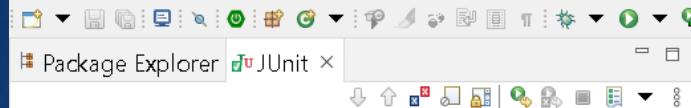
The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - springboot-test/src/test/java/com/formacionspringboot/app/ProductoTest.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations. The left sidebar is the Package Explorer, showing the project structure:

- Repos
- segundo [boot] [devtools]
- server-discovery [boot] [github_formacion master]
- springboot-apirest [boot] [devtools] [repository mast]
- springboot-test [boot] [devtools]
 - src/main/java
 - com.formacionspringboot.app
 - SpringbootTestApplication.java
 - com.formacionspringboot.app.dao
 - ProductoDao.java
 - com.formacionspringboot.app.entity
 - Producto.java
 - src/main/resources
 - application.properties
 - src/test/java
 - com.formacionspringboot.app
 - ProductoTest.java
 - ProductoTest
 - testBuscarProductoPorNombre()
 - testGuardarProducto()
 - SpringbootTestApplicationTests.java
- JRE System Library [JavaSE-11]
- Maven Dependencies
- src
- target

```
17 public class ProductoTest {  
18  
19     @Autowired  
20     private ProductoDao productoDao;  
21  
22     @Test  
23     @Rollback(false)  
24     public void testGuardarProducto() {  
25         Producto producto = new Producto("TV Samsung HD", 4000);  
26         Producto productoGuardado = productoDao.save(producto);  
27  
28         assertNotNull(productoGuardado);  
29     }  
30     @Test  
31     public void testBuscarProductoPorNombre() {  
32         String nombre= "TV Samsung HD";  
33         Producto producto = productoDao.findByNombre(nombre);  
34  
35         assertThat(producto.getNombre()).isEqualTo(nombre);  
36     }  
37  
38 }
```

edipse-workspace - springboot-test/src/test/java/com/formacionspringboot/app/ProductoTest.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help



Finished after 6,38 seconds

Runs: 1/1 Errors: 0 Failures: 0

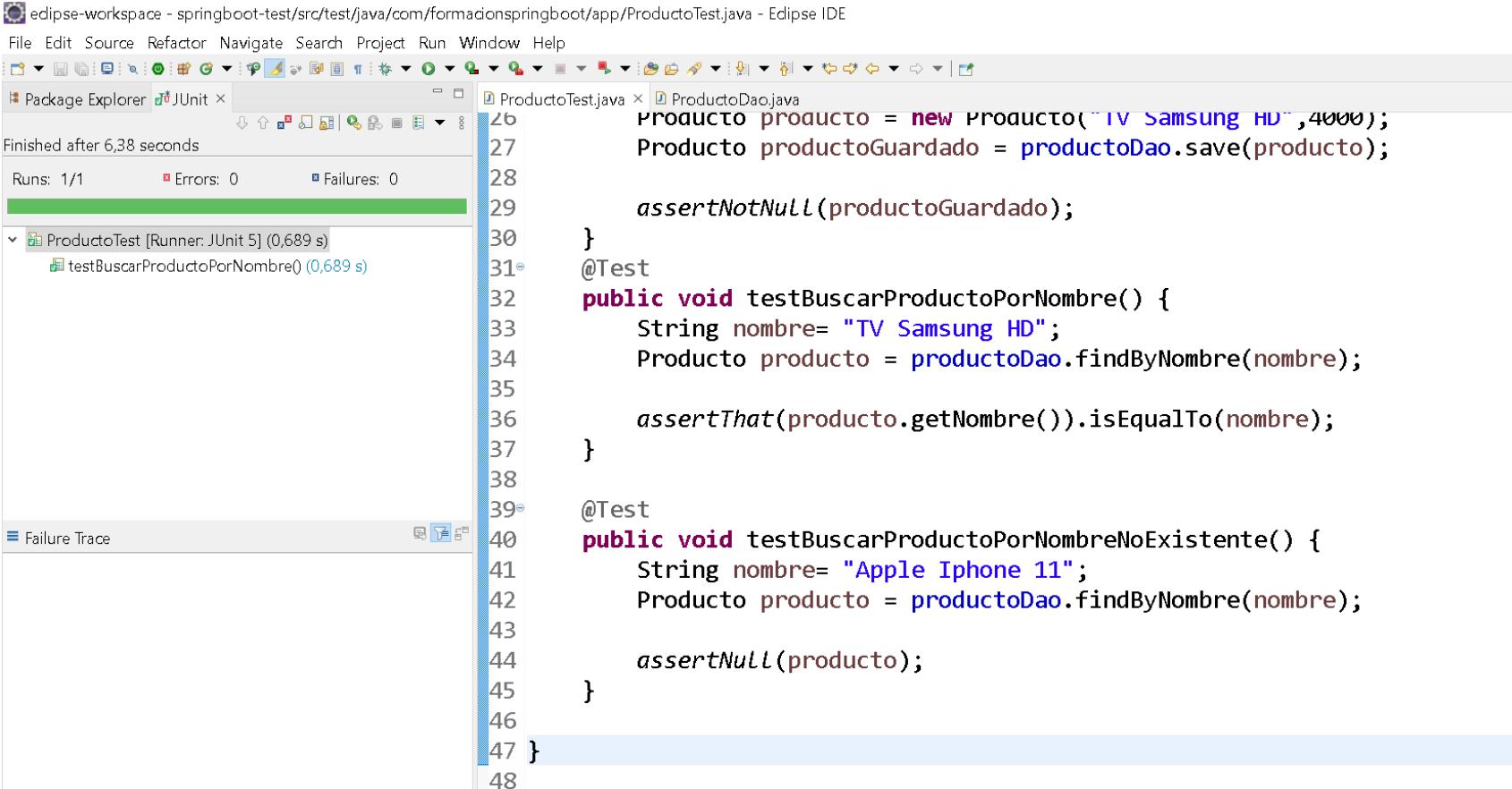
ProductoTest [Runner: JUnit 5] (0,689 s)
 └ testBuscarProductoPorNombre() (0,689 s)

Failure Trace

```
17 public class ProductoTest {  
18  
19     @Autowired  
20     private ProductoDao productoDao;  
21  
22     @Test  
23     @Rollback(false)  
24     public void testGuardarProducto() {  
25         Producto producto = new Producto("TV Samsung HD",4000);  
26         Producto productoGuardado = productoDao.save(producto);  
27  
28         assertNotNull(productoGuardado);  
29     }  
30     @Test  
31     public void testBuscarProductoPorNombre() {  
32         String nombre= "TV Samsung HD";  
33         Producto producto = productoDao.findByNombre(nombre);  
34  
35         assertThat(producto.getNombre()).isEqualTo(nombre);  
36     }  
37  
38 }
```

```
ProductoTest.java x ProductoDao.java
26     Producto producto = new Producto("TV Samsung HD",4000);
27     Producto productoGuardado = productoDao.save(producto);
28
29     assertNotNull(productoGuardado);
30 }
31 @Test
32 public void testBuscarProductoPorNombre() {
33     String nombre= "TV Samsung HD";
34     Producto producto = productoDao.findByNombre(nombre);
35
36     assertThat(producto.getNombre()).isEqualTo(nombre);
37 }
38
39 @Test
40 public void testBuscarProductoPorNombreNoExistente() {
41     String nombre= "Apple Iphone 11";
42     Producto producto = productoDao.findByNombre(nombre);
43
44     assertNull(producto);
45 }
46
47 }
48 <
```

ejecutamos



The screenshot shows the Eclipse IDE interface with the following details:

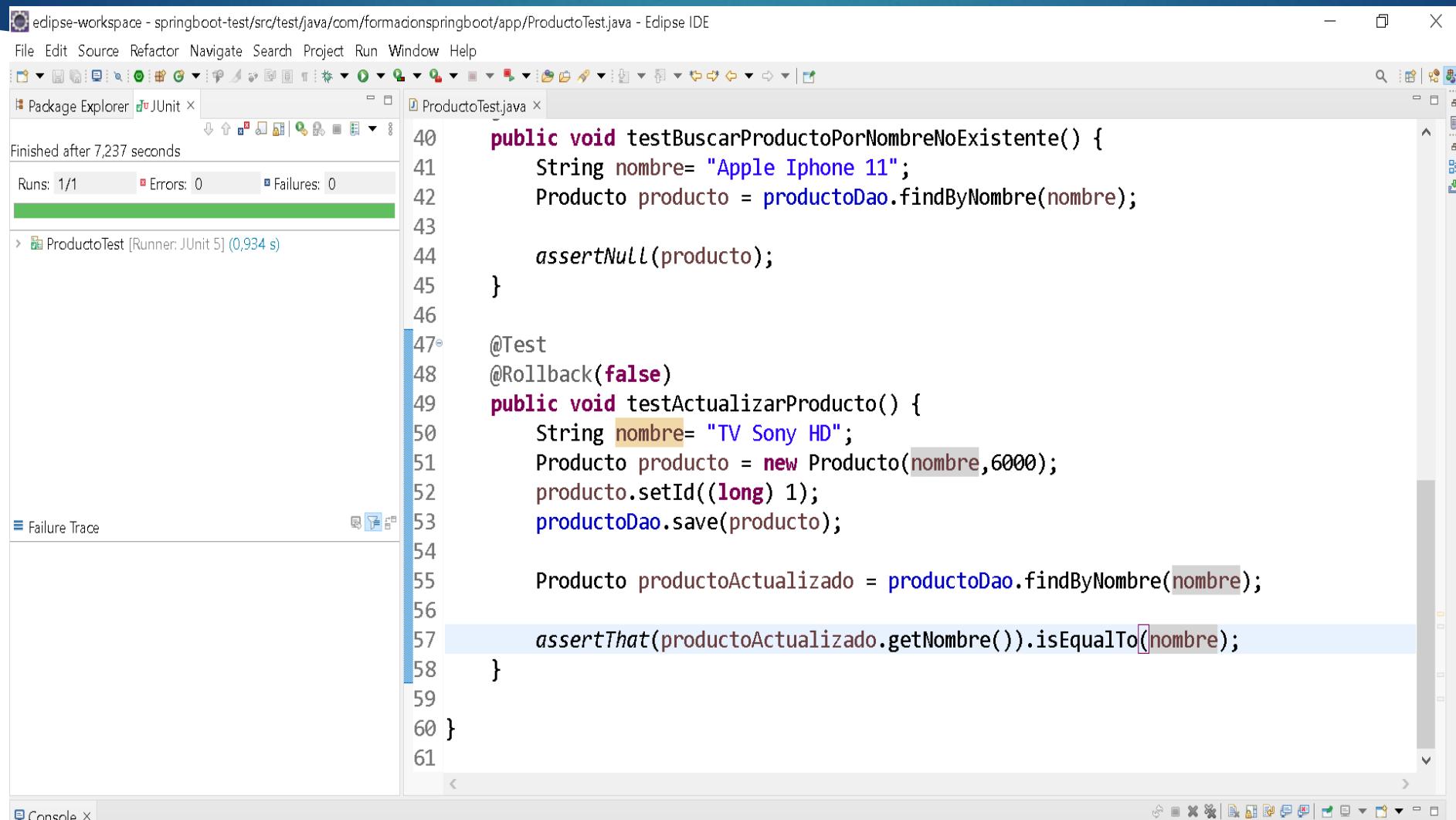
- Title Bar:** eclipse-workspace - springboot-test/src/test/java/com/formacionspringboot/app/Productotest.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar icons.
- Left Sidebar:** Package Explorer (selected), JUnit (open), and a failure trace table.
- Middle Area:** A status bar indicating "Finished after 6,38 seconds" with "Runs: 1/1", "Errors: 0", and "Failures: 0".
- Right Area:** The code editor displays `ProductoTest.java` with the following content:

```
26 Producto producto = new Producto("TV Samsung HD",4000);
27 Producto productoGuardado = productoDao.save(producto);
28
29 assertNotNULL(productoGuardado);
30 }
31 @Test
32 public void testBuscarProductoPorNombre() {
33     String nombre= "TV Samsung HD";
34     Producto producto = productoDao.findByNombre(nombre);
35
36     assertThat(producto.getNombre()).isEqualTo(nombre);
37 }
38
39 @Test
40 public void testBuscarProductoPorNombreNoExistente() {
41     String nombre= "Apple Iphone 11";
42     Producto producto = productoDao.findByNombre(nombre);
43
44     assertNULL(producto);
45 }
46
47 }
48 }
```

Método actualizar

```
ProductoTest.java ×
0  public void testBuscarProductoPorNombreNoExistente() {
1      String nombre= "Apple Iphone 11";
2      Producto producto = productoDao.findByNombre(nombre);
3
4      assertNull(producto);
5  }
6
7  @Test
8  @Rollback(false)
9  public void testActualizarProducto() {
0      String nombre= "TV Sony HD";
1      Producto producto = new Producto(nombre,6000);
2      producto.setId((long) 1);
3      productoDao.save(producto);
4
5      Producto productoActualizado = productoDao.findByNombre(nombre);
6
7      assertThat(productoActualizado.getNombre()).isEqualTo(nombre);
8  }
9
0 }
```

Ejecutamos

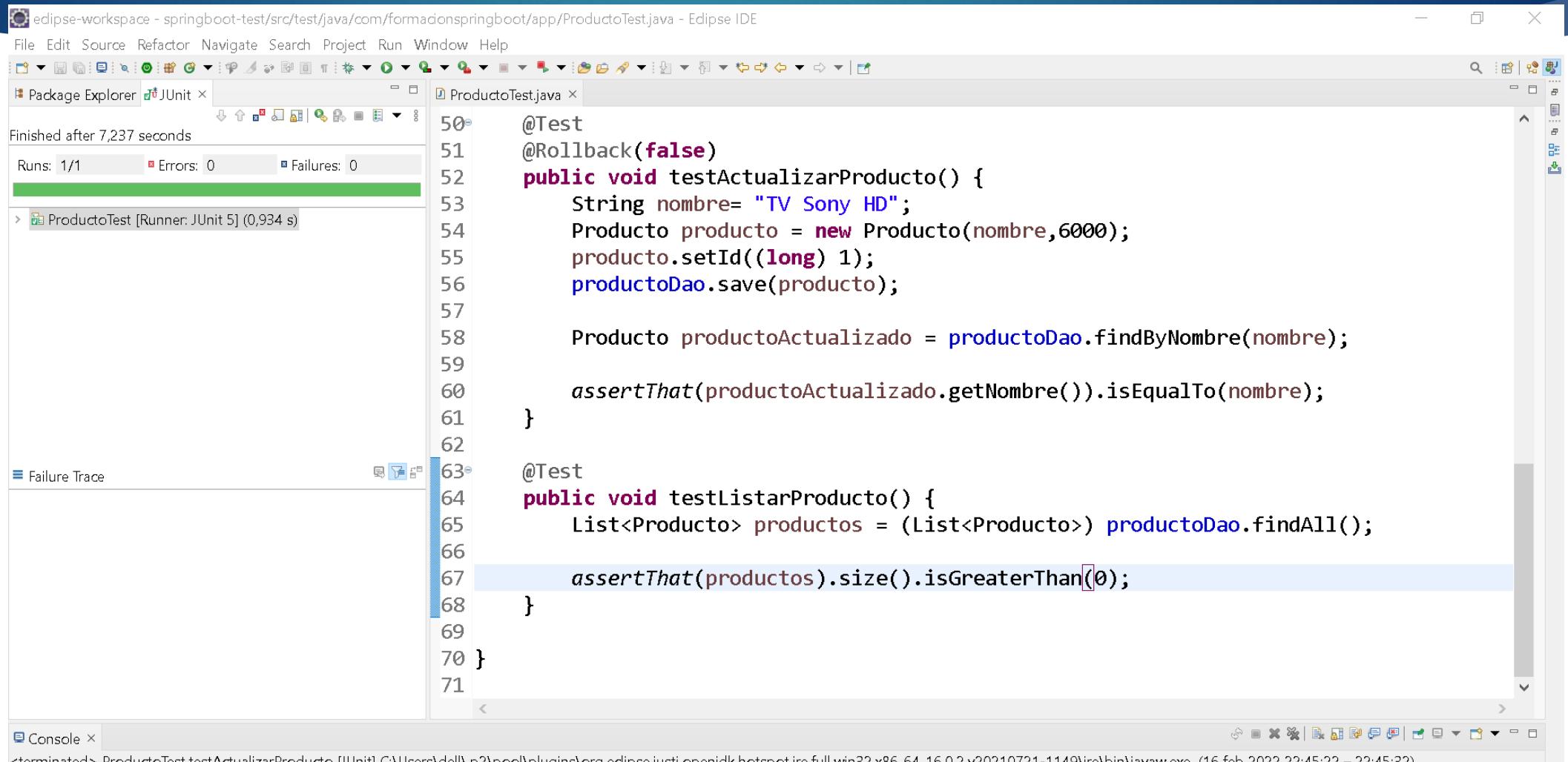


```
eclipse-workspace - springboot-test/src/test/java/com/formacionspringboot/app/ProductoTest.java - Edipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit x
Finished after 7,237 seconds
Runs: 1/1 Errors: 0 Failures: 0
ProductTest [Runner: JUnit 5] (0,934 s)
public void testBuscarProductoPorNombreNoExistente() {
    String nombre= "Apple Iphone 11";
    Producto producto = productoDao.findByNombre(nombre);
    assertNull(producto);
}

@Test
@Rollback(false)
public void testActualizarProducto() {
    String nombre= "TV Sony HD";
    Producto producto = new Producto(nombre,6000);
    producto.setId((long) 1);
    productoDao.save(producto);

    Producto productoActualizado = productoDao.findByNombre(nombre);
    assertThat(productoActualizado.getNombre()).isEqualTo(nombre);
}
```

Método Listar



The screenshot shows the Eclipse IDE interface with the following details:

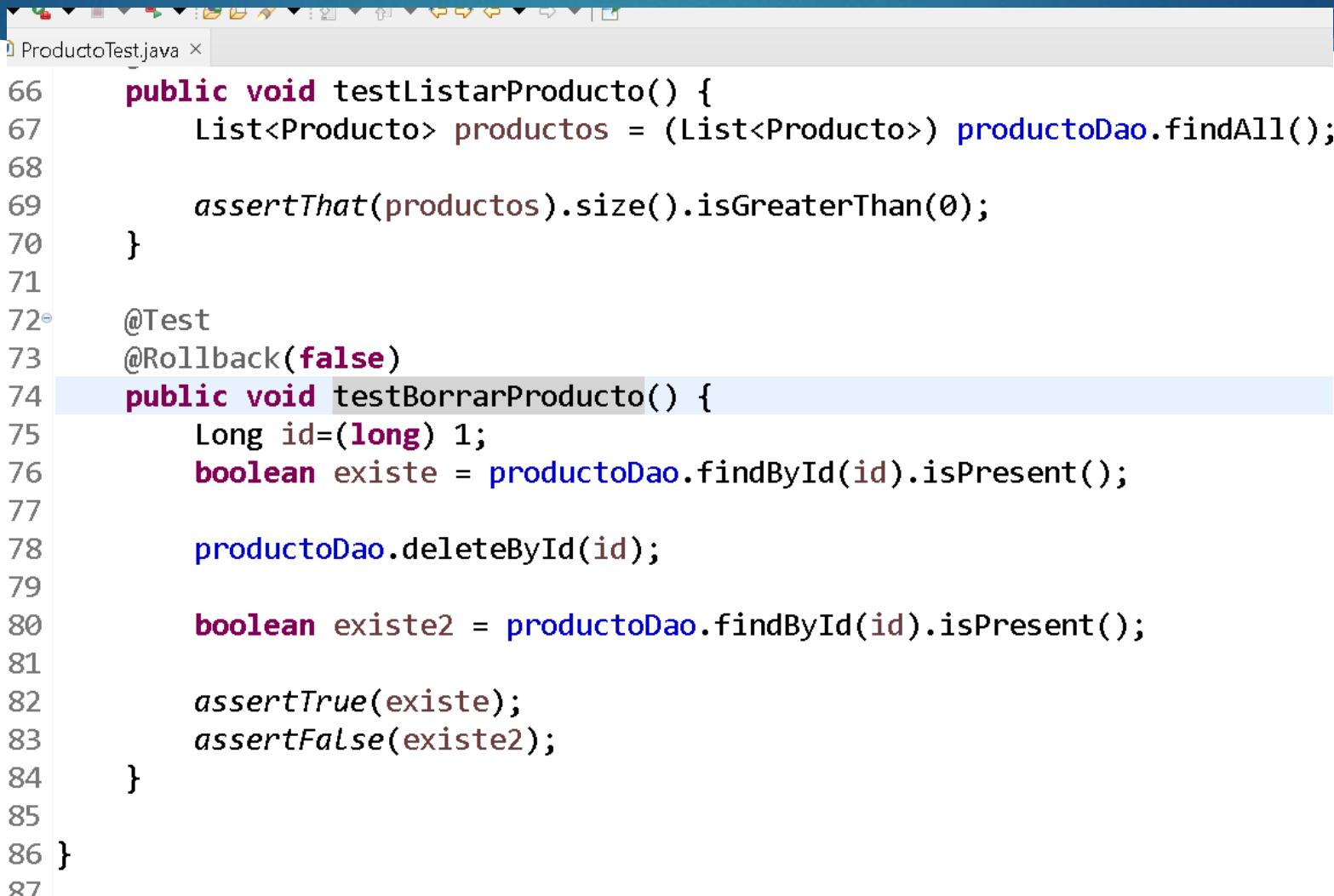
- Title Bar:** eclipse-workspace - springboot-test/src/test/java/com/formacionspringboot/app/ProductoTest.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar icons.
- Left Sidebar:** Package Explorer (selected), JUnit (green icon). Status message: Finished after 7,237 seconds. Run statistics: Runs: 1/1, Errors: 0, Failures: 0.
- Central Area:** Code editor for `ProductoTest.java`. The code contains two test methods: `testActualizarProducto()` and `testListarProducto()`.
- Code Editor Content:**

```
50  @Test
51  @Rollback(false)
52  public void testActualizarProducto() {
53      String nombre= "TV Sony HD";
54      Producto producto = new Producto(nombre,6000);
55      producto.setId((long) 1);
56      productoDao.save(producto);
57
58      Producto productoActualizado = productoDao.findByNombre(nombre);
59
60      assertThat(productoActualizado.getNombre()).isEqualTo(nombre);
61  }
62
63  @Test
64  public void testListarProducto() {
65      List<Producto> productos = (List<Producto>) productoDao.findAll();
66
67      assertThat(productos.size()).isGreaterThan(0);
68  }
69
70 }
71 }
```
- Bottom Status Bar:** Shows the command prompt output: `terminated: [java] [JUnit] C:\Users\diego\OneDrive\Documentos\NetBeansProjects\formacionspringboot\target\classes\com\formacionspringboot\app\ProductoTest.java:16: error: cannot find symbol` followed by the timestamp `(16 feb 2022 22:45:22 - 22:45:22)`.

Ejecutamos

```
eclipse-workspace - springboot-test/src/test/java/com/formationspringboot/app/ProductoTest.java - Edipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit x
Finished after 7,237 seconds
Runs: 1/1 Errors: 0 Failures: 0
> ProductoTest [Runner: JUnit 5] (0.934 s)
50  @Test
51  @Rollback(false)
52  public void testActualizarProducto() {
53      String nombre= "TV Sony HD";
54      Producto producto = new Producto(nombre,6000);
55      producto.setId((long) 1);
56      productoDao.save(producto);
57
58      Producto productoActualizado = productoDao.findByNombre(nombre);
59
60      assertThat(productoActualizado.getNombre()).isEqualTo(nombre);
61  }
62
63  @Test
64  public void testListarProducto() {
65      List<Producto> productos = (List<Producto>) productoDao.findAll();
66
67      assertThat(productos.size()).isGreaterThan(0);
68  }
69
70 }
71
Console x
<terminated> ProductoTest.testActualizarProducto [JUnit] C:\Users\dell\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.2.v20210721-1149\jre\bin\javaw.exe (16 feb 2022 22:45:22 - 22:45:32)
```

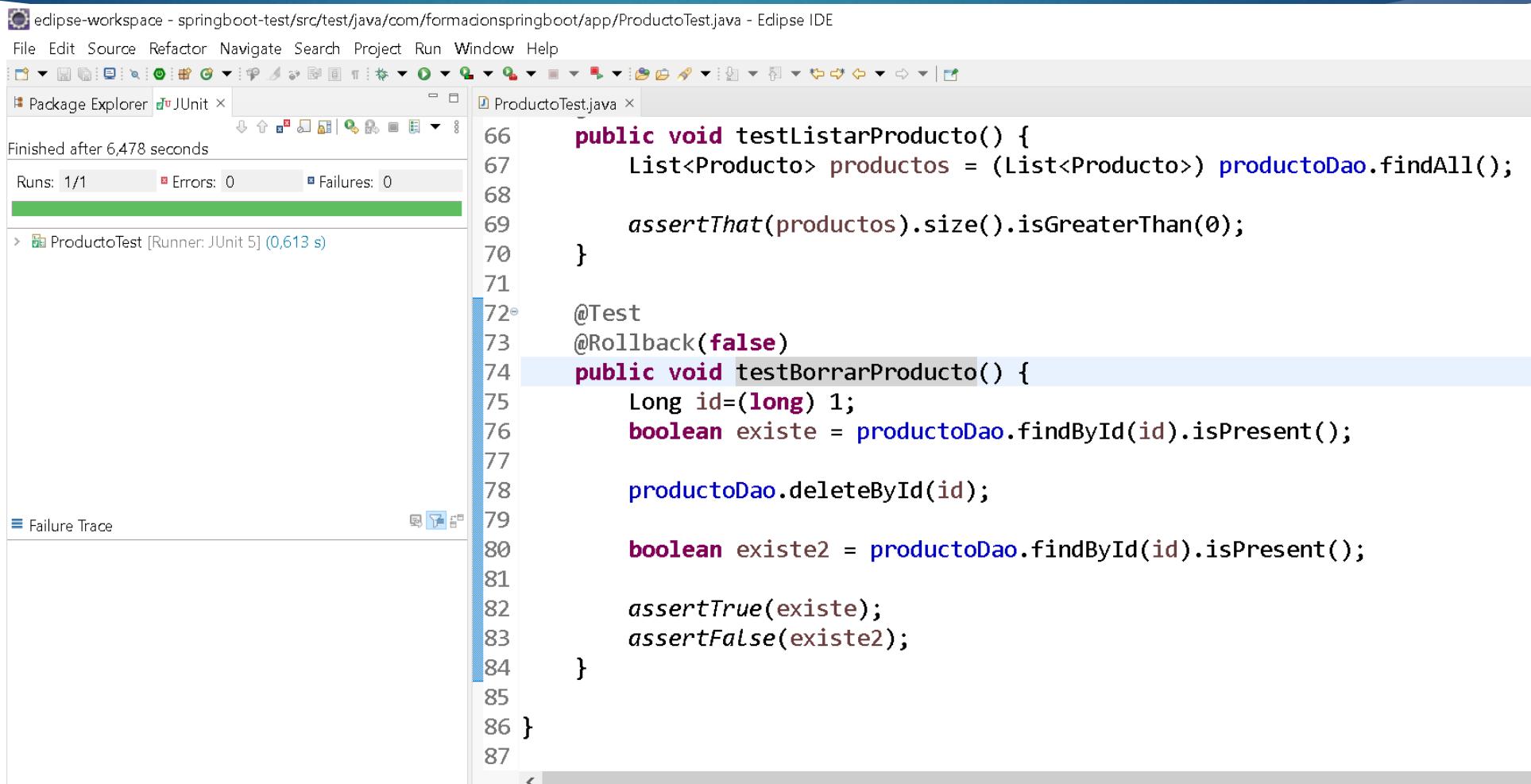
Método eliminar



The screenshot shows a Java test class named `ProductoTest.java` in an IDE. The code is annotated with line numbers and highlights specific sections. The highlighted section is the `testBorrarProducto()` method, which demonstrates how to delete a product from a database and verify its absence.

```
1  package com.empresa.modelo.test;
2
3  import org.junit.*;
4  import static org.junit.Assert.*;
5  import com.empresa.modelo.*;
6  import com.empresa.modelo.dao.*;
7
8
9  public class ProductoTest {
10
11     @Test
12     @Rollback(false)
13     public void testListarProducto() {
14         List<Producto> productos = (List<Producto>) productoDao.findAll();
15
16         assertThat(productos).size().isGreaterThanOrEqualTo(0);
17     }
18
19
20     @Test
21     @Rollback(false)
22     public void testBorrarProducto() {
23         Long id=(long) 1;
24         boolean existe = productoDao.findById(id).isPresent();
25
26         productoDao.deleteById(id);
27
28         boolean existe2 = productoDao.findById(id).isPresent();
29
30         assertTrue(existe);
31         assertFalse(existe2);
32     }
33
34 }
```

Ejecutamos



The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - springboot-test/src/test/java/com/formacionspringboot/app/ProductoTest.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, Find, and Run.

The left sidebar displays the "Package Explorer" and "JUnit" view. The "JUnit" view shows a summary: "Finished after 6,478 seconds", "Runs: 1/1", "Errors: 0", and "Failures: 0". Below this, it lists "ProductoTest [Runner: JUnit 5] (0,613 s)".

The main editor area contains the Java code for "ProductoTest.java". The code defines two test methods: "testListarProducto" and "testBorrarProducto". The "testBorrarProducto" method is currently selected and highlighted with a blue background.

```
public void testListarProducto() {
    List<Producto> productos = (List<Producto>) productoDao.findAll();
    assertThat(productos.size()).isGreaterThan(0);
}

@Test
@Rollback(false)
public void testBorrarProducto() {
    Long id=(long) 1;
    boolean existe = productoDao.findById(id).isPresent();

    productoDao.deleteById(id);

    boolean existe2 = productoDao.findById(id).isPresent();

    assertTrue(existe);
    assertFalse(existe2);
}
```