

Taller Extracurricular de Verano 2022

“Spring Data JPA - Persistencia de Datos”

Instructor:

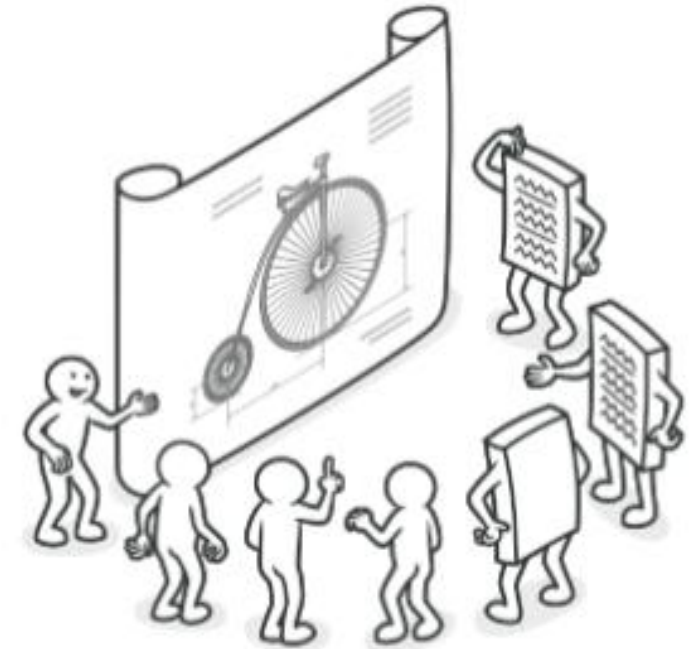
- **Dr. Christian Mauricio Castillo Estrada**

Mayo de 2022

Design Patterns

Los **patrones de diseño (Design Patterns)** son soluciones habituales a problemas comunes en el diseño de software. Cada patrón es como un plano que se puede personalizar para resolver un problema de diseño particular de cierto código.

El patrón no es una porción específica de código, sino un concepto general para resolver un problema particular. Puedes seguir los detalles del patrón e implementar una solución que encaje con las realidades de un software.



Patrones de Diseño

- Los patrones son un juego de herramientas que brindan soluciones a problemas habituales en el diseño de software.
- Una analogía de un algoritmo sería una receta de cocina: ambos cuentan con pasos claros para alcanzar una meta, es decir, preparar un platillo. Por otra parte, un patrón de diseño es semejante a un plano arquitectónico, puedes imaginar y observar el resultado y sus funciones, pero el orden exacto de la implementación depende del desarrollador.



Data Access Object (DAO)

Un **Data Access Object** implementa los mecanismos de acceso requeridos para trabajar con la fuente de datos. (...) Los DAOs ocultan completamente los detalles de implementación de la fuente de datos a sus clientes. Ya que la interfaz expuesta por los DAOs a sus clientes no cambia cuando la implementación de la fuente de datos subyacente cambia, este patrón permite adaptarse a diferentes esquemas de almacenamiento sin afectar a sus clientes.

Esencialmente, DAO actúa como un adaptador entre los componentes y la fuente de datos. La forma más común de este **patrón es una clase que contiene operaciones CRUD** para cada tipo de entidad del modelo de dominio.



C R U D

C CREATE

R READ

U UPDATE

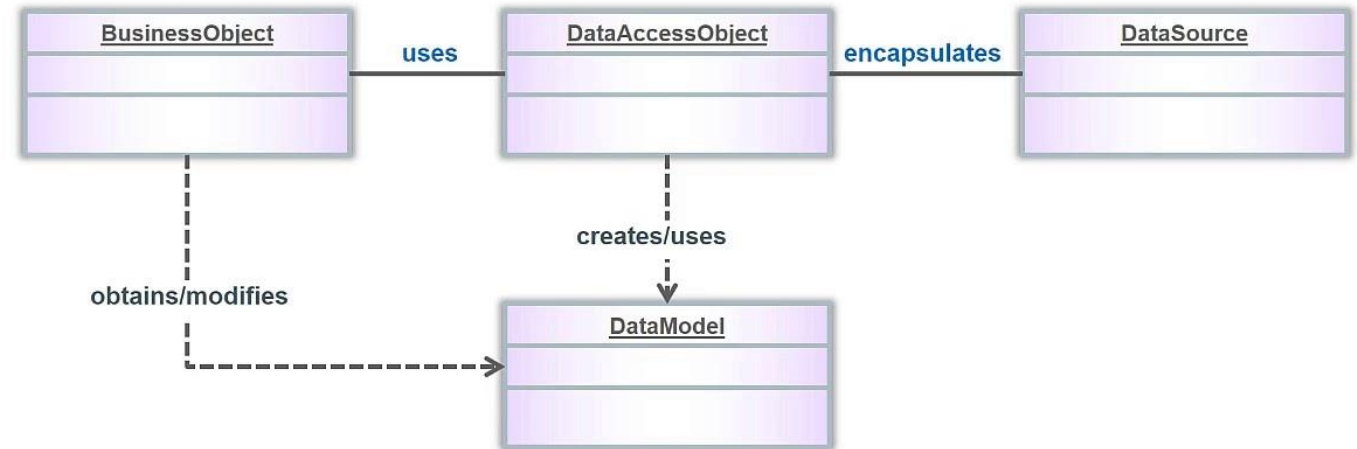
D DELETE

Data Access Object (DAO)

El patrón **Data Access Object (DAO)** pretende principalmente independizar la aplicación de la forma de acceder a la base de datos, o cualquier otro tipo de repositorio de datos.

Para ello se centraliza el código relativo al acceso al repositorio de datos en las clases llamadas DAO. Fuera de las clases DAO no debe haber ningún tipo de código que acceda al repositorio de datos.

The Data Access Object (DAO) Pattern



ORACLE

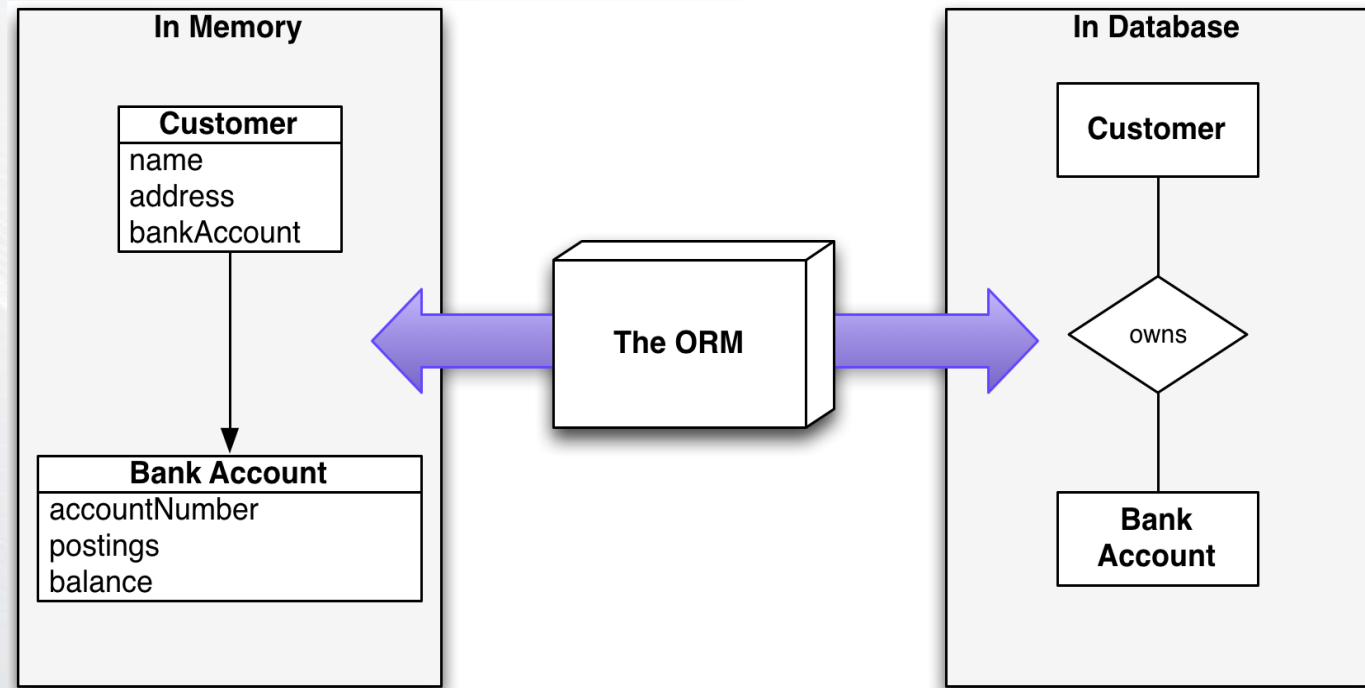
Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

12 - 1

ORM

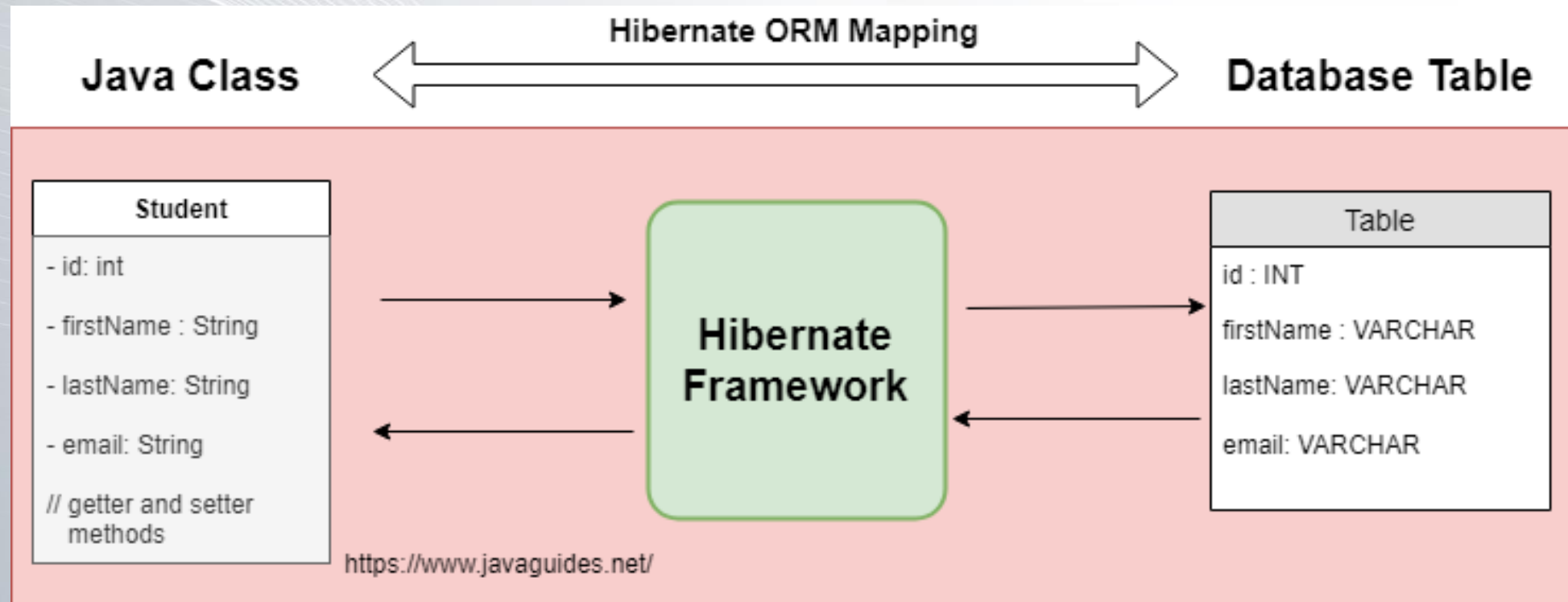
Las siglas ORM significan “**Object-Relational Mapping**” y en español “**Mapeo Objeto-Relacional**”. El ORM es simplemente el código que escribimos para manipular la información de nuestras clases en una base de datos relacional.

“Hibernate es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones.”





Hibernate ORM



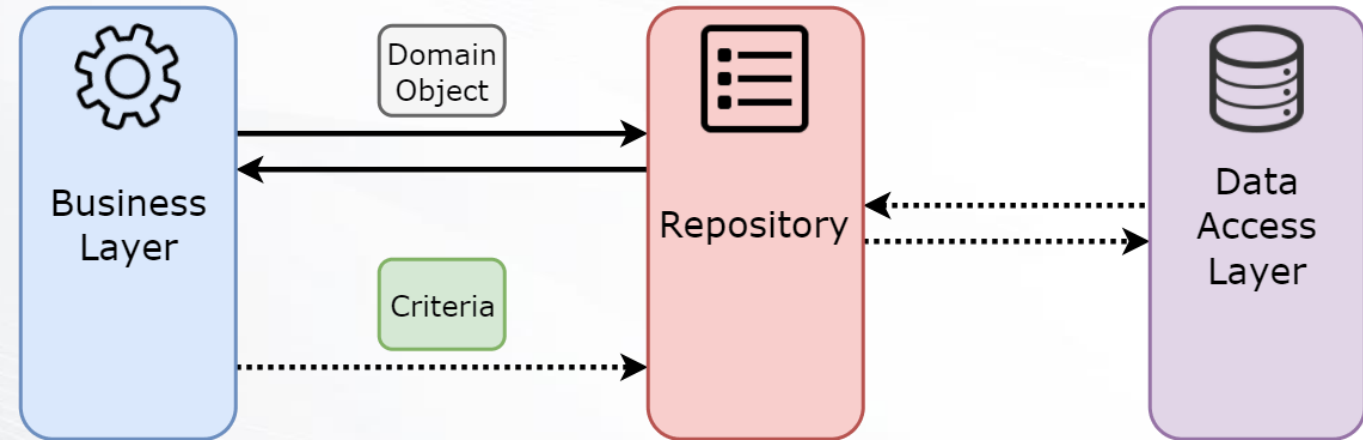
Hibernate usa JDBC para todas las comunicaciones de la base de datos; por tal motivo, requiere de la especificación de un driver JDBC para interactuar con la base de datos relacional como PostgreSQL, MySQL, DB2, etc.

Patrón de Diseño Repository

Según la definición original de **Martin Fowler** en su libro **Patterns of Enterprise Application Architecture**:

Conceptualmente, un Repositorio encapsula el conjunto de objetos persistidos en un almacén de datos y las operaciones realizadas sobre ellos, proveyendo una concepción mas orientada a objetos de la capa de persistencia. Además, apoya el objetivo de lograr una separación limpia y una dependencia en un solo sentido entre el dominio y la capa de acceso a datos. (...). Un Repository reduce la cantidad de código necesitada para tratar con todas las consultas que se llevan a cabo.

El proyecto **Spring Data JPA** permite a los desarrolladores de aplicaciones trabajar con almacenes de datos usando una interfaz consistente que hace uso de una abstracción llamada Repositorio.



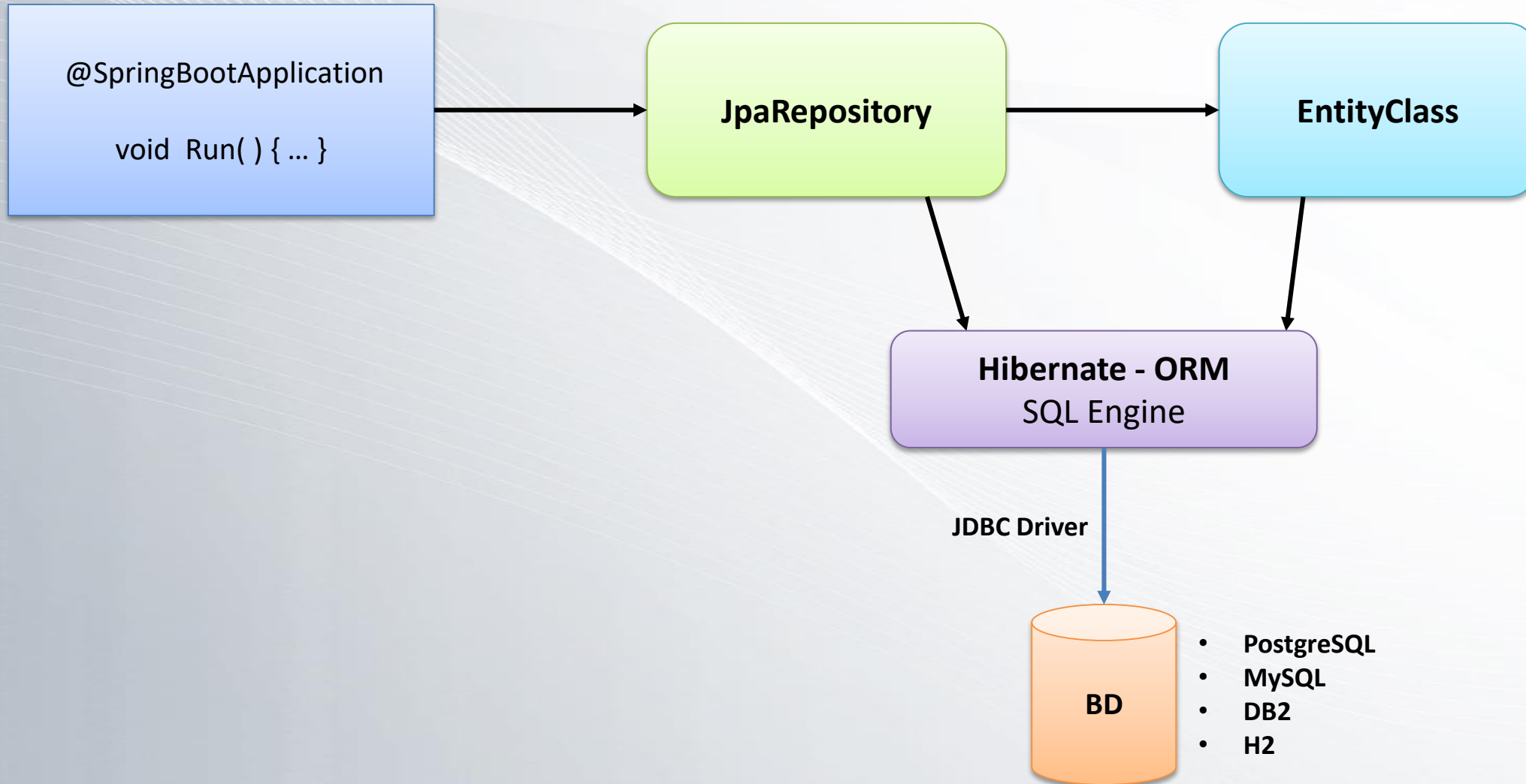
Un repositorio de datos de Spring se modela a partir del patrón de diseño denominado repositorio popularizado por el diseño basado en dominios. Además de la interfaz del Repositorio, Spring Data también proporciona dos interfaces principales más: **CrudRepository** que define el contrato para la funcionalidad CRUD básica (crear, leer, actualizar y eliminar); y **PagingAndSortingRepository** que amplía CrudRepository definiendo un contrato para paginación y clasificación.

Spring Data

<https://springframework.guru/spring-boot-web-application-part-3-spring-data-jpa/>

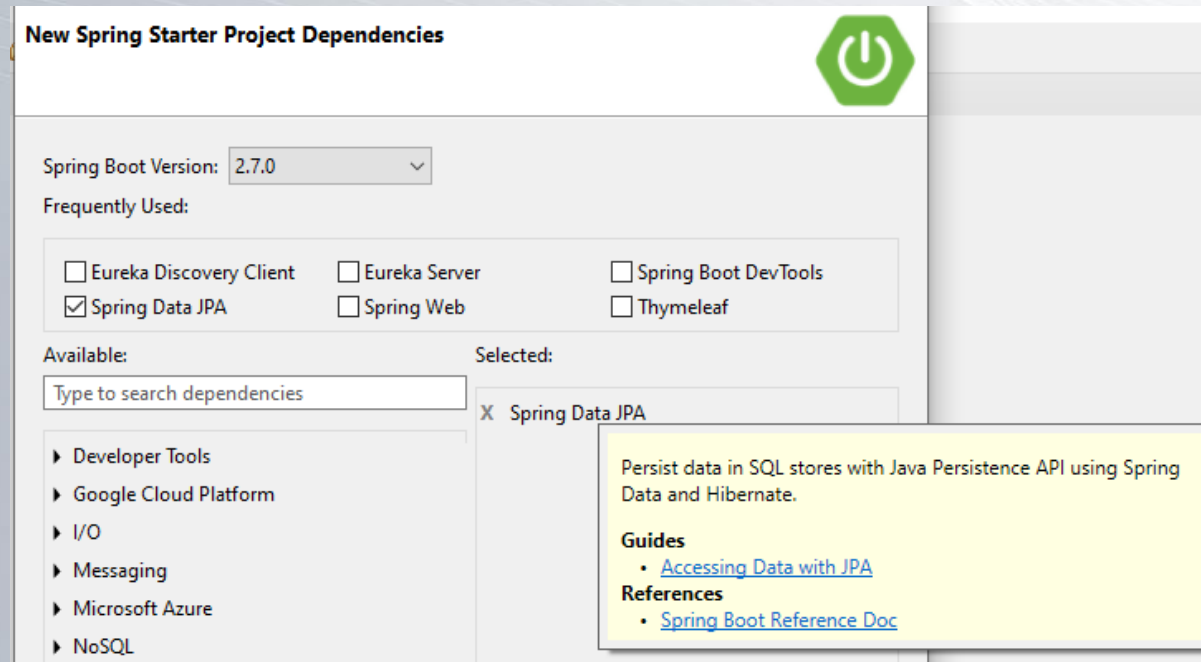
El uso de Spring Data JPA proporciona un significativo ahorro de tiempo al interactuar con la base de datos. Spring Data JPA implementa el patrón de repositorio. Este patrón de diseño fue definido originalmente por Eric Evans y Martin Fowler, en su libro Domain Driven Design.

Una alternativa común a Spring Data JPA sería usar el patrón DAO ampliamente aceptado. El patrón DAO es muy similar al patrón Repository. La ventaja de usar Spring Data JPA es que escribirá mucho menos código. Spring Data JPA funciona de manera muy similar a Spring Integration Gateways, donde usted define una interfaz y Spring proporciona la implementación en tiempo de ejecución.



Hibernate y Spring Boot

Cuando se incluye la dependencia Spring Data JPA en el Maven POM, Hibernate se incluye de forma predeterminada. Las dependencias de Spring Data JPA incluyen Hibernate. Por tal motivo, Spring Boot configurará automáticamente las propiedades predeterminadas, para ello es necesario especificar la dependencia necesaria dentro del archivo pom.xml



```
<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>
</dependencies>
```

New Spring Starter Project

Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

Description:

Package:

Working sets

☒ Add project to working sets

Working sets:

New Spring Starter Project Dependencies

Spring Boot Version:

Frequently Used:

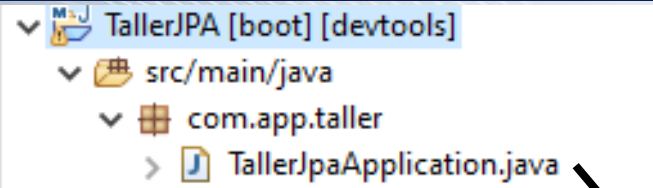
☐ Eureka Discovery Client ☐ Eureka Server ☒ Spring Boot DevTools

☒ Spring Data JPA ☐ Spring Web ☐ Thymeleaf

Available: Selected:

☒ MySQL Driver

X Spring Boot DevTools
X Spring Data JPA
X MySQL Driver



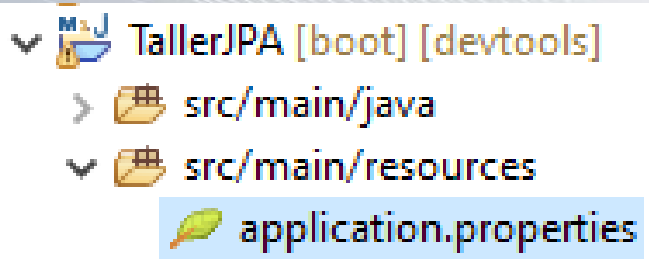
Opción 1

```
1 package com.app.taller;
2 import org.springframework.boot.CommandLineRunner;
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class TallerJpaApplication implements CommandLineRunner {
8
9     public static void main(String[] args) {
10         SpringApplication.run(TallerJpaApplication.class, args);
11     }
12
13     public void run(String... args) throws Exception {
14         System.out.println("Bienvenidos al Taller de Spring DATA JPA ");
15     }
16
17 }
18
```

▼ TallerJPA [boot] [devtools]
▼ src/main/java
▼ com.app.taller
> TallerJpaApplication.java

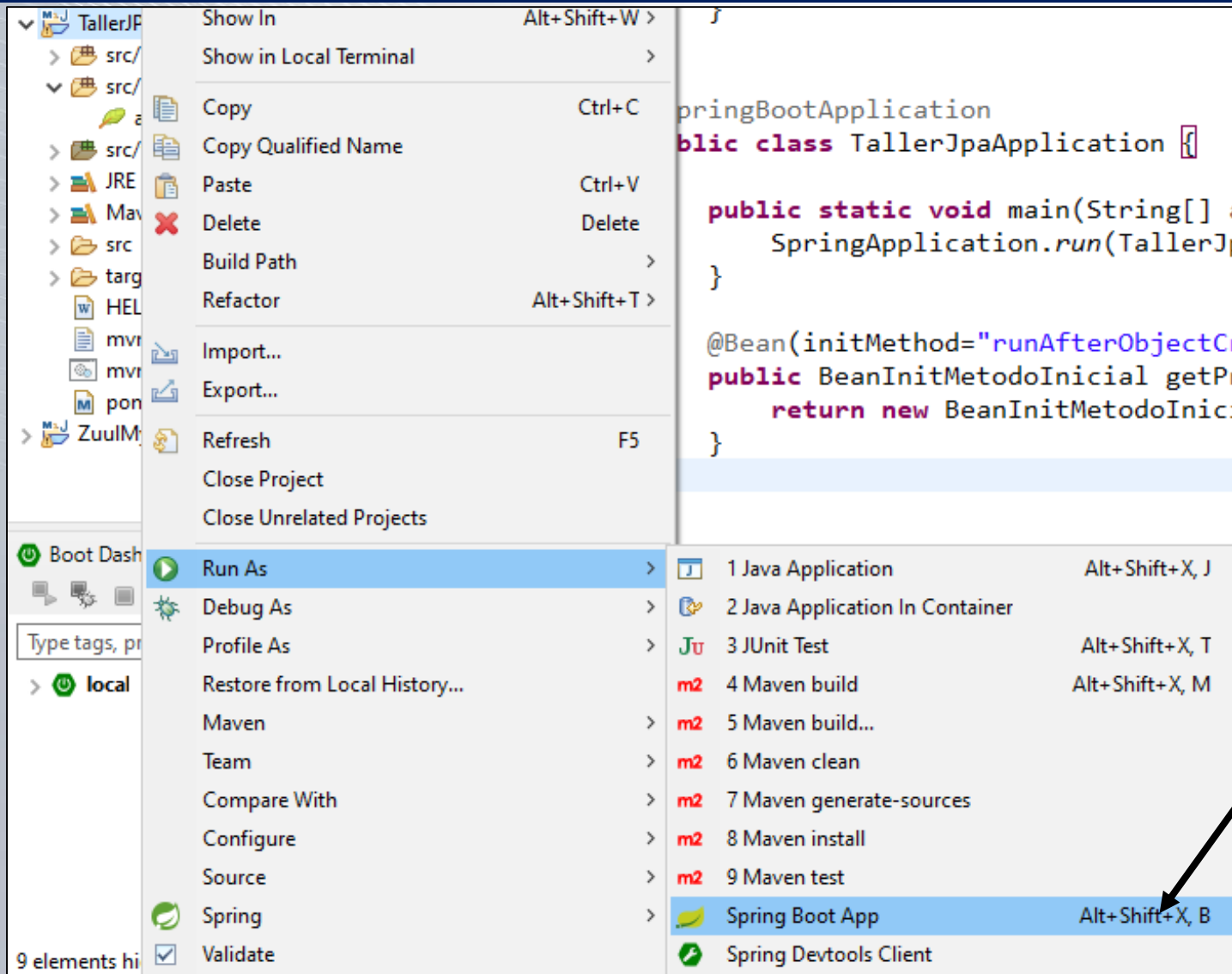
Opción 2

```
1 package com.app.taller;
2 import org.springframework.boot.SpringApplication;
3 import org.springframework.boot.autoconfigure.SpringBootApplication;
4 import org.springframework.context.annotation.Bean;
5 import com.app.taller.BeanInitMetodoInicial;
6
7 class BeanInitMetodoInicial{
8     public void runAfterObjectCreated() {
9         System.out.println("Bienvenidos al Taller de Spring DATA JPA ");
10    }
11 }
12
13 @SpringBootApplication
14 public class TallerJpaApplication {
15
16     public static void main(String[] args) {
17         SpringApplication.run(TallerJpaApplication.class, args);
18     }
19
20     @Bean(initMethod="runAfterObjectCreated")
21     public BeanInitMetodoInicial getPrincipalBean() {
22         return new BeanInitMetodoInicial();
23     }
24 }
25
```



TallerJPA [boot] [devtools]
src/main/java
src/main/resources
application.properties

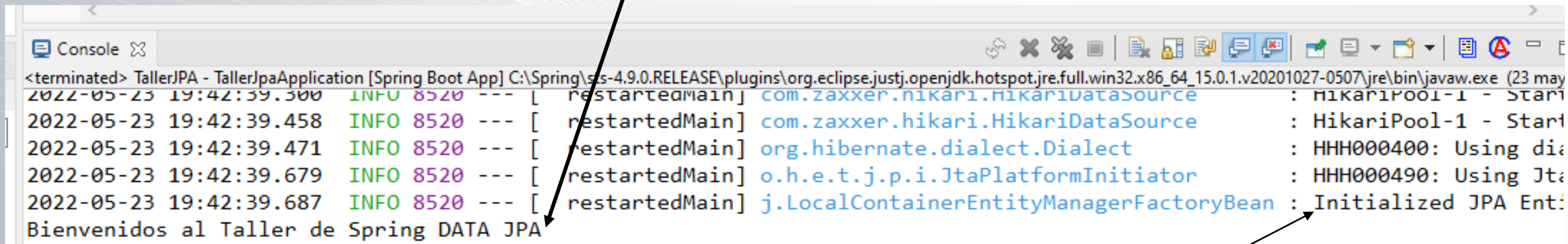
```
1 # DATASOURCE (MYSQL 8.0)
2 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
3 spring.datasource.url=jdbc:mysql://localhost:3306/empresa?serverTimezone=${user.timezone}
4 spring.datasource.username = root
5 spring.datasource.password = unach22
6 #JPA
7 spring.jpa.show-sql=true
8 spring.jpa.hibernate.ddl-auto=update
9 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
10 # Table names physically
11 spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
12 |
13
```

Para ejecutar el proyecto, se debe seleccionar la opción:

Run As -> Spring Boot App

Se imprime en mensaje de texto en la consola del IDE
Spring Tool Suite



```
<terminated> TallerJPA - TallerJpaApplication [Spring Boot App] C:\Spring\s4s-4.9.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.1.v20201027-0507\jre\bin\javaw.exe (23 may
2022-05-23 19:42:39.300 INFO 8520 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start
2022-05-23 19:42:39.458 INFO 8520 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start
2022-05-23 19:42:39.471 INFO 8520 --- [ restartedMain] org.hibernate.dialect.Dialect : HHH000400: Using dia
2022-05-23 19:42:39.679 INFO 8520 --- [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using Jta
2022-05-23 19:42:39.687 INFO 8520 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA Enti
Bienvenidos al Taller de Spring DATA JPA
```

Se inicializa una instancia del Entity Manager del JPA - Hibernate

Initialized JPA EntityManagerFactory for persistence unit

GRACIAS