



Spring Boot – Spring Data JPA

Sesión 8



¿Qué es Spring Boot?

Spring Boot es el punto de comienzo para crear aplicaciones basadas en Spring Framework, Spring boot esta diseñado para ejecutarse lo más rápido y con la mínimo configuración posible.



Spring Boot Starters

Starters son un conjunto de dependencias que tu puedes incluir en tu proyecto, en una sola parada, sin la necesidad que necesites incluir cada una de las dependencias de manera independiente.

Usualmente estos starters son incluidos en tus archivos pom (Maven) o build (Gradle).



Autoconfiguración

La Autoconfiguración de Spring Boot, intenta de manera automática configurar la aplicación Spring basada en las dependencias que fueron agregadas.

Lo que permite al usuario no preocuparse por configurar recursos, estos serian algunos de los recursos que son automáticamente configurados por Spring Boot:

- DispatcherServlet
- Datasources
- EntityManager
- JSON Marshallers



Spring Framework

Como definición podemos decir que **Spring es un framework de código abierto para la creación de aplicaciones empresariales Java**, con soporte para Groovy y Kotlin. Tiene una estructura modular y una gran flexibilidad para implementar diferentes tipos de arquitectura según las necesidades de la aplicación.

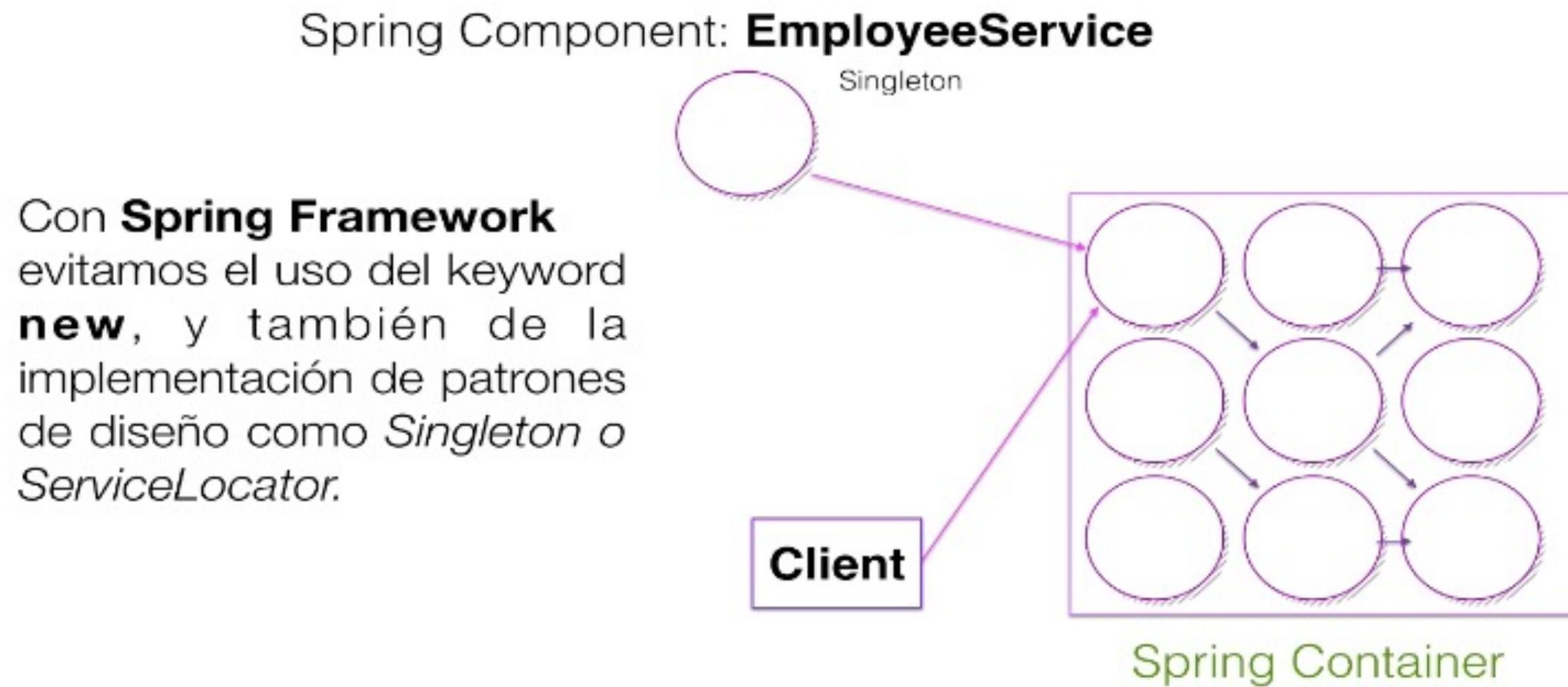
A diferencia de una librería, un framework es:

- **Un conjunto de artefactos software**, es decir, que puede incluir una librería, de conceptos y de metodologías.
- Nos provee de un **mecanismo genérico para resolver uno o más problemas** de un tipo determinado.
- **Es extensible** a través de código escrito por los usuarios.
- Ofrece **facilidad para el desarrollo y despliegue**.

Si tuviéramos que desarrollar una aplicación web, podríamos **utilizar un framework que nos facilite la tarea, que nos aporte soluciones a ese desarrollo**. Uno de ellos podría ser, por ejemplo Spring MVC, que nos permitiría crear fácilmente una aplicación web, ya que nos



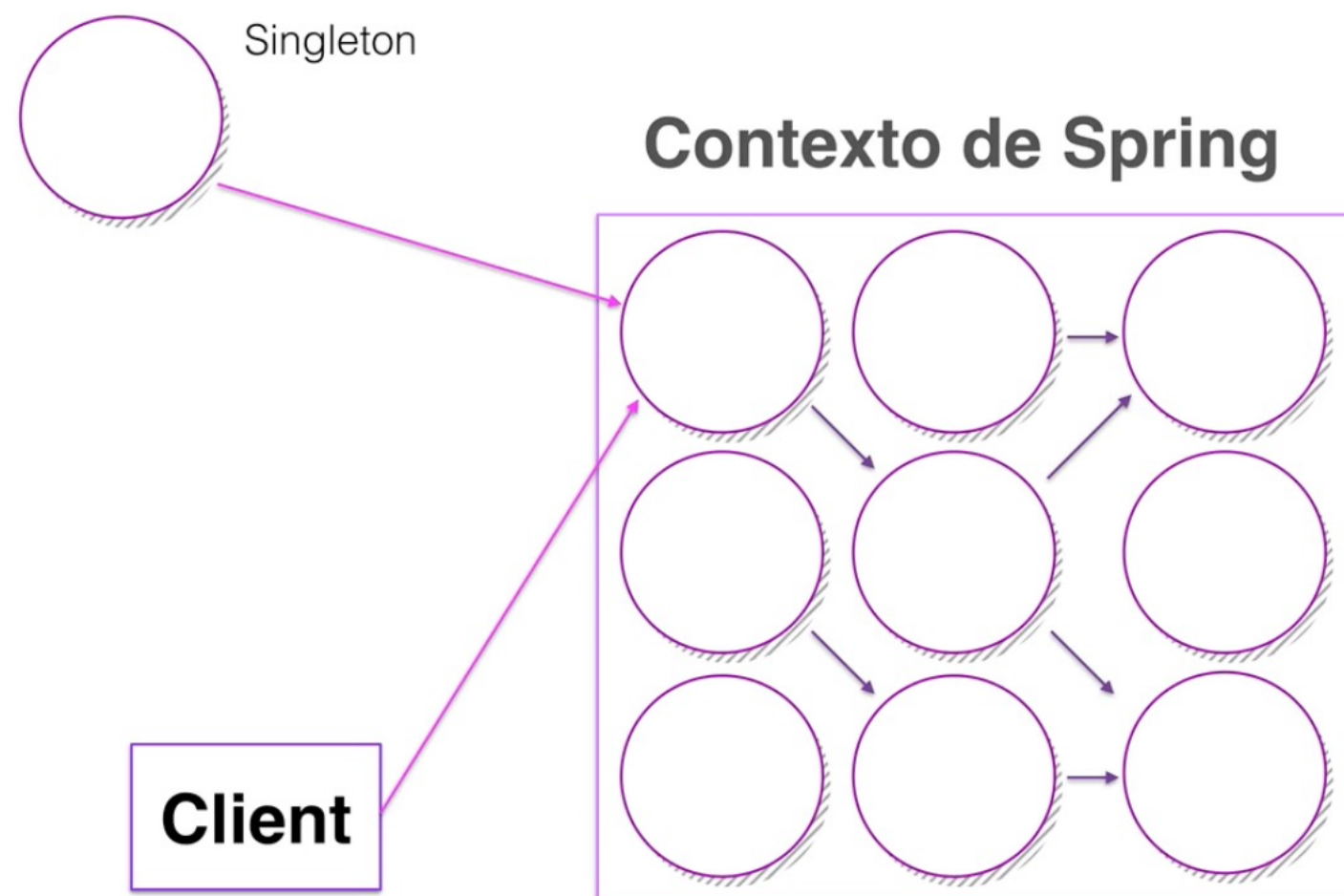
Inyección de Dependencias





@Component

Esta anotación marca una clase Java como un Bean, por lo tanto éste es tomado por el mecanismo de escaneo de componentes de Spring y lo integra al contexto de aplicación de Spring.





@Service

Esta anotación es una especialización de la anotación `Component`, no provee ningún tipo de comportamiento adicional, pero es buena practica denotar a las clases que representan a la capa de servicio ya que describe de mejor manera el intento.

@Repository

La anotación `@Repository` también es una especialización de `@Component`, pero además importa las clases de acceso a datos (DAO) dentro del contenedor de Spring, como parte de la mejora, hace que las excepciones no checadas “unchecked” lanzadas por los métodos DAO que sea trasladadas a una excepción de tipo `DataAccessException`.




@Controller

La anotación @Controller también es una especialización de @Component, pero ésta es mas usada en el contexto de Spring MVC.



Spring Initializr



Project
☐ Maven Project ☒ Gradle Project

Language
☒ Java ☐ Kotlin ☐ Groovy

Spring Boot
☐ 2.5.1 (SNAPSHOT) ☒ 2.5.0 ☐ 2.4.7 (SNAPSHOT) ☐ 2.4.6
☐ 2.3.12 (SNAPSHOT) ☐ 2.3.11

Project Metadata

Group

com.geekshirt

Artifact

geekshirt-order-service

Name

geekshirt-order-service

Description

Demo project for Spring Boot

Package name

com.geekshirt.order-service

Packaging

☒ Jar ☐ War

Java

☐ 16 ☒ 11 ☐ 8

Dependencies ADD DEPENDENCIES... ⌘ + B

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

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

Lombok DEVELOPER TOOLS
Java annotation library which helps to reduce boilerplate code.

H2 Database SQL
Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.

Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

GENERATE ⌘ + ↵

EXPLORE CTRL + SPACE

SHARE...



Thymeleaf

Thymeleaf es un motor de plantillas, es decir, **es una tecnología que nos va a permitir definir una plantilla y, conjuntamente con un modelo de datos, obtener un nuevo documento, sobre todo en entornos web.**

Ventajas de Thymeleaf

Permite realizar tareas que se conocen como natural templating. Es decir, como está basada en añadir atributos y etiquetas, sobre todo HTML, va a permitir que nuestras plantillas se puedan renderizar en local, y esa misma plantilla después utilizarla también para que sea procesada dentro del motor de plantillas. Por lo cual **las tareas de diseño y programación se pueden llevar conjuntamente.**

Es integrable con muchos de los frameworks más utilizados, como por ejemplo Spring MVC, Play, Java EE... Y está basado en el uso de nuevas etiquetas, de nuevos atributos.



Thymeleaf

Tipos de expresiones

Permite trabajar con varios tipos de expresiones:

- **Expresiones variables:** Son quizás las más utilizadas, como por ejemplo `${...}`
- **Expresiones de selección:** Son expresiones que nos permiten reducir la longitud de la expresión si prefijamos un objeto mediante una expresión variable, como por ejemplo `*{...}`
- **Expresiones de mensaje:** Que nos permiten, a partir de ficheros properties o ficheros de texto, cargar los mensajes e incluso realizar la internalización de nuestras aplicaciones, como por ejemplo `#{...}`
- **Expresiones de enlace:** Nos permiten crear URL que pueden tener parámetros o variables, como por ejemplo `@{...}`
- **Expresiones de fragmentos:** Nos van a permitir dividir nuestras plantillas en plantillas más pequeñas e ir cargándolas según las vayamos necesitando, como por ejemplo `~{...}`

Por defecto, cuando se trabaja con Thymeleaf se suele hacer con el **lenguaje de expresiones OGNL** (Object-Graph Navigation Language), aunque cuando trabajamos conjuntamente con Spring MVC podemos utilizar el SpEL (Spring Expression Language).



Thymeleaf

Tipos de expresiones

Permite trabajar con varios tipos de expresiones:

- **Expresiones variables:** Son quizás las más utilizadas, como por ejemplo `${...}`
- **Expresiones de selección:** Son expresiones que nos permiten reducir la longitud de la expresión si prefijamos un objeto mediante una expresión variable, como por ejemplo `*{...}`
- **Expresiones de mensaje:** Que nos permiten, a partir de ficheros properties o ficheros de texto, cargar los mensajes e incluso realizar la internalización de nuestras aplicaciones, como por ejemplo `#{...}`
- **Expresiones de enlace:** Nos permiten crear URL que pueden tener parámetros o variables, como por ejemplo `@{...}`
- **Expresiones de fragmentos:** Nos van a permitir dividir nuestras plantillas en plantillas más pequeñas e ir cargándolas según las vayamos necesitando, como por ejemplo `~{...}`

Por defecto, cuando se trabaja con Thymeleaf se suele hacer con el **lenguaje de expresiones OGNL** (Object-Graph Navigation Language), aunque cuando trabajamos conjuntamente con Spring MVC podemos utilizar el SpEL (Spring Expression Language).



Spring Boot – Spring Data JPA

Sesión 8



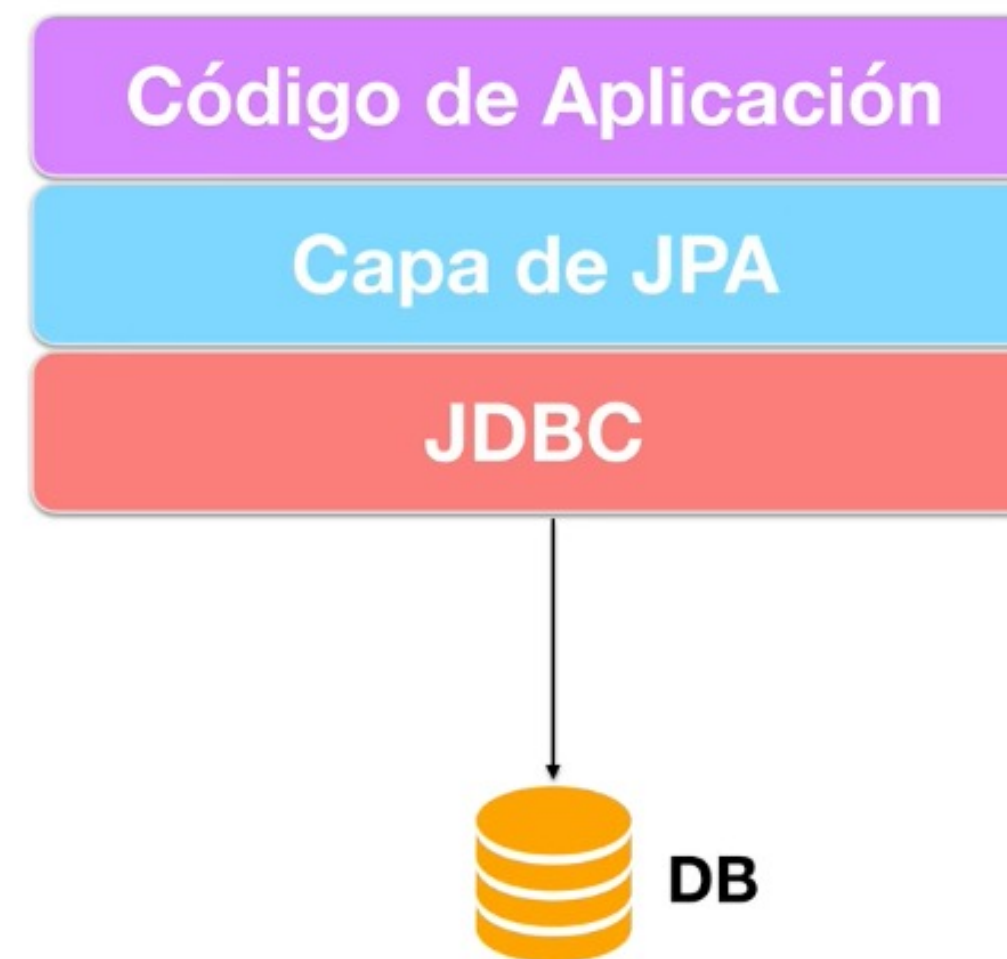
Introducción a JPA

Java Persistence API provee una abstracción fácil de usar sobre JDBC para que el código pueda ser aislado de la base de datos, definiciones de los proveedores y optimizaciones.

Provee también puede ser descrito como un motor de mapeo objeto - relacional (ORM).

Introducido en Java EE 5/ EJB 3.0

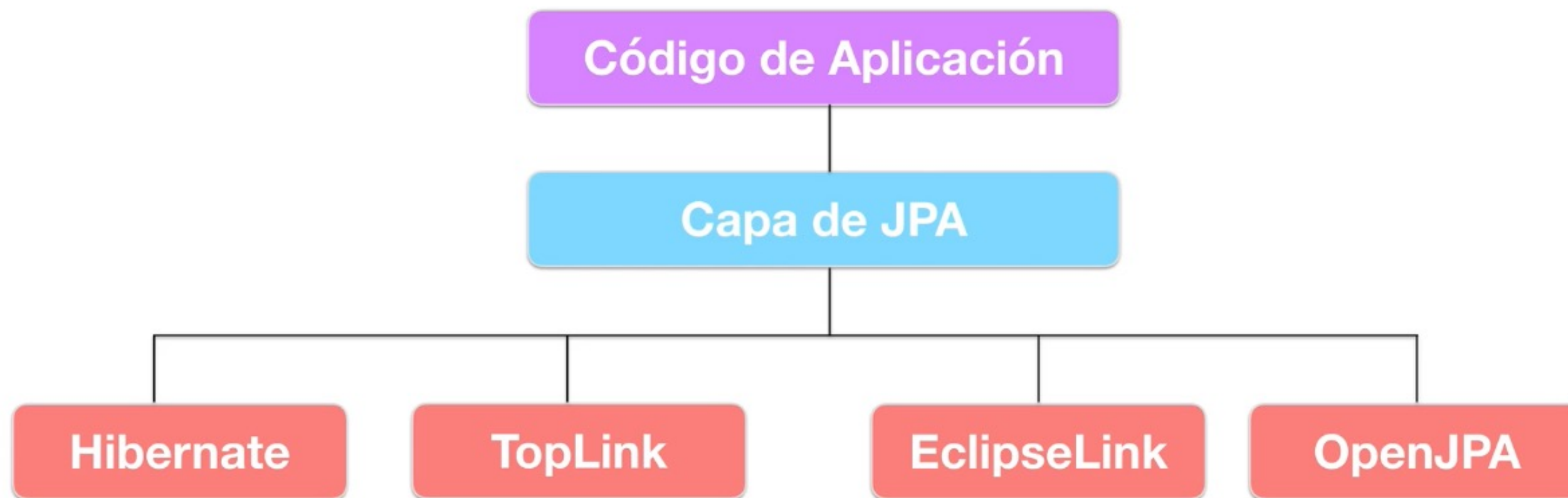
Utilizado en Java EE y Java SE.





Introducción a JPA

Provee un API, donde cualquier proveedor puede implementarla.





Mapecto Objeto – Relacional (ORM)

Modelo Orientado a Objetos	Modelo Relacional
Clase	Tabla
Objeto	Registro
Atributos	Columnas
Composici3n o Agregaci3n	Relaci3n
Metodos	Stored Procedures



Mapecto Objeto – Relacional (ORM)

```
@Entity
@Table(name = "Usuario")
class Usuario
{
    @Id
    @Column (name = "id")
    String nombreUsuario;

    @Column (nullable = false)
    String contraseña;

    @Column (name = "nombre_completo")
    String nombreCompleto;

    @Transient
    boolean logeado;
}
```

Tabla: **Usuario**

Id	contrasena	nombre_completo
rperez	XYRHSJ!98109	Roberto Perez



Mapecto Objeto – Model Base Relacional

```
@Entity
@Table(name = "Departamento")
class Departamento
{
    @Id
    @Column (name = "ID")
    Long id;

    @Column (name = "DESCRIPCION")
    String descripcion;

    @OneToMany(fetch = FetchType.LAZY, mappedBy = "departamento")
    private List<Empleado> empleados;
}
```

```
@Entity
@Table(name = "Empleado")
class Empleado
{
    @Id
    @Column (name = "ID")
    Long id;

    @Column (name = "NOMBRE")
    String nombre;

    @Column (name = "APELLIDO")
    String apellido;

    @ManyToOne
    private Departamento departamento;
}
```

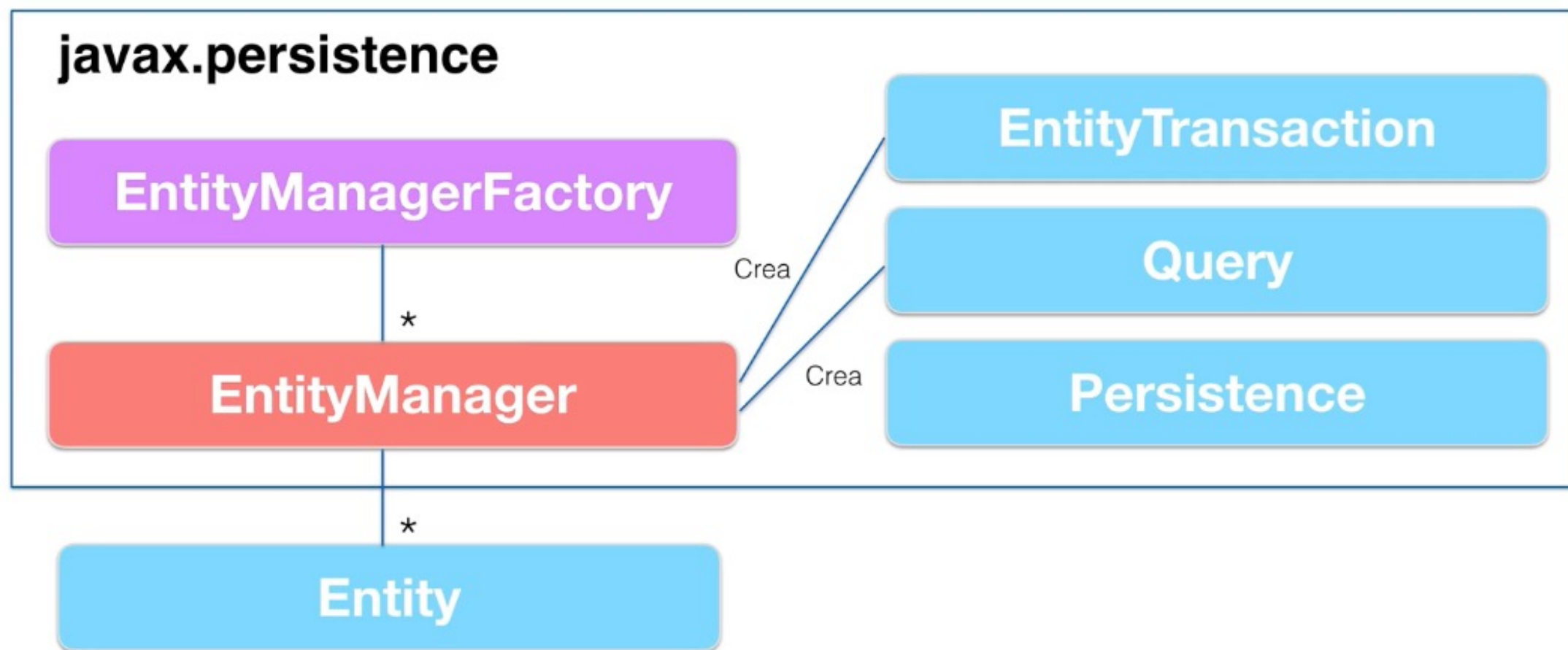
Tabla: **Empleado**

ID	NOMBRE	APELLIDO	DEPARTAMENTO_ID
rperez	Roberto	Perez	10



Entity Manager

Es el principal actor de todas las acciones de persistencia, ningún objeto puede llegar a ser persistente si no interactúa con éste componente, además se encarga de administrar el mapeo objeto relacional, otorgando un conjunto de operaciones para efectuar consultas, actualización y persistencia de objetos.





Entity Manager

Ejemplo:

```
Departamento departamento = em.find(Departamento.class, 1L);
```

```
Empleado emp = new Empleado();  
emp.setDepartamento(departamento);
```

```
departamento.getEmpleados().add(emp);
```

```
em.persist(emp);
```



Java Persistence Query Language (JPQL)

Ejemplos

```
SELECT c  
FROM Cliente AS c  
WHERE c.tieneCredito = TRUE
```

```
Select emp  
FROM Empleado as emp  
WHERE emp.id = 10
```

```
SELECT emp  
FROM Empleado as emp  
WHERE emp.direccion.estado IN ('Chiapas', 'Yucatan')
```

```
SELECT emp.direccion.estado  
FROM Empleado as emp  
WHERE emp.direccion.estado IS NOT NULL
```

```
SELECT emp  
FROM Empleado as emp  
WHERE LENGTH(emp.apellido) > 10
```

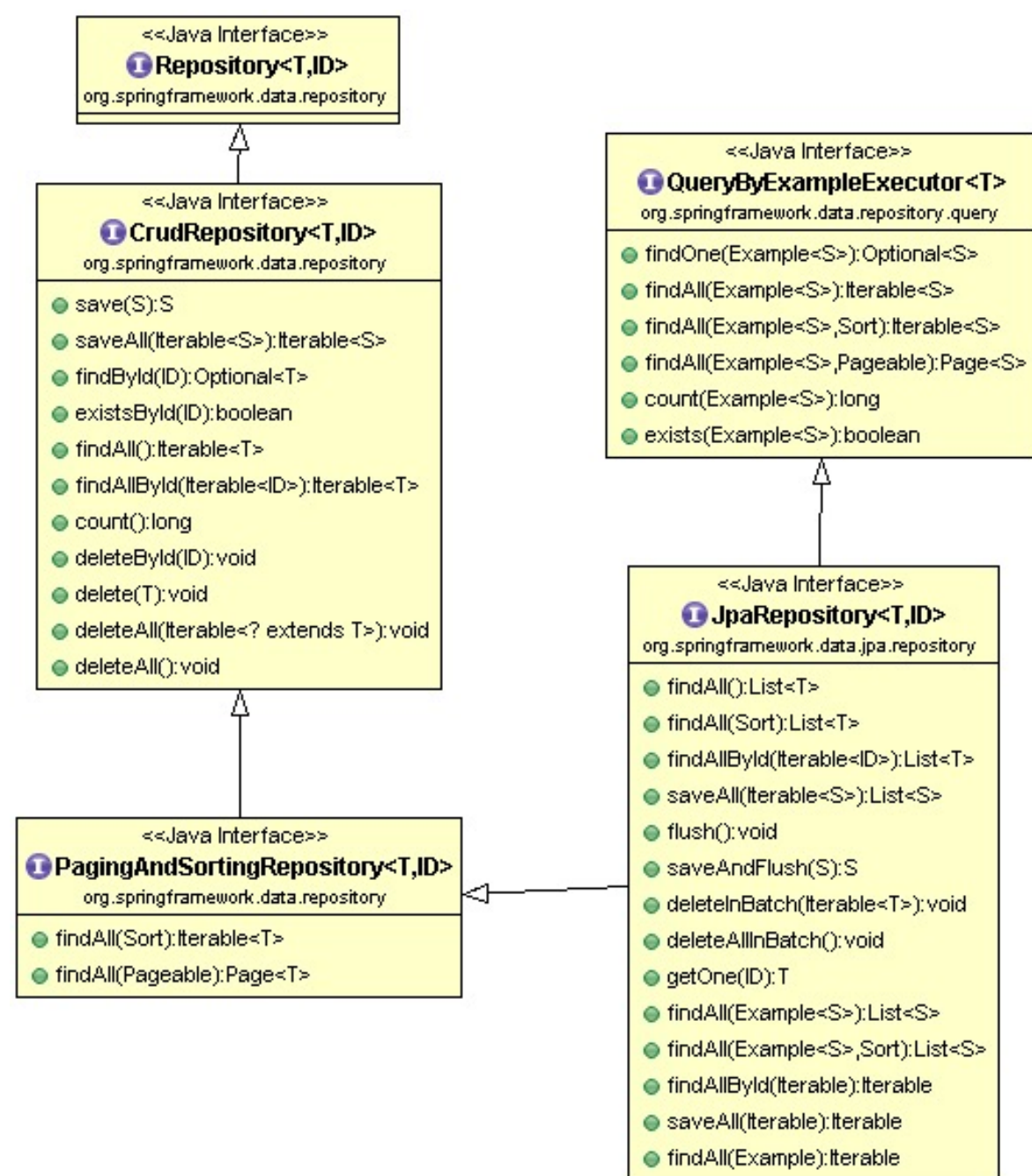


Beneficios de JPA

- Simplifica la persistencia de datos
- Mejora en el mantenimiento
- Efectúa un punto de abstracción de la base de datos
- Mejora la productividad
- Ampliamente usado por la comunidad



Spring Data JPA



Presentation layer

Service layer

Persistence layer

Database

Spring MVC
front-springmvc

Service layer
service-springdatajpa

Spring Data JPA
persistence-springdatajpa

Database



Spring Data JPA

