

Segunda parte

Persistencia de datos con Spring Data JPA



¿Qué es JPA?

- **J**ava **P**ersistence **A**PI (JPA) es una especificación de **J**ava **E**E que permite a los desarrolladores Java hacer un mapeo entre los objetos y las tablas de una base de datos (**O**bject **R**elational **M**apping) para facilitar la administración de los datos relacionales en las aplicaciones.

OBJECT

```
public class Pelicula {  
  
    private int id;  
    private String titulo;  
    private int duracion;  
    private String clasificacion;  
    private String genero;  
    private String imagen;  
    private Date fechaEstreno;  
    private String estatus;  
  
    // getters y setters  
}
```

MAPPING

RELATIONAL

Peliculas	
id	INT(11)
titulo	VARCHAR(150)
duracion	INT(11)
clasificacion	ENUM('A', 'B', 'C')
genero	VARCHAR(45)
imagen	VARCHAR(200)
fechaEstreno	DATE
estatus	ENUM(...)
Indexes	

¿Qué es Spring Data JPA? (1)

- Spring Data JPA es un módulo que forma parte del proyecto **Spring Data** y básicamente nos ayuda a simplificar el desarrollo de la persistencia de datos utilizando el concepto de repositorios (algo similar al patrón de diseño DAO **Data Access Object**).
- En términos sencillos este módulo de Spring Data agrega una capa de abstracción al API de JPA (podríamos decir es una forma más sencilla y mejorada de trabajar con JPA).

Beneficios de Spring Data JPA

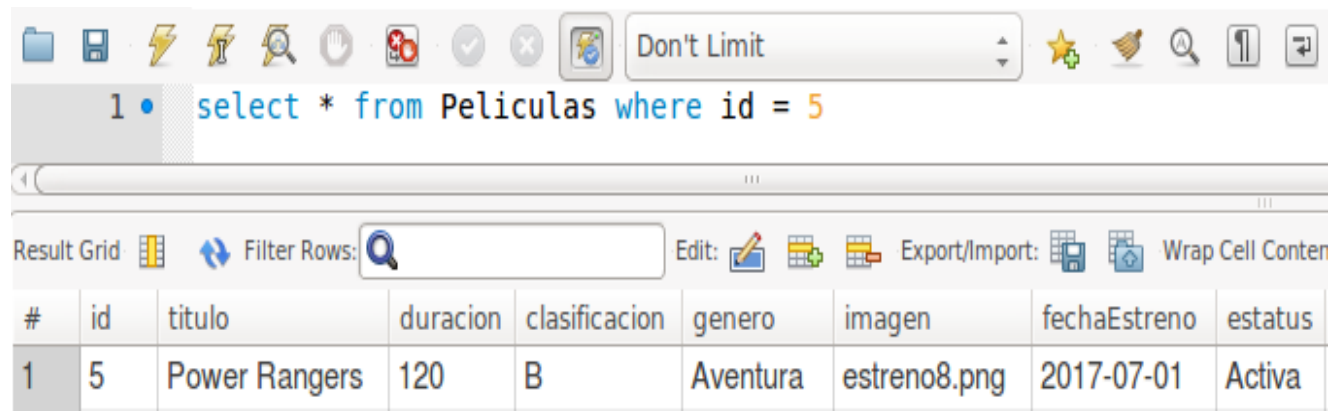
- ✓ Desarrollo ágil de la capa de persistencia de datos utilizando bases de datos relacionales.
- ✓ No es necesario escribir código SQL nativo (aunque también es posible).
- ✓ Más fácil que JDBC.
- ✓ Permite al desarrollador enfocarse más en la lógica de negocio de la aplicación y olvidarse del manejo de Excepciones (menos excepciones SQLException).
- ✓ Código más fácil de entender y mantener.

¿Qué es Spring Data JPA? (2)

Ejemplo de persistencia de datos

```
// 1. Crear objeto a guardar en la BD
Película película = new Película();
película.setId(5);
película.setTitulo("Power Rangers");
película.setDuracion(120);
película.setClasificacion("B");
película.setGenero("Aventura");
película.setImagen("estreno8.png");
película.setFechaEstreno(new Date());
película.setEstatus("Activa")

// 2. Persistir (guardar) objeto en la BD
repositoryPelículas.save(película);
```



The screenshot shows a database query tool interface. At the top, there's a toolbar with various icons. Below it, a text box contains the SQL query: `select * from Películas where id = 5`. Below the query, there's a "Result Grid" section. It includes a "Filter Rows" search bar and a table of results. The table has columns: #, id, titulo, duracion, clasificacion, genero, imagen, fechaEstreno, and estatus. The first row of data shows: 1, 5, Power Rangers, 120, B, Aventura, estreno8.png, 2017-07-01, and Activa.

#	id	titulo	duracion	clasificacion	genero	imagen	fechaEstreno	estatus
1	5	Power Rangers	120	B	Aventura	estreno8.png	2017-07-01	Activa

3. Se genera todo el código SQL de forma automática y se ejecuta en la base de datos.

Configuración de Spring Data JPA (1)

➤ Procedimiento general para configurar Spring Data JPA:

1. Descargar dependencias.
2. Configuración de los beans: dataSource, transaction manager y entity manager factory.
3. Configuración de Spring Data JPA (Repositorios).

Archivo pom.xml

```
<dependency>
  <groupId>org.springframework.data</groupId>
  <artifactId>spring-data-jpa</artifactId>
  <version>2.2.0.RELEASE</version>
</dependency>
```

1

<https://spring.io/projects/spring-data-jpa>

```
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-core</artifactId>
  <version>5.4.6.Final</version>
</dependency>
```

2

<https://hibernate.org/orm>

```
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-validator</artifactId>
  <version>6.0.17.Final</version>
</dependency>
```

3

<https://hibernate.org/validator>

```
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>5.1.48</version>
</dependency>
```

4

<https://dev.mysql.com/downloads/connector/j>

Configuración de Spring Data JPA (2)

- Agregar los namespaces spring-jpa y spring-tx (JPA y manejo de transacciones).
- Agregar las siguientes declaraciones en el archivo XML (root-context.xml) para habilitar Spring Data JPA.

```
<jpa:repositories base-package="net.itinajero.app.repository" />
```

1

```
<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
```

2

```
  <property name="driverClassName" value="com.mysql.jdbc.Driver" />
```

```
  <property name="url" value="jdbc:mysql://localhost:3306/cineapp?useSSL=false" />
```

```
  <property name="username" value="root" />
```

```
  <property name="password" value="admin" />
```

```
</bean>
```

```
<bean id="jpaVendorAdapter" class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">
```

3

```
  <property name="generateDdl" value="false" />
```

```
  <property name="showSql" value="true"></property>
```

```
  <property name="databasePlatform" value="org.hibernate.dialect.MySQL5Dialect" />
```

```
</bean>
```

```
<bean id="entityManagerFactory" class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
```

4

```
  <property name="packagesToScan" value="net.itinajero.app.model" />
```

```
  <property name="dataSource" ref="dataSource" />
```

```
  <property name="jpaVendorAdapter" ref="jpaVendorAdapter" />
```

```
</bean>
```

```
<bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager">
```

5

```
  <property name="entityManagerFactory" ref="entityManagerFactory" />
```

```
</bean>
```

Estructura de la base de datos - cineapp

