

Creación Clase Entidad Producto



Creación de la Clase de Entidad Producto

En esta guía, crearemos la clase de **entidad Producto** dentro del paquete `gm.inventarios.modelo`. Esta clase representará la tabla `producto` en la base de datos. Además, configuraremos Spring Boot para que **genere automáticamente la tabla en MySQL** al ejecutar la aplicación.

Configuración de la Clase `Producto.java`

- Paso 1: Crear la Clase `Producto.java`

En el directorio de tu proyecto, dentro del paquete `gm.inventarios.modelo`, crea el archivo `Producto.java` y escribe el siguiente código:

```
package gm.inventarios.modelo;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
```

```
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Producto {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Integer idProducto;
    String descripcion;
    Double precio;
    Integer existencia;
}
```

- Paso 2: Explicación del Código

Cada parte del código cumple una función específica:

- **@Entity**: Indica que esta clase será mapeada a una tabla en la base de datos.
 - **@Id**: Define `idProducto` como la clave primaria de la tabla.
 - **@GeneratedValue(strategy = GenerationType.IDENTITY)**: Hace que el `idProducto` sea autoincrementable en MySQL.
 - **@Data**: Genera automáticamente métodos `getter`, `setter`, `toString()`, `equals()` y `hashCode()`.
 - **@NoArgsConstructor**: Crea un constructor vacío (necesario para JPA).
 - **@AllArgsConstructor**: Genera un constructor con todos los atributos.
 - **Atributos (idProducto, descripcion, precio, existencia)**: Se convertirán en columnas en la base de datos.
-

Configuración en `application.properties`

- Paso 3: Configurar la Conexión a MySQL

Para que Spring Boot cree la base de datos y la tabla, recordemos que ya hemos configurado previamente el archivo `application.properties` en la carpeta `src/main/resources`:

```
# Configuración de la base de datos MySQL
spring.datasource.url=jdbc:mysql://localhost:3306/inventario_db?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=admin
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
# Más configuraciones...
```

¿Qué hacen estas propiedades?

- **spring.datasource.url**: Conecta con MySQL en el puerto 3306 y la base de datos inventario_db.
 - ?createDatabaseIfNotExist=true → Crea la base de datos si no existe.
 - **spring.datasource.username** y **spring.datasource.password**: Credenciales de MySQL.
 - **spring.datasource.driver-class-name**: Especifica el driver JDBC de MySQL.
 - **spring.jpa.hibernate.ddl-auto=update**: Permite que Spring Boot **cree y actualice automáticamente la tabla producto**.
 - **spring.jpa.show-sql=true**: Muestra en consola las consultas SQL generadas.
-

Ejecución del Proyecto

- Paso 4: Ejecutar la Aplicación

Para ejecutar el proyecto y que se genere la tabla en MySQL, usa uno de estos métodos:

Desde IntelliJ IDEA

1. Abre InventariosApplication.java.
 2. Haz clic en el botón Run .
-

Verificar la Creación de la Tabla en MySQL

- Paso 5: Comprobar la Tabla en MySQL

Después de ejecutar la aplicación, abre MySQL Workbench y revisa que se haya creado correctamente la tabla de Producto:

Field	Type	Null	Key	Default	Extra
id_producto	int	NO	PRI	NULL	auto_increment
descripcion	varchar(255)	YES		NULL	
precio	double	YES		NULL	
existencia	int	YES		NULL	

Si ves esta tabla, significa que **Spring Boot ha creado exitosamente la tabla producto en MySQL**.

Actualización del archivo pom.xml.

Es posible que lombok no genere correctamente el código de los métodos get/set, constructores o método toString. Para hacer una prueba y corregir cualquier problema podemos hacer lo siguiente:

Archivo pom.xml:

Agregar la versión de lombok en la sección de annotationProcessorPaths, el archivo pom.xml queda así:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.4.2</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>gm</groupId>
  <artifactId>inventarios</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>inventarios</name>
  <description>Sistema de Inventarios</description>
  <properties>
    <java.version>21</java.version>
  </properties>
  <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>
    <dependency>
      <groupId>com.mysql</groupId>
      <artifactId>mysql-connector-j</artifactId>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <optional>true</optional>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>
```

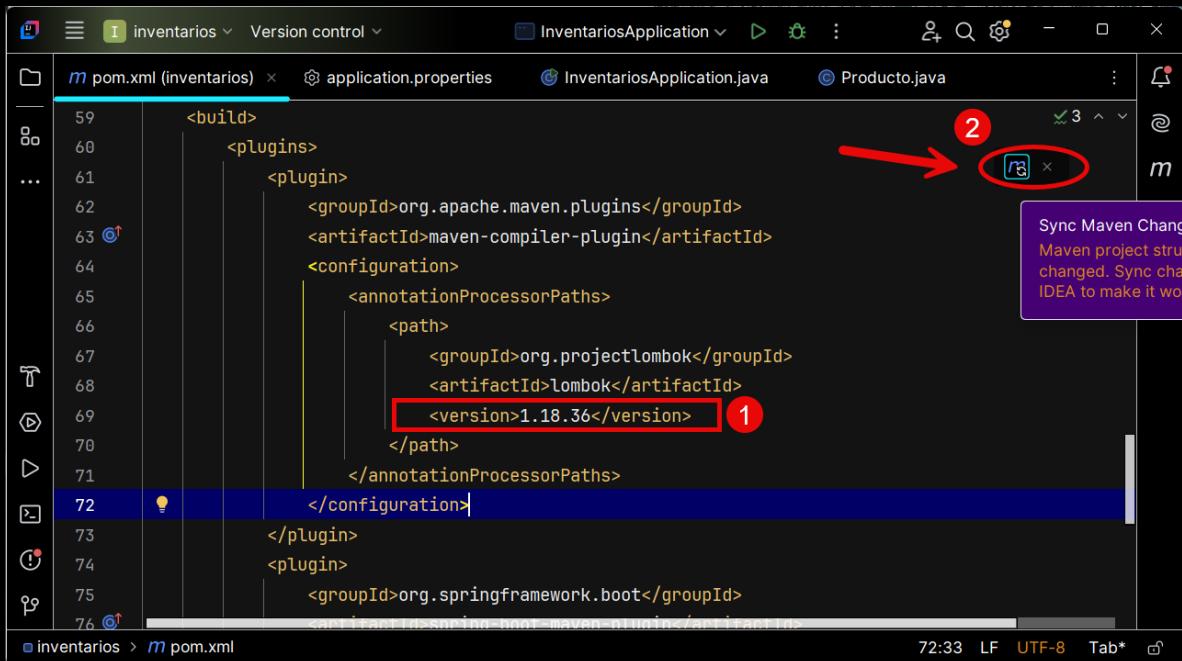
```

<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <configuration>
                <annotationProcessorPaths>
                    <path>
                        <groupId>org.projectlombok</groupId>
                        <artifactId>lombok</artifactId>
                        <version>1.18.36</version>
                    </path>
                </annotationProcessorPaths>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <configuration>
                <excludes>
                    <exclude>
                        <groupId>org.projectlombok</groupId>
                        <artifactId>lombok</artifactId>
                    </exclude>
                </excludes>
            </configuration>
        </plugin>
    </plugins>
</build>

</project>

```

Una vez agregada la versión de lombok, se debe sincronizar los cambios de archivo pom.xml como se ve en la siguiente imagen:



Prueba InventariosApplication.java

Por último, pueden hacer una prueba desde InventariosApplication.java:

```
package gm.inventarios;

import gm.inventarios.modelo.Producto;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

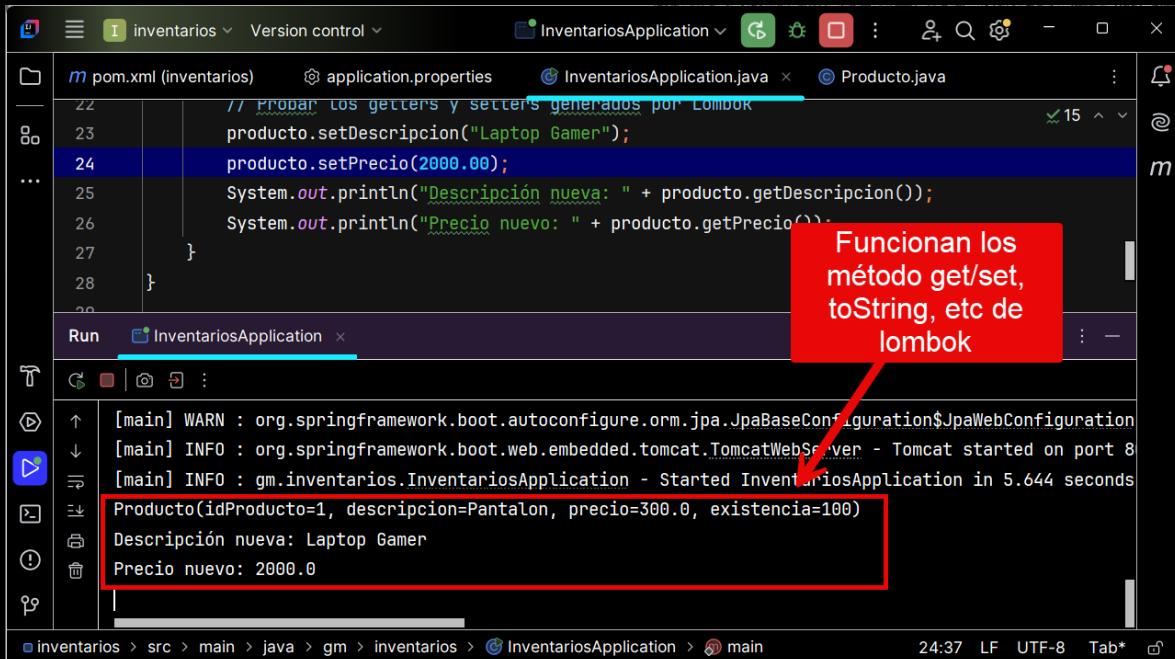
@SpringBootApplication
public class InventariosApplication {
    public static void main(String[] args) {
        SpringApplication.run(InventariosApplication.class, args);

        // Prueba de Lombok: Crear un objeto Producto y usar toString()
        Producto producto = new Producto();
        producto.setIdProducto(1);
        producto.setDescripcion("Pantalon");
        producto.setPrecio(300.0);
        producto.setExistencia(100);

        // Imprimir el objeto usando toString() de Lombok
        System.out.println(producto);

        // Probar los getters y setters generados por Lombok
        producto.setDescripcion("Laptop Gamer");
        producto.setPrecio(2000.00);
        System.out.println("Descripción nueva: " + producto.getDescripcion());
        System.out.println("Precio nuevo: " + producto.getPrecio());
    }
}
```

Ejecutar InventariosApplication.java:



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

- Project Structure:** The project is named "inventarios". The file "InventariosApplication.java" is the active file.
- Code Editor:** The code is written in Java, utilizing Lombok annotations. A tooltip from the IDE indicates: "Funcionan los método get/set, toString, etc de lombok".
- Run Tab:** The "Run" tab is selected, showing the configuration for "InventariosApplication".
- Output Window:** The output window displays the application's logs and the printed results. It shows the application starting, the updated product details, and the new price.

```
[main] WARN : org.springframework.boot.autoconfigure.orm.jpa.JpaBaseConfiguration$JpaWebConfiguration
[main] INFO : org.springframework.boot.web.embedded.tomcat.TomcatWebServer - Tomcat started on port 8
[main] INFO : gm.inventarios.InventariosApplication - Started InventariosApplication in 5.644 seconds
Producto(idProducto=1, descripcion=Pantalon, precio=300.0, existencia=100)
Descripción nueva: Laptop Gamer
Precio nuevo: 2000.0
```

Conclusión

- Creamos la clase `Producto.java` con JPA y Lombok.
- Configuramos `application.properties` para conectar Spring Boot con MySQL.
- Ejecutamos la aplicación y verificamos que la tabla `producto` se creó automáticamente en la base de datos.

Con esto, ya tenemos nuestra entidad `Producto` lista para usar en el sistema de inventarios.  

Saludos!

Ing. Ubaldo Acosta

Fundador de [GlobalMentoring.com.mx](#)