

Configuración Inicial Spring Boot



Guía de Configuración Inicial del Proyecto Spring Boot - Sistema de Inventarios

En esta guía configuraremos nuestro **Sistema de Inventarios** en Spring Boot con conexión a MySQL y ajustes esenciales para el inicio de la aplicación.

1 Configuración del archivo `application.properties`

El archivo `application.properties` define la configuración global de nuestra aplicación.

```
# Conexion a mysql
spring.datasource.url=jdbc:mysql://localhost:3306/inventario_db?createDatabaseIfNotExists=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
# Configurar el patrón del log (en lugar de logback-spring.xml)
logging.pattern.console=[%thread] %-5level: %logger - %msg%n

# Configurar el nivel de log (INFO en este caso)
logging.level.root=INFO
#Opcional, configurar niveles por paquetes
#logging.level.com.miapp=DEBUG
#logging.level.org.springframework=WARN
```

```
#server.port=8081  
spring.main.banner-mode=off
```

A continuación, analizamos las propiedades agregadas:

📌 Conexión a MySQL

```
# Conexion a mysql  
spring.datasource.url=jdbc:mysql://localhost:3306/inventario_db?createDatabaseIfN  
otExist=true  
spring.datasource.username=root  
spring.datasource.password=admin  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

◆ Explicación:

- spring.datasource.url:** Define la **URL de conexión a MySQL**.
- localhost:3306: Conexión local en el puerto 3306.
 - inventario_db: Nombre de la base de datos.
 - createDatabaseIfNotExist=true: **Si la BD no existe, la crea automáticamente.**
- spring.datasource.username** y **spring.datasource.password:** Usuario y contraseña de la base de datos.
- **⚠ Importante:** Si MySQL tiene otro usuario o clave, actualiza estos valores.
- spring.datasource.driver-class-name:** Especifica el **driver JDBC** para MySQL.

📌 Configuración de Hibernate y JPA

```
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.show-sql=true
```

◆ Explicación:

- spring.jpa.hibernate.ddl-auto=update:** **Spring Boot actualizará automáticamente la estructura de la base de datos** según las entidades (@Entity).
- Valores posibles:
 - none: No realiza cambios en la BD.
 - create: Crea la BD desde cero en cada inicio (**⚠ borra los datos**).
 - update: Solo actualiza la estructura **sin borrar datos** (recomendado en desarrollo).
 - validate: Verifica la estructura, pero **no la modifica**.

- `spring.jpa.show-sql=true`: **Muestra en la consola todas las consultas SQL ejecutadas**, útil para depuración.
-

📌 Configuración de Logs (Sin `logback-spring.xml`)

```
# Configurar el patrón del log (en lugar de logback-spring.xml)
logging.pattern.console=[%thread] %-5level: %logger - %msg%n

# Configurar el nivel de log (INFO en este caso)
logging.level.root=INFO
```

◆ Explicación:

- `logging.pattern.console`: Define el **formato de los logs en consola**.

- `[thread]`: Hilo de ejecución.
- `%-5level`: Nivel de log (INFO, DEBUG, ERROR).
- `%logger`: Nombre del logger.
- `%msg%n`: Mensaje del log.

- `logging.level.root=INFO`: Establece el **nivel de logs global**.

- Puedes configurarlo por paquetes específicos:

```
logging.level.com.miapp=DEBUG
logging.level.org.springframework=WARN
```

- Esto haría que **com.miapp** muestre logs en DEBUG, pero **Spring** solo en WARN.
-

📌 Configuración del Servidor

```
#server.port=8081
spring.main.banner-mode=off
```

◆ Explicación:

- `server.port=8081`: **Cambia el puerto donde se ejecuta la aplicación** (por defecto es 8080). Si esta línea está comentada (#), usará 8080.

- `spring.main.banner-mode=off`: **Desactiva el banner de Spring Boot** al iniciar la aplicación (opcional, solo para una consola más limpia).
-

2 Ejecutar la Aplicación desde `InventariosApplication.java`

Nuestro punto de entrada en Spring Boot es la clase `InventariosApplication.java`, ubicada en el paquete `gm.inventarios`.

📌 Código de `InventariosApplication.java`

```
package gm.inventarios;

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);
    }
}
```

◆ Explicación:

- `@SpringBootApplication`: Es la anotación principal de Spring Boot, que activa **autoconfiguración, escaneo de componentes y configuración de Spring**.
- `SpringApplication.run(InventariosApplication.class, args)`: **Inicia la aplicación Spring Boot.**

3 Cómo Ejecutar la Aplicación

Para iniciar la aplicación, abre la terminal en el directorio del proyecto y ejecuta:

◆ Usando IntelliJ IDEA

- Abre `InventariosApplication.java` y haz clic en **Run ➔**.
 - Verás en la consola que el servidor está corriendo en `http://localhost:8080/` (o `8081` si lo cambiaste).
-

📌 Conclusión

1 Configuramos MySQL, Hibernate y Logs en `application.properties`.

2 Entendimos para qué sirve cada propiedad.

3 Ejecutamos la aplicación desde `InventariosApplication.java`.

Con esto, ya tenemos la configuración inicial de nuestro **backend en Spring Boot** listo para empezar a desarrollar  . Así que comencemos 😊

Saludos!

Ing. Ubaldo Acosta

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