Introducción a Spring Boot

Complement - Introduccio a Spring ...
PDF

Explicame con ejemplos como va Spring Boot

¡Perfecto! Te voy a explicar **cómo funciona Spring Boot paso a paso**, con ejemplos sencillos, para que lo entiendas desde cero.

✅ 1. ¿Qué es Spring Boot?

Spring Boot es un framework de Java para crear aplicaciones (normalmente web o APIs) **rápido y sin mucha configuración**.

Antes (con Spring normal):
❌ Mucha configuración
❌ Muchos archivos XML
❌ Difícil empezar

Con Spring Boot:
✅ Todo viene preparado
✅ Servidor incluido
✅ Solo programas

👉 Piensa en Spring Boot como:

> "Spring + todo configurado automáticamente"

✅ 2. ¿Qué puedo hacer con Spring Boot?

Con Spring Boot puedes crear:

✔️ APIs REST (lo más común)
✔️ Páginas web
✔️ Aplicaciones con base de datos
✔️ Microservicios

Lo más típico:

> 📌 Crear una **API REST** para que otras apps se conecten.

✅ 3. Estructura básica de un proyecto

Cuando creas un proyecto Spring Boot tienes algo así:

```
css

miProyecto
 └── src/main/java
     └── com.ejemplo.demo
         ├── DemoApplication.java ← MAIN
         └── controller
             └── HolaController.java
```

Dos partes clave:

Parte	Para qué sirve
Main	Arranca la app
Controller	Atiende peticiones

✅ 4. Clase principal (Main)

Esta clase **inicia todo**:

```java
java

@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

}
```

¿Qué hace esto?

Cuando ejecutas:

```java
SpringApplication.run(...)
```

Spring:

✔️ Arranca Tomcat
✔️ Carga tus clases
✔️ Prepara rutas
✔️ Abre el puerto 8080

👉 Es como darle al botón "ON".

✅ 5. Crear un Controller (lo más importante)

Un **controller** responde a las peticiones del navegador o apps.

Ejemplo simple:

```java
@RestController
public class HolaController {

    @GetMapping("/hola")
    public String hola() {
        return "Hola mundo";
    }

}
```

📌 ¿Qué significa esto?

Parte	Significado
@RestController	Es un controlador

Parte	Significado
@GetMapping	Atiende GET
"/hola"	Ruta
return	Respuesta

👉 Si entras en el navegador:

```bash
http://localhost:8080/hola
```

Verás:

```nginx
Hola mundo
```

🎉 Ya tienes una API funcionando.

✅ 6. Rutas con parámetros

Puedes enviar datos en la URL.

Ejemplo:

```java
@GetMapping("/saludar")
public String saludar(@RequestParam String nombre) {
    return "Hola " + nombre;
}
```

👉 En el navegador:

```bash
```

```
http://localhost:8080/saludar?nombre=Juan
```

Respuesta:

```nginx
Hola Juan
```

✅ 7. Usar varios métodos

Puedes tener muchas rutas:

```java
@RestController
public class HolaController {

    @GetMapping("/hola")
    public String hola() {
        return "Hola";
    }

    @GetMapping("/adios")
    public String adios() {
        return "Adiós";
    }

}
```

✅ 8. Peticiones POST (enviar datos)

GET = pedir datos

POST = enviar datos

Ejemplo POST:

```java
```

```java
@PostMapping("/guardar")
public String guardar(@RequestBody String datos) {
    return "Recibido: " + datos;
}
```

👉 Desde Postman / frontend:

Body:

```json
Hola servidor
```

Respuesta:

```makefile
Recibido: Hola servidor
```

✅ 9. Devolver JSON (muy importante)

Normalmente devolvemos objetos → JSON.

Modelo:

```java
public class Persona {

    private String nombre;
    private int edad;

    // getters y setters
}
```

Controller:

```java
```

```java
@GetMapping("/persona")
public Persona persona() {

    Persona p = new Persona();
    p.setNombre("Ana");
    p.setEdad(20);

    return p;
}
```

Respuesta:

```json
{
  "nombre": "Ana",
  "edad": 20
}
```

👉 Spring convierte automáticamente a JSON.

✅ 10. Códigos HTTP (ResponseEntity)

Para controlar respuestas:

```java
@GetMapping("/ok")
public ResponseEntity<String> ok() {
    return ResponseEntity.ok("Todo bien");
}
```

Ejemplo error:

```java
@GetMapping("/error")
public ResponseEntity<String> error() {
```

```
    return ResponseEntity.status(404).body("No existe");
}
```

✅ 11. Dependencias (pom.xml)

Ejemplo básico:

```xml
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

Esto incluye:

✔️ Tomcat
✔️ Controllers
✔️ JSON
✔️ REST

Sin esto → no hay web.

✅ 12. Flujo completo (cómo funciona todo)

Cuando alguien entra a tu API:

```markdown
Navegador/App
    ↓
Controller
    ↓
Método Java
    ↓
Respuesta
```

Ejemplo real:

```bash
```

```
Cliente → /persona
      → Controller
      → crea Persona
      → devuelve JSON
```

✅ 13. Ejemplo completo sencillo

Proyecto mínimo:

Main

java

```java
@SpringBootApplication
public class App {

    public static void main(String[] args) {
        SpringApplication.run(App.class, args);
    }

}
```

Controller

java

```java
@RestController
public class ApiController {

    @GetMapping("/hola")
    public String hola() {
        return "Hola";
    }

    @GetMapping("/suma")
    public int suma(@RequestParam int a,
                    @RequestParam int b) {
        return a + b;
    }
```

```
}
```

Uso:

```bash

/hola → Hola
/suma?a=5&b=3 → 8
```

✅ 14. Resumen mental (muy importante)

Memoriza esto:

🧠 Spring Boot =

✅ Proyecto Java
✅ Clase Main
✅ Controllers
✅ Rutas
✅ JSON
✅ Servidor integrado

🧠 Siempre hay:

```less

@SpringBootApplication → arranca
@RestController → recibe
@GetMapping/PostMapping → rutas
```

🚀 ¿Qué puedo enseñarte después?

Si quieres, podemos seguir con:

👉 Base de datos (MySQL + JPA)
👉 Proyecto real paso a paso
👉 Login usuarios
👉 CRUD completo
👉 Spring Security

Dime:

👉 ¿Lo estás usando en clase?
👉 ¿Quieres que hagamos un proyecto desde cero juntos?

la forma correcta de importar springboot es la siguiente?

```
<?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 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>T5-SpringBoot</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>

<maven.compiler.source>23</maven.compiler.source>

<maven.compiler.target>23</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <dependency>
```