Fundamentos de Spring Boot

Qué hace realmente Spring Boot por ti y cómo crear tu primer endpoint REST

- Objetivos del módulo
  - Comprender **qué es Spring Boot** y por qué existe.
  - Entender la **estructura mínima** de un proyecto.
  - Crear y ejecutar tu **primer endpoint REST funcional**.

1. ¿Por qué Spring Boot?

Spring Boot es el marco que te evita el infierno de configuraciones XML del Spring clásico.
Se encarga de tres cosas fundamentales:

Área	*Qué hace*	*Ejemplo*
Auto-configuración	Detecta tus dependencias y configura todo automáticamente.	Si añades spring-boot-starter-web, ya tienes Tomcat, JSON y MVC listos.
Servidor embebido	Levanta Tomcat o Jetty dentro del JAR.	Ejecutas mvn spring-boot:run y ya tienes tu API en 8080.
Convención > Configuración	Usa valores por defecto sensatos.	Archivos application.properties o application.yml para overrides.

2. Estructura básica de un proyecto Spring boot

```
src/
├─ main/
│   ├─ java/com/payoyo/journey/hello/
│   │   ├─ Application.java
│   │   └─ HelloController.java
│   └─ resources/
│       └─ application.properties
└─ test/
    └─ java/.../ApplicationTests.java
```

- Application.java → punto de entrada
- HelloController.java → donde defines tus rutas
- application.properties → configuración básica (puerto, versión, etc.)

3. Tu primer endpoint REST

```java
package com.payoyo.curso.spring.boot.module_01_fundamentos;

import java.util.Map;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/api")
public class HelloController {

    @GetMapping("/hello")
    public Map<String, String> hello() {
        return Map.of("message", "Hola Mundo desde spring boot");
    }

}
```

- Ejecuta: mvn spring-boot:run
- Y abre el navegador en: http://localhost:8080/api/hello

4. Entendiendo la magia

4.1. *@SpringBootApplication* combina 3 notaciones

- **@Configuration** -> Esta clase define beans (componentes) de configuración para el contenedor de Spring. Con eso, Spring entiende: Esta clase tiene métodos que crean objetos gestionados (beans) por el contenedor.

- **@EnableAutoConfiguration ->** Activa la configuración automática de Spring Boot basada en las dependencias del classpath, le dice a Spring Boot: Busca configuraciones automáticas dentro de spring.factories y actívalas según el contexto
- **@ComponentScan ->** Busca clases con anotaciones de Spring en el paquete actual y subpaquetes. le dice: Escanea este paquete y sus subpaquetes para registrar todos los componentes