Spring vs Spring Boot

🌱Spring Framework

Definition:
A comprehensive framework for building Java enterprise applications.

Key Features: - Modular architecture (Core, AOP, MVC, Security, etc.) - Dependency Injection (IoC) - Supports Aspect-Oriented Programming - Allows fine-grained configuration (XML, Annotations, Java-based)

Drawbacks: - Requires a lot of boilerplate code - Complex configuration for simple apps - Developer has to set up servers and dependency management manually

Example: To create a web app, you must: - Configure web.xml - Define DispatcherServlet - Add dependencies manually in pom.xml

---

🌳Spring Boot

Definition:
A rapid application development tool built on top of Spring. It simplifies building production-ready apps.

Key Features: - Auto-configuration (based on classpath) - Embedded servers (Tomcat, Jetty, etc.) - No need for XML configuration - Production-ready features (metrics, health checks via Actuator) - Starter dependencies (e.g., `spring-boot-starter-web`, `spring-boot-starter-data-jpa`)

Benefits: - Fast setup and development - Minimal configuration required - Microservices-friendly

Example: Creating a web app: - Add `spring-boot-starter-web` in pom.xml - Write a `@RestController` - Run main class with `@SpringBootApplication`

🐒Summary Table

Feature	Spring Framework	Spring Boot
Configuration	Manual	Auto-configured
Server Setup	Manual (external)	Embedded server (Tomcat, Jetty)
Project Setup	Complex	Quick with Spring Initializr
XML Configuration	Common	Avoided (Annotation-based)
Suitable for	Large enterprise apps	Microservices & quick dev

Feature	Spring Framework	Spring Boot
Dependency Management	Manual	Starter POMs provided
Learning Curve	Steeper	Easier