Spring Boot Basics

"An Introduction to Rapid Application Development with Spring Boot"

Presenter Name: Rama Shanker

Date: 28/01/2025

Agenda

What is Spring Boot?

Features of Spring Boot

Spring Boot vs. Spring Framework

Core Components

How to Create a Spring Boot Application

Demo (Optional)

Q&A

What is Spring Boot?

A framework for building stand-alone, production-ready Spring applications.

Simplifies Spring Framework setup with embedded servers, default configurations, and reduced boilerplate.

Suitable for Microservices architecture.

Features of Spring Boot

Auto-Configuration: Automatically configures Spring Beans based on application needs.

Embedded Server: Comes with embedded Tomcat, Jetty, or Undertow.

Opinionated Defaults: Provides sensible defaults to get started quickly.

Production-Ready: Includes monitoring and health checks via Actuator.

Microservice Support: Ideal for creating lightweight, distributed services.

Spring Boot vs. Spring Framework

Feature	Spring Framework	Spring Boot
Configuration	Requires XML or Java Config	Auto-configured, minimal setup
Embedded Server	Requires external setup	Comes with an embedded server
Setup Time	Time-consuming	Rapid development
Production Tools	Limited	Built-in tools like Actuator

Core Components of Spring Boot

Starter Dependencies: Simplifies Maven/Gradle dependencies (e.g., spring-boot-starter-web).

Spring Boot CLI: Command-line tool for running Spring Boot applications.

Embedded Servers: Run applications without external server dependencies.

Spring Boot Actuator: Monitor and manage applications in production.

Spring Boot Initializer: Web-based tool to generate Spring Boot projects (https://start.spring.io)

Spring Boot Architecture

Spring Boot Application Layer:

Spring Boot Starter: Predefined dependencies for simplifying configuration (e.g., spring-boot-starter-web, spring-boot-starter-data-jpa).

Spring Boot CLI: Optional tool for running Groovy scripts and rapid development.

Controller Layer:

Handles HTTP requests and defines RESTful endpoints.

Controllers use annotations like @RestController, @RequestMapping, and @GetMapping.

Service Layer:

Contains the business logic of the application.

Uses @Service annotation.

Repository/Data Access Layer:

Interacts with the database using JPA, Hibernate, or other data persistence APIs.

Uses @Repository annotation.

Includes Spring Data JPA and CRUD Repository abstractions.

Model Layer:

Contains entity classes that represent the applicationâ€™s data structure.

Annotations like @Entity, @Table, and @Id are used for mapping.

Configuration Layer:

Auto-configuration provided by Spring Boot to reduce boilerplate code.

Includes application.properties or application.yml for external configurations.

Embedded Server:

Default servers like Tomcat, Jetty, or Undertow for running the application.

Dependency Injection (DI) Container:

Manages beans and their lifecycle.

Uses @Component, @Bean, @Autowired, and other annotations for dependency injection.

Spring Boot Actuator:

Provides production-ready features like health checks, metrics, and monitoring.

Spring Security (Optional):

Handles authentication, authorization, and security.

How to Create a Spring Boot Application

Using Spring Initializr

Go to https://start.spring.io.

Choose dependencies (e.g., Web, JPA, H2).

Generate and download the project.

Code Example:

```
@RestController
public class HelloController {
    @GetMapping("/hello")
    public String sayHello() {
        return "Hello, Spring Boot!";
    }
}
```

Run the Application:

Use mvn spring-boot:run or run the main() method.

Demo

Build a simple "Hello World" Spring Boot application.

Add a REST endpoint and run the application.

Summary

- Spring Boot simplifies application development with less configuration.
- Key features include auto-configuration, embedded servers, and production-ready tools.
- Perfect for microservices architecture.