Admissions & Support:
support@mockattempt.com
Phone / WhatsApp: +91 95400 07917
Course Code: SPBT • **Updated:** 05 Nov 2026
https://mockattempt.com/courses/springboot-training

SPBT • JAVA ENTERPRISE BACKEND & MICROSERVICES ENGINEERING | **BEGINNER TO PRINCIPAL SPRING BOOT 3 & JAVA 21 ARCHITECT**

100% LIVE INDUSTRY EXPERT COHORT + INTERNSHIP LETTER

Springboot Training

Master Spring Boot 3 from IoC/DI, Starters, Auto-Configuration & DispatcherServlet REST APIs to ProblemDetail Exception Handling, Jakarta Validation, Spring AOP, HikariCP + Spring Data JPA (solving N+1), Caching, Spring Security 6 JWT & Actuator — with runnable Java 21 programs, 386 Topic Mock Tests (17,860 MCQs), and Spoken Voice + MockAttempt AI Interviews at ₹3,299 (Incl. 18% GST).

Duration: 16–18 Weeks • Live + Hands-On Projects (208 Hrs) • **Modules:** 16 • **Live Topics:** 53 • **Subtopics:** 424+ • **Labs & Tasks:** 106 • **Mocks / AI Vivas:** 386 / 386 • **Projects:** 16 Module + 1 Live Client Prod

All-Inclusive Direct Fee (Incl. 18% GST • Zero Hidden Charges): Total Payable: ₹3,299 (Net Tuition ₹2,796 + Included 18% GST ₹503) • **Split Option:** 100% Credit to Job Track • **Effective ROI:** ~₹16/Guided Hr (~₹62/Live Topic)

Faculty: 100% Active Working IT Professionals • **Live Batch Tracks (7:00 AM – 8:00 PM IST Flexible Slots):** Weekday Live Online Batch (Mon – Fri • 1 Hr/Day • 7:00 AM – 8:00 PM) (Monday to Friday (1 Hour Live Session / Day) • 07:00 AM – 08:00 PM IST (Students Can Choose Any 1-Hour Slot Based on Trainer Availability) • 17–19 Weeks Fast-Track) | Weekend Live Online Batch (Sat – Sun • 2 Hrs/Day • 7:00 AM – 8:00 PM) (Saturday & Sunday (2 Hours Live Session / Day) • 07:00 AM – 08:00 PM IST (Students Can Choose Any 2-Hour Slot Based on Trainer Availability) • 19–21 Weeks Weekend Track)

✓ **Trainer LeetCode & HackerRank Mentorship (Java 21 LTS): Live trainer support to crack LeetCode & HackerRank problems, optimize complexity, and handle edge cases.**

Core Technologies & Engineering Stack: Spring Boot 3.4 • Java 21 Virtual Threads • Spring Web MVC & REST • Spring Data JPA & Hibernate 6 • Spring AOP & Caching • Spring Security 6 & JWT • Actuator, OpenAPI 3 & JUnit 5

1. PROGRAM MASTERY OUTCOMES & ELIGIBILITY

End-to-End Engineering Outcomes:

1. Understand Spring Boot Auto-Configuration (@ConditionalOnClass, @ConditionalOnMissingBean), Constructor DI, and Embedded Tomcat with Java 21 Virtual Threads
2. Engineer production RESTful APIs with @RestController, ResponseEntity, @Valid Jakarta Validation, RFC-9457 ProblemDetail @RestControllerAdvice, and OpenAPI 3 Swagger UI
3. Eliminate Hibernate N+1 query bottlenecks using FetchType.LAZY + @EntityGraph, guard @Transactional CGLIB proxy boundaries, and accelerate reads with @Cacheable
4. Clear 386 Timed Mock Exams (17,500 Spring Boot MCQs) and 350 Spoken Voice + MockAttempt AI Java Backend Technical Interviews
5. Clean Spring Boot 3.3 architecture with constructor injection, immutable DTO records, and Bean Validation (`@Valid`).
6. Crack LeetCode and HackerRank problems in Java 21 LTS with live trainer mentorship on algorithmic/query patterns, edge cases, and optimal complexity.
7. Spring Security 6 stateless OAuth2/JWT filter chains, CORS configuration, and method-level RBAC authorization.
8. Hibernate / Spring Data JPA performance: solving N+1 queries, optimistic locking, projections, and Flyway migrations.

Eligibility & Prerequisites: Basic Core Java OOP & Collections familiarity — covers IoC, Maven Starters, and Spring Boot 3 fundamentals step-by-step from scratch • Any laptop/PC with JDK 21, Maven/Gradle, and IntelliJ IDEA Community / STS / VS Code • No prior Springboot industry experience required — Module 1 starts from environment setup, core mental models, and plain-English fundamentals. • Ideal for B.Tech / BCA / MCA / B.Sc IT students, non-CS beginners, career switchers, and working developers upgrading to 2026 production standards. • Commitment to attend live Weekday or Weekend classes, complete daily hands-on lab checklists, and practice topic-wise Mock Tests & AI Interviews.

2. DETAILED ZERO-TO-EXPERT MODULE, TOPIC & SUBTOPIC SYLLABUS (16 MODULES • 53 LIVE TOPICS • 424+ SUBTOPICS)

Progressive Zero-to-Expert Mastery Stages: Modules 1–4: Stage 1: Day-1 Foundations & Core Mental Models (16 Topics / 128 Subtopics) • Modules 5–8: Stage 2: Applied Engineering, Data & API Workflows (14 Topics / 112 Subtopics) • Modules 9–12: Stage 3: Internals, Concurrency, Security & Performance (11

Topics / 88 Subtopics) • Modules 13–16: Stage 4: Production Architecture, Testing, CI/CD & Expert Mastery (12 Topics / 96 Subtopics)

MODULE 1 OF 16 • STAGE 1 • 12 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 1: Spring Boot 3 Architecture, IoC/DI Foundations & Spring vs Spring Boot vs Spring MVC

Understand Inversion of Control (IoC), why Spring Boot eliminates boilerplate XML/WAR ceremony over classic Spring Framework, how the 4-layer enterprise architecture flows, and Java 21 Virtual Threads.

Objectives: ✓ Spring Boot 3 Overview — Opinionated Defaults, Embedded Servers & Zero-XML Philosophy — How Spring Boot wraps the Spring IoC Container (`ApplicationContext`) with opinionated Starter dependencies, Conditional Auto-Configuration, an embedded Tomcat/Jetty/Undertow server, and production-ready Actuator metrics. • ✓ Spring Framework vs Spring MVC vs Spring Boot — IoC Container, `DispatcherServlet` & Auto-Setup — Clear architectural boundaries: **Spring Core** provides IoC & Dependency Injection; **Spring MVC** is the web module (`DispatcherServlet`, `@Controller`); **Spring Boot** auto-wires both with curated BOM versions and embedded servers. • ✓ Spring Boot 4-Layer Architecture — Presentation, Business Service, Persistence & Database Flow — Strict separation of concerns: **1. Presentation Layer** (`@RestController`, HTTP/DTO validation), **2. Service Layer** (`@Service`, business rules & `@Transactional` boundaries), **3. Persistence Layer** (`@Repository`, JPA/JDBC), and **4. Database**. • ✓ Core Spring Boot 3.4 Features — `SpringApplication`, `CommandLineRunner`, Profiles & Java 21 Virtual Threads — Customizing `SpringApplication` (`Banner.Mode`, `ApplicationRunner` / `CommandLineRunner` startup tasks, `ApplicationEvent` pub/sub), environment-specific `@Profile`, and enabling Project Loom **Java 21 Virtual Threads** with a single property!. • ✓ Build, verify, and ship Module Capstone: Layered Course Catalog Engine with Constructor DI & Java 21 Virtual Threads.

1.1 Spring Boot 3 Overview — Opinionated Defaults, Embedded Servers & Zero-XML Philosophy

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: SpringBootApplication & Opinionated auto-configuration • Core Syntax, CLI & Production APIs: @SpringBootApplication + SpringApplication.run(App.class, args) • Deep-Dive Mechanics: Embedded Tomcat 10.1 & Jakarta namespace migration • Advanced Patterns: ApplicationContext IoC container & Stand-alone executable JAR • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Minimal Stand-Alone Spring Boot 3.4 Application & Bean Count Inspector • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Spring Boot 3 Overview

Purpose: To understand and apply Spring Boot 3 Overview — Opinionated Defaults, Embedded Servers & Zero-XML Philosophy in real Springboot systems — specifically: How Spring Boot w...

What Topic Is Next → Next Topic 1.2: Spring Framework vs Spring MVC vs Spring Boot — IoC Container, `DispatcherServlet` & Auto-Setup

1.2 Spring Framework vs Spring MVC vs Spring Boot — IoC Container, `DispatcherServlet` & Auto-Setup

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: IoC Dependency Injection & DispatcherServlet web module • Core Syntax, CLI & Production APIs: ApplicationContext (Spring Core) + DispatcherServlet (Spring MVC) + @EnableAutoConfiguration... • Deep-Dive Mechanics: ConditionalOnMissingBean back-off & Zero web.xml ceremony • Advanced Patterns: BOM version alignment & Embedded vs external servlet container • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Constructor Dependency Injection Without XML or Field `@Autowired` • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Spring Framework vs Spring MVC vs Spring Boot

Purpose: To understand and apply Spring Framework vs Spring MVC vs Spring Boot — IoC Container, `DispatcherServlet` & Auto-Setup in real Springboot systems — specifically: Clear a...

What Topic Is Next → Next Topic 1.3: Spring Boot 4-Layer Architecture — Presentation, Business Service, Persistence & Database Flow

1.3 Spring Boot 4-Layer Architecture — Presentation, Business Service, Persistence & Database Flow

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Presentation Controller layer & Service business transaction layer • Core Syntax, CLI & Production APIs: @RestController -> @Service (@Transactional) -> @Repository (JpaRepository) -> HikariDataSou... • Deep-Dive Mechanics: Repository persistence layer & DTO vs Entity boundary • Advanced Patterns: Stateless singleton beans & Short DB connection hold time • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Clean 3-Bean Layered Flow Mapping Domain Model to Immutable `record` DTO • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Spring Boot 4-Layer Architecture

Purpose: To understand and apply Spring Boot 4-Layer Architecture — Presentation, Business Service, Persistence & Database Flow in real Springboot systems — specifically: Strict s...

What Topic Is Next → Next Topic 1.4: Core Spring Boot 3.4 Features — `SpringApplication`, `CommandLineRunner`, Profiles & Java 21 Virtual Threads

1.4 Core Spring Boot 3.4 Features — `SpringApplication`, `CommandLineRunner`, Profiles & Java 21 Virtual Threads

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Spring.threads.virtual.enabled=true & ApplicationRunner vs CommandLineRunner • Core Syntax, CLI & Production APIs: CommandLineRunner / ApplicationRunner + spring.threads.virtual.enabled=true • Deep-Dive Mechanics: SpringApplication customization & ApplicationEvent decoupled events • Advanced Patterns: Lazy initialization trade-offs & Graceful shutdown lifecycle • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Startup Verification Hook Using `ApplicationRunner` Bean • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Core Spring Boot 3.4 Features

Purpose: To understand and apply Core Spring Boot 3.4 Features — `SpringApplication`, `CommandLineRunner`, Profiles & Java 21 Virtual Threads in real Springboot systems — specific...

What Topic Is Next → Module 1 Capstone (Layered Course Catalog Engine with Constructor DI & Java 21 Virtual Threads) → Next Topic 2.1: `Spring Initializr` (`start.spring.io`) — Project Metadata, Maven Wrapper (`mvnw`) & Directory Layout

Module 1 Hands-On Capstone Project (100 Marks): Layered Course Catalog Engine with Constructor DI & Java 21 Virtual Threads — Architect a 4-layer Spring Boot 3.4 service (`CourseController` -> `CourseService` -> `CourseRepository` -> immutable `CourseRecord`) using zero `@Autowired` field injection, `spring.threads...

MODULE 2 OF 16 • STAGE 1 • 11 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 2: Project Bootstrapping (`Spring Initializr`), Maven Starters, Executable Fat JARs & DevTools

Bootstrap clean Maven/Gradle Spring Boot projects, master BOM dependency management via Starters, understand `BOOT-INF` Nested Fat JAR classloading + Docker Layered JARs, and accelerate local loops with DevTools.

Objectives: ✓ `Spring Initializr` (`start.spring.io`) — Project Metadata, Maven Wrapper (`mvnw`) & Directory Layout — Generating standardized builds via `start.spring.io` (or IDE wizard / `curl https://start.spring.io/starter.zip`), why committing Maven Wrapper (`mvnw` / `mvnw.cmd`) guarantees reproducible CI builds, and root package placement rules. • ✓ Spring Boot Starters & `spring-boot-starter-parent` BOM Dependency Management — How opinionated `spring-boot-starter-*` poms (`starter-web`, `starter-data-jpa`, `starter-security`, `starter-test`) bundle cohesive transitive libraries while `spring-boot-dependencies` Bill of Materials (BOM) guarantees tested version compatibility. • ✓ Executable Fat JARs (`JarLauncher`, `BOOT-INF`) vs External Tomcat WAR Deployment — How `spring-boot-maven-plugin` (`repackage` goal) creates a self-contained **Executable Nested JAR** (`Main-Class: JarLauncher`, `Start-Class: YourApp`, `BOOT-INF/classes`, `BOOT-INF/lib`), Docker Layered JAR caching, and `SpringBootServletInitializer` for legacy WARs. • ✓ Spring Boot DevTools (`RestartClassLoader`), LiveReload & Rapid Local Iteration — How `spring-boot-devtools` uses a dual-ClassLoader trick (**Base ClassLoader** keeps immutable ` ~/.m2` JARs warm in memory; **RestartClassLoader** reloads ONLY your edited `target/classes` in ~300 ms!), disables template caches in dev, and turns itself OFF in packaged JARs. • ✓ Build, verify, and ship Module Capstone: Docker-Layered Spring Boot 3 Maven Build & Custom Company Starter Blueprint.

2.1 `Spring Initializr` (`start.spring.io`) — Project Metadata, Maven Wrapper (`mvnw`) & Directory Layout [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Root package @ComponentScan rule! & Maven Wrapper (mvnw) reproducibility • Core Syntax, CLI & Production APIs: curl https://start.spring.io/starter.zip -d javaVersion=21 -d dependencies=web,data-jpa,vali... • Deep-Dive Mechanics: Src/main/java vs src/main/resources & Src/test/java mirror package • Advanced Patterns: Java 21 LTS target & JAR vs WAR packaging choice • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Verifying Root Package Component-Scan Coverage • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for `Spring Initializr` (`start.spring.io`)

Purpose: To understand and apply `Spring Initializr` (`start.spring.io`) — Project Metadata, Maven Wrapper (`mvnw`) & Directory Layout in real Springboot systems — specifically: G...

What Topic Is Next → Next Topic 2.2: Spring Boot Starters & `spring-boot-starter-parent` BOM Dependency Management

2.2 Spring Boot Starters & `spring-boot-starter-parent` BOM Dependency Management [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Spring-boot-starter-parent & Spring-boot-dependencies BOM import • Core Syntax, CLI & Production APIs: <parent><groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-parent</a... • Deep-Dive Mechanics: Curated transitive dependencies & Mvnw dependency:tree • Advanced Patterns: Excluding Tomcat for Undertow/Jetty & Third-party starter naming convention • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Maven BOM Import Pattern When Custom Corporate Parent Exists • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Spring Boot Starters

Purpose: To understand and apply Spring Boot Starters & `spring-boot-starter-parent` BOM Dependency Management in real Springboot systems — specifically: How opinionated `spring-b...

What Topic Is Next → Next Topic 2.3: Executable Fat JARs (`JarLauncher`, `BOOT-INF`) vs External Tomcat WAR Deployment

2.3 Executable Fat JARs (`JarLauncher`, `BOOT-INF`) vs External Tomcat WAR Deployment [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Spring-boot-maven-plugin repackage & BOOT-INF/classes & BOOT-INF/lib • Core Syntax, CLI & Production APIs: ./mvnw clean package && java -jar target/academy-0.0.1-SNAPSHOT.jar • Deep-Dive Mechanics: JarLauncher & Start-Class manifest & Docker Layered JAR caching • Advanced Patterns: SpringBootServletInitializer for WAR & Provided scope for external Tomcat • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Extending `SpringBootServletInitializer` When Packaging a Deployable `.war` • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Executable Fat JARs (`JarLauncher`, `BOOT-INF`) vs E...

Purpose: To understand and apply Executable Fat JARs (`JarLauncher`, `BOOT-INF`) vs External Tomcat WAR Deployment in real Springboot systems — specifically: How `spring-boot-mave...

What Topic Is Next → Next Topic 2.4: Spring Boot DevTools (`RestartClassLoader`), LiveReload & Rapid Local Iteration

2.4 Spring Boot DevTools (`RestartClassLoader`), LiveReload & Rapid Local Iteration [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Base vs RestartClassLoader architecture & Sub-second local hot restart • Core Syntax, CLI & Production APIs: spring-boot-devtools (<optional>true</optional>) — Base ClassLoader + RestartClassLoader • Deep-Dive Mechanics: Automatic template cache disable in dev & Auto-disabled in java -jar production • Advanced Patterns: <optional>true</optional> in pom.xml & Never enable remote devtools in prod • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Inspecting ClassLoader Identity Under DevTools vs Standard JVM Launch • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Spring Boot DevTools (`RestartClassLoader`)

Purpose: To understand and apply Spring Boot DevTools (`RestartClassLoader`), LiveReload & Rapid Local Iteration in real Springboot systems — specifically: How `spring-boot-devtoo...

What Topic Is Next → Module 2 Capstone (Docker-Layered Spring Boot 3 Maven Build & Custom Company Starter Blueprint) → Next Topic 3.1: Core Stereotype & DI Annotations — `@SpringBootApplication`, `@Service`, `@Bean`, `@Qualifier` & Scopes

Module 2 Hands-On Capstone Project (100 Marks): Docker-Layered Spring Boot 3 Maven Build & Custom Company Starter Blueprint — Configure a Java 21 `pom.xml` inheriting `spring-boot-starter-parent:3.4.x`, extract layered JAR segments (`dependencies`, `spring-boot-loader`, `application`) for fast 2-second Docker image...

MODULE 3 OF 16 • STAGE 1 • 13 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 3: Auto-Configuration Internals, Core Stereotype Annotations & Externalized Configuration

Demystify `@SpringBootApplication`, master Constructor Injection + Bean Scopes, inspect `AutoConfiguration.imports` + `@ConditionalOn*` evaluation reports (`--debug`), and bind type-safe immutable `@ConfigurationProperties` Java 21 `record`s.

Objectives: ✓ Core Stereotype & DI Annotations — `@SpringBootApplication`, `@Service`, `@Bean`, `@Qualifier` & Scopes — The 3-in-1 `@SpringBootApplication` meta-annotation, `@Component` vs `@Service` vs `@Repository` (which adds `PersistenceExceptionTranslationPostProcessor`!), `@Configuration` (`proxyBeanMethods`) + `@Bean`, `@Primary` vs `@Qualifier`, and Singleton vs Prototype bean scopes. • ✓ Spring Boot 3 Auto-Configuration Internals — `AutoConfiguration.imports` & `@ConditionalOn*` Guards — How Spring Boot 3 moved from legacy `spring.factories` to **`META-INF/spring/org.springframework.boot.autoconfigure.AutoConfiguration.imports`**, evaluates `@ConditionalOnClass`, `@ConditionalOnMissingBean`, `@ConditionalOnProperty`, and prints the Positive/Negative match report via `--debug`. • ✓ Externalized Config — `application.yml`, Property Precedence, `@Profile` & `@ConfigurationProperties` Records — 12-Factor externalized configuration: `application.yml` vs `.properties`, OS Environment Variables (`SPRING_DATASOURCE_URL` relaxed binding!) overriding packaged files, profile-specific `application-prod.yml`, and validated `@ConfigurationProperties` Java 21 `record`s. • ✓ Embedded Server Tuning — `server.port`, Random Test Ports (`0`), Graceful Shutdown & Compression — Configuring `server.port` (CLI `--server.port=9090`, `SERVER_PORT`, or `server.port=0` for OS-assigned random free port in integration tests!), `server.shutdown=graceful` (draining in-flight requests on `SIGTERM`!), and HTTP Gzip/Zstd response compression. • ✓ Build, verify, and ship Module Capstone: Type-Safe Multi-Environment Payment Gateway Auto-Configuration & `@ConfigurationProperties` Engine.

3.1 Core Stereotype & DI Annotations — `@SpringBootApplication`, `@Service`, `@Bean`, `@Qualifier` & Scopes [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: 3-in-1 @SpringBootApplication & Constructor DI over field @Autowired • Core Syntax, CLI & Production APIs: @SpringBootApplication = @SpringBootConfiguration + @EnableAutoConfiguration + @ComponentSca... • Deep-Dive Mechanics: Stateless singleton thread-safety! & @Repository exception translation • Advanced Patterns: @Primary vs @Qualifier resolution & @Configuration CGLIB @Bean proxying • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Disambiguating Multiple Strategy Beans Cleanly with `@Primary` and `@Qualifier` • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Core Stereotype

Purpose: To understand and apply Core Stereotype & DI Annotations — `@SpringBootApplication`, `@Service`, `@Bean`, `@Qualifier` & Scopes in real Springboot systems — specifically:...

What Topic Is Next → Next Topic 3.2: Spring Boot 3 Auto-Configuration Internals — `AutoConfiguration.imports` & `@ConditionalOn*` Guards

3.2 Spring Boot 3 Auto-Configuration Internals — `AutoConfiguration.imports` & `@ConditionalOn*` Guards [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: AutoConfiguration.imports (Boot 3+) & @ConditionalOnClass ASM check • Core Syntax, CLI & Production APIs: @AutoConfiguration + @ConditionalOnClass(DataSource.class) + @ConditionalOnMissingBean • Deep-Dive Mechanics: @ConditionalOnMissingBean user override & @ConditionalOnProperty toggle • Advanced Patterns: --debug ConditionsEvaluationReport & Exclude = DataSourceAutoConfiguration.class • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Writing a Self-Backing-Off `@AutoConfiguration` Class • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Spring Boot 3 Auto-Configuration Internals

Purpose: To understand and apply Spring Boot 3 Auto-Configuration Internals — `AutoConfiguration.imports` & `@ConditionalOn*` Guards in real Springboot systems — specifically: How...

What Topic Is Next → Next Topic 3.3: Externalized Config — `application.yml`, Property Precedence, `@Profile` & `@ConfigurationProperties` Records

3.3 Externalized Config — `application.yml`, Property Precedence, `@Profile` & `@ConfigurationProperties` Records [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Immutable @ConfigurationProperties record & @Validated fail-fast startup check • Core Syntax, CLI & Production APIs: @Validated @ConfigurationProperties(prefix = "academy") public record AcademyProps(@NotBlank... • Deep-Dive Mechanics: OS env var relaxed binding & Application-{profile}.yml precedence • Advanced Patterns: Duration & DataSize unit binding & Never commit prod secrets to git • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Type-Safe Validated `@ConfigurationProperties` Using a Java 21 `record` • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Externalized Config

Purpose: To understand and apply Externalized Config — `application.yml`, Property Precedence, `@Profile` & `@ConfigurationProperties` Records in real Springboot systems — specifi...

What Topic Is Next → Next Topic 3.4: Embedded Server Tuning — `server.port`, Random Test Ports (`0`), Graceful Shutdown & Compression

3.4 Embedded Server Tuning — `server.port`, Random Test Ports (`0`), Graceful Shutdown & Compression [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Server.shutdown=graceful drain! & Server.port=0 ephemeral test port • Core Syntax, CLI & Production APIs: server.port=8080 | server.shutdown=graceful | spring.lifecycle.timeout-per-shutdown-phase=30... • Deep-Dive Mechanics: WebServerInitializedEvent port lookup & Server.tomcat.threads.max tuning • Advanced Patterns: Server.compression.enabled=true & PortInUseFailureAnalyzer diagnostics • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Programmatic Server Port & Graceful Shutdown Customizer • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Embedded Server Tuning

Purpose: To understand and apply Embedded Server Tuning — `server.port`, Random Test Ports (`0`), Graceful Shutdown & Compression in real Springboot systems — specifically: Config...

What Topic Is Next → Module 3 Capstone (Type-Safe Multi-Environment Payment Gateway Auto-Configuration & `@ConfigurationProperties` Engine) → Next Topic 4.1: `spring-boot-starter-web` — Embedded Tomcat, Jackson `ObjectMapper` & MVC Auto-Wiring

Module 3 Hands-On Capstone Project (100 Marks): Type-Safe Multi-Environment Payment Gateway Auto-Configuration & `@ConfigurationProperties` Engine — Create an immutable `@Validated @ConfigurationProperties(prefix = "academy.payment") record PaymentProps(...)` failing fast at startup if API timeouts or URLs are invalid, paired with `@Prof...

MODULE 4 OF 16 • STAGE 1 • 13 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 4: Spring Web MVC, `DispatcherServlet` Lifecycle & Production RESTful Web Services

Master `spring-boot-starter-web`, trace the `DispatcherServlet` Front Controller lifecycle step-by-step, and build clean HTTP REST endpoints with `@RestController`, `ResponseEntity<T>`, `201 Created` `Location` headers, and idempotent verbs.

Objectives: ✓ `spring-boot-starter-web` — Embedded Tomcat, Jackson `ObjectMapper` & MVC Auto-Wiring — What `spring-boot-starter-web` auto-configures: Embedded Apache Tomcat (`8080`), `DispatcherServlet` mapped to `/`, `MappingJackson2HttpMessageConverter` (`ObjectMapper` + `JavaTimeModule`), static asset serving, and `MultipartResolver`. • ✓ `DispatcherServlet` Front Controller Lifecycle — `HandlerMapping`, `HandlerInterceptor` & Message Converters — Step-by-step HTTP request pipeline: Filter Chain -> **`DispatcherServlet`** -> `RequestMappingHandlerMapping` -> `HandlerInterceptor.preHandle()` -> `RequestMappingHandlerAdapter` (resolving `@PathVariable`/`@RequestBody`) -> Controller -> `HttpMessageConverter` JSON response!. • ✓ Routing & Parameter Binding — `@PathVariable`, `@RequestParam`, `@RequestHeader` & `ResponseEntity<T>` — Clean URI path variables (`/courses/{slug}`) vs optional query filters (`@RequestParam(defaultValue = "0") int page`), explicit parameter names (`-parameters` compiler flag in Boot 3!), and full status/header control with `ResponseEntity<T>`. • ✓ Resource Mutation — `@PostMapping` (`201 Created` + `Location`), `PUT` vs `PATCH` & `@DeleteMapping` (`204`) — Deserializing JSON payloads with `@RequestBody`, building RFC-9110 `Location` headers via `ServletUriComponentsBuilder.fromCurrentRequest()`, understanding HTTP **Idempotency** (`GET`, `PUT`, `DELETE` are idempotent; `POST` is NOT!), and returning `204 No Content`. • ✓ Build, verify, and ship Module Capstone: RESTful Enrollment & Cohort Management API (`201 Created` + `Location` + Cursor Pagination).

4.1 `spring-boot-starter-web` — Embedded Tomcat, Jackson `ObjectMapper` & MVC Auto-Wiring [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Embedded Tomcat 10.1 default & MappingJackson2HttpMessageConverter • Core Syntax, CLI & Production APIs: spring-boot-starter-web -> Tomcat + DispatcherServlet + Jackson ObjectMapper • Deep-Dive Mechanics: Java 21 record JSON serialization & Jackson2ObjectMapperBuilderCustomizer • Advanced Patterns: Servlet MVC vs WebFlux classpath rule & MultipartFile upload limits • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: RestController Returning an Immutable Java 21 `record` Serialized by Jackson • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for `spring-boot-starter-web`

Purpose: To understand and apply `spring-boot-starter-web` — Embedded Tomcat, Jackson `ObjectMapper` & MVC Auto-Wiring in real Springboot systems — specifically: What `spring-boot...

What Topic Is Next → Next Topic 4.2: `DispatcherServlet` Front Controller Lifecycle — `HandlerMapping`, `HandlerInterceptor` & Message Converters

4.2 `DispatcherServlet` Front Controller Lifecycle — `HandlerMapping`, `HandlerInterceptor` & Message Converters [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: DispatcherServlet Front Controller & RequestMappingHandlerMapping • Core Syntax, CLI & Production APIs: DispatcherServlet -> RequestMappingHandlerMapping -> HandlerAdapter -> HttpMessageConverter • Deep-Dive Mechanics: HandlerInterceptor preHandle/afterCompletion & HttpMessageConverter content negotiation • Advanced Patterns: @RestController = @Controller + @ResponseBody & Single-read ServletInputStream trap • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Lightweight Request Timing `HandlerInterceptor` Registered via `WebMvcConfigurer` • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for `DispatcherServlet` Front Controller Lifecycle

Purpose: To understand and apply `DispatcherServlet` Front Controller Lifecycle — `HandlerMapping`, `HandlerInterceptor` & Message Converters in real Springboot systems — specific...

What Topic Is Next → Next Topic 4.3: Routing & Parameter Binding — `@PathVariable`, `@RequestParam`, `@RequestHeader` & `ResponseEntity<T>`

4.3 Routing & Parameter Binding — `@PathVariable`, `@RequestParam`, `@RequestHeader` & `ResponseEntity<T>` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: @PathVariable resource identity & @RequestParam filter & pagination • Core Syntax, CLI & Production APIs: @GetMapping("/courses/{slug}") public ResponseEntity<CourseDto> getBySlug(@PathVariable Stri... • Deep-Dive Mechanics: Cap max page size <= 100! & ResponseEntity status & headers • Advanced Patterns: Boot 3.2+ -parameters compiler flag & @RequestHeader metadata binding • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Type-Safe `@PathVariable` & Capped `@RequestParam` Endpoint • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Routing

Purpose: To understand and apply Routing & Parameter Binding — `@PathVariable`, `@RequestParam`, `@RequestHeader` & `ResponseEntity<T>` in real Springboot systems — specifically:...

What Topic Is Next → Next Topic 4.4: Resource Mutation — `@PostMapping` (`201 Created` + `Location`), `PUT` vs `PATCH` & `@DeleteMapping` (`204`)

4.4 Resource Mutation — `@PostMapping` (`201 Created` + `Location`), `PUT` vs `PATCH` & `@DeleteMapping` (`204`) [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: 201 Created + Location URI header & ServletUriComponentsBuilder • Core Syntax, CLI & Production APIs: URI location = ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand... • Deep-Dive Mechanics: PUT full replace vs PATCH partial & HTTP Idempotency & Idempotency-Key • Advanced Patterns: 204 No Content on DELETE & Never return 200 OK for errors • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Creating a Resource with `201 Created` and RFC-Compliant `Location` Header • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Resource Mutation

Purpose: To understand and apply Resource Mutation — `@PostMapping` (`201 Created` + `Location`), `PUT` vs `PATCH` & `@DeleteMapping` (`204`) in real Springboot systems — specific...

What Topic Is Next → Module 4 Capstone (RESTful Enrollment & Cohort Management API (`201 Created` + `Location` + Cursor Pagination)) → Next Topic 5.1: Global Exception Handling — `@RestControllerAdvice`, `@ExceptionHandler` & RFC-9457 `ProblemDetail`

Module 4 Hands-On Capstone Project (100 Marks): RESTful Enrollment & Cohort Management API (`201 Created` + `Location` + Cursor Pagination) — Build an idiomatic `/api/v1/cohorts` REST controller returning `ResponseEntity<CohortDto>` with `ServletUriComponentsBuilder` `Location` headers on `POST`, `204 No Content` on `DELETE`, and...

MODULE 5 OF 16 • STAGE 2 • 13 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 5: Global Exception Handling (`ProblemDetail`), Jakarta Validation, DTO Filtering & API Versioning

Centralize error handling with `@RestControllerAdvice` + Spring 6 RFC-9457 `ProblemDetail`, validate nested DTOs with Jakarta Bean Validation (`@Valid`), prevent Mass-Assignment/data leaks with DTO projections, and version APIs cleanly.

Objectives: ✓ Global Exception Handling — `@RestControllerAdvice`, `@ExceptionHandler` & RFC-9457 `ProblemDetail` — Replacing scattered controller `try/catch` blocks with a single `@RestControllerAdvice` aspect, extending `ResponseEntityExceptionHandler`, and returning Spring Framework 6's native **`ProblemDetail`** (`application/problem+json` RFC-9457 standard!). • ✓ Jakarta Bean Validation (`spring-boot-starter-validation`) — `@Valid`, Nested DTOs & Custom Validators — Note that Spring Boot 2.3+ extracted Hibernate Validator into explicit **`spring-boot-starter-validation`**! Enforcing `@NotBlank`, `@Email`, `@Size`, `@Positive`, cascading `@Valid` on nested objects/lists, and `@Validated` for method/query parameters. • ✓ DTO Projections, Static (`@JsonIgnore`) & Dynamic (`MappingJacksonValue`) Filtering + Content Negotiation — Preventing sensitive field exposure (`passwordHash`, `internalMargin`) and Mass-Assignment attacks (`role = "ADMIN"`): dedicated Request/Response Java `record` DTOs, `@JsonProperty(access = WRITE_ONLY)`, `@JsonView`, `MappingJacksonValue` `SimpleBeanPropertyFilter`, and XML/JSON Content Negotiation. • ✓ REST API Versioning Strategies, `Spring HATEOAS` (`EntityModel`) & Internationalization (`i18n`) — Four industry API versioning patterns (**URI Path** `/api/v1/users`, **Custom Header** `X-API-VERSION=1`, **Media-Type Accept** `application/vnd.company.v1+json`, **Query Param**), hypermedia links with `spring-boot-starter-hateoas` (`WebMvcLinkBuilder`), and `Accept-Language` `MessageSource` `i18n`. • ✓ Build, verify, and ship Module Capstone: RFC-9457 `ProblemDetail` Error Engine, Custom `@ValidCoupon` Validator & Versioned REST API.

5.1 Global Exception Handling — `@RestControllerAdvice`, `@ExceptionHandler` & RFC-9457 `ProblemDetail`

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: @RestControllerAdvice global handler & RFC-9457 ProblemDetail (Spring 6+) • Core Syntax, CLI & Production APIs: @RestControllerAdvice + @ExceptionHandler(NotFoundException.class) -> ProblemDetail.forStatu... • Deep-Dive Mechanics: ResponseEntityExceptionHandler & 404 vs 409 Conflict vs 422 • Advanced Patterns: Never expose stacktraces in prod! & Correlation traceId in error properties • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Centralized `@RestControllerAdvice` Returning RFC-9457 `ProblemDetail` • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Global Exception Handling

Purpose: To understand and apply Global Exception Handling — `@RestControllerAdvice`, `@ExceptionHandler` & RFC-9457 `ProblemDetail` in real Springboot systems — specifically: Rep...

What Topic Is Next → Next Topic 5.2: Jakarta Bean Validation (`spring-boot-starter-validation`) — `@Valid`, Nested DTOs & Custom Validators

5.2 Jakarta Bean Validation (`spring-boot-starter-validation`) — `@Valid`, Nested DTOs & Custom Validators

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Explicit spring-boot-starter-validation! & @Valid before @RequestBody required! • Core Syntax, CLI & Production APIs: spring-boot-starter-validation -> @Valid @RequestBody EnrollmentReq(@NotBlank String name, @... • Deep-Dive Mechanics: @NotBlank vs @NotNull vs @NotEmpty & Cascading @Valid on nested fields • Advanced Patterns: MethodArgumentNotValidException 400 & Custom @Constraint + ConstraintValidator • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Validated Request `record` Catching Blank Names & Negative Fees Automatically • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Jakarta Bean Validation (`spring-boot-starter-valida...

Purpose: To understand and apply Jakarta Bean Validation (`spring-boot-starter-validation`) — `@Valid`, Nested DTOs & Custom Validators in real Springboot systems — specifically:...

What Topic Is Next → Next Topic 5.3: DTO Projections, Static (`@JsonIgnore`) & Dynamic (`MappingJacksonValue`) Filtering + Content Negotiation

5.3 DTO Projections, Static (`@JsonIgnore`) & Dynamic (`MappingJacksonValue`) Filtering + Content Negotiation

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Never bind @RequestBody to @Entity! & OWASP Mass-Assignment prevention • Core Syntax, CLI & Production APIs: Dedicated Java record DTOs + @JsonIgnore / @JsonView / SimpleBeanPropertyFilter • Deep-Dive Mechanics: Java record Request/Response DTOs & @JsonIgnore vs WRITE_ONLY access • Advanced Patterns: @JsonView & MappingJacksonValue & Accept header Content Negotiation • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Safe Public Profile Projection Stripping Sensitive Internal Fields • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for DTO Projections

Purpose: To understand and apply DTO Projections, Static (`@JsonIgnore`) & Dynamic (`MappingJacksonValue`) Filtering + Content Negotiation in real Springboot systems — specificall...

What Topic Is Next → Next Topic 5.4: REST API Versioning Strategies, `Spring HATEOAS` (`EntityModel`) & Internationalization (`i18n`)

5.4 REST API Versioning Strategies, `Spring HATEOAS` (`EntityModel`) & Internationalization (`i18n`)

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: URI vs Header vs Media-Type versioning & Additive non-breaking evolution • Core Syntax, CLI & Production APIs: URI (/api/v1/...) vs Header (headers = "X-API-VERSION=2") + EntityModel.of(dto, linkTo(metho... • Deep-Dive Mechanics: Spring HATEOAS EntityModel & Link & WebMvcLinkBuilder type-safe links • Advanced Patterns: MessageSource & LocaleContextHolder i18n & Deprecation & Sunset HTTP headers • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Header-Based API Versioning & `LocaleContextHolder` Internationalization • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for REST API Versioning Strategies

Purpose: To understand and apply REST API Versioning Strategies, `Spring HATEOAS` (`EntityModel`) & Internationalization (`i18n`) in real Springboot systems — specifically: Four i...

What Topic Is Next → Module 5 Capstone (RFC-9457 `ProblemDetail` Error Engine, Custom `@ValidCoupon` Validator & Versioned REST API) → Next Topic 6.1: Spring AOP Architecture — CGLIB Subclass Proxies, JoinPoints, Pointcuts & Self-Invocation Trap

Module 5 Hands-On Capstone Project (100 Marks): RFC-9457 `ProblemDetail` Error Engine, Custom `@ValidCoupon` Validator & Versioned REST API — Implement a global `@RestControllerAdvice` mapping `ResourceNotFoundException` (`404`), `MethodArgumentNotValidException` (`400` with per-field violation map), and `DataIntegrityViolationExc...

MODULE 6 OF 16 • STAGE 2 • 11 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 6: Aspect-Oriented Programming (`Spring AOP`) — CGLIB Proxies, Pointcuts & Cross-Cutting Advices

Modularize cross-cutting concerns (latency telemetry, audit logging, rate-limiting, retry policies) cleanly using `Spring AOP`, understand CGLIB vs JDK Dynamic Proxies + the Self-Invocation trap, and write `@Before`, `@AfterReturning`, `@AfterThrowing`, and `@Around` aspects.

Objectives: ✓ Spring AOP Architecture — CGLIB Subclass Proxies, JoinPoints, Pointcuts & Self-Invocation Trap — Core AOP vocabulary (**Aspect**, **JoinPoint**, **Advice**, **Pointcut**, **Weaving**), why Spring Boot 2+/3+ defaults to **CGLIB** class-based runtime proxies (`spring.aop.proxy-target-class=true`), and how `@Transactional`, `@Cacheable`, and `@PreAuthorize` are ALL powered by Spring AOP!. • ✓ `@Before`, `@After`, `@AfterReturning` & `@AfterThrowing` — Security, Audit & Failure Logging Advices — Choosing the right advice lifecycle: **`@Before`** (pre-check / argument audit via `JoinPoint.getArgs()`), **`@AfterReturning`** (inspect return value when successful), **`@AfterThrowing`** (alert on unhandled exceptions without swallowing them!), and **`@After`** (`finally` cleanup). • ✓ `@Around` Advice with `ProceedingJoinPoint` — SLA Latency Profiling & Idempotent Retry Interceptors — The most powerful AOP advice: `@Around` wraps target execution completely, decides whether/how many times to call `pjp.proceed()`, measures nanoTime SLA latency in `try/finally`, and binds Micrometer `Timer` metrics. • ✓ Build, verify, and ship Module Capstone: Custom `@TrackLatency` & `@AuditAction` Spring AOP Telemetry Suite.

6.1 Spring AOP Architecture — CGLIB Subclass Proxies, JoinPoints, Pointcuts & Self-Invocation Trap [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: CGLIB subclass runtime proxy & Self-invocation this.method() bypass! • Core Syntax, CLI & Production APIs: spring-boot-starter-aop -> @Aspect + @Component + Pointcut (execution / @annotation / within... • Deep-Dive Mechanics: Avoid final on proxied bean methods & Aspect, JoinPoint, Pointcut, Advice • Advanced Patterns: Annotation-driven @annotation pointcut & Powers @Transactional & @Cacheable • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Verifying Spring CGLIB Proxy Wrapper Class Name at Runtime • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Spring AOP Architecture

Purpose: To understand and apply Spring AOP Architecture — CGLIB Subclass Proxies, JoinPoints, Pointcuts & Self-Invocation Trap in real Springboot systems — specifically: Core AOP...

What Topic Is Next → Next Topic 6.2: `@Before`, `@After`, `@AfterReturning` & `@AfterThrowing` — Security, Audit & Failure Logging Advices

6.2 `@Before`, `@After`, `@AfterReturning` & `@AfterThrowing` — Security, Audit & Failure Logging Advices [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: @Before JoinPoint inspection & @AfterReturning result binding • Core Syntax, CLI & Production APIs: @Before("@annotation(Audited)") | @AfterReturning(pointcut = "...", returning = "result") |... • Deep-Dive Mechanics: @AfterThrowing exception telemetry & @After finally semantics • Advanced Patterns: Never log plaintext PII/passwords! & Refactoring-safe @annotation pointcuts • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Audit Aspect Logging Successful Enrollment Outcomes via `@AfterReturning` • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for `@Before`

Purpose: To understand and apply `@Before`, `@After`, `@AfterReturning` & `@AfterThrowing` — Security, Audit & Failure Logging Advices in real Springboot systems — specifically: C...

What Topic Is Next → Next Topic 6.3: `@Around` Advice with `ProceedingJoinPoint` — SLA Latency Profiling & Idempotent Retry Interceptors

6.3 `@Around` Advice with `ProceedingJoinPoint` — SLA Latency Profiling & Idempotent Retry Interceptors [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: ProceedingJoinPoint.proceed()! & Always return pjp.proceed() result! • Core Syntax, CLI & Production APIs: @Around("@annotation(TimedOp)") public Object profile(ProceedingJoinPoint pjp) throws Throwa... • Deep-Dive Mechanics: Try/finally monotonic System.nanoTime() & Never swallow Throwable silently • Advanced Patterns: @Order aspect execution precedence & Micrometer @Timed integration • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Production `@Around` SLA Profiler Using Monotonic `System.nanoTime()` • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for `@Around` Advice with `ProceedingJoinPoint`

Purpose: To understand and apply `@Around` Advice with `ProceedingJoinPoint` — SLA Latency Profiling & Idempotent Retry Interceptors in real Springboot systems — specifically: The...

What Topic Is Next → Module 6 Capstone (Custom `@TrackLatency` & `@AuditAction` Spring AOP Telemetry Suite) → Next Topic 7.1: Spring JDBC & `HikariCP` — Connection Pool Sizing & Modern `JdbcClient` / `JdbcTemplate`

Module 6 Hands-On Capstone Project (100 Marks): Custom `@TrackLatency` & `@AuditAction` Spring AOP Telemetry Suite — Create a custom `@Target(ElementType.METHOD) @Retention(RetentionPolicy.RUNTIME) @interface TrackLatency` annotation and an `@Aspect` using `@Around` (`ProceedingJoinPoint`) that logs method...

MODULE 7 OF 16 • STAGE 2 • 15 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 7: Data Persistence — `HikariCP`, `JdbcClient`, H2 Database, `Spring Data JPA` & Solving `N+1`

Connect to relational databases via ultra-fast `HikariCP`, write fluent SQL with Spring 6.1 `JdbcClient` / `JdbcTemplate`, run rapid local/test schemas on H2 + Flyway, and master `Spring Data JPA` (`JpaRepository`, `@Transactional`, `FetchType.LAZY`, `@EntityGraph`).

Objectives: ✓ Spring JDBC & `HikariCP` — Connection Pool Sizing & Modern `JdbcClient` / `JdbcTemplate` — Why Spring Boot defaults to **`HikariCP`** (fastest zero-overhead JDBC pool on the JVM), pool sizing formula (`connections = (core_count * 2) + effective_spindle_count`), and Spring Boot 3.2+ fluent **`JdbcClient`** mapping directly to Java `record`s (`DataClassRowMapper`). • ✓ Embedded H2 Database & Schema Migrations — Dev Prototyping vs Production `Flyway` / `Liquibase` — Running zero-install `jdbc:h2:mem:academydb` in `dev` profile (`spring.h2.console.enabled=true`), initializing `schema.sql`/`data.sql` (`defer-datasource-initialization`), and why production MUST use **`Flyway`** or **`Liquibase`** with `ddl-auto: validate`!. • ✓ `Spring Data JPA` & Hibernate 6 — `@Entity`, `JpaRepository`, `@Query`, Pagination & `@Transactional` — JPA Persistence Context (1st-level cache) & Automatic Dirty Checking, `JpaRepository<T, ID>`, derived query methods, custom JPQL `@Query`, `Page<T>` vs `Slice<T>` (`Slice` skips the expensive `SELECT COUNT(*)`!), and `@Transactional(readOnly = true)` optimizations. • ✓ JPA Entity Relationships (`@ManyToOne`, `@OneToMany`) — `FetchType.LAZY` & Crushing the `N+1` Trap — Bidirectional mapping (`@OneToMany(mappedBy = "course")` + `@ManyToOne(fetch = FetchType.LAZY)` `@JoinColumn` owning side!), why JPA's

default `EAGER` on `@ManyToOne` / `@OneToOne` is a silent performance killer, and solving **`N+1` queries** via `@EntityGraph`, `JOIN FETCH`, or DTO Projections. • ✓ Build, verify, and ship Module Capstone: Zero-`N+1` Spring Data JPA Academy Ledger with Optimistic `@Version` Locking.

7.1 Spring JDBC & `HikariCP` — Connection Pool Sizing & Modern `JdbcClient` / `JdbcTemplate` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: HikariCP default connection pool & JdbcClient fluent named parameters • Core Syntax, CLI & Production APIs: jdbcClient.sql("SELECT code, title, fee FROM courses WHERE code = :code").param("code", code... • Deep-Dive Mechanics: Right-sized pool (10-20 conns) & SQL injection parameterized defense • Advanced Patterns: Connection-timeout & max-lifetime & DataAccessException translation • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Fluent Parameterized Query Mapped to a Java 21 `record` Using `JdbcClient` • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Spring JDBC

Purpose: To understand and apply Spring JDBC & `HikariCP` — Connection Pool Sizing & Modern `JdbcClient` / `JdbcTemplate` in real Springboot systems — specifically: Why Boo...

What Topic Is Next → Next Topic 7.2: Embedded H2 Database & Schema Migrations — Dev Prototyping vs Production `Flyway` / `Liquibase`

7.2 Embedded H2 Database & Schema Migrations — Dev Prototyping vs Production `Flyway` / `Liquibase` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Ddl-auto=validate in production! & Flyway versioned SQL migrations • Core Syntax, CLI & Production APIs: spring.datasource.url=jdbc:h2:mem:testdb | prod: spring.jpa.hibernate.ddl-auto=validate + Fl... • Deep-Dive Mechanics: Never enable H2 console in prod! & Defer-datasource-initialization • Advanced Patterns: H2 dialect vs Testcontainers real DB & Jdbc:h2:mem:db;DB_CLOSE_DELAY=-1 • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Safe Production vs Local Profile Configuration Guard • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Embedded H2 Database

Purpose: To understand and apply Embedded H2 Database & Schema Migrations — Dev Prototyping vs Production `Flyway` / `Liquibase` in real Springboot systems — specifically: Running...

What Topic Is Next → Next Topic 7.3: `Spring Data JPA` & Hibernate 6 — `@Entity`, `JpaRepository`, `@Query`, Pagination & `@Transactional`

7.3 `Spring Data JPA` & Hibernate 6 — `@Entity`, `JpaRepository`, `@Query`, Pagination & `@Transactional` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Spring.jpa.open-in-view=false! & @Transactional(readOnly = true) • Core Syntax, CLI & Production APIs: interface CourseRepository extends JpaRepository<CourseEntity, Long> { @Query("...") Slice<C... • Deep-Dive Mechanics: Hibernate 1st-level Dirty Checking & Slice<T> vs Page<T> COUNT(*) cost • Advanced Patterns: Java record DTO JPQL projections & @Version optimistic locking • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Spring Data JPA Repository Returning an Interface/Record Projection Slice • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for `Spring Data JPA`

Purpose: To understand and apply `Spring Data JPA` & Hibernate 6 — `@Entity`, `JpaRepository`, `@Query`, Pagination & `@Transactional` in real Springboot systems — specifically: J...

What Topic Is Next → Next Topic 7.4: JPA Entity Relationships (`@ManyToOne`, `@OneToMany`) — `FetchType.LAZY` & Crushing the `N+1` Trap

7.4 JPA Entity Relationships (`@ManyToOne`, `@OneToMany`) — `FetchType.LAZY` & Crushing the `N+1` Trap [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: @ManyToOne defaults to EAGER — set LAZY! & N+1 SELECT problem & @EntityGraph • Core Syntax, CLI & Production APIs: @ManyToOne(fetch = FetchType.LAZY) @JoinColumn(name = "course_id") + @EntityGraph(attributeP... • Deep-Dive Mechanics: JOIN FETCH vs Hibernate BatchSize & Owning side @JoinColumn vs mappedBy • Advanced Patterns: Never use Lombok @Data on @Entity! & LazyInitializationException & DTOs • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Eliminating `N+1` Queries Using Declarative `@EntityGraph` on `JpaRepository` • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for JPA Entity Relationships (`@ManyToOne`, `@OneToMany`...

Purpose: To understand and apply JPA Entity Relationships (`@ManyToOne`, `@OneToMany`) — `FetchType.LAZY` & Crushing the `N+1` Trap in real Springboot systems — specifically: Bidi...

What Topic Is Next → Module 7 Capstone (Zero-`N+1` Spring Data JPA Academy Ledger with Optimistic `@Version` Locking) → Next Topic 8.1: Spring Cache Abstraction — `@EnableCaching`, `@Cacheable(sync = true)`, `@CachePut` & `@CacheEvict`

Module 7 Hands-On Capstone Project (100 Marks): Zero-`N+1` Spring Data JPA Academy Ledger with Optimistic `@Version` Locking — Model `Course` (1) <-> (N) `CohortBatch` entities with explicit `FetchType.LAZY` on `@ManyToOne`, fetch paginated DTO summaries in 1 SQL query via JPQL Constructor/Record Projection, disable...

MODULE 8 OF 16 • STAGE 2 • 11 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 8: High-Speed Caching (`@Cacheable`), Cache Providers (`Caffeine` / `Redis`) & `Thymeleaf` SSR

Slash database read load by 95% using Spring's Cache Abstraction (`@EnableCaching`, `@Cacheable`, `@CachePut`, `@CacheEvict`), compare in-memory `Caffeine`/`EhCache` vs distributed `Redis`, and render XSS-safe HTML with `Thymeleaf`.

Objectives: ✓ Spring Cache Abstraction — `@EnableCaching`, `@Cacheable(sync = true)`, `@CachePut` & `@CacheEvict` — Decoupling business logic from cache storage via `CacheManager`, SpEL cache keys (`key = "#courseCode"`), conditional caching (`condition` / `unless`), **`sync = true` Cache Stampede protection**, and cache invalidation on writes (`@CacheEvict`). • ✓ Cache Providers (`Caffeine`, `EhCache 3`, `Redis`) — Local JVM Heap Speed vs Distributed Consistency — Choosing the right backing store: **Caffeine** (Window-TinyLFU near-optimal hit rate in nanosecond JVM RAM), **EhCache 3** (`JCache` JSR-107 off-heap/disk tiers), and **Redis** (`spring-boot-starter-data-redis` shared across N Kubernetes pods with JSON serialization & jittered TTLs). • ✓ Server-Side Rendering with `Thymeleaf` — Natural HTML5 Templates, `th:text` XSS Escaping & Emails — Returning view names from `@Controller` (not `@RestController`!) backed by `src/main/resources/templates/*.html`, `Model.addAttribute()`, automatic HTML escaping with `th:text`, iteration (`th:each`), fragments, and rendering HTML PDFs / emails via `SpringTemplateEngine`. • ✓ Build, verify, and ship Module Capstone: Two-Tier Course Catalog Cache (`Caffeine` + TTL Eviction) & `Thymeleaf` Certificate Previewer.

8.1 Spring Cache Abstraction — `@EnableCaching`, `@Cacheable(sync = true)`, `@CachePut` & `@CacheEvict` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Never use unbounded default Map in prod! & @Cacheable(sync = true) stampede lock • Core Syntax, CLI & Production APIs: @EnableCaching + @Cacheable(value = "catalog", key = "#slug", sync = true) + @CacheEvict(val... • Deep-Dive Mechanics: @CachePut vs @Cacheable semantics & @CacheEvict invalidation on update • Advanced Patterns: SpEL key = "#slug" & unless & AOP proxy self-invocation rule applies • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Stampede-Safe `@Cacheable` Catalog Lookup Paired with `@CacheEvict` • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Spring Cache Abstraction

Purpose: To understand and apply Spring Cache Abstraction — `@EnableCaching`, `@Cacheable(sync = true)`, `@CachePut` & `@CacheEvict` in real Springboot systems — specifically: Dec...

What Topic Is Next → Next Topic 8.2: Cache Providers (`Caffeine`, `EhCache 3`, `Redis`) — Local JVM Heap Speed vs Distributed Consistency

8.2 Cache Providers (`Caffeine`, `EhCache 3`, `Redis`) — Local JVM Heap Speed vs Distributed Consistency [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Caffeine Window-TinyLFU local L1 & Redis distributed multi-pod L2 • Core Syntax, CLI & Production APIs: spring.cache.type=caffeine | spring.cache.caffeine.spec=maximumSize=5000,expireAfterWrite=10... • Deep-Dive Mechanics: Local cache multi-pod staleness trap! & Always bound maximumSize + TTL! • Advanced Patterns: JSON serializer over Java Serialization & TTL jitter against cache avalanche • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Configuring Bounded `Caffeine` Cache Spec with Size & Write-Expiry Caps • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Cache Providers (`Caffeine`, `EhCache 3`, `Redis`)

Purpose: To understand and apply Cache Providers (`Caffeine`, `EhCache 3`, `Redis`) — Local JVM Heap Speed vs Distributed Consistency in real Springboot systems — specifically: Ch...

What Topic Is Next → Next Topic 8.3: Server-Side Rendering with `Thymeleaf` — Natural HTML5 Templates, `th:text` XSS Escaping & Emails

8.3 Server-Side Rendering with `Thymeleaf` — Natural HTML5 Templates, `th:text` XSS Escaping & Emails [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: @Controller vs @RestController view resolution & Th:text automatic XSS escaping! • Core Syntax, CLI & Production APIs: @Controller + Model.addAttribute("course", dto) + Fallback</... • Deep-Dive Mechanics: Never use th:utext on user input & Templates/*.html & static/* assets • Advanced Patterns: Spring.thymeleaf.cache=true in prod & SpringTemplateEngine for HTML emails • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: SSR `@Controller` Populating a `Model` for `templates/course-card.html` • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Server-Side Rendering with `Thymeleaf`

Purpose: To understand and apply Server-Side Rendering with `Thymeleaf` — Natural HTML5 Templates, `th:text` XSS Escaping & Emails in real Springboot systems — specifically: Retur...

What Topic Is Next → Module 8 Capstone (Two-Tier Course Catalog Cache (`Caffeine` + TTL Eviction) & `Thymeleaf` Certificate Previewer) → Next Topic 9.1: OpenAPI 3.1 & Swagger UI (`springdoc-openapi`) — `@Operation`, `@Schema` & Bearer JWT Docs

Module 8 Hands-On Capstone Project (100 Marks): Two-Tier Course Catalog Cache (`Caffeine` + TTL Eviction) & `Thymeleaf` Certificate Previewer — Configure `@EnableCaching` with `CaffeineCacheManager` (`maximumSize=1000, expireAfterWrite=10m`), guard hot course lookups with `@Cacheable(cacheNames = "courses", key = "#slug", sync = tru...

MODULE 9 OF 16 • STAGE 3 • 13 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 9: OpenAPI 3 / Swagger UI, `Spring Security 6` Filter Chains & `Actuator` Observability

Auto-generate interactive OpenAPI 3.1 Swagger docs (`springdoc-openapi`), lock down microservices with modern `Spring Security 6` Lambda DSL (`SecurityFilterChain`, Stateless JWT, `@PreAuthorize`, `BCrypt`), and expose Kubernetes probes + Prometheus metrics via `Actuator`.

Objectives: ✓ OpenAPI 3.1 & Swagger UI (`springdoc-openapi`) — `@Operation`, `@Schema` & Bearer JWT Docs — Replacing deprecated `Springfox` (broken on Boot 3 `jakarta.*`!) with **`org.springdoc:springdoc-openapi-starter-webmvc-ui`**, auto-generating `/v3/api-docs` + `/swagger-ui.html` from Java `record` DTOs & Jakarta `@Valid` annotations, and gating Swagger UI in production. • ✓ `Spring Security 6` Architecture — `SecurityFilterChain`, Lambda DSL, Stateless JWT, CORS & `@PreAuthorize` — How `DelegatingFilterProxy` -> `FilterChainProxy` -> **`SecurityFilterChain`** intercepts requests BEFORE `DispatcherServlet`! Spring Security 6 Lambda DSL (`http.csrf(...)`, `authorizeHttpRequests`), `BCryptPasswordEncoder`, OAuth2 / JWT Resource Server, and `@EnableMethodSecurity` (`@PreAuthorize`). • ✓ `Spring Boot Actuator` & Micrometer — Kubernetes Liveness/Readiness Probes, Custom `HealthIndicator` & Prometheus — Production observability with `spring-boot-starter-actuator`: `/actuator/health/liveness` (restart deadlocked pod) vs `/actuator/health/readiness` (stop routing traffic while DB warms up), Micrometer dimensional metrics (`/actuator/prometheus`), and OpenTelemetry distributed tracing. • ✓ Build, verify, and ship Module Capstone: Hardened `Spring Security 6` Stateless JWT Resource Server + `Actuator` + OpenAPI 3 Gateway.

9.1 OpenAPI 3.1 & Swagger UI (`springdoc-openapi`) — `@Operation`, `@Schema` & Bearer JWT Docs [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Springdoc-openapi v2+ for Boot 3! & Never use abandoned Springfox • Core Syntax, CLI & Production APIs: springdoc-openapi-starter-webmvc-ui + @Tag + @Operation(summary = "...") + @SecurityRequirem... • Deep-Dive Mechanics: /v3/api-docs & /swagger-ui.html & @Valid constraints -> OpenAPI schema • Advanced Patterns: Bearer JWT @SecurityScheme & Gate Swagger UI in production! • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Annotating a REST Endpoint & `OpenAPI` Bean with `@Operation` Metadata • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for OpenAPI 3.1

Purpose: To understand and apply OpenAPI 3.1 & Swagger UI (`springdoc-openapi`) — `@Operation`, `@Schema` & Bearer JWT Docs in real Springboot systems — specifically: Replacing de...

What Topic Is Next → Next Topic 9.2: `Spring Security 6` Architecture — `SecurityFilterChain`, Lambda DSL, Stateless JWT, CORS & `@PreAuthorize`

9.2 `Spring Security 6` Architecture — `SecurityFilterChain`, Lambda DSL, Stateless JWT, CORS & `@PreAuthorize` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: @Bean SecurityFilterChain Lambda DSL! & WebSecurityConfigurerAdapter removed! • Core Syntax, CLI & Production APIs: @Bean SecurityFilterChain filterChain(HttpSecurity http) + @EnableMethodSecurity + PasswordE... • Deep-Dive Mechanics: SessionCreationPolicy.STATELESS JWT & BCryptPasswordEncoder / Argon2 • Advanced Patterns: @EnableMethodSecurity @PreAuthorize & FilterChain runs BEFORE DispatcherServlet • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Modern `Spring Security 6` Stateless `SecurityFilterChain` Bean Configuration • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for `Spring Security 6` Architecture

Purpose: To understand and apply `Spring Security 6` Architecture — `SecurityFilterChain`, Lambda DSL, Stateless JWT, CORS & `@PreAuthorize` in real Springboot systems — specifica...

What Topic Is Next → Next Topic 9.3: `Spring Boot Actuator` & Micrometer — Kubernetes Liveness/Readiness Probes, Custom `HealthIndicator` & Prometheus

9.3 `Spring Boot Actuator` & Micrometer — Kubernetes Liveness/Readiness Probes, Custom `HealthIndicator` & Prometheus [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: NEVER expose include=* publicly! & /actuator/heapdump & /env leak risk • Core Syntax, CLI & Production APIs: management.endpoints.web.exposure.include=health,info,prometheus | management.endpoint.healt... • Deep-Dive Mechanics: Liveness vs Readiness K8s probes! & Management.server.port isolation • Advanced Patterns: Micrometer MeterRegistry & Prometheus & Custom HealthIndicator bean • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Custom `HealthIndicator` Reporting Assessment Engine Readiness • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for `Spring Boot Actuator`

Purpose: To understand and apply `Spring Boot Actuator` & Micrometer — Kubernetes Liveness/Readiness Probes, Custom `HealthIndicator` & Prometheus in real Springboot systems — spe...

What Topic Is Next → Module 9 Capstone (Hardened `Spring Security 6` Stateless JWT Resource Server + `Actuator` + OpenAPI 3 Gateway) → Next Topic 10.1: Spring Boot Testing Pyramid — `JUnit 5`, `Mockito`, `@WebMvcTest` (`MockMvc`), `@DataJpaTest` & `@SpringBootTest`

Module 9 Hands-On Capstone Project (100 Marks): Hardened `Spring Security 6` Stateless JWT Resource Server + `Actuator` + OpenAPI 3 Gateway — Configure a `SecurityFilterChain` bean permitting `/api/v1/public/**` and `/actuator/health/**`, requiring `ROLE_ADMIN` via `@PreAuthorize("hasRole('ADMIN')")` on curriculum mutations, hashi...

MODULE 10 OF 16 • STAGE 3 • 12 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 10: Automated Testing (`@SpringBootTest`, Slice Tests) & Cloud-Native REST Best Practices

Ship bug-free releases with `spring-boot-starter-test` (`JUnit 5`, `Mockito`, `AssertJ`), fast Test Slices (`@WebMvcTest` `MockMvc`, `@DataJpaTest`, `Testcontainers`), and master Richardson Maturity Model + Cloud-Native 12-Factor production standards.

Objectives: ✓ Spring Boot Testing Pyramid — `JUnit 5`, `Mockito`, `@WebMvcTest` (`MockMvc`), `@DataJpaTest` & `@SpringBootTest` — What `spring-boot-starter-test` bundles (`JUnit Jupiter 5`, `Mockito`, `AssertJ`, `Hamcrest`, `JsonPath`, `MockMvc`), why **Slice Annotations** (`@WebMvcTest` loads ONLY web layer; `@DataJpaTest` loads ONLY JPA + transactional rollback!) beat heavy `@SpringBootTest` overuse, and Boot 3.4 `@MockitoBean`. • ✓ Richardson Maturity Model (Levels 0–3) & Production Cloud-Native Spring Boot Best Practices — Evaluating API maturity via the **Richardson Maturity Model** (Level 0: Swamp of POX single URI -> Level 1: Resource URIs -> Level 2: HTTP Verbs + Status Codes -> Level 3: HATEOAS Hypermedia Controls), Bucket4j / Redis Rate Limiting (`429 Too Many Requests`), Timeouts on `RestClient`, and Structured JSON Logging (`spring.logging.structured.format.console=ecs` in Boot 3.4!). • ✓ Build, verify, and ship Module Capstone: Full Testing Pyramid Suite (`Mockito` Unit + `@WebMvcTest` `MockMvc` + `@DataJpaTest`) & Production Readiness Gate.

10.1 Spring Boot Testing Pyramid — `JUnit 5`, `Mockito`, `@WebMvcTest` (`MockMvc`), `@DataJpaTest` & `@SpringBootTest` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Unit vs @WebMvcTest vs @SpringBootTest & MockMvc HTTP status & jsonPath • Core Syntax, CLI & Production APIs: @ExtendWith(MockitoExtension.class) (Unit) | @WebMvcTest + MockMvc + @MockitoBean (Web Slice... • Deep-Dive Mechanics: @MockitoBean (Spring Boot 3.4+) & @DataJpaTest auto-rollback slice • Advanced Patterns: ApplicationContext test caching & Testcontainers real DB parity • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Fast `@WebMvcTest` Controller Slice Test Using `MockMvc` & `jsonPath` • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Spring Boot Testing Pyramid

Purpose: To understand and apply Spring Boot Testing Pyramid — `JUnit 5`, `Mockito`, `@WebMvcTest` (`MockMvc`), `@DataJpaTest` & `@SpringBootTest` in real Springboot systems — spe...

What Topic Is Next → Next Topic 10.2: Richardson Maturity Model (Levels 0–3) & Production Cloud-Native Spring Boot Best Practices

10.2 Richardson Maturity Model (Levels 0–3) & Production Cloud-Native Spring Boot Best Practices [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Richardson Maturity Levels 0 to 3 & RestClient with explicit timeouts! • Core Syntax, CLI & Production APIs: Richardson Level 2/3 + RestClient with explicit connect/read timeouts + Boot 3.4 Structured... • Deep-Dive Mechanics: Resilience4j CircuitBreaker & Retry & 429 Too Many Requests rate limiting • Advanced Patterns: Boot 3.4 Structured JSON Logging & 12-Factor stateless horizontal scale • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Production `RestClient` Configured with Explicit Connect & Read Timeouts • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Richardson Maturity Model (Levels 0–3)

Purpose: To understand and apply Richardson Maturity Model (Levels 0–3) & Production Cloud-Native Spring Boot Best Practices in real Springboot systems — specifically: Evaluating...

What Topic Is Next → Module 10 Capstone (Full Testing Pyramid Suite (`Mockito` Unit + `@WebMvcTest` `MockMvc` + `@DataJpaTest`) & Production Readiness Gate) → Next Topic 11.1: Modern Lambda SecurityFilterChain, CORS & Password Encoders

Module 10 Hands-On Capstone Project (100 Marks): Full Testing Pyramid Suite (`Mockito` Unit + `@WebMvcTest` `MockMvc` + `@DataJpaTest`) & Production Readiness Gate — Write a 3-tier automated test suite for the Enrollment Service: (1) pure `@ExtendWith(MockitoExtension.class)` unit test verifying business rules in 5 ms, (2) `@WebMvcTest` `MockMvc` slice t...

MODULE 11 OF 16 • STAGE 3 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Spring Security 6 FilterChains, OAuth2/JWT & Method Authorization

Production-grade Springboot Training engineering module covering Spring Security 6 FilterChains, OAuth2/JWT & Method Authorization with hands-on labs and capstone build.

Objectives: ✓ Modern Lambda SecurityFilterChain, CORS & Password Encoders — Secure REST endpoints cleanly with Argon2/BCrypt and structured ProblemDetail 401/403s. • ✓ RSA JWT Issuance, Refresh Token Rotation & OAuth2 Resource Server — Verify signed claims and tenant IDs across endpoints with zero session memory bloat. • ✓ @PreAuthorize Method RBAC & Security Audit Listeners — Enforce fine-grained ownership checks directly on business service methods. • ✓ Build, verify, and ship Module Capstone: Stateless JWT + Role-Scoped Enterprise Spring Boot API.

11.1 Modern Lambda SecurityFilterChain, CORS & Password Encoders

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of CORS • Integration with RSA JWT Issuance, Refresh Token Rotation & OAuth2 Resource Server • Live Debugging, Telemetry & Technical Interview Trade • What Is Modern Lambda SecurityFilterChain, CORS & Password Encoders & Why Industry Teams Use It • In Springboot Training, Modern Lambda SecurityFilterChain, CORS & Password Encoders exists for one clear real • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply Modern Lambda SecurityFilterChain, CORS & Password Encoders in real Springboot systems — specifically: Secure REST endpoints cleanly with Argon2/B...

What Topic Is Next → Next Topic 11.2: RSA JWT Issuance, Refresh Token Rotation & OAuth2 Resource Server

11.2 RSA JWT Issuance, Refresh Token Rotation & OAuth2 Resource Server

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Refresh Token Rotation • Integration with @PreAuthorize Method RBAC & Security Audit Listeners • Live Debugging, Telemetry & Technical Interview Trade • What Is RSA JWT Issuance, Refresh Token Rotation & OAuth2 Resource Server & Why Industry Teams Use It • In Springboot Training, RSA JWT Issuance, Refresh Token Rotation & OAuth2 Resource Server exists for one clear real • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply RSA JWT Issuance, Refresh Token Rotation & OAuth2 Resource Server in real Springboot systems — specifically: Verify signed claims and tenant IDs a...

What Topic Is Next → Next Topic 11.3: @PreAuthorize Method RBAC & Security Audit Listeners

11.3 @PreAuthorize Method RBAC & Security Audit Listeners

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Security Audit Listeners • Integration with Modern Lambda SecurityFilterChain, CORS & Password Encoders • Live Debugging, Telemetry & Technical Interview Trade • What Is @PreAuthorize Method RBAC & Security Audit Listeners & Why Industry Teams Use It • In Springboot Training, @PreAuthorize Method RBAC & Security Audit Listeners exists for one clear real • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply @PreAuthorize Method RBAC & Security Audit Listeners in real Springboot systems — specifically: Enforce fine-grained ownership checks directly on...

What Topic Is Next → Module 11 Capstone (Stateless JWT + Role-Scoped Enterprise Spring Boot API) → Next Topic 12.1: Eliminating N+1 Queries via @EntityGraph, JOIN FETCH & Record DTOs

Module 11 Hands-On Capstone Project (100 Marks): Stateless JWT + Role-Scoped Enterprise Spring Boot API — Design, implement, test, and document 'Stateless JWT + Role-Scoped Enterprise Spring Boot API' applying every production concept from Module 11 (Spring Security 6 FilterChains, OAuth2/JWT &...

MODULE 12 OF 16 • STAGE 3 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Hibernate/JPA Performance, Flyway Migrations & Redis Caching

Production-grade Springboot Training engineering module covering Hibernate/JPA Performance, Flyway Migrations & Redis Caching with hands-on labs and capstone build.

Objectives: ✓ Eliminating N+1 Queries via @EntityGraph, JOIN FETCH & Record DTOs — Slash 150 lazy-loading SQL queries down to 1 indexed projection query. • ✓ Optimistic Locking (@Version), HikariCP Tuning & Flyway SQL Migrations — Prevent concurrent lost updates and version every schema change reproducibly. • ✓ Spring @Cacheable with Redis TTLs & Targeted Eviction — Serve hot reference and catalog endpoints in < 4ms under peak concurrency. • ✓ Build, verify, and ship Module Capstone: N+1-Free Order Engine with Flyway & Two-Tier Redis Cache.

12.1 Eliminating N+1 Queries via @EntityGraph, JOIN FETCH & Record DTOs

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of JOIN FETCH • Integration with Optimistic Locking (@Version), HikariCP Tuning & Flyway SQL Migrations • Live Debugging, Telemetry & Technical Interview Trade • What Is Eliminating N+1 Queries via @EntityGraph, JOIN FETCH & Record DTOs & Why Industry Teams Use It • In Springboot Training, Eliminating N+1 Queries via @EntityGraph, JOIN FETCH & Record DTOs exists for one clear rea... • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply Eliminating N+1 Queries via @EntityGraph, JOIN FETCH & Record DTOs in real Springboot systems — specifically: Slash 150 lazy-loading SQL queries d...

What Topic Is Next → Next Topic 12.2: Optimistic Locking (@Version), HikariCP Tuning & Flyway SQL Migrations

12.2 Optimistic Locking (@Version), HikariCP Tuning & Flyway SQL Migrations [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of HikariCP Tuning • Integration with Spring @Cacheable with Redis TTLs & Targeted Eviction • Live Debugging, Telemetry & Technical Interview Trade • What Is Optimistic Locking (@Version), HikariCP Tuning & Flyway SQL Migrations & Why Industry Teams Use It • In Springboot Training, Optimistic Locking (@Version), HikariCP Tuning & Flyway SQL Migrations exists for one clear... • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply Optimistic Locking (@Version), HikariCP Tuning & Flyway SQL Migrations in real Springboot systems — specifically: Prevent concurrent lost updates...

What Topic Is Next → Next Topic 12.3: Spring @Cacheable with Redis TTLs & Targeted Eviction

12.3 Spring @Cacheable with Redis TTLs & Targeted Eviction [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Targeted Eviction • Integration with Eliminating N+1 Queries via @EntityGraph, JOIN FETCH & Record DTOs • Live Debugging, Telemetry & Technical Interview Trade • What Is Spring @Cacheable with Redis TTLs & Targeted Eviction & Why Industry Teams Use It • In Springboot Training, Spring @Cacheable with Redis TTLs & Targeted Eviction exists for one clear real • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply Spring @Cacheable with Redis TTLs & Targeted Eviction in real Springboot systems — specifically: Serve hot reference and catalog endpoints in < 4m...

What Topic Is Next → Module 12 Capstone (N+1-Free Order Engine with Flyway & Two-Tier Redis Cache) → Next Topic 13.1: Spring Kafka Producers, Consumer Groups, Retries & Dead-Letter Topics

Module 12 Hands-On Capstone Project (100 Marks): N+1-Free Order Engine with Flyway & Two-Tier Redis Cache — Design, implement, test, and document 'N+1-Free Order Engine with Flyway & Two-Tier Redis Cache' applying every production concept from Module 12 (Hibernate/JPA Performance, Flyway Migration...

MODULE 13 OF 16 • STAGE 4 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Async Messaging (Kafka/RabbitMQ), Resilience4j & Scheduled Locks

Production-grade Springboot Training engineering module covering Async Messaging (Kafka/RabbitMQ), Resilience4j & Scheduled Locks with hands-on labs and capstone build.

Objectives: ✓ Spring Kafka Producers, Consumer Groups, Retries & Dead-Letter Topics — Decouple slow invoice/notification pipelines from synchronous customer APIs. • ✓ Resilience4j CircuitBreaker, Bulkhead & Exponential Backoff Retry — Isolate degraded third-party gateways so upstream Spring threads never hang. • ✓ Transactional Outbox & ShedLock Distributed Cron Jobs — Guarantee reliable event publishing and single-pod execution across scaled replicas. • ✓ Build, verify, and ship Module Capstone: Fault-Tolerant Event-Driven Billing Service with Circuit Breakers.

13.1 Spring Kafka Producers, Consumer Groups, Retries & Dead-Letter Topics [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Consumer Groups • Integration with Resilience4j CircuitBreaker, Bulkhead & Exponential Backoff Retry • Live Debugging, Telemetry & Technical Interview Trade • What Is Spring Kafka Producers, Consumer Groups, Retries & Dead • In Springboot Training, Spring Kafka Producers, Consumer Groups, Retries & Dead • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply Spring Kafka Producers, Consumer Groups, Retries & Dead-Letter Topics in real Springboot systems — specifically: Decouple slow invoice/notificatio...

What Topic Is Next → Next Topic 13.2: Resilience4j CircuitBreaker, Bulkhead & Exponential Backoff Retry

13.2 Resilience4j CircuitBreaker, Bulkhead & Exponential Backoff Retry [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Bulkhead • Integration with Transactional Outbox & ShedLock Distributed Cron Jobs • Live Debugging, Telemetry & Technical Interview Trade • What Is Resilience4j CircuitBreaker, Bulkhead & Exponential Backoff Retry & Why Industry Teams Use It • In Springboot Training, Resilience4j CircuitBreaker, Bulkhead & Exponential Backoff Retry exists for one clear real • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply Resilience4j CircuitBreaker, Bulkhead & Exponential Backoff Retry in real Springboot systems — specifically: Isolate degraded third-party gateways...

What Topic Is Next → Next Topic 13.3: Transactional Outbox & ShedLock Distributed Cron Jobs

13.3 Transactional Outbox & ShedLock Distributed Cron Jobs [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of ShedLock Distributed Cron Jobs • Integration with Spring Kafka Producers, Consumer Groups, Retries & Dead • Live Debugging, Telemetry & Technical Interview Trade • What Is Transactional Outbox & ShedLock Distributed Cron Jobs & Why Industry Teams Use It • In Springboot Training, Transactional Outbox & ShedLock Distributed Cron Jobs exists for one clear real • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply Transactional Outbox & ShedLock Distributed Cron Jobs in real Springboot systems — specifically: Guarantee reliable event publishing and single-po...

What Topic Is Next → Module 13 Capstone (Fault-Tolerant Event-Driven Billing Service with Circuit Breakers) → Next Topic 14.1: @WebMvcTest Slices, Mockito & @ServiceConnection Testcontainers

Module 13 Hands-On Capstone Project (100 Marks): Fault-Tolerant Event-Driven Billing Service with Circuit Breakers — Design, implement, test, and document 'Fault-Tolerant Event-Driven Billing Service with Circuit Breakers' applying every production concept from Module 13 (Async Messaging (Kafka/RabbitMQ),...

MODULE 14 OF 16 • STAGE 4 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Testcontainers CI Suites, OpenTelemetry Actuator & Docker/K8s Prod

Production-grade Springboot Training engineering module covering Testcontainers CI Suites, OpenTelemetry Actuator & Docker/K8s Prod with hands-on labs and capstone build.

Objectives: ✓ @WebMvcTest Slices, Mockito & @ServiceConnection Testcontainers — Run automated integration tests against real PostgreSQL and Redis containers in CI. • ✓ Actuator Readiness/Liveness Probes, Micrometer Prometheus & Tracing — Track p99 latency, JVM heap, Hikari pools, and trace IDs across every request. • ✓ Layered Spring Boot Dockerfiles, Virtual Threads & Graceful Shutdown — Deploy fast-booting, zero-downtime Java 21 containers to cloud clusters. • ✓ Build, verify, and ship Module Capstone: Production-Ready Containerized Spring Boot 3 Release Pipeline.

14.1 @WebMvcTest Slices, Mockito & @ServiceConnection Testcontainers

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Mockito • Integration with Actuator Readiness/Liveness Probes, Micrometer Prometheus & Tracing • Live Debugging, Telemetry & Technical Interview Trade • What Is @WebMvcTest Slices, Mockito & @ServiceConnection Testcontainers & Why Industry Teams Use It • In Springboot Training, @WebMvcTest Slices, Mockito & @ServiceConnection Testcontainers exists for one clear real • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply @WebMvcTest Slices, Mockito & @ServiceConnection Testcontainers in real Springboot systems — specifically: Run automated integration tests against...

What Topic Is Next → Next Topic 14.2: Actuator Readiness/Liveness Probes, Micrometer Prometheus & Tracing

14.2 Actuator Readiness/Liveness Probes, Micrometer Prometheus & Tracing

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Micrometer Prometheus • Integration with Layered Spring Boot Dockerfiles, Virtual Threads & Graceful Shutdown • Live Debugging, Telemetry & Technical Interview Trade • What Is Actuator Readiness/Liveness Probes, Micrometer Prometheus & Tracing & Why Industry Teams Use It • In Springboot Training, Actuator Readiness/Liveness Probes, Micrometer Prometheus & Tracing exists for one clear re... • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply Actuator Readiness/Liveness Probes, Micrometer Prometheus & Tracing in real Springboot systems — specifically: Track p99 latency, JVM heap, Hikari...

What Topic Is Next → Next Topic 14.3: Layered Spring Boot Dockerfiles, Virtual Threads & Graceful Shutdown

14.3 Layered Spring Boot Dockerfiles, Virtual Threads & Graceful Shutdown

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Virtual Threads • Integration with @WebMvcTest Slices, Mockito & @ServiceConnection Testcontainers • Live Debugging, Telemetry & Technical Interview Trade • What Is Layered Spring Boot Dockerfiles, Virtual Threads & Graceful Shutdown & Why Industry Teams Use It • In Springboot Training, Layered Spring Boot Dockerfiles, Virtual Threads & Graceful Shutdown exists for one clear r... • Even if you come from a non • Runnable Code Implementation (JAVA)

Purpose: To understand and apply Layered Spring Boot Dockerfiles, Virtual Threads & Graceful Shutdown in real Springboot systems — specifically: Deploy fast-booting, zero-downtime...

What Topic Is Next → Module 14 Capstone (Production-Ready Containerized Spring Boot 3 Release Pipeline) → Next Topic 15.1: RFC 7807 ProblemDetails Global Error Responses & Jakarta Bean Validation Groups

Module 14 Hands-On Capstone Project (100 Marks): Production-Ready Containerized Spring Boot 3 Release Pipeline — Design, implement, test, and document 'Production-Ready Containerized Spring Boot 3 Release Pipeline' applying every production concept from Module 14 (Testcontainers CI Suites, OpenTelemetr...

MODULE 15 OF 16 • STAGE 4 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Spring Boot 3 Production Resilience: Caching, Rate Limiting & Async Events

Self-explanatory list-and-bullet production mastery module covering Spring Boot 3 Production Resilience: Caching, Rate Limiting & Async Events in Springboot Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken MockAttempt AI Viva.

Objectives: ✓ RFC 7807 ProblemDetails Global Error Responses & Jakarta Bean Validation Groups — Purpose: Return standardized, client-friendly JSON error contracts across every REST controller. • ✓ Spring Cache Abstraction with Redis TTLs, Cache Stampede Prevention & Eviction — Purpose: Accelerate read-heavy catalog/profile endpoints while keeping cached entries fresh on updates. • ✓ Resilience4j Retries, TimeLimiters, Circuit Breakers & @TransactionalEventListener — Purpose: Isolate downstream service failures and trigger side effects only after DB transactions commit. • ✓ Build, verify, and document 'Resilient Spring Boot 3 Order API with Redis Cache, Circuit Breakers & ProblemDetails' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + MockAttempt AI Technical Interviews.

15.1 RFC 7807 ProblemDetails Global Error Responses & Jakarta Bean Validation Groups

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for RFC 7807 ProblemDetails Global Error Responses & Jakarta Bean Validation G... • Performance Optimization, Complexity & Latency Impact of RFC 7807 ProblemDetails Global Error Responses & Jakarta B... • Live Debugging, Stack • Runnable Code Implementation (JAVA) • Hands • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for RFC 7807 ProblemDetails Global Error Responses • Production Hardening & Security: Spring Security 6 JWT/OAuth2 RBAC & OWASP API Top-10

Purpose: To understand and apply RFC 7807 ProblemDetails Global Error Responses & Jakarta Bean Validation Groups in real Springboot systems — specifically: Return standardized, cl...

What Topic Is Next → Next Topic 15.2: Spring Cache Abstraction with Redis TTLs, Cache Stampede Prevention & Eviction

15.2 Spring Cache Abstraction with Redis TTLs, Cache Stampede Prevention & Eviction [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Spring Cache Abstraction with Redis TTLs, Cache Stampede Prevention & Evic... • Performance Optimization, Complexity & Latency Impact of Spring Cache Abstraction with Redis TTLs, Cache Stampede P... • Live Debugging, Stack • Runnable Code Implementation (JAVA) • Hands • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Spring Cache Abstraction with Redis TTLs • Production Hardening & Security: Spring Security 6 JWT/OAuth2 RBAC & OWASP API Top-10

Purpose: To understand and apply Spring Cache Abstraction with Redis TTLs, Cache Stampede Prevention & Eviction in real Springboot systems — specifically: Accelerate read-heavy ca...

What Topic Is Next → Next Topic 15.3: Resilience4j Retries, TimeLimiters, Circuit Breakers & @TransactionalEventListener

15.3 Resilience4j Retries, TimeLimiters, Circuit Breakers & @TransactionalEventListener [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Resilience4j Retries, TimeLimiters, Circuit Breakers & @TransactionalEvent... • Performance Optimization, Complexity & Latency Impact of Resilience4j Retries, TimeLimiters, Circuit Breakers & @Tr... • Live Debugging, Stack • Runnable Code Implementation (JAVA) • Hands • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Resilience4j Retries • Production Hardening & Security: Spring Security 6 JWT/OAuth2 RBAC & OWASP API Top-10

Purpose: To understand and apply Resilience4j Retries, TimeLimiters, Circuit Breakers & @TransactionalEventListener in real Springboot systems — specifically: Isolate downstream s...

What Topic Is Next → Module 15 Capstone (Resilient Spring Boot 3 Order API with Redis Cache, Circuit Breakers & ProblemDetails) → Next Topic 16.1: Slice Tests (@WebMvcTest, @DataJpaTest) & Real-DB Integration with Testcontainers

Module 15 Hands-On Capstone Project (100 Marks): Resilient Spring Boot 3 Order API with Redis Cache, Circuit Breakers & ProblemDetails — Architect, implement, test, and document 'Resilient Spring Boot 3 Order API with Redis Cache, Circuit Breakers & ProblemDetails' applying every bullet-wise engineering concept from Module 15...

MODULE 16 OF 16 • STAGE 4 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Testcontainers Integration Suites, Micrometer Tracing & Spring Interviews

Self-explanatory list-and-bullet production mastery module covering Testcontainers Integration Suites, Micrometer Tracing & Spring Interviews in Springboot Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken MockAttempt AI Viva.

Objectives: ✓ Slice Tests (@WebMvcTest, @DataJpaTest) & Real-DB Integration with Testcontainers — Purpose: Verify JPA queries, Flyway migrations, and security filters against a real containerized database. • ✓ Spring Boot Actuator, Micrometer Metrics & Distributed Trace Propagation — Purpose: Expose production health/readiness endpoints and latency histograms for cloud monitoring. • ✓ Spring IoC/Proxy Pitfalls, N+1 Query Fixes & Enterprise Interview Scenarios — Purpose: Spot @Transactional self-invocation bugs and Hibernate N+1 bottlenecks instantly. • ✓ Build, verify, and document 'Testcontainers PostgreSQL Suite + Actuator Prometheus/OpenTelemetry Stack' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + MockAttempt AI Technical Interviews.

16.1 Slice Tests (@WebMvcTest, @DataJpaTest) & Real-DB Integration with Testcontainers [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Slice Tests (@WebMvcTest, @DataJpaTest) & Real • Performance Optimization, Complexity & Latency Impact of Slice Tests (@WebMvcTest, @DataJpaTest) & Real • Live Debugging, Stack • Runnable Code Implementation (JAVA) • Hands • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Slice Tests (@WebMvcTest, @DataJpaTest) • Production Hardening & Security: Spring Security 6 JWT/OAuth2 RBAC & OWASP API Top-10

Purpose: To understand and apply Slice Tests (@WebMvcTest, @DataJpaTest) & Real-DB Integration with Testcontainers in real Springboot systems — specifically: Verify JPA queries, F...

What Topic Is Next → Next Topic 16.2: Spring Boot Actuator, Micrometer Metrics & Distributed Trace Propagation

16.2 Spring Boot Actuator, Micrometer Metrics & Distributed Trace Propagation [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Spring Boot Actuator, Micrometer Metrics & Distributed Trace Propagation • Performance Optimization, Complexity & Latency Impact of Spring Boot Actuator, Micrometer Metrics & Distributed Tra... • Live Debugging, Stack • Runnable Code Implementation (JAVA) • Hands • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Spring Boot Actuator • Production Hardening & Security: Spring Security 6 JWT/OAuth2 RBAC & OWASP API Top-10

Purpose: To understand and apply Spring Boot Actuator, Micrometer Metrics & Distributed Trace Propagation in real Springboot systems — specifically: Expose production health/readi...

What Topic Is Next → Next Topic 16.3: Spring IoC/Proxy Pitfalls, N+1 Query Fixes & Enterprise Interview Scenarios

16.3 Spring IoC/Proxy Pitfalls, N+1 Query Fixes & Enterprise Interview Scenarios [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Spring IoC/Proxy Pitfalls, N+1 Query Fixes & Enterprise Interview Scenario... • Performance Optimization, Complexity & Latency Impact of Spring IoC/Proxy Pitfalls, N+1 Query Fixes & Enterprise In... • Live Debugging, Stack • Runnable Code Implementation (JAVA) • Hands • Idiomatic SPBT Implementation: constructor DI, Record DTOs, @Transactional boundaries & Resilience4j • Java 21 + Spring Boot 3 IoC & Hibernate 6 Persistence Context Execution & Memory Lifecycle for Spring IoC/Proxy Pitfalls • Production Hardening & Security: Spring Security 6 JWT/OAuth2 RBAC & OWASP API Top-10

Purpose: To understand and apply Spring IoC/Proxy Pitfalls, N+1 Query Fixes & Enterprise Interview Scenarios in real Springboot systems — specifically: Spot @Transactional self-in...

What Topic Is Next → Module 16 Capstone → End-of-Course Live Client Project: MediTrack Secure Enterprise REST API, RBAC Portal & Scheduled Billing Service

Module 16 Hands-On Capstone Project (100 Marks): Testcontainers PostgreSQL Suite + Actuator Prometheus/OpenTelemetry Stack — Architect, implement, test, and document 'Testcontainers PostgreSQL Suite + Actuator Prometheus/OpenTelemetry Stack' applying every bullet-wise engineering concept from Module 16 (Testcontai...

3. LIVE CLIENT PRODUCTION PROJECT & CAREER SCOPE

End-of-Course Live Client Production Project + Official Internship Confirmation Letter

Project: MediTrack Secure Enterprise REST API, RBAC Portal & Scheduled Billing Service • **Confirmed Role:** Java Spring Boot 3 Backend Engineering Intern

Domain & Production Infra: Healthcare Clinic Network & Insurance Claims Client (Java 21 + Spring Boot 3.3 + Spring Data JPA + Spring Security 6 JWT + PostgreSQL + Docker)

- Architected layered Spring Boot 3 REST APIs using DTO validation, @ControllerAdvice problem details, and OpenAPI docs.
- Secured endpoints with stateless Spring Security 6 filter chains, JWT access/refresh tokens, and method-level @PreAuthorize.
- Eliminated N+1 Hibernate JPA queries via entity graphs, Flyway migrations, and paginated projections.
- Validated APIs with JUnit 5, MockMvc, and Testcontainers before deploying Spring Actuator health-checked containers.

Target Job Roles: Spring Boot Backend Developer, Java REST API & Microservices Engineer, Enterprise Software Engineer (Java/Spring), Backend SDE-1 / SDE-2

Realistic Annual CTC Scope: **Fresher / Entry-Level (0–1 Yr):** ₹5.5 LPA – ₹10.0 LPA **Mid-Level Spring Microservices Engineer (1–3 Yrs):** ₹10.5 LPA – ₹21.0 LPA **Senior Java Microservices Architect (3–6+ Yrs):** ₹22.0 LPA – ₹42.0+ LPA

MockAttempt Academy Pvt. Ltd. • Email: support@mockattempt.com • Phone/WhatsApp: +91 95400 07917 • Web: https://mockattempt.com/courses/springboot-training

Try 2 Free Topic-Wise Timed Topic Mock Tests & Spoken MockAttempt AI Interviews daily on every topic above before enrolling.