Spring Boot Internals – Questions & Answers

1. How does Spring Boot decide which auto-configuration to apply?

Spring Boot scans the classpath and applies auto-configuration classes using @Conditional annotations like @ConditionalOnClass, @ConditionalOnBean, and @ConditionalOnProperty.

2. What happens internally when you add spring-boot-starter-web?

It brings Spring MVC, Jackson, validation APIs, and an embedded Tomcat server. Auto-configuration sets up DispatcherServlet and web infrastructure.

3. Why does Spring Boot prefer convention over configuration?

To reduce boilerplate code, improve productivity, and enforce best practices without requiring excessive manual configuration.

4. How does Spring Boot load application.properties internally?

Spring Boot loads it into the Environment using PropertySources, making values accessible via @Value and @ConfigurationProperties.

5. What is the exact startup flow of a Spring Boot application?

main() → SpringApplication.run() → create ApplicationContext → load environment → auto-configure beans → start embedded server.

6. Difference between @ComponentScan and @SpringBootApplication?

@ComponentScan scans for beans, while @SpringBootApplication combines @ComponentScan, @Configuration, and @EnableAutoConfiguration.

7. How does Spring Boot detect embedded Tomcat and configure it?

It detects Tomcat classes on the classpath and applies ServletWebServerFactory auto-configuration.

8. What happens if two beans of same type exist and no @Qualifier?

Spring throws NoUniqueBeanDefinitionException because it cannot decide which bean to inject.

9. How does Spring Boot handle profile-specific configuration?

It loads profile-specific files like application-dev.properties when spring.profiles.active is set.

10. What is the role of SpringFactoriesLoader?

It loads auto-configuration and extension classes from META-INF/spring.factories.

11. How does Spring Boot reduce XML configuration completely?

By using Java-based configuration, annotations, and auto-configuration.

12. Difference between @RestController and @Controller internally?

@RestController adds @ResponseBody, returning JSON/XML directly instead of views.

13. How does Spring Boot manage dependency versions automatically?

Through the spring-boot-dependencies BOM which controls compatible library versions.

14. What is the lifecycle of a Spring Bean in Spring Boot?

Instantiation → Dependency Injection → Initialization → Ready for use → Destruction.

15. How does Spring Boot handle externalized configuration?

Via property files, environment variables, command-line arguments, and config servers.

16. What happens if application.yml and application.properties both exist?

application.yml takes precedence over application.properties.

17. How does Spring Boot integrate with Actuator internally?

Actuator endpoints are registered using auto-configuration and exposed via MVC.

18. Difference between @Configuration and normal class?

@Configuration uses CGLIB proxies to ensure singleton beans.

19. How does Spring Boot create DataSource automatically?

It checks the classpath and properties, then auto-configures the DataSource.

20. What is the real use of CommandLineRunner?

To execute logic after the application context is fully initialized.

21. How does Spring Boot handle exception translation?

Through @Repository and PersistenceExceptionTranslationPostProcessor.

22. Difference between @EnableAutoConfiguration and @Import?

@EnableAutoConfiguration conditionally loads configs, @Import loads explicitly.

23. What happens if you exclude an auto-configuration class?

Spring Boot skips loading that auto-configuration during startup.

24. How does Spring Boot support microservices so easily?

By providing embedded servers, external config, and Spring Cloud integration.

25. What is the difference between fat jar and normal jar?

Fat jar contains the app and all dependencies, normal jar does not.

26. How does Spring Boot handle logging by default?

It uses Logback with sensible default configuration.

27. How does Spring Boot decide server port priority?

Command-line args > environment variables > application properties.

28. What happens internally when you hit a REST endpoint?

Request → DispatcherServlet → Controller → Response conversion.

29. Why is Spring Boot preferred for cloud-native apps?

Because of fast startup, external config, and easy containerization.

30. What are the most common Spring Boot performance mistakes?

Too many beans, blocking I/O, no caching, and heavy startup logic.