Spring Boot - Practice Sheet

10 conceptual Q&A covering auto-config, JPA, security, and testing

Q1. What is Spring Boot and how does it differ from the Spring Framework?

A: Spring Boot is a Spring module that provides auto-configuration, embedded servers, and starter POMs. Unlike plain Spring, it needs zero XML and runs as a self-contained JAR, removing the need to deploy to a separate app server.

Q2. How does Spring Boot auto-configuration work under the hood?

A: @EnableAutoConfiguration loads candidate classes from AutoConfiguration.imports. Each class uses @ConditionalOnClass or @ConditionalOnMissingBean to activate only when specific dependencies or beans are present on the classpath.

Q3. What does @SpringBootApplication combine and why is each part needed?

A: It merges @Configuration (declares beans), @EnableAutoConfiguration (activates Boot's auto-config machinery), and @ComponentScan (recursively discovers @Component, @Service, @Repository in the package tree).

Q4. How does Spring Boot externalize and prioritize configuration?

A: Boot loads sources in ascending-priority order: application.yml, profile files, OS environment variables, and CLI args each override the previous. @ConfigurationProperties binds a prefix to a typed POJO with IDE-validation support.

Q5. What is Spring Boot Actuator and what does it expose?

A: Actuator adds management HTTP endpoints: /actuator/health (liveness/readiness), /actuator/metrics (counters, timers), /actuator/loggers (change log levels at runtime), and /actuator/env. Each endpoint is securable and toggleable.

Q6. How do you configure a relational database in Spring Boot?

A: Add spring-boot-starter-data-jpa and a JDBC driver. Set spring.datasource.url, username, and password in application.properties. Boot auto-wires a DataSource, EntityManagerFactory, and JpaTransactionManager without any explicit @Bean definitions.

Q7. How do you replace the default embedded Tomcat with Jetty?

A: Exclude spring-boot-starter-tomcat from the web starter, then add spring-boot-starter-jetty. Auto-configuration detects Jetty on the classpath and selects it. Undertow is a third option available the same way.

Q8. How do you implement global REST exception handling in Spring Boot?

A: @RestControllerAdvice marks a class as a cross-cutting error handler. Methods annotated @ExceptionHandler(SomeEx.class) intercept matched exceptions and return a ResponseEntity with a structured error body and HTTP status code.

Q9. What are Spring Boot profiles and how do you activate one?

A: Profiles separate environment configuration. application-prod.yml is only loaded when the prod profile is active. Activate via spring.profiles.active=prod in properties, the SPRING_PROFILES_ACTIVE env var, or --spring.profiles.active=prod CLI arg.

Q10. How do you write a Spring Boot integration test?

A: @SpringBootTest starts a full application context. Add @AutoConfigureMockMvc to inject a MockMvc for HTTP-level assertions without a real server. Use @MockBean to replace infrastructure beans (e.g., database, email) with Mockito doubles.