Spring Boot Interview Questions and Answers

For 2–3 Years of Experience

Core Spring Boot (10 Questions)

1. What is Spring Boot's primary goal?

A: Simplify Spring app setup with auto-configuration, embedded servers, and opinionated defaults.

2. How does @SpringBootApplication work?

A: Combines @Configuration, @EnableAutoConfiguration, and @ComponentScan to bootstrap the app.

3. What are Spring Boot Starters?

A: Predefined dependency bundles (e.g., spring-boot-starter-web) to reduce manual dependency management.

4. How to override default Spring Boot configurations?

A: Define custom beans or properties in application.properties/application.yml.

5. Explain embedded servers in Spring Boot.

A: Servers like Tomcat/Jetty bundled within the app, so no external deployment is needed.

6. What is spring-boot-devtools?

A: A module for automatic app restarts and live reload during development.

7. How to exclude auto-configuration classes?

A: Use @EnableAutoConfiguration(exclude = {DataSourceAutoConfiguration.class}).

8. What is CommandLineRunner?

A: An interface to execute code after the app starts (e.g., run() method).

9. How to read custom properties in Spring Boot?

A: Use @Value("${property.key}") or @ConfigurationProperties for grouped properties.

10. What is the bootstrap.properties file?

A: Used in Spring Cloud Config to load external configuration before the main app starts.

REST APIs (8 Questions)

1. How to version a REST API in Spring Boot?

A: Use URL paths (e.g., /api/v1/users) or headers (Accept-Version).

2. What's the difference between @RestController and @Controller?

A: @RestController = @Controller + @ResponseBody (auto-serializes responses to JSON/XML).

3. How to validate request bodies in Spring Boot?

A: Use @Valid with @RequestBody and define constraints (e.g., @NotNull).

4. How to document APIs in Spring Boot?

A: Use Swagger/OpenAPI with springdoc-openapi dependency.

5. What is HATEOAS? How to implement it?

A: Hypermedia-driven APIs. Use spring-boot-starter-hateoas and EntityModel.

6. How to handle file uploads?

A: Use MultipartFile in a @PostMapping method.

7. How to return XML instead of JSON?

A: Add produces = MediaType.APPLICATION_XML_VALUE to the endpoint and include XML converters.

8. What is ResponseEntity?

A: A wrapper for HTTP responses (status code, headers, body).

Data Persistence (JPA/Hibernate) (10 Questions)

1. What is the difference between CrudRepository and JpaRepository?

A: CrudRepository = basic CRUD. JpaRepository adds JPA-specific features (e.g., flushing, batch ops).

2. How to create a custom query with Spring Data JPA?

A: Use @Query("JPQL") or method names (e.g., findByNameAndAge()).

3. What is the N+1 problem? How to fix it?

A: Too many queries due to lazy loading. Fix with JOIN FETCH in JPQL or @EntityGraph.

4. How to enable JPA auditing (e.g., createdDate)?

A: Use @EnableJpaAuditing and annotate fields with @CreatedDate, @LastModifiedDate.

5. What is @Transactional?

A: Defines a database transaction scope (ACID properties).

6. How to use a composite primary key?

A: Create a class with @Embeddable and use @EmbeddedId in the entity.

7. How to configure multiple data sources?

A: Define separate DataSource, EntityManager, and TransactionManager beans with @Primary.

8. What is PagingAndSortingRepository?

A: Extends CrudRepository to support pagination and sorting.

9. How to map entity inheritance hierarchies?

A: Use @Inheritance(strategy = InheritanceType.SINGLE_TABLE/JOINED/TABLE_PER_CLASS).

10. How to use native SQL queries?

A: Set nativeQuery = true in @Query.

Security (Spring Security) (5 Questions)

1. How do you secure a Spring Boot application?

A: Use Spring Security by adding the spring-boot-starter-security dependency, configuring HTTP security, and defining security rules.

2. What is the difference between @PreAuthorize and @Secured?

A: @PreAuthorize uses SpEL for method-level security, while @Secured is a simpler, role-based annotation.

3. How to implement JWT authentication?

A: Generate a token on successful authentication and validate it in subsequent requests by extracting the token from headers.

4. What is the role of AuthenticationManager?

A: It handles user authentication and delegates requests to the configured AuthenticationProvider.

5. How to handle CSRF protection in Spring Security?

A: By default, Spring Security enables CSRF protection. To disable it, configure HttpSecurity with .csrf().disable().

Microservices (Spring Cloud) (7 Questions)

1. What is Spring Cloud?

A: A framework providing tools for building distributed systems and microservices (e.g., service discovery, load balancing, config server).

2. What is Eureka? How does it work?

A: A service discovery tool for registering and locating microservices. Eureka Server acts as a registry, and clients use Eureka Client to discover services.

3. Explain API Gateway and how to implement it.

A: A single entry point for routing requests to various services. Implemented using Spring Cloud Gateway or Zuul.

4. What is Circuit Breaker pattern?

A: A resilience pattern preventing cascading failures by stopping requests to unresponsive services (e.g., Hystrix, Resilience4j).

5. How to implement centralized configuration?

A: Use Spring Cloud Config Server to externalize configuration files for all microservices.

6. What is Feign Client?

A: A declarative HTTP client for making REST calls between microservices, simplifying inter-service communication.

7. What is Sleuth and Zipkin?

A: Sleuth provides distributed tracing, while Zipkin is a tool for visualizing trace data and diagnosing latency issues.

Testing (JUnit, Mockito) (5 Questions)

1. What is the difference between JUnit 4 and JUnit 5?

A: JUnit 5 introduced modular architecture, better extension support, and annotations like @BeforeAll, @AfterAll.

2. How to mock dependencies using Mockito?

A: Use @Mock annotation or Mockito.mock() method and Mockito.when() to define mock behavior.

3. How to test REST controllers in Spring Boot?

A: Use MockMvc for unit testing controllers without starting the server.

4. What is the purpose of @MockBean in Spring Boot tests?

A: To provide mock implementations of dependencies for integration tests.

5. How to test Spring Data JPA repositories?

A: Use @DataJpaTest annotation to test only JPA components without starting the full context.

Best Practices and Performance Optimization (5 Questions)

1. How to reduce application startup time in Spring Boot?

A: Disable unused auto-configurations, use lazy initialization, and optimize bean creation.

2. What are the best practices for logging in Spring Boot?

A: Use SLF4J with Logback, apply logging levels properly, and utilize MDC for traceability.

3. How to handle exceptions globally?

A: Use @ControllerAdvice with @ExceptionHandler methods.

4. How to improve query performance with JPA? A: Use projections, DTOs, pagination, and batch fetching.

5. How to monitor Spring Boot applications?
A: Use Spring Boot Actuator and integrate with tools like Prometheus, Grafana, or ELK Stack.