Spring Boot Interview Questions and Answers

1. Basics of Spring Boot

Q1: What is Spring Boot and why is it used?

Answer: Spring Boot simplifies Java application development by providing auto-configuration, embedded servers, and production-ready features.

Q2: What are the main features and advantages of using Spring Boot for application development?

Answer:

- **Auto-configuration:** Spring Boot automatically configures Spring beans based on dependencies in the classpath, reducing boilerplate code.
- **Embedded Servers:** Comes with embedded servers like Tomcat, Jetty, or Undertow, eliminating the need to deploy WAR files.
- **Spring Boot Actuator:** Provides built-in production-ready features like health checks, metrics, and monitoring.
- **Starter Dependencies:** Simplifies dependency management with curated, versioned dependencies.
- **Spring Boot CLI:** Command-line interface for quickly running and testing applications.
- **Spring Boot DevTools:** Provides tools to improve development experience with auto-restart and live reload.

Q3: What are Spring Boot starters?

Answer: Spring Boot Starters are pre-configured Maven dependencies that simplify adding specific features to your Spring Boot application. When you include a starter, you get all its transitive dependencies automatically.

Examples:

- spring-boot-starter-web for web apps
- spring-boot-starter-security for security features

Q4: Explain the @SpringBootApplication annotation.

Answer: The @SpringBootApplication annotation combines three essential annotations:

- **@Configuration:** Marks the class as a source of bean definitions
- **@EnableAutoConfiguration:** Tells Spring Boot to auto-configure the application based on dependencies
- **@ComponentScan:** Tells Spring to scan the package and sub-packages for components

Q5: How to create a Spring Boot application?

Answer: There are several ways to create a Spring Boot application:

- Using the @SpringBootApplication annotation with SpringApplication.run()
- Using Spring Initializr (https://start.spring.io/)
- Using Spring Boot CLI with Groovy scripts
- Using Maven/Gradle by configuring the necessary dependencies and plugins

2. Spring Boot Configuration

Q6: How to configure Spring Boot using application.properties?

Answer: You can set properties like database configurations, logging settings, and server ports in application.properties.

Example:

```
server.port=8080
spring.datasource.url=jdbc:mysql://localhost:3306/mydb
```

Q7: What is the difference between application.properties and application.yml?

Answer: Both are used for configuration, but they differ in format:

- application.properties uses key-value pairs format, ideal for flat configurations
- application.yml uses YAML format with hierarchical structure, better for complex or nested configurations

Q8: How to externalize configuration in Spring Boot?

Answer: Configuration can be externalized through:

1. application.properties or application.yml files
2. Environment variables (SPRING_<PROPERTY_NAME>)
3. Command-line arguments
4. Profiles (application-dev.properties, application-prod.properties)
5. Config Server (for microservices)
6. Config Data API (from Spring Boot 2.4)

Q9: What is @Value used for?

Answer: The @Value annotation injects property values into Spring beans.

Example:

java
```
@Value("${company.maxEmployees}")
private int maxEmployees;
```

Q10: What are Spring Profiles and how do they work?

Answer: Spring Profiles define different configurations for different environments (dev, prod, test).

Example: `spring.profiles.active=dev`

3. Spring Boot RESTful Web Services

Q11: What is @RestController in Spring Boot?

Answer: @RestController combines @Controller and @ResponseBody, indicating that the class handles RESTful requests and returns data (JSON, XML) instead of views.

Q12: How to create a simple RESTful API using Spring Boot?

Answer: Create a @RestController class with appropriate mappings (@GetMapping, @PostMapping, etc.).

Q13: How to handle HTTP request methods like GET, POST, PUT, DELETE in Spring Boot?

Answer: Use @GetMapping, @PostMapping, @PutMapping, and @DeleteMapping annotations to map HTTP methods to handler methods.

Q14: What is the use of @RequestBody and @ResponseBody annotations?