P Mahendra

Spring Boot Important Questions and Answers

1. What is Spring Boot?

Spring Boot is an open-source Java-based framework used to create stand-alone, production-ready Spring applications with minimal configuration. It simplifies the development process by providing default configurations, embedded servers, and various dependencies.

2. How does Spring Boot differ from Spring Framework?

Feature	Spring Framework	Spring Boot
Configuration	Requires XML or Java-based configuration	Auto-configured with default settings
Embedded Server	Not included, requires setup	Comes with embedded servers (Tomcat, Jetty, Undertow)
Dependency Management	Manually managed	Uses Spring Boot Starter dependencies
Microservices Support	Requires manual setup	Provides built-in support

3. What are the key features of Spring Boot?

- **Auto-Configuration**: Automatically configures Spring applications based on dependencies.
- **Spring Boot Starters**: Pre-defined set of dependencies for different applications.
- **Embedded Server**: Includes Tomcat, Jetty, or Undertow without needing external setup.
- **Actuator**: Provides monitoring and management capabilities.
- **Spring Boot CLI**: Allows running Spring Boot applications from the command line.

4. What is Spring Boot Starter?

Spring Boot Starters are dependency descriptors that simplify the inclusion of related libraries in a project. Examples include:

- spring-boot-starter-web - For web applications with Spring MVC.
- spring-boot-starter-data-jpa - For database interaction using JPA.
- spring-boot-starter-security - For authentication and authorization.

5. What is Spring Boot Auto-Configuration?

Spring Boot Auto-Configuration automatically configures components based on classpath dependencies. It reduces manual configuration and provides ready-to-use beans.

6. What is the purpose of the application.properties or application.yml file?

These files are used to configure various aspects of a Spring Boot application, such as:

- Server port: server.port=8081
- Database connection: spring.datasource.url=jdbc:mysql://localhost:3306/mydb • Logging level: logging.level.org.springframework=DEBUG

7. What are Spring Boot Actuators?

Spring Boot Actuators provide production-ready features for monitoring and managing applications. Common actuator endpoints include:

- /actuator/health - Application health status.
- /actuator/info - General application information.
- /actuator/metrics - Performance metrics.

8. What is Spring Boot DevTools?

Spring Boot DevTools is a module that enhances the development experience by enabling:

- Automatic application restart when code changes.
- LiveReload support for UI updates.
- Configurable properties to disable caching.

9. What is the difference between @RestController and @Controller?

Annotation	**Purpose**
@Controller	Used for MVC controllers to return view templates.
@RestController	A combination of @Controller and @ResponseBody, used to return JSON responses.

10. How to handle exceptions in Spring Boot?

Spring Boot provides multiple ways to handle exceptions:

- **Using @ExceptionHandler**:

```
@ControllerAdvice
public class GlobalExceptionHandler {
    @ExceptionHandler(Exception.class)
    public ResponseEntity<String> handleException(Exception
        return new ResponseEntity<>(e.getMessage(), HttpStat
    }
}
```

- **Using @ResponseStatus**:

```
@ResponseStatus(HttpStatus.NOT_FOUND)
public class ResourceNotFoundException extends RuntimeExcept
    public ResourceNotFoundException(String message) {
        super(message);
    }
}
```

11. What is the difference between @SpringBootApplication and @EnableAutoConfiguration?

Annotation	**Description**
@SpringBootApplication	Combines @Configuration, @EnableAutoConfiguration, and @ComponentScan.
@EnableAutoConfiguration	Enables Spring Boot's auto-configuration mechanism.

12. How does Spring Boot integrate with databases?

Spring Boot supports multiple databases using:

- JDBC (spring-boot-starter-jdbc)
- JPA (spring-boot-starter-data-jpa with Hibernate)
- Spring Data MongoDB (spring-boot-starter-data-mongodb)

13. What is Spring Boot's default embedded server? How can we change it?

Spring Boot uses Tomcat as the default embedded server. It can be changed to Jetty or Undertow by excluding springboot-starter-tomcat and adding dependencies like:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jetty</artifactId>
</dependency>
```

14. How do you create a Spring Boot application?

- Using **Spring Initializr** (https://start.spring.io/)
- Using **Spring Boot CLI**: spring init --dependencies=web myapp
- Manually creating a Maven/Gradle project with required dependencies.

15. What are Spring Profiles?

Spring Profiles allow setting environment-specific configurations. Example:

- application-dev.properties for development.
- application-prod.properties for production. Activate a profile using:

  spring.profiles.active=dev

16. What is the purpose of @ConfigurationProperties?

It is used to map external properties to Java objects. Example:

```java
@Component
@ConfigurationProperties(prefix = "app")
public class AppConfig {
    private String name;
    private String version;
    // getters and setters
}
```

17. What is Spring Boot Security?

Spring Boot Security provides authentication and authorization. By default, it secures all endpoints with a generated password. To configure a custom username and password:

```
spring.security.user.name=admin
spring.security.user.password=admin123
```

18. How to enable CORS in Spring Boot?

Using @CrossOrigin annotation:

```
@RestController
@RequestMapping("/api")
@CrossOrigin(origins = "*")
public class MyController {
    @GetMapping("/data")
    public String getData() {
        return "Hello, World!";
    }
}
```

19. How do you schedule tasks in Spring Boot?

Using @Scheduled annotation:

```
@Scheduled(fixedRate = 5000)
public void scheduledTask() {
    System.out.println("Task executed every 5 seconds");
}
```

20. What are some common Spring Boot interview topics?

- Spring Boot vs. Spring Framework
- Microservices architecture
- Spring Boot Actuator
- Spring Boot Security
- Spring Boot REST APIs
- Exception handling
- Spring Boot caching

21. What is Spring Boot Actuator?

Spring Boot Actuator is a sub-project of Spring Boot that provides production-ready features

- Health checks.
- Metrics.
- Application info.
- Environment details.
  It helps monitor and manage the application in production.

22. Explain the internal working of Spring Boot.

Spring Boot works by automatically setting up default configurations based on the tools our project uses. It includes builtin servers like Tomcat to run our applications. Special starter packages make it easy to connect with other technologies. We can customize settings with simple annotations and properties files. The Spring Application class starts the app, and Spring Boot Actuator offers tools for monitoring and managing it.

23. What does the @SpringBootApplication annotation do internally?

The @SpringBootApplication annotation is a convenience annotation that combines @Configuration, @EnableAutoConfiguration, and @ComponentScan. This triggers Spring's auto-configuration mechanism to automatically configure the application based on its included dependencies, scans for Spring components, and sets up configuration classes

24) Is it possible to change the port of the embedded Tomcat server in Spring Boot?

Yes, we can change the default port of the embedded Tomcat server in Spring Boot. This can be done by setting the server.port property in the application.properties or application.yml file to the desired port number.

25) What is the default port of Tomcat in Spring Boot?

The default port for Tomcat in Spring Boot is 8080. This means when a Spring Boot application with an embedded Tomcat server is run, it will, by default, listen for HTTP requests on port 8080 unless configured otherwise

26) What are Spring Boot Actuator endpoints?

Spring Boot Actuator is like a toolbox for monitoring and managing our Spring Boot application. It gives us endpoints (think of them as special URLs) where we can check health, view configurations, gather metrics, and

more. It's super useful for keeping an eye on how your app is doing.

In a production environment (which is like the real world where your app is being used by people), these endpoints can reveal sensitive information about your application. Imagine leaving our diary open in a public place – we wouldn't want that, right? Similarly, we don't want just anyone peeking into the internals of your application.

27) How can we secure the actuator endpoints?

Limit Exposure: By default, not all actuator endpoints are exposed. We can control which ones are available over the web. It's like choosing what parts of your diary are okay to share.

Use Spring Security: We can configure Spring Security to require authentication for accessing actuator endpoints.

Use HTTPS instead of HTTP.

Actuator Role: Create a specific role, like ACTUATOR_ADMIN, and assign it to users who should have access. This is like giving a key to only trusted people.

28)What is Spring Profiles? How do you start an application with a certain profile?

Spring Profiles provide a way to segregate parts of our application configuration and make it only available in certain environments. For example, we can define database configurations for development, testing, and production environments without them interfering with each other. To start an application with a specific profile, we can use the Dspring.profiles.active=profile_name parameter in our command line when launching the application, or set the spring.profiles.active property in our application's configuration files.