## Top 10 Spring Boot Interview Questions and Answers [2025] by [Amigoscode](https://www.youtube.com/watch?v=cNzFPWuju8k)

1. **What is Spring Boot and how it differs from Spring Framework?**

+ Spring Framework is mainly about dependency injection. We had to configure everything ourselves in xml.

+ Spring Boot is an extension of the Spring Framework that simplifies application development by:

* + Auto-configuring the application based on dependencies
  + Providing embedded servers like Tomcat/Jetty
  + Eliminating the need for XML configuration
  + Including production-ready features like health checks and metrics

In contrast, the traditional Spring Framework requires manual configuration and external server deployment.

*• Basically we get a lot of pre-config out-of-the-box just by adding an annotation(@SpringBootApplication)*

1. **Advantages of Spring Boot**

* Faster development with auto-configuration
* Reduced boilerplate code
* Embedded servers for standalone apps
* Microservice-ready architecture
* Production-ready with Actuator
* Easy dependency management with spring-boot-starter-\* dependencies

1. **What is Spring Boot Starter?**

A starter is a set of convenient dependency descriptors you can include in your application. Example: spring-boot-starter-web bundles Spring MVC, Jackson, and embedded Tomcat.

*It simplifies dependency management by grouping commonly used libraries for specific functionalities*

1. **What is Auto Configuration in Spring Boot?**

Spring Boot uses @EnableAutoConfiguration (often included via @SpringBootApplication) to automatically configure beans based on the classpath and properties files.

For example, if spring-boot-starter-web is on the classpath, it auto-configures Spring MVC, DispatcherServlet, and Tomcat.

1. **Purpose of @SpringBootApplication annotation**

It’s basically a wrapper for 3 next important annotations:

* *@Configuration:* marks the class as the source of Bean definitions
* *@ComponentScan:* this is where to find and specify those Beans
* *@EnableAutoConfiguration*: tells Spring to auto-configure everything that is in the ‘classpath’

1. **How to create REST APIs with Spring Boot**

* Annotate the controller class with *@RestController*
* Use *@RequestMapping* or specific mappings like *@GetMapping*, *@PostMapping*, etc.

1. **What is Spring Boot Actuator?**

Provides production-ready features like:

* *Health checks (/actuator/health)*

If you have your application behind the load balancer, so it can keep on sending requests to this end-point and see if the application is healthy or not

* *Metrics (/actuator/metrics)*

CP usage, memory, etc

* *Info (/actuator/info)*
* *Thread dumps, env variables, beans, and more*

1. **How to handle exceptions in Spring Boot**

Basically the best way is to create your own GlobalExceptionHandler class, and annotate it with @ControllerAdvice. And each exception as methods, be annotated with @ExceptionHanlder. This so you can have full-control of the status codes that go back to the clients.

1. **What are Spring Profiles?**

Profiles allow you to define different configurations for different environments (dev, test, prod), for example:

○ In application-dev.properties: spring.datasource.url=...

○ Activate with: --spring.profiles.active=dev or via application.properties

1. **How does Spring Boot support externalized configuration?**

Spring Boot supports properties from different sources, like:

* + application.properties or application.yml
  + Command-line args (which override file configs)
  + Environment variables
  + Config server (Spring Cloud Config)

This allows flexible configuration without modifying code.