Spring boot interview Topic

**1.what is spring boot:**

Spring Boot is a framework built on top of the Spring Framework that simplifies the process of setting up, configuring, and running Java-based applications. It aims to reduce the complexity and boilerplate code commonly associated with traditional Spring applications.

**Use Case:**

**Faster Development:** With its auto-configuration and minimal setup, developers can focus more on writing application logic rather than configuration.

**No External Web Server:** Since it includes an embedded web server, you can run the application as a simple executable JAR, making deployment and testing easier.

**Active Community:** Spring Boot has an extensive and active community, which means regular updates, fixes, and support.

**Key Features:**

* Auto Configuration:
* Embedded Servers:
* Minimal Configuration:

**2. Spring Boot Setup**

* Creating a Spring Boot application from scratch
* Spring Initializr (https://start.spring.io) for creating projects
* Directory structure of a Spring Boot project

**3. Embedded Servers**

* How Spring Boot embeds web servers like Tomcat, Jetty, or Undertow
* Running Spring Boot applications with embedded servers
* Changing the server port and other configurations

**3.Spring Boot Auto Configuration**

* How Spring Boot auto-configures beans for you based on dependencies in the class path
* Using @EnableAutoConfiguration or @SpringBootApplication to enable auto configuration

**4.Spring Boot Annotations**

**1. @SpringBootApplication**

A combination of three annotations:

* @Configuration: Marks the class as a configuration class (like a Spring bean).
* @EnableAutoConfiguration: Tells Spring Boot to automatically configure your application based on dependencies.
* @ComponentScan: Scans for Spring components (like @Service, @Controller, etc.) in the same package or sub-packages.
* In short: @SpringBootApplication is used to mark the main class that runs the Spring Boot application.

**2. @RestController, @Controller, and @Service:**

@RestController: Used for building RESTful APIs. It combines @Controller and @ResponseBody (so methods return data directly, not views).

@Controller: Used for traditional web controllers that return views (e.g., HTML).

@Service: Marks a class as a service, typically used for business logic or service layer components.

In short: @RestController for REST APIs, @Controller for web views, and @Service for business logic.

**3. @Value, @ConfigurationProperties, and @Profile**

* @Value: Injects a value from properties or YAML files into a field (e.g., configuration properties).
* @Configuration Properties: Binds a group of properties to a Java class (e.g., multiple related config values).
* @Profile: Specifies that a bean is only available in a certain environment (e.g., @Profile("dev") for development)

**5. Spring Boot Dev-Tools**

Hot reloading for development

Enabling Spring Boot Dev Tools for faster feedback in development

Configuration of Dev Tools (e.g., live reload)

**6. Spring Boot Actuator**

* **Monitoring and Management**  
  How to use Spring Boot Actuator for application health checks, metrics, and logging.
* **Custom Endpoints**  
  Implementing custom endpoints for monitoring and management.

**7. Spring Boot Profiles**

* **Using Multiple Profiles for Different Environments**  
  Setting up and using profiles like dev, test, prod for different configurations.

**8. Spring Boot Testing**

* **Unit Testing with @SpringBootTest**  
  Writing unit tests for Spring Boot applications using annotations like @SpringBootTest.
* **Mocking Dependencies with Mockito**  
  Using Mockito to mock dependencies in tests.

**9. Spring Boot Data Access**

* **Spring Data JPA / Spring JDBC**  
  Implementing persistence using Spring Data JPA or Spring JDBC templates.
* **Repositories**  
  Using Crud Repository and JPA Repository for database operations.
* **Database Configuration**  
  Configuring databases and connection pools in Spring Boot.

**10. Spring Boot Security**

* **Securing REST APIs with Spring Security**  
  Configuring basic authentication, JWT, or OAuth in Spring Boot applications.
* **Custom Authentication and Authorization**  
  Implementing custom login or user roles and permissions.

**11. Spring Boot RESTful Services**

* **Building REST APIs**  
  Designing REST APIs with Spring Boot and using annotations like @RequestMapping, @GetMapping, etc.
* **Error Handling in REST APIs**  
  Using @ControllerAdvice to handle exceptions globally in REST APIs.

**. Spring Boot RESTful Services**

* **Building REST APIs**  
  Designing REST APIs with Spring Boot and using annotations like @RequestMapping, @GetMapping, etc.
* **Error Handling in REST APIs**  
  Using @ControllerAdvice to handle exceptions globally in REST APIs.

**12. Spring Boot Integration**

* **Message Brokers (e.g., RabbitMQ, Kafka)**  
  Integrating with messaging queues for asynchronous processing.
* **External API Integration**  
  Calling external APIs using Rest Template or Web-Client.

**13. Spring Boot Microservices**

* **Building Microservices with Spring Boot**  
  How to build and deploy a microservices architecture using Spring Boot.
* **Spring Cloud**  
  Use Spring Cloud for service discovery (Eureka), configuration management (Config Server), and API Gateway (Zuul).

**14. Spring Boot Profiles and Deployment**

* **Deploying Spring Boot Applications**  
  Creating executable JARs/WARs with Spring Boot and deploying to servers or cloud platforms.
* **Spring Boot with Docker**  
  Containerizing Spring Boot applications using Docker.

**15. Spring Boot Logging**

* **Logging Mechanisms in Spring Boot**  
  Integrating with logging frameworks like SLF4J, Log-back, and customizing logs in Spring Boot.