26. What is Spring Boot and why is it popular in microservices?

Spring Boot is a framework that simplifies the development of Spring-based applications. It provides auto-configuration, embedded servers, and production-ready features like health checks.
Why it's popular:
- Rapid development
- Embedded Tomcat
- Easy integration with Spring Cloud
- Simplified dependency management

27. What is Spring Cloud and how does it support microservices?

Spring Cloud provides tools for developers to quickly build some of the common patterns in distributed systems (e.g., configuration management, service discovery, circuit breakers, routing).
Examples: Eureka, Config Server, Zuul, Spring Cloud Gateway, Hystrix

28. How does Spring Boot help in creating RESTful APIs?

Spring Boot uses @RestController and @RequestMapping annotations to expose endpoints easily. Built-in JSON support via Jackson and customizable responses make REST API development very simple.

29. What is @RestController in Spring Boot?

@RestController is a convenience annotation that combines @Controller and @ResponseBody. It is used to create RESTful web services.

30. What is the use of @SpringBootApplication annotation?

It is a combination of @Configuration, @EnableAutoConfiguration, and @ComponentScan. It sets up the Spring context and application entry point.

31. What is embedded Tomcat in Spring Boot?

Spring Boot includes an embedded Tomcat server, so there's no need to deploy WAR files to external servers. You can run your app with `java -jar`.

32. What is application.properties or application.yml?

These files are used to define application configuration such as port, DB details, and custom properties in key-value format.

33. What is Spring Boot Actuator?

It provides production-ready features like health checks, metrics, info, environment details, etc. Available at endpoints like /actuator/health.

34. How does Spring Boot handle logging?

It uses Logback by default and can be configured via application.properties or logback.xml. Logs can be customized for levels and output format.

35. What is Spring Boot DevTools?

It enables hot swapping, live reload, and configurations to ease development. Automatically restarts application when code changes.

36. How do you handle exceptions in Spring Boot?

Use @ControllerAdvice along with @ExceptionHandler to manage global or specific exception handling in a centralized way.

37. What is the difference between @Component, @Service, and @Repository?

All are Spring-managed components.
- @Component: Generic stereotype
- @Service: Business logic layer
- @Repository: Data access layer (adds exception translation)

38. What is dependency injection in Spring Boot?

It is the process of injecting required objects into a class via constructor or field, managed by Spring’s IoC container.

39. What is @Autowired in Spring Boot?

Used to inject bean dependencies automatically by Spring’s IoC container.

40. How do you connect Spring Boot with databases?

Use Spring Data JPA or JDBC. Configuration is done via application.properties. Define a model, repository, and Spring Boot does the rest.

41. What is Spring Data JPA?

A part of Spring Data project. It simplifies database access using Repository interfaces and generates SQL queries automatically.

42. What is @Entity and @Id in Spring Boot?

@Entity marks a class as a JPA entity.
@Id defines the primary key of the entity.

43. What is the role of @EnableDiscoveryClient?

It registers the service with a discovery server like Eureka. It enables service registration and discovery.

44. How do you define REST API paths in Spring Boot?

Using @RequestMapping, @GetMapping, @PostMapping annotations. Example: @GetMapping("/users")

45. How can we externalize configuration in Spring Boot?

Use Config Server from Spring Cloud to centralize config management. You can load properties from Git or file system.