### **Dependency Injection**

1. What is Dependency Injection (DI) in Spring Boot?
2. What are the different types of Dependency Injection in Spring Boot?
3. What is the difference between Constructor Injection and Setter Injection in Spring Boot?
4. Can you explain what Field Injection is in Spring Boot and its drawbacks?
5. What is the role of @Autowired annotation in Spring Boot, and how is it used?
6. What happens when Spring Boot cannot find a suitable bean to inject using @Autowired?
7. How does Spring Boot handle circular dependencies when using @Autowired?
8. What is the use of @Inject annotation in Spring Boot, and how does it differ from @Autowired?
9. How do you inject dependencies into Spring Boot beans using the @Bean method in a configuration class?
10. How do you use @Qualifier annotation in Spring Boot to resolve ambiguous bean injections?
11. What is the role of @Primary annotation in Spring Boot, and how does it resolve bean conflicts?
12. How would you use constructor injection over setter injection in Spring Boot for better immutability?
13. Can you explain the difference between **by-name** and **by-type** dependency injection in Spring?
14. How can you inject a bean into a Spring Boot @Configuration class?
15. What is the significance of @Configuration and how does it relate to dependency injection in Spring Boot?

### **Other Dependency Injection Questions**

1. How can you configure a custom bean definition in Spring Boot through Java configuration?
2. What is the role of the **ApplicationContext** in dependency injection in Spring Boot?
3. What is the difference between using @Component, @Service, @Repository, and @Controller in Spring Boot when performing dependency injection?
4. How does Spring Boot handle dependency injection in a multi-threaded environment (e.g., prototype scope beans)?

What is Spring Boot and how is it different from the Spring Framework?

1. What are the main features of Spring Boot?
2. What is Spring Boot Starter?
3. What is the significance of the @SpringBootApplication annotation?
4. How does Spring Boot's auto-configuration work?
5. What is the role of application.properties or application.yml in Spring Boot?
6. How do you configure a database in a Spring Boot application?
7. What is Spring Boot Actuator?
8. How do you create a REST API using Spring Boot?
9. What is Spring Boot’s @Value annotation used for?
10. Explain the difference between @Component, @Service, and @Repository in Spring Boot.
11. How does Spring Boot handle exception handling?
12. What is the Spring Boot DevTools, and how does it improve the developer experience?
13. What is the difference between @RequestParam, @PathVariable, and @RequestBody in Spring Boot?
14. What is the SpringApplication.run() method used for in Spring Boot?
15. What is Spring Boot’s embedded web server, and how does it work?
16. What is the role of @EnableAutoConfiguration in Spring Boot?
17. What is Spring Boot’s profile feature, and how do you use it?
18. How do you secure a Spring Boot application?
19. How do you externalize configuration in Spring Boot?
20. What is the purpose of @ConfigurationProperties in Spring Boot?
21. How does Spring Boot handle logging?
22. What is the use of @Bean annotation in Spring Boot?
23. How can you create a multi-module Spring Boot application?
24. What is the difference between @RestController and @Controller in Spring Boot?
25. What is the role of @RequestMapping in Spring Boot, and how does it differ from @GetMapping or @PostMapping?
26. What are the different ways to configure Spring Boot for different environments (e.g., development, production)?
27. How do you enable and use caching in a Spring Boot application?
28. What is the difference between @Entity and @Table annotations in Spring Boot with JPA?
29. What are Spring Boot profiles, and how do you use them for different environments?
30. How do you handle scheduled tasks in Spring Boot?
31. How do you perform unit testing in Spring Boot?
32. How do you set up and use Swagger for API documentation in Spring Boot?
33. What is Spring Boot’s support for asynchronous programming?
34. What is the difference between @EnableWebSecurity and @EnableGlobalMethodSecurity in Spring Security?
35. What is the spring-boot-starter-test and what libraries are included in it?
36. How can you monitor Spring Boot applications using Prometheus and Grafana?
37. What is the role of @Transactional in Spring Boot applications?
38. What is the difference between @EnableAutoConfiguration and @ImportAutoConfiguration in Spring Boot?
39. How do you configure Spring Boot to run with an external servlet container instead of the embedded one?
40. What are the common causes of memory leaks in Spring Boot applications, and how can you prevent them?
41. How does Spring Boot handle versioning of REST APIs?
42. What is the role of @SpringBootTest annotation in testing?
43. How can you configure Spring Boot to run in a specific profile (e.g., dev, prod)?
44. How can you deploy a Spring Boot application as a WAR file instead of a JAR file?
45. How do you integrate Spring Boot with an external authentication provider (e.g., OAuth2, LDAP)?
46. How can you limit the number of concurrent requests in a Spring Boot application?
47. What are the advantages of using Spring Boot over traditional Java web applications?
48. How do you handle file uploads and downloads in Spring Boot?
49. What is the purpose of the @EnableAsync annotation in Spring Boot, and how do you configure it?
50. How can you create a custom Spring Boot starter?
51. What are the different ways to secure a Spring Boot application (e.g., authentication, authorization)?
52. How do you configure a Spring Boot application to use multiple databases?
53. How do you implement and use Spring Boot’s @Scheduled annotation for periodic tasks?
54. What is the difference between @ResponseBody and @RestController in Spring Boot?
55. How do you manage transactions in Spring Boot with Spring Data JPA?
56. How do you configure Spring Boot to handle cross-origin resource sharing (CORS)?
57. What is the Spring Boot DevTools library, and how does it aid in development?
58. How does Spring Boot manage dependency versions, and how can you override them?
59. How do you implement custom error handling in a Spring Boot REST API?

### **Bean Scope**

1. What are the different types of bean scopes available in Spring and Spring Boot?
2. What is the default bean scope in Spring Boot? Explain its behavior.
3. What is the difference between Singleton and Prototype bean scopes?
4. How do you define a bean with @Scope annotation in Spring Boot?
5. How does the request scope work in Spring Boot, and when would you use it?
6. What is the session scope in Spring Boot, and how is it used in web applications?
7. Can you explain the difference between @Scope("singleton") and @Scope("prototype") in Spring Boot?
8. How does Spring Boot handle the lifecycle of a @Scope("singleton") bean?
9. What is the application scope in Spring and when should it be used?
10. What is the globalSession scope in Spring Boot, and in what kind of applications would it be used?

**Interview Question:**

**Wipro-**

* 1. We have @requestmapping so why we use getmapping and putmapping,detelemapping.
  2. What is hibernate JPA and findall and findbyid which query internally excuted.
  3. What is api gateway and what is webclient explain configuration
  4. How to communicate two services.
  5. Hashmap internal working
  6. Hashset internal working
  7. **Palindrom number**

int num = 121;

int rev = 0, rem, original;

original = num;

while (num != 0) {rem = num % 10;

rev = rev \* 10 + rem;

num /= 10;}

* 1. Difference between Inner join and outer join.

9.) Difference b/w concurrent collection and synchronized collection

10.) Fail fast and fail safe and what is use of copyonwritearraylist

11.) Which security use your project explain and to authenticate

12.) How to handle globle exception which annotation use

13.) java 8 features

14.) why we use @functionalinterface

15.) Bean scope and what is default scope

16.) Explain factory design pattern.

17.) **find maximam {34,45,56,67} java8**

List<Integer> list = Arrays.*asList*(34,45,56,67);

Optional<Integer> max = list.stream().max(Integer::compareTo);

18.) **I have employee class name, salary so I want to sorted salary**

1. List<EmpSortsal> em = Arrays.*asList*(e1,e2,e3,e4);

List<EmpSortsal> sortBySal = em.stream().sorted(Comparator.*comparing*(EmpSortsal :: getSalary)).collect(Collectors.*toList*());

System.*out*.println(sortBySal);

1. Comparator<EmpSortsal> emSal = (o1, o2) -> o1.getSalary().compareTo(o2.getSalary())
2. List<EmpSortsal> sortBySal = em.stream().sorted(emSal).toList();

19.) **Second highest number (1, 5, 3, 9, 7, 5, 9); and remove duplicates**

List<integer> numbers = Arrays.asList(**1, 5, 3, 9, 7, 5, 9**);

20) what is stream api.

1. Synchronization in thread
2. What is stream and suppose that we have arraylist and linked list so why we need stream
3. What is lambda expression.
4. What is functional interface
5. Circuit breker and fault tolerance (I have one api and we have deploy in our server and this api first time fail and second time give the response)
6. Spring profile
7. @bean
8. @component and @service
9. @restcontroller and @controller
10. Slf 4j and lockback detail and slf4j logging levels
11. Daemon threads
12. Race condition
13. How to create Custom annotation
14. Comparator java 8 (default order)
15. Application.properties and application.yml and which one is prefer
16. Inside the Application.yml file tags so how to use service and controller-@value
17. What is saga design pattern
18. Explain Junit test cases
19. What is multithreding why we use multithreding and where use in your project.
20. What is yield method and sleep method
21. Explain Api gateway
22. VAPT security
23. Solid principal
24. Serial version in serialization
25. JWT token
26. Encryption and how to encrypt using 256
27. Create rest api
28. Suppose that I have employee table and in employee table I have huge data so how to fix using jpa
29. Store processor jpa write code
30. Bean lifecycle