94 Spring Boot interview questions to hire top engineers

Questions

1. What is Spring Boot, and why do developers use it?

2. Can you explain the concept of 'auto-configuration' in Spring Boot?

3. What are Spring Boot Starters, and how do they simplify dependency management?

4. How would you create a simple 'Hello, World!' application using Spring Boot?

5. What is the purpose of the `@SpringBootApplication` annotation?

6. Explain what an embedded server is in the context of Spring Boot.

7. How do you configure different environments (e.g., development, production) in a Spring Boot application?

8. What are some common Spring Boot annotations you've used or heard of?

9. How does Spring Boot handle exception handling?

10. Can you describe how to access properties defined in `application.properties` or `application.yml`?

11. What is the difference between `@RestController` and `@Controller` in Spring Boot?

12. How can you perform database operations using Spring Boot and Spring Data JPA?

13. What is a Spring Bean, and how are beans managed in a Spring Boot application?

14. Explain how Spring Boot simplifies the process of creating RESTful APIs.

15. What are some advantages of using Spring Boot over traditional Spring development?

16. How would you deploy a Spring Boot application to a cloud platform like AWS or Azure?

17. What is Spring Data REST, and how does it help in building RESTful services?

18. Explain how Spring Boot handles security.

19. How do you add logging to a Spring Boot application?

20. What are Spring Boot Actuators, and what kind of information do they provide?

21. How can you test a Spring Boot application?

22. If you have a Spring Boot application that needs to interact with external APIs, how would you approach that?

23. What is Spring Boot and why do developers use it?

24. Can you explain the concept of 'auto-configuration' in Spring Boot?

25. What are Spring Boot starters and how do they simplify project setup?

26. How would you create a simple REST API endpoint using Spring Boot?

27. What is the purpose of the `@SpringBootApplication` annotation?

28. Explain what a dependency injection is and how Spring Boot helps in achieving that?

29. What is an embedded server in Spring Boot? Why is it useful?

30. How can you configure different environments (dev, prod) in Spring Boot?

31. What are some common Spring Boot annotations you've used?

32. How do you handle exceptions in a Spring Boot application?

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

34. Explain how to connect to a database using Spring Boot.

35. What is Spring Data JPA? How does it simplify database operations?

36. How do you test a Spring Boot application? What types of tests can you write?

37. What are some advantages of using Spring Boot over traditional Spring?

38. How can you add custom dependencies to your Spring Boot project?

39. What is the role of a `pom.xml` file in a Spring Boot project?

40. Explain the concept of Spring Beans and how they are managed in Spring Boot.

41. How would you secure a Spring Boot API endpoint?

42. What are Spring Boot Actuators? What kind of information do they expose?

43. How do you handle logging in a Spring Boot application?

44. What is a message queue, and how could you use one in a Spring Boot application?

45. If you had a slow-running API endpoint, how would you troubleshoot it using Spring Boot's tools?

46. How would you deploy a Spring Boot application to a cloud environment?

47. What is the difference between `@RestController` and `@Controller` in Spring Boot?

48. Explain how Spring Boot simplifies the creation of microservices.

49. How does Spring Boot simplify the process of creating scheduled tasks?

50. How does Spring Boot handle auto-configuration, and what are some ways to customize or disable it?

51. Explain Spring Boot's actuator module and its benefits for monitoring and managing applications.

52. Describe how you would implement exception handling in a Spring Boot REST API, including global exception handling.

53. What are Spring Boot starters, and how do they simplify dependency management in a Spring Boot project?

54. How would you configure different environments (e.g., development, testing, production) in a Spring Boot application?

55. Explain how Spring Boot integrates with databases using Spring Data JPA. Include details on repository interfaces.

56. Describe the process of securing a Spring Boot application with Spring Security, including authentication and authorization.

57. How would you implement caching in a Spring Boot application, and what are some of the caching providers you can use?

58. Explain how to use Spring Boot's testing support to write unit and integration tests for your application.

59. Describe how you would configure logging in a Spring Boot application, including setting log levels and output formats.

60. How can you use Spring Boot to create and consume RESTful web services?

61. Explain how Spring Boot handles transaction management, and how you can use annotations to control transactions.

62. What are some best practices for deploying a Spring Boot application to a cloud environment like AWS or Azure?

63. How does Spring Boot support asynchronous processing, and how can you use annotations like @Async?

64. Describe the process of creating a custom Spring Boot starter.

65. Explain how Spring Boot simplifies the process of working with message queues like RabbitMQ or Kafka.

66. How would you monitor your Spring Boot application in production, including metrics and health checks?

67. Describe how you would configure and use Spring Boot's support for internationalization (i18n) and localization (l10n).

68. Explain how to use Spring Boot with different types of databases (e.g., relational, NoSQL).

69. How would you handle file uploads and downloads in a Spring Boot application?

70. Describe how Spring Boot supports building reactive applications with Spring WebFlux.

71. How does Spring Boot's auto-configuration work internally, and how can you customize or disable specific auto-configurations?

72. Explain your experience with Spring Boot Actuator. What custom metrics have you implemented, and how did you secure the Actuator endpoints?

73. Describe a scenario where you used Spring Boot's embedded database support (e.g., H2). What were the advantages and disadvantages compared to using an external database?

74. How have you implemented caching in a Spring Boot application? What caching providers have you used (e.g., Redis, Caffeine), and what are the trade-offs?

75. Explain how you have handled asynchronous tasks in Spring Boot. What are the pros and cons of using @Async vs. a message queue like RabbitMQ or Kafka?

76. Describe your experience with Spring Boot's testing framework. What types of tests have you written (e.g., unit, integration, end-to-end), and how did you handle testing external dependencies?

77. How does Spring Boot simplify the process of creating RESTful APIs? What are some best practices for designing and implementing REST APIs with Spring Boot?

78. Explain how you've handled security in a Spring Boot application. What authentication and authorization mechanisms have you used (e.g., OAuth2, JWT), and how did you protect against common web vulnerabilities?

79. How have you used Spring Boot's dependency injection features in complex applications? Can you provide an example of using constructor injection, setter injection, and field injection?

80. Describe your experience with Spring Boot's data access features. What ORM frameworks have you used (e.g., JPA, Hibernate), and how did you optimize database queries?

81. How have you monitored and managed Spring Boot applications in production? What tools have you used for logging, metrics, and alerting?

82. Explain how you have used Spring Boot's configuration management features. How did you handle different environments (e.g., development, testing, production), and how did you externalize configuration?

83. How would you implement a custom health indicator in Spring Boot Actuator to monitor a specific aspect of your application?

84. Describe a situation where you needed to create a custom Spring Boot starter. What were the components you included in the starter, and how did you make it configurable?

85. How do you handle transactions in Spring Boot, especially when dealing with multiple data sources or microservices?

86. Explain how you would implement rate limiting in a Spring Boot application to protect against abuse.

87. Describe a scenario where you used Spring Boot's `CommandLineRunner` or `ApplicationRunner` interfaces. What kind of tasks did you perform during application startup?

88. How have you used Spring Boot's profiles feature to manage different application configurations for various environments?

89. Explain how you would implement a background job scheduler in a Spring Boot application. What are the pros and cons of using @Scheduled vs. a more robust scheduler like Quartz?

90. How have you handled versioning of REST APIs in a Spring Boot application? What strategies did you use (e.g., URI versioning, header versioning), and what are the trade-offs?

91. Describe a situation where you used Spring Boot's event publishing and listening mechanism. What types of events did you publish, and how did you handle them asynchronously?

92. How do you approach troubleshooting performance issues in Spring Boot applications? What tools and techniques do you use to identify bottlenecks?