Spring Boot Microservices Interview Questions and Answers

Author: Ramesh Fadatare
Interview Microservices Spring Boot

In this blog post, we will discuss some commonly asked interview questions related to Spring Boot Microservices along with their answers.

1. What are microservices?

Microservices is an architectural style that structures an application as a collection of small, loosely coupled, and independently deployable services. Each service represents a specific business capability and can be developed, deployed, and scaled independently.

2. What is Spring Boot?

Spring Boot is a framework built on top of the Spring framework that simplifies the development of Java applications, including microservices. It provides a convention-over-configuration approach, auto-configuration, and embedded servers, allowing developers to quickly create production-ready microservices with minimal setup.

3. What are the advantages of using Spring Boot for microservices?

Simplified development: Spring Boot provides a range of features and defaults, reducing the amount of code and configuration required to develop microservices.

Auto-configuration: It automatically configures various components based on classpath dependencies, reducing the need for manual configuration.

Embedded servers: Spring Boot includes embedded servers like Tomcat, Jetty, or Undertow, making it easy to deploy microservices as standalone JAR files.

Cloud-native support: Spring Boot integrates well with cloud platforms and provides support for building cloud-native microservices.

4. How do you communicate between microservices in Spring Boot?

In Spring Boot, microservices can communicate with each other using various mechanisms:

Synchronous HTTP/REST: Microservices can communicate via HTTP using RESTful APIs, exchanging data in JSON or XML formats. For example, Spring Boot provides RestTamplete, and WebClient classes to make REST API calls from one microservice to another microservice.

Messaging: Asynchronous communication can be achieved using message queues or message brokers like RabbitMQ or Apache Kafka.

Service Discovery: Microservices can use service discovery mechanisms like Netflix Eureka or Spring Cloud Consul to locate and communicate with other services dynamically.

5. What are service registration and discovery in microservices?

Service registration and discovery is a mechanism that allows microservices to register themselves with a service registry and discover other services. Service registries maintain a registry of available services, including their network locations and metadata. Service discovery clients can query the registry to obtain the necessary information to communicate with other services.

6. How does Spring Cloud help with building microservices?

Spring Cloud is a framework that provides a set of tools and libraries to simplify the development of microservices using Spring Boot. It offers features like service discovery, client-side load balancing, distributed tracing, circuit breakers, and more. Spring Cloud integrates with other popular technologies like Netflix OSS components and provides a consistent and opinionated approach to building microservices.

7. What is circuit breaking in microservices and how is it implemented in Spring Boot?

Circuit breaking is a design pattern used to handle and prevent cascading failures in microservice architectures. It involves monitoring the calls to external services and, if a certain threshold of failures is reached, tripping a circuit breaker to stop further requests and return a fallback response.

In Spring Boot, circuit breaking can be implemented using libraries like Netflix Hystrix or Resilience4j, which provide annotations and configuration options to define circuit breakers and fallback behaviors.

8. How do you ensure data consistency across multiple microservices?

Ensuring data consistency in a distributed microservices environment can be challenging. Some common approaches include:

Using the Saga pattern: A Saga is a sequence of local transactions that together form a larger, coordinated transaction. Sagas help maintain consistency by using compensating transactions to undo previous actions if a failure occurs.

Applying event-driven architecture: Services can communicate through events, and changes in one service can be propagated asynchronously to other services, ensuring eventual consistency.

Leveraging distributed transactions: Although not recommended in all scenarios, distributed transactions can be used to maintain data consistency across multiple microservices. Technologies like JTA (Java Transaction API) can be used in such cases.

9. What is API Gateway in microservices and how does it help?

An API Gateway is a service that acts as an entry point for all client requests in a microservices architecture. It sits between the clients and the microservices, routing requests, handling authentication/authorization, enforcing security, and providing additional cross-cutting concerns like rate limiting, logging, and monitoring. API Gateways centralize these concerns, simplifying the clients' interactions with the microservices.

10. What are the challenges of testing microservices in Spring Boot?

Testing microservices can be challenging due to their distributed nature.

Some common challenges include:

- Setting up and managing test environments with multiple microservices and their dependencies. Orchestrating integration tests involving multiple microservices and ensuring their dependencies are available.
- Testing scenarios involving eventual consistency and dealing with network-related failures.
- Managing test data and ensuring data isolation during testing.

11. What is Spring Cloud Circuit Breaker?

Inter-service Communication is common in the Microservice architecture, if one of the services is down, the other service which is communicating with it should be able to handle this failure gracefully.

Spring Cloud provides **Spring Cloud Circuit Breaker** module to implement the Circuit Breaker pattern in the Microservices project.

Spring Cloud Circuit breaker provides an abstraction across different circuit breaker implementations. It provides a consistent API to use in your applications allowing you the developer to choose the circuit breaker implementation that best fits your needs for your app.

Supported Implementations

- Resilience4J
- Spring Retry

Refer official page to know more: Spring Cloud Circuit Breaker

Watch my YouTube video to understand how Circuit Breaker Pattern works:

12. What is Spring Cloud Bus?

This module contains a lightweight message broker implementation, which is mainly used to broadcast some messages to the other services

Refer official page to know more: Spring Cloud Bus

13. What is Spring Cloud Stream?

This module mainly allows us to implement asynchronous communication between our microservices using event-driven architecture.

We can use Apache Kafka or RabbitMQ as a message broker to implement event-driven microservices.

Refer official page to know more: Spring Cloud Stream

14. What is Spring Cloud OpenFeign?

Feign is a declarative web service client. It makes writing web service clients easier.

To use Feign create an interface and annotate it. It has pluggable annotation support including Feign annotations and JAX-RS annotations.

Feign also supports pluggable encoders and decoders. Spring Cloud adds support for Spring MVC annotations and for using the same HttpMessageConverters used by default in Spring Web.

Spring Cloud integrates Eureka, as well as Spring Cloud LoadBalancer to provide a load-balanced HTTP client when using Feign.

Check out the complete example: Spring Cloud OpenFeign

Conclusion

In this blog post, we discussed some commonly asked interview questions related to Spring Boot Microservices along with their answers. Also, check out Spring Cloud Interview Questions