Spring Boot and Microservices Roadmap for Beginners [2025 Update]

Ramesh Fadatare Follow 6 min read · Feb 18, 2025

In this article, let's look at the roadmap for learning Spring Boot and Microservices. This roadmap is for beginners who want to learn to build Restful web services and microservices using Spring Boot.

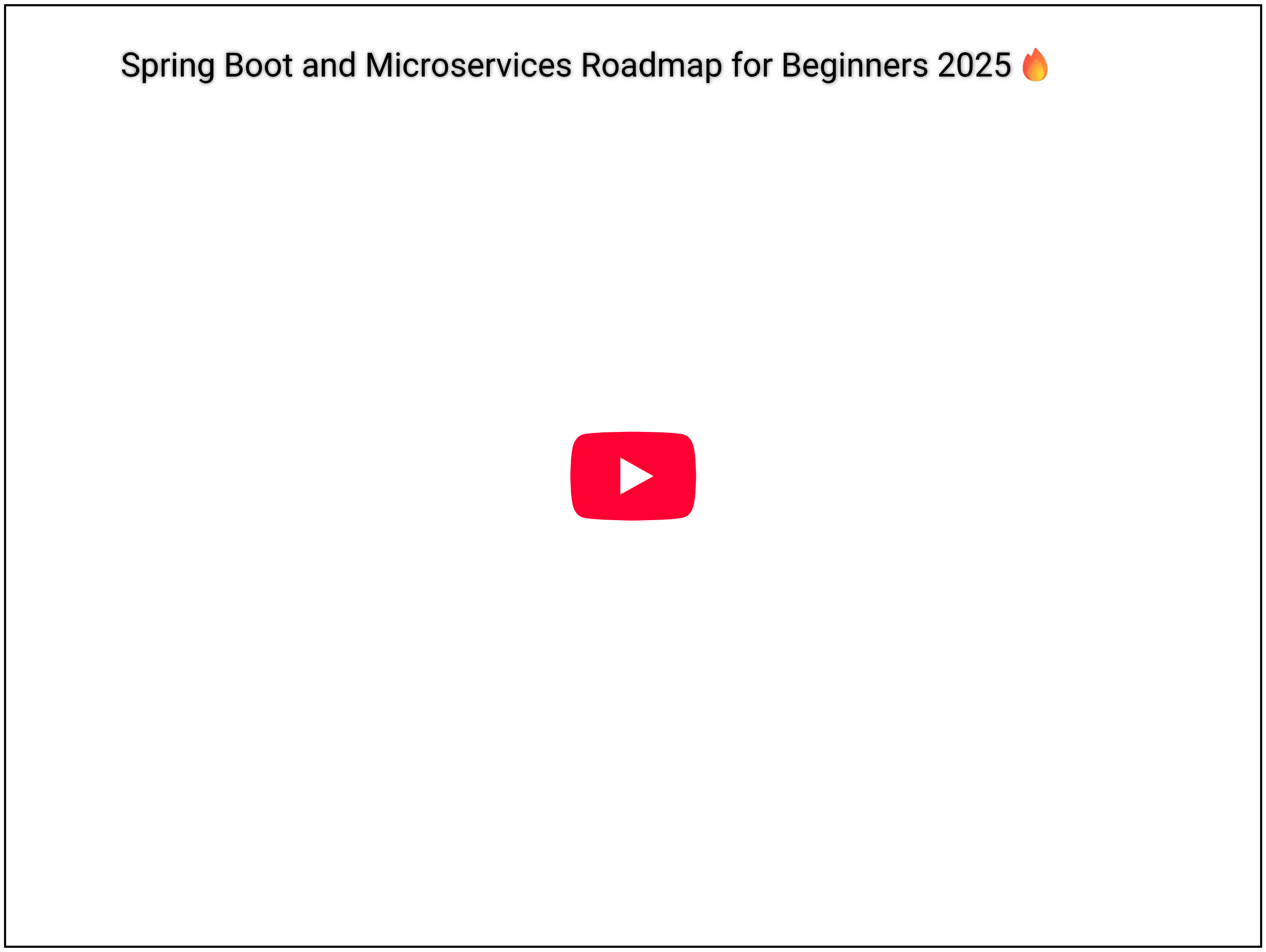
> *Check out my top Udemy course:* ***Building Microservices with Spring Boot and Spring Cloud****.*

Note: This roadmap is my personal opinion based on my experience with Spring Boot and Microservices, but there could be other opinions from different developers.

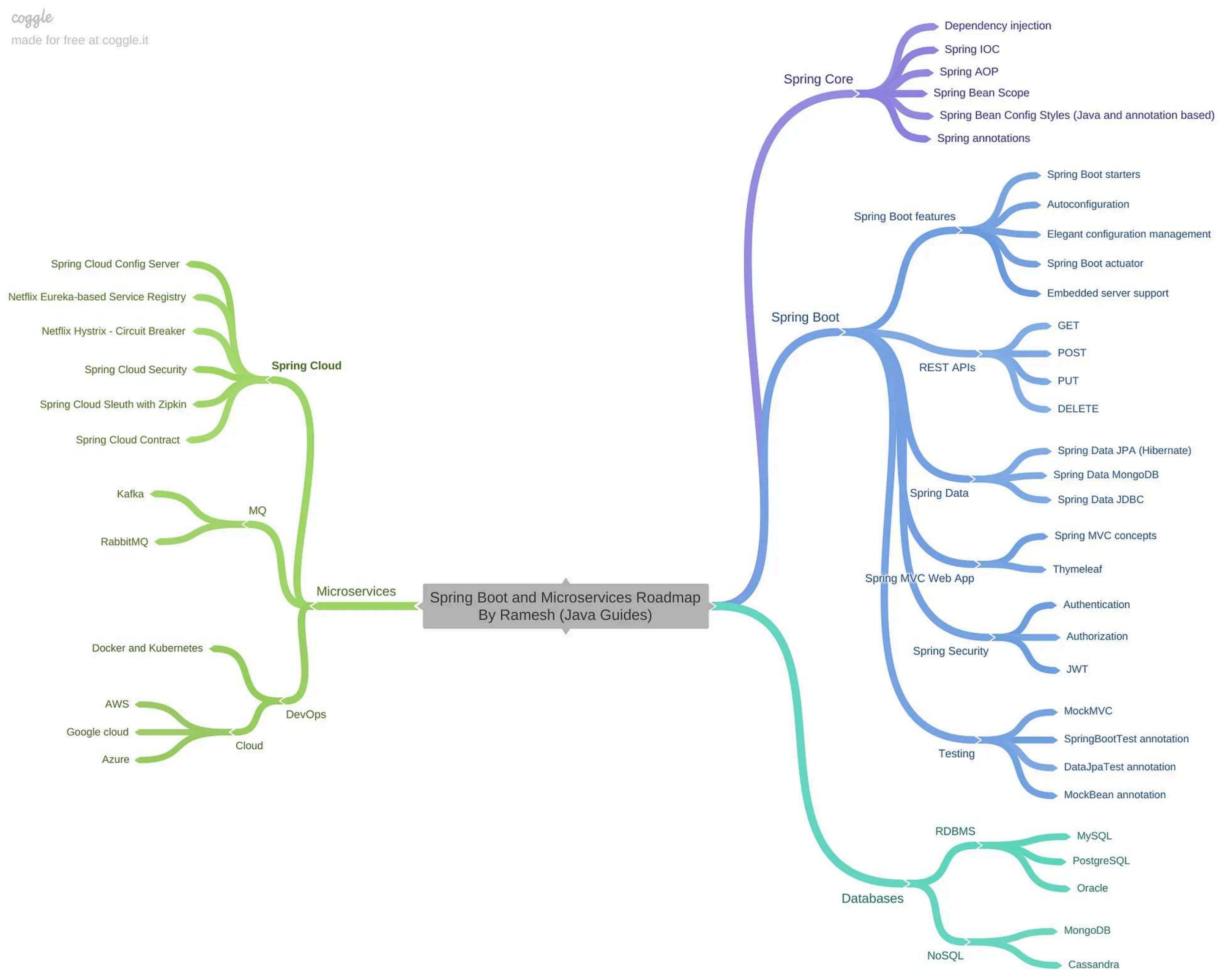
Spring Boot is a very popular Java framework for building Restful web services and microservices.

Nowadays, MicroServices is the hot buzzword in software development, and many organizations prefer building their enterprise applications using MicroServices architecture. In the Java community, SpringBoot is the most widely used framework for building both monoliths and microservices.

YouTube Video



Spring Boot and Microservices Roadmap — Cheat Sheet 2025



1. Learn Spring Core

I have seen many times that people (especially beginners) directly jump on to SpringBoot without having any prior Spring knowledge, and everything looks magical. It works great as long as you go with the defaults, and the moment you need to customize it, it feels complex. So I highly recommend that before learning Spring Boot directly, you can learn Spring core fundamentals such as:

- Dependency injection
- Spring IOC
- Spring AOP
- Spring beans
- Spring bean life cycle
- Spring configuration styles (XML, Java-based, and annotation-based)
- Learn important annotations

By learning Spring Core, you will be able to understand how to use DI, IOC, and Spring beans in Spring-based applications.

Spring 6 and Spring Boot 3 for Beginners (Includes 7 Projects) — This is my project-oriented and bestseller course on Udemy.

In this course, you will learn Spring Core, Spring Boot 3, REST API, Spring MVC, WebFlux, Spring Security, Spring Data JPA, Docker, Thymeleaf & Building Projects. **Build 7 Spring Boot Real-Time Projects.**

2. Learn Spring Boot

2.1 Learn Spring Boot features

Next, start learning Spring Boot. You can begin by learning Spring boot features such as:

- Spring Boot starters
- Spring Boot autoconfiguration
- Elegant configuration management

- Spring Boot Actuator
- Easy-to-use embedded servlet container support

2.2 Learn Building REST APIs

Learn how to build REST APIs using Spring boot:

- GET
- POST
- PUT
- DELETE

Check out my free course to learn to build REST APIs using Spring Boot at **Spring Boot Restful Web Services Tutorial | Full Course**

2.3 Repository/DAO Layer

Next, learn how to build a repository/DAO layer in the Spring Boot application.

The goal of the Spring Data module is to significantly reduce the amount of boilerplate code required to implement data access layers for various persistence stores.

Spring Data provides different modules to work on different persistence stores:

- For JPA — **Spring Data JPA**
- For JDBC — **Spring Data JDBC**

- For MongoDB — **Spring Data MongoDB**

By learning Spring Data JPA, you can develop a repository/DAO layer without writing boilerplate code.

2.4 Spring MVC Web Application

To develop a Spring MVC web application, you need to know Spring MVC and how Spring MVC works internally.

To develop a view layer in the Spring MVC web application, I am going to suggest you use the Thymeleaf template. You can also use JSP, but JSP is something line outdated, and nowadays, developers use Thymeleaf to develop a view layer in the Spring MVC web application.

To learn about thymeleaf, check out **my thymeleaf crash course**.

2.5 Learn Spring Security

Spring Security is a framework that provides authentication, authorization, and protection against common attacks.

Spring security is a de facto standard for securing web applications and Restful web services in Java.

Also, learn JWT for token-based authentication to secure REST APIs.

By Learning Spring security, you will be able to secure web applications and REST APIs.

2.6 Testing Spring Boot Application

Learn the JUnit framework and Mokito for unit testing in Java. As a Java programmer, you should know how to test your core logic using the JUnit framework.

Spring boot provides @SpringBootTest and @DataJpaTest annotations to test controllers and repositories.

> To learn and master the JUnit framework at **JUnit tutorials**

Check out my popular Udemy course to **learn about spring boot application testing**.

3. Databases

As a Java programmer, you should have good knowledge of databases.

Here are the commonly used relational databases and No SQL databases:

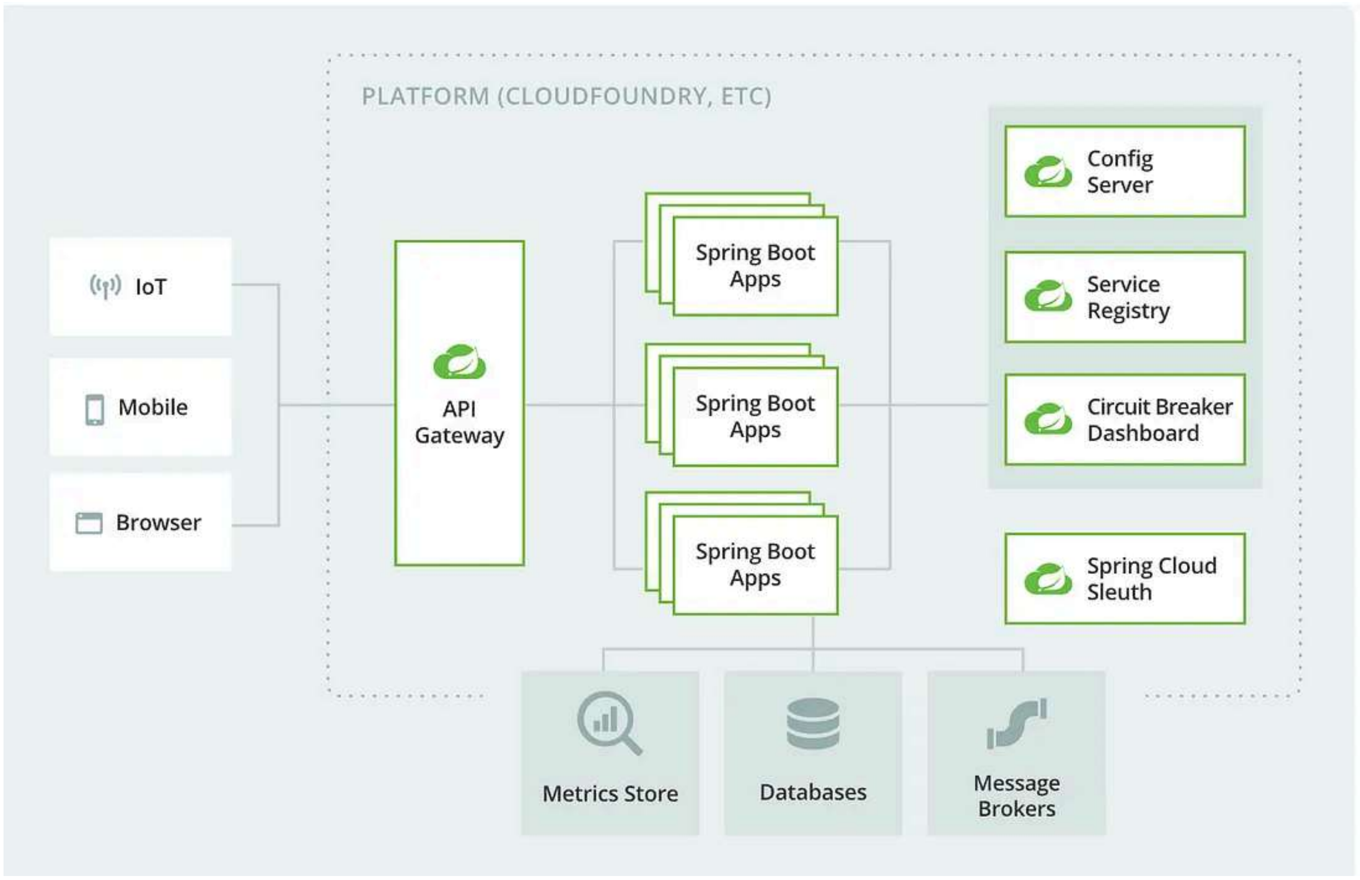
3.1 RDBMS:

- MySQL
- PostgreSQL
- MS-SQL server
- Oracle

3.2 No SQL Databases:

- MongoDB
- Cassandra

4. Microservices



Spring Boot and Spring Cloud are a great combination for developing microservices in the Java community.

Spring Boot is the most popular and widely used Java framework for building MicroServices. Spring Boot makes it easy to create standalone, production-grade Spring-based Applications that you can “just run.”

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, intelligent routing, micro-proxy, control bus, one-time tokens, global locks, leadership election, distributed sessions, cluster state).

4.1 Spring Cloud Modules

Following are just a few of the Spring Cloud modules that can be used to address distributed application concerns:

Spring Cloud Zuul Proxy:

Spring Cloud provides a Zuul proxy, similar to **Nginx**, that can be used to create an API Gateway.

API Gateway, aka Edge Service, provides a unified interface for a set of microservices so that clients do not need to know about all the details of microservices internals.

Spring Cloud Config Server:

To externalize the configuration of applications in a central config server with the ability to update the configuration values without requiring to restart of the applications. We can use **Spring Cloud Config Server** with git or Consul or ZooKeeper as a config repository.

Service Registry and Discovery:

As there could be many services and we need the ability to scale up or down dynamically, we need Service Registry and Discovery mechanism so that service-to-service communication should not depend on hard-coded hostnames and port numbers. Spring Cloud provides Netflix Eureka-based Service Registry and Discovery support with just minimal configuration. We can also use Consul or ZooKeeper for Service Registry and Discovery.

Circuit Breaker:

In microservices-based architecture, one service might depend on another service, and if one service goes down, then failures may cascade to other services as well. Spring Cloud provides Netflix Hystrix-based Circuit Breaker to handle these kinds of issues.

Spring Cloud Data Streams:

These days, we may need to work with huge volumes of data streams using Kafka or Spark, etc. Spring Cloud Data Streams provides higher-level abstractions to use those frameworks in an easier manner.

Spring Cloud Security:

Some of the microservices need to be accessible to authenticated users only, and most likely, we might want a Single Sign-On feature to propagate the authentication context across services. Spring Cloud Security provides authentication services using OAuth2.

Distributed Tracing:

One of the pain points with microservices is the ability to debug issues. One simple end-user action might trigger a chain of microservice calls, there should be a mechanism to trace the related call chains. We can use Spring Cloud Sleuth with Zipkin to trace the cross-service invocations.

These are just a few of Spring Cloud's features. To explore more, visit https://projects.spring.io/spring-cloud/.

4.2 MQ (Message Queues) — RabbitMQ or Apache Kafka

Spring Cloud Stream is a framework built on top of Spring Boot and Spring Integration that helps in creating **event-driven** or **message-driven** microservices.

Communication between endpoints is driven by messaging-middleware parties like **RabbitMQ** or **Apache Kafka**. Services communicate by publishing domain events via these endpoints or channels.

RabbitMQ and **Apache Kafka** are common messaging brokers that microservices can use to communicate by publishing domain events via these endpoints or channels.

4.3 DevOps

Docker and Kubernetes

Docker helps to "create" containers, and Kubernetes allows you to "manage" them at runtime. Use Docker for packaging and shipping the app. Employ Kubernetes to deploy and scale your app.

Startups or small companies with fewer containers usually can manage them without having to use Kubernetes, but as the companies grow, their infrastructure needs will rise; hence, the number of containers will increase, which can be difficult to manage. This is where **Kubernetes** comes into play.

Clouds

- AWS
- GCP
- Azure

References (Credits)

- **https://spring.io/microservices**
- **https://www.javaguides.net/p/spring-boot-tutorial.html**
- **https://spring.io/projects/spring-cloud**

My Recommended Udemy Courses

Spring 6 and Spring Boot 3 for Beginners (Includes 7 Projects)

Building Microservices with Spring Boot and Spring Cloud

Building Real-Time REST APIs with Spring Boot — Blog App

Testing Spring Boot Application with JUnit and Mockito

Spring Boot Microservices Roadmaps

Written by Ramesh Fadatare

4.2K followers · 19 following

Follow

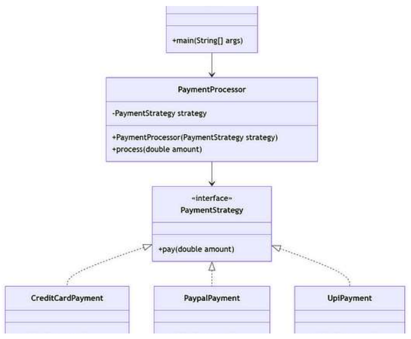
Founder: https://www.javaguides.net YouTuber: https://www.youtube.com/c/javaguides (170K) Udemy Bestseller Instructor:https://www.udemy.com/user/ramesh-fadatare

No responses yet

Write a response

What are your thoughts?

More from Ramesh Fadatare



In JavaGuides by Ramesh Fadatare

Why Senior Java Developers Love the Strategy Pattern

The Strategy Pattern is one of the most practical patterns in Java. It helps you write...

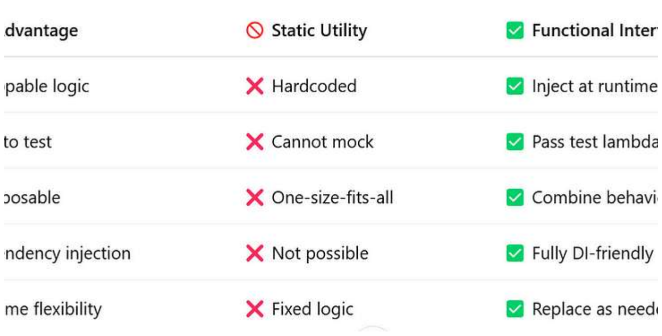
May 27 388 9

In Javarevisited by Ramesh Fadatare

The 15 Reusable Java Snippets I Use in Every Project (2025 Edition)

Instead of rewriting the same code repeatedly, I keep these 15 reusable,...

Jun 15 92 6



dvantage	🚫 Static Utility	✅ Functional Inter
pable logic	❌ Hardcoded	✅ Inject at runtime
to test	❌ Cannot mock	✅ Pass test lambda
posable	❌ One-size-fits-all	✅ Combine behavi
ndency injection	❌ Not possible	✅ Fully DI-friendly
me flexibility	❌ Fixed logic	✅ Replace as need

In Javarevisited by Ramesh Fadatare

Can You Inject a Prototype Bean Into a Singleton in Spring? | Aske...

Can I inject a prototype bean into a singleton bean? The short answer is: yes, but not in the...

May 28 136 6

Ramesh Fadatare

🚫 Stop Writing Utility Classes the Old Way: Use Functional Interface...

Java developers love utility classes. They're easy to spot—final classes with static...

Apr 18 963 34

See all from Ramesh Fadatare

Recommended from Medium

Kavya's Programming Path

IBM Java Developer Interview—Experience

Problem: Reverse a String Without Using Built-in Functions

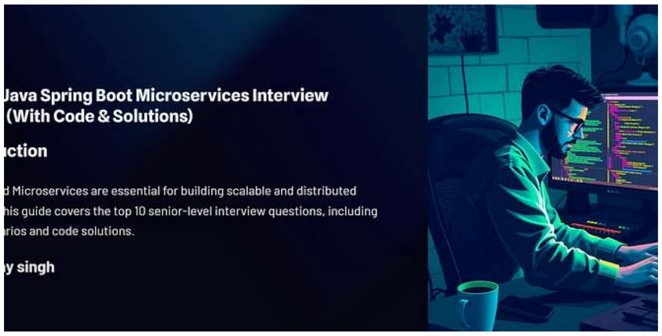
Jun 24 28

In Stackademic by Pudari Madhavi

Microservices Interview Questions and Answers

Microservices is no longer just a buzzword—it's the backbone of modern software...

6d ago 11



Java Spring Boot Microservices Interview (With Code & Solutions)
ction
d Microservices are essential for building scalable and distributed
his guide covers the top 10 senior-level interview questions, including
rios and code solutions.
y singh

Sanjay Singh

🚀 Top 10 Java Spring Boot Microservices Interview Question...

🔍 Introduction

Feb 25 54

In Javarevisited by Devrim Ozcay

Top 10 Java Interview Questions That Actually Tripped Me Up

"I thought I knew Java... until my interview turned into a live debugging session."

Jun 21 103 2

Arvind Kumar

Top 10 Java Interview Questions Every Senior Engineer Should Be...

Let's be honest: Java interviews are no longer about "What's the difference between ==...

Java Interview

6 Essential Design Patterns Every Java Developer Must Know

When your Java codebase starts feeling like a jungle of conditionals, copy-paste fixes, or...

6d ago 52

Jun 23 20 1

See more recommendations