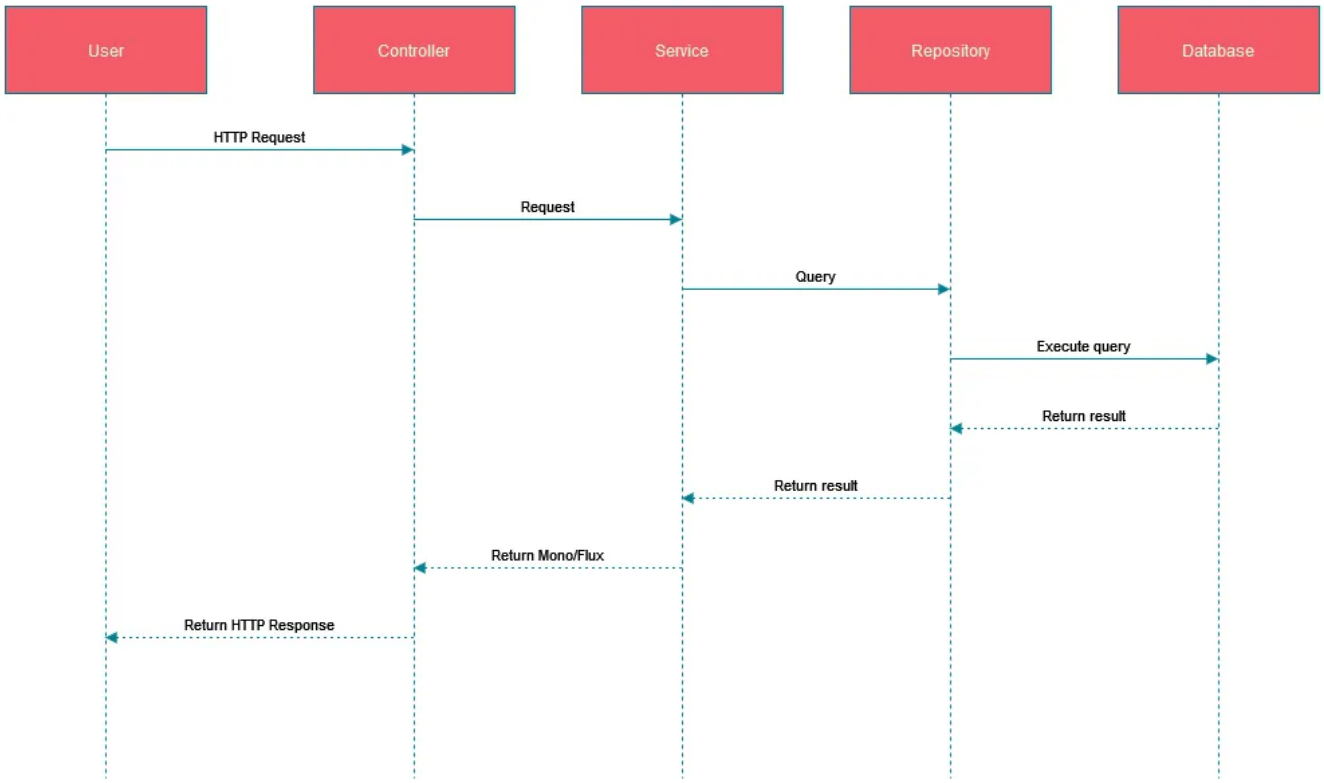
Member-only story

Difference between Mono and Flux in Spring WebFlux?

Understanding difference between Mono and Flux to write better reactive code in Spring WebFlux

Soma · Follow
Published in Javarevisited
7 min read · Apr 22



Hello folks, if you are preparing for *Java, Spring and Spring Boot interview* then you know that how important are the questions like difference between X and Y. This is the type of questions I have seen most of the times during interviews and preparing them not only give you edge over other candidates on interviews but also improve your understanding of underlying concepts.

In past few articles we have seen:

1. difference between @Bean and @Compoenent
2. difference between @RequestParam and @PathVariable,
3. difference between @ `Contorller`
4. and @Controller vs @Service @Repository and in this article, we will see the *difference between Mon and Flux in Spring*.

If you don't know, Spring WebFlux is a reactive web framework for building non-blocking, event-driven web applications. It provides two main types of reactive streams, Mono and Flux, which are used to represent a single value and a stream

of values respectively.

While both Mono and Flux are **used for processing asynchronous data,** they have different characteristics and are suited for different use cases and that's what you will learn in this article.

Now that we know the basics, its time to deep dive into Mon and Flux and see how they are used as well as learn key differences between Mono and Flux in Spring WebFlux and when to use each of them.

But, if you are not a Medium member yet then I highly recommend you to join Medium and read my other member only articles for your interview preparation. You can **join Medium here**

What is Mono? How to use it? Example

In Spring WebFlux, **Mono is a reactive type that represents zero or one value.** It's often used in scenarios where you want to **return a single object,** such as when retrieving data from a database or making an HTTP request. Mono is also a key component of reactive programming in Spring, which allows applications to handle more requests with fewer resources.

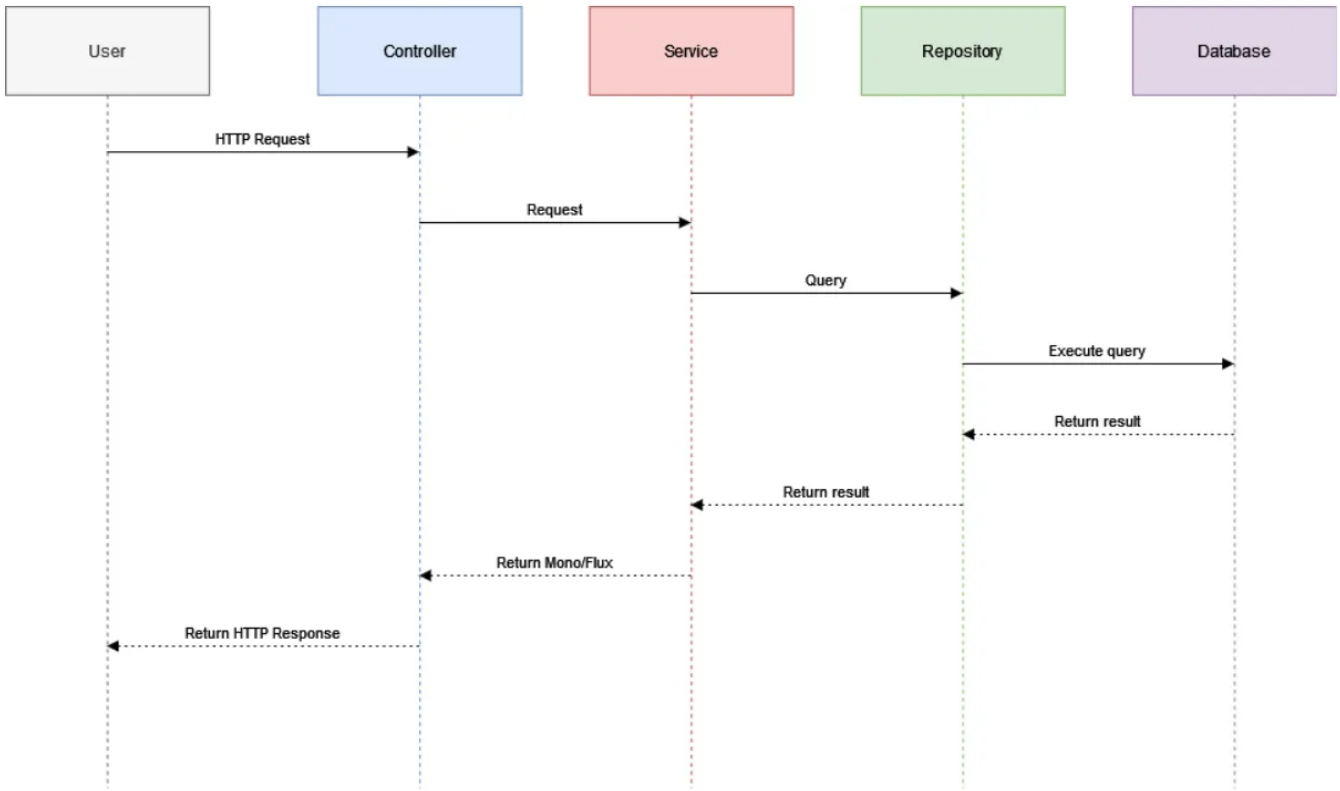
Here is an example of creating a Mono object in Spring WebFlux:

```
Mono<String> mono = Mono.just("Hello World");
```

In this example, we are creating a **Mono object that contains a String with the value "Hello World".** The `Mono.just()` method is used to create the Mono object, which takes a single value as its argument. This Mono object can be returned as the result of an HTTP request or used in other reactive programming scenarios.

It's worth noting that **Mono is non-blocking**, which means that it allows other requests to be processed while waiting for the result. This is in contrast to traditional blocking code, **which waits for a response before moving on to the next task**. Reactive programming with Mono is an effective way to create efficient and responsive applications.

Now that you know what is Mono, how many values it can return and when to use it, its time to see what is Flux, its twin brother in Spring web flux.



What is Flux? How to use it? Example

Flux is also a reactive streams library in Spring WebFlux that helps to handle asynchronous data streams in a non-blocking manner. It is *designed to handle zero to N elements* and emits data asynchronously as soon as it becomes available.

Flux uses the Reactive Streams specification, which is a standard for building reactive applications. The Reactive Streams specification is based on four interfaces: Publisher, Subscriber, Subscription, and Processor. Flux is an implementation of the Publisher interface and can be subscribed to by a Subscriber.

Here is a code example that shows how to create a Flux object in Spring:

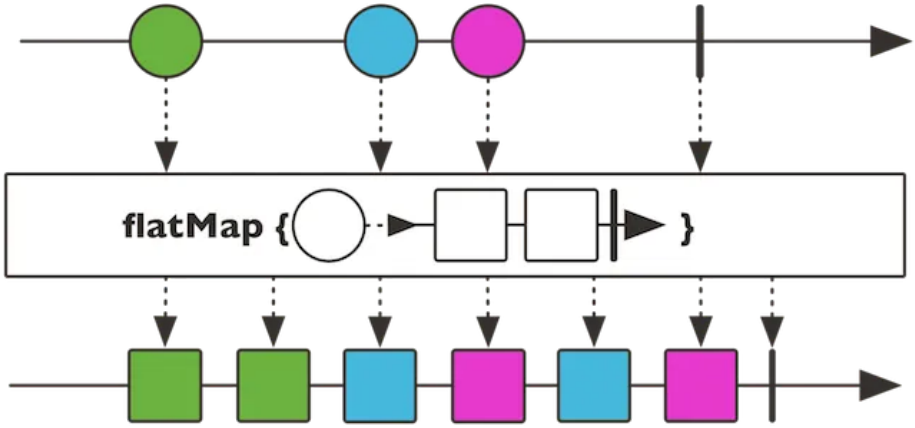
```
Flux<String> flux = Flux.just("apple", "banana", "orange");

flux.subscribe(System.out::println);
```

In the above example, we have created a Flux that emits three elements: `“apple”` , “banana”, and “orange”. We have then subscribed to the Flux using the `subscribe()` method and passed in a lambda expression that prints each element to the console.

Flux can also be created from other sources such as Iterable, Array, or other reactive streams libraries. It provides various operators to transform, filter, and combine data streams. By using *Flux*, you can write reactive code that can handle large amounts of data and perform operations on them in a non-blocking way.

Now that we know what is Mono and Flux, its time to check out the difference between these two for interview purpose. You can also transform or convert Mono to Flux as shown below:



What is difference between Mono and Flux in WebFlux?

The main difference between Mono and Flux in Spring WebFlux is the number of elements that they can emit. A Mono is a reactive type that can emit at most one element, whereas a Flux can emit multiple elements.

In other words, a Mono represents a stream of zero or one element, whereas a Flux represents a stream of zero to many elements. This difference is reflected in their respective APIs, where the Mono has methods like `map`, `filter`, and `flatMap`, similar to the traditional `Optional` class in Java, whereas the Flux has methods like `concat`, `merge`, and `zip`, which are used to combine multiple streams.

To understand the difference, **consider a scenario where an API call is made to retrieve a user's account balance from a bank**. In this case, the balance will be a single value that can be returned in a Mono.

On the other hand, if we want to retrieve **a list of recent transactions for the account,** there can be multiple values, and a Flux is better suited for this scenario.

Here is another example of a Mono and a Flux:

```
// Mono example
Mono.just("Hello, world!")
    .subscribe(System.out::println);

// Flux example
Flux.just(1, 2, 3, 4, 5)
    .map(i -> i * i)
    .filter(i -> i % 2 == 0)
    .subscribe(System.out::println);
```

You can see that Mono returns a single values and Flux can return multiple values, that's they main difference between them.

Java and Spring Interview Preparation Material

Before any Java and Spring Developer interview, I always use to read the below resources

Grokking the Java Interview

Grokking the Java Interview: click here

I have personally bought these books to speed up my preparation.

You can get your sample copy here, check the content of it and go for it

Grokking the Java Interview [Free Sample Copy]: click here

> *If you want to prepare for the Spring Boot interview you follow this consolidated ebook, it also contains microservice questions from spring boot interviews.*

Grokking the Spring Boot Interview

You can get your copy here — **Grokking the Spring Boot Interview**

Conclusion

That's all about **difference between Mono and Flux in Spring WebFlux framework**. The use of Mono and Flux provides Java developers with a powerful and flexible way to handle asynchronous data streams. While both Mono and Flux can be used for reactive programming, they have different use cases and behaviors.

As we have seen, Mono is used when you want to work with a single value, while Flux is used when you want to work with a potentially unbounded stream of values.

Mono is designed for simple use cases that require only a single value, such as retrieving data from a database or making a network call. Flux, on the other hand, is designed for complex use cases that involve streams of data, such as real-time data processing or event-driven systems.

Understanding the differences between Mono and Flux is important when building reactive applications with Spring WebFlux. By selecting the appropriate type for your use case, you can ensure that your application is efficient, scalable, and responsive and also bug free.

By the way, if you are preparing for Java and Spring interview then In my earlier articles, I have also shared *21 Software Design Pattern questions*, 10 *Microservice Scenario based questions*, *20 SQL queries from Interviews*, ***50 Microservices questions***, *60 Tree Data Structure Questions*, 15 System Design Questions, and *35 Core Java Questions* and *21 Lambda and Stream questions* which you can use for your Java interview preparation.

There are a lots of frequently asked questions for Java developers in those articles.

And, if you are not a Medium member yet then you can **join Medium here**

Join Medium with my referral link - Soma

Read this and every story from thousands of great writers on Medium). Read more from great writers on Medium like...

medium.com

Other **Java and Spring Interview articles** you may like to read:

Top 10 Microservices Problem Solving Questions for 5 to 10 years experienced Developers

From Service Discovery to Security: 10 Problem-Solving Questions to Test the Microservices Skills of Developers with 5...

medium.com

20 Advanced Core Java Interview Questions for Experienced Developers of 5 to 10 Years

These are difficult and advanced core Java questions for 5 to 10 years Experienced Java developers

medium.com

Why Every Developer Should Learn Docker in 2023?

With Docker, your app will work on every machine, not just yours.

medium.com

Spring Boot | Spring Framework | Java | Programming | Microservices

Follow

Written by Soma

4.1K Followers · Editor for Javarevisited

Java and React developer, Join Medium (my favorite Java subscription) using my link👉 https://medium.com/@somasharma_81597/membership

More from Soma and Javarevisited

Implementation	implementation	an implementation	features
Persistence API	Hibernate provides its own API	Defines a set of standard APIs for ORM	Builds on JPA APIs, adds more functionality
Database Support	Supports various databases through dialects	Depends on the JPA implementation used	Depends on the JPA implementation used
Transaction Management	Provides its own transaction management	Depends on the JPA implementation used	Depends on the JPA implementation used
Query Language	Hibernate Query Language (HQL)	JPQL (Java Persistence Query Language)	JPQL (Java Persistence Query Language)
Caching	Provides first-level and second-level caching	Depends on the JPA implementation used	Depends on the JPA implementation used
Configuration	XML, annotations, or Java-based configuration	XML, annotations, or Java-based configuration	XML, annotations, or Java-based configuration
Integration	Can be used independently or with JPA	Works with any JPA-compliant implementation	Works with any JPA-compliant implementation

Soma in Javarevisited

Difference between Hibernate, JPA, and Spring Data JPA?

Hello folks, if you are preparing for Java Developer interviews then part from preparing Core Java, Spring Boot, and Microservices, you...

✨ · 10 min read · May 26

253 1

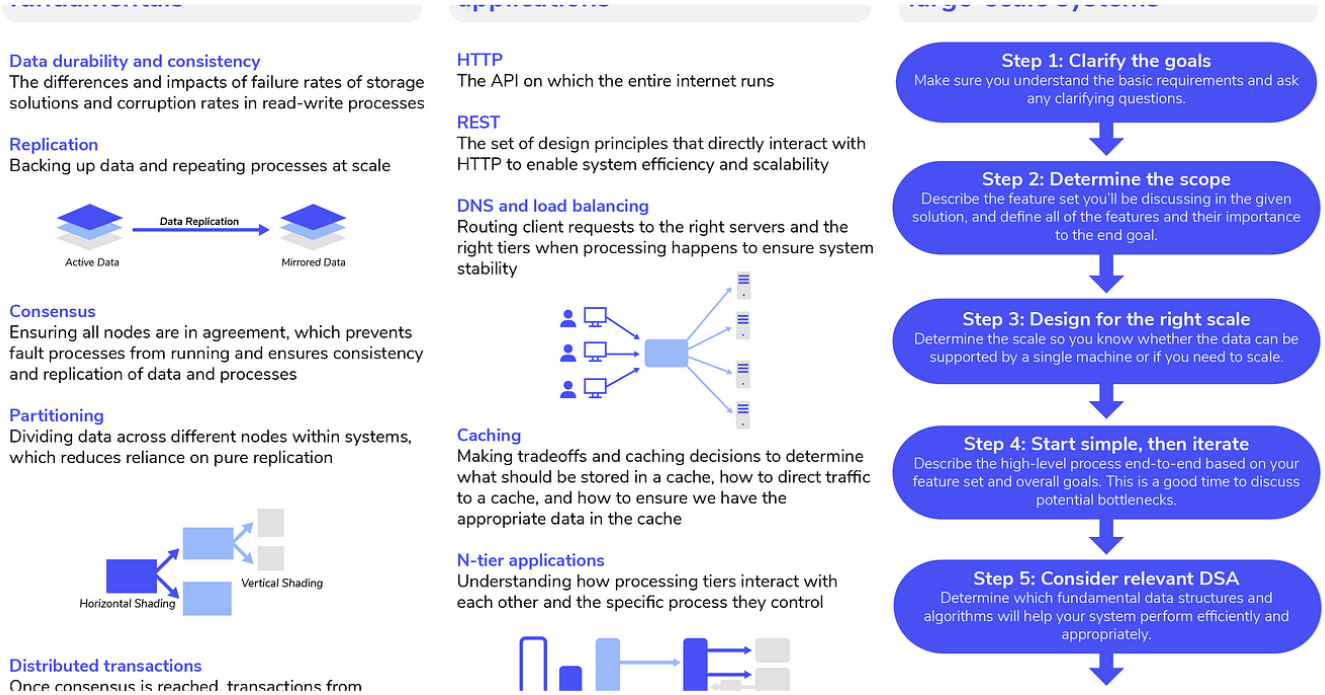
javinpaul in Javarevisited

Top 10 Websites to Learn Python Programming for FREE [2023]

Hello guys, if you are here then let me first congratulate you for making the right decision to learn Python programming language, the...

13 min read · Apr 22, 2020

328 6



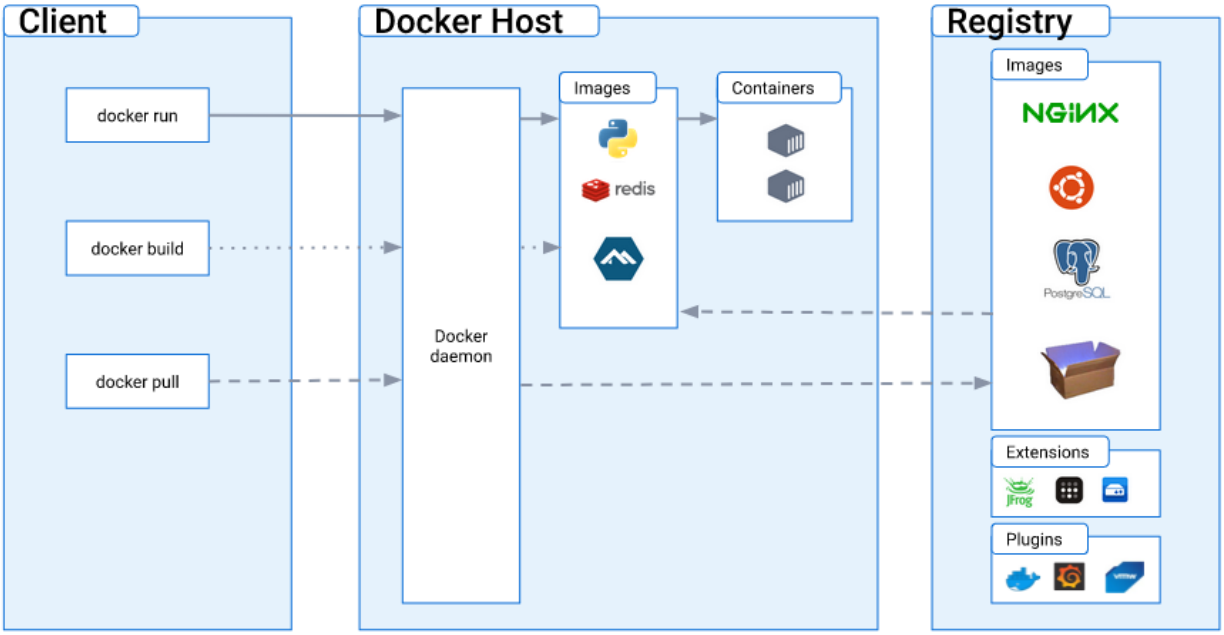
javinpaul in Javarevisited

Top 3 System Design Cheat Sheets for Developers

3 System Design Cheat Sheet you can print and put on your desktop to revise before Tech interviews

6 min read · Jul 19

204 1



Soma in Javarevisited

How Docker works internally? Magic Behind Containerization

Exploring the Inner Workings of Docker: Unveiling the Magic Behind Containerization

6 min read · Jun 30

183

See all from Soma

See all from Javarevisited

Recommended from Medium

Rupert Waldron

Create a non-blocking REST Api using Spring @Async and Polling

Get the great asynchronous, non-blocking experience you deserve with Spring's basic Rest Api, polling and @Aysnc annotation.

12 min read · Feb 26

17

Hamza Nassour in Javarevisited

What Happens Internally When You Start A Spring Boot Application(Part1)

ApplicationContext creation/registration , AutoConfiguration .. .

4 min read · Feb 19

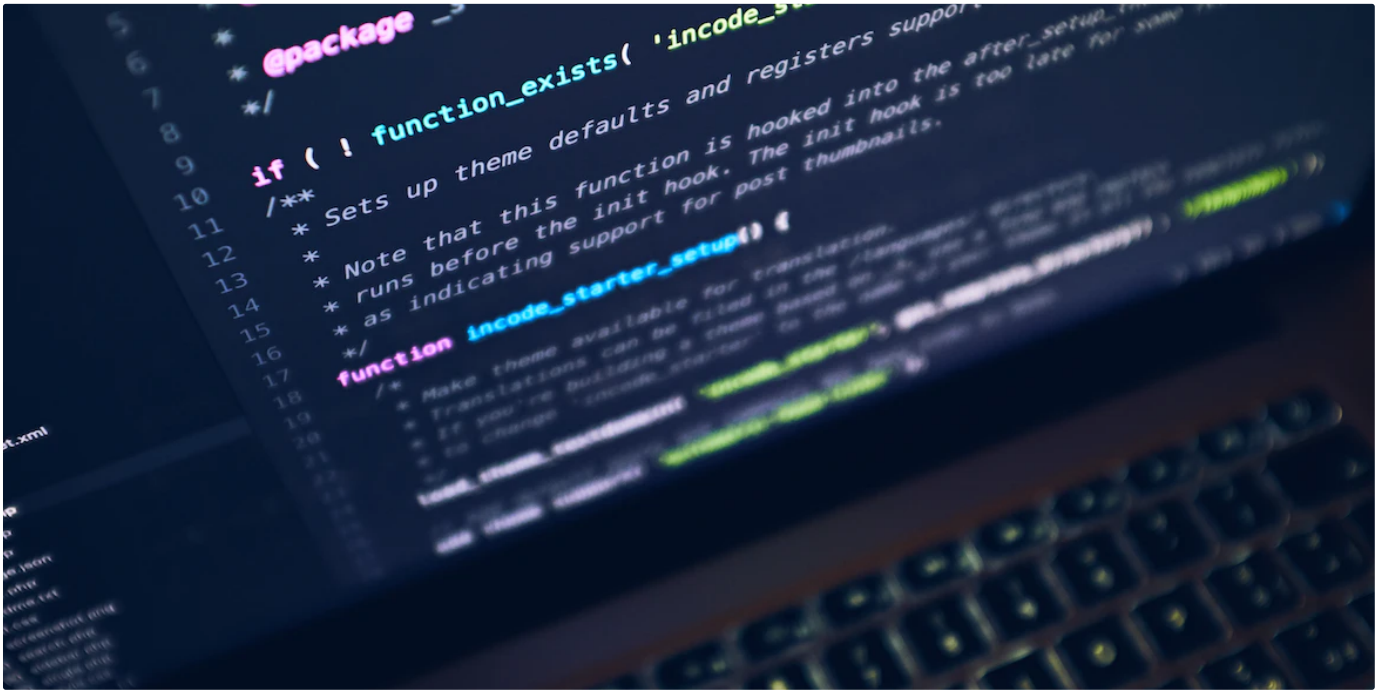
Amit Himani

Distributed Transaction in Spring Boot Microservices

Distributed transactions in microservices refer to transactions that involve multiple microservices, each handling a part of the...

✨ · 5 min read · Feb 17

49 3



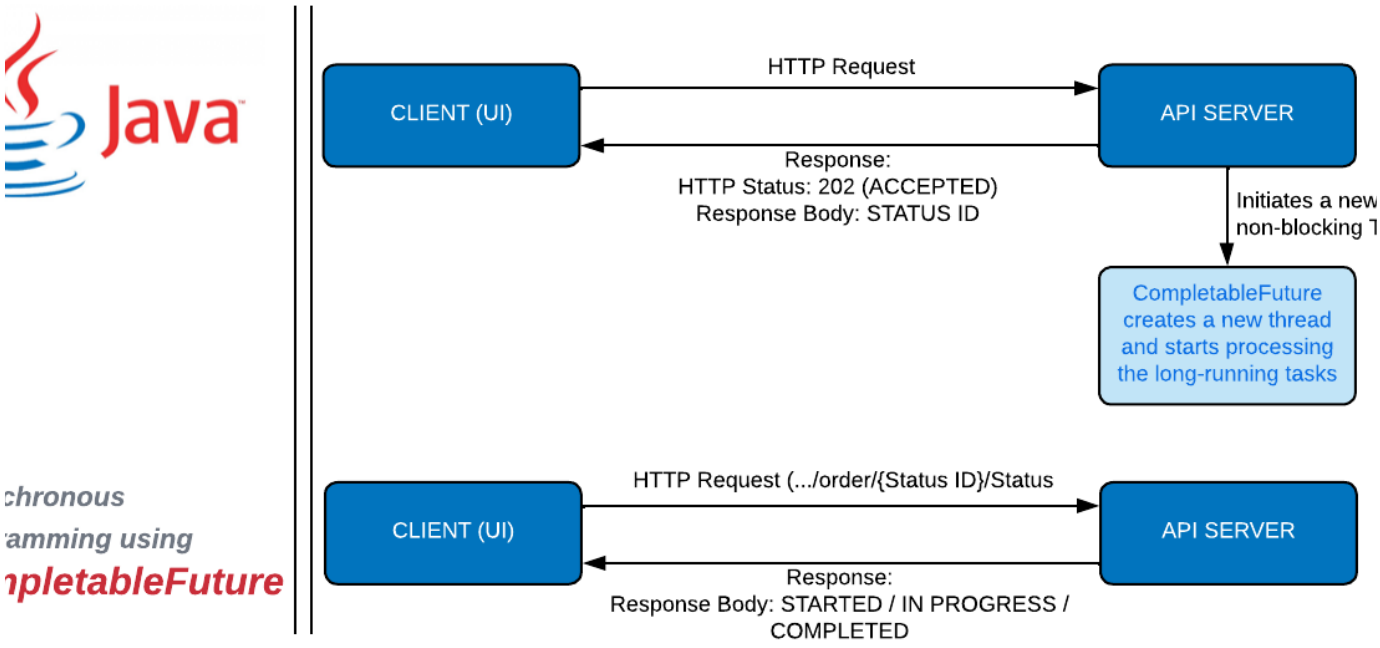
Tech Is Beautiful in Dev Genius

Mastering Concurrency in Java: 8 Best Practices Every Java Developer Should Follow

Boost Your Java Development Skills with These Essential Concurrency Best Practices

✨ · 8 min read · Feb 19

73



Sumant Mishra

Java: Asynchronous programming using CompletableFuture

In this post, I will discuss how to execute a long-running process in asynchronous as non-blocking code.

3 min read · Feb 10

4 1

Abhishek Anand in CodeX

Optimising Performance with GraalVM: A Guide to Migrating a Spring Boot Project to Native Image

Everything you need to know to migrate a Spring Boot project to native image on GraalVM.

4 min read · Feb 11