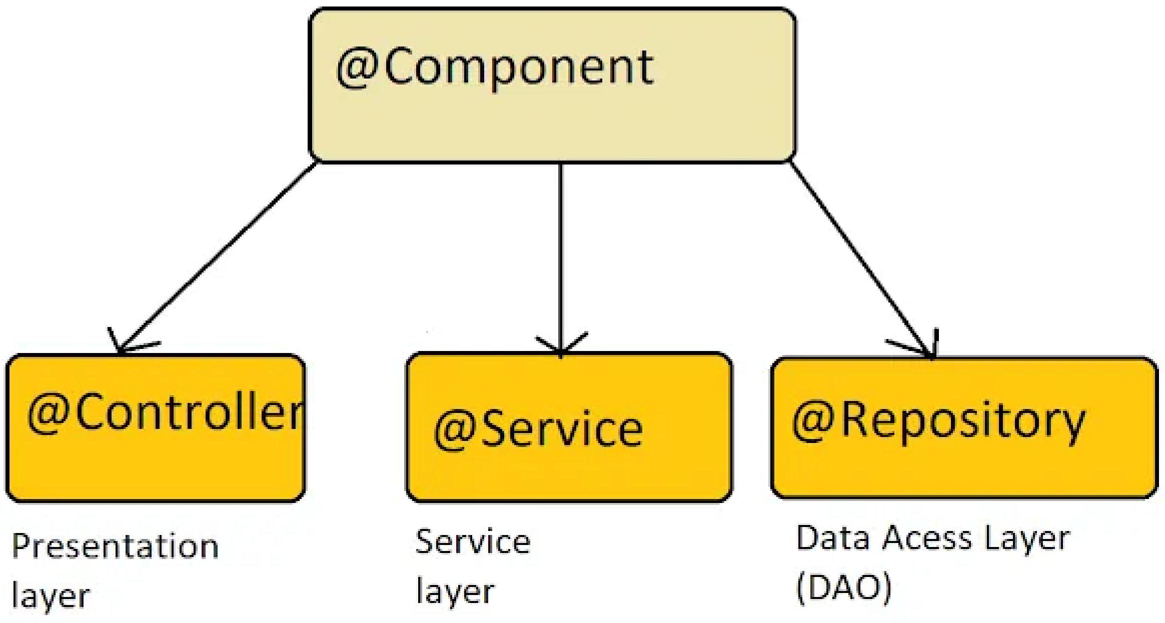
✦ Member-only story

Difference between @Controller, @Service, and @Repository Annotations in Spring Framework?

While all three are stereotype annotation in Spring and can be use to represent bean where exactly they are used is the key for answering this popular Spring interview question

Soma · Follow
Published in Javarevisited
7 min read · Mar 23

Listen Share More



Hello friends, If you are preparing for Java and Spring Developer interview, it is important to understand the different annotations used in the Spring framework like what they do and when to use them. In the past, I have also told you about difference between @ `Contorller` and @ `RestController` annotation and in this article, we will take a look at another set of popular Spring annotations.

Three commonly used annotations are `@Service` , `@Controller` , and `@Repository` . While these annotations may seem similar, they have different purposes and are used in different parts of the Spring application.

By the way, if you are preparing for interview In my earlier articles, I have also shared ***25 Advanced Java questions***, *21 Software Design Pattern 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.

In this article, we will explore the differences between `@Service` , `@Controller`, and `@Repository` in Spring from an interview point of view. We will cover the basic definitions of each annotation, where they are typically used, and how they can be implemented in a Spring application.

We will also discuss some *common Spring interview question*s related to these annotations, such as when to use them and what are the benefits of using them.

So, let's dive in and explore the differences between @Service, @Controller, and @Repository in Spring. And, if you are not a Medium member then I highly recommend you to join Medium and read my other member only articles for your interview preparation. 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

What is @Controller Annotation in Spring Framework?

In Spring Framework, particularly Spring MVC, the Controller annotation is used to mark a class as a controller that handles requests for the web application. When a class is annotated with `@Controller` , Spring considers it as a web controller that is capable of handling incoming HTTP requests.

It is one of the most commonly used annotations in Spring MVC (Model-View-Controller) architecture, which allows developers to build flexible and loosely coupled web applications.

The `@Controller` annotation is usually used in combination with other annotations like `@RequestMapping` , `@GetMapping` , `@PostMapping` , etc. to map requests to specific methods within the controller. The methods within the controller handle the requests and return a response to the client.

By using the `@Controller` annotation, developers can easily build RESTful web services, web applications, and APIs. **It provides a clear separation between the presentation layer and the business logic layer of the** application, making it easier to maintain and test.

Overall, the `@Controller` annotation is an essential component of Spring MVC architecture, providing a simple and flexible way to handle incoming HTTP requests and produce appropriate responses.

Here's an example of a controller class in Spring using the `@Controller` annotation:

```
@Controller
public class MyController {

 @RequestMapping("/hello")
 public String sayHello() {
 return "Hello World!";
 }
}
```

In this example, the `@Controller` annotation is used to mark the class as a controller. The `@RequestMapping` annotation is used to map the `/hello` URL to the `sayHello()` method, which returns the string "Hello World!".

When a user navigates to the `/hello` URL, Spring will invoke the `sayHello()` method in the `MyController` class and return the string "Hello World!". This string could be rendered in a view and displayed in the user's web browser.

What is @Repository annotation in spring?

In Spring Framework, `@Repository` is an annotation used to indicate that the annotated class is a repository, which is responsible for data access and manipulation.

It is typically used in combination with other Spring technologies, such as Spring Data, to provide a convenient way to implement the persistence layer of an application.

Here is an example of a class annotated with `@Repository`:

```java
@Repository
public class CustomerRepositoryImpl implements CustomerRepository {

    private final EntityManager entityManager;

    @Autowired
    public CustomerRepositoryImpl(EntityManager entityManager) {
        this.entityManager = entityManager;
    }

    @Override
    public List<Customer> getAllCustomers() {
        TypedQuery<Customer> query = entityManager.createQuery("SELECT c FROM C
        return query.getResultList();
    }

    @Override
    public void saveCustomer(Customer customer) {
        entityManager.persist(customer);
    }
}
```

In this example, `@Repository` is used to indicate that the `CustomerRepositoryImpl` class is a repository. The class implements the `CustomerRepository` interface, which defines the methods for data access and manipulation.

The `EntityManager` object is injected using the `@Autowired` annotation, which is a Spring annotation used for dependency injection. The `getAllCustomers()` method uses the `entityManager` to create a query and retrieve all customers from the database. The `saveCustomer()` method uses the `entityManager` to persist a new customer to the database.

Overall, `@Repository` is a useful annotation in Spring Framework that helps developers to create and manage repositories for their applications.

What is @Service annotation in spring framework?

The `@Service` annotation is used in Spring to mark a class as a service provider. It is a specialized version of the `@Component` annotation and can be used interchangeably with `@Component` in most scenarios.

The main purpose of `@Service` is to provide a clear and concise way to indicate that a class is providing some business logic or service functionality. This annotation is particularly useful in larger Spring applications where there are many different components that need to interact with each other.

Here is an example of how to use the `@Service` annotation in a Spring application:

```
@Service
public class UserService {
    private final UserRepository userRepository;

    public UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    public List<User> getAllUsers() {
        return userRepository.findAll();
    }

    public User getUserById(Long id) {
        return userRepository.findById(id).orElse(null);
    }

    // Other business logic methods...
}
```

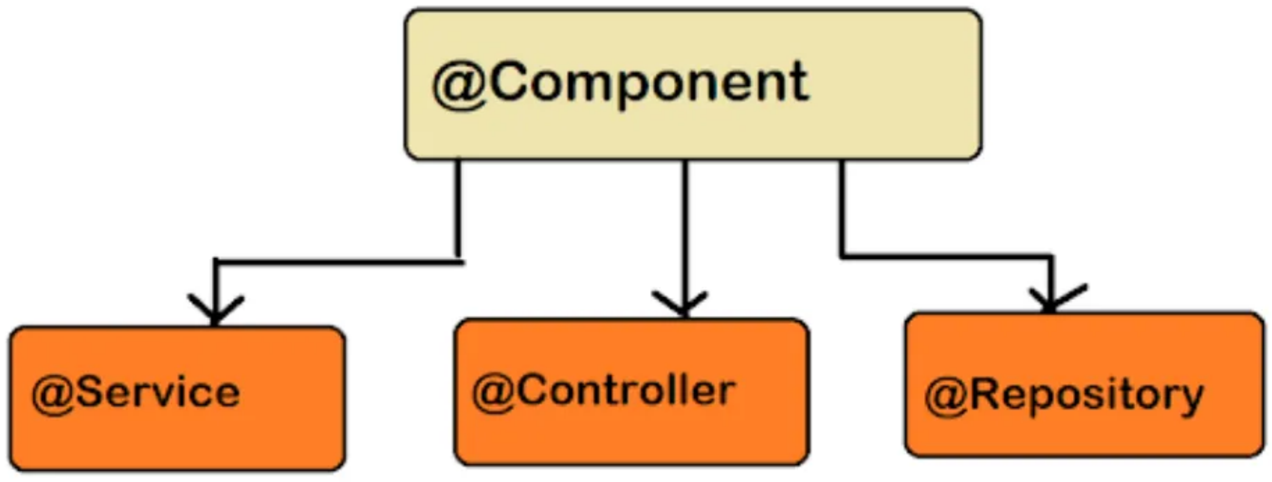
In this example, the `UserService` class is marked with the `@Service` annotation to indicate that it is providing business logic for the application. The class has a constructor that takes a `UserRepository` instance as a dependency, which is injected by Spring.

The `UserService` class provides several methods that perform various business logic operations on user data. These methods make use of the `UserRepository` instance to interact with the database.

By using the `@Service` annotation, we can clearly identify the `UserService` class as a service provider in the application, which can be helpful for developers who are trying to understand the overall architecture of the system.

Difference between Service, Controller, and Repository Annotations in Spring Framework

here are the key differences between `@Service` , `@Controller` , and `@Repository` annotations in Spring framework point by point, so that you can understand them better:



1. Purpose

- @ `Service` : Used to define a service class that contains business logic.
- @ `Controller` : Used to define a controller class that handles user requests and returns responses.
- @ `Repository` : Used to define a repository class that interacts with a database or persistence layer.

2. Layer

- @ `Service` : Generally used in the service layer of a Spring MVC application.

- @ `Controller` : Generally used in the presentation layer of a Spring MVC application.
- @ `Repository` : Generally used in the persistence layer of a Spring MVC application.

3. Exception handling

- @Service: Typically throws checked exceptions that are handled by the caller.
- @Controller: Uses Spring MVC's exception handling mechanism to handle exceptions.
- @Repository: Does not handle exceptions, rather, they are propagated to the caller.

4. Dependency injection

- @ `Service` : Typically, other service or repository objects are injected into a service object.
- @ `Controller` : Typically, a service object is injected into a controller object.
- @ `Repository` : Typically, a database connection or a session factory is injected into a repository object.

5. Transaction management

- @ `Service` : Typically manages transactions at the service layer.
- @ `Controller` : Does not manage transactions directly.
- @ `Repository` : Typically manages transactions at the persistence layer.

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**

That's all about **difference between Service, Controller, and Repository annotation in Spring Framework.** In summary, @Service, @Controller, and @Repository are all used for different purposes and belong to different layers of a Spring MVC application.

They have different exception handling mechanisms, dependency injection practices, and transaction management styles.

By understanding the differences between these annotations, you will be better prepared for your Spring interview and have a deeper understanding of the Spring framework.

Whether you are new to Spring or have experience with the framework, this blog post will provide valuable insights and practical knowledge that you can apply in your Spring projects.

Additionally, you can also prepare Microservices Questions like *difference between API Gateway and Load Balancer*, *SAGA Pattern*, *how to manage transactions in Microservices*, and *difference between SAGA and CQRS Pattern*, they are quite popular on interviews.

And, if you are not a Medium member then I highly recommend you to join Medium and read great stories from great authors from real field. You can **join Medium here**

Spring Spring Boot Java Programming Development

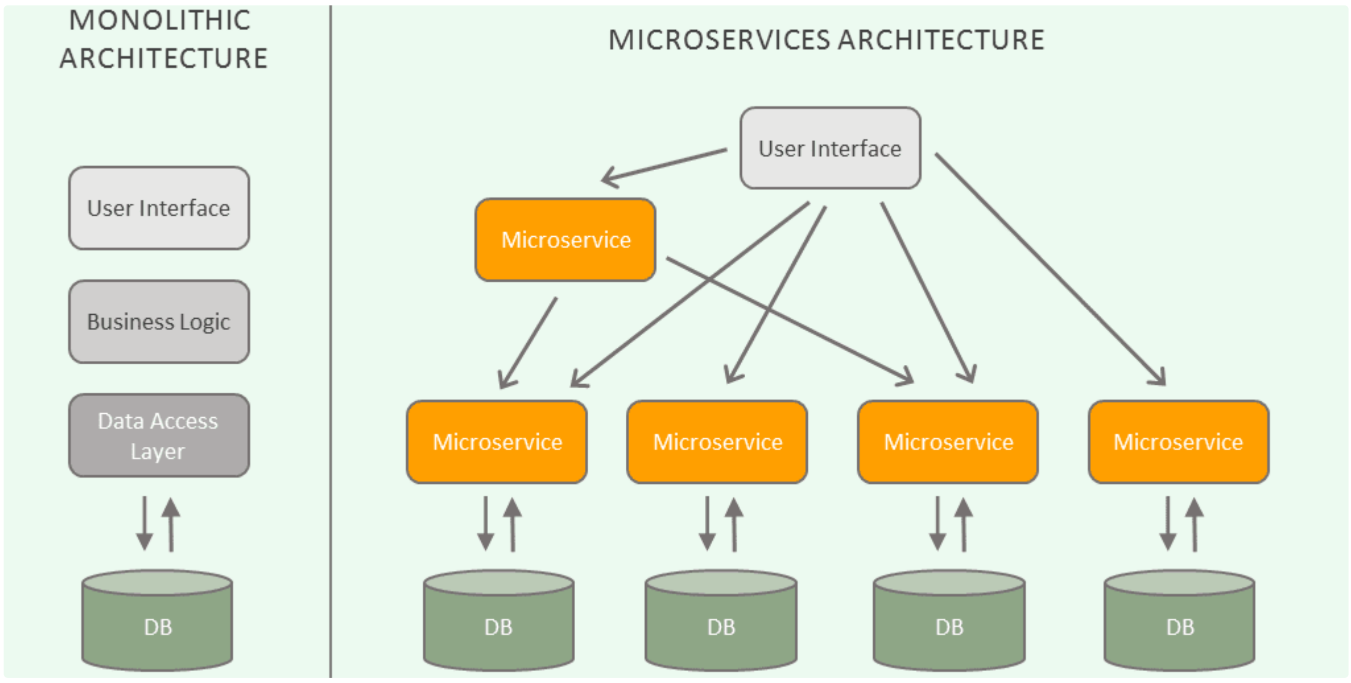
Follow

Written by Soma

4.2K 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



Soma in Javarevisited

18 Advanced Microservices Interview Questions for Experienced Developers: Challenges and Insights

From Microservice Architect to Scalability: 18 Interview Questions to Test the Microservices Skills of Developers with 5 to 10 Years of...

✦ · 13 min read · 3 days ago

114

Open in app ↗

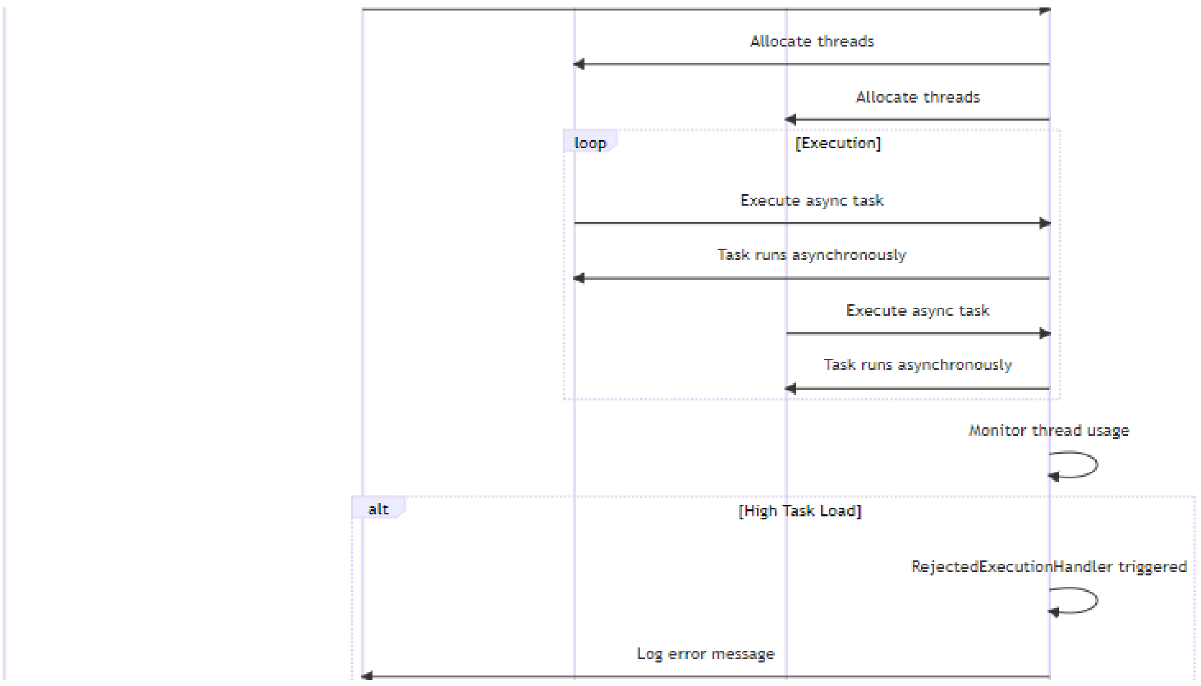
Ajay Rathod in Javarevisited

Comprehensive Spring-Boot Interview Questions and Answers for Senior Java Developers: Series-25

Greetings everyone,

9 min read · Aug 8

106 1



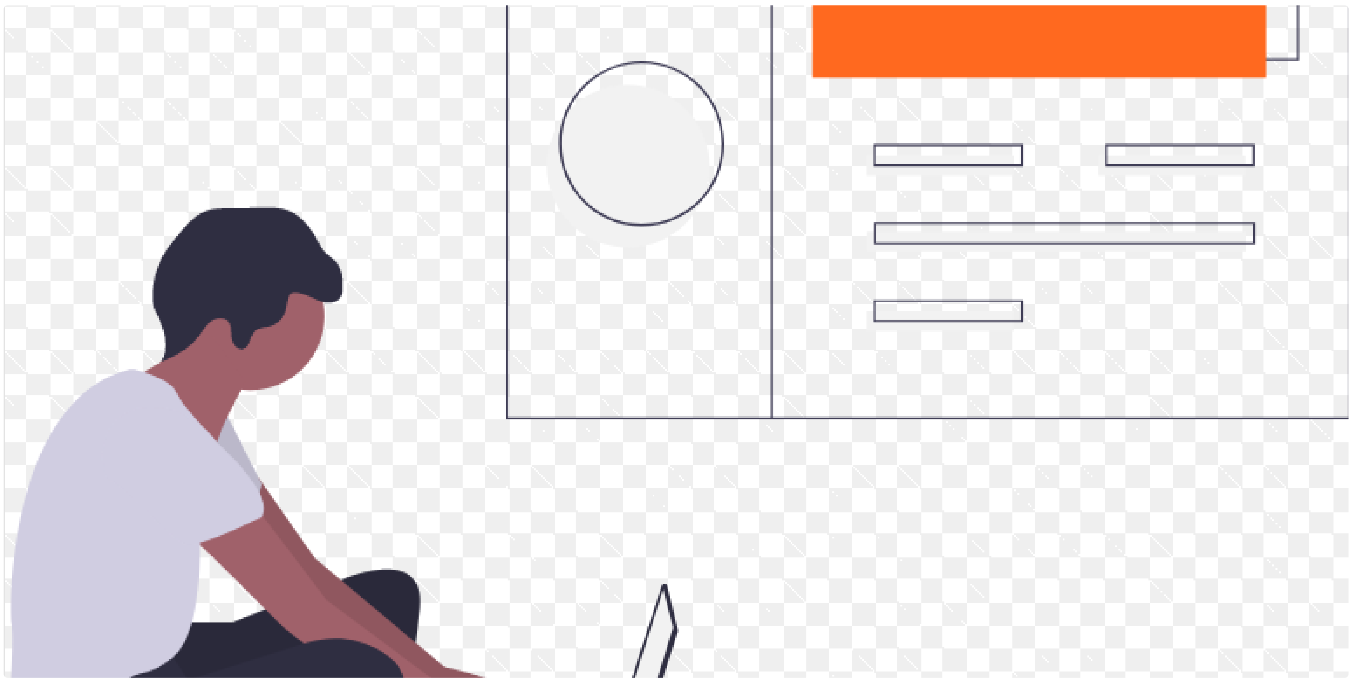
Srikanth Dannarapu in Javarevisited

Using Async Schedulers in Spring Boot

Problem Statement: Efficiently Managing Multiple Asynchronous Schedulers in a Spring Boot Application

5 min read · Aug 7

109 1



Soma in Javarevisited

Difference between @RequestMapping and @GetMapping Annotations in Spring MVC Web Framework

Hello friends , if you are preparing for Java Developer and Spring Developer interviews then you must prepare for questions like...

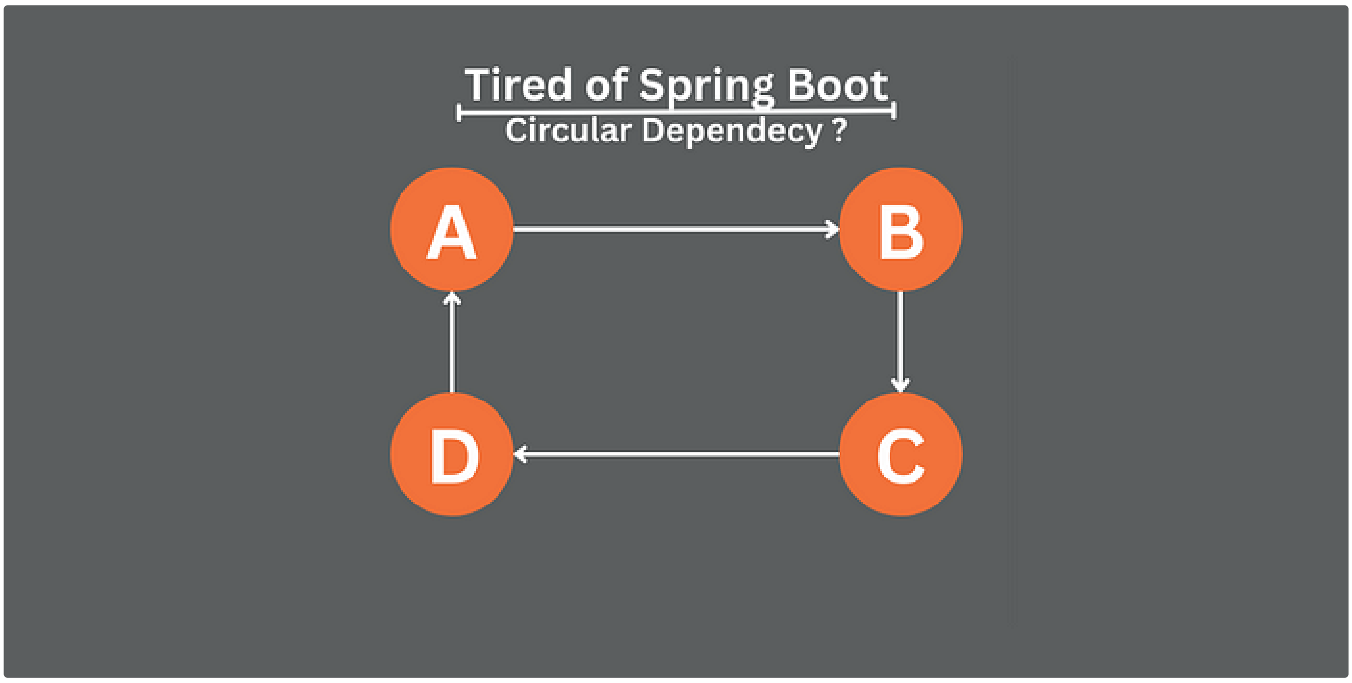
· 6 min read · Aug 19

49

See all from Soma

See all from Javarevisited

Recommended from Medium



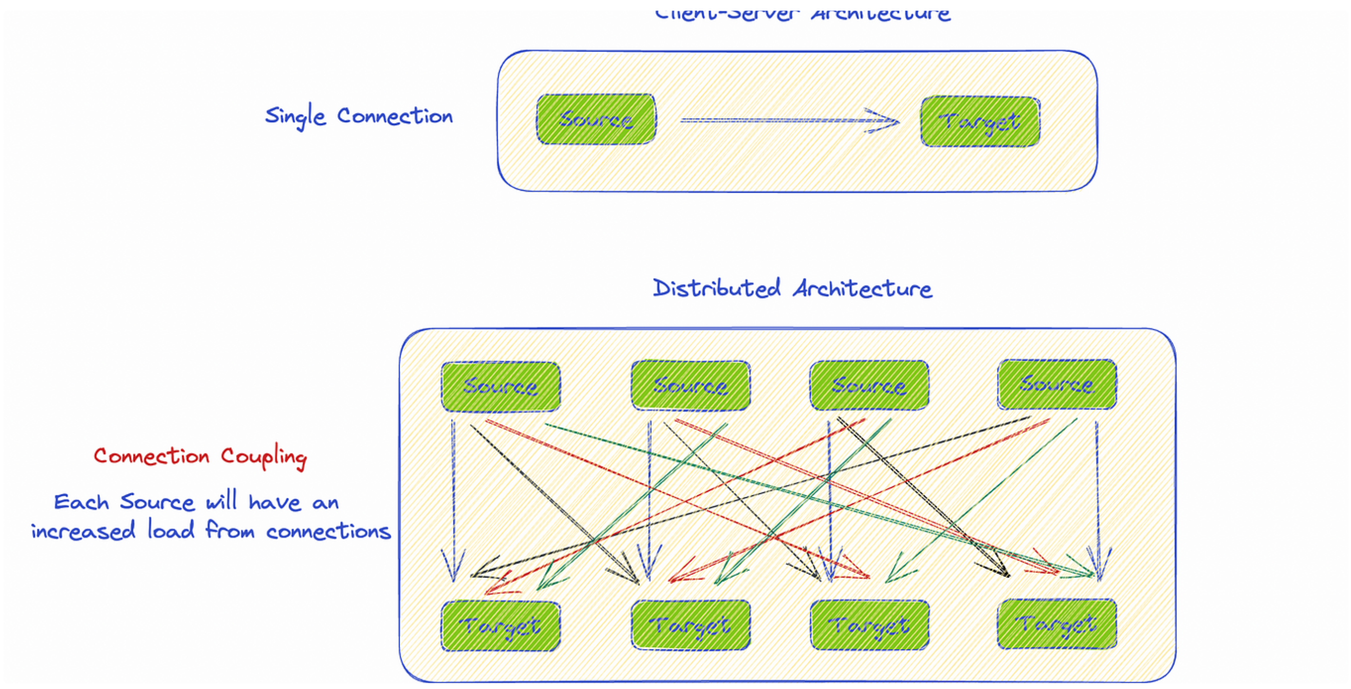
karthik jeyapal

Circular Dependency in Spring Boot: How to Detect and Fix It

In Spring Boot, circular dependency occurs when two or more beans depend on each other. This can happen when a bean requires another bean...

7 min read · Apr 16

3



Akshat Vashishtha

Kafka with Spring Boot using docker-compose

Prior understanding to Kafka, we should understand the problem Kafka try to solve. In simple Client-Server architecture source machine…

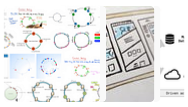
4 min read · May 9

38

Lists

It's never too late or early to start something

15 stories · 91 saves

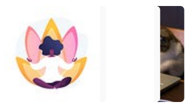


General Coding Knowledge

20 stories · 255 saves

Coding & Development

11 stories · 129 saves



Stories to Help You Grow as a Software Developer

19 stories · 308 saves

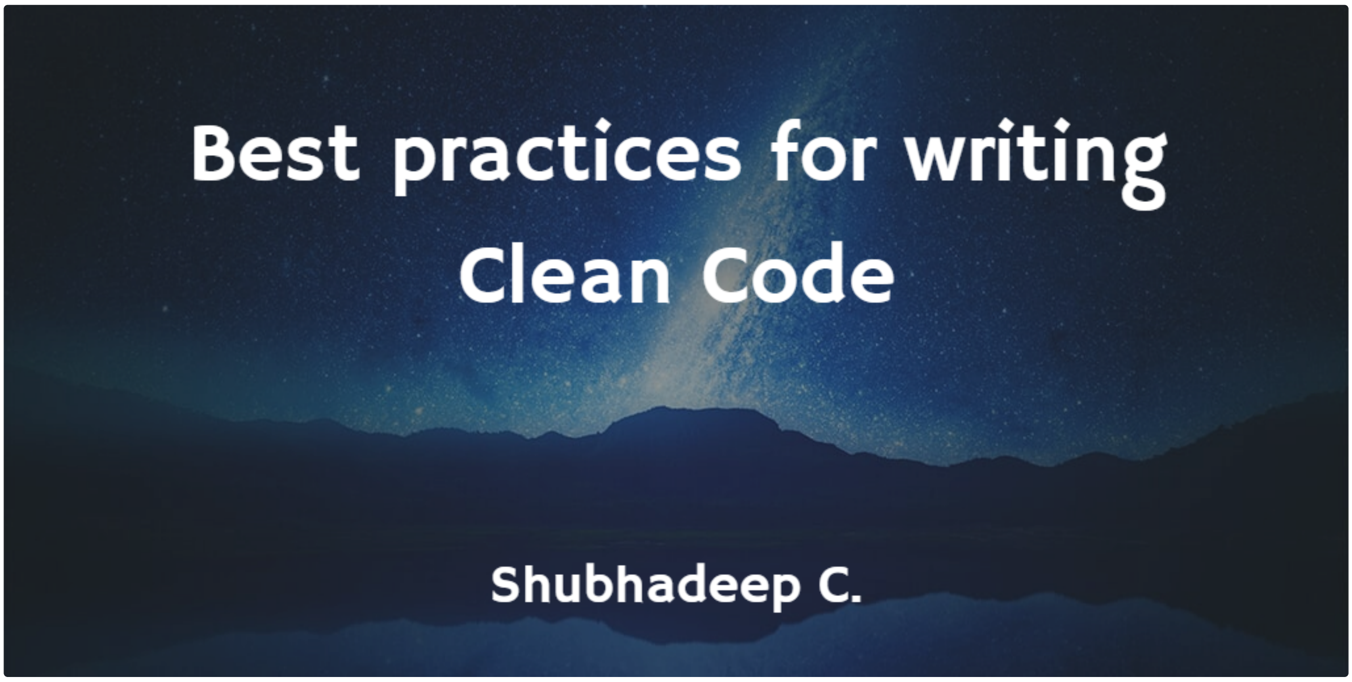
Eray Araz

Spring Webhook

Introduction

3 min read · Apr 2

108 1



Shubhadeep Chattopadhyay

Best practices for Clean Code

Clean code is a set of programming practices that emphasize the readability, maintainability, and simplicity of code. Writing clean code is...

7 min read · Mar 3

1.1K 29

Ionut Anghel

REST Endpoint Best Practices Every Developer Should Know

5 min read · Mar 18

72 1

Syed Habib Ullah

Top Most Asked Java Interview Questions at Accenture, Infosys, Capgemini, Wipro, Cognizant…

As Java is one of the most popular programming languages used in software development today, it's no surprise that Java skill is highly…

9 min read · Apr 1

68 2

See more recommendations