

[Open in app](#)

★ Member-only story

What is Circuit Breaker Design Pattern in Microservices? Spring Cloud Netflix Hystrix Example in Java?

Learn how to use Netflix Hystrix to create a circuit breaker in Java Microservices and prevent cascading failures.

Soma · [Follow](#)

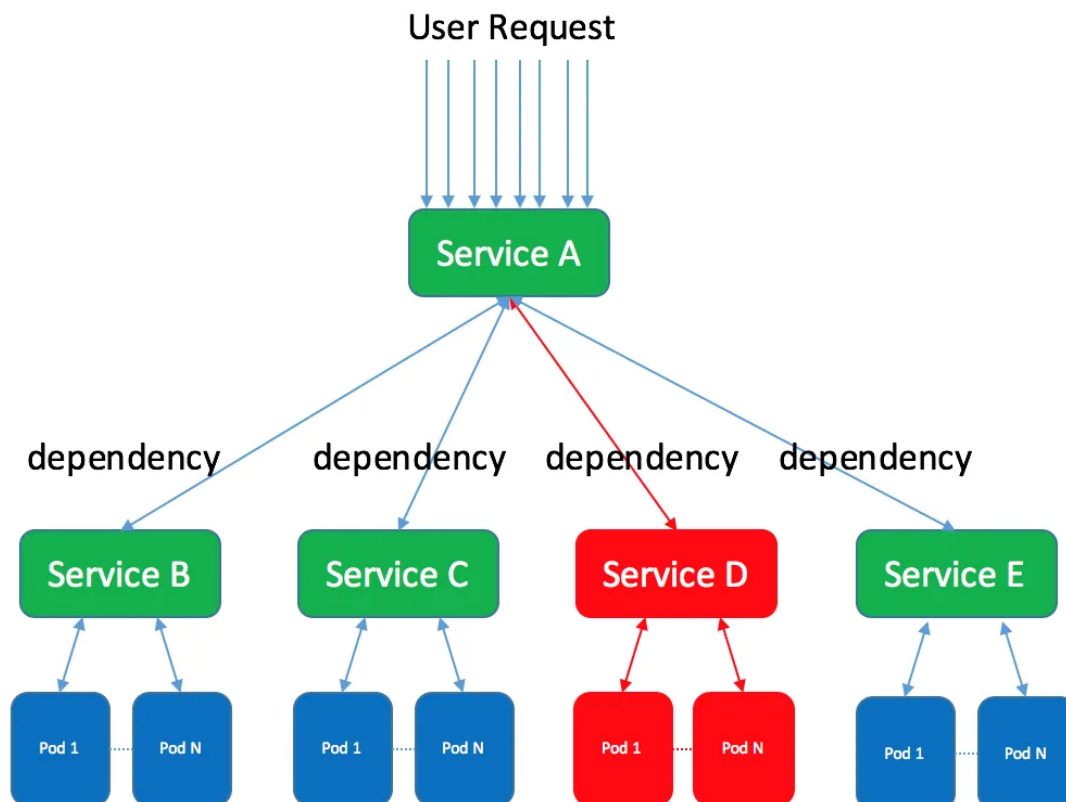
Published in Javarevisited

6 min read · Mar 25

Listen

Share

More



In today's world of Microservices architecture, ensuring the resilience and fault tolerance of your system is critical. One way to achieve this is through

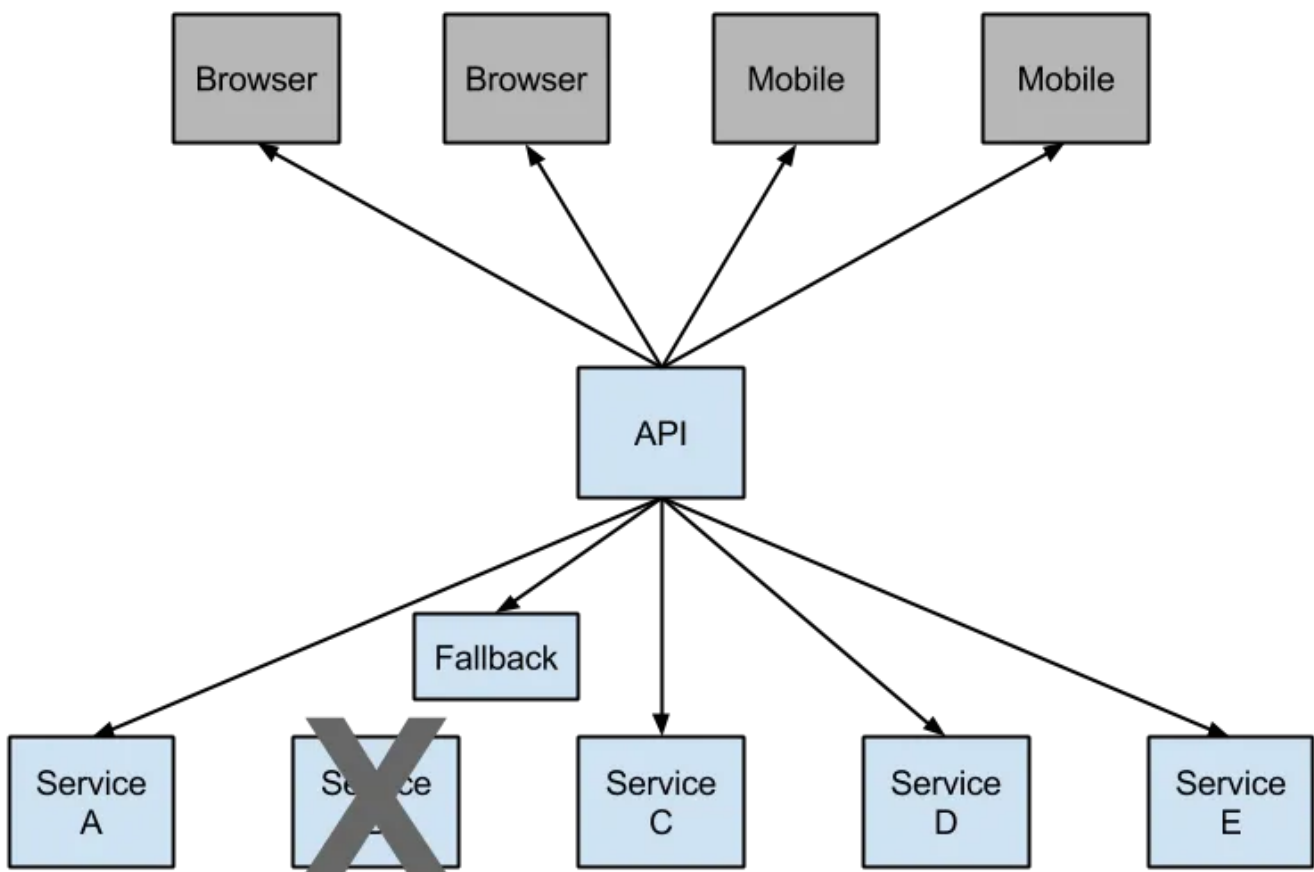
the **Circuit Breaker design pattern**, one of the **10 essential Microservice design pattern**, which allows you to detect and recover from failures in external services.

In the past, I have shared *Microservices design principles and best practices* as well as explained SAGA, CQRS, Load Balancer, and API Gateway pattern in detail and in this article, I will explain what is Circuit breaker design pattern and what problem does it solve.

We'll take a closer look at the Circuit Breaker pattern and explore how to implement it in Java using **Spring Cloud Netflix Hystrix**. By the end of this article, you'll have a better understanding of how to improve the resilience of your microservices architecture using the Circuit Breaker pattern.

The **Circuit Breaker pattern** is a design pattern used in distributed systems to **prevent cascading failures** when one or more services fail. The Circuit Breaker pattern works by **providing a fallback mechanism** when a service fails, rather than continuing to send requests that are likely to fail.

As shown in below diagram, where you can see that when Service B is not available it falls to Fallback, which could be another instance of same service, or another service, or return a fallback response that can be returned to the client immediately



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 without interruptions. 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...

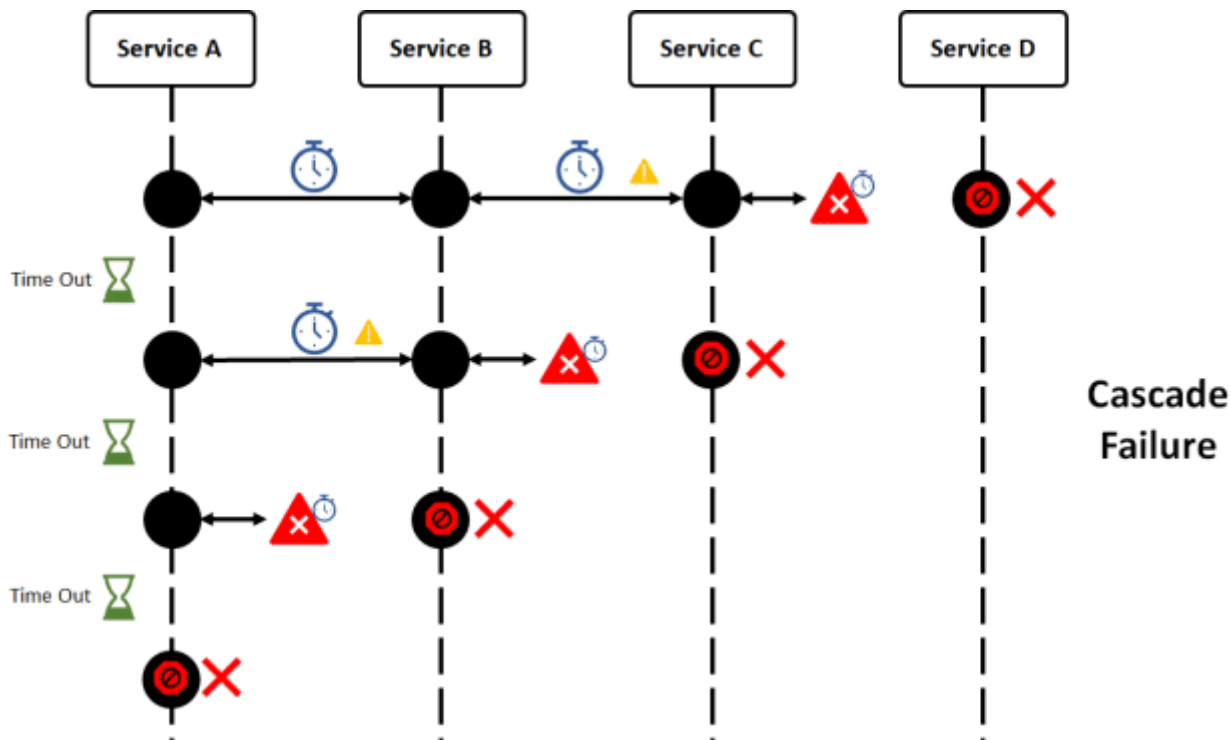
medium.com

How Does Circuit Breaker Pattern work?

The Circuit Breaker pattern operates similarly to an electrical circuit breaker. When a service fails, the Circuit Breaker trips and prevents any further requests from being sent to that service.

Instead, the **Circuit Breaker provides a fallback response that can be returned to the client immediately**. This helps prevent the failure from propagating to other services and causing a cascading failure.

In addition to providing a fallback mechanism, the Circuit Breaker pattern also includes functionality to monitor the status of the service. This involves periodically **sending test requests to the service to determine its status**. If the service has recovered, the Circuit Breaker can be reset, and requests can resume.



How to implement Circuit Breaker Pattern in Java?

There are many ways to implement circuit break pattern in Java. For example, You can use various libraries such as Netflix Hystrix, Resilience4j, Or Istio to implement circuit breaker pattern in Java. These libraries provide configurable Circuit Breaker implementations that can be easily integrated into a microservices architecture.

Here's an example of how to implement the Circuit Breaker pattern in Java using Spring Cloud and Netflix Hystrix:

First, you need to add the Hystrix dependency to your pom.xml:

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
</dependency>
```

Next, you can create a REST endpoint that calls a remote service using Hystrix:

```
@RestController
public class OrderController {

    @Autowired
    private OrderService OrderService;

    @GetMapping("/order-endpoint")
    public String OrderEndpoint() {
        return OrderService.remoteService();
    }

    @Component
    class OrderService {

        @HystrixCommand(fallbackMethod = "fallback")
        public String remoteService() {
            // Call the remote service here
            return "Success";
        }

        public String fallback() {
            return "Fallback";
        }
    }
}
```

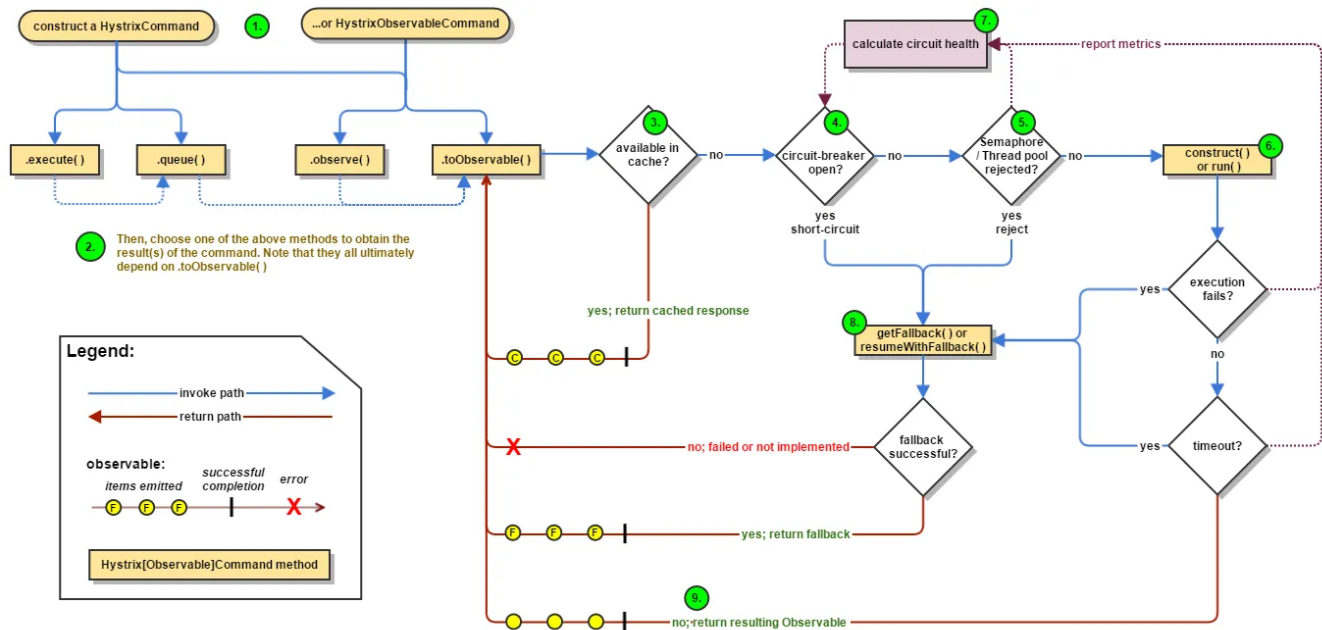
In this example, `OrderService` calls a remote service using Hystrix's `@HystrixCommand` annotation. If the remote service fails, the `fallback` method is called instead. This is the fallback mechanism provided by the Circuit Breaker pattern.

You can configure the behavior of the Circuit Breaker using properties in the `application.yml` file:

```
hystrix:
  command:
    default:
      circuitBreaker:
        requestVolumeThreshold: 10
        errorThresholdPercentage: 50
        sleepWindowInMilliseconds: 5000
```

In this example, the Circuit Breaker is configured to trip if 50% or more of the requests fail, and it will remain open for 5 seconds before trying again.

That's a basic example of how to implement the Circuit Breaker pattern in Java using Spring Cloud and Netflix Hystrix.



What are Pros and Cons of Circuit Breaker pattern?

Here are some of the pros and cons of using the Circuit Breaker pattern:

Pros:

1. Prevents cascading failures

The Circuit Breaker pattern helps prevent a failure in one service from causing a cascade of failures throughout the system.

2. Improves system resilience

By providing a fallback mechanism and monitoring the status of services, the Circuit Breaker pattern improves the resilience of the system.

3. Reduces load on failing services

When a service fails, the Circuit Breaker pattern stops sending requests to that service, reducing the load on the service and allowing it to recover more quickly.

4. Provides fault tolerance

The Circuit Breaker pattern provides a mechanism for handling faults, making the system more fault-tolerant.

Cons:

1. Adds complexity

Nothing comes at free of cost. Implementing the Circuit Breaker pattern can add complexity to the system, requiring additional code and configuration.

2. May increase latency

Another drawback of introducing circuit breaker pattern is increased latency. The fallback mechanism provided by the Circuit Breaker pattern may introduce additional latency into the system, especially if the fallback response is slow.

3. Requires monitoring

To be effective, the Circuit Breaker pattern requires monitoring the status of services, which can be time-consuming and resource-intensive.

4. May mask underlying issues

If the fallback response provided by the Circuit Breaker pattern is too generic, it may mask underlying issues with the service.

Overall, the *Circuit Breaker pattern* is a useful tool for improving the resilience of distributed systems, but it's important to weigh the pros and cons before deciding whether to use it in a specific situation.

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

In summary, the **Circuit Breaker pattern** is a **design pattern** used to prevent cascading failures in distributed systems. It works by providing a fallback mechanism when a service fails and includes functionality to monitor the status of the service.

You can also use libraries such as Netflix Hystrix, Resilience4j, or Istio to implement circuit-breaker pattern in Java Microservices.

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](#)**

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 Stories you may like

21 Lambda and Stream Interview Questions for Experienced Java Programmers

My favorite Java interview questions from Lambda expression and Stream API for experienced Java developers

medium.com

21 Software Design Pattern Interview Questions and Answers

Design patterns are important topic for Java interviews, here are common design pattern questions you can prepare for...

medium.com

25 Spring Framework Interview Questions for 1 to 3 Years Experienced Java Programmers

Preparing for Java Interviews? Don't forget to prepare for Spring framework questions, here are the popular Spring...

medium.com

Microservices

Microservice Architecture

Java

Programming

Spring Boot



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

Messaging Model	Traditional	Publish/Subscribe	Traditional
Scalability	Clustering/Network of Brokers	Partitioning	Clustering/Network of Brokers
Performance	Moderate	High	High
Data Persistence	On Disk (default), In-memory	On Disk	On Disk (default), Database
Integration	Programming Languages, Databases, Web Servers	Data Processing Systems, Databases, Data Sources	JMS Clients, Apache Camel, Apache CXF
Suitable For	Strict Ordering, Reliable Delivery, Moderate-High	Streaming Data, High Message Rates	High Data Durability, High Performance



Soma in Javarevisited

Difference between RabbitMQ, Apache Kafka, and ActiveMQ

Hello folks, if you are preparing for Java Developer interviews the along with Spring Boot, and Microservices, you should also prepare...

🌟 · 8 min read · May 19



208



4



Ajay Rathod in Javarevisited

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

Greetings everyone,

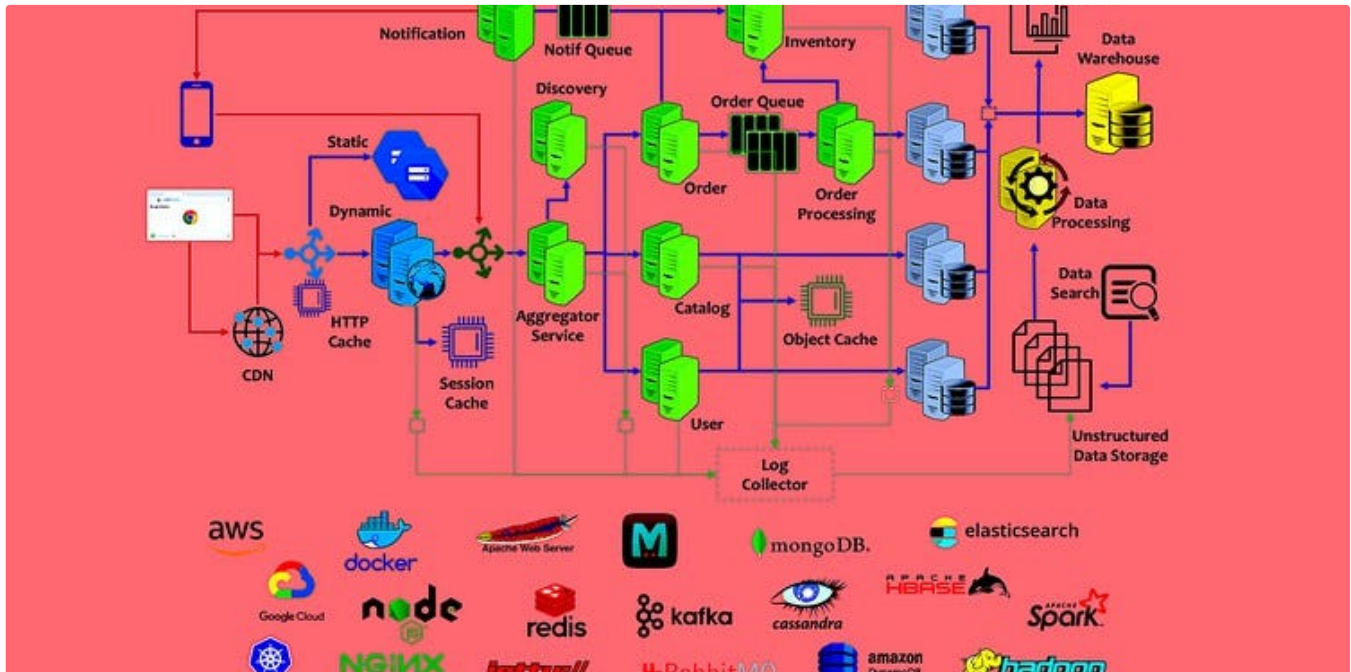
9 min read · 6 days ago



79



1



javinpaul in Javarevisited

My Favorite Udemy Courses to Learn System Design in 2023

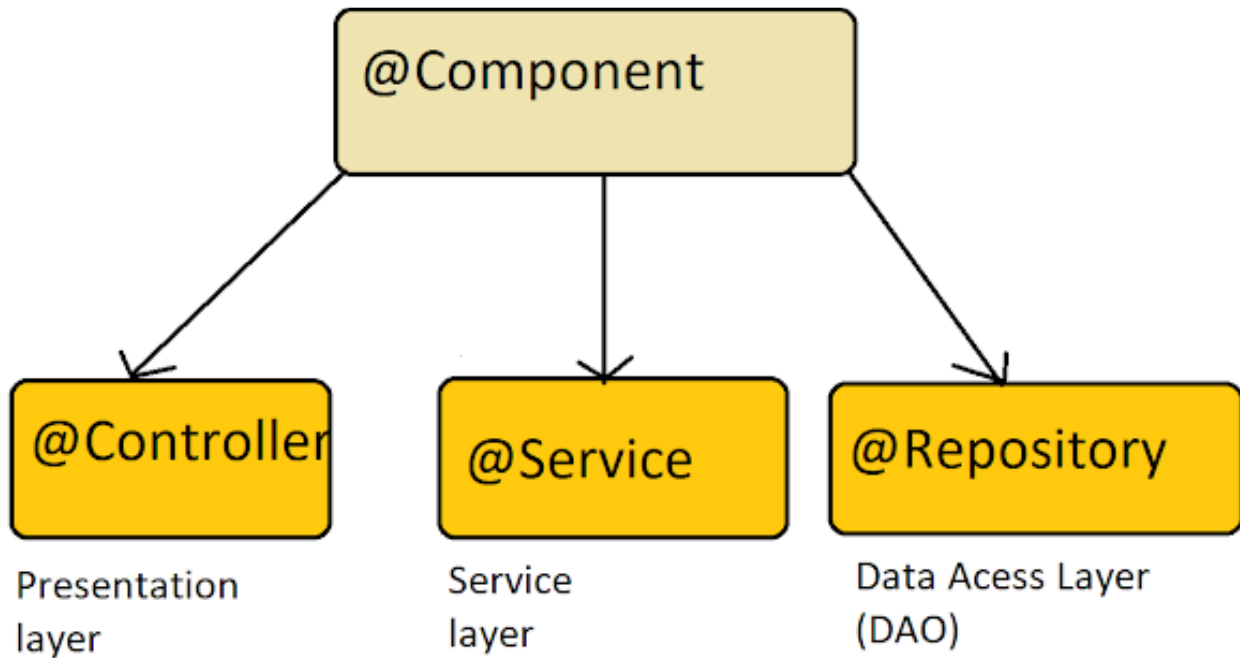
These are the best System Design courses you can join to not only prepare for System Design interviews but also to learn Software...


11 min read · 6 days ago



108





 Soma in Javarevisited

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...

🌟 · 7 min read · Mar 23



238



1



See all from Soma

See all from Javarevisited

Recommended from Medium



Anto Semeraro in Level Up Coding

Microservices Orchestration Best Practices and Tools

Optimizing microservices communication and coordination through orchestration pattern with an example in C#

★ · 10 min read · Mar 27



69



Daniel Zielinski

Senior Java Software Developer Interview Questions—part 1

1. Anemic Model vs Rich Model ?

🌟 · 3 min read · Feb 18



17

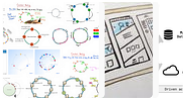


Lists



It's never too late or early to start something

13 stories · 69 saves



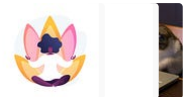
General Coding Knowledge

20 stories · 204 saves



Coding & Development

11 stories · 101 saves



Stories to Help You Grow as a Software Developer

19 stories · 271 saves




Daryl Goh

Spring Boot: RequestEntity vs ResponseEntity | RequestBody vs ResponseBody

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can “just run”.

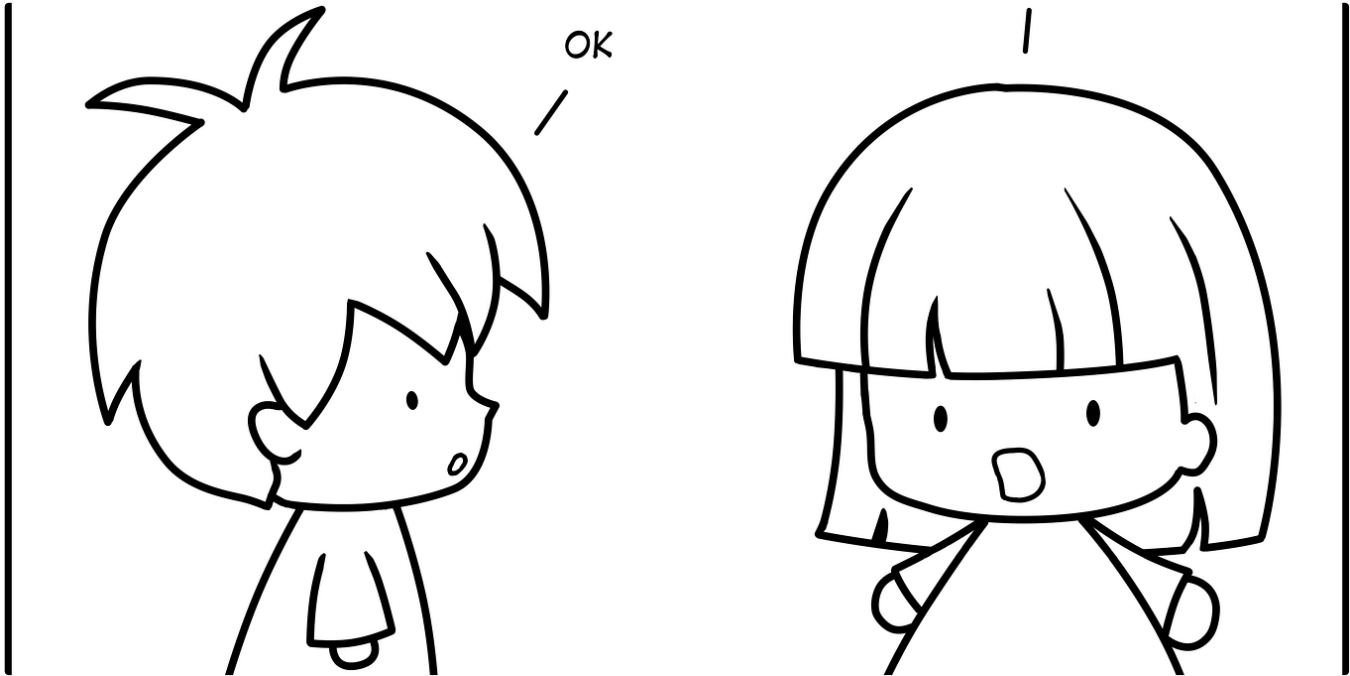
★ · 5 min read · May 21

 40









Julie Zhuo in The Year of the Looking Glass

Average Manager vs. Great Manager

Explained in 10 sketches

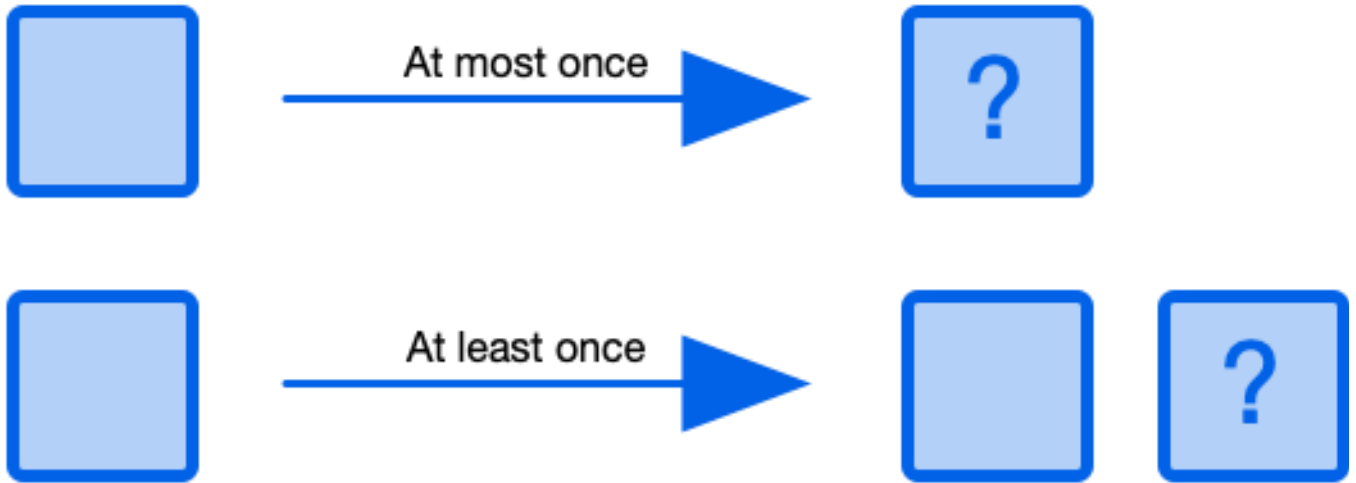
2 min read · Aug 12, 2015

 20K

 186







 Andy Bryant

Processing guarantees in Kafka


Each of the projects I've worked on in the last few years has involved a distributed message system such as AWS SQS, AWS Kinesis and more...

21 min read · Nov 16, 2019

 1.91K  5




 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

 43

 2





See more recommendations