Member-only story

Difference between Spring Framework, Spring Boot, and Spring Cloud?

Spring, Spring Boot, and Spring Cloud are all components of the Spring Framework ecosystem, but they serve different purposes and target different aspects of application development. Here's a breakdown of the differences between them:

Soma · Follow
Published in Javarevisited
10 min read · Jun 23

Listen Share ••• More

	Spring Framework	Spring Boot	Spring Cloud
Purpose	Comprehensive application framework for Java	Streamlined framework for creating Spring applications	Tools and libraries for building distributed systems
Complexity	Moderate	Minimal	Moderate to high
Configuration	Manual	Auto-configuration	Configuration management for distributed systems
Deployment	Requires external server/container	Stand-alone applications with embedded server	Enhances Spring applications in a distributed environment
Features	Dependency injection, AOP, data access, transaction	Opinionated defaults, simplified configuration	Service discovery, load balancing, circuit breakers, tracing
Use Case	Enterprise-level Java applications	Rapid application development, microservices	Cloud-native applications, distributed systems
Integration	Can be used in Spring Boot and Spring Cloud projects	N/A	Can be used with Spring Framework and Spring Boot
Learning Curve	Relatively steep	Moderate	Moderate to steep

Hello folks, if you are preparing for Spring Framework and Java developer interviews and looking for resources like frequently asked spring framework

questions then you have come to the right place.

In the past, I have shared several resources for Java interviews like ***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* , if you haven't checked them yet, I suggest you to look at the them before your interview.

When diving into the vast world of Spring, it's essential to comprehend the distinctions between three key components: Spring Framework, Spring Boot, and Spring Cloud. Although interconnected, each serves a distinct purpose in simplifying and enhancing Java application development.

The Spring Framework is a comprehensive, widely adopted application framework that addresses the complexity of enterprise Java development. It offers a modular approach and tackles various concerns such as dependency injection, aspect-oriented programming, and data access.

Spring Boot, an extension of the Spring Framework, focuses on streamlining the creation and deployment of stand-alone, production-grade Spring applications. It simplifies configuration, auto-configuration, and offers opinionated defaults to expedite development.

Spring Cloud, on the other hand, builds upon Spring Boot and provides tools and libraries for developing and managing distributed systems and cloud-native applications. It addresses challenges like service discovery, load balancing, distributed configuration, and circuit breakers.

Understanding the differences between these components is crucial for effectively utilizing the Spring ecosystem and choosing the right tools to tackle specific development requirements. In this article, we will explore each component in detail, highlighting their unique features and use cases.

Now that you have basic idea of what they are, let's deep dive into each of them and find more more differences to understand them better.

What is Spring Framework? what problem it solves?

The Spring Framework is a popular open-source application framework for building Java applications. It provides comprehensive infrastructure support and a flexible programming model for developing enterprise-level Java applications.

The primary problem that the Spring Framework aims to solve is the complexity of enterprise application development. Traditional Java EE (Enterprise Edition) development often involves writing a significant amount of boilerplate code and dealing with complex configuration. This can make the development process time-consuming and error-prone.

The Spring Framework simplifies and streamlines Java development by offering a lightweight and modular approach. It provides a set of cohesive modules that address various concerns of enterprise application development, such as dependency injection, aspect-oriented programming, transaction management, data access, security, and more.

Here are some key features and benefits of the Spring Framework:

Dependency Injection:

Spring's core feature is the Inversion of Control (IoC) container, which manages object creation and their dependencies. It reduces the coupling between

components and promotes loose coupling, making the code more modular, testable, and maintainable.

Aspect-Oriented Programming (AOP):

AOP is a programming paradigm that allows modularizing cross-cutting concerns. Spring supports AOP by providing declarative mechanisms to add behaviors such as logging, caching, and security to application components.

Simplified Data Access:

Spring provides a consistent abstraction layer over different data access technologies, including JDBC, ORM (Object-Relational Mapping) frameworks like Hibernate, JPA (Java Persistence API), and NoSQL databases. It simplifies database operations and reduces boilerplate code.

Transaction Management:

Spring offers a transaction management framework that supports declarative and programmatic transaction management. It simplifies the handling of database transactions, making it easier to manage and ensure data integrity.

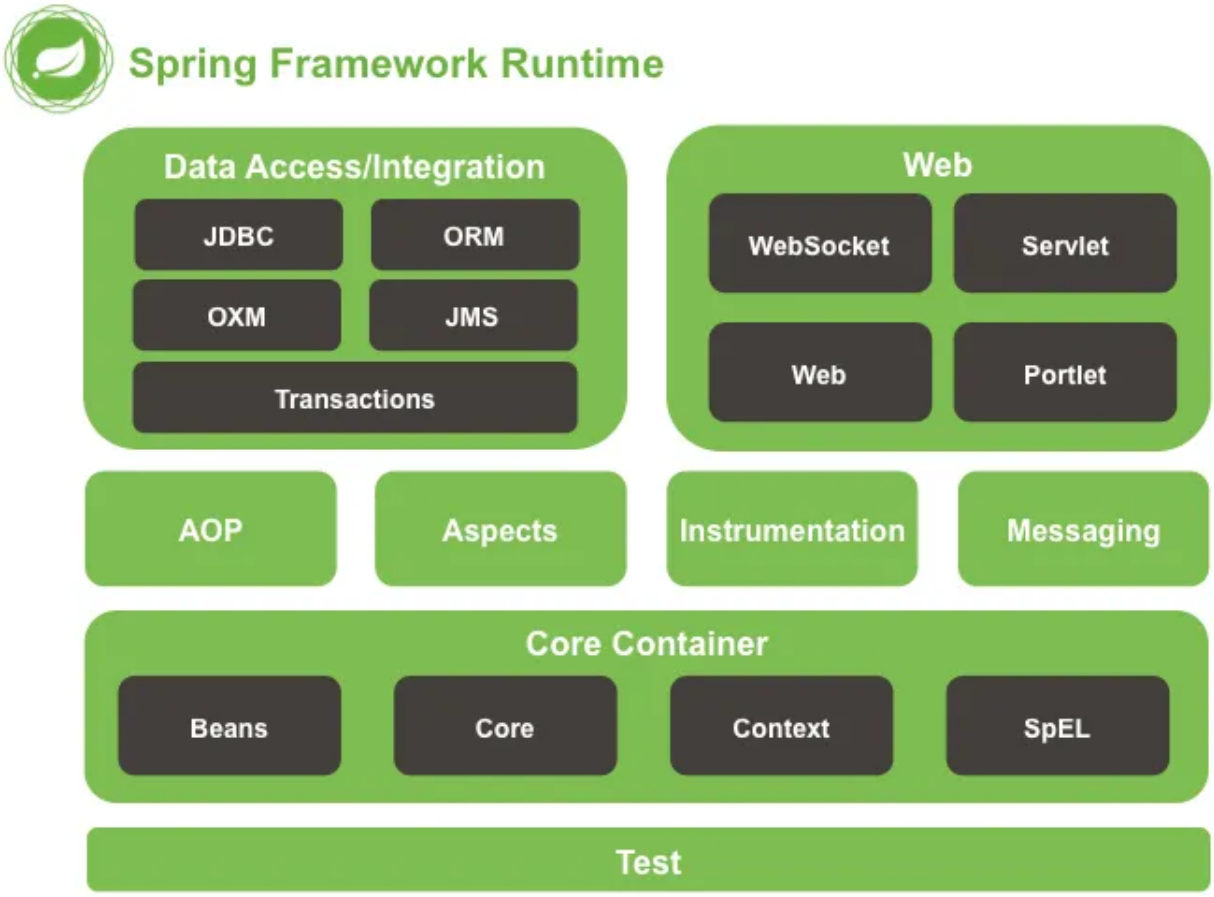
MVC Web Framework:

Spring MVC (Model-View-Controller) is a powerful web framework that provides a clean separation of concerns for building web applications. It supports the development of RESTful services and provides features like request mapping, data binding, and validation.

Integration with other frameworks:

Spring integrates well with other popular frameworks and technologies, such as Java EE containers, Java Message Service (JMS), JavaMail, Quartz Scheduler, and more.

In short, Spring Framework makes Java Development easier then just using plan old JDK and Java APIs. It offer features which enable modern software development like DI and IOC and makes unit testing easier.



What is Spring Boot? What problem it solves?

Spring Boot is an opinionated framework and an extension of the Spring Framework that aims to simplify the development of stand-alone, production-grade Spring-based applications. It provides a streamlined and convention-over-configuration approach, allowing developers to quickly create and deploy applications with minimal boilerplate code and configuration.

The primary problem that Spring Boot solves is the complexity and time-consuming nature of setting up and configuring Spring applications. Traditionally, developers had to manually configure various components and dependencies of a Spring application, which could be a daunting and error-prone task, especially for beginners or when starting new projects.

Spring Boot alleviates these challenges by offering the following key features:

Auto-configuration:

Spring Boot automatically configures the application based on the classpath and the dependencies it detects. It provides sensible defaults and eliminates the need for extensive manual configuration. Developers can override the defaults and fine-tune the configuration as needed.

How does auto-configuration work in Spring Boot?

Demystifying the Magic: Understanding the Inner Workings of Spring Boot's Auto-Configuration Feature

medium.com

Starter Dependencies:

Spring Boot provides a collection of "starter" dependencies that simplify the management of required dependencies. These starters include pre-configured dependencies for specific features or technologies, such as web applications, databases, security, messaging, and more. Developers can include the relevant starters in their project, and Spring Boot takes care of setting up the necessary dependencies.

Embedded Servers:

Spring Boot embeds servlet containers (like Tomcat, Jetty, or Undertow) directly into the application, eliminating the need for deploying the application on a separate server. This simplifies the deployment process and makes the application self-contained and easily executable.

Production-Ready Features:

Spring Boot provides a range of production-ready features out of the box. It includes metrics, health checks, monitoring, and management endpoints that enable efficient monitoring and management of the application. It also supports externalized configuration, allowing easy configuration management without modifying the application code.

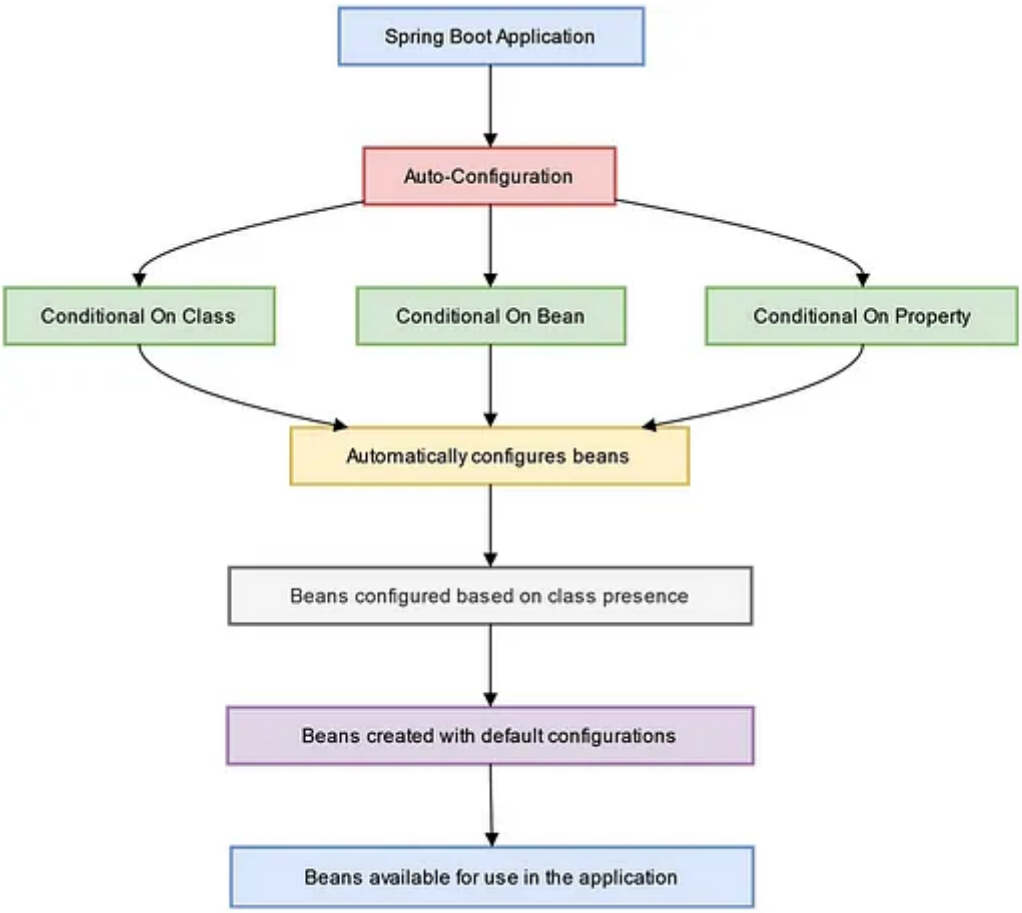
DevTools:

Spring Boot offers a set of developer tools that enhance the development experience. It includes automatic application restart, live reload of changes, enhanced logging, and other features that speed up the development cycle.

By leveraging Spring Boot, developers can rapidly create production-ready applications with minimal effort. It eliminates the need for manual configuration,

reduces boilerplate code, and provides a set of tools and features that streamline the development and deployment process.

Spring Boot is particularly well-suited for microservices architecture, cloud-native development, and building standalone, self-contained applications.



What is Spring Cloud? What problem it solves?

Spring Cloud is a framework that provides a set of tools and libraries to build distributed systems and cloud-native applications. It builds on top of the Spring Boot framework and leverages its features to address the challenges of developing and managing distributed applications in a cloud environment.

The primary problem that Spring Cloud aims to solve is the complexity of developing and managing distributed systems. As applications are increasingly deployed in cloud environments and designed as microservices, developers face challenges such as service discovery, load balancing, distributed configuration management, circuit breakers, and more.

Spring Cloud provides abstractions and components that simplify the implementation of these distributed system patterns.

Here are some key features and components provided by Spring Cloud:

Service Discovery:

Spring Cloud integrates with service discovery systems like Netflix Eureka, Consul, or Kubernetes Service Discovery. It allows services to register themselves and discover other services dynamically, enabling flexible and resilient communication between services.

Service Registry Design Pattern in Microservices Explained

Discover, Load Balance, and Scale: Understanding the Service Registry Pattern in Microservices Architecture

medium.com

Load Balancing:

Spring Cloud includes client-side load balancing capabilities. It integrates with libraries like Netflix Ribbon and provides out-of-the-box load balancing algorithms to distribute requests across multiple instances of a service.

Difference between API Gateway and Load Balancer in Microservices?

Hello folks, what is difference between API Gateway and Load balancer is one of the popular Microservice interview...

medium.com

Distributed Configuration

Spring Cloud provides centralized and dynamic configuration management. It allows applications to retrieve their configurations from a distributed configuration server, such as Spring Cloud Config, enabling easy and dynamic configuration updates without the need for redeployment.

Circuit Breakers

Circuit breakers help to handle and prevent cascading failures in distributed systems. Spring Cloud integrates with libraries like Netflix Hystrix to provide circuit breaker patterns that allow services to handle failures and timeouts gracefully.

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

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

medium.com

Distributed Tracing

Spring Cloud integrates with distributed tracing systems like Zipkin or Jaeger. It enables monitoring and tracing requests across multiple services, providing insights into the performance and behavior of the entire distributed system.

How to manage transactions in Distributed Systems and Microservices?

In this article you will learn about 3 ways to manage transactions in distributed systems and microservices such as...

medium.com

API Gateway

Spring Cloud includes an API gateway component, such as Netflix Zuul or Spring Cloud Gateway. It acts as a single entry point for client requests, providing routing, filtering, and load balancing capabilities.

What is API Gateway Pattern in Microservices Architecture? What Problem Does it Solve?

The API Gateway can help in managing authentication, request routing, load balancing, and caching in Microservices...

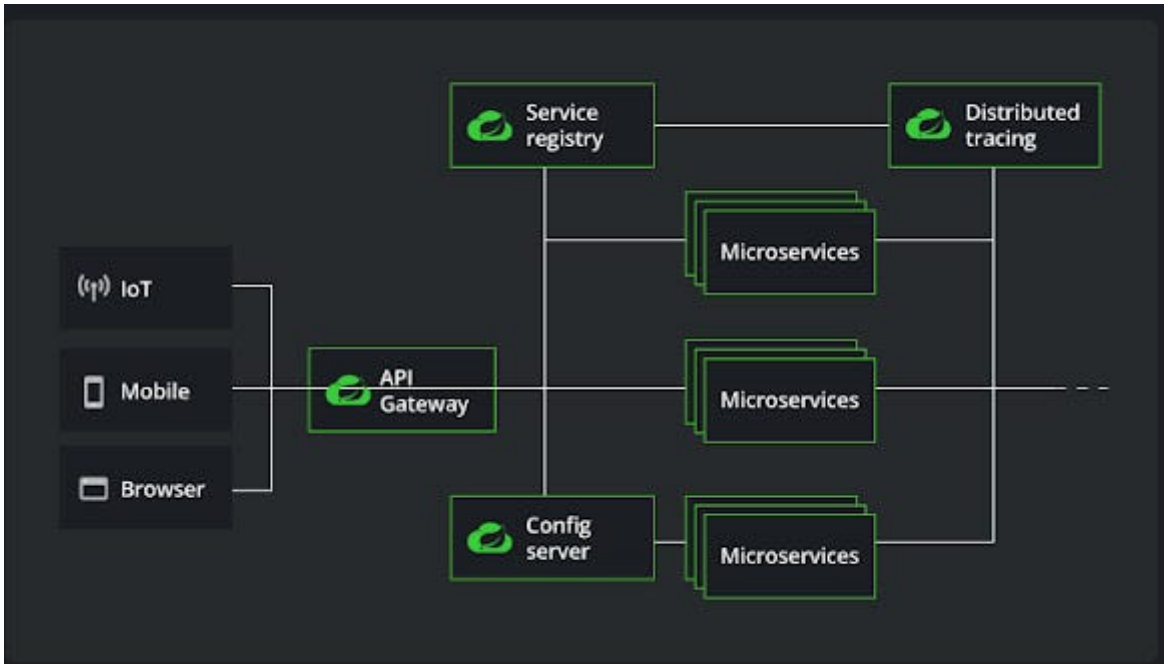
medium.com

Service Mesh Integration

Spring Cloud integrates with popular service mesh frameworks like Istio or Linkerd. It allows seamless integration of service mesh capabilities such as traffic management, observability, security, and resiliency into Spring-based applications.

By leveraging Spring Cloud, developers can focus on building the business logic of their distributed applications while abstracting away the complexities of distributed system patterns. It promotes resilience, scalability, and maintainability of

distributed systems, making it easier to develop, deploy, and manage cloud-native applications.



Difference between Spring Framework, Spring Boot, and Spring Cloud

In summary, the main differences between Spring, Spring Boot, and Spring Cloud are as follows:

Scope

Spring Framework provides the foundational modules for building Java applications, whereas Spring Boot simplifies the development of stand-alone Spring applications with a focus on convention-over-configuration. Spring Cloud, on the other hand, focuses on building distributed systems and microservices architectures.

Configuration

Spring Framework requires explicit configuration and setup for various components, whereas Spring Boot provides auto-configuration, reducing the need for manual configuration. Spring Cloud provides additional configuration options for distributed systems, such as service discovery and distributed configuration management.

Development Approach

Spring Framework follows a traditional development approach with explicit configuration and flexibility for customization. Spring Boot promotes rapid application development with convention-over-configuration and starter dependencies. Spring Cloud provides tools and libraries for building distributed systems and microservices architectures.

And, if you like tables for comparison, here is an ice table which compares Spring Framework, Spring Boot, and Spring Cloud into various parameters:

	Spring Framework	Spring Boot	Spring Cloud
Purpose	Comprehensive application framework for Java	Streamlined framework for creating Spring applications	Tools and libraries for building distributed systems
Complexity	Moderate	Minimal	Moderate to high
Configuration	Manual	Auto-configuration	Configuration management for distributed systems
Deployment	Requires external server/container	Stand-alone applications with embedded server	Enhances Spring applications in a distributed environment
Features	Dependency injection, AOP, data access, transaction	Opinionated defaults, simplified configuration	Service discovery, load balancing, circuit breakers, tracing
Use Case	Enterprise-level Java applications	Rapid application development, microservices	Cloud-native applications, distributed systems
Integration	Can be used in Spring Boot and Spring Cloud projects	N/A	Can be used with Spring Framework and Spring Boot
Learning Curve	Relatively steep	Moderate	Moderate to steep

That's all about **difference between Spring Framework, Spring Boot, and Spring Cloud in Java world**. Overall, Spring Framework, Spring Boot, and Spring Cloud work together to provide a robust ecosystem for developing Java applications, catering to different aspects of application development, from the core framework to rapid application development and building distributed systems.

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

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**

Join Medium with my referral link - Soma

As a Medium member, a portion of your membership fee goes to writers you read, and you get full access to every story...

medium.com

Other **Java and Microservices articles** you may like

What is difference between @Component and @Bean annotation in Spring?

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

medium.com

Top 20 Spring Framework and Spring Boot Scenario based Questions for Java Developers

Mastering Practical Application of Spring and Spring Boot with These Real-World Interview Questions for Java Developers

medium.com

50+ Spring Boot Interview Questions and Answers for Java Programmers

These are 50+ Spring Boot Interview Questions and answers to Crack your Next Java Developer interview

medium.com

Java Spring Spring Boot Microservices Spring Framework

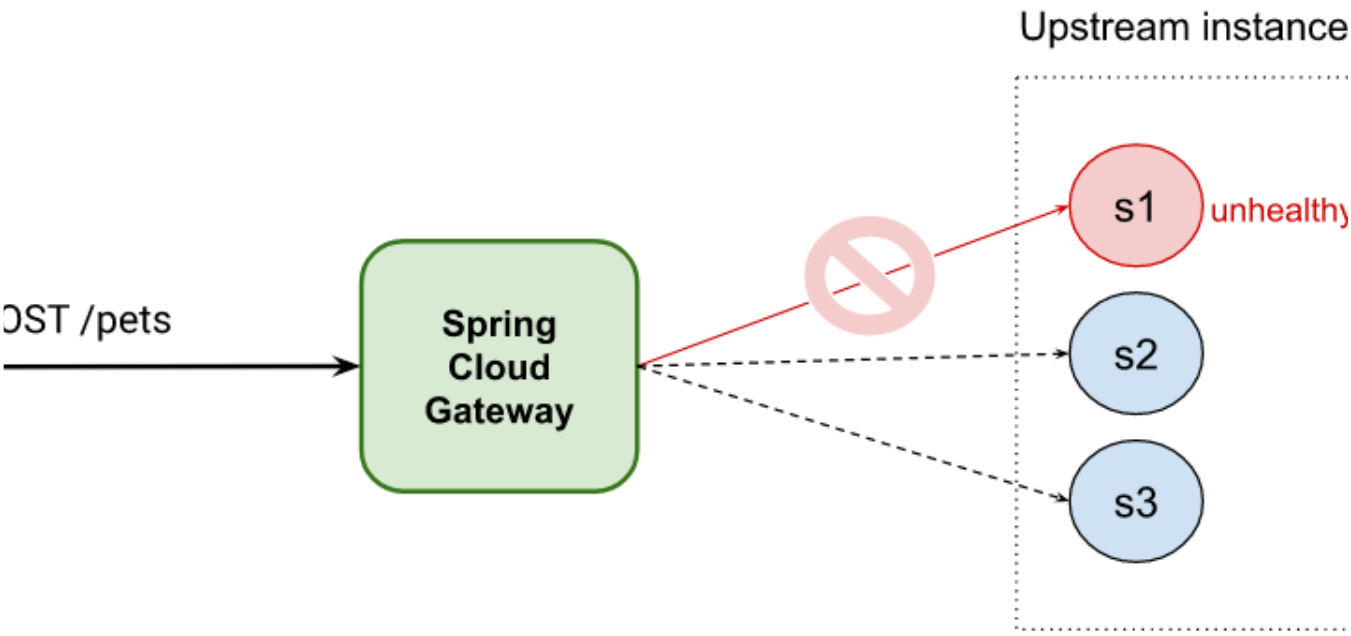
Follow

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



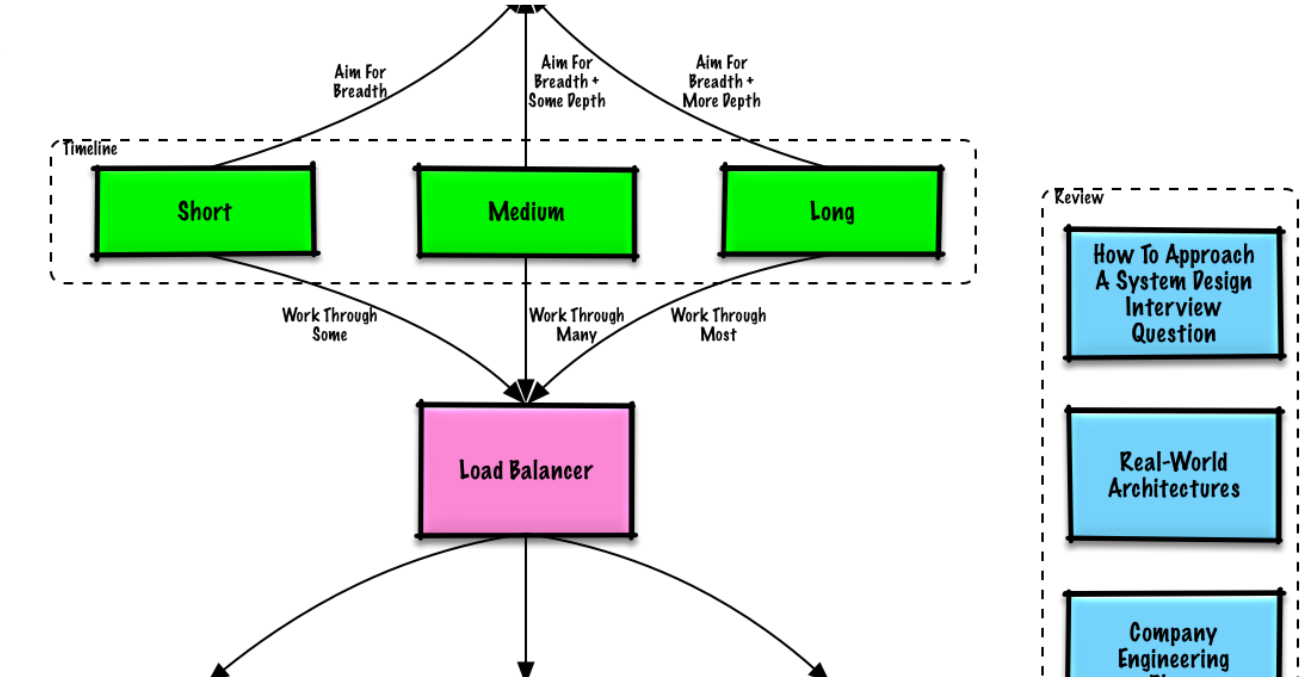
Soma in Javarevisited

Top 20 Spring Cloud Interview Questions with Answers for Java Developers

Common Spring Cloud Questions for Java Interviews with answers

✦ · 9 min read · Jul 23

184



javinpaul in Javarevisited

10 Best System Design Courses for Coding Interviews in 2023

These are the best online courses to learn System Design from Udemy, edX, and Educative for coding interviews in 2023.

18 min read · Oct 22, 2020

1K 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

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

See all from Soma

See all from Javarevisited

Recommended from Medium

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

30 2

Hamza Nassour in Javarevisited

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

ApplicationContext creation/registration , AutoConfiguration .. .

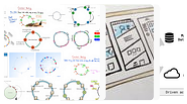
4 min read · Feb 19

93

Lists

It's never too late or early to start something

13 stories · 67 saves

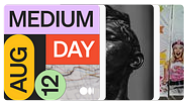


General Coding Knowledge

20 stories · 193 saves

New_Reading_List

174 stories · 61 saves



Staff Picks

393 stories · 211 saves

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

16

Daniel Zielinski

Decorator Design Pattern. Real live Java example with Spring Boot.

Decorator Design Pattern. Real live Java example with Spring Boot.

✨ · 6 min read · Feb 25

19 1

kiarash shamaii

Logging in Five minutes

The process of storing application execution details to a file is called as Logging. With log messages we can understand execution flow of...

8 min read · 6 days ago

14

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

See more recommendations