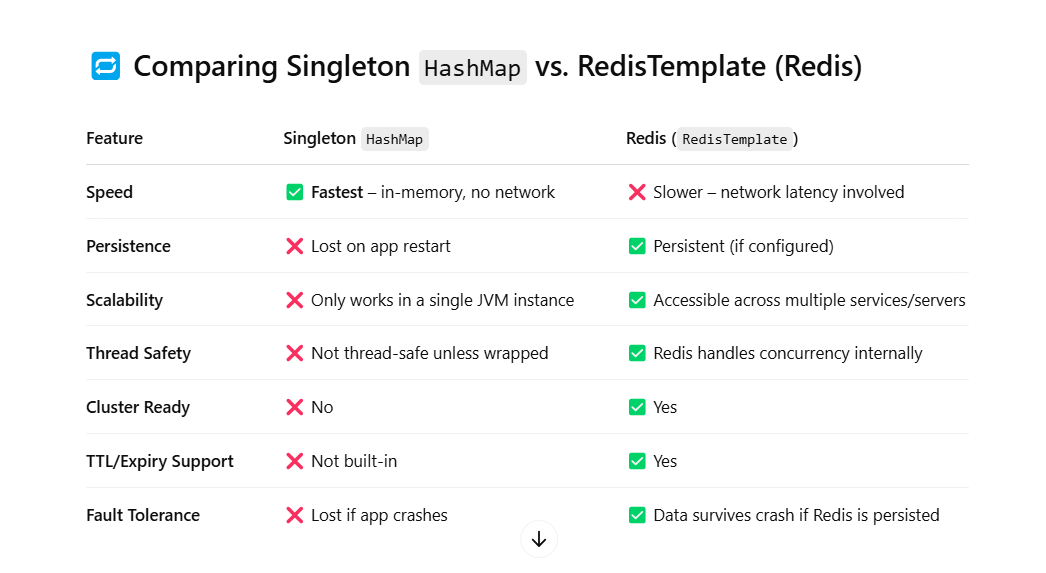
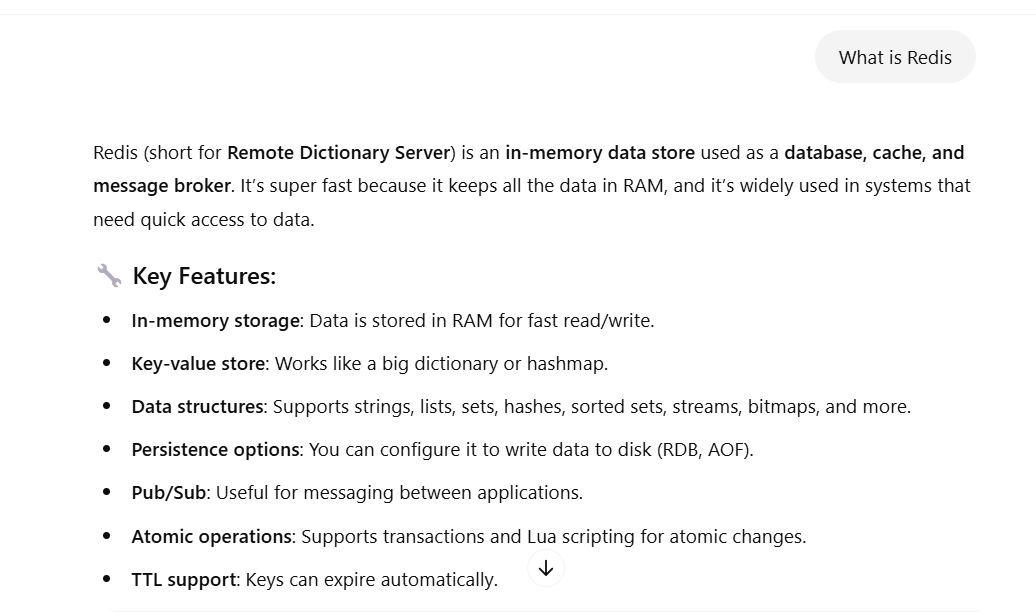
Q: Comparing Singleton Cache Data VS Redis Cache?



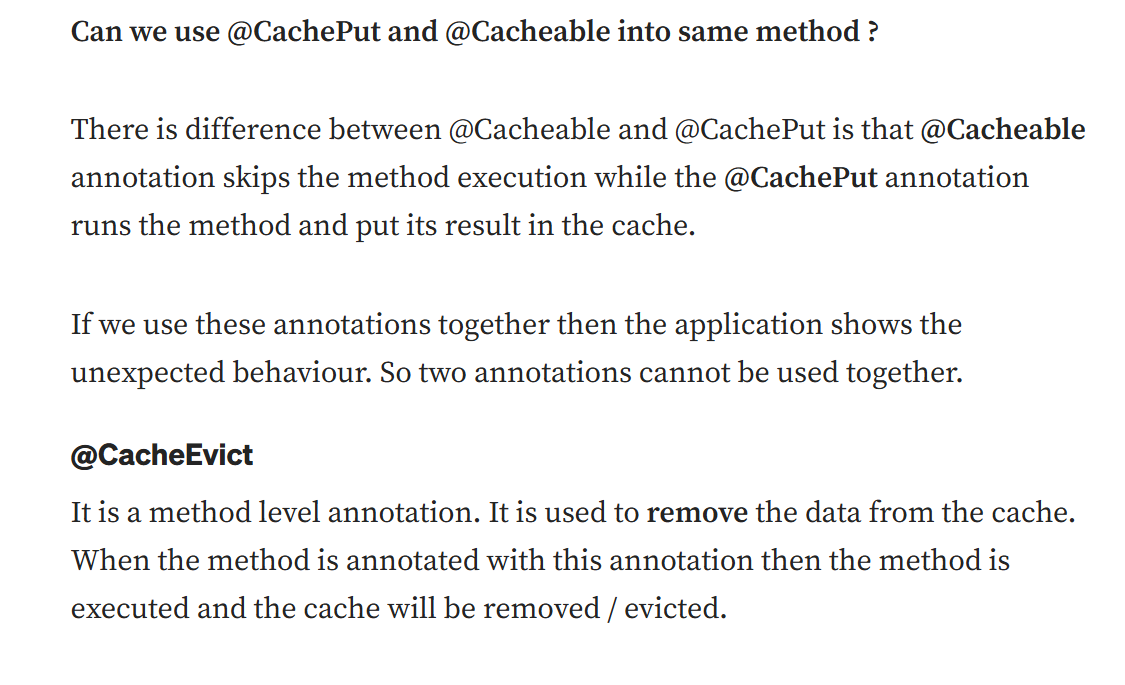
Q: What is Redis?

A:- \*\*\* Use as a database, **cache**, **message broker**.



Q: Can we use **@CachePut** and **@Cacheeable** on the same method?

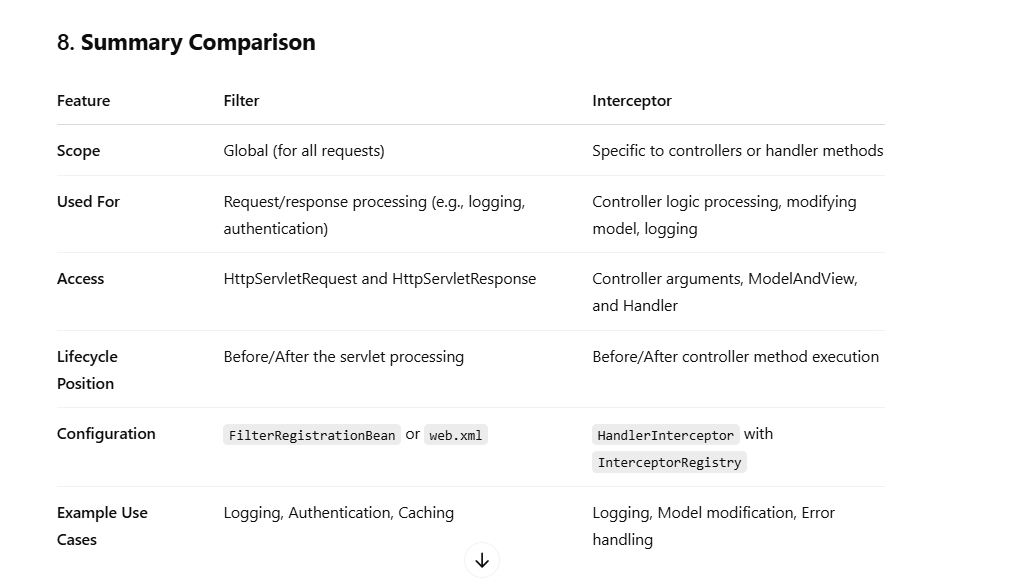
A: \*\*\* **@Cacheable** will be ignored, and **@Cacheput** will be used to perform the function.



Q: Different b/w **Filter** and **Interceptor** in Spring Boot.

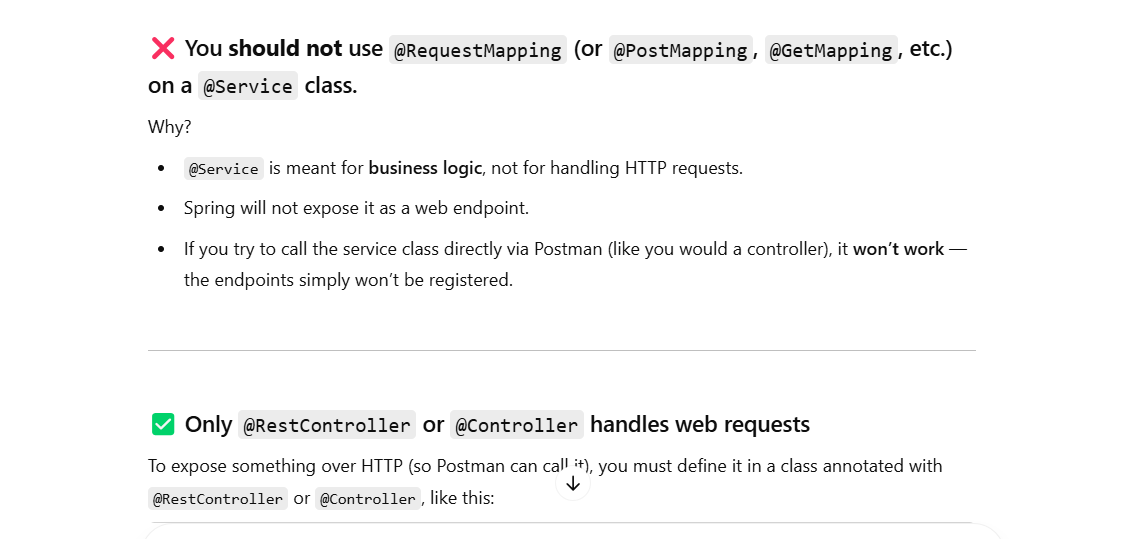
A: \*\*\* [**Servelet**] Before/After the **Servelet** processing

[**Interceptor**] Before/After **Controller** method execution



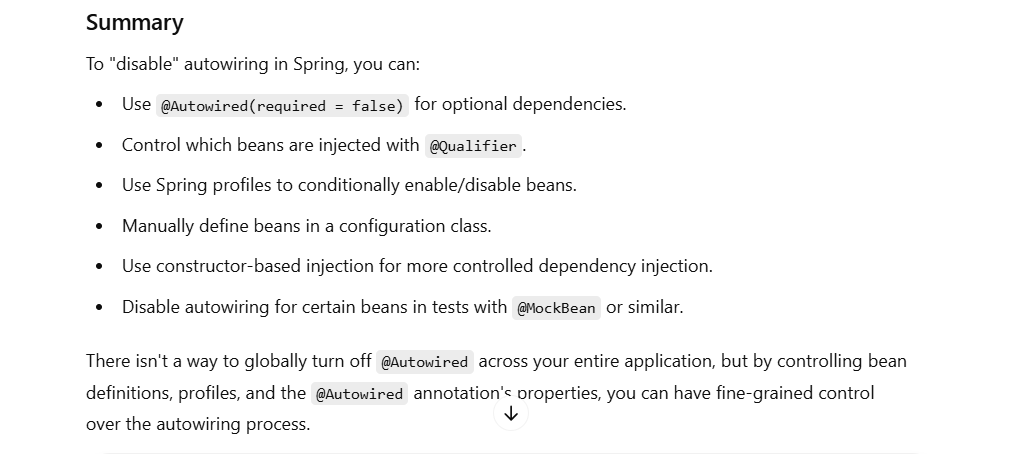
Q: Can we use **@RequestMapping** and **@PostMapping** or **@GetMapping** in the **@Service**?

A: \*\*\* You can use but this Spring **will not expose the endpoint,** will not work, and the method not invoke unless **@Controller** and **@RestController** use.



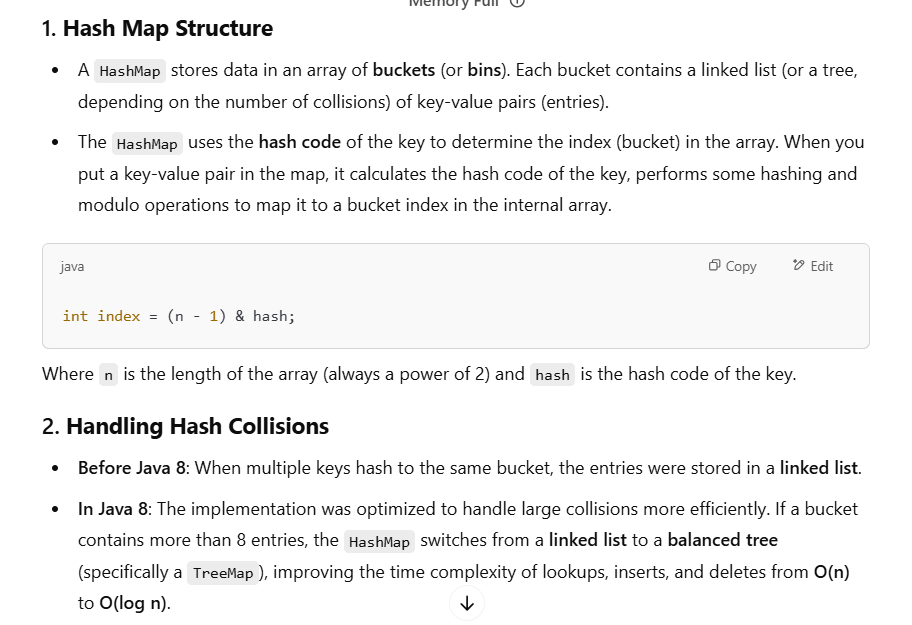
Q: Can we disabled @Autowired?

A: **\*\*\*** There isn’t a way to globally turn off the @Autowired but we can disabled from other way.



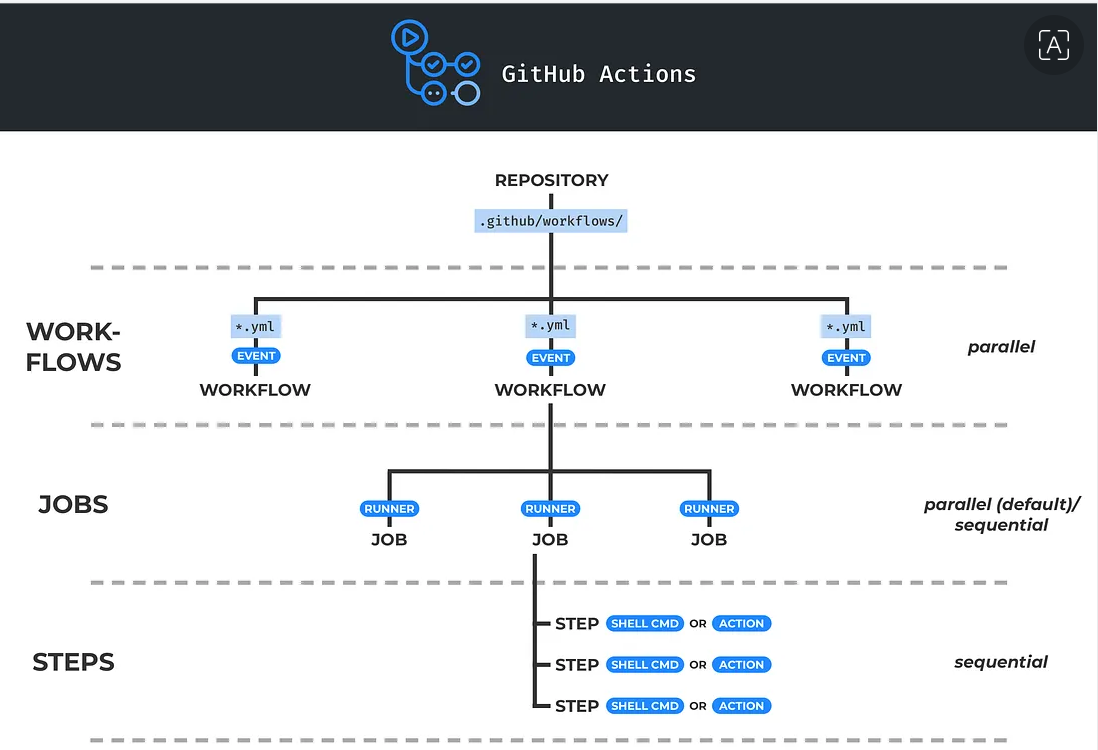
Q: How HashMap Work?

A: \*\*\* **Array of bucket and Each bucket have Linked list or Tree depend on the number of collision of key value paire**.



Q: What is Github Actiosn?

A: Github use use to automate the CI/CD.



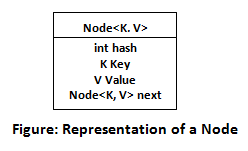
Q: Write a program to get the number from the String.

A: Here is the example for extract the value from the patter

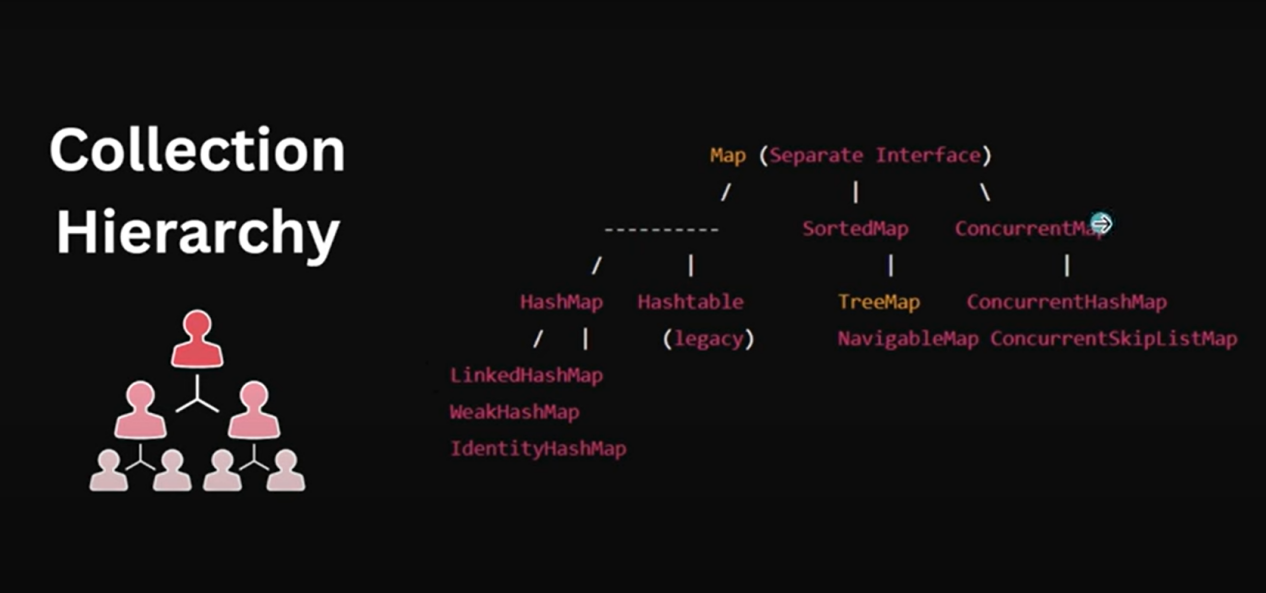


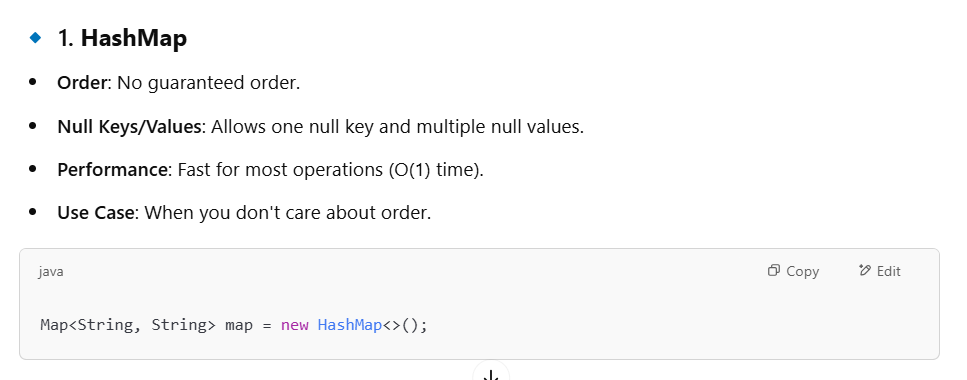
Q: Can you expalin the innternal filed of HashMap?

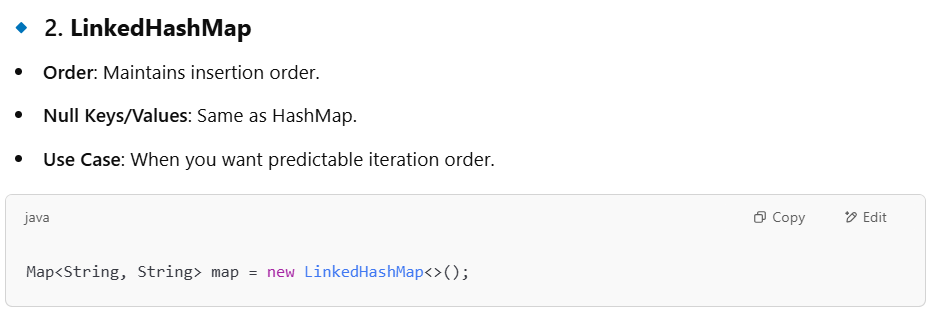
A: Here is the **bucket link list** case.

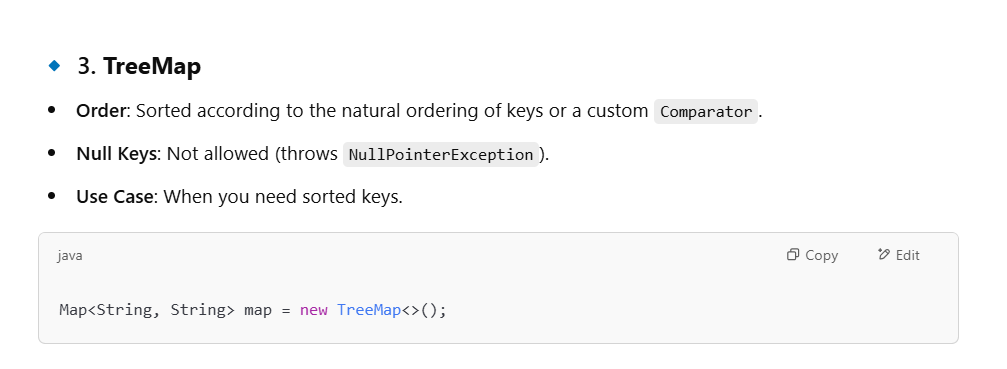


Q: HashMap Collection?

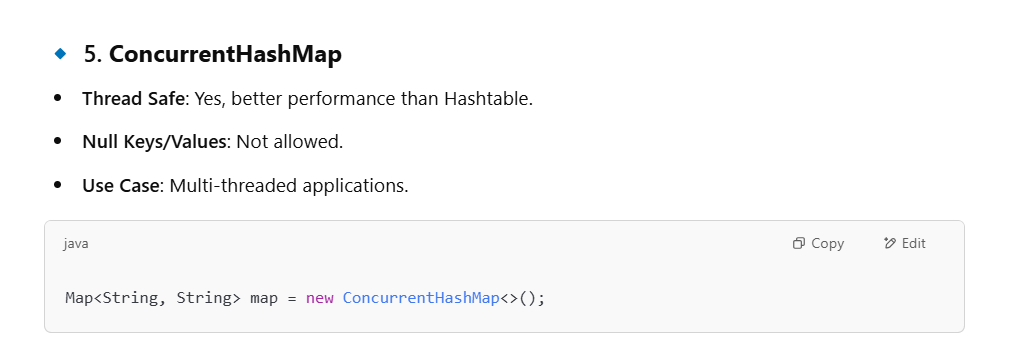


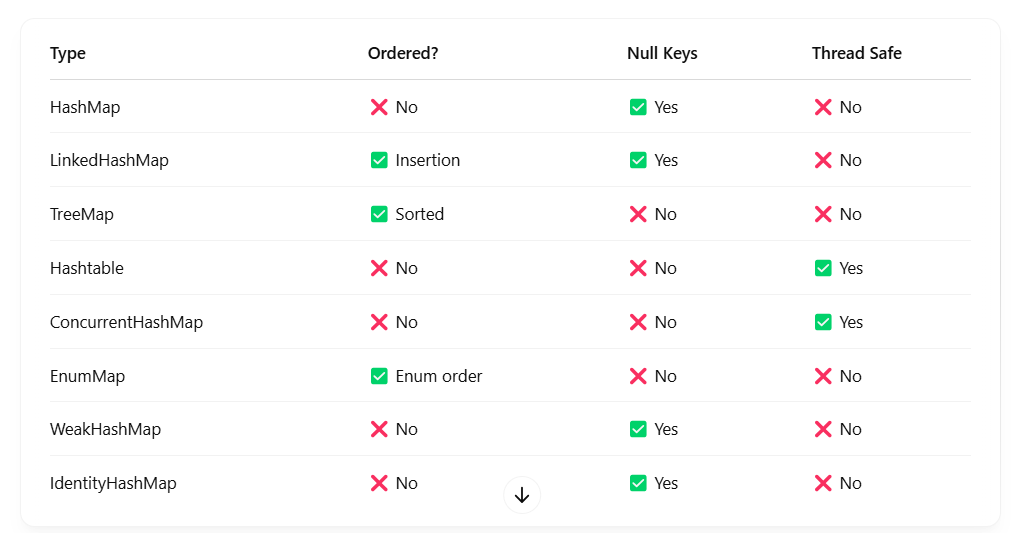






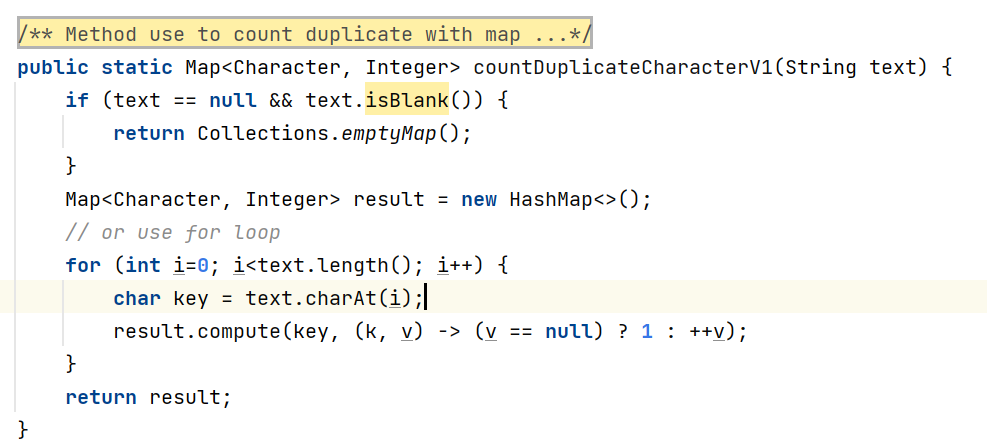
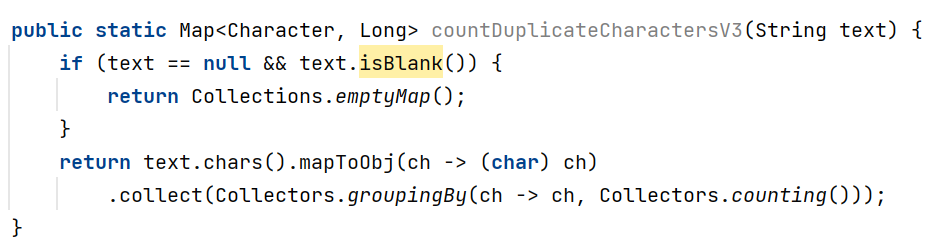






Q: Count duplicate character?

A: Example of counting duplicate characters with **grouping** and **computing**.



Q: Segment-based Locking in Java 7 (ConcurrentHashMap)

A: In Java 7 and earlier, ConcurrentHashMap internally used segment-based locking to allow concurrent access with minimal locking contention.

1. ConcurrentHashMap is **divided into segments (default: 16 segments)**.
2. Each segment is like a smaller **synchronized** HashMap.
3. **Each segment has its own lock**, so 16 threads can write to different segments concurrently.
4. This improves concurrency compared to synchronizing the whole map.

How Locking Works:

**Read operations** (like get()) are lock-free, so they're very fast and thread-safe without blocking.

**Write operations** (like put(), remove()) acquire a lock on the specific segment that contains the key’s bucket. Other threads can still operate on different segments simultaneously.

Improvements in Java 8:

In Java 8, ConcurrentHashMap was rewritten:

No segments. It now uses **bucket-level locking** with **CAS (Compare-And-Swap)** for better scalability.

Uses **Node arrays, linked lists, and tree bins (like TreeMap)** internally for better performance.

Q: Why Do SSL Errors Happen?

A: There are a few common triggers:

❌ The SSL certificate is expired.

❌ It’s self-signed or not issued by a trusted Certificate Authority.

❌ The domain name doesn’t match the certificate.

❌ The certificate chain is incomplete or misconfigured.

Q: How I add encription on dabase filed using java?

A: We can use crypto.Cipher

@Convert(converter = EncryptionConverter.class)

private String email;

String encryptedEmail = **encryptionConverter.convertToDatabaseColumn("user@example.com")**;

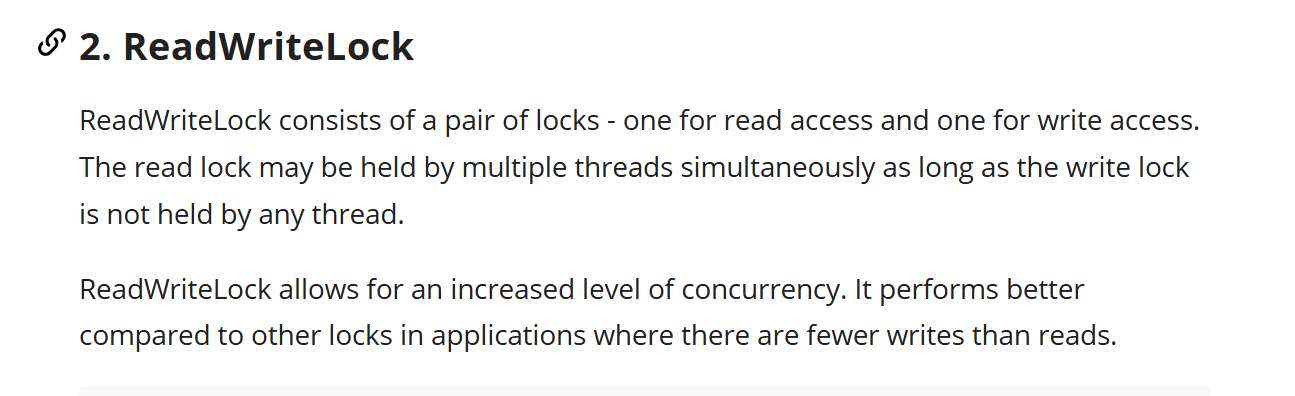
User user = userRepository.findByEmail(encryptedEmail);



Q: What is **ReadWrite Lock**?

A: A **ReadWrite Lock** in Java is a concurrency utility that allows **multiple threads to read** a shared **resource simultaneously,** but **only one thread to write to it exclusively**. This is **useful when reads are more frequent than writes**. it helps to improve performance and reduce contention.

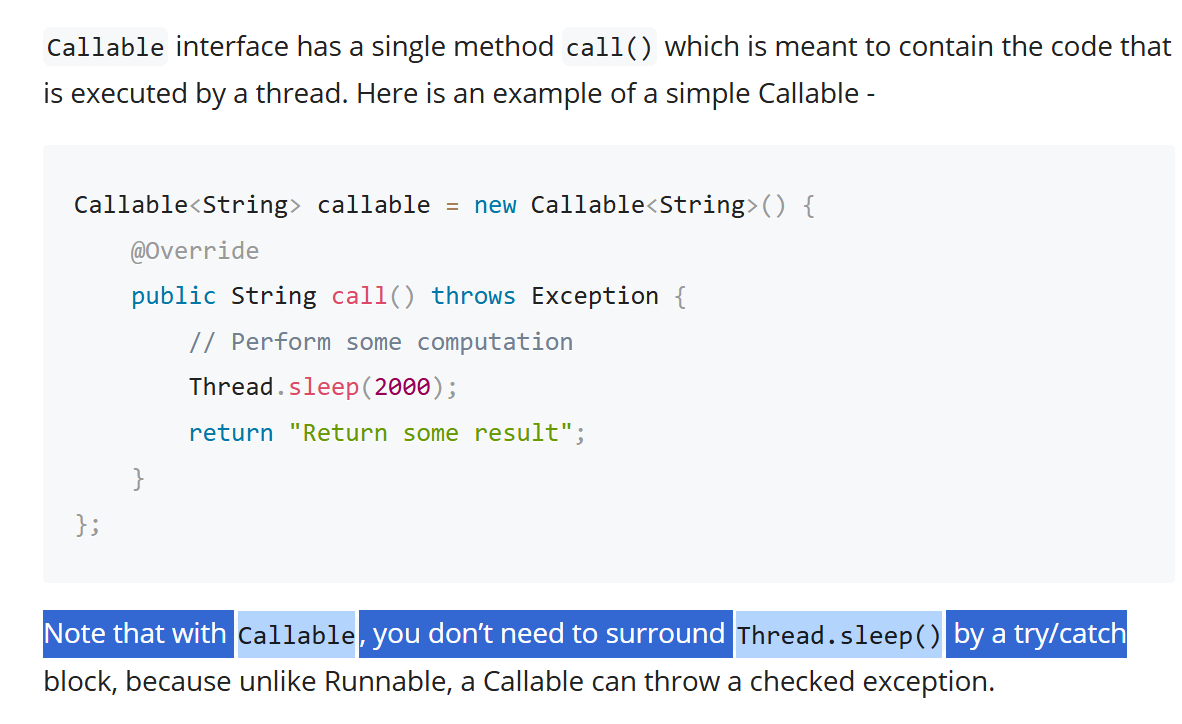
Java provides this through the ReadWriteLock interface, and the most commonly used implementation.





Q: What is Callable interface?

A: \*\*\* **Key point we can return any type of data or throw check excpetion as well.**



Q: Different b/w JNDI and Application properites?

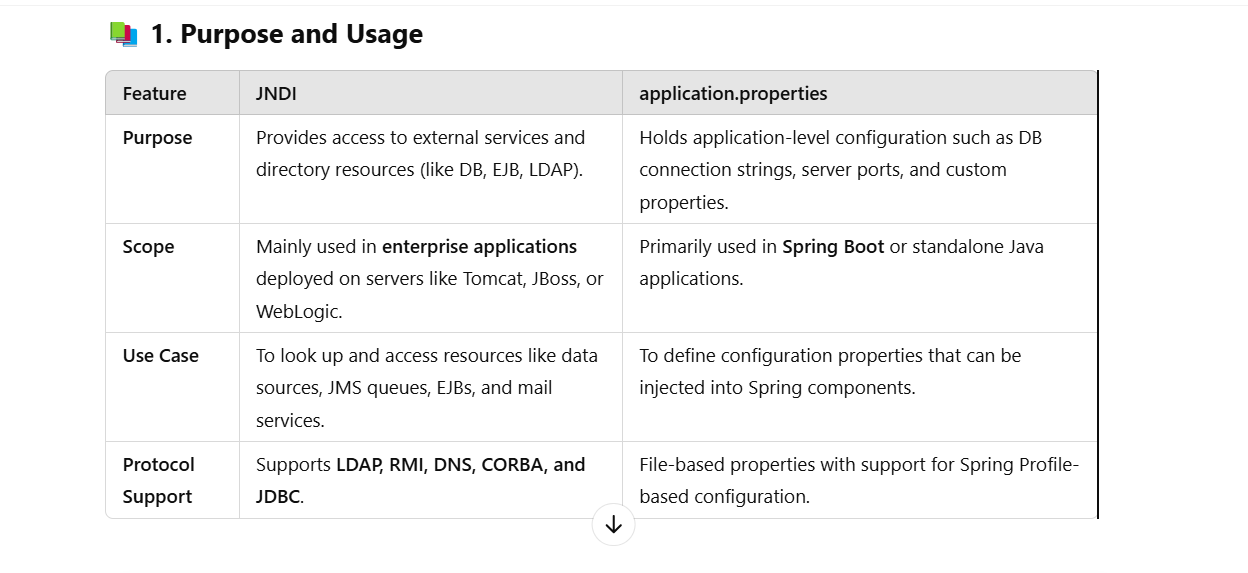
A:

* JNDI

1. JNDI (**Java Naming and Directory Interface**) is used to look up **resources**
2. JNDI lookup or **@Resourc**
3. Standard in Java EE

* Application Properties

1. **@Value, @ConfigurationProperties**
2. Tied to Spring Boot.



Q: What is Maven?

A: Maven is a build automation and project management tool **primarily used for Java projects**.

It simplifies the process of managing **project dependencies**, **building source code**, **packaging applications**, and more.

It automatically downloads and includes required libraries (dependencies) from a central repository (https://repo.maven.apache.org/maven2). **You declare dependencies in a pom.xml** (Project Object Model) file. **It compiles code, runs tests, and packages applications (e.g., JARs, WARs)**.

You just run commands like mvn clean install or mvn package.

Q: What is local m2?

A: **.m2 is a hidden directory located in your user’s home folder**. It acts as a local cache for all the Maven dependencies and plugins your projects need. When you build a project with Maven, **it looks for dependencies in .m2/repository first. If not found locally, it downloads them from the central Maven repository** (like https://repo.maven.apache.org) and stores them in .m2.

This speeds up builds and avoids repeated downloads.

Q: What is **setting.xml file in m2**?

A: Global settings (for all users):

${MAVEN\_HOME}/conf/settings.xml

User-specific settings (for your machine):

~/.m2/settings.xml

(That's the one we usually edit.)

Q: What is docker?

A: Docker is a platform that helps developers and system administrators **build, ship, and run applications.** **Docker file create the docker image and docker container run the image.**

Q: What is Docker Compose?

A: Docker Compose is a tool that helps you **define and run multi-container applications**.

Q: What is mounting in Docker?

A: In Docker, **mounting means connecting a folder on your host machine** (your laptop or server) to a folder inside the Docker container.

**This is usually done with a volume or a bind mount** — both allow files to be shared between the host and the container.

Q: Why use mounting?

A: Because **containers are temporary** — **once they stop or are removed, any data inside is gone unless you mount a persistent folder**.

1) Save logs, uploads, or database files outside the container.

2) Share configuration or source files with the container.

3) Watch logs in real-time from your host.

**Dockerfile**: **A script that tells Docker how to build your image** (e.g., which base image to use, what to install, and what commands to run).

**Image**: **A snapshot of your app and its environment.** You build an image and then run containers based on it.

**Container**: A lightweight, **isolated environment for running an app**. Think of it like a **mini virtual machine**, but faster and usesless resources.

**Docker Hub**: **A cloud registry where you can find and share Docker images.**

Q: What is Docker Compose?

A: Docker compose is a tool to define and run multi-container docker applications. With docker compose, you configure all your application’s services (containers) in a single yaml file, then spin up all of them with a single command.