In **Spring Boot**, both application.propertreies and application.yml (or .yaml) are used for external configuration. They serve the same purpose but differ in syntax and structure.

application.properties

server.port

key=value

**--------------------------------------------------------------------------------------------------------------**

**\*\*\*Exclude Tomcat**

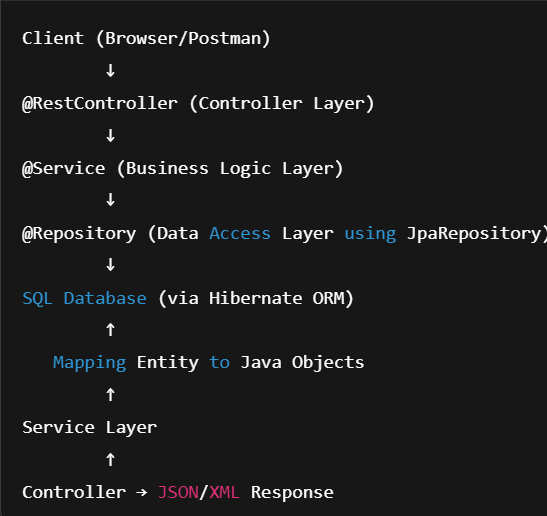
Tomcat is included by default via spring-boot-starter-web. You need to exclude it.

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-jetty</artifactId>

</dependency>



A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.A black screen with white text

AI-generated content may be incorrect.

**== always checks reference equality** — it literally checks whether two variables point to the exact same object in memory.

**.equals() method by default (in Object class) also checks reference equality** — yes, *initially* both behave the same **if you don’t override .equals()** in your class.1

**Why use volatile?**

In a multi-threaded environment, each thread may cache variables locally. Without volatile, updates made by one thread may not be visible immediately to others.

* When you pass arguments to methods in Java, **the method receives a copy of the value**, not the original variable itself.
* For **primitive types** (int, double, boolean, etc.), the actual value is copied.
* For **objects**, the **reference to the object** is copied — but *the reference itself* is passed by value.

A screenshot of a black box

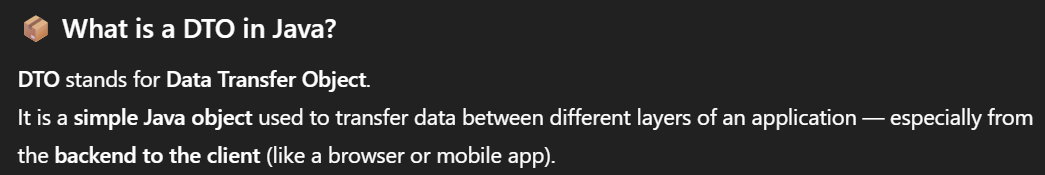
AI-generated content may be incorrect.

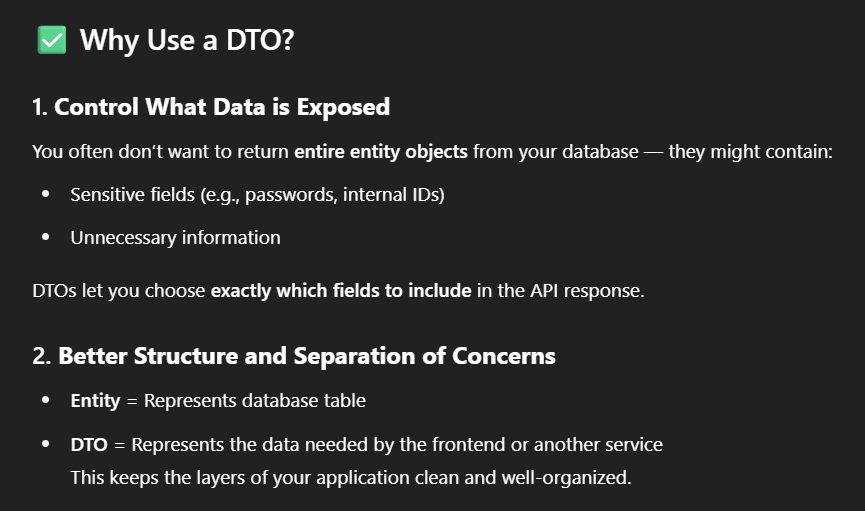
A black background with white text

AI-generated content may be incorrect.

A screenshot of a computer code

AI-generated content may be incorrect.





A black screen with white text

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

In Java, a **ClassLoader** is part of the Java Runtime Environment that **loads classes into memory** when they are needed at runtime.

Bootstrap, extension, application, custom

**Multithreading** is a Java feature that allows **concurrent execution** of two or more parts of a program (called **threads**) for **maximum CPU utilization**.