LOGGING

INFO - Just gives the information

ERROR - gives error

WARN - gives warning

OFF - turn off logging

TRACE - for stack trace

FATAL - for severe issues

Frameworks ::

Java logging API - By default it has this in jdk(built -in)

Apache log4j -

SLF4j -

LogBack -

Flogger by Google(not using this much) -

**Java logging API:**

Logger is a built in class in package java.util.Logging.

private static final Logger logger = Logger.getLogger(MyApp.class.getName());

This line creates a logger instance for the class MyApp, using Java's built-in logging API.

| **Part** | **Meaning** |
| --- | --- |
| private | The logger is only accessible inside the class MyApp |
| static | Shared across all instances of the class (no need to recreate it) |
| final | Cannot be reassigned once initialized (a best practice) |
| Logger | Java’s built-in logging class from java.util.logging |
| logger | The variable name (you can name it anything, but logger is conventional) |
| Logger.getLogger(...) | Static method that returns a logger instance |
| MyApp.class.getName() | Passes the fully qualified class name (e.g., com.example.MyApp) to label log messages from this class. It helps identify **which class** the log message came from. |

By default java logs in console.

Eg:

A computer screen shot of a program

AI-generated content may be incorrect.

* Jdk has java logging api in its library by default but to enable or use slf4j we need to add external jars explicitly like this in java project…Right click on project -> build path-> configure build path-> under libraries-> select class path -> click on ‘Add external jars’ -> then add jars -> Apply -> you will see them under class path.

**In Java Maven Project** : We add these slf4j dependencies in pom.xml , maven will automatically searches for corresponding jars in repositories and add them in project under maven dependencies in project.( no need to manually download or add them if its maven project)

**🔁 Maven’s Dependency Lookup Order:**

When you add a dependency in pom.xml, Maven checks in this order:

1. **Local Repository** (~/.m2/repository)  
   → If the JAR already exists here, it uses it immediately.
2. **Repositories in settings.xml –** only If organization provides …
3. **Remote Repositories** (e.g. Maven Central, custom repos)  
   → If not found locally, Maven fetches from the remote repo (like Maven Central) and saves it in .m2.

C: -> Users-> suchi-> .m2 -> repository (this is local repo)

C: -> Users-> suchi-> .m2 -> settings.xml (this is overridden settings.xml of organization level repo)

Open source – Central repository

***Spring Boot project :***

**We usually don't need to add a logging dependency manually in a Spring Boot project — because Spring Boot includes it by default.**

**🔧 Here's Why:**

Spring Boot uses **Spring Boot Starter Logging**, which includes:

* **SLF4J** (Simple Logging Facade for Java)
* **Logback** (as the default logging implementation)
* This starter is included automatically through spring-boot-starter dependencies.

A screenshot of a computer

AI-generated content may be incorrect.

**when would you need to manually add a logging dependency?**

Only if:

1. You want to **use a different logging framework** (e.g., Log4j2 or java.util.logging)
2. You want to **disable Spring Boot’s default logging** and configure everything yourself
3. You're using **Maven exclusions** and removed logging by accident