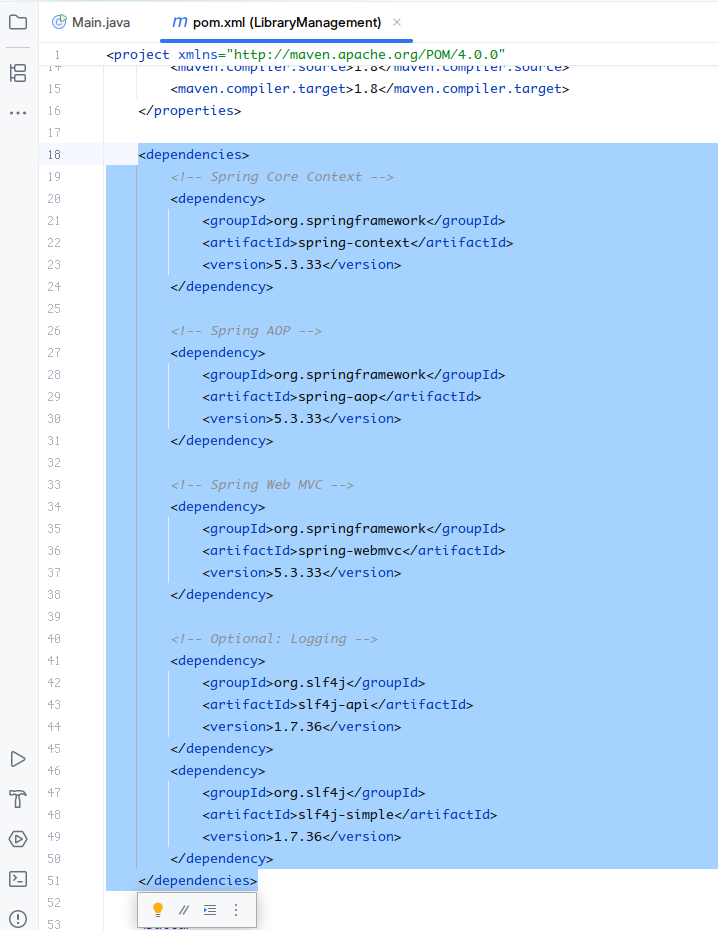
|  |
| --- |
| *Exercise 4: Creating and Configuring a Maven Project (****Solutions****)* |

1. **Adding Spring Dependencies in pom.xml:** Spring dependencies like spring-context, spring-aop, and spring-webmvc are added in the <dependencies> section of pom.xml. spring-context provides core features like dependency injection. spring-aop supports aspect-oriented programming for modularizing cross-cutting concerns. spring-webmvc enables building Spring-based web applications using the Model-View-Controller pattern.
2. **Configuring the Maven Compiler Plugin:** In the <build> section, the maven-compiler-plugin is used to tell Maven which Java version to use for compiling. By setting both source and target to 1.8, it ensures your code is written and compiled with Java 8 compatibility, which is required for running Spring 5.x reliably.





**spring-context** This dependency brings in Spring’s core features, especially for creating and managing beans using dependency injection (DI) and inversion of control (IoC). It allows you to define beans in XML or Java config and lets Spring handle the wiring.

**spring-aop** This enables Aspect-Oriented Programming in your project. It allows separation of cross-cutting concerns like logging, transaction management, or security, without cluttering your business logic. It works by intercepting method calls and injecting behavior before or after execution.

**spring-webmvc** This provides support for building web applications using the Spring MVC architecture. It includes features like controllers, view resolvers, request mappings, and form handling, making it ideal for creating Java-based web apps.

**slf4j-api** This is a simple logging abstraction layer that decouples your application from a specific logging framework. It acts as a common API that you can plug different logging systems like Logback or Log4j into.

**slf4j-simple** This is a lightweight, simple implementation of the SLF4J API. It lets you output logs to the console without configuring a complex logging system, making it ideal for development and small projects.

**maven-compiler-plugin** This plugin is part of the Maven build lifecycle. It compiles your Java source files and ensures they are compatible with a specific Java version. Setting <source> and <target> to 1.8 ensures that all code is compiled to run on Java 8, which is important for compatibility with Spring 5.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*