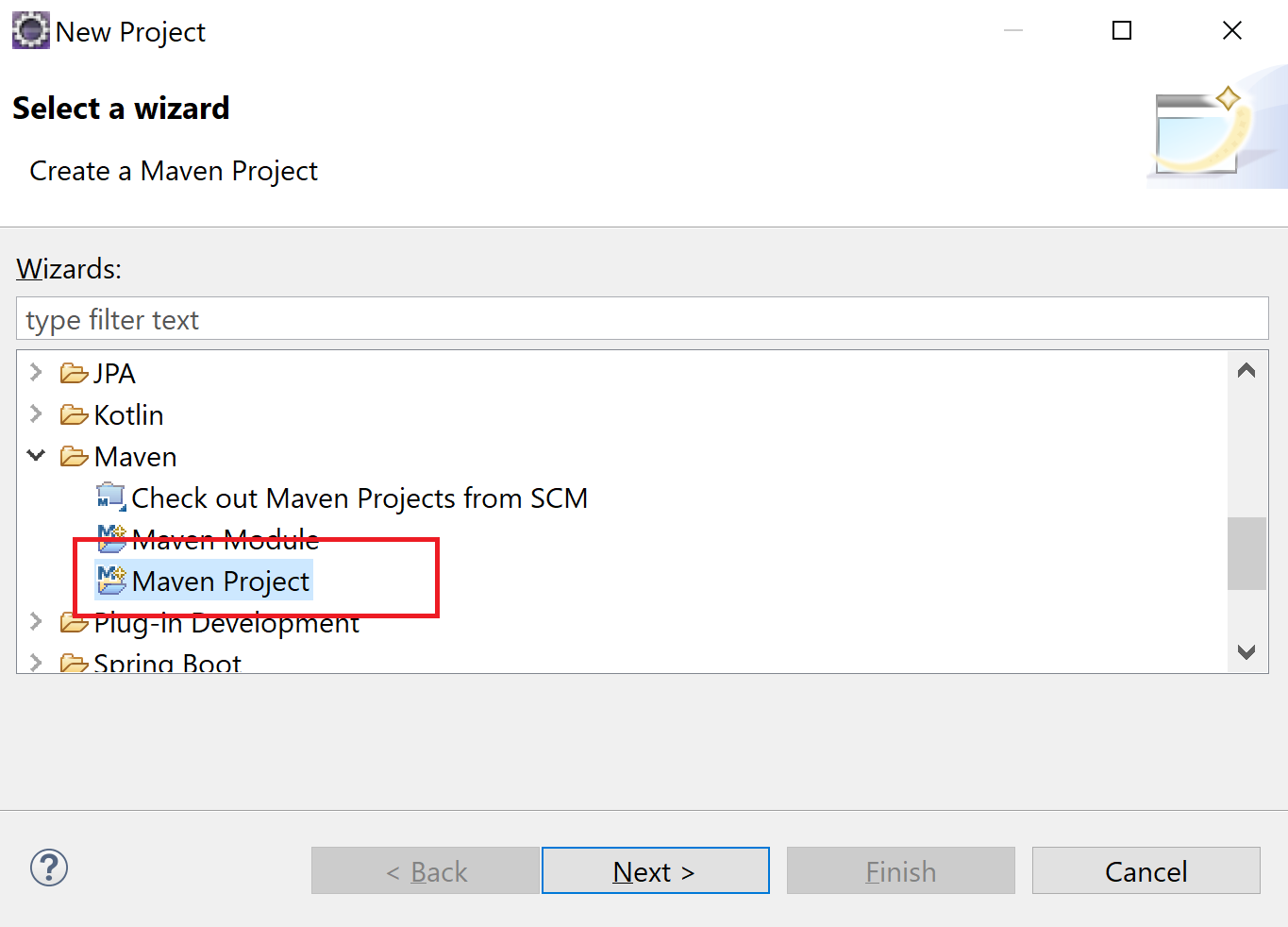
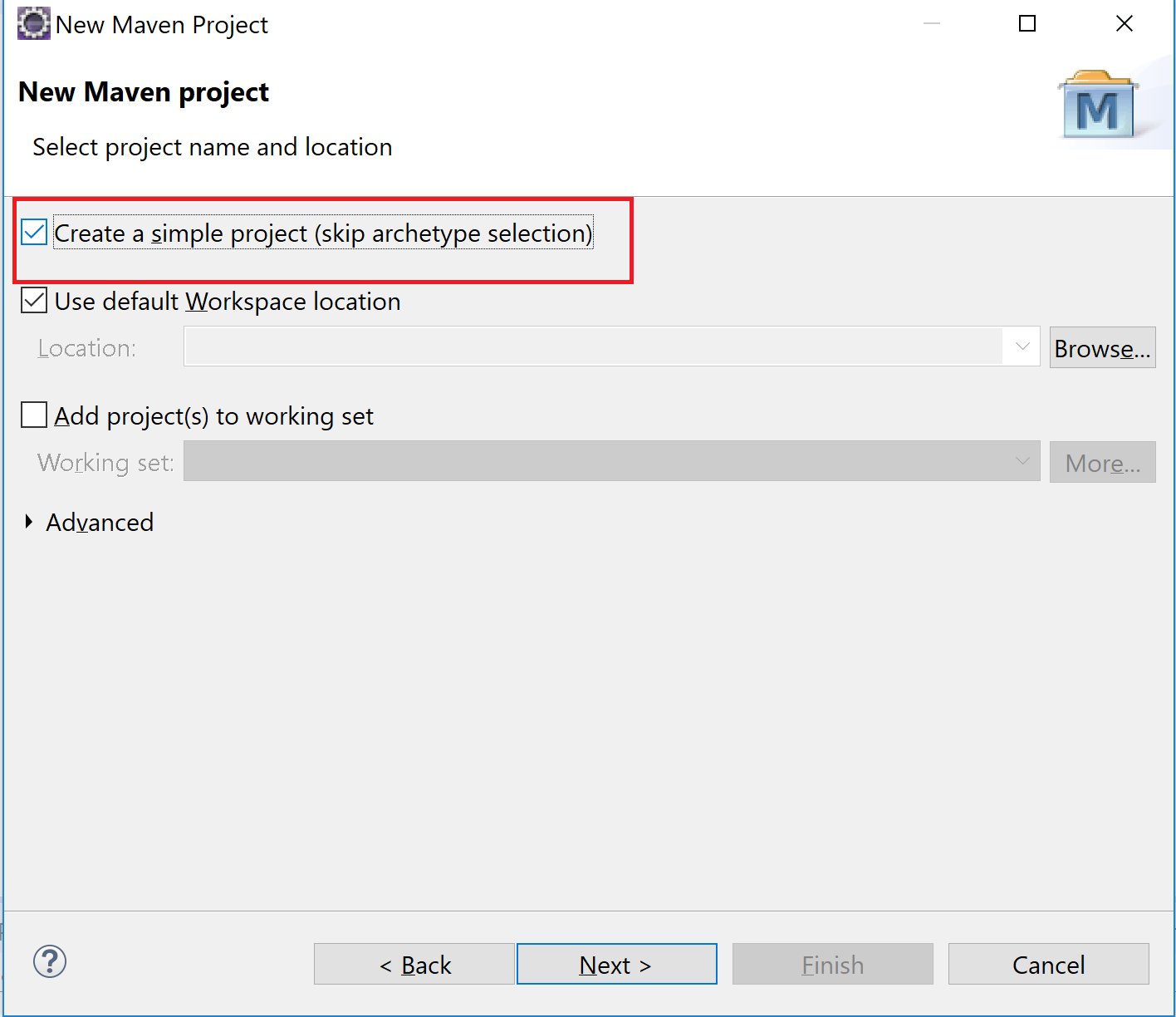
Spring core: Desktop application

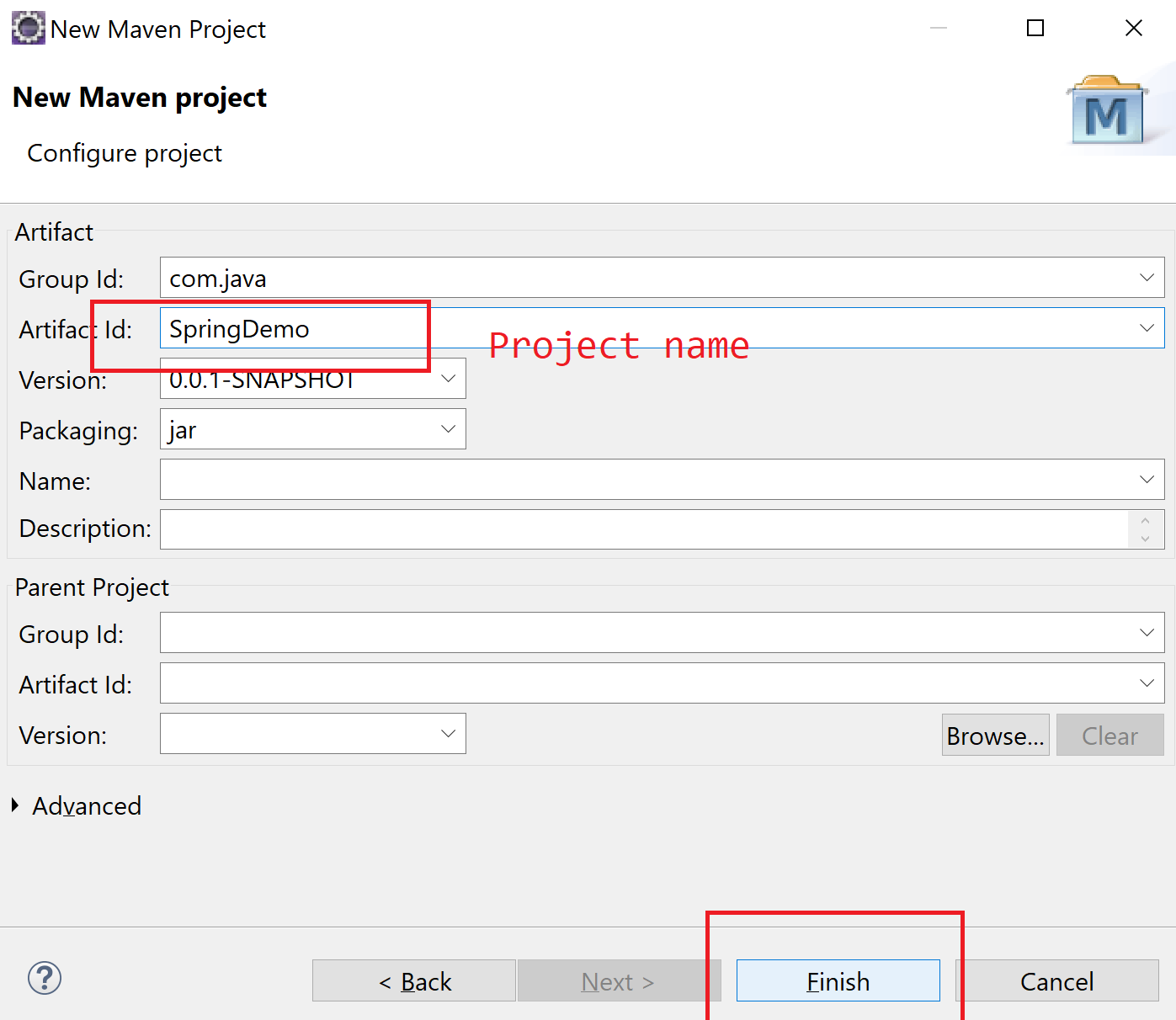
1. **Dependency injection/ Inversion of control**
   1. Constructor injection
   2. Setter injection
   3. Method injection

Loose coupling : 1 layer if u do any changes , it does not affect other layers

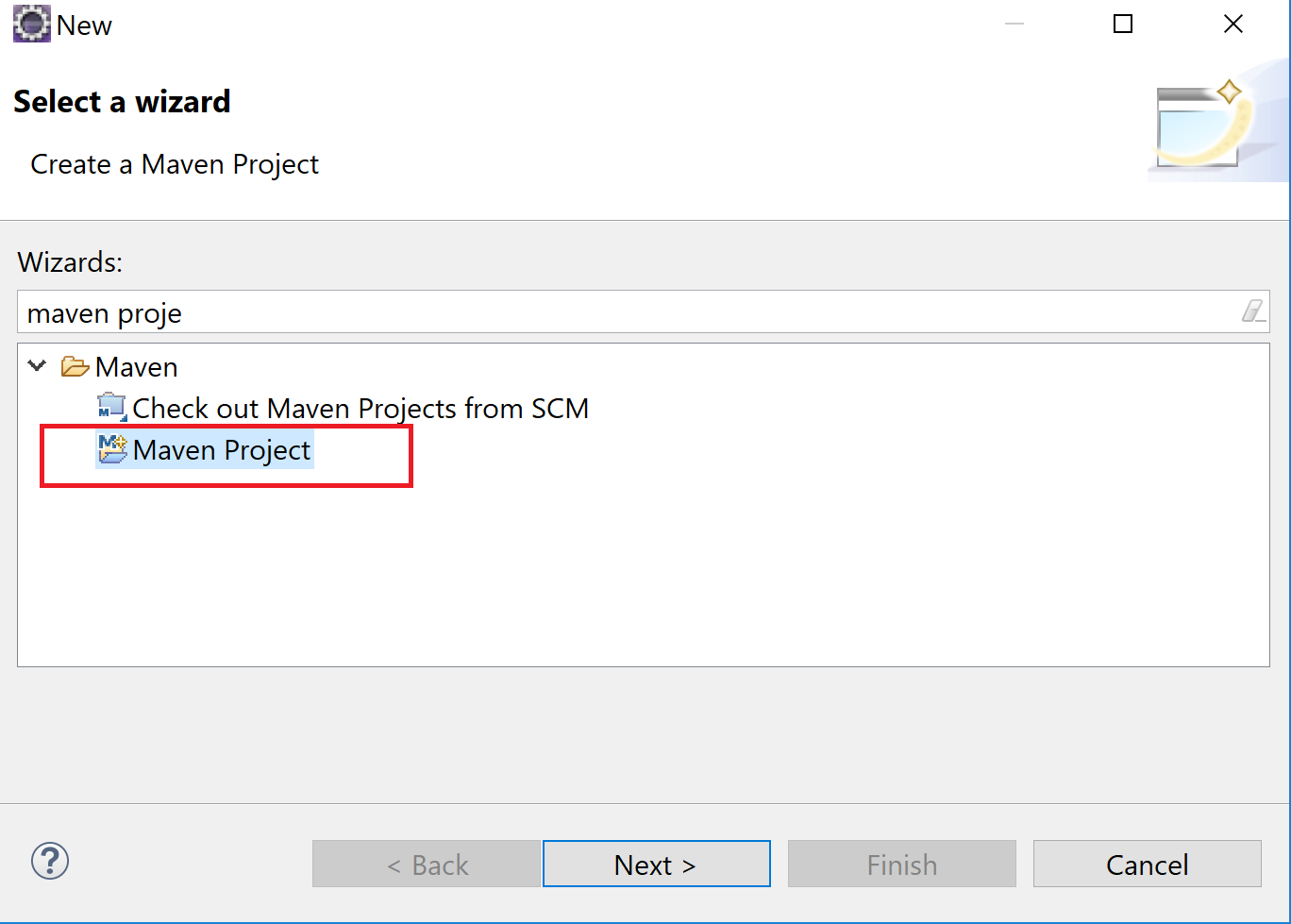
1. Interfaces for reference
2. Objects are created by spring and injected as a dependency. We can configure this
   1. Xml config
   2. Java config
   3. Annotations







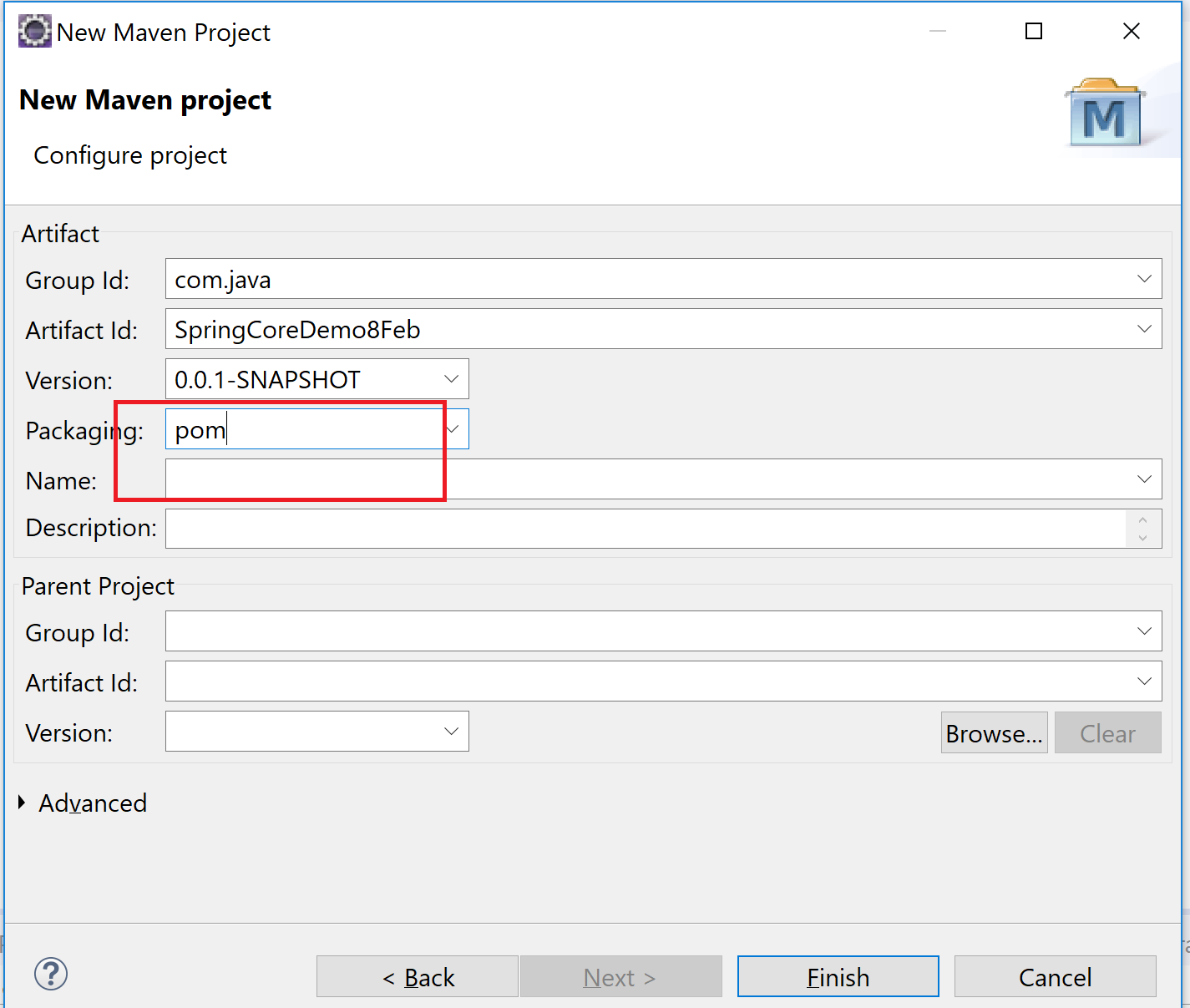
**Maven module**

****

1 Project can have multiple modules inside it.

In a project, we can define dependencies/plugins in pom.xml. (Parent pom) So all the modules will inherit the same dependencies/plugin. So we don’t need to configure it again in all the projects.

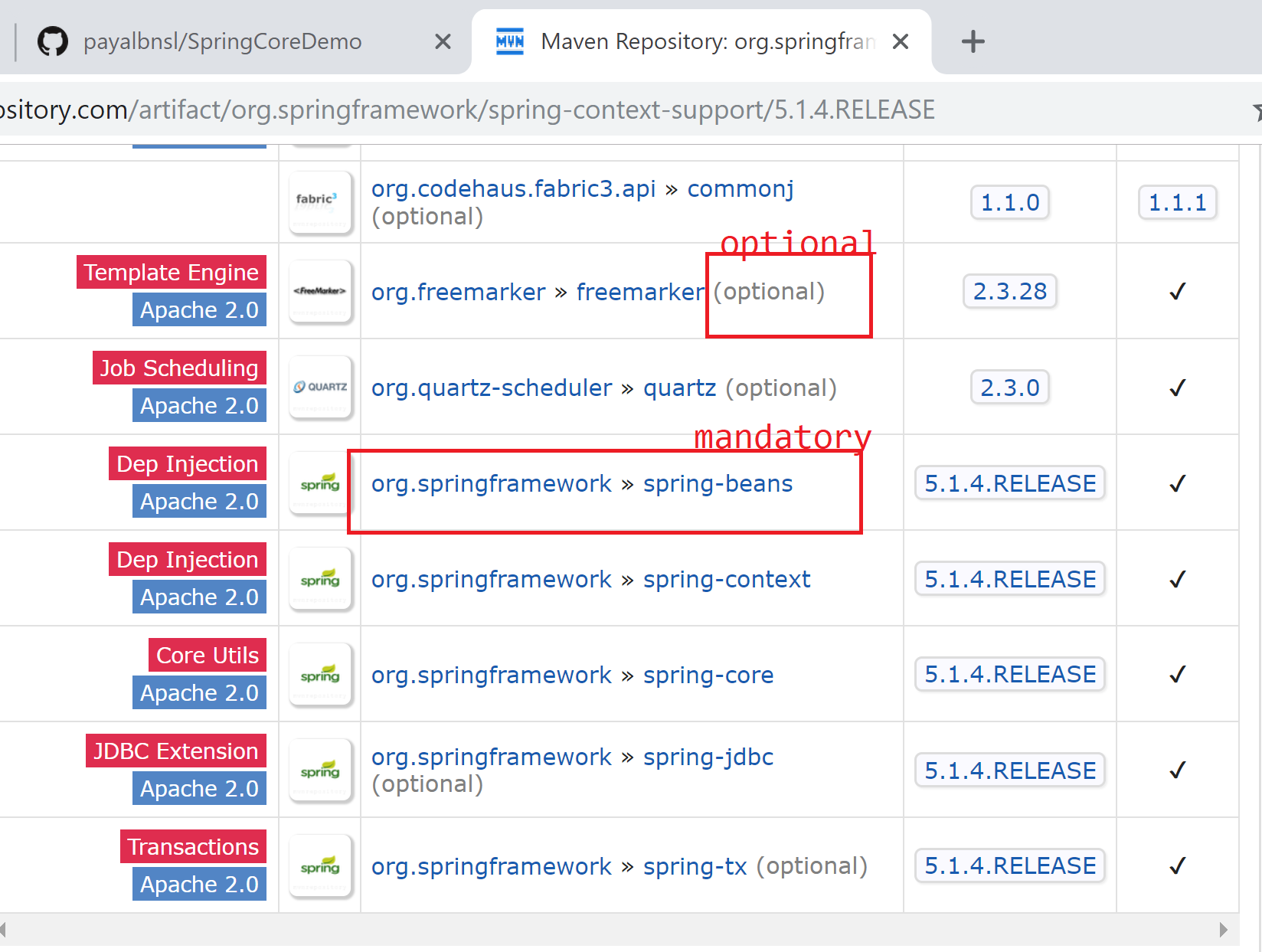
**Banking Application: It has various modules: Account manager, Loan, Fixed deposits. We may want some standard. Jdk1.8, spring 5.1. So I can create a parent pom and specify these dependencies/ plugins over there. So all modules may inherit the parent pom.**

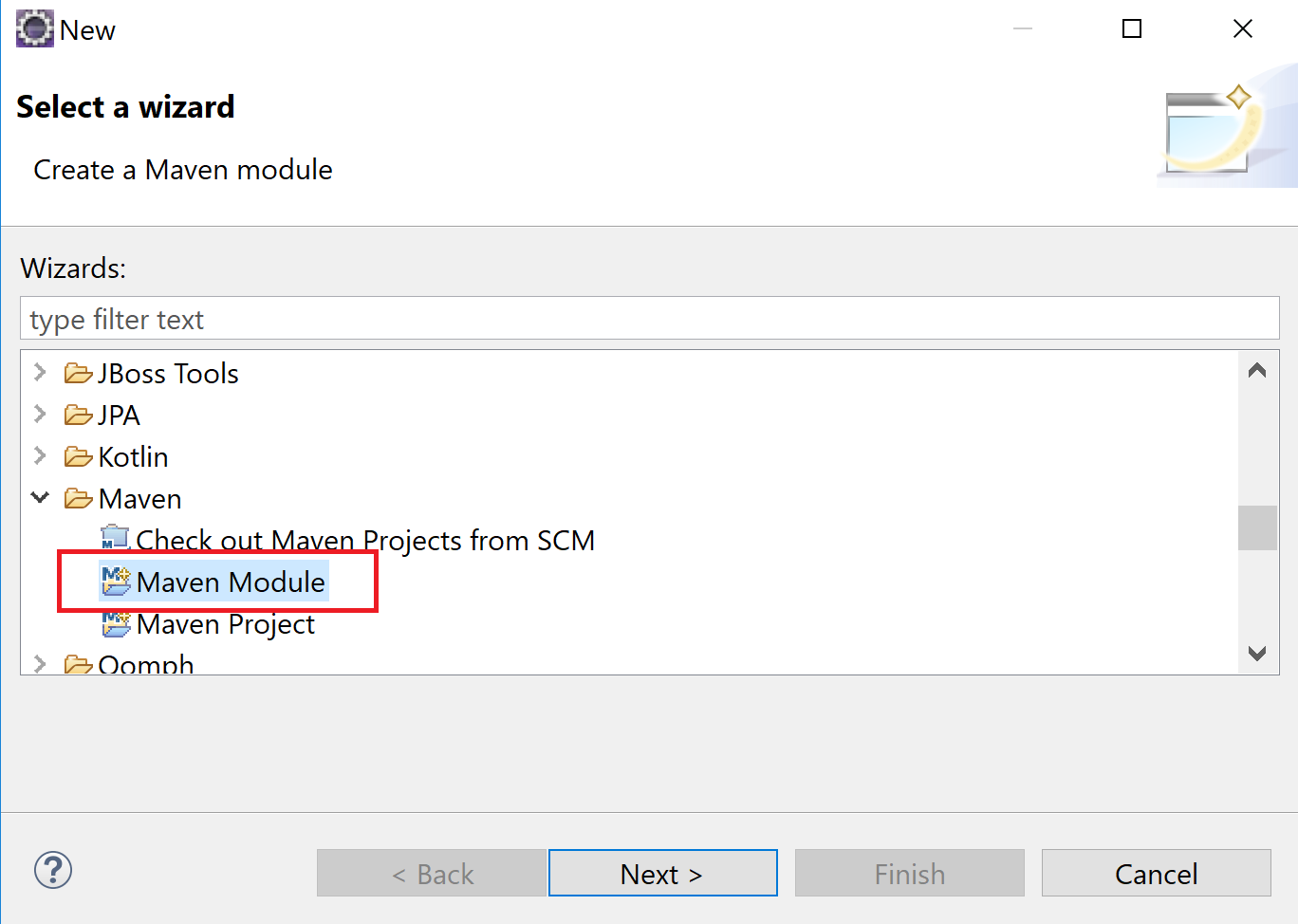
****

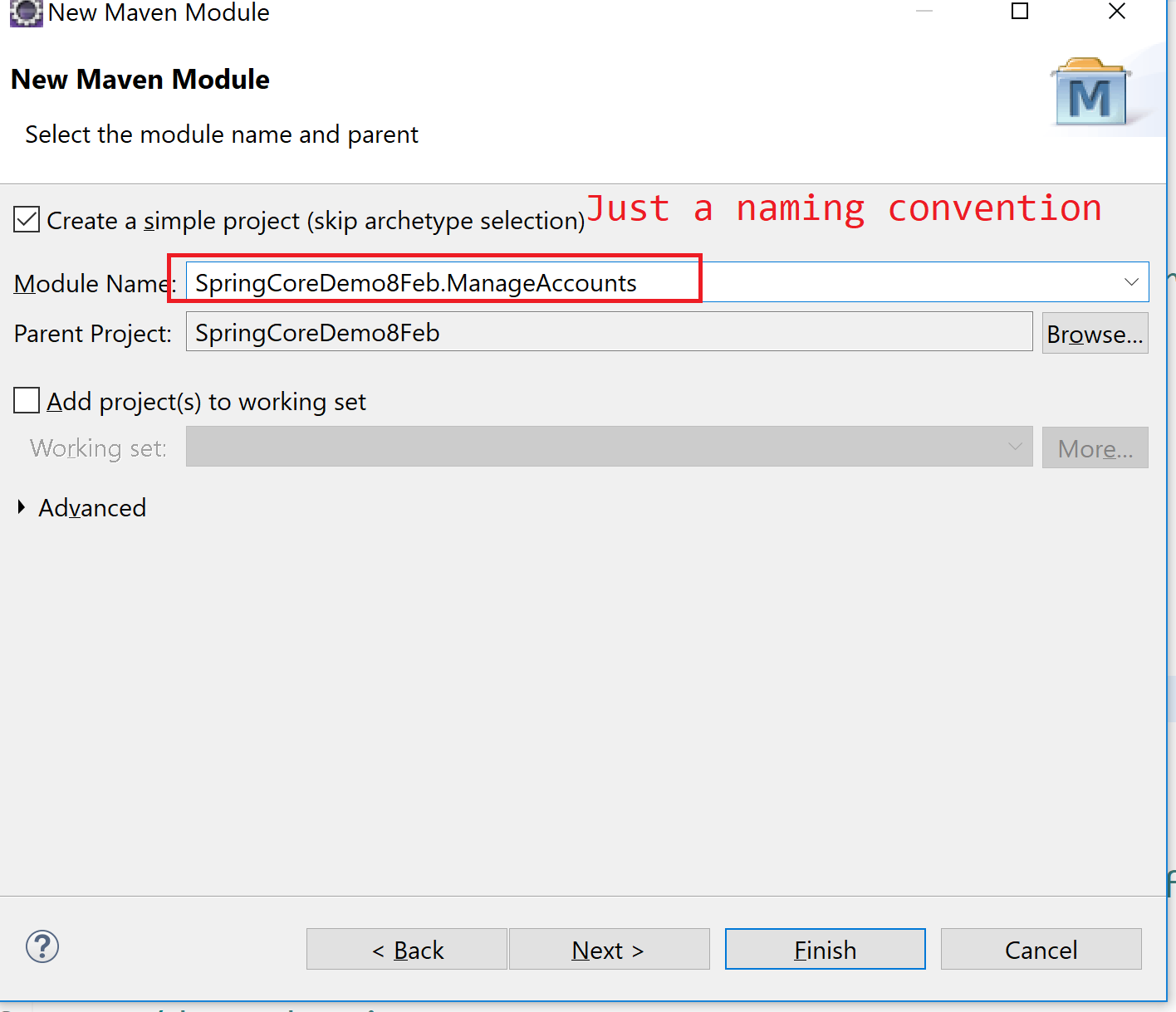
If we use maven as a build tool: compile/ package/ run test cases: It uses various plugins

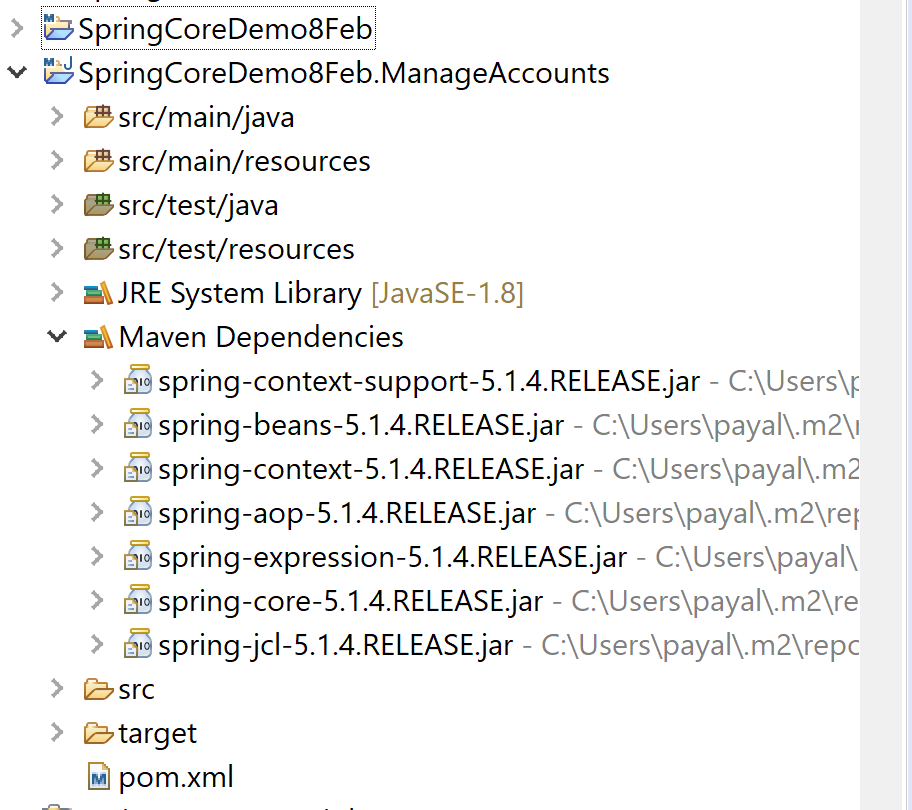
1. Maven-compiler-plugin: compile
2. Maven-surefire-plugin: test cases

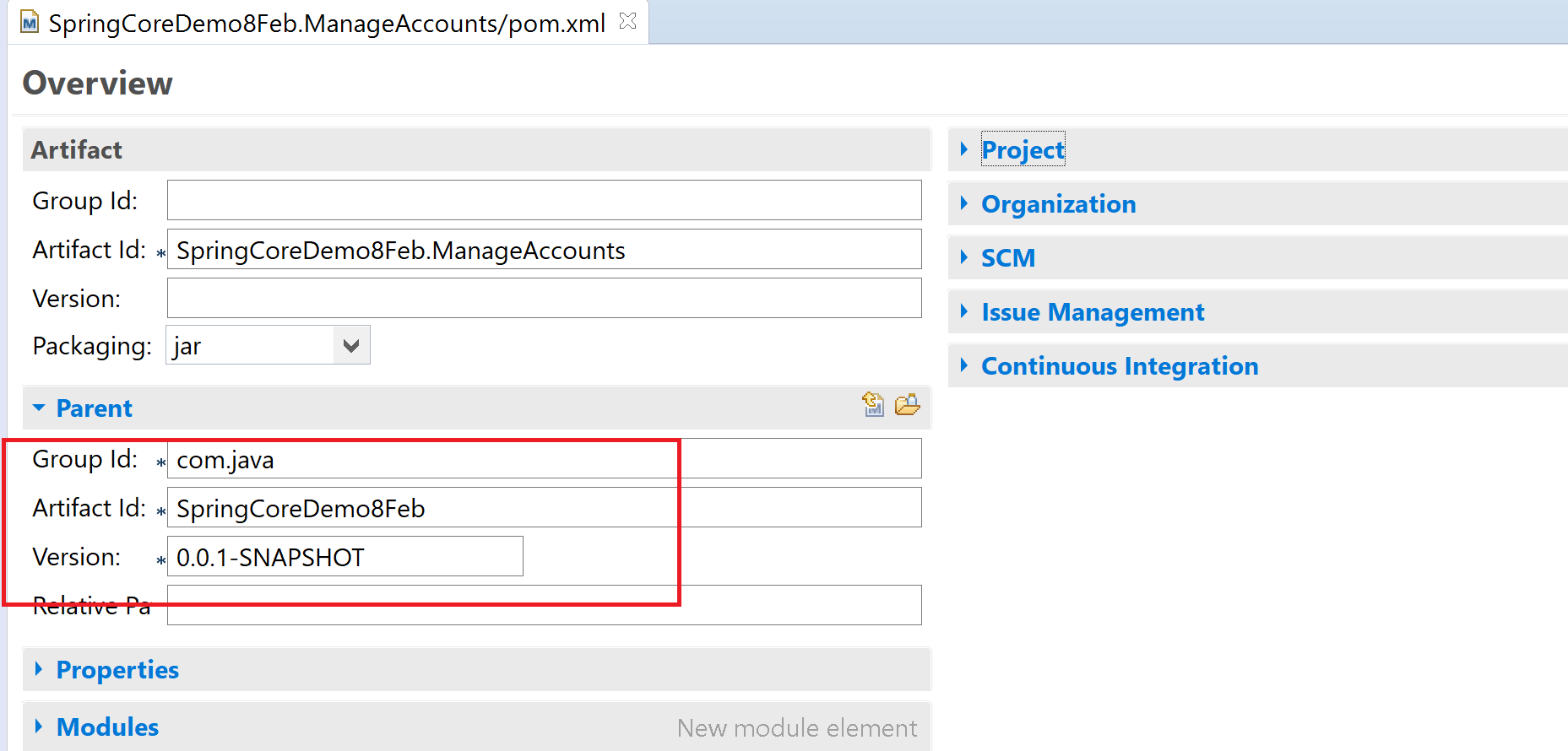
Compiler: jdk 1.8





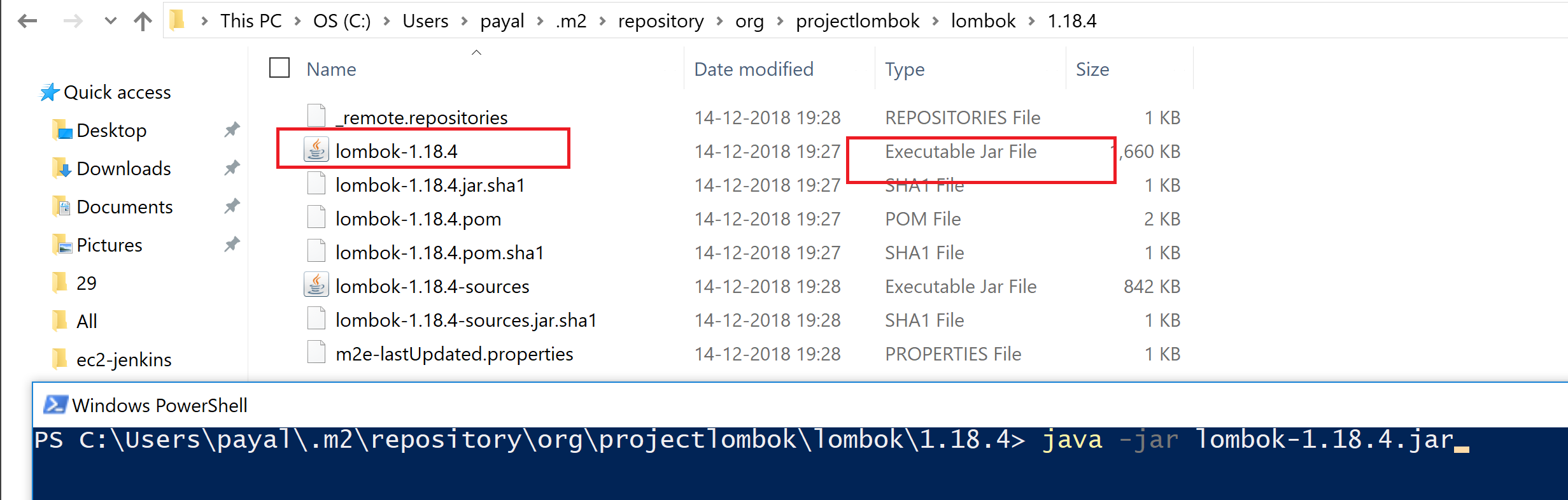




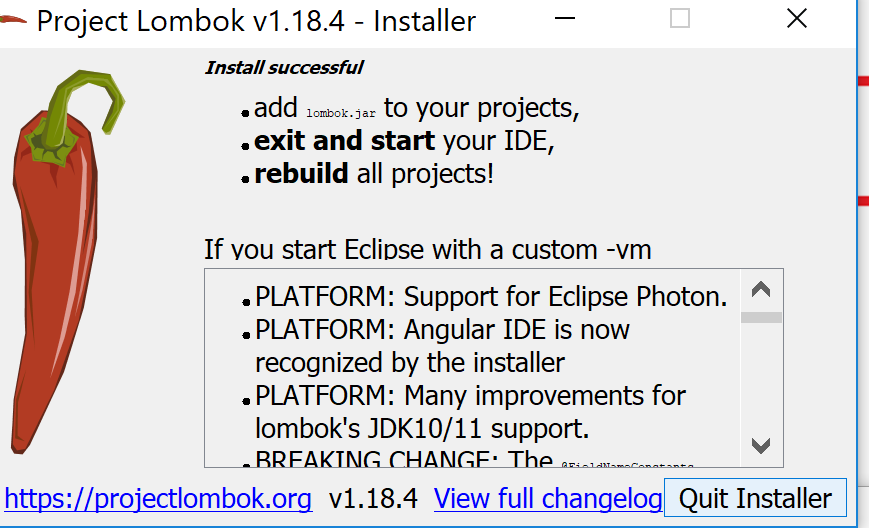


Maven: build tool, manage project (modules)

**Lombok configuration**



Give the path to your eclipse.exe. Next



Restart your eclipse.

Connection Pool: tomcat-dbcp library

<dependency>

<groupId>org.apache.tomcat</groupId>

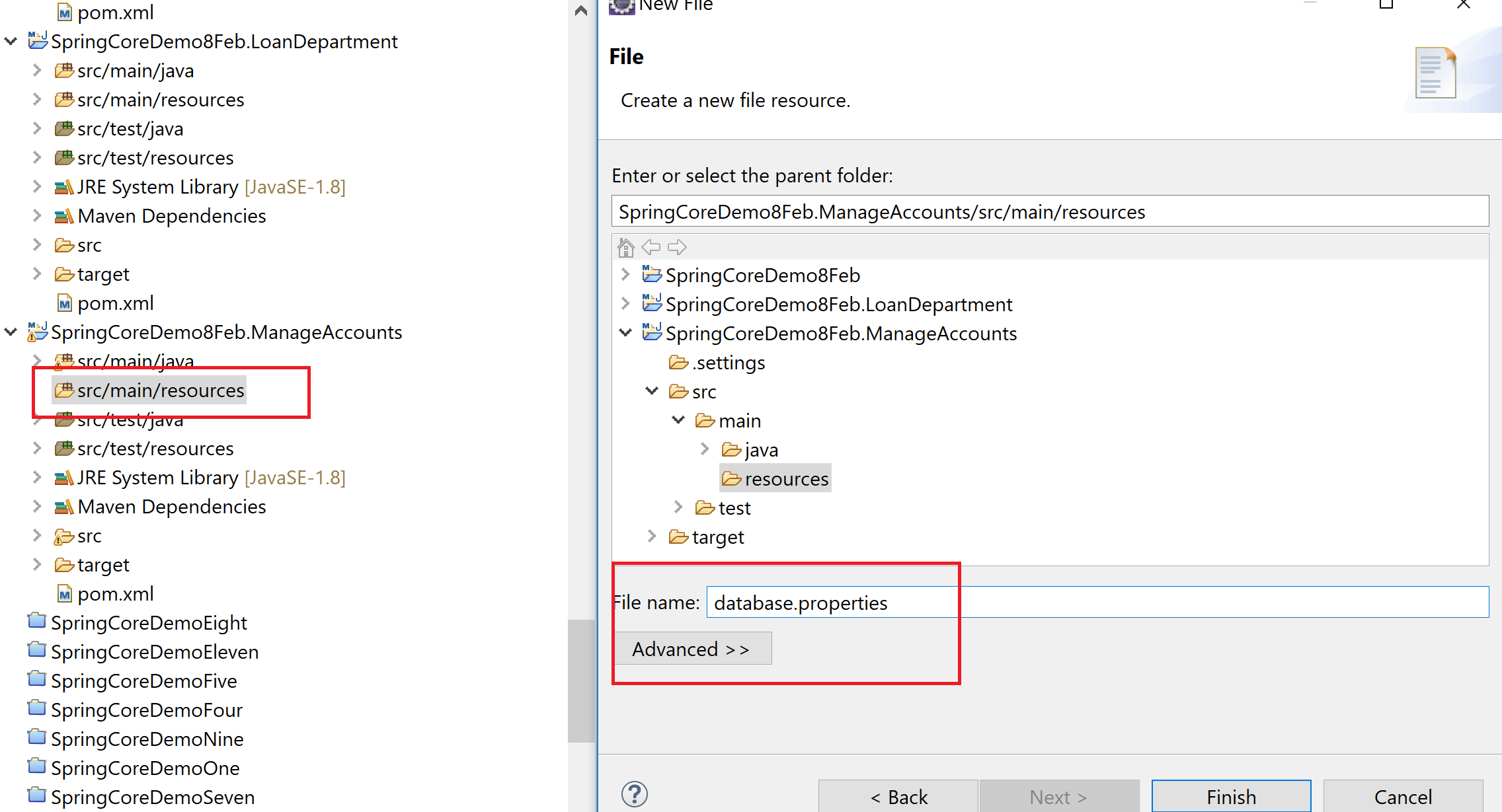
<artifactId>tomcat-dbcp</artifactId>

<version>9.0.10</version>

</dependency>

It provides you the class BasicDataSource which is like a connection pool.

**Externalize properties**



Property file: key value pair

Service: non-business code. Logging, managing the transactions, security

Client > Service -> Repository

Spring MVC: web applications. Built on top of spring core