Programming Advanced Java

WEEK 2 - MAVEN

Goals

The **junior-colleague**

- can explain what Maven is.
- can identify the Maven-related files: settings.xml and pom.xml.
- can describe the 3 build lifecycles of Maven.
- can explain that each build lifecycle is made up of phases.
- can explain that each build phase is made up of plugin goals.
- can identify the project coordinates.
- can describe the Maven Standard Directory Layout.
- can explain what a dependency is.
- can describe the different dependency scopes.
- can explain what a transitive dependency is.

- can create a Maven project.
- can execute maven phases and goals from CLI.
- can manage dependencies with Maven.
- can configure and use a logging framework.

Overview

1. What is Maven?
2. Installation and Configuration
3. Simple Maven Project
4. Maven Standard Directory Layout
5. Project Coordinates
6. Maven Build Lifecycles
7. A Build Lifecycle is Made Up Of Phases
8. Plugins and Goals
9. Dependencies
10. Dependency Scopes
11. Logging Framework
12. Exercise
13. Transitive dependencies
14. Executable package

1. What is Maven?

- Project management tool
- More than a build tool
  - Generating sources
  - Compiling sources and test sources
  - Packaging
  - Health checks
  - Reporting

https://maven.apache.org/what-is-maven.html

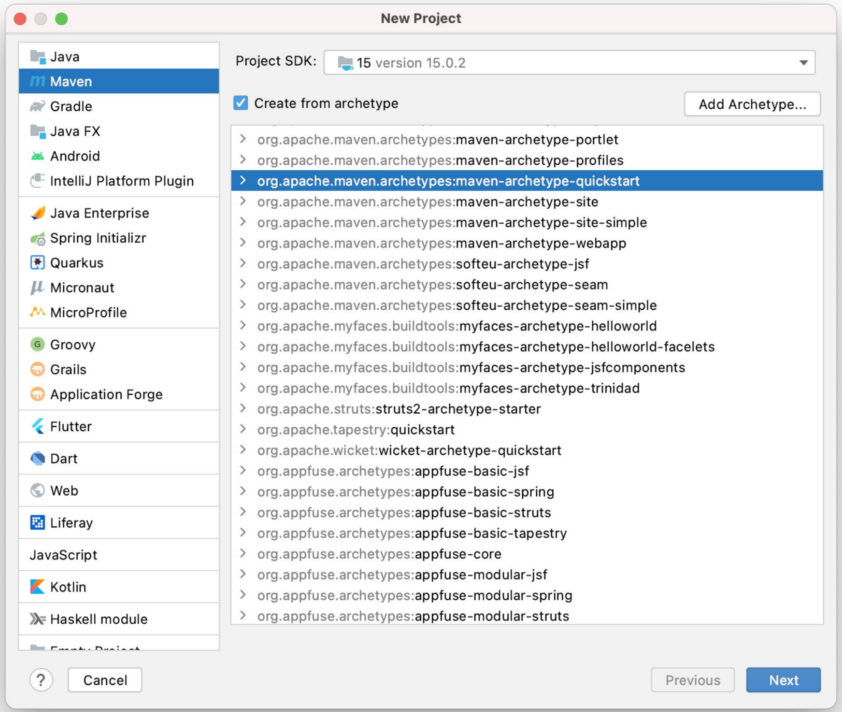
2. Installation and configuration

Download: https://maven.apache.org/download.cgi

Configuration: https://maven.apache.org/guides/getting-started/windows-prerequisites.html

```
$ mvn -version
Apache Maven 3.6.1 (d66c9c0b3152b2e69ee9bac180bb8fcc8e6af555; 2019-04-04T21:00:29+02:00)
Maven home: /usr/local/Cellar/maven/3.6.1/libexec
Java version: 14.0.2, vendor: Oracle Corporation, runtime:
/Library/Java/JavaVirtualMachines/jdk-14.0.2.jdk/Contents/Home
Default locale: nl_BE, platform encoding: UTF-8
OS name: "mac os x", version: "10.16", arch: "x86_64", family: "mac"
```

3. A Maven Project



New Project

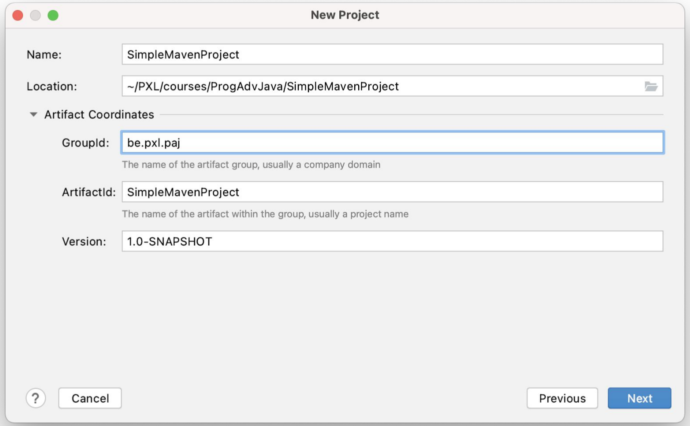
- Java
- Maven
- Gradle
- Java FX
- Android
- IntelliJ Platform Plugin
- Java Enterprise
- Spring Initializr
- Quarkus
- Micronaut
- MicroProfile
- Groovy
- Grails
- Application Forge
- Flutter
- Dart
- Web
- Liferay
- JavaScript
- Kotlin
- Haskell module

Project SDK: 15 version 15.0.2

☑ Create from archetype — Add Archetype...

```
org.apache.maven.archetypes:maven-archetype-portlet
org.apache.maven.archetypes:maven-archetype-profiles
org.apache.maven.archetypes:maven-archetype-quickstart
org.apache.maven.archetypes:maven-archetype-site
org.apache.maven.archetypes:maven-archetype-site-simple
org.apache.maven.archetypes:maven-archetype-webapp
org.apache.maven.archetypes:softeu-archetype-jsf
org.apache.maven.archetypes:softeu-archetype-seam
org.apache.maven.archetypes:softeu-archetype-seam-simple
org.apache.myfaces.buildtools:myfaces-archetype-helloworld
org.apache.myfaces.buildtools:myfaces-archetype-helloworld-facelets
org.apache.myfaces.buildtools:myfaces-archetype-jsfcomponents
org.apache.myfaces.buildtools:myfaces-archetype-trinidad
org.apache.struts:struts2-archetype-starter
org.apache.tapestry:quickstart
org.apache.wicket:wicket-archetype-quickstart
org.appfuse.archetypes:appfuse-basic-jsf
org.appfuse.archetypes:appfuse-basic-spring
org.appfuse.archetypes:appfuse-basic-struts
org.appfuse.archetypes:appfuse-basic-tapestry
org.appfuse.archetypes:appfuse-core
org.appfuse.archetypes:appfuse-modular-jsf
org.appfuse.archetypes:appfuse-modular-spring
org.appfuse.archetypes:appfuse-modular-struts
```

? Cancel — Previous Next



New Project

Name:	SimpleMavenProject
Location:	~/PXL/courses/ProgAdvJava/SimpleMavenProject

Artifact Coordinates

GroupId:	be.pxl.paj
	The name of the artifact group, usually a company domain
ArtifactId:	SimpleMavenProject
	The name of the artifact within the group, usually a project name
Version:	1.0-SNAPSHOT

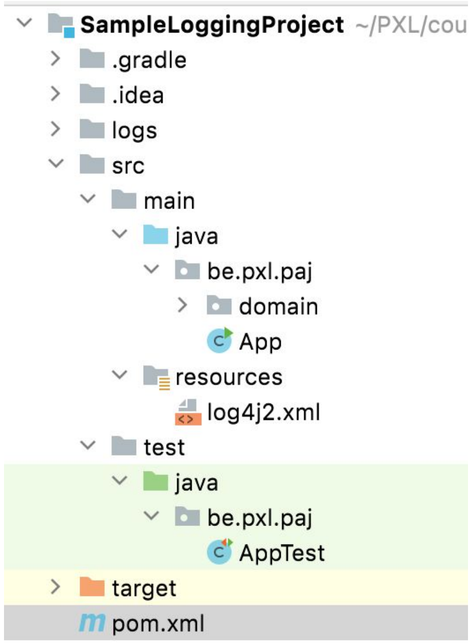
? Cancel — Previous Next

Or command-line:

```
$ mvn archetype:generate
```

HOGESCHOOL PXL

4. Maven Standard Directory Layout



```
SampleLoggingProject ~/PXL/cou
  .gradle
  .idea
  logs
  src
    main
      java
        be.pxl.paj
          domain
          App
      resources
        log4j2.xml
    test
      java
        be.pxl.paj
          AppTest
  target
  pom.xml
```

```
$ mvn package
$ cd target
$ java -cp ./SampleLoggingProject-1.0-SNAPSHOT.jar be.pxl.paj.App
```

5. Project Coordinates

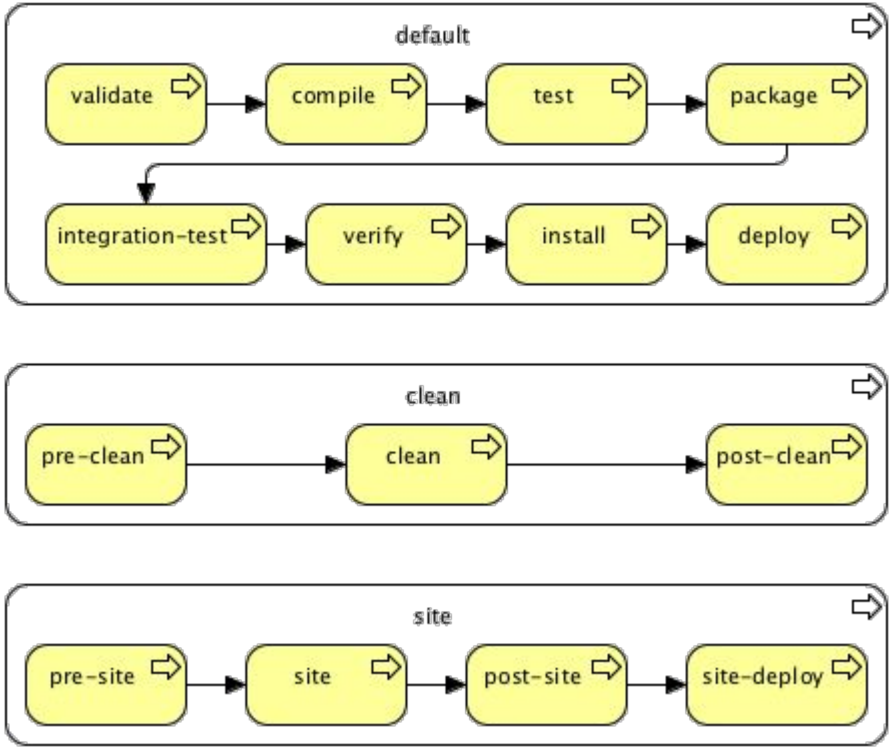
```
<?xml version="1.0" encoding="UTF-8"?>
<project ...>
 <modelVersion>4.0.0</modelVersion>

 <groupId>be.pxl.paj</groupId>
 <artifactId>SampleLoggingProject</artifactId>
 <version>1.0-SNAPSHOT</version>

 <name>SampleLoggingProject</name>

 <properties>
   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
   <maven.compiler.source>17</maven.compiler.source>
   <maven.compiler.target>17</maven.compiler.target>
 </properties>

 </project>
```

6. Maven built-in life cycles

3 build lifecycles: default, clean and site

clean	handles the cleanup of directories and files generated during the build process	mvn clean
default	handles the build and distribution of the project	mvn [plugin:goal]* [phase]*
site	create project documentation	mvn site

7. A Build Lifecycle Is Made Up Of Phases

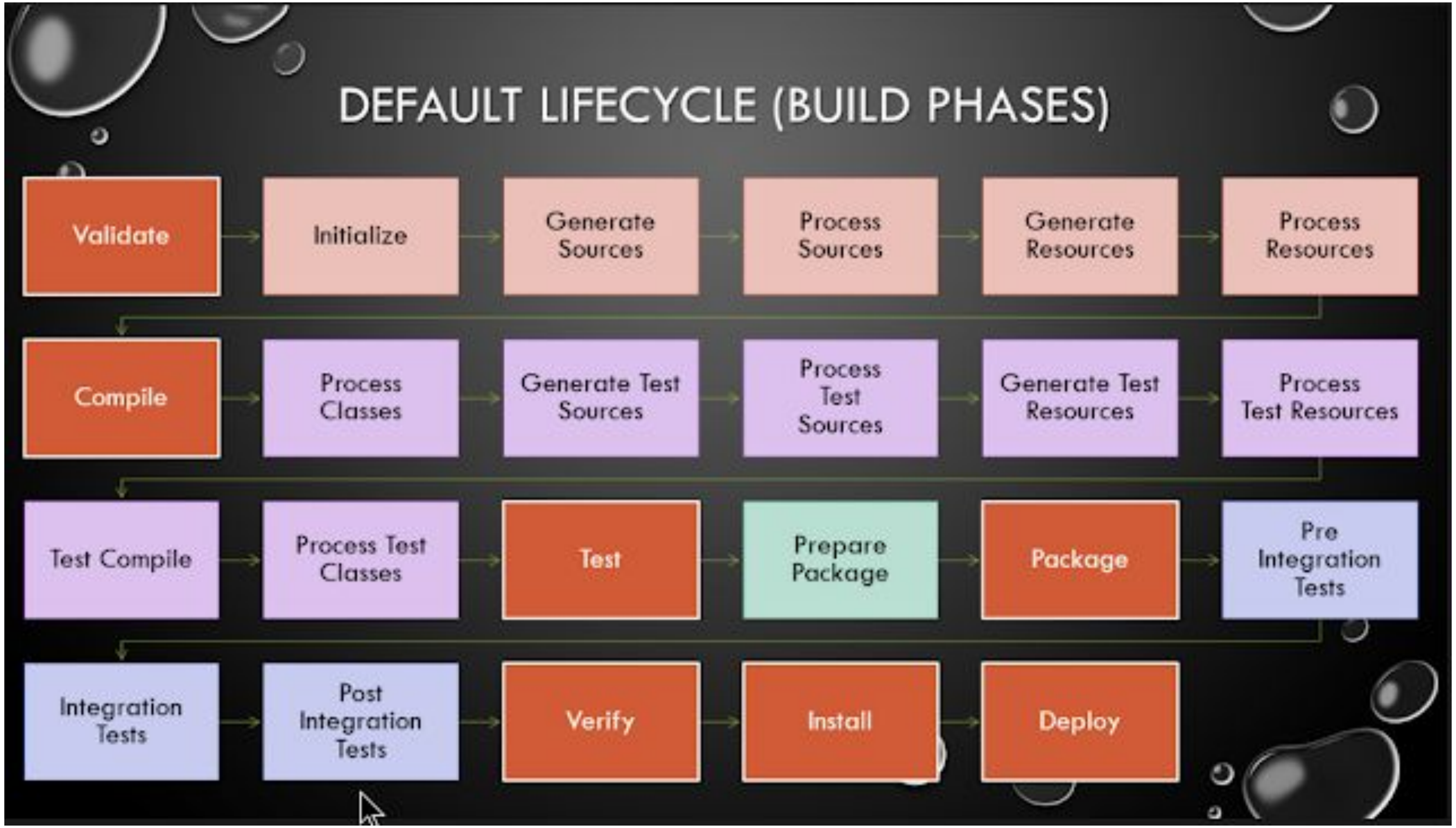


Each build lifecycle is defined by a sequence of phases.

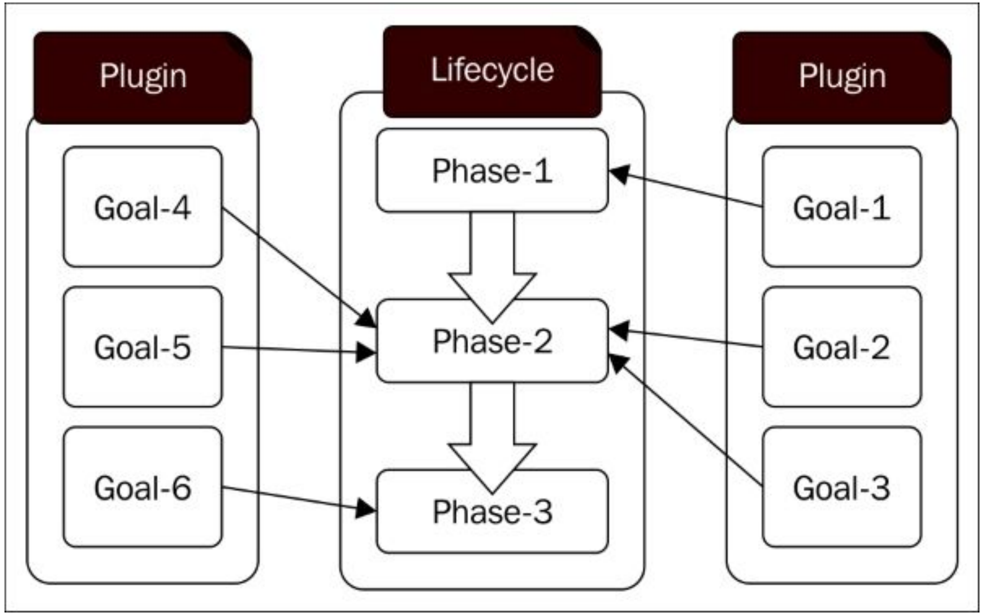
Whenever a maven command for any life cycle is invoked, maven executes the phases till and up to the invoked phase.

```
$mvn clean install
```

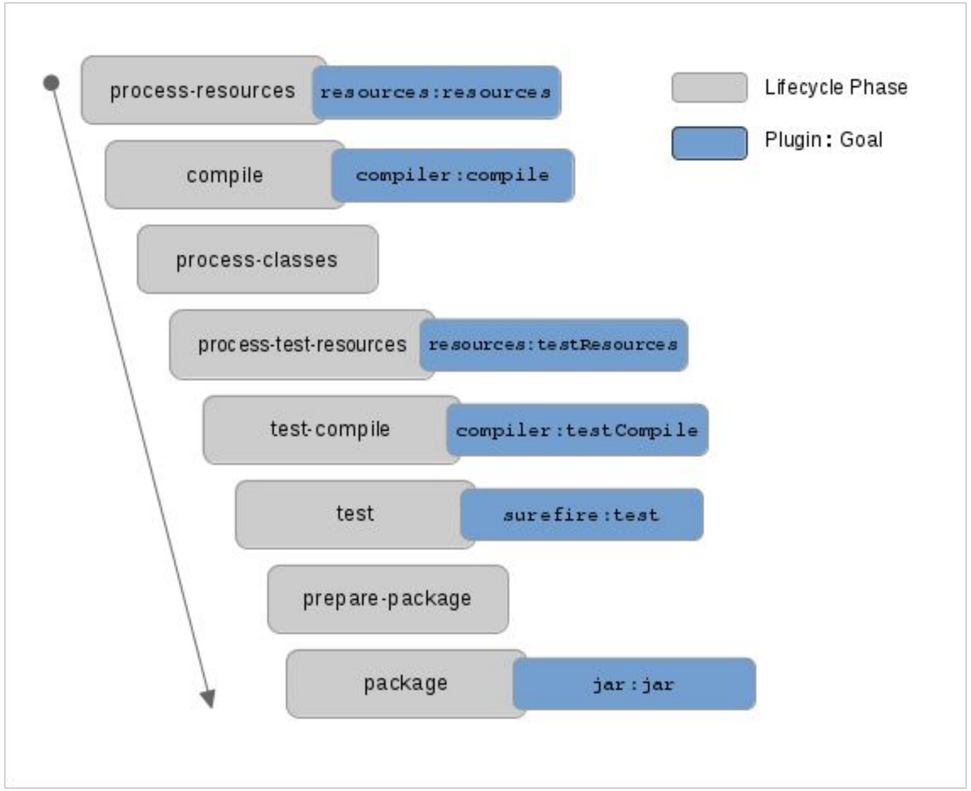
HOGESCHOOL PXL


DEFAULT LIFECYCLE (BUILD PHASES)
Validate
Initialize
Generate Sources
Process Sources
Generate Resources
Process Resources
Compile
Process Classes
Generate Test Sources
Process Test Sources
Generate Test Resources
Process Test Resources
Test Compile
Process Test Classes
Test
Prepare Package
Package
Pre Integration Tests
Integration Tests
Post Integration Tests
Verify
Install
Deploy

8. Plugins and goals

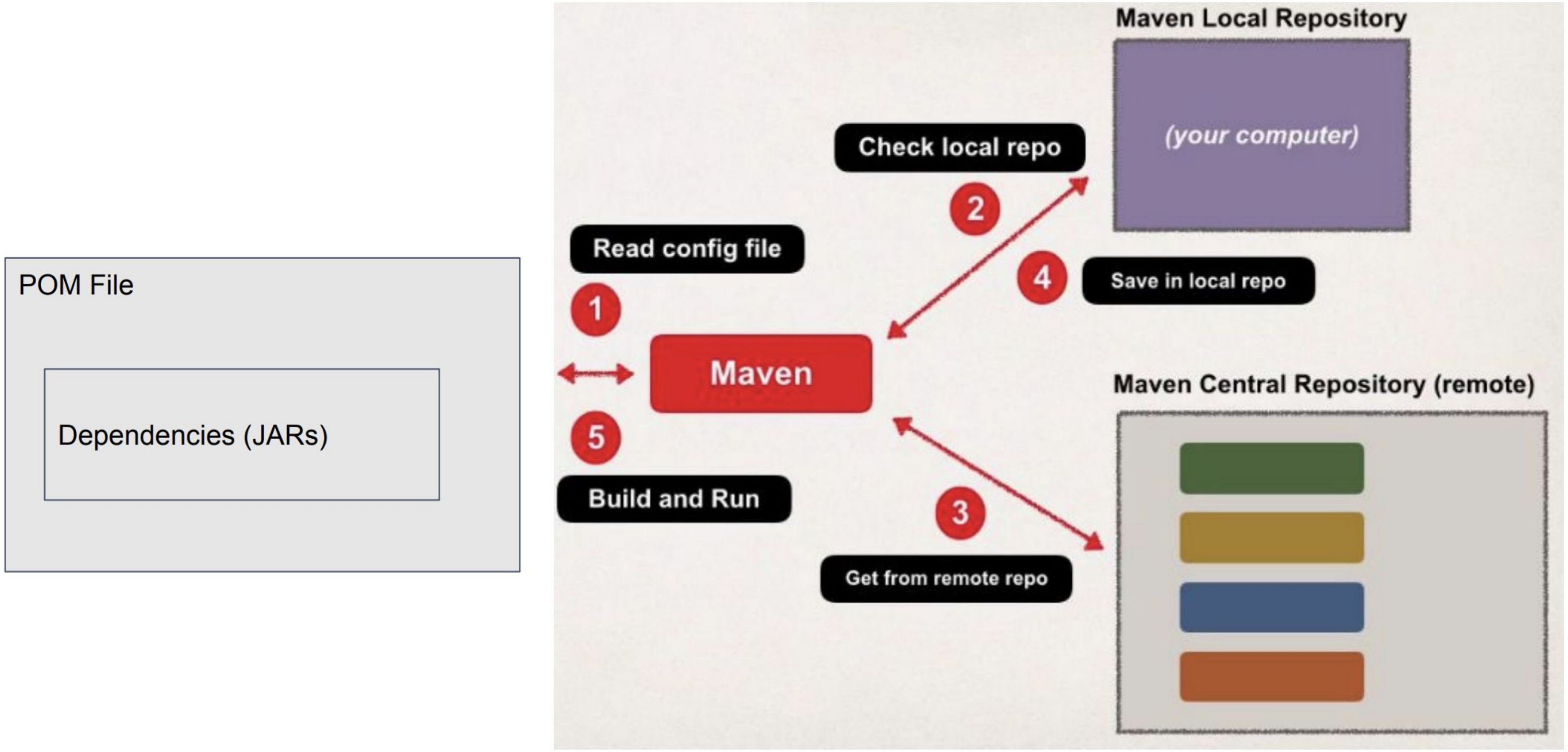


8. Plugins and goals



9. Dependencies

```
Windows: C:\Users\<User_Name>\.m2
Linux: /home/<User_Name>/.m2
Mac: /Users/<user_name>/.m2
```



9. Dependencies

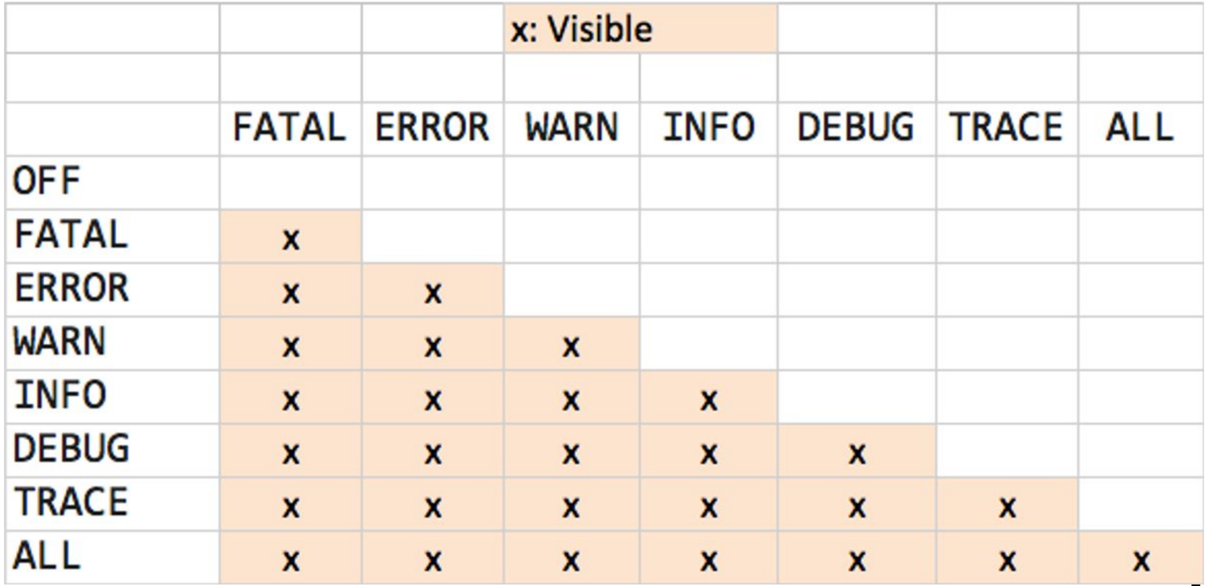
```xml
<?xml version="1.0" encoding="UTF-8"?>

<project …>

  <dependencies>
   <dependency>
     <groupId>junit</groupId>
     <artifactId>junit</artifactId>
     <version>4.13.2</version>
     <scope>test</scope>
   </dependency>
 </dependencies>

 </project>
```

10. Dependency scopes

compile	This is the default scope. Dependencies with this scope are available on the classpath for all the build tasks.
provided	Dependencies that are provided at runtime by JDK or a container. The provided dependencies are available at compile-time and in the test classpath.
runtime	The dependencies with this scope are only required at runtime. They are not needed at compile-time and in the test classpath.
test	These dependencies are only needed for executing tests.
system	Similar to provided but specific jar is provided.
import	All dependencies listed in another pom are included.

11. Logging framework

https://logging.apache.org/log4j/2.x/



			x: Visible				
	FATAL	ERROR	WARN	INFO	DEBUG	TRACE	ALL
OFF							
FATAL	x						
ERROR	x	x					
WARN	x	x	x				
INFO	x	x	x	x			
DEBUG	x	x	x	x	x		
TRACE	x	x	x	x	x	x	
ALL	x	x	x	x	x	x	x

12. Exercise

Make log4j 2 available in the project SampleLoggingProject.

Define a static logger variable in the class App.

Configure log4j 2 to write to a file log/superhero.log (see textbook or https://mkyong.com/logging/log4j2-xml-example/)

Use the logger variable to create some logging with different log levels.

Test different log level configurations.

13. Transitive dependencies

```
$ mvn dependency:tree
```

```
[INFO] --- maven-dependency-plugin:2.8:tree (default-cli) @
SimpleMavenProject ---
[INFO] be.pxl.paj:SimpleMavenProject:jar:1.0-SNAPSHOT
[INFO] +- junit:junit:jar:4.11:test
[INFO] |  \- org.hamcrest:hamcrest-core:jar:1.3:test
[INFO] +- org.apache.logging.log4j:log4j-api:jar:2.14.0:compile
[INFO] \- org.apache.logging.log4j:log4j-core:jar:2.14.0:compile
```

14. Executable package

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-assembly-plugin</artifactId>
    <executions>
        <execution>
            <phase>package</phase>
            <goals>
                <goal>single</goal>
            </goals>
            <configuration>
                <archive>
                <manifest>
                    <mainClass>
                        com.baeldung.executable.ExecutableMavenJar
                    </mainClass>
                </manifest>
                </archive>
                <descriptorRefs>
                    <descriptorRef>jar-with-dependencies</descriptorRef>
                </descriptorRefs>
            </configuration>
        </execution>
    </executions>
</plugin>
```