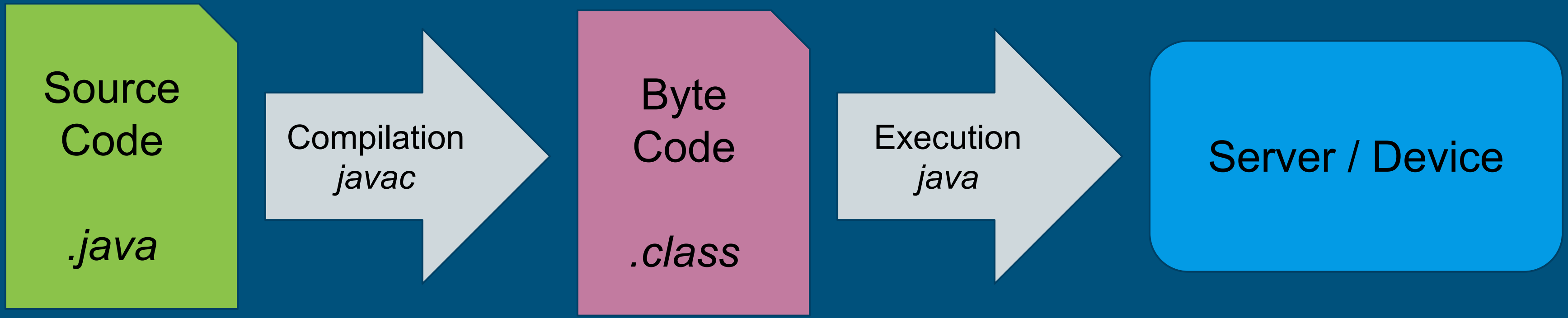
Lesson 01.B Software Project Management

Maven™

DEVOPS - ITI 4
HEI 2021-2022

How JAVA works ?

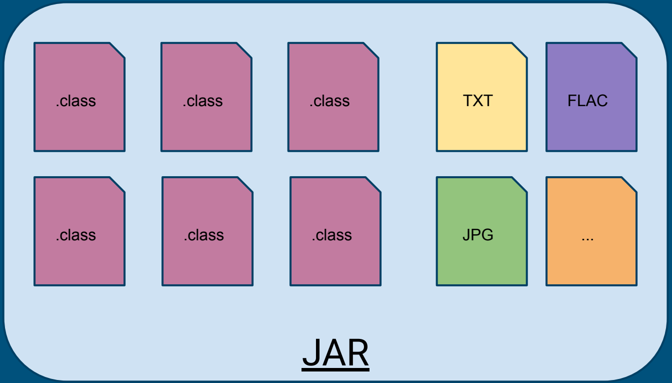
From Code to Program



Packaging

- JAR : Package file format with
  - Java class files
  - Associated metadata
  - Associated resources

- Related formats
  - WAR : Web Application archive
  - APK : Android Application Package



JAVA Programmers write source code to create JAR, WAR, APK …

What is Software project management ?

Build Management

- Source code must be compiled in byte code to be executed by JVM
- Byte code must be packaged in library with resources to be used
- Compilation and packaging must always be done in the same way

Dependency Management

- A program contains librairies (JAR) with its version
- Each library need its own libraries in correct version
- Some libraries may need same library with different version

Software Project Management

Software Project Management is a tool to help developper to automate and ensure quality during the build of a project

What is Maven ?

Build production?
Automated with Maven
Code quality?
Automated with SonarQube
And integration testing?
Automated with Cucumber
Cool...
Test environment deployment?
Automated with Puppet and Rundeck
So what are they all doing here?
Well, they're focusing on the last thing we didn't manage to automate yet...
...shipping good code!
CommitStrip.com

Maven’s Overview

- “Accumulator of knowledge” (Yiddish word)
- Hosted by the Apache Software Foundation

Project's life

- Maven project have unique identifier composed by :
    - Group : Unique project identity for all mavens projects
    - Artifact : Name of the project (unique for the group)

- Maven project have version number :
    - Developing versions are called *snapshots* (0.0.1-SNAPSHOT)
    - For each version,
        - the final and stable version is called *release* (0.0.1)
        - release version is unique

Project Object Model (POM)

- Contains :
  - Information about the project
  - Configuration details

- Xml file (pom.xml) in the project home

Dependencies recovery

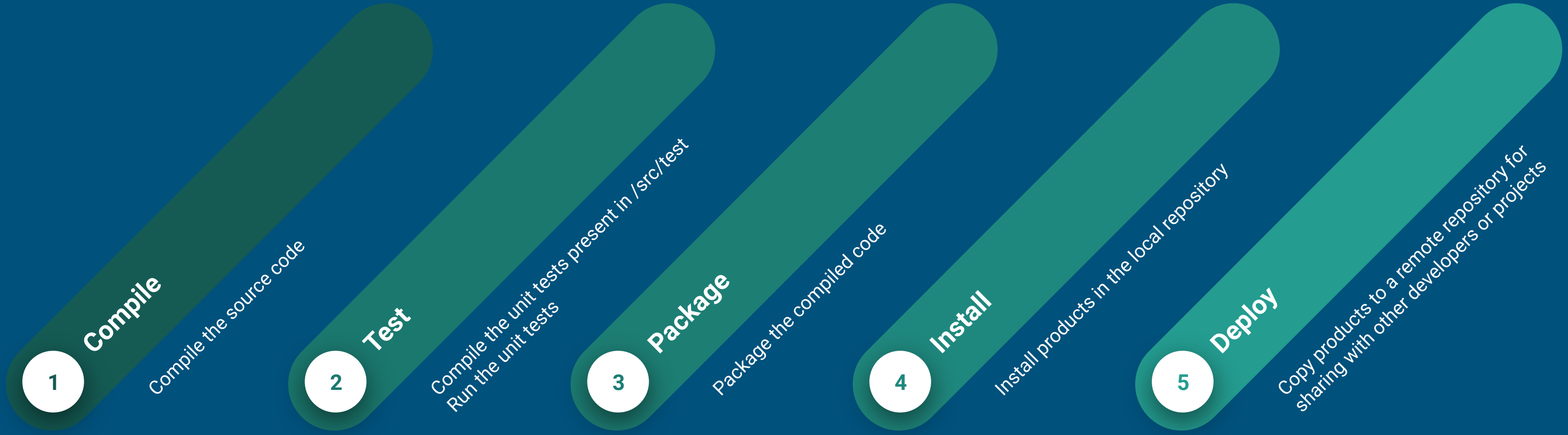
- Dependencies are stored in repositories :
    - Local repository
      (default : *$HOME/.m2/repository*)
    - Remote repositories
      (default : https://repo.maven.apache.org)

- Dependencies are recovered in priority in :
    1. Local repository
    2. Remote repository (downloaded and stored in local repository)

Convention over configuration

Directory	**Contents**
Project home	pom.xml and following directory
src/main/java	Deliverable Java source code for the project.
src/main/resources	Deliverable resources for the project, such as properties files.
src/main/webapp	Specific web code source
src/test/java	Testing Java source code for the project.
src/test/resources	Resources necessary for testing.
target	Compiled files and project archive

Lifecycle phases


1
Compile
Compile the source code
2
Test
Compile the unit tests present in /src/test
Run the unit tests
3
Package
Package the compiled code
4
Install
Install products in the local repository
5
Deploy
Copy products to a remote repository for sharing with other developers or projects

These lifecycle phases are executed sequentially to complete the lifecycle

How to use Maven ?

Minimal POM

- Root element : project

- Mandatory element
  - modelVersion : should be set to *4.0.0*
  - groupId : Id of the project’s group
  - artifactId : Id of the artifact
  - version : version of the artifact under the specified group
  - packaging : type of archive generated (jar/war)

```xml
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>hei.devops</groupId>
    <artifactId>lesson-02</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>
</project>
```

Properties Element

- Element under project with tag properties
  - To define properties create a tag (property name) with the value
  - To call property value use ${tag}

- Very useful properties :
  - project.build.sourceEncoding : should be set to *UTF-8*
  - maven.compiler.target : target argument for java compiler (default value : 1.6)
  - maven.compiler.source : source argument for java compiler (default value : 1.6)

```
<properties>
   <project.build.sourceEncoding>
         UTF-8
  </project.build.sourceEncoding>
  <java.version>1.8</java.version>
  <maven.compiler.target>
         ${java.version}
  </maven.compiler.target>
  <maven.compiler.source>
         ${java.version}
  </maven.compiler.source>
</properties>
```

Dependencies Element

- Element under project with tag dependencies
    - To declare libraries
    - contains dependency element

- Dependency element contains :
    - groupId : Library project's group id
    - artifactId : Library artifact id
    - version : Version of the library artifact
    - scope (not mandatory) :
        - used to limit the scope of a dependency
        - example test : library is only available during test step

```xml
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.assertj</groupId>
        <artifactId>assertj-core</artifactId>
        <version>3.8.0</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.mockito</groupId>
        <artifactId>mockito-all</artifactId>
        <version>1.9.5</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Maven command lines

Actions	**Command**
Clean up project	*mvn clean*
Execute unit tests	*mvn test*
Package compiled code in target directory	*mvn package*
Install products in the local repository	*mvn install*
Upload products in the remote repository	*mvn deploy*

You can merge clean action and other action (goals) :
mvn clean install

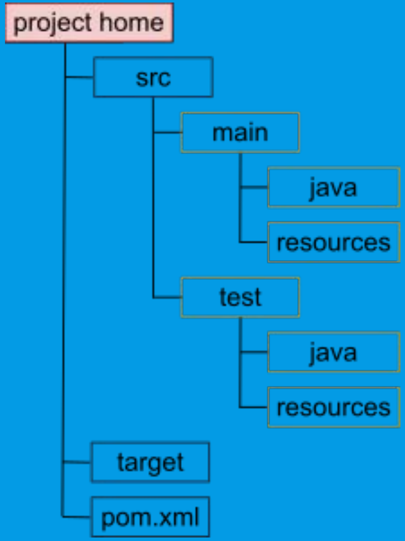
Question

What's going to happen ?

pom.xml

```xml
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>hei.devops</groupId>
    <artifactId>lesson-02</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <dependencies>
    …
    </dependencies>
</project>
```



```
mvn clean install
```

Thank you for you attention !

Links :

- Sources
  - http://www.commitstrip.com/
  - https://giphy.com/
  - https://maven.apache.org/