Déploiement d’un premier projet maven.

Sep 2018

Walid YAICH
walid.yaich@esprit.tn

Plan

- Un simple projet Java
- Difficultés
- Solution
- Apache Maven
- La structure par défaut d’un projet Maven
- Le POM (Project Object Model)
- Maven Workflow
- Phase
- Lifecycle
- Notez bien

Un simple projet Java

Les **dépendances** de cette classe sont :

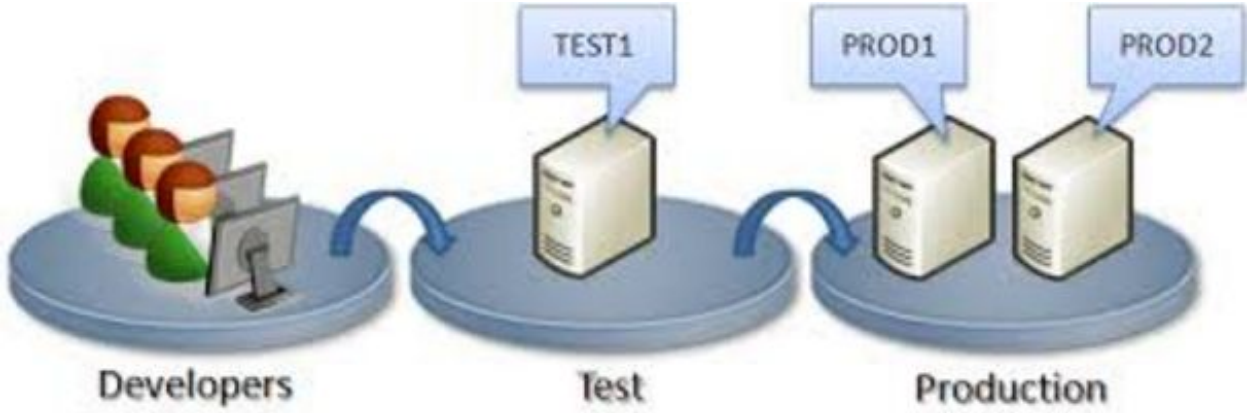
- Httpclient
- json

```java
public class CallRestWebService {

    public static final String endpoint = "http://ip-api.com/json";

    public static void main(String[] args) {
            HttpClient client = new DefaultHttpClient();
            HttpGet request = new HttpGet(endpoint);
            String ip = "not found";
            try {
                HttpResponse response = client.execute(request);
                String jsonResponse = EntityUtils.toString(response.getEntity());
                System.out.println("Response as String : " + jsonResponse);
                JSONObject responseObj = new JSONObject(jsonResponse);

                ip = responseObj.getString("query");
            } catch (IOException e) {
                e.printStackTrace();
            }
    }
}
```

```
Response as String : {"as":"AS37492 ORANGE-","city":"Le
Bardo","country":"Tunisia","countryCode":"TN","isp":"Orange
Tunisia","lat":36.8092,"lon":10.1406,"org":"ORANGE-","query":"196.229.170.30","region":"1
","regionName":"Gouvernorat de Tunis","status":"success","timezone":"Africa/
Tunis","zip":""}
```

Difficultés

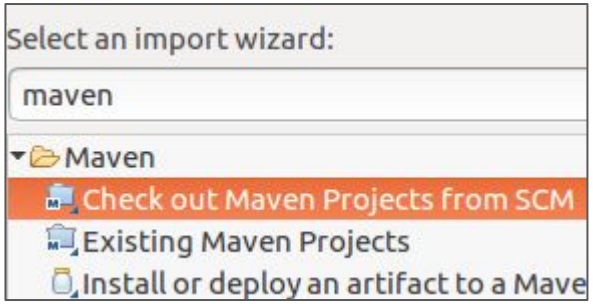
- Recherche des dépendances sur internet ⇒ perte de temps
- Faire pointer le projet sur un chemin local ⇒ projet qui fonctionne qu'en local
- Mettre les dépendances dans le projet ⇒ mauvaise pratique
- Compiler avec la ligne de commande ⇒ compliqué
- Utilisation de l'IDE pour compiler ⇒ quoi faire pour livrer ?



Déploiement : Le transfert de l'application vers le serveur cible

Solution

1) Vérifier la JDK : echo %JAVA_HOME%
2) Ajouter les bin Maven a la Variable d'environnement PATH
3) File->import



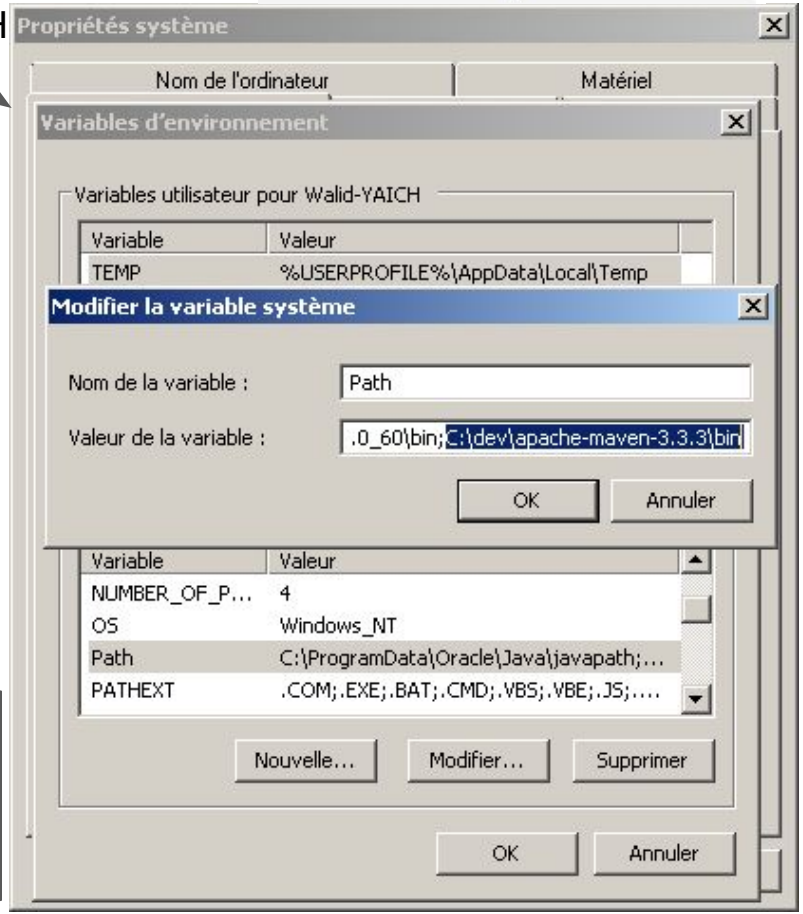
4) https://gitlab.com/Walid-YAICH/callRestWebService.git

5) Aller au dossier qui contient le pom.xml

mvn clean compile exec:java

Output :

```
Response as String : {"as":"AS37492 ORANGE-","city":"Le
Bardo","country":"Tunisia","countryCode":"TN","isp":"Orange
Tunisia","lat":36.8092,"lon":10.1406,"org":"ORANGE-","query":"196.229.170.30","region":"1
","regionName":"Gouvernorat de Tunis","status":"success","timezone":"Africa/
Tunis","zip":""}
```



Apache Maven

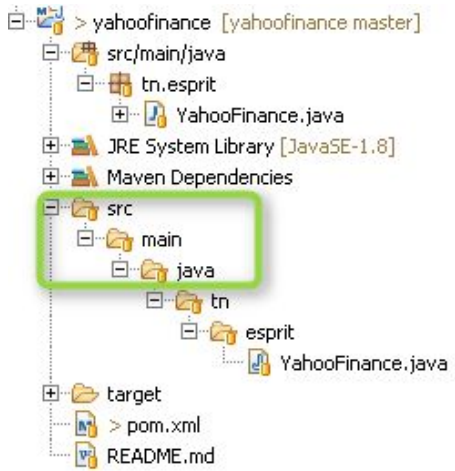
Apache Maven est un outil de construction, basé sur :

- le concept du **POM** (Project Object Model).
- le respect de **conventions.**

Apache Maven est **Open source.**

La structure par défaut d'un projet Maven

Maven recommande l'utilisation de cette structure de dossier, mais, c'est possible aussi de faire autrement.



`src/main/java`	Application/Library sources
`src/main/resources`	Application/Library resources
`src/main/resources-filtered`	Application/Library resources w
`src/main/filters`	Resource filter files
`src/main/webapp`	Web application sources
`src/test/java`	Test sources
`src/test/resources`	Test resources
`src/test/resources-filtered`	Test resources which are filtere
`src/test/filters`	Test resource filter files
`src/it`	Integration Tests (primarily for p
`src/assembly`	Assembly descriptors
`src/site`	Site
`LICENSE.txt`	Project's license
`NOTICE.txt`	Notices and attributions require
`README.txt`	Project's readme

Le POM (Project Object Model)

- **modelVersion** : La version du modèle de POM utilisée
- **groupId** : Le groupe ou l'organisation qui développe le projet. C'est une des clés utilisée pour identifier de manière unique le projet et ainsi éviter les conflits de noms.
- **artifactId** : C'est le nom du livrable sans version (jar, ear ...).
- **version** : La version en cours du projet (Le suffixe -SNAPSHOT indique une version en cours de développement).

```
<modelVersion>4.0.0</modelVersion>
<groupId>tn.esprit.javaee</groupId>
<artifactId>yahoofinance</artifactId>
<version>0.0.1-SNAPSHOT</version>
```

Le POM (Project Object Model)

La balise **<dependencies>** devra contenir toutes les dépendances nécessaires au fonctionnement du projet.

```
<dependencies>
    <dependency>
        <groupId>org.json</groupId>
        <artifactId>json</artifactId>
        <version>20160810</version>
    </dependency>

    <dependency>
        <groupId>org.apache.httpcomponents</groupId>
        <artifactId>httpclient</artifactId>
        <version>4.1.1</version>
    </dependency>

</dependencies>
```

Moteur de recherche des dépendances Maven :

http://mvnrepository.com

Le POM (Project Object Model)

Pour utiliser ce plugin :
mvn compile **exec:java**

exec : l'identifiant du plugin

java : l'identifiant du goal

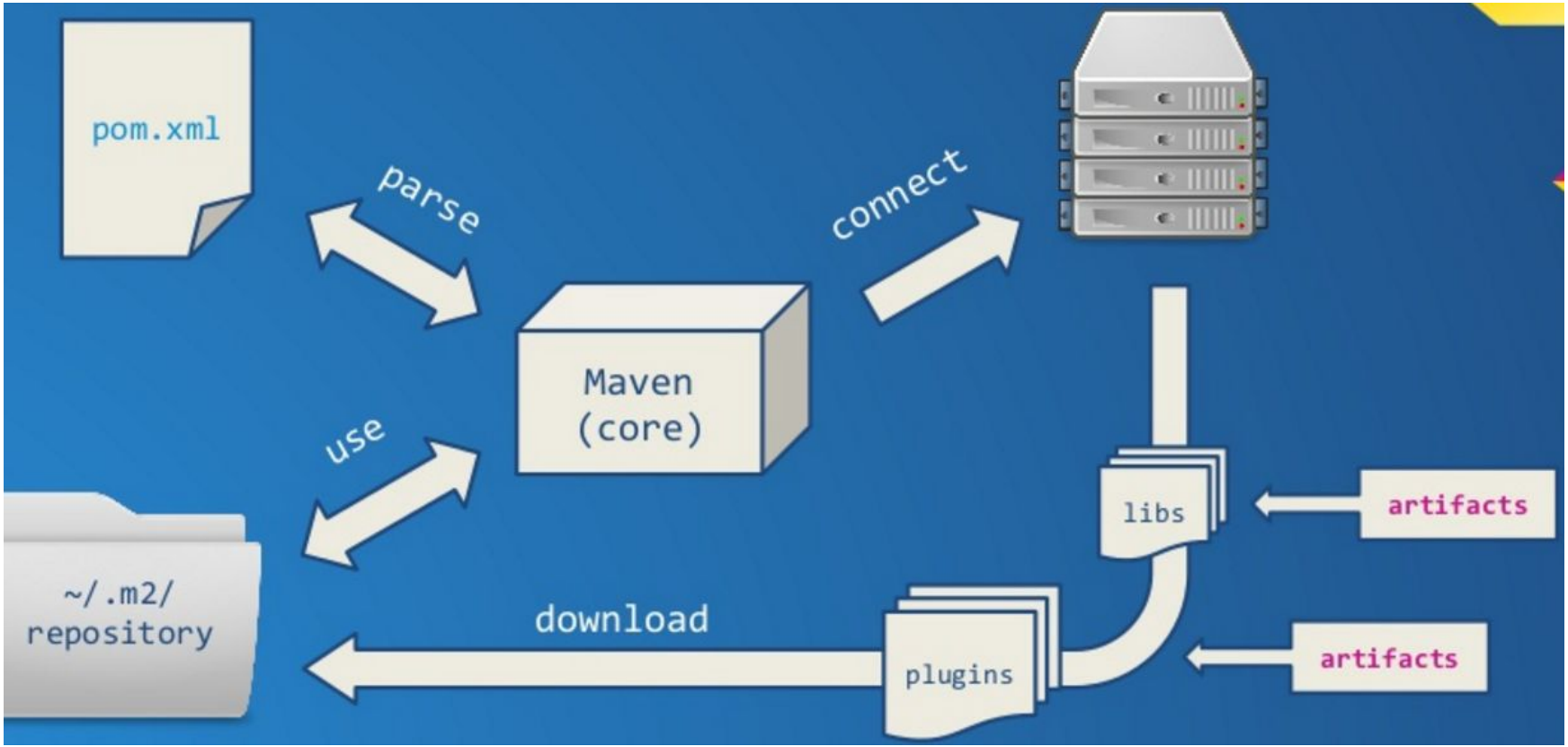
```
<plugins>
    <plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>exec-maven-plugin</artifactId>
        <version>${exec-maven-plugin.version}</version>
        <executions>
            <execution>
                <goals>
                    <goal>java</goal>
                </goals>
            </execution>
        </executions>
        <configuration>
            <mainClass>${project-mainClass}</mainClass>
        </configuration>
    </plugin>
</plugins>
```

Le POM (Project Object Model)

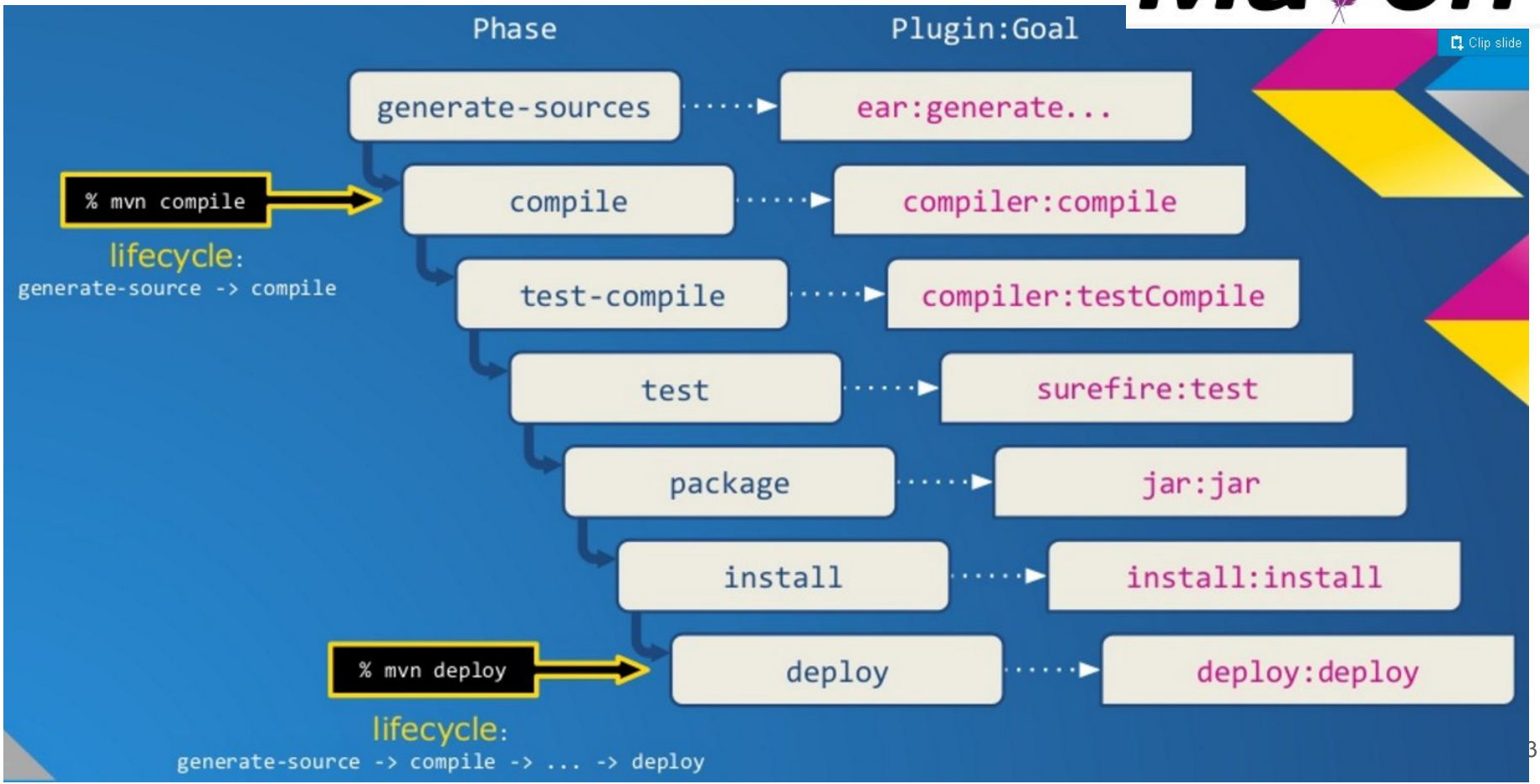
```xml
<properties>
    <maven.compiler.target>1.8</maven.compiler.target>
    <maven.compiler.source>1.8</maven.compiler.source>
    <exec-maven-plugin.version>1.3.2</exec-maven-plugin.version>
    <project-mainClass>tn.esprit.YahooFinance</project-mainClass>
</properties>

<plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>exec-maven-plugin</artifactId>
    <version>${exec-maven-plugin.version}</version>
    <executions>
        <execution>
            <goals>
                <goal>java</goal>
            </goals>
        </execution>
    </executions>
    <configuration>
        <mainClass>${project-mainClass}</mainClass>
    </configuration>
</plugin>
```

maven.compiler.target et **maven.compiler.source** sont les propriétés du plugin Apache Maven Compiler Plugin, c'est le plugin qui nous permet de compiler notre projet Java, ces propriétés sont utiles pour spécifier la version du compilateur souhaitée(source et target).

Maven Workflow



Lifecycle


Maven™
Phase
Plugin:Goal
Clip slide
generate-sources
ear:generate...
% mvn compile
compile
compiler:compile
lifecycle:
generate-source -> compile
test-compile
compiler:testCompile
test
surefire:test
package
jar:jar
install
install:install
% mvn deploy
deploy
deploy:deploy
lifecycle:
generate-source -> compile -> ... -> deploy

Notez bien

l'IDE est juste un **intermédiaire** entre le développeur et les différents produits tel que Maven, Wildfly et Git.