

MAVEN

Gestion de projet java :
compilation, tests unitaires, packaging, reporting

→ Homogénéiser les pratiques

→ Faciliter la compilation, la production d'exécutable

→ Automatiser la gestion de dépendances

→ Génération de site de reporting

Commandes maven usuelles pour réaliser des tâches

Build

maven compile

Compilation (avec recherches sur Internet des jars nécessaires)

maven test

Réalisation des tests unitaires

maven package

Packaging (production des jar, war, ear, ...)

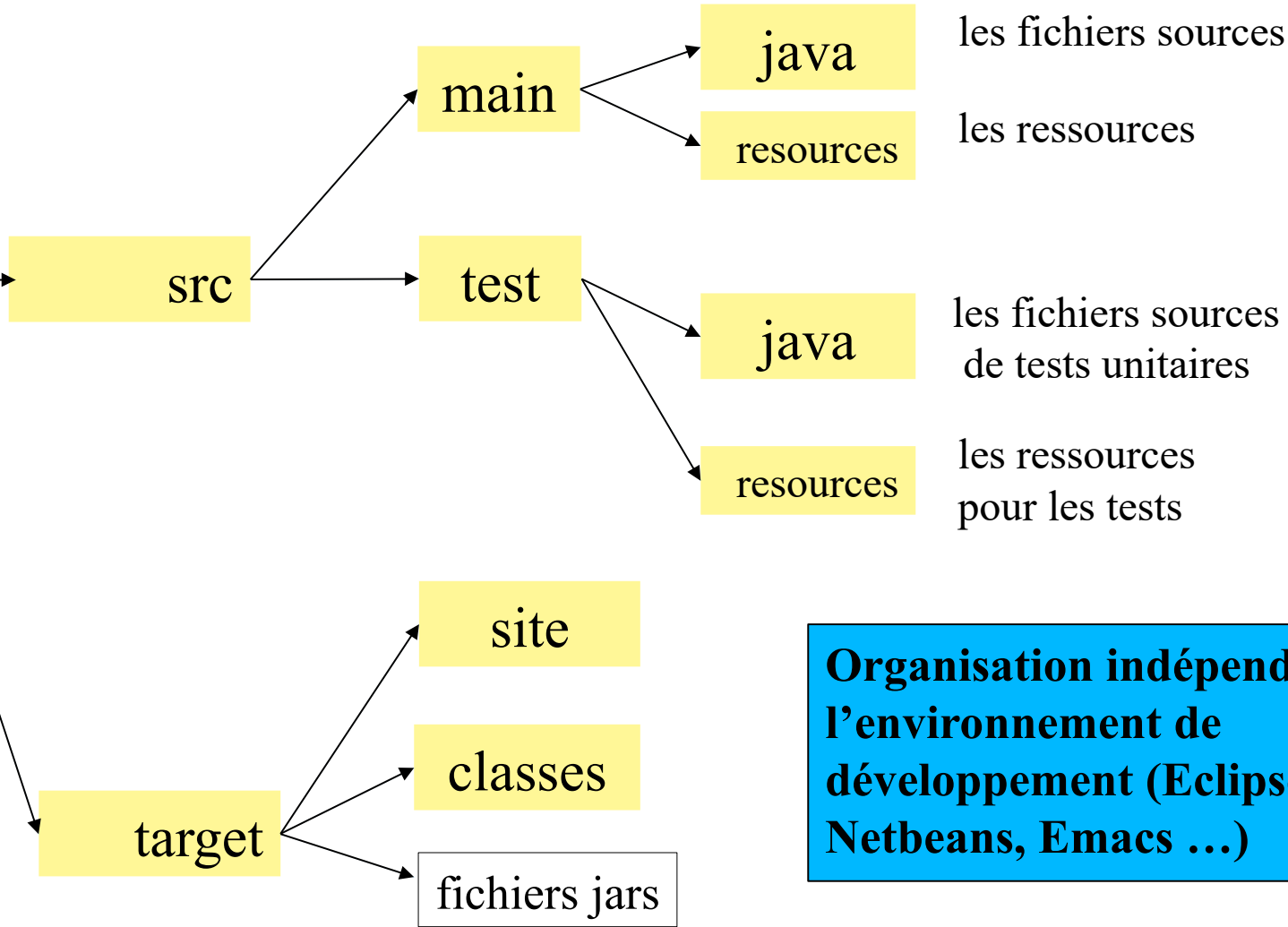
Reporting (génération du site web du projet)

maven site

- bilan des tests unitaires
- documentation javadoc

Les fichiers dans un projet MAVEN organisation par défaut

pom.xml : LE fichier de configuration



Organisation indépendante de l'environnement de développement (Eclipse, Netbeans, Emacs ...)

MAVEN : le fichier de configuration pom.xml

Le fichier pom (Project Object Model) minimum

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>   <!-- 4.0.0 pour Maven 2 et 3-->

  <groupId>fr.parisdescartes.parchemal</groupId>
  <artifactId>test</artifactId>
  <version>0.1</version>
  <packaging>jar</packaging>   // maven package produira test-0.1.jar

</project>
```

MAVEN : cycles de vie, phases de cycle

maven compile **compile, test et package**
maven test sont des phases
maven package du **cycle de vie "par défaut"**

maven site **site**
 est une phase
 du **cycle de vie "Site"**

3 cycles de vie définis dans Maven : "par défaut" "Site" et "Clean"

A chaque cycle de vie correspond une suite ordonnée de **phases**

MAVEN :

Six phases du cycle de vie par défaut

compile

Compile les sources

test-compile

Compile les classes de tests

test

Lance les tests unitaires

package

Prépare la distribution du projet. (archives Jar, War, Ear...)

install

Installe le package en local sur la machine pour pouvoir être réutilisé comme dépendance.

deploy

Déploie le package sur un serveur pour qu'il puisse être réutilisé par tout le monde.



cycle de vie par défaut

compile

test-compile

test

package

install

deploy

Quelques commandes maven usuelles utilisant le cycle de vie par défaut

maven compile compile

maven test compile, test-compile, test

maven package compile, test-compile, test,package

maven deploy compile, test-compile, test,package,install,deploy

Les phases précédentes sont aussi appelées

MAVEN :

Deux autres cycles de vie : Clean et Site

cycle de vie
Clean

pre-clean

clean

post-clean

Quelques commandes maven usuelles
utilisant les cycles Clean et Site

maven clean

cycle de vie Site

pre-site

site

post-site

site-deploy

maven site

maven site-deploy

Terminologie maven

Terme Maven	exemple	
Cycle de vie	"par défaut"	3 cycles de vie : "par défaut" "Site" "Clean"
Phase de cycle de vie	testcompile	Un cycle de vie est une suite ordonnée de phase. Chaque phase donne lieu à l'exécution de goals
Goal		
Plugin		

MAVEN :

Phases et goals

Les actions sont réalisées par des **goals**

A une **phase d'un cycle** peut être associé un (*ou plusieurs*) **goal**

Si aucun goal n'est associé, aucune action ne correspond à la phase

clean

clean:clean

compile

compiler:compile

Test-compile

compiler:testCompile

test

surefire:test

package

Dépend de la valeur de packaging dans pom.xml

install

install:install

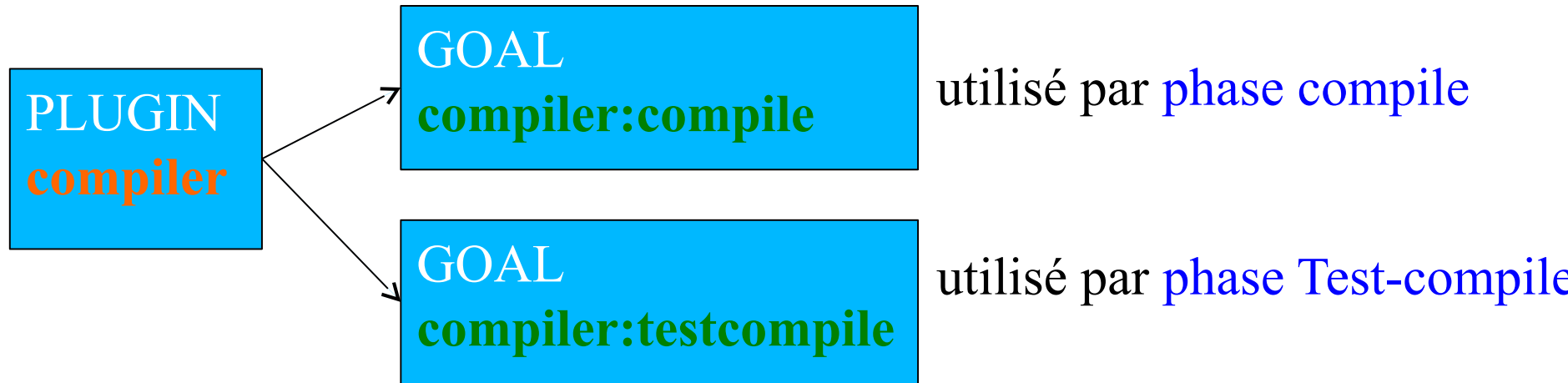
deploy

deploy:deploy

MAVEN : **plugin** et **goals**

Les goals appartiennent tous à un plugin.

Un plugin est une unité dans laquelle un ou plusieurs goals sont définis



MAVEN : **plugins** et **goals** associés

PLUGIN

GOAL

clean	clean:clean
compiler	compiler:compile, compiler:testcompile
deploy	deploy:deploy, deploy:deploy-file
install	install:install, install:install-file, install:install-help
resources	resources:resources:resources:testResources, resources:copy-resources
surefire	surefire:test (execute les tests unitaires)
surefire-report	surefire-report:report
site	site:site, site:deploy, site:run, site:stage, site:stage-deploy, site:attach-descriptor, site:jar, site:effective-site
ear	ear:ear, ear:generate-application-xml
jar	jar:jar, jar:test-jar
war	war:war, war:exploded, war:inplace, war:manifest
javadoc	javadoc:javadoc, ... (14 goals)
archetype	archetype:create archetype:create-from-project

Terminologie maven

Terme Maven	exemple	
Cycle de vie	"par défaut"	3 cycles de vie : "par défaut" "Site" "Clean"
Phase de cycle de vie	testcompile	Un cycle de vie est une suite ordonnée de phase. Chaque phase donne lieu à l'exécution de goals
Goal	compile:testcompile	Une commande maven donne lieu à l'exécution de goals Une phase de cycle de vie peut être relié à un ou plusieurs goals : l'exécution d'une phase se traduit par l'exécution des goals associés
Plugin	compile	Un goal appartient à un plugin. Un plugin est composé de un ou plusieurs goals

La commande maven syntaxe

maven [<phase>/<goal>]*

exemples :

maven clean install 2 phases

maven surefire-report:report 1 goal

maven clean install surefire-report:report 2 phases et 1 goal

MAVEN

pom et super pom

super pom

livré avec maven

Définit les plugins à utiliser, la structure des fichiers ...

pom.xml du projet

définit les spécificités du projet.

Il hérite du super pom

super pom

pom.xml du projet

effective pom

MAVEN

Téléchargement et utilisation

Téléchargement

maven.apache.org (version 3.5.0)

Utilisation

En commande ligne

cd <adresse du répertoire du projet (qui contient pom.xml) >
maven **package** (compile, fait les tests, construit le jar)

Avec eclipse (Maven intégré)

run as ... maven build ... préciser **package**

MAVEN

création de projets maven

En commande ligne

mvn archetype:create

-DgroupId=**up5.mi.pary**

-DartifactId=**test**

-DarchetypeArtifactId=**maven-archetype-quickstart**

créé :

- le fichier pom.xml
- la structure des fichiers
- une classe App et une classe AppTest

Avec eclipse

Nouveau Projet ... projet Maven

choisir **maven-archetype-quickstart**

MAVEN : pom.xml

Fixer la version de la JVM à utiliser

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.3</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
      </configuration>
    </plugin>
  </plugins>
</build>
```

Sous Eclipse : maven/update project pour prendre en compte le nouveau pom.xml

```
<dependencies>
```

```
  <dependency>
```

```
    <groupId>junit</groupId>
```

```
    <artifactId>junit</artifactId>
```

```
    <version>4.12</version>
```

```
    <scope>test</scope> ne sert que pour la phase de test
```

```
  </dependency>
```

```
</dependencies>
```

La recherche du jar est faite sur le repository maven
Le jar est stocké sur le repository local

MAVEN repositories

Les jars sont stockés dans des repositories

repository local

situé dans le répertoire ~/.m2

repository maven

<http://repo.maven.apache.org/maven2/>

repository spécifique

<http://www.mi.parisdescartes.fr/~pary/java/repository>

Si un jar est recherché et qu'il n'est pas dans le repository local :

- il est recherché dans les autres repositories
- puis il est stocké dans le repository local

Un jar peut nécessiter la récupération d'autres jars (dépendances)

MAVEN : compléments

Utiliser aussi un repository « interne »

```
<repositories>
  <repository>
    <id>repository</id>
    <url>http://www.mi.parisdescartes.fr/~pary/java/repository</url>
  </repository>
</repositories>
```

```
<dependency>
  <groupId>up5.mi.pary</groupId>
  <artifactId>cours.java</artifactId>
  <version>1.2.0</version>
</dependency>
```

MAVEN

pom.xml : sous balises usuelles

```

<project ...>
  <modelVersion>4.0.0</modelVersion>
  <groupId>...</groupId><artifactId>...</artifactId>
  <version>...</version><packaging>...</packaging>
  <build>
    <plugins>
      <!-- les plugins à utiliser pour le build -->
      <plugin> ..... </plugin>
    </plugins>
  </build>

  <dependencies> <!-- les artifacts nécessaires -->
    <dependency> ..... </dependency>
  </dependencies>

  <repositories> <!-- les repositories autres que le repository local et celui de maven.org -->
    <repository> ..... </repository>
  </repositories>

  <reporting>
    <plugins> <!-- les plugins à utiliser pour le reporting -->
      <plugin> ..... </plugin>
    </plugins>
  </reporting>

```

MAVEN : site du projet



cadrage

Last Published: 2015-10-28 | Version: 0.0.1-SNAPSHOT [cadrage](#)

Project Documentation

- ▼ Project Information
 - Dependencies
 - Dependency
 - Convergence
 - Dependency Information
 - About**
 - Plugin Management
 - Project Plugins
 - Project Summary
- Project Reports

Built by:  **maven**

About cadrage

Ceci est un projet créé pour illustrer le cours Maven de Yannick Parchemal

cadrage

Last Published: 2015-10-28 | Version: 0.0.1-SNAPSHOT

Project Documentation

▶

Project Information

▼


Project Reports

Surefire Report

JavaDocs

Test JavaDocs

Built by:



maven

Generated Reports

This document provides an overview of the various reports that are automatically generated by [Maven](#) 📦. Each report is briefly described below.

Overview

Document	Description
Surefire Report	Report on the test results of the project.
JavaDocs	JavaDoc API documentation.
Test JavaDocs	Test JavaDoc API documentation.

Surefire Report

Summary

[\[Summary\]](#)
[\[Package List\]](#)
[\[Test Cases\]](#)

Tests	Errors	Failures	Skipped	Success Rate	Time
3	0	0	0	100%	0.005

Note: failures are anticipated and checked for with assertions while errors are unanticipated.

Maven

Ajout d'information dans le site

```
<reporting> <plugins> .....
```

Ajout du bilan des tests unitaires

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-surefire-report-plugin</artifactId>
  <version>2.19</version>
</plugin>
```

Ajout de la documentation javadoc dans le reporting

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-javadoc-plugin</artifactId>
  <version>2.10.3</version>
  <configuration> <show>public</show></configuration>
</plugin>
```