APACHE MAVEN

PLUGINS

PLUGINS

QUELQUES PLUGINS - CORE

- **maven-clean-plugin**
  - Supprime le répertoire de sortie utilisé pour le projet (par défaut ${basedir}/target
- **maven-compiler-plugin**
  - Compiler les sources Java
- **maven-deploy-plugin**
  - Déploie un artefact dans un dépôt distant
- **maven-failsafe-plugin**
  - Lance les tests Junit dans un classloader isolé
- **maven-install-plugin**
  - Installe un artefact dans le dépôt local
- **maven-resources-plugin**
  - Copie des ressources vers le répertoire de sortie (pour un packaging par exemple)
- **maven-site-plugin**
  - Générer un site pour le projet courant
- **maven-surefire-plugin**
  - Exécute des tests unitaires dans un classloader isolé

QUELQUES PLUGINS - PACKAGING

- **maven-ear-plugin**
  - construit un EAR pour le projet courant
- **maven-ejb-plugin**
  - construit un EJB pour le projet courant
- **maven-jar-plugin**
  - construit un JAR pour le projet courant
- **maven-rar-plugin**
  - construit un RAR pour le projet courant
- **maven-war-plugin**
  - construit un WAR pour le projet courant
- **maven-shade-plugin**
  - construit un "Uber-JAR" (JAR qui inclut toutes les dépendances) pour le projet courant
- **maven-source-plugin**
  - construit un JAR contenant les sources du projet courant

QUELQUES PLUGINS - REPORTING

- **maven-changelog-plugin**
  - Génère une liste de changement récents depuis le gestionnaire de version du projet
- **maven-changes-plugin**
  - Génère un rapport depuis un gestionnaire de bugs
- **maven-checkstyle-plugin**
  - Génère un rapport checkstyle
- **maven-javadoc-plugin**
  - Génère la documentation java du projet
- **maven-surefire-report**
  - Génère un rapport d'exécution des tests

QUELQUES PLUGINS - OUTILS

- **maven-ant-plugin**
  - Génère un fichier de build ANT pour le projet
- **maven-antrun-plugin**
  - Exécute des tâches ANT
- **maven-archetype-plugin**
  - Génère un modèle de projet
- **maven-assembly-plugin**
  - Construit un livrable personnalisable (zip par exemple) du projet
- **maven-dependency-plugin**
  - Permet de manipuler (copier, désarchiver) et analyser des dépendances
- **maven-enforcer-plugin**
  - Permet de vérifier les contraintes de l'environnement d'exécution (version de maven, version du JDK, …)

QUELQUES PLUGINS - AUTRES

- **maven-eclipse-plugin**
  - Génère les fichiers de projet Eclipse
- **sql-maven-plugin**
  - Exécute des scripts SQL
- **cargo-maven2-plugin**
  - Démarrer/Arrêter/Configurer un container Java EE et y déployer des packages

CONFIGURER PLUGINS

```
<build>
  <finalName>${project.groupId}-banko</finalName>
  <plugins>
    <plugin>
      <groupId>org.apache.maven</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.3</version>
      <configuration>
          <source>1.8</source>
          <target>1.8</target>
      </configuration>
    </plugin>
  </plugins>
</build>
```

GOAL

- Un "goal" dans Maven représente une action (tâche) à exécuter par un plugin.

- Lancer le goal "clean" du plugin "maven-clean-plugin"
  - mvn clean:**clean**

- Générer les fichiers projets Eclipse
  - mvn eclipse:**eclipse**

CYCLE DE VIE DU BUILD

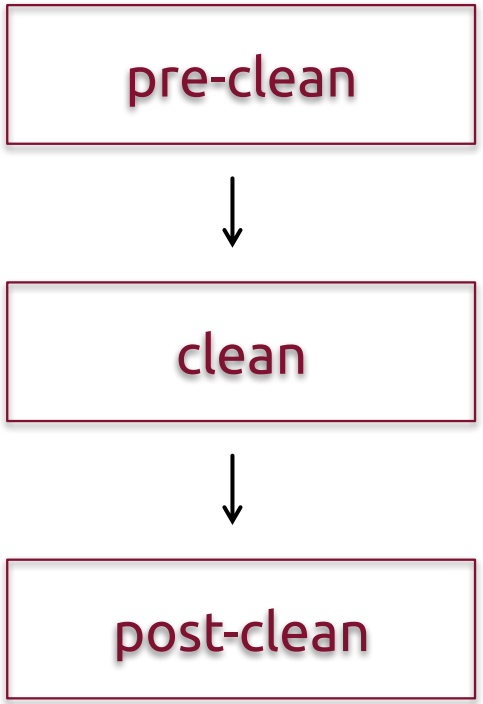
CYCLE DE VIE

- Le cycle de vie d'un projet Maven est composé d'une **séquence de phases** dans lesquelles sont exécutés des **actions** ("goals")

- Maven propose 3 types de cycle de vie :
  - **clean**
  - **default** (build)
  - **site**

CYCLE DE VIE "CLEAN"

- L'exécution de la commande "mvn clean" invoque le cycle de vie "clean".
- Il est composé de 3 phases : **pre-clean**, **clean** et **post-clean**
- L'action "clean" du plugin "maven-clean-plugin" supprime le répertoire de build (par défaut ${basedir}/target)



PLUGINS, GOAL ET PHASE

- Il est possible de configurer l'exécution d'une tâche ("goal") d'un plugin dans une phase Maven.

```
<build>
      <plugins>
          <plugin>
            <groupId>org.codehaus.mojo</groupId>
            <artifactId>exec-maven-plugin</artifactId>
            <version>1.4.0</version>
            <configuration>
                <executable>cp</executable>
                <arguments>
                    <argument>-R</argument>
                    <argument>target</argument>
                    <argument>target-backup</argument>
                </arguments>
            </configuration>
            <executions>
              <execution>
                <phase>pre-clean</phase>
                <goals><goal>exec</goal></goals>
              </execution>
            </executions>
      </plugin>
    </plugins>
</build>
```

CYCLE DE VIE "BUILD" (1)

- Il s'agit du cycle de vie par défaut qui commence par une phase "validate" et se termine par la phase "deploy".

Phase du cycle de vie	Description
validate	Valide le projet et la disponibilité de toutes les informations requises pour le build
generate-sources	Génère le code source nécessaire pour l'inclure à la compilation
process-sources	Traite le code source, pour filtrer certaines valeurs par exemple
generate-resources	Génère les ressources à inclure dans le package
process-resources	Copie et traite les ressources dans leur répertoire destination, afin qu'elles soient prêtes pour le packaging
compile	Compile le code source du projet
process-classes	Traite à posteriori les fichiers générés par la compilation, pour modifier du bytecode par exemple
generate-test-sources	Génère le code source des tests pour l'inclure à la compilation
process-test-sources	Traite le code source des tests, pour filtrer certaines valeurs par exemple

Phase du cycle de vie	Description
generate-test-resources	Génère les ressources pour les tests
process-test-resources	Copie et traite les ressources de test dans le répertoire destination des tests
test-compile	Compile le code source des tests dans le répertoire destination des tests
test	Exécute les tests en utilisant le framework de test approprié. Le code de ces tests ne doit pas être nécessaire au packaging ni au déploiement.
prepare-package	Effectue les opérations nécessaires pour la préparation du package avant que celui-ci ne soit réellement créé. Il en résulte souvent une version dézippée et prête à été packagée du futur package (Maven 2.1+)
package	Package le code compilé dans un format distribuable tel que JAR, WAR ou EAR
pre-integration-test	Effectue les actions nécessaires avant de lancer les tests d'intégration, comme configurer un environnement par exemple.
integration-test	Traite et déploie si nécessaire le package dans l'environnement où les tests pourront être exécutés
post-integration-test	Effectue les actions nécessaires à la fin de l'exécution des tests d'intégration, comme nettoyer l'environnement par exemple

Phase du cycle de vie	Description
verify	Lance des points de contrôle pour vérifier que le package est valide et qu'il passe les critères qualité
install	Installe le package dans le dépôt local, celui- ci pourra ainsi être utilisé comme dépendance par d'autres projets locaux
deploy	Copie le package final sur le dépôt distant. Permet de partager le package avec d'autres utilisateurs et projets (souvent pertinent pour une vraie livraison)

CYCLE DE VIE "SITE"

- Le cycle de vie "site" contient 4 phases
  - pre-site
  - site
  - post-site
  - site-deploy

- Exemple d'exécution appliquant ce cycle de vie :
  - mvn site:site
  - mvn site:deploy

CYCLE DE VIE / TYPE DE PACKAGE

- La balise packaging influe sur les étapes requises pour construire un projet.

JAR : CYCLE DE VIE

Phase du cycle de vie	Goal
process-resources	resources:resources
compile	compiler:compile
process-test-resources	resources:testResources
test-compile	compiler:testCompile
test	surefire:test
package	jar:jar
install	install:install
deploy	deploy:deploy

POM: CYCLE DE VIE

Phase du cycle de vie	Goal
package	site:attach-descriptor
install	install:install
deploy	deploy:deploy

PLUGINS : CYCLE DE VIE

Phase du cycle de vie	Goal
generate-resources	plugin:descriptor
process-resources	resources:resources
compile	compiler:compile
process-test-resources	resources:testResources
test-compile	compiler:testCompile
test	surefire:test
package	jar:jar, plugin:addPluginArtifactMetadata
install	install:install, plugin:updateRegistry
deploy	deploy:deploy

EJB : CYCLE DE VIE

Phase du cycle de vie	Goal
rocess-resources	resources:resources
compile	compiler:compile
process-test-resources	resources:testResources
test-compile	compiler:testCompile
test	surefire:test
package	ejb:ejb
install	install:install
deploy	deploy:deploy

WAR : CYCLE DE VIE

Phase du cycle de vie	Goal
process-resources	resources:resources
compile	compiler:compile
process-test-resources	resources:testResources
test-compile	compiler:testCompile
test	surefire:test
package	war:war
install	install:install
deploy	deploy:deploy

EAR : CYCLE DE VIE

Phase du cycle de vie	Goal
generate-resources	ear:generate-application-xml
process-resources	resources:resources
package	ear:ear
install	install:install
deploy	deploy:deploy

TP 02

- Configurer un plugin
- Comprendre les phases Maven