

# FAQ Maven 2

Date de publication : 17/07/2006

Dernière mise à jour : 17/05/2009

Cette faq a été réalisée à partir des questions fréquemment posées sur les forums de <http://www.developpez.com> et de l'expérience personnelle des auteurs.

Nous tenons à souligner que cette FAQ ne garantit en aucun cas que les informations qu'elle propose soient correctes. Les auteurs font leur maximum, mais l'erreur est humaine. Cette faq ne prétend pas non plus être complète. Si vous trouvez une erreur, ou que vous souhaitez nous aider en devenant rédacteur, veuillez contacter le responsable **Eric Reboisson**, ou poster dans le fil **Participez ici**.

Sur ce, nous vous souhaitons une bonne lecture.

**L'équipe Java**

## Ont contribué à cette FAQ :

Jibee - Emmanuel Venisse (<http://www.mergere.com>) - Denis  
Cabasson ([Site web](#)) - Eric Reboisson ([Site Web](#)) ([Blog](#)) -

1. Accueil (3) .....	4
2. Terminologie et documentation (10) .....	5
3. Installation et configuration (7) .....	9
4. Utilisation (42) .....	12
4.1. Gestion des dépendances (5) .....	22
4.2. Site (2) .....	25
4.3. Documentation (9) .....	26
5. Les projets multimodules (2) .....	32
6. Développement de plugins (6) .....	34
7. Continuum, serveur d'intégration continue (14) .....	38
7.1. Documentation (2) .....	39
7.2. Installation et configuration (8) .....	40
7.3. Utilisation (4) .....	44
8. Proxies d'entreprise (5) .....	46
8.1. Documentation (2) .....	47
8.2. Archiva (3) .....	48

[Sommaire > Accueil](#)

### Quels sont les droits de reproduction de cette FAQ ?

**Auteurs :** [Eric Reboisson](#) ,

Les codes sources présentés sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Pour le reste, ce document constitue une oeuvre intellectuelle protégée par les droits d'auteurs.

Pour toutes les contributions de cette page : Copyright © 2007 Developpez LLC : Tous droits réservés Developpez LLC. Aucune reproduction ne peut en être faite sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts. Cette page est déposée à la SACD.

### Comment participer à cette FAQ ?

**Auteurs :** [Eric Reboisson](#) ,

Cette FAQ est ouverte à toute collaboration. Pour éviter la multiplication des versions, il serait préférable que toute collaboration soit transmise aux administrateurs de la FAQ. Plusieurs compétences sont actuellement recherchées pour améliorer cette FAQ :

**Rédacteur :** bien évidemment, toute nouvelle question/réponse est la bienvenue.

**Web designer :** toute personne capable de faire une meilleure mise en page, une feuille de style ou de belles images...

**Correcteur :** malgré nos efforts, des fautes d'orthographe ou de grammaire peuvent subsister. Merci de contacter les administrateurs si vous en débusquez une... Idem pour les liens erronés.

### Comment bien utiliser cette FAQ ?

**Auteurs :** [Eric Reboisson](#) ,

**Le but :**

Cette FAQ a été conçue pour être la plus simple possible d'utilisation. Elle tente d'apporter des réponses simples et complètes aux questions auxquelles sont confrontés tous les débutants (et les autres).

**L'organisation :**

Les questions sont organisées par thème, les thèmes pouvant eux-mêmes contenir des sous thèmes. Lorsqu'une question porte sur plusieurs thèmes, celle-ci est insérée dans chacun des thèmes rendant la recherche plus facile.

**Les réponses :**

Les réponses contiennent des explications et des codes sources. Certaines sont complétées de fichier à télécharger contenant un programme de démonstration. Ces programmes sont volontairement très simples afin qu'il soit aisé de localiser le code intéressant. Les réponses peuvent également être complétées de liens vers d'autres réponses ou vers un autre site en rapport.

**Nouveautés et mises à jour :**

Lors de l'ajout ou de la modification d'une question/réponse, un indicateur est placé à côté du titre de la question. Cet indicateur reste visible pour une durée de 15 jours afin de vous permettre de voir rapidement les modifications apportées.

J'espère que cette FAQ pourra répondre à vos questions. N'hésitez pas à nous faire part de tout commentaire/remarque/critique.

## Sommaire > Terminologie et documentation

### Qu'est ce que Maven ?

Auteurs : [Eric Reboisson](#) ,

Maven est essentiellement un outil de gestion et de compréhension de projet.

Maven offre des fonctionnalités de :

- **Construction** , compilation
- **Documentation**
- **Rapport**
- **Gestion des dépendances**
- **Gestion des sources**
- **Mise à jour de projet**
- **Déploiement**


### Où trouver de la documentation ?


Auteurs : [Eric Reboisson](#) , [Denis Cabasson](#) ,

Sur developpez.com :

 [Introduction à Maven 2 par Denis Cabasson](#)

Le site de Maven à l'adresse suivante  <http://maven.apache.org>

De la documentation sur les plugins sur le site de Maven à l'adresse suivante :  <http://maven.apache.org/plugins/index.html>



Mojo est un projet de la fondation codehaus qui héberge un grand nombre de plugins pour Maven. La documentation concernant ces plugins peut être trouvée à l'adresse suivante :  <http://mojo.codehaus.org/>

 [Mergere propose en téléchargement gratuit !\[\]\(990e790e6efb89997a442ee76392bbf4\_img.jpg\) un ebook gratuit intitulé "Better Builds with Maven" sur Maven 2.](#)

### Où trouver du support commercial pour Maven 2 ?

Auteurs : [Emmanuel Venisse](#) ,

 [DevZuz](#) est un cabinet de consultants réunissant les plus grands noms du PMC (Project Management Committee) de Maven 2, [Brett Porter](#), [Maria Odea Ching](#) ou encore [Emmanuel Venisse](#) (que nos forumeurs connaissent bien).

Ce cabinet présente donc une expertise reconnue sur les solutions basées sur Maven et propose en particulier en une offre packagée de Maven/Continuum/Archiva sous le nom de  [Maestro](#) et un ebook gratuit intitulé  ["Better Builds with Maven"](#) sur Maven 2, écrit là encore par des membres influents du projet.

 [Sonatype](#) est un autre cabinet de consultants regroupant d'autres membres du PMC Maven. Cette société a également un livre gratuit en ligne :  [Maven: The Definitive Guide](#).

### Qu'est ce que le POM ?

Auteurs : [Eric Reboisson](#) ,

Le POM (Project Object Model) est une façon de décrire, de manière déclarative, un projet au sens de Maven.

Cette description est contenue dans le fichier pom.xml présent dans le repertoire de base du projet.  
Le fichier pom.xml contient donc tous les éléments permettant de gérer le cycle de vie du projet.

#### Exemple d'un fichier pom.xml

```
<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/maven-v4_0_0.xsd">

  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>Maven Quick Start Archetype</name>
  <url>http://maven.apache.org</url>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

</project>
```

- **project** : c'est la balise racine de tous les fichiers pom.xml.
- **modelVersion** : cette balise indique la version de POM utilisée. Bien que cette version ne change pas fréquemment, elle est obligatoire afin de garantir la stabilité d'utilisation.
- **groupId** : cette balise permet d'identifier un groupe qui a créé le projet. Cette clé permet d'organiser et de retrouver plus facilement et rapidement le projet.
- **artifactId** : cette balise indique un nom unique utilisé pour nommer les artifacts à construire.
- **packaging** : type de packaging du projet ( ex : JAR, WAR, EAR, etc.).
- **version** : version de l'artifact généré par le projet.
- **name** : nom du projet.
- **url** : adresse du site du projet.
- **description** : description du projet.
- **dependencies** : balise permettant de gérer les dépendances.

Voir aussi :  [La documentation de référence du POM](#)

## Qu'est-ce qu'un archetype ?

**Auteurs : Emmanuel Venisse , Eric Reboisson ,**

Un archetype est un template de projet.

Le fait d'utiliser des archetypes pour initialiser un projet permet de gagner du temps et de respecter une certaine convention.

Voir aussi :

 [Introduction to Archetypes](#)


 **Guide to Creating Archetypes****Qu'est ce qu'une dépendance ?****Auteurs : Denis Cabasson , Eric Reboisson ,**

Une dépendance est une référence vers un artefact spécifique contenu dans un repository.  
Cet artefact est nécessaire pour une ou plusieurs phases du cycle de vie du projet.  
L'exemple le plus simple est une dépendance sur une bibliothèque jar qui permet d'en utiliser le contenu dans le projet.

**Qu'est ce qu'un artefact ?****Auteurs : Denis Cabasson , Eric Reboisson ,**

Dans Maven, un artefact est un élément spécifique issu de la construction du logiciel.  
Dans JAVA, les artefacts les plus communs sont des JARs, mais ce peut être aussi un fichier WAR, un EAR, un ZIP, etc...

**Qu'est ce que le groupId/artifactId ?****Auteurs : Denis Cabasson , Eric Reboisson ,**

Le groupId est l'identifiant du groupe, à l'origine du projet. GroupId suit les mêmes règles de nommage que les packages  Java (exemple : fr.masociete.monprojet), et on choisit généralement comme groupId le nom du top package du projet.  
L'artifactId est l'identifiant du projet au sein de ce groupe.  
L'artifactId est utilisé par défaut pour construire le nom de l'artefact final (exemple : pour un artifactId=monprojet, le nom du fichier jar généré sera monprojet-version.jar).

**Qu'est ce qu'un SNAPSHOT ?****Auteurs : Denis Cabasson , Eric Reboisson ,**

Par convention, une version en cours de développement d'un projet voit son numéro de version suivi d'un -SNAPSHOT. Ainsi un projet en version 2.0-SNAPSHOT signifie que cette version est une pré-version de la version 2.0, en cours de développement.  
Ce concept de SNAPSHOT est particulièrement important pour Maven. En effet, dans la gestion des dépendances, Maven va chercher à mettre à jour les versions SNAPSHOT régulièrement pour prendre en compte les derniers développements.  
Utiliser une version SNAPSHOT permet de bénéficier des dernières fonctionnalités d'un projet, mais en contre-partie, cette version peut être (et est) appelée à être modifiée de façon importante, sans aucun préavis.

Pour plus de renseignements sur la gestion des versions dans Maven, voir aussi :  [Dependency Mediation and Conflict Resolution](#)

### Qu'est ce que le repository local, distant ?

Auteurs : [Eric Reboisson](#) ,

Un repository local est un répertoire sur le poste du développeur permettant de stocker, suivant la même arborescence, tous les artefacts téléchargés depuis le(s) repository distant(s).

Un projet ayant pour POM :

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>fr.masociete</groupId>
  <artifactId>monprojet</artifactId>
  <version>1.0</version>
</project>
```

sera stocké, dans le repository, suivant cette arborescence :  $\${repository\_home}/fr/masociete/monprojet/1.0/$

## Sommaire > Installation et configuration

### Comment installer Maven ?

Auteurs : [Eric Reboisson](#) ,

Télécharger l'archive maven-2.x.x-bin.zip sur <http://maven.apache.org>  
Décompresser l'archive et copier le répertoire maven-2.x.x à l'endroit de votre choix.  
Ajouter une variable d'environnement %MVN\_HOME% avec pour valeur le chemin du répertoire d'installation maven-2.x.x et ajouter %MVN\_HOME%/bin au PATH.  
Dans une console de commande lancer mvn --version pour vérifier que l'installation est correcte ( la version de Maven 2 doit s'afficher ).

### Comment utiliser un proxy ?

Auteurs : [Denis Cabasson](#) , [Eric Reboisson](#) ,


Dans le fichier settings.xml, situé dans %MVN\_HOME%/conf ou %USER\_HOME%/m2/settings.xml :

#### Configuration du proxy

```
<settings>
...
  <proxies>

    <proxy>
      <active>true</active>
      <protocol>http</protocol>
      <host>proxy.somewhere.com</host>
      <port>8080</port>
      <username>proxyuser</username>
      <password>somepassword</password>
      <nonProxyHosts>www.google.com|*.somewhere.com</nonProxyHosts>
    </proxy>


  </proxies>
...
</settings>
```

 **Attention, les nonProxyHost ne sont pas réellement supportés pour le moment.**

 **Il est préférable de configurer %USER\_HOME%/m2/settings.xml car celui-ci est partagé par toutes les versions de Maven et par Continuum, donc une seule configuration à faire.**

### Comment modifier les options de lancement de Maven ?

Auteurs : [Denis Cabasson](#) , [Eric Reboisson](#) ,

La variable système MAVEN\_OPTS permet d'indiquer des arguments supplémentaires au lancement de Maven.  
En particulier, certains plugins peuvent utiliser beaucoup de mémoire, et les réglages par défaut de la  JVM sont parfois insuffisants.  
Dans ce cas, il suffit de mettre la variable système :

```
MAVEN_OPTS=-Xmx256m -Xms64m
```

Ces options sont alors utilisées par la  JVM lors du lancement de Maven.

## Comment utiliser les versions de SNAPSHOT des plugins ?

Auteurs : Denis Cabasson , Eric Reboisson ,

Les dernières versions des plugins Maven (SNAPSHOT) ne sont pas mises en place sur le repository central. Pour les utiliser, il faut déclarer d'autres repositories, contenant ces versions en cours de développement.

Dans le fichier settings.xml, situé dans %MVN\_HOME%/conf ou %USER\_HOME%/m2/settings.xml il faut configurer :

### Utilisation des SNAPSHOT

```
<profiles>
<profile>
  <id>Maven-Snapshots</id>
  <repositories>
    <repository>
      <id>Maven Snapshots</id>
      <url>http://svn.apache.org/maven-snapshot-repository</url>
      <snapshots>
        <enabled>true</enabled>
      </snapshots>
      <releases>
        <enabled>true</enabled>
      </releases>
    </repository>
  </repositories>
  <pluginRepositories>
    <pluginRepository>
      <id>Maven Snapshots</id>
      <url>http://svn.apache.org/maven-snapshot-repository</url>
      <snapshots>
        <enabled>true</enabled>
      </snapshots>
      <releases>
        <enabled>true</enabled>
      </releases>
    </pluginRepository>
  </pluginRepositories>
</profile>
</profiles>
```

Et

### Activation du profil

```
<activeProfiles>
```

#### Activation du profil

```
<activeProfile>Maven-Snapshots</activeProfile>
</activeProfiles>
```

Vous pouvez répéter l'opération et ajouter un profil utilisant le repository de Mojo : <http://snapshots.maven.codehaus.org/maven2/>

Pour plus d'informations :

 [Maven : Guide to Testing Development Versions of Plugins](#)

 [Mojo : Snapshot Repository](#)

 *Il est préférable de configurer %USER\_HOME%/m2/settings.xml car celui-ci est partagé par toutes les versions de Maven et par Continuum, donc une seule configuration à faire.*

#### Comment installer le Plug-in Maven 2.x pour Eclipse ?

Auteurs : [Eric Reboisson](#) ,

L'adresse web où trouver le Plug-in est : <http://m2eclipse.codehaus.org/>

Pour installer le Plug-in il suffit de suivre la procédure  [Comment installer un Plugin ?](#) de la  [FAQ Eclipse](#).

#### Comment installer le Plug-in Maven 2.x pour Netbeans ?

Auteurs : [Emmanuel Venisse](#) , [Eric Reboisson](#) ,

L'adresse web où trouver le Plug-in est :  <http://mevenide.codehaus.org/m2-site/index.html>

Pour installer le Plug-in, il suffit de suivre la procédure :  [Procédure d'installation du Plug-in Maven 2 pour Netbeans](#)

#### Comment modifier le paramétrage du repository local ?

Auteurs : [Eric Reboisson](#) ,

La valeur par défaut indiquant le repository local est %USER\_HOME%/m2/repository mais elle peut être modifiée. Dans le fichier settings.xml, situé dans %MVN\_HOME%/conf ou %USER\_HOME%/m2/settings.xml :

#### Configuration du proxy

```
<settings>
...
<localRepository>chemin du répertoire</localRepository>
...
</settings>
```

Le chemin vers le repository local doit être absolu.

 *Il est préférable de configurer %USER\_HOME%/m2/settings.xml car celui-ci est partagé par toutes les versions de Maven et par Continuum, donc une seule configuration à faire.*

## Sommaire > Utilisation

### Quelle est la structure des répertoires préconisée par Maven ?

Auteurs : [Denis Cabasson](#) , [Eric Reboisson](#) ,

Maven propose une structure de répertoire "type" pour tous les projets. Il est bien entendu possible d'adopter une autre structure, mais cela est bien souvent très coûteux et compliqué pour un intérêt discutable.

La structure par défaut est la suivante :

#### Structure par défaut d'un projet Maven

```

|-- src
|  |-- main
|  |  |-- java      > Contient le code source java
|  |  |-- resources > Contient les ressources à inclure dans le jar produit (configuration, ...)
|  |  |-- filters   > Filtres permettant de modifier les ressources lors de l'assemblage du jar
|  |  |-- assembly  > Descripteur pour les assemblées Maven
|  |  |-- config    > Fichier de configurations extérieurs au jar
|  |  |-- webapp    > Le contenu de l'application web (projet web)
|  |-- test
|  |  |-- java      > Contient le code source java des tests
|  |  |-- resources > Contient les ressources pour les tests
|  |  |-- filters   > Filtres à appliquer aux ressources de test
|-- site      > Le site de documentation
|-- target    > Répertoire où Maven génère ses fichiers/artefacts
|-- pom.xml   > Descripteur du projet
    
```

Voir aussi :  [Introduction to the Standard Directory Layout](#)

### Comment indiquer à Maven une structure de répertoires différente ?

Auteurs : [Eric Reboisson](#) ,

Il se peut que vous ayez besoin d'utiliser votre propre structure de répertoires, cela se fait en modifiant dans le fichier pom.xml :

#### Exemple de paramétrage des répertoires

```

<project>
  <build>
    ...
    <sourceDirectory>${basedir}/messources</sourceDirectory>
    <scriptSourceDirectory>${basedir}/mescripts</scriptSourceDirectory>
    <testSourceDirectory>${basedir}/mestest</testSourceDirectory>
    <outputDirectory>${basedir}/mesclasses</outputDirectory>
    <testOutputDirectory>${basedir}/mesclassestests</testOutputDirectory>
    ...
  </build>
</project>
    
```

### Comment créer un projet de base ?

Auteurs : [Eric Reboisson](#) ,

Exécuter la commande suivante pour créer un projet basique :

```
mvn archetype:create -DgroupId=fr.monGroupId -DartifactId=MaBaseDeDonnees -  
Dpackagename=fr.monGroupId
```

## Comment créer un projet WEB ?

Auteurs : [Eric Reboisson](#) ,

Dans une console de commande DOS, exécuter la commande suivante pour créer un projet basique :

```
mvn archetype:create -DgroupId=fr.monGroupId -DartifactId=MonApplicationWeb  
-Dpackagename=fr.monGroupId -DarchetypeArtifactId=maven-archetype-webapp
```

## Comment utiliser un projet Maven avec Eclipse ?

Auteurs : [Eric Reboisson](#) ,

Eclipse a besoin de connaître le chemin du repository local de Maven.

Pour cela exécuter la commande suivante :

```
mvn -Declipse.workspace="C:\workspace_eclipse" eclipse:add-maven-repo
```

Pour utiliser Maven comme un outil externe dans Eclipse, dans la barre de menu sélectionner *Window > Preferences*, puis sélectionner *Run/debug > String Substitution*.

Ajouter une nouvelle variable, par exemple *maven\_exec* avec pour valeur *%MVN\_HOME%\bin\mvn.bat*.

Sélectionner ensuite *Run > External Tools*, sélectionner *Program*, dans le champ *location* indiquer la variable *maven\_exec*.

Indiquer dans le champ *working directory* le chemin vers votre projet Maven.

Enfin indiquer les arguments à passer à la commande *mvn* dans le champ des arguments ( e.g : *eclipse:eclipse* pour indiquer la génération de descripteurs de projet Eclipse )

Exécuter ensuite ce programme, puis importer votre projet dans le workspace d'Eclipse.



A noter que vous pouvez également utiliser le Plug-in décrit dans [FAQ Comment installer le Plug-in Maven 2.x pour Eclipse ?](#)

## Comment indiquer à Maven une compilation avec le JDK 1.5 (TIGER) ?

Auteurs : [Eric Reboisson](#) ,

Ajouter au fichier *pom.xml* les balises suivantes :

### Compilation avec JDK 1.5

```
<build>  
  
  <plugins>  
  
    <plugin>  
      <groupId>org.apache.maven.plugins</groupId>  
      <artifactId>maven-compiler-plugin</artifactId>
```

### Compilation avec JDK 1.5

```
<configuration>
  <source>1.5</source>
  <target>1.5</target>
</configuration>
</plugin>

</plugins>

</build>
```

## Comment nettoyer un projet Maven ?

**Auteurs : Eric Reboisson ,**

**Il suffit de taper la commande suivante :**

### Nettoyer un projet

```
mvn clean
```

Les répertoires *project.build.directory* (par défaut *target*), *project.build.outputDirectory* (par défaut *target/classes*), *project.build.testOutputDirectory* (par défaut *target/test-classes*) et *project.reporting.outputDirectory* (par défaut *target/site*) seront entièrement supprimés.

**Il est vivement conseillé d'exécuter la tâche clean avant toute autre opération.**

## Comment ajouter des membres sur un projet ?

**Auteurs : Eric Reboisson ,**

**Dans le fichier pom.xml du projet , ajouter une entrée comme ceci :**

### Ajout d'un membre au projet

```
<developers>

  <developer>
    <id>prenomnom</id>
    <name>Prénom NOM</name>
    <email>prenom.nom@domaine.com</email>
    <url>http://www.societe.fr</url>
    <organization>SOCIETE</organization>
    <organizationUrl>http://www.societe.fr</organizationUrl>

    <roles>
      <role>administrateur</role>
    </roles>

    <timezone>0</timezone>
    <properties><messenger>test</messenger></properties>
  </developer>

</developers>
```

L'id est utilisé par les outils d'intégration continue tel que Continuum pour faire le mapping entre l'auteur d'un commit dans l'outil de gestion de conf est le développeur déclaré dans le POM.

### Comment générer un descripteur Ant pour un projet Maven ?

**Auteurs :** Eric Reboisson ,

**Dans une console de commandes, accéder au répertoire du projet, et exécuter la commande suivante :**

#### Générer un fichier Ant build.xml

```
mvn ant:ant
```

Un fichier build.xml est alors créé à la racine du projet.

### Comment compiler uniquement les sources des tests ?

**Auteurs :** Eric Reboisson ,

**Dans une console de commandes, accéder au répertoire du projet, et exécuter la commande suivante :**

#### Compiler les tests

```
mvn compiler:testCompile
```

### Comment compiler uniquement les sources ?

**Auteurs :** Eric Reboisson ,

**Dans une console de commandes, accéder au répertoire du projet, et exécuter la commande suivante :**

#### Compiler les sources

```
mvn compiler:compile
```

### Comment générer un JAR des sources ?

**Auteurs :** Eric Reboisson ,

**Dans une console de commandes, accéder au répertoire du projet, et exécuter la commande suivante :**

#### Générer le JAR

```
mvn source:jar
```

Le fichier JAR est généré dans le répertoire */target* du projet.

## Comment déployer un JAR de sources dans le repository ?

Auteurs : Denis Cabasson , Eric Reboisson ,

En modifiant dans le fichier pom.xml :

### Générer un JAR des sources

```
<build>
  <plugins>
    ...
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-source-plugin</artifactId>
      <executions>
        <execution>
          <id>bind-sources</id>
          <goals>
            <goal>jar</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
    ...
  </plugins>
</build>
```

## Comment générer un JAR des sources des tests ?

Auteurs : Eric Reboisson ,

Dans une console de commandes, accéder au répertoire du projet, et exécuter la commande suivante :

### Générer le JAR

```
mvn source:test-jar
```

Le fichier JAR est généré dans le répertoire */target* du projet.

## Comment exécuter uniquement les tests sur un projet ?

Auteurs : Eric Reboisson ,

Dans une console de commandes, accéder au répertoire du projet, et exécuter la commande suivante :

### Exécuter les tests

**Exécuter les tests**

```
mvn test
```

**Comment lister les profils actifs d'un projet ?****Auteurs : Eric Reboisson ,****Dans une console de commandes, accéder au répertoire du projet, et exécuter la commande suivante :****Liste des profils actifs**

```
mvn help:active-profiles
```

**Comment connaître le paramétrage d'un projet ?****Auteurs : Eric Reboisson ,****Dans une console de commandes, accéder au répertoire du projet, et exécuter la commande suivante :****Paramétrage du projet**

```
mvn help:effective-pom
```

**Ou :****Paramétrage du projet**

```
mvn help:effective-settings
```

**Comment vérifier la validité d'un projet Maven ?****Auteurs : Eric Reboisson ,****Dans une console de commandes, accéder au répertoire du projet, et exécuter la commande suivante :****Générer le JAR**

```
mvn validate
```

**Cette commande vérifie que le projet possède toutes les informations nécessaires à son bon fonctionnement avec Maven.****Comment exécuter un script Ant sur une phase ?****Auteurs : Eric Reboisson ,****Ajouter dans le fichier pom.xml les balises suivantes :****Exemple : associer un script Ant à la phase validate**

```
<project>  
...
```

**Exemple : associer un script Ant à la phase validate**

```
<build>
  <plugins>
    <plugin>
      <artifactId>maven-antrun-plugin</artifactId>
      <executions>
        <execution>
          <phase>validate</phase>
          <configuration>
            <tasks>
              <echo file="{basedir}/hello.txt">hello world</echo>
            </tasks>
          </configuration>
          <goals>
            <goal>run</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
...
</project>
```

Dans une console de commandes, accéder au répertoire du projet, et exécuter la commande suivante :

```
mvn validate
```

Le script Ant redirigeant "hello world" dans un fichier hello.txt à la racine du projet, a été traité en même temps que la phase validate.

Voir aussi :  [About Maven Ant Plugin](#)

**Comment créer une release d'un projet ?**

**Auteurs : Jibee , Eric Reboisson ,**

Une release se déroule en deux étapes :

- 1 ***release:prepare* : exécutée une seule fois par release, cette tâche effectue toutes les manipulations nécessaires au sein du projet et du gestionnaire de sources (SCM) aboutissant à une version taggée dans le SCM.**
- 2 ***release:perform* : exécutée autant de fois que nécessaire, cette tâche permet une reconstruction à partir de la version précédemment taggée (par le *release:prepare*) et aboutit à un déploiement de l'artefact sur le référentiel distant (remote repository)**

**Pour une *release:prepare*, dans une console de commandes, accéder au répertoire du projet, et exécuter la commande suivante :**

**La commande *release:prepare* inclut plusieurs saisies utilisateurs avec à chaque fois une valeur par défaut.**

```
mvn release:prepare
```

**Pour que la commande *release:prepare* utilise les valeurs par défaut implicitement, la commande est à lancer en mode batch :**

```
mvn --batch-mode release:prepare
```

Pour une *release:perform*, dans une console de commandes, accéder au répertoire du projet, et exécuter la commande suivante :

```
mvn release:perform
```

Voir aussi :  [About Maven Release plugin](#)

### Comment modifier une des valeurs par défaut de la commande release:prepare ?

Auteurs : [Jibee](#) , [Eric Reboisson](#) ,

La commande *release:prepare* se base sur un fichier *release.properties* créé par la commande elle-même à la racine du projet.

Il est possible de créer soit même et par avance le fichier *release.properties* en renseignant les valeurs par défaut.

Par exemple, pour définir le tag utilisé dans le SCM pour la release :

```
scm.tag=MonReleaseTag
```

### Comment lancer une commande sur une version taggée ?

Auteurs : [Jibee](#) , [Eric Reboisson](#) ,

Il est possible par exemple de déployer sur le référentiel une ancienne version du projet :

#### Déployer un tag précédent du projet

```
mvn scm:bootstrap -Dtag=[TAG_SCM] -Dgoals=deploy
```

[TAG\_SCM] étant le libellé du tag dans le gestionnaire de sources correspondant à la release sur laquelle on souhaite travailler.

### Comment utiliser des propriétés du settings.xml dans le pom.xml ?

Auteurs : [Eric Reboisson](#) ,

Il est possible de paramétrer dans le fichier *settings.xml* des propriétés utilisables dans le *pom.xml*.

Il faut définir premièrement la propriété dans le fichier *settings.xml* :

#### Configuration dans settings.xml

```
<settings>
  ...
  <profiles>
    <profile>
      <id>inject-application-home</id>
      <properties>
```

### Configuration dans settings.xml

```
<application-home>/path/to/application</application-home>
</properties>
</profile>
</profiles>

<activeProfiles>
  <activeProfile>inject-application-home</activeProfile>
</activeProfiles>
</settings>
```

Et ensuite l'utiliser dans le pom.xml, un exemple avec la génération du rapport des tags :

```
</project>
...
<reporting>
  <plugins>

    <plugin>
      <groupId>org.codehaus.mojo</groupId>
      <artifactId>taglist-maven-plugin</artifactId>
      <version>2.0</version>

      <configuration>
        <outputDirectory>${application-home}</outputDirectory>
      </configuration>
    </plugin>

  </plugins>

</reporting>
...
</project>
```

### Comment accéder aux variables d'environnement dans le pom.xml ?

Auteurs : [Eric Reboisson](#) ,

Il suffit d'utiliser la notation `${env.X}` dans le fichier pom.xml, où X doit être remplacé par le nom d'une variable d'environnement.

Par exemple, pour accéder au PATH sous Windows ce sera `${env.PATH}` que vous utiliserez dans le pom.xml.

### Comment désactiver les tests dans une phase d'installation ?

Auteurs : [Eric Reboisson](#) ,

En ajoutant l'option suivante à la ligne de commande :

#### Désactivation des tests en ligne de commande

```
mvn -Dmaven.test.skip=true install
```

Ou en désactivant la réalisation des tests dans le pom.xml :

#### Désactivation des tests dans le pom.xml

```
<plugin>
```

**Désactivation des tests dans le pom.xml**

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

  <configuration>
    <skip>true</skip>
  </configuration>

</plugin>
```

**Comment désinstaller un artefact du repository local ?****Auteurs : Eric Reboisson ,**

**Il n'y a pas de commande pour effectuer cette opération, il suffit de supprimer le répertoire de l'artefact dans le repository.**

Sommaire > Utilisation > Gestion des dépendances

## Où se trouvent les dépendances ?

Auteurs : Denis Cabasson , Eric Reboisson ,

Maven recherche les dépendances dans le repository dit central, présent à l'adresse : <http://www.ibiblio.org/maven2>  
Sur ce serveur, dans un sous-répertoire correspondant au groupId (les '.' permettent de séparer chacun des sous-répertoires) puis à l'artifactId, puis enfin à la version, se trouve le fichier POM de la dépendance ainsi que le fichier jar (ou autre selon le type d'artefact).

## Comment ajouter une dépendance ?

Auteurs : Denis Cabasson , Eric Reboisson ,

Pour ajouter une dépendance à un projet, il suffit de la déclarer dans le pom.xml.  
Maven s'occupera de la récupérer (si il s'agit d'un projet ayant une licence compatible avec la redistribution) :

### Configurer l'ajout d'une dépendance

```
<project>
...
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
...
</project>
```

Une dépendance est décrite par les informations groupId/artifactId/version/type qui permettent de l'identifier de façon unique.

Si l'artefact est un jar, il n'est pas nécessaire de spécifier son type.

## Comment changer la version d'une dépendance ?

Auteurs : Denis Cabasson , Eric Reboisson ,

On doit préciser à Maven la version de la dépendance que l'on souhaite utiliser :

### Changer la version d'une dépendance

```
<project>
...
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
...
</project>
```

### Changer la version d'une dépendance

`</project>`

La version est obligatoire, sauf si on indique une dépendance déjà déclarée dans un POM parent au sein d'un tag `<dependencyManagement>`.

On peut préciser un numéro de version comme dans l'exemple ci-dessus.

On peut également donner une plage de versions à Maven (dans le tableau suivant x désigne la version choisie par Maven) :

Syntaxe	Signification
<code>(,1.0]</code>	$x \leq 1.0$
<code>[1.2,1.3]</code>	$1.2 \leq x \leq 1.3$
<code>[1.0,2.0)</code>	$1.0 \leq x < 2.0$
<code>[1.5,)</code>	$x \geq 1.5$
<code>(,1.0],[1.2,)</code>	$x \leq 1.0$ or $x \geq 1.2$ . Plusieurs ensembles peuvent être déclarés, séparés par des ','
<code>(,1.1),(1.1)</code>	Une version autre que la 1.1.

### Qu'est-ce que le scope d'une dépendance ?

Auteurs : Denis Cabasson , Eric Reboisson ,

Le scope (la portée) d'une dépendance est déclaré par la balise `scope` dans la déclaration d'une dépendance :

#### Un exemple de dépendance avec le scope test

```
<project>
...
  <dependencies>

    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>

  </dependencies>
...
</project>
```

La dépendance ci-dessus a le scope `test`, ce qui signifie qu'elle doit être utilisée pour compiler les tests et les exécuter. Les scopes possibles sont :

- **compile** : la dépendance est nécessaire pour toutes les phases du build
- **test** : la dépendance est nécessaire pour la compilation/l'exécution des tests
- **runtime** : la dépendance est nécessaire à l'exécution, mais pas à la compilation
- **provided** : la dépendance est nécessaire à la compilation, mais ne sera pas packagée avec le projet. Typiquement, un container lui fournira cette dépendance (par exemple `servlet-api`).

Voir aussi :  [Introduction to the Dependency Mechanism](#)

## Comment utiliser une dépendance qui n'est pas dans le repository central ?

Auteurs : [Denis Cabasson](#) , [Eric Reboisson](#) ,

Certaines dépendances, de part leur licence, ne peuvent pas être distribuées via le repository central de Maven. Il faut alors trouver le jar correspondant à cette dépendance et le charger dans le repository local en utilisant la commande :

### Chargement d'un jar dans un repository

```
mvn install:install-file -Dfile=<chemin-du-fichier> -DgroupId=<group-id> \
-DartifactId=<artifact-id> -Dversion=<version> -Dpackaging=<packaging> -DgeneratePom=true
```

Un POM basique sera installé dans le repository local. Si vous connaissez les dépendances de cet artefact, vous devez éditer ce POM et y ajouter les dépendances, ainsi vous bénéficierez du mécanisme de dépendances transitives en utilisant cet artefact.

Le packaging est généralement un fichier jar.

Si le jar a une licence compatible avec la redistribution, mais n'est pas présent dans le repository, il s'agit d'une erreur.

Vous pouvez donc ouvrir un dossier dans  <http://jira.codehaus.org/browse/MEV> (après avoir vérifié que c'est bien une erreur, voir aussi :  [Guide to uploading artifacts to Ibiblio](#)).

Voir aussi :

 [Guide to installing 3rd party JARs](#)

Sommaire > Utilisation > Site

## Qu'est que le format APT ?

Auteurs : **Eric Reboisson** ,

APT est l'acronyme pour "Almost Plain Text", c'est un format d'écriture indiquant la mise en forme du texte à l'aide d'une syntaxe wiki.

Voir aussi :  [The APT format](#)

## Comment créer les descripteurs de génération du site ?

Auteurs : **Eric Reboisson** ,

**Positionnez vous dans le répertoire parent du projet.**

**Dans une console de commandes, exécuter la commande suivante :**

### Les fichiers XML descripteurs du site

```
mvn archetype:create -DgroupId=MonGroupe -DartifactId=MonArtefact -  
DarchetypeGroupId=org.apache.maven.archetypes -DarchetypeArtifactId=maven-archetype-site
```

Sommaire > Utilisation > Documentation

## Comment générer le site d'un projet ?

Auteurs : [Eric Reboisson](#) ,

Dans une console de commandes, accéder au répertoire du projet, et exécuter la commande suivante :

### Création du site d'un projet

```
mvn site
```

Le répertoire `/target/site` situé dans votre projet contient alors le site.

## Comment générer la javadoc pour un projet ?

Auteurs : [Eric Reboisson](#) ,

Ajouter dans le fichier pom.xml les balises suivantes :

### Configuration de la javadoc

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

    <plugin>

      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-javadoc-plugin</artifactId>

      <configuration>
        <minmemory>128m</minmemory>
        <maxmemory>512m</maxmemory>
        ...
      </configuration>
    </plugin>

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

Exécuter ensuite la commande suivante qui créera le site du projet avec le rapport javadoc :

```
mvn site
```

Voir aussi :  [About Maven Javadoc Plugin](#)

## Comment vérifier la qualité du code avec checkstyle ?

Auteurs : [Eric Reboisson](#) ,

Ajouter dans le fichier pom.xml les balises suivantes :

### Configuration de checkstyle

```
<project>
...
  <reporting>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-checkstyle-plugin</artifactId>
      </plugin>
    </plugins>
  </reporting>
...
</project>
```

Exécuter ensuite la commande suivante qui créera le site du projet avec le rapport checkstyle :

```
mvn site
```

Voir aussi :  [About Maven Checkstyle Plugin](#)

### Comment vérifier la qualité du code avec PMD ?

Auteurs : [Eric Reboisson](#) ,

Ajouter dans le fichier pom.xml les balises suivantes :

### Configuration de checkstyle

```
<project>
...
  <reporting>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-pmd-plugin</artifactId>
      </plugin>
    </plugins>
  </reporting>
...
</project>
```

Exécuter ensuite la commande suivante qui créera le site du projet avec le rapport PMD :

```
mvn site
```

Voir aussi :

 [PMD : un outil pour l'audit de code](#)



## About Maven PMD Plugin

## Comment connaître l'activité d'un projet ?

Auteurs : [Eric Reboisson](#) ,

Ajouter au fichier pom.xml les balises suivantes :

### Configuration du rapport d'activité

```
<project>
...
  <reporting>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-changelog-plugin</artifactId>
      </plugin>
    </plugins>
  </reporting>
...
  <scm>
    <connection>scm:svn:http://chemin_du_projet</connection>
    <developerConnection>scm:svn:https://chemin_du_projet</developerConnection>
    <url>scm:svn:http://chemin_du_projet</url>
  </scm>
...
</project>
```

Le paramétrage du SCM est nécessaire car les rapports d'activité se basent dessus.

Dans une console de commandes, accéder au répertoire du projet, et exécuter la commande suivante :

### Générer les rapports d'activité

```
mvn site
```

Le répertoire `/target/site` situé dans votre projet contient maintenant trois rapports d'activité :

- **changelog** : rapport indiquant toutes les activités sur le SCM.
- **dev-activity** : rapport indiquant par développeur le nombre de commits, de fichiers modifiés.
- **file-activity** : rapport indiquant les fichiers qui ont été révisés.

## Comment générer un rapport croisé des sources ?

Auteurs : [Eric Reboisson](#) ,

Un rapport croisé des sources permet de naviguer dans les sources du projet à partir du site généré par Maven.

Ajouter au fichier pom.xml les balises suivantes :

### Configuration du rapport d'activité

```
<project>
...
  <build>
    ...
  </build>
```

### Configuration du rapport d'activité

```
<reporting>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-jxr-plugin</artifactId>
    </plugin>
  </plugins>
</reporting>
...
</project>
```

Dans une console de commandes, accéder au répertoire du projet, et exécuter la commande suivante :

### Génération du site

```
mvn site
```

## Comment analyser les métriques avec JDepend ?

Auteurs : [Eric Reboisson](#) ,

Ajouter dans le fichier pom.xml les balises suivantes :

### Configuration du rapport des métriques

```
<project>
  ...
  <reporting>
    <plugins>
      <plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>jdepend-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </reporting>
  ...
</project>
```

Dans une console de commandes, accéder au répertoire du projet, et exécuter la commande suivante :

```
mvn site
```

Voir aussi :



[About Maven JDepend Plugin](#)



[Analyse de la qualité du code Java avec JDepend 2.7](#)

## Comment générer un rapport des tags ?

Auteurs : [Eric Reboisson](#) ,

Ajouter dans le fichier pom.xml les balises suivantes :

### Configuration du rapport des tags pour les TODO et @todo

```
<project>
  ...
  <reporting>
    <plugins>
      <plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>taglist-maven-plugin</artifactId>
        <version>2.0-beta-1</version>

        <configuration>
          <tags>TODO, @todo</tags>
        </configuration>
      </plugin>
    </plugins>
  </reporting>
  ...
</project>
```

Dans une console de commandes, accéder au répertoire du projet, et exécuter la commande suivante :

```
mvn site
```

Voir aussi :  [Maven 2 Taglist Plugin](#)

### Comment générer un rapport de couverture des tests ?

Auteurs : Eric Reboisson ,

Nous allons utiliser pour ce faire plugin Cobertura.

Un outil d'analyse de couverture des tests permet de savoir quelles portions de code n'ont pas été testée, et ainsi agir en fonction.

Ajouter dans le fichier pom.xml les balises suivantes :

### Configuration du rapport des tags pour les TODO et @todo

```
<project>
  ...
  <build>
    ...
    <plugins>
      ...
      <plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>cobertura-maven-plugin</artifactId>
        <executions>
          <execution>
            <goals>
              <goal>clean</goal>
            </goals>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>

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

#### Configuration du rapport des tags pour les TODO et @todo

```
<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>cobertura-maven-plugin</artifactId>
</plugin>
</plugins>
</reporting>
</project>
```

Dans une console de commandes, accéder au répertoire du projet, et exécuter la commande suivante :

```
mvn site
```

Voir aussi :  [Maven 2 Cobertura Plugin](#)

[Sommaire](#) > [Les projets multimodules](#)

## Comment déclarer un projet multimodules ?

**Auteurs :** [Denis Cabasson](#) , [Eric Reboisson](#) ,

Voilà l'exemple d'une arborescence de trois modules, avec un projet parent :

### Arborescence à trois modules

```
parent
|-- module1
| |-- pom.xml
|-- module2
| |-- pom.xml
|-- module3
| |-- pom.xml
|-- pom.xml
```

Le fichier pom.xml du parent déclare les modules enfants et doit avoir comme packaging pom :

### Déclaration des trois modules

```
<project>
...
<modules>
  <module>module1</module>
  <module>module2</module>
  <module>module3</module>
</modules>
...
</project>
```

Ainsi lorsqu'une action sera lancée sur le projet parent, elle le sera également sur les projets enfants. Les enfants doivent eux-aussi déclarer le parent dans leur pom.xml :

### Déclaration du parent

```
<project>

  <parent>
    <groupId>com.organisation</groupId>
    <artifactId>parent</artifactId>
    <version>1</version>
  </parent>

</project>
```

De cette façon, toute la configuration déclarée dans le parent sera utilisée dans le module enfant par héritage. Il est vivement conseillé de nommer les modules de la même façon que les répertoires les hébergeant, ainsi que l'artifactId.

## Qu'est-ce que le <dependencyManagement> ?

**Auteurs :** [Denis Cabasson](#) , [Eric Reboisson](#) ,

La balise dependencyManagement, dans le POM du projet parent permet de déclarer une version et un scope "préférée" pour une dépendance.

Si un des modules enfants utilise cette dépendance, sans préciser de version ou de scope, celle déclarée dans le `dependencyManagement` sera utilisée.

Exemple :

#### Gestion des dépendances avec le `dependencyManagement`

```
<project>
..
  <dependencyManagement>

    <dependencies>

      <dependency>
        <groupId>commons-lang</groupId>
        <artifactId>commons-lang</artifactId>
        <version>2.1</version>
        <scope>compile</scope>
      </dependency>

    </dependencies>

  </dependencyManagement>
..
</project>
```

De cette façon, si un module enfant utilise `commons-lang` :

#### Utilisation de la dépendance du parent

```
<project>
...
  <dependencies>

    <dependency>
      <groupId>commons-lang</groupId>
      <artifactId>commons-lang</artifactId>
    </dependency>

  </dependencies>
...
</project>
```

La version choisie sera la 2.1 avec le scope `compile`.

Ce mécanisme permet de monter très facilement une version de dépendance dans tous les modules enfants.

[Sommaire](#) > Développement de plugins

## Comment créer un plugin Java ?

Auteurs : [Jibee](#) , [Eric Reboisson](#) ,**Dans une console de commandes, exécuter la commande suivante :**

## Création d'un squelette de plugin

```
mvn archetype:create -DgroupId=exempleplugin -DartifactId=exemple-plugin -
DarchetypeGroupId=org.apache.maven.archetypes
-DarchetypeArtifactId=maven-archetype-mojo -DarchetypeVersion=1.0-alpha-4
```

Vous trouverez une classe  Java nommée `/exemple-plugin/src/main/java/exempleplugin/MyMojo.java` contenant le code du plugin.

Ce modèle de plugin généré par l'archetype a créé un goal nommé `touch`.

Pour utiliser ce plugin, il faut maintenant l'installer.

Dans une console de commandes, accéder au répertoire du projet de plugin, et exécuter la commande suivante :

## Installation du plugin

```
mvn install
```

Puis, exécuter la commande suivante :

## Exécution du goal touch

```
mvn exempleplugin:exemple-plugin:touch
```

Vous verrez alors un répertoire `target` créé dans le répertoire projet et un fichier `touch.txt` à l'intérieur.

## Comment définir un paramètre au plugin ?

Auteurs : [Jibee](#) , [Eric Reboisson](#) ,**En ajoutant une variable de classe dans le fichier  Java du plugin comme ceci :**

```
/**
 * Mon paramètre.
 * @parameter expression="${mymavenparameter}"
 * @required
 */
private String myJavaParameter;
```

Nous venons de créer une variable `myJavaParameter`, de type `String`, obligatoire (`@required`) et correspondant au paramètre `mymavenparameter` du `pom.xml` du projet exécutant le plugin :

## Utilisation du paramètre

```
<project>
...
  <build>
    <plugins>
      <plugin>
```

### Utilisation du paramètre

```
<groupId>exempleplugin</groupId>
<artifactId>exemple-plugin</artifactId>
<configuration>
  <mymavenparameter>valeur</mymavenparameter>
</configuration>
</plugin>
</plugins>
</build>
...
</project>
```

### Comment récupérer les propriétés du projet exécutant le plugin ?

Auteurs : Jibee , Eric Reboisson ,

En ajoutant la variable de classe suivante (avec l'annotation) :

#### Récupération des propriétés de projet

```
/**
 * Projet en cours de deploiement.
 * @parameter expression="${project}"
 */
private org.apache.maven.project.MavenProject project;
```

Vous disposez maintenant d'une variable `project` qui sera automatiquement initialisée par Maven à l'exécution. Vous avez maintenant accès aux informations telles que le `groupId`, l'`artifactId`, la version du projet, etc.

### Comment ajouter une saisie utilisateur pendant l'exécution d'un plugin ?

Auteurs : Jibee , Eric Reboisson ,

Pour cela, il faut utiliser l'[API](#) `plexus-interactivity-api`. Ajoutez la dépendance dans votre `pom.xml` :

#### Ajout de plexus-interactivity-api

```
<dependency>
  <groupId>org.codehaus.plexus</groupId>
  <artifactId>plexus-interactivity-api</artifactId>
  <version>1.0-alpha-4</version>
</dependency>
```

Définir la variable de classe suivante dans votre classe Mojo :

#### Récupération des propriétés de projet

```
/**
 * inputHandler
 * @component
 */
private InputHandler inputHandler;
```

Et ajouter le code suivant dans le code de votre plugin pour demander une saisie utilisateur : `String inputValue = inputHandler.readLine();`

## Comment mapper un goal sur une phase du cycle de vie de Maven ?

Auteurs : Jibee , Eric Reboisson ,

Pour mapper un goal sur la commande `mvn package` par exemple, il faut ajouter l'annotation suivante dans la classe  Java du goal en question :

### Récupération des propriétés de projet

```
/**
 * Mon goal
 *
 * @goal touch
 *
 * @phase package
 */
```

Pour activer cette configuration, il faut ajouter dans le `pom.xml` du projet exécutant le plugin une balise `execution` :

### Association du goal

```
<build>
  <executions>
    <execution>
      <id>goal-execution-id</id>
      <goals>
        <goal>touch</goal>
      </goals>
    </execution>
  </executions>
</build>
```

Avec cette configuration, le goal `touch` s'exécutera à chaque appel de la commande `mvn package`, en complément de celle-ci.

## Comment voir les sources d'un plugin ?

Auteurs : Eric Reboisson ,

Les sources des plugins Maven sont accessibles, ce qui permet à la fois de voir comment est réalisé un plugin, mais également de proposer votre participation.

Prenons pour exemple le plugin `clean`, dont la page principale est <http://maven.apache.org/plugins/maven-clean-plugin>.

Les sources du plugin `clean` sont visibles dans un navigateur via  ViewVC à l'adresse suivante <http://svn.apache.org/viewvc/maven/plugins/trunk/maven-clean-plugin>.

On peut également créer une copie locale du plugin `clean` en utilisant  Subversion avec la commande suivante :

### Checkout du projet du plugin clean

**Checkout du projet du plugin clean**

```
svn checkout http://svn.apache.org/repos/asf/maven/plugins/trunk/maven-clean-plugin maven-clean-plugin
```

**lien : [FAQ](#) [FAQ SCM : Subversion](#)**

Sommaire > Continuum, serveur d'intégration continue


Sommaire > Continuum, serveur d'intégration continue > Documentation

## Qu'est ce que Continuum ?

Auteurs : **Eric Reboisson** ,

Continuum est un serveur d'intégration continue principalement pour les projets  **Java**.

Le site officiel de Continuum est  <http://maven.apache.org/continuum/>

Continuum permet dont d'effectuer l'intégration d'un projet  **Java** (compilation, tests, etc.) à des intervalles paramétrés.

Cette pratique s'accorde avec les techniques de développement "agiles" et Continuum est entièrement compatible avec les technologies d'intégration suivantes :

- **Maven 1**
- **Maven 2**
- **Ant**
- **Shell scripts**

## Comment signaler un bug ou soumettre une évolution dans Continuum ?

Auteurs : **Eric Reboisson** ,

Il suffit pour cela de poster votre requête dans le **JIRA** (un outil de gestion de demandes) de Continuum à l'adresse

 <http://jira.codehaus.org/browse/CONTINUUM>




*Attention, vous devez être inscrit pour pouvoir poster sur ce site.*

Sommaire > Continuum, serveur d'intégration continue > Installation et configuration

### Comment installer Continuum ?

Auteurs : [Eric Reboisson](#) ,

Télécharger l'archive `continuum-x.x.x-bin.zip` de Continuum sur  <http://maven.apache.org/continuum>  
Décompresser l'archive et copier le répertoire `continuum-x.x.x` à l'endroit de votre choix.  
Voilà c'est terminé !!!

### Comment lancer Continuum comme un service ?

Auteurs : [Eric Reboisson](#) ,

Pour installer Continuum comme un service sous Windows, il faut premièrement démarrer le serveur Continuum.  
Puis exécuter le fichier suivant :

#### Installation de Continuum comme service

```
%CONTINUUM_HOME%/bin/win32/InstallService.bat
```

Un service nommé `continuum` est alors créé.  
Il faut ensuite assigner un propriétaire de ce service, qui est par défaut `System`, à un utilisateur réel de la machine.  
Ainsi Continuum pourra retrouver facilement les fichiers de configuration tel que le fichier `settings.xml` de Maven dans ``${user.home}`/m2`.

### Comment désinstaller le service Continuum ?

Auteurs : [Eric Reboisson](#) ,

Exécuter le fichier suivant :

#### Désinstallation du service Continuum

```
%CONTINUUM_HOME%/bin/win32/UninstallService.bat
```

Le service nommé `continuum` est alors supprimé.

### Comment modifier les définitions de build d'un projet ?

Auteurs : [Emmanuel Venisse](#) , [Eric Reboisson](#) ,

Cliquer sur '*Show Projects*', puis cliquer sur le nom du projet.  
Dans la page qui apparaît, vous avez la possibilité d'éditer les définitions de build ou d'en ajouter de nouvelles.

### Comment configurer la notification par mail ?

Auteurs : [Eric Reboisson](#) ,

Il faut premièrement créer une notification dans le `pom.xml` :

### Ajout d'une notification par mail au pom.xml

```
<ciManagement>

  <system>continuum</system>

  <notifiers>
    <notifier>
      <type>mail</type>

      <configuration>
        <address>prenom.nom@email.com</address>
      </configuration>

    </notifier>
  </notifiers>
</ciManagement>
```

Editer le fichier `%CONTINUUM_HOME%/apps/continuum/conf/application.xml`, et modifier les informations suivantes :

### Configuration de la notification par mail

```
<component>

  <role>org.codehaus.plexus.mailsender.MailSender</role>
  <implementation>org.codehaus.plexus.mailsender.javamail.JavamailMailSender</implementation>

  <configuration>
    <smtp-host>adresse_smtp</smtp-host>
    <smtp-port>25</smtp-port>
    <sslProvider>com.sun.net.ssl.internal.ssl.Provider</sslProvider>
    <username>login</username>
    <password>password</password>
    <sslMode>false</sslMode>
  </configuration>

</component>
```

On peut également paramétrer les balises suivantes pour indiquer le nom et l'email de l'expéditeur :

### Configuration de la notification par mail

```
<component>
...
  <configuration>
    <from-mailbox>prenom.nom@email.com</from-mailbox>
    <from-name>Nom</from-name>
  </configuration>
...
</component>
```

Il est nécessaire de redémarrer Continuum après une modification dans le fichier `application.xml`.

## Comment configurer la notification par IRC ?

Auteurs : Eric Reboisson ,

Pour notifier sur un canal IRC le résultat d'un build il faut créer un notifier dans le pom.xml.

La notification paramétrée ci-dessous enverra le message sur le canal IRC `#test` du serveur `irc.codehaus.org`.

### Ajout d'une notification par IRC au pom.xml

```
<ciManagement>
  <system>continuum</system>

  <notifiers>
    <notifier>
      <type>irc</type>

      <configuration>
        <host>irc.masociete.org</host>
        <port>6667</port>
        <channel>#test</channel>
      </configuration>

    </notifier>
  </notifiers>
</ciManagement>
```

### Comment configurer la notification par Google Talk ?

Auteurs : Eric Reboisson ,

Ouvrir la liste des projets dans l'interface web de Continuum.

Cliquer sur le lien du nom d'un projet.

Au niveau de la gestion des *Notifiers* cliquer sur le bouton *Add* puis choisir *Jabber* dans la liste et cliquer sur *Next*.

Saisir les informations suivantes puis cliquer sur *Submit* :

- Jabber Host : talk.google.com
- Jabber Port : 5222
- Jabber Login : votre compte Google Talk sans '@gmail.com'
- Jabber Domain Name : gmail.com
- Is it a SSL Connection? : laisser décocher si votre connexion n'est pas SSL

### Comment changer le port d'écoute HTTP de Continuum ?

Auteurs : Eric Reboisson ,

Dans le fichier `${CONTINUUM_HOME}/apps/continuum/conf/application.xml` , modifier la valeur du port de *http-listener*, par exemple pour écouter sur le port 8066 le paramétrage est :

#### Modifier le port d'écoute HTTP

```
<application>
...
<services>
  <service>
    <id>jetty</id>
    <configuration>
      <webapps>
        <webapp>
          <file>${plexus.home}/lib/continuum-web-1.0.3.jar</file>
          <context>/continuum</context>
          <extraction-path>${plexus.home}/webapp</extraction-path>
          <listeners>
            <http-listener>
              <port>8066</port>
            </http-listener>
          </listeners>
        </webapp>
      </webapps>
    </configuration>
  </service>
</services>
```

**Modifier le port d'écoute HTTP**

```
    </listeners>
  </webapp>
</webapps>
</configuration>
</service>

...
</application>
```

Sommaire > Continuum, serveur d'intégration continue > Utilisation

### Comment démarrer et se connecter au serveur Continuum ?

Auteurs : **Eric Reboisson** ,

Exécuter le fichier suivant pour démarrer le serveur :

#### Lancement de Continuum sous Windows

```
%CONTINUUM_HOME%/bin/win32/run.bat
```

Dans un navigateur, saisir l'adresse <http://localhost:8080/continuum> et paramétrer les informations pour l'administrateur puis valider.


Pour se connecter à Continuum, toujours à l'adresse <http://localhost:8080/continuum> cliquer sur le lien login et saisir un nom d'utilisateur et un mot de passe (ceux de l'administrateur par exemple) et valider.

### Comment ajouter un projet Maven au serveur Continuum ?

Auteurs : **Eric Reboisson** ,

Dans le menu de gauche, cliquer sur *Add Project > Maven 2.0+ Project* puis sélectionner le fichier pom.xml du projet que vous voulez intégrer et valider.

Le projet est alors ajouté à la liste.

 *Pour qu'un projet puisse être ajouté, la configuration d'un gestionnaire de sources est nécessaire.*

*Ci-dessous, une configuration locale à ajouter au pom.xml du projet :*

#### Configuration locale d'un SCM

```
<project>
...
  <scm>
    <connection>scm:local|[chemin du répertoire des projets]|[Nom du projet]</connection>
  </scm>
...
</project>
```

lien :  [Maven SCM](#)

### Comment lancer l'intégration d'un projet ?

Auteurs : **Eric Reboisson** ,

Dans le menu de gauche, cliquer sur *Show Projects* puis cliquer sur le lien *Build Now* à droite du projet.

Si l'intégration s'est bien déroulée un triangle vert apparaît à gauche du nom du projet.

### Comment modifier l'intervalle entre deux builds ?

Auteurs : **Emmanuel Venisse** , **Eric Reboisson** ,

Cliquer sur le lien '*Schedules*' dans le menu à gauche et modifier/ajouter les schedulers.

**Ensuite, éditer les définitions de build de vos projets en leur assignant un de vos schedulers.**

Sommaire > Proxies d'entreprise

Sommaire > Proxies d'entreprise > Documentation






## Qu'est ce qu'un proxy ?

Auteurs : **Eric Reboisson** ,

Un proxy permet d'accéder à des ressources, d'utiliser les ressources disponibles en cache ou de les télécharger si ce n'est pas le cas.

## Quels sont les proxies d'entreprise existants ?

Auteurs : **Eric Reboisson** ,

Nom	Lien
Archiva	 <a href="http://maven.apache.org/archiva">http://maven.apache.org/archiva</a>
Proximity	 <a href="http://proximity.abstracthorizon.org">http://proximity.abstracthorizon.org</a>
Artifactory	 <a href="http://www.jfrog.org/sites/artifactory/latest/">http://www.jfrog.org/sites/artifactory/latest/</a>
Dead Simple Maven Proxy	 <a href="http://www.pdark.de/dsmp">http://www.pdark.de/dsmp</a>
Standard Maven Proxy	 <a href="http://maven-proxy.codehaus.org">http://maven-proxy.codehaus.org</a>

lien :  [Quel proxy d'entreprise Maven 2 utilisez-vous ?](#)

Sommaire > Proxies d'entreprise > Archiva

## Archiva, c'est quoi ?

Auteurs : [Eric Reboisson](#) ,

Archiva est une application tierce permettant la gestion de repositories.

Les utilisations sont multiples :


- maintenance de vos propres repositories
- navigation, recherche d'artefacts
- sécurité

Pour information, Archiva est à ce jour en version bêta au sein de la communauté Apache.

lien :  <http://maven.apache.org/archiva/>

## Comment installer Archiva ?

Auteurs : [Eric Reboisson](#) ,

Télécharger l'archive `apache-archiva-x.x-x-x-bin.tar.gz` d'Archiva sur  <http://maven.apache.org/archiva/download.html>

Décompresser l'archive et copier le répertoire `apache-archiva-x.x-x-x` à l'endroit de votre choix.

Voilà c'est terminé !!!

lien :  <http://maven.apache.org/archiva/>

## Comment exécuter Archiva ?

Auteurs : [Eric Reboisson](#) ,

Il suffit d'exécuter le fichier de commandes `%ARCHIVA_HOME%/bin/votreplateforme/run.extension`.

Par exemple sur Windows :

```
%ARCHIVA_HOME%/bin/windows-x86-32/run.bat
```

Une fois le démarrage d'Archiva terminé, vous pouvez y accéder en tapant l'adresse <http://localhost:8080/archiva> dans un navigateur.

Vous devrez alors, à votre premier accès, créer un utilisateur administrateur.