

PREMIER PROJET MAVEN DANS ECLIPSE

Eclipse & Maven

Cours & Tutoriaux

[Retour au blog Java le soir](#)

Maven est un projet Open source, porté par la fondation Apache. La page officielle de Maven se trouve ici : <http://maven.apache.org/>.

Le développement de Maven est aujourd'hui assez largement financé par la société Sonatype (<http://www.sonatype.com/>). On peut télécharger une documentation assez complète sur leur site, à la page <https://www.sonatype.com/ebooks>.

Ce tutorial a pour objet de configurer un premier projet Maven dans Eclipse, de comprendre comment le configurer et comment utiliser le gestionnaire de dépendances.

Table des matières

1. Création d'un projet Maven dans Eclipse
2. Configuration d'un projet Maven dans Eclipse
3. Ajout d'une dépendance à notre projet
4. Choix d'une installation de Maven dans Eclipse

1. CRÉATION D'UN PROJET MAVEN DANS ECLIPSE

On part pour cela d'un projet Java vide, sous Eclipse. Pour créer un tel projet, on pourra se référer aux tutoriaux [Premiers pas avec Eclipse](#) et [Premiers pas avec son projet dans Eclipse](#), sur Java le soir.

La création d'un nouveau projet Maven se fait de façon classique, comme tout autre projet.

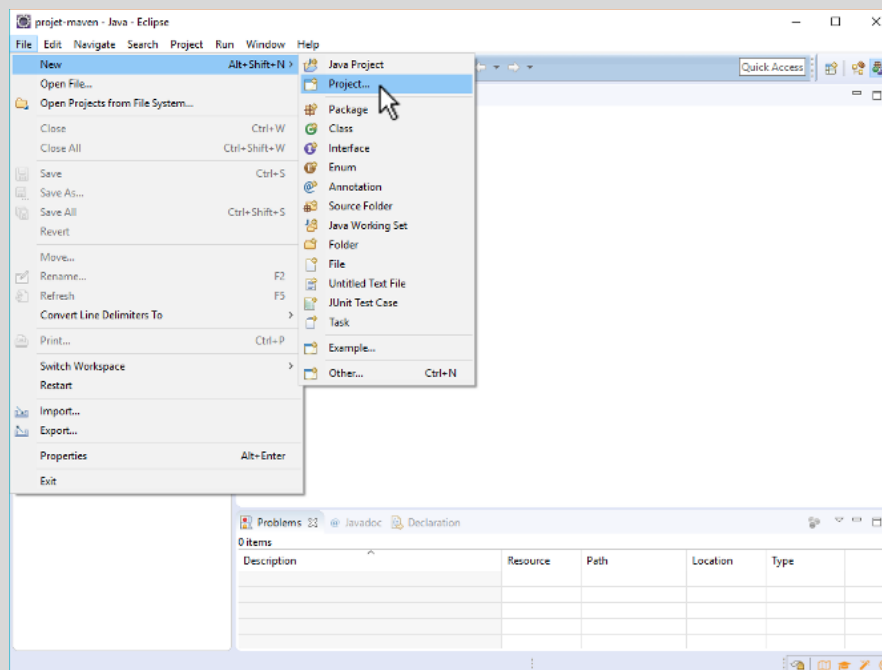


Figure 1. Création d'un nouveau projet dans Eclipse

Le panneau de création de projet s'ouvre alors. Il existe de nombreuses options dans ce panneau, on s'intéresse ici à l'option Maven, que l'on peut filtrer en tapant Maven dans la barre de filtrage.

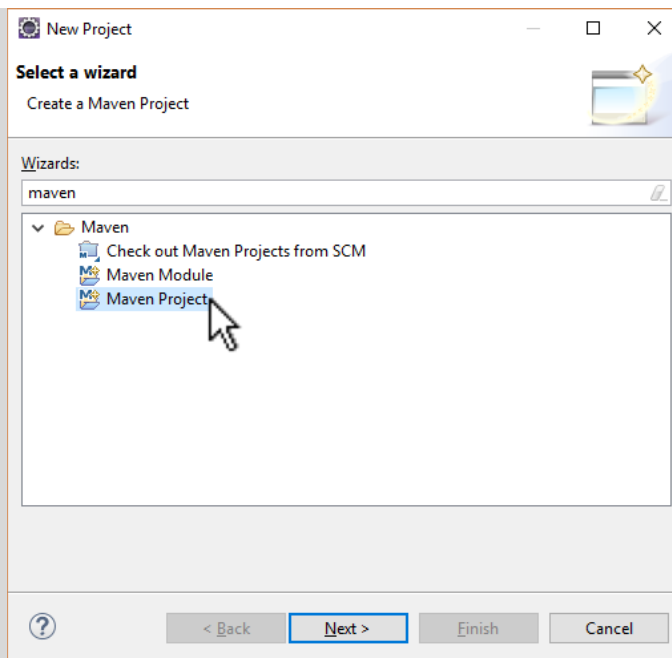


Figure 2. Panneau de création de projet

On peut en fait créer un projet Maven de trois façons.

- La première consiste à récupérer un projet Maven existant en provenance d'un SCM. SCM signifie Source Control Management. Il s'agit donc d'un projet dont le code source existe et est géré par un système du type CVS, SVN, Git ou Mercurial. Le code source peut se trouver sur un disque local, ou sur un serveur distant, tel que Github ou Bitbucket.
- La seconde option permet de créer un module Maven. Un module Maven peut être vu comme un sous-projet d'un projet Maven plus large. Nous n'entrons pas dans plus de détails dans ce document.
- La troisième option consiste en la création d'un projet Maven de base. C'est cette option que nous allons utiliser.

Créer un projet Maven simple requiert tout de même un peu plus d'informations que la création d'un projet Java de base. Le deuxième panneau nous demande essentiellement deux choses. Tout d'abord si nous souhaitons utiliser un archétype Maven. Un archétype Maven est en fait un modèle de projet, utile pour les projets un peu complexe ou nécessitant une configuration fastidieuse à créer. C'est le cas d'applications web, Java EE, Spring, etc... La seconde chose est l'endroit où l'on souhaite enregistrer notre projet. Il n'y a pas de raison pour nous de choisir un autre endroit que l'endroit par défaut.

Nous n'allons pas utiliser d'archétype, on coche donc la case qui permet de sauter la phase de sélection de l'archétype, et on laisse la localisation par défaut pour ce projet.

Eclipse & Maven

Cours & Tutoriaux

[Retour au blog Java le soir](#)

Table des matières

1. Création d'un projet Maven dans Eclipse
2. Configuration d'un projet Maven dans Eclipse
3. Ajout d'une dépendance à notre projet
4. Choix d'une installation de Maven dans Eclipse

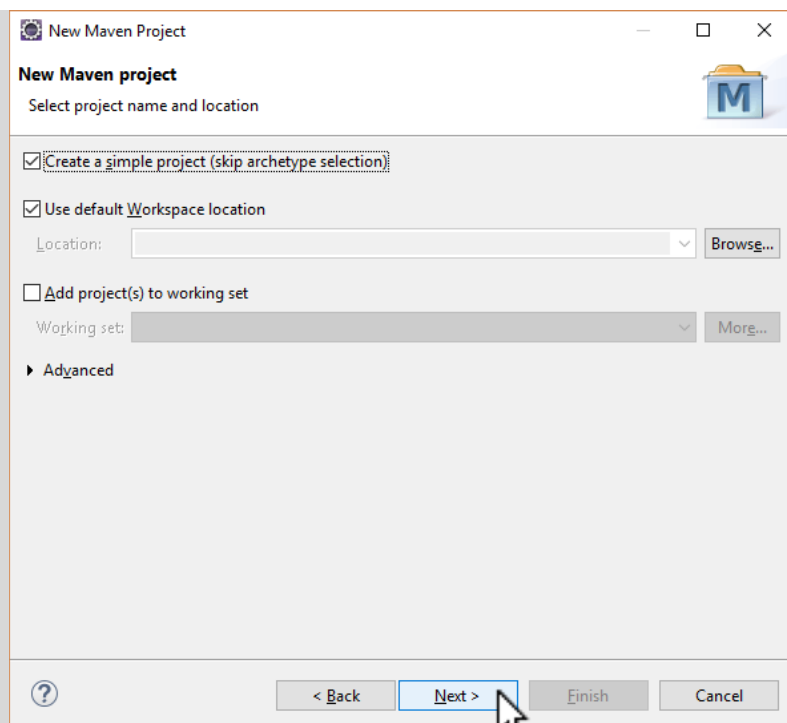


Figure 3. Premier panneau de configuration d'un projet Maven

Une fois cette première étape passée, Maven nous demande des informations très importantes, dont il faut bien comprendre la signification.

Tout d'abord, il faut savoir que Maven reconnaît tous les projets qu'il gère par trois paramètres.

- Le `groupId`, qui constitue la racine du nom de tous les projets gérés par Maven. Le `groupId` est unique pour une organisation qui crée des projets logiciels (entreprises, associations, etc...) et appartient en général à cette organisation. La façon naturelle pour choisir ce nom racine est de procéder comme pour le choix des noms de package : d'utiliser un nom de domaine que l'on possède.
- Vient ensuite l'`artifactId`. Il s'agit du nom du projet proprement dit. Un projet ne change pas de nom ni de groupe, il garde donc le même `groupId` et `artifactId` tout au long de son existence. L'`artifactId` doit donc être choisi avec soin.
- Le dernier paramètre est le numéro de version. Un numéro de version évolue dans la vie d'un projet. Un projet donné, repéré par un `groupId` et `artifactId` peut donc exister en plusieurs versions.

Eclipse & Maven

Cours & Tutoriaux

Retour au blog Java le soir

Table des matières

1. Création d'un projet Maven dans Eclipse
2. Configuration d'un projet Maven dans Eclipse
3. Ajout d'une dépendance à notre projet
4. Choix d'une installation de Maven dans Eclipse

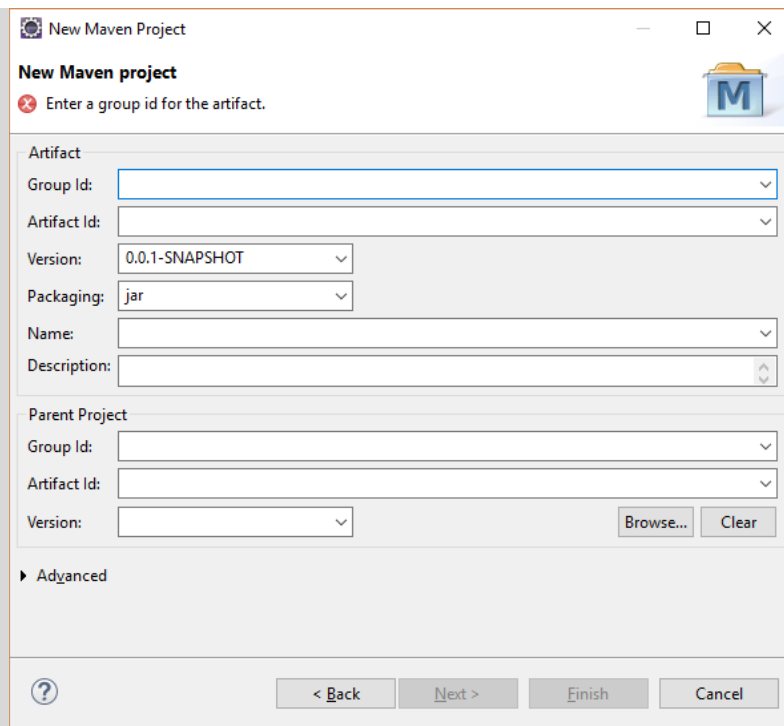


Figure 4. Deuxième panneau de configuration d'un projet Maven

Remplissons donc ce panneau avec nos valeurs afin de terminer la création de notre projet. On ne touche pas à l'élément packaging, qui est JAR pour les projets Java classiques.

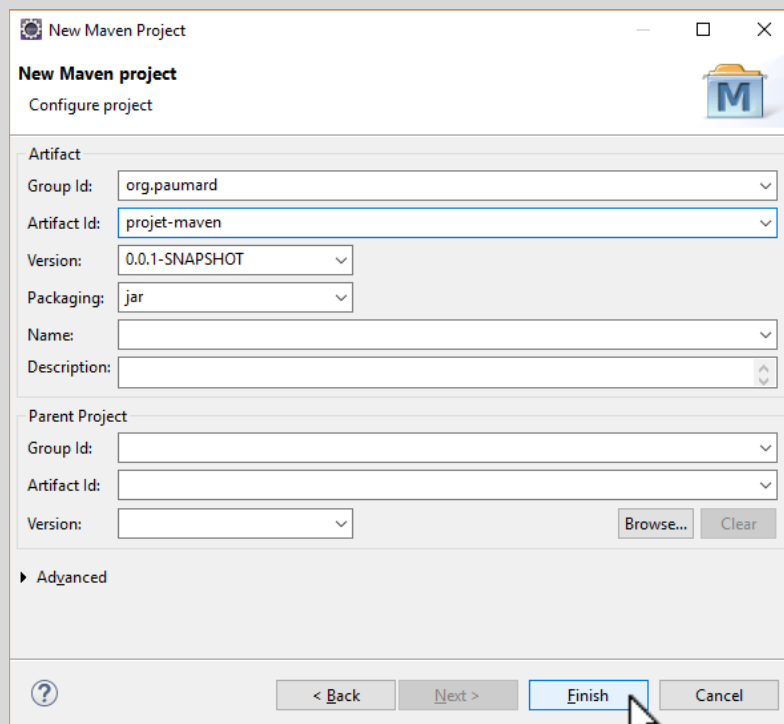
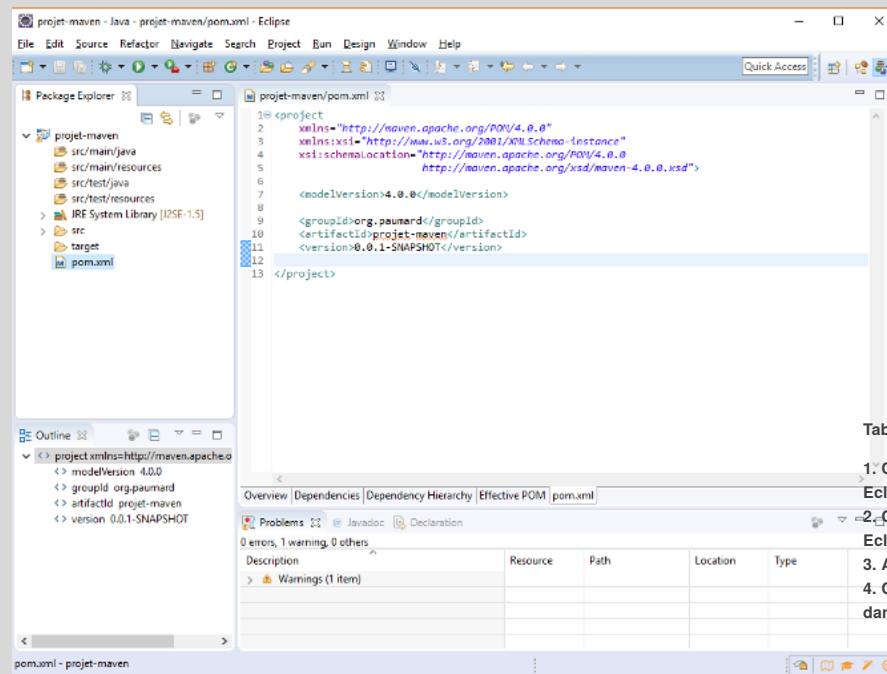


Figure 5. Deuxième panneau de configuration d'un projet Maven, rempli

2. CONFIGURATION D'UN PROJET MAVEN DANS ECLIPSE

Notre projet créé apparaît donc dans la vue Package explorer d'Eclipse sous la forme suivante.



Eclipse & Maven

Cours & Tutoriaux

[Retour au blog Java le soir](#)

Table des matières

1. Création d'un projet Maven dans Eclipse
2. Configuration d'un projet Maven dans Eclipse
3. Ajout d'une dépendance à notre projet
4. Choix d'une installation de Maven dans Eclipse

Figure 6. Projet créé dans Eclipse

On peut remarquer d'emblée plusieurs choses.

Tout d'abord, Maven crée des projets Java avec une structure particulière. Il crée quatre répertoires de code source : `src/main/java` et `src/test/java`, pour le code source proprement dit de notre projet et des tests. Ces deux répertoires ont deux statuts très différents. Les deux autres répertoires créés sont `src/main/resources` et `src/test/resources`. Ce sont des répertoires dans lequel on range est fichiers de ressources, différents du code.

Le code source de notre application doit être créé dans `src/main/java`.

On peut redéfinir cette structure standard (pour Maven) de répertoires, mais il n'y a pas d'intérêt à le faire.

Maven nous a ensuite créé un répertoire `target`. C'est le répertoire qu'il utilise pour produire tous les fichiers de notre projet. Ce sont les fichiers des classes compilées, les rapports de test, les fichiers JAR, WAR, etc...

Enfin Maven nous a créé un fichier `pom.xml`. Ce fichier, appelé fichier POM (Project Object Model) est le fichier central de notre projet Maven, dans lequel Maven enregistre toutes les informations dont il a besoin. La structure complète d'un fichier POM peut être très complexe, nous n'en verrons que quelques aspects.

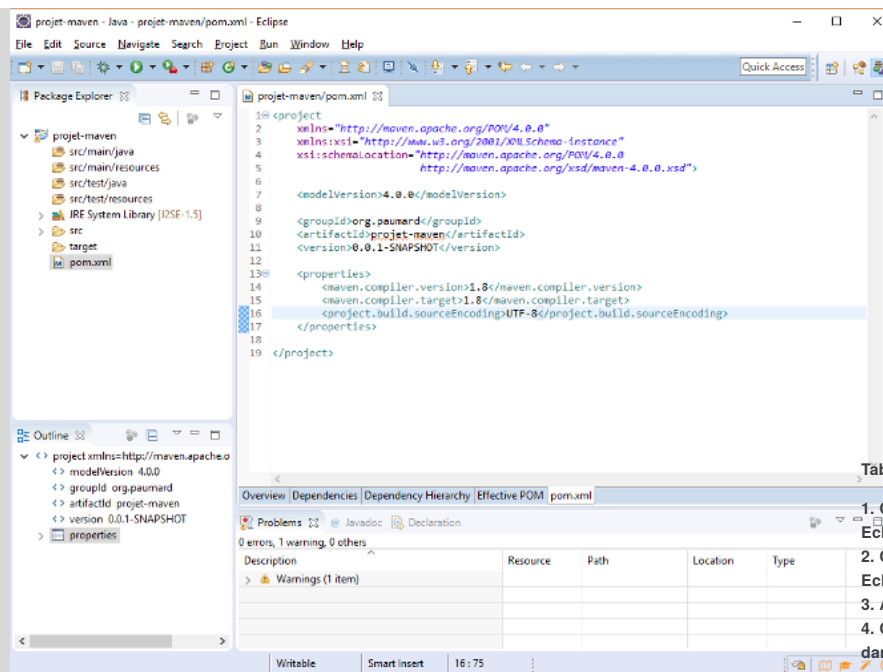
On peut aussi remarquer que Maven a configuré notre projet comme un projet Java 5. C'est le premier point que nous allons revoir, passer en Java 8, un peu plus à jour que Java 5, qui date tout de même de 2004.

La capture d'écran suivante nous montre le fichier POM ouvert. On reconnaît les différents éléments que nous avons rempli dans ce fichier XML. Il est parfaitement possible de créer un projet Maven entièrement à la main en créant ce fichier à partir de rien.

La première chose à configurer est la version de Java que nous voulons utiliser, ainsi qu'un point de configuration qui peut paraître un peu obscur à ce stade : l'encodage des fichiers source.

Comme on le voit sur la capture d'écran suivante, on a ajouté les lignes suivantes à notre fichier POM.

```
<properties>
  <maven.compiler.source>1.8</maven.compiler.source>
  <maven.compiler.target>1.8</maven.compiler.target>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
```



Eclipse & Maven

Cours & Tutoriaux

Retour au blog Java le soir

Table des matières

1. Création d'un projet Maven dans Eclipse
2. Configuration d'un projet Maven dans Eclipse
3. Ajout d'une dépendance à notre projet
4. Choix d'une installation de Maven dans Eclipse

Figure 7. Propriétés supplémentaires dans le fichier POM

Ces propriétés font ce que l'on pense qu'elles font. Elles indiquent à Maven que notre code source est du Java 8, qu'il doit être compilé en Java 8, et que les fichiers source sont encodés en UTF-8.

Le problème est que malgré cette déclaration, Eclipse continue à nous dire que notre projet est un projet Java 5. Pour lui faire entendre raison, il faut en fait rafraîchir la configuration de notre projet Maven. Cela se fait dans le menu contextuel de notre projet Eclipse, sous l'option Maven, se trouve une commande de rafraîchissement, comme dans la capture suivante.

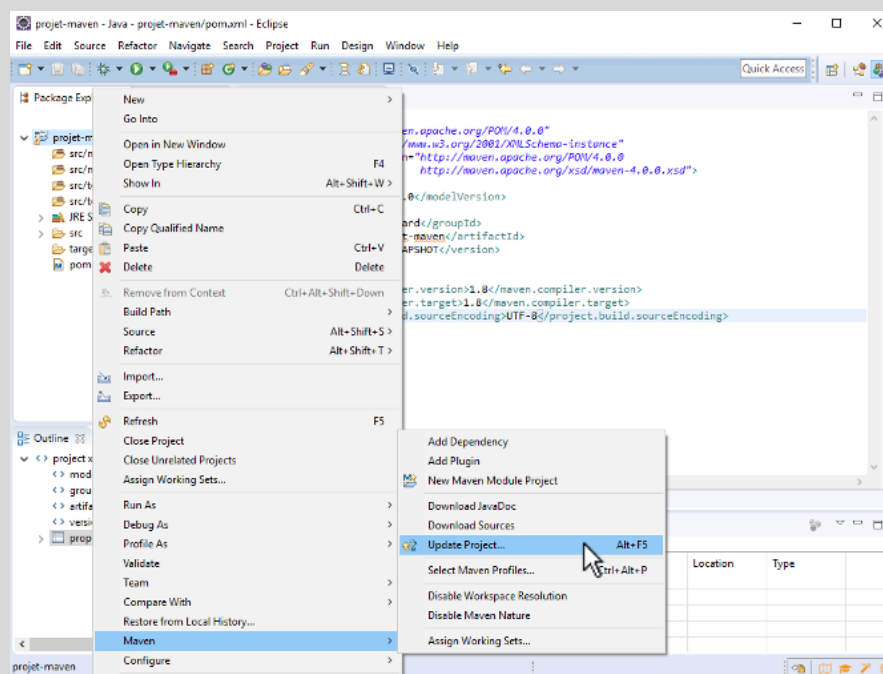
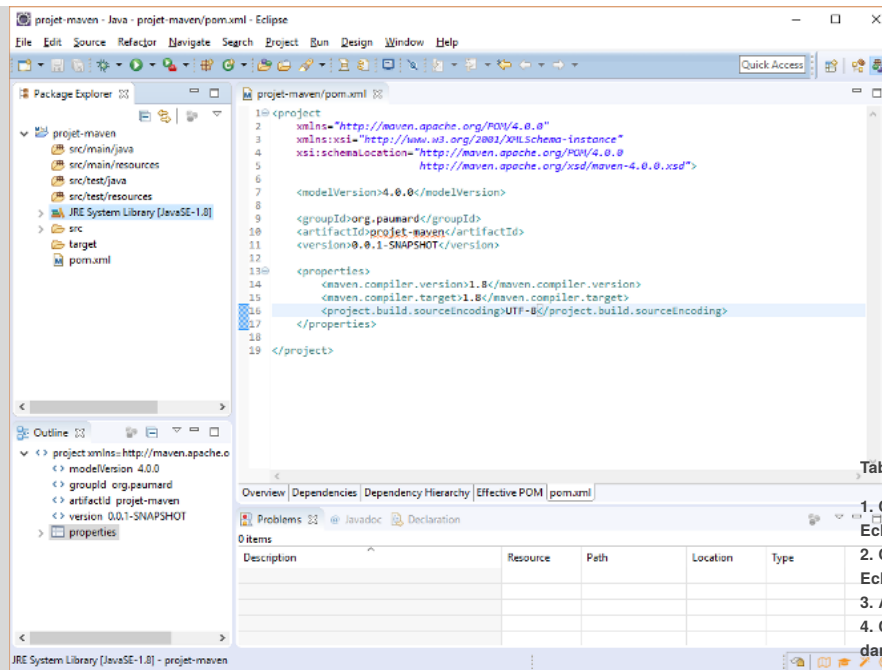


Figure 8. Mise à jour de la configuration de notre projet

On peut configurer certains aspects de ce rafraîchissement. Nous n'allons pas entrer dans ces détails et conserver la configuration par défaut.

Notre projet se met correctement à jour, et passe en Java 8 comme nous l'avons déclaré.



Eclipse & Maven

Cours & Tutoriaux

Retour au blog Java le soir

Table des matières

1. Création d'un projet Maven dans Eclipse
2. Configuration d'un projet Maven dans Eclipse
3. Ajout d'une dépendance à notre projet
4. Choix d'une installation de Maven dans Eclipse

Figure 9. Projet mis à jour par rapport au fichier POM

3. AJOUT D'UNE DÉPENDANCE À NOTRE PROJET

Une des premières utilités de Maven est de gérer automatiquement pour nous les dépendances de nos projets Java. Un projet industriel peut dépendre de dizaines, voire de centaines d'autres projets, certains développés en interne, d'autre en externe, qu'ils soient Open-source ou non.

Maven propose un mécanisme très pratique de déclaration de ces dépendances. Il gère le téléchargement automatique de ces dépendances dans un cache sur notre disque local, et la déclaration de ces dépendances dans Eclipse.

Ajoutons le code suivant dans notre POM.

```
<dependencies>
  <dependency>
    <groupId>dom4j</groupId>
    <artifactId>dom4j</artifactId>
    <version>1.6.1</version>
  </dependency>
</dependencies>
```

Cette déclaration indique à Maven que nous souhaitons mettre Dom4J en version 1.6.1 en dépendance de notre projet. Maven va commencer par regarder dans son cache local si cette dépendance s'y trouve. Si ce n'est pas le cas, il va interroger un repository de dépendances sur Internet (en général il s'agira de Maven Central : <http://search.maven.org/>, on peut également redéfinir ce point) afin de savoir si cette dépendance existe. Si tel est le cas, elle sera ajoutée au cache local.

Maven est assez malin pour se rendre compte que notre dépendance Dom4J dépend en fait elle-même d'un autre projet : xml-apis. Il va donc également aller chercher cette dépendance, dite transitive. En fait il n'y a rien de magique là-dedans, cette dépendance est déclarée dans le POM de Dom4J.

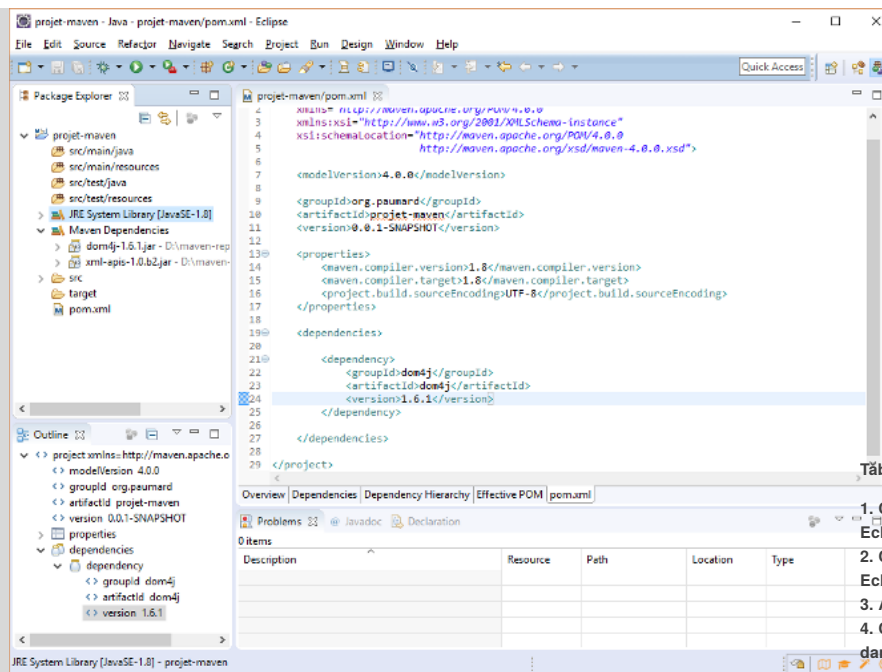


Figure 10. Ajout d'une dépendance Maven

Maven est également capable d'aller chercher le code source et la Javadoc de nos dépendances. On peut le configurer pour qu'il le fasse par défaut. Si ce n'est pas le cas, on peut demander ces téléchargement élément par élément, dans le menu contextuel de la dépendance.

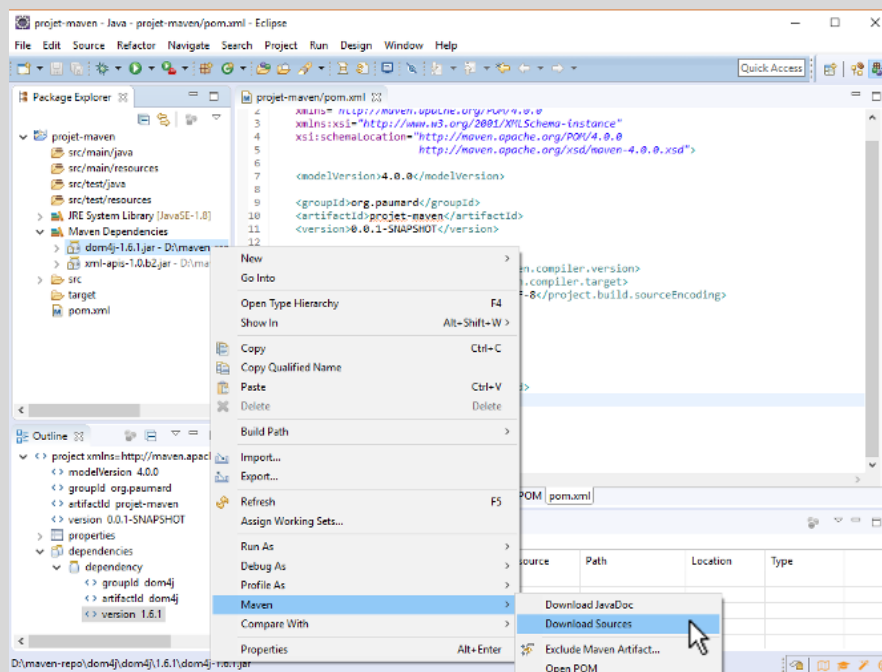


Figure 11. Téléchargement des sources et de la Javadoc

4. CHOIX D'UNE INSTALLATION DE MAVEN DANS ECLIPSE

Comme nous l'avons dit en début de ce tutorial, Maven est un outil séparé d'Eclipse, qui peut d'ailleurs se télécharger séparément.

Lorsque l'on télécharge Eclipse, on télécharge avec lui une version de Maven, qui est embarquée dans Eclipse. On peut

Eclipse & Maven

Cours & Tutoriaux

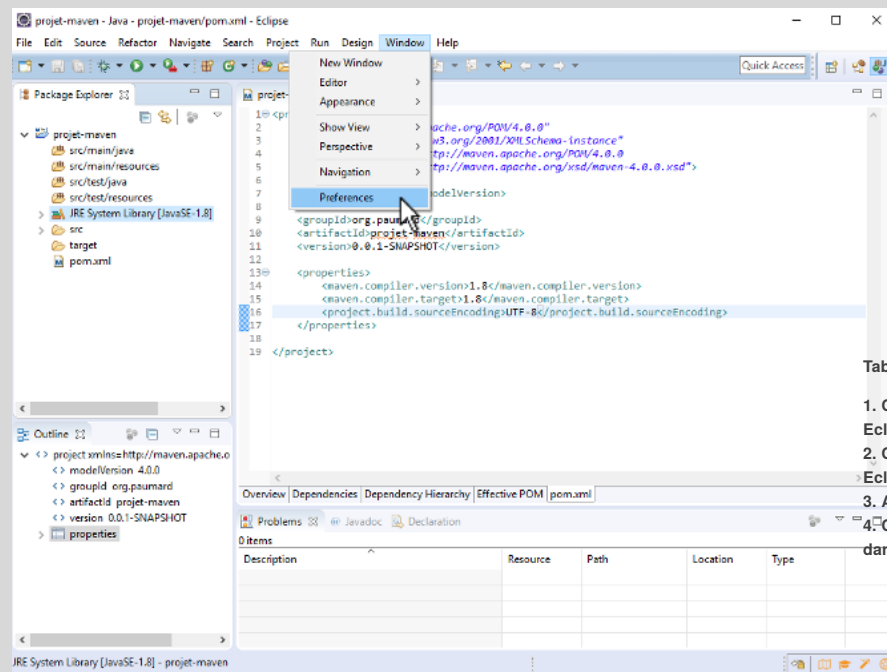
Retour au blog Java le soir

Table des matières

1. Création d'un projet Maven dans Eclipse
2. Configuration d'un projet Maven dans Eclipse
3. Ajout d'une dépendance à notre projet
4. Choix d'une installation de Maven dans Eclipse

aussi choisir d'utiliser une autre version de Maven, installée dans un répertoire de notre disque, avec sa configuration propre.

Pour ce faire, il faut aller chercher dans les Préférences d'Eclipse (Menu Window).



Eclipse & Maven

Cours & Tutoriaux

Retour au blog Java le soir

Table des matières

1. Création d'un projet Maven dans Eclipse
2. Configuration d'un projet Maven dans Eclipse
3. Ajout d'une dépendance à notre projet
4. Choix d'une installation de Maven dans Eclipse

Figure 12. Préférences de Maven

On peut filtrer l'ensemble des préférences avec le mot-clé Maven, et obtenir l'ensemble des préférences Eclipse pour Maven.

La première que nous allons voir est la préférence Installation, qui nous permet de choisir l'installation de Maven qu'Eclipse doit utiliser. On peut aller chercher une installation locale au travers du bouton Add... et naviguer vers le bon répertoire.

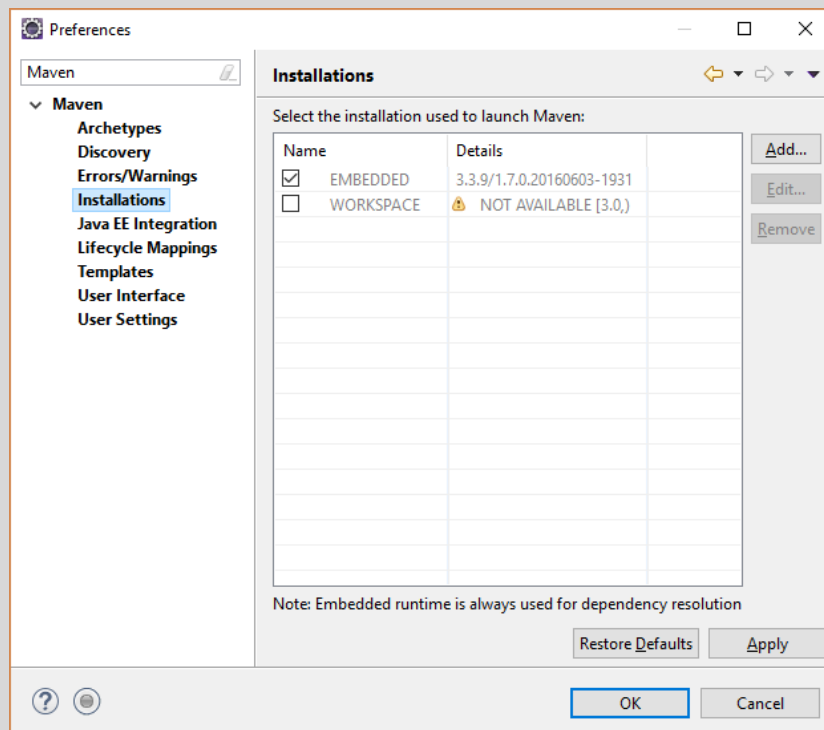


Figure 13. Installations de Maven

La deuxième préférence que nous voyons ici, permet d'avoir des informations sur la configuration de Maven. En particulier, on peut lire ici le répertoire que Maven utilise comme cache. Un cache bien utilisé peut rapidement atteindre

plusieurs Go de données. Il n'est donc pas inutile de s'arranger pour que tous nos projets Eclipse pointe bien vers le même Maven, et vers le même cache.

Il est possible d'utiliser plusieurs installations de Maven, dans plusieurs versions différentes. Cela dit, toutes ces installations peuvent pointer vers le même cache. La structure du cache Maven est très simple, et reste compatible de version en version.

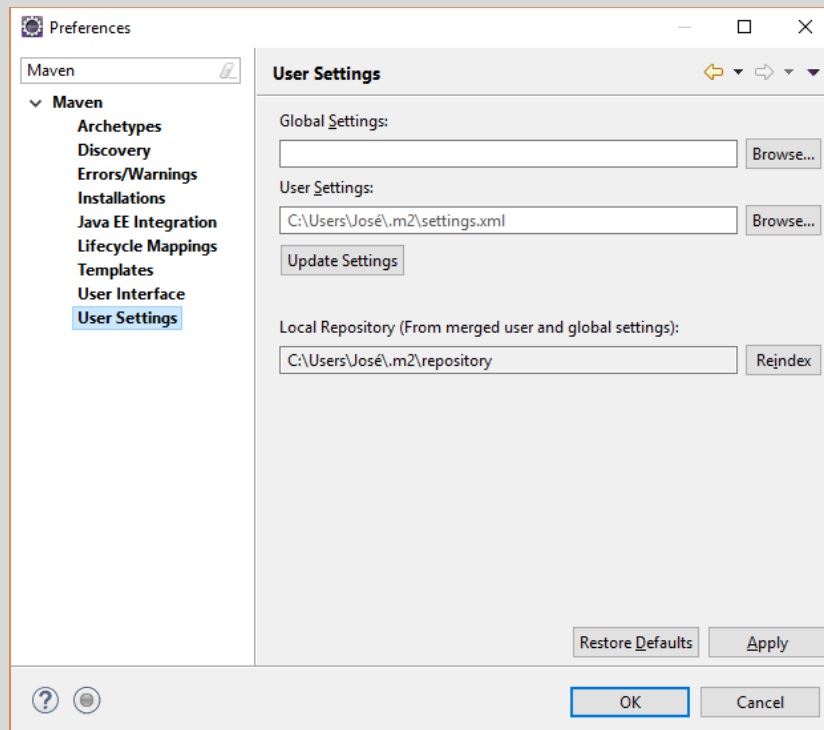


Figure 14. Repository Maven

[Eclipse & Maven](#)

[Cours & Tutoriaux](#)

[Retour au blog Java le soir](#)

Table des matières

1. Création d'un projet Maven dans Eclipse
2. Configuration d'un projet Maven dans Eclipse
3. Ajout d'une dépendance à notre projet
4. Choix d'une installation de Maven dans Eclipse