

| | |
|---|-------------------------------------|
|  | M1 Info DL |
| | Développement collaboratif, Qualité |

TP 2 – Prise en main d'Apache Maven et d'IntelliJ¹

Ce TP et les suivants s'effectuent avec l'environnement de développement IntelliJ IDEA Ultimate.

Préliminaires *si vous utilisez les machines de l'UPS*

1. Pré-configurez IntelliJ pour éviter le dépassement de quota du lecteur « C: ». Pour cela, téléchargez sur le lecteur « Z: » et décompressez l'archive « Fix_IntelliJ_UPS.zip », puis double-cliquez sur le script « Configure IntelliJ at UPS.bat ».
2. Créez un dossier pour ce TP dans votre espace de travail sur le lecteur réseau « Z: ».
3. Lancez IntelliJ ; choisissez l'option « Licence Server » et spécifiez l'adresse du serveur de licence (en faisant attention à ne pas introduire d'espaces avec le copier-coller) : <http://fsi-ens-vjetons8.univ-tlse3.fr:8080>
4. Créez un dossier **MvnRepo** dans votre espace sur le lecteur « Z: ». Ce dossier correspondra au repository Maven local ⇒ toutes les dépendances de vos projets seront rapatriées dans ce dossier. Ce dossier n'est donc pas spécifique à l'UE DeQo et pourra être utilisé dans les autres UEs nécessitant Java.

Exercice 1 – Création du projet my-simple-stack

1. Dans IntelliJ, créez un nouveau projet de type Maven en **cochant l'option « Create from archetype »** puis sélectionnez l'archetype « maven-archetype-quickstart » version RELEASE.
Si la question vous est posée, **spécifiez le JDK** : créez un nouveau JDK pointant sur le JDK « jdk1.8.0_112 » se trouvant dans le dossier « C:\Program files\Java ».
Configurez votre projet en choisissant le **groupid** « *deqo.votre_quadrigramme* » et l'**artifactId** « my-simple-stack ». Note : le **quadrigramme** de François Dupond est « fdup ».
Une fois le projet créé, vous développerez vos classes dans le **package Java** « *deqo.votre_quadrigramme.mysimplestack* » (mysimplestack sans trait d'union, puisque un nom de packaging Java ne peut pas contenir de traits d'union).
Choisissez d'utiliser la version de Maven 3 embarquée avec IntelliJ et mettez à jour le « Local repository » pour pointer sur le dossier **MvnRepo** (cf. Figure 1).
Spécifiez le nom du projet « my-simple-stack » et la localisation du projet dans un dossier ad-hoc de votre espace de travail sur le lecteur « Z: ».

¹ IntelliJ IDEA Java édité par la société JetBrains



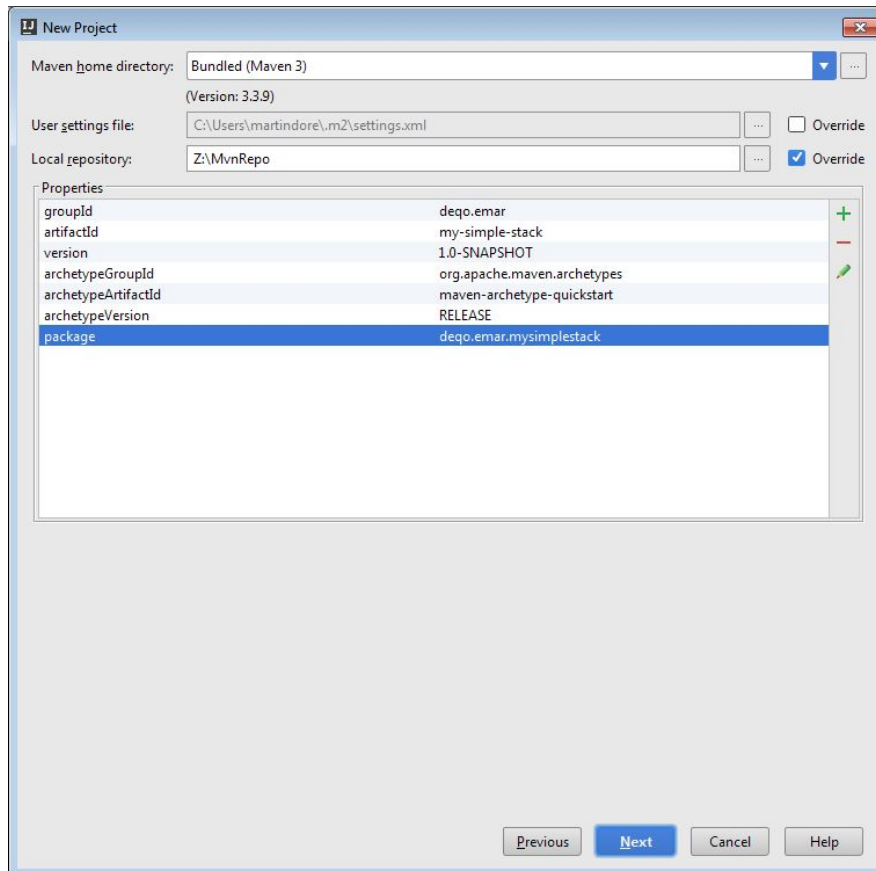


Figure 1. Configuration de Maven et choix du « Local repository »

2. Rendez visible la fenêtre «Maven projects» (menu View→Tool Windows→Maven Projects) puis cliquez sur le bouton « Reimport all Maven projects » pour resynchroniser la configuration Maven avec votre IDE.
3. Modifiez le POM de votre projet de sorte que celui-ci utilise la dernière version de JUnit 4 à la place de la version 3.8.1. Les informations sur les dépendances sont à rechercher sur le site : <http://search.maven.org/>. Cliquez sur le bouton « Reimport all Maven projects » pour resynchroniser la configuration Maven avec votre IDE.
4. Modifiez le POM de votre projet pour qu'il utilise les plugins ci-dessous (suivez les liens pour obtenir les éléments XML à rajouter dans le POM). Rappel : ces plugins de la catégorie « Reporting » seront utilisés lors de la génération du site du projet par Apache Maven (cf. **support de cours**).
 1. JXR - <https://maven.apache.org/plugins/maven-jxr-plugin/usage.html>
 2. Checkstyle - <https://maven.apache.org/plugins/maven-checkstyle-plugin/usage.html>
 3. FindBugs - <http://www.mojohaus.org/findbugs-maven-plugin/usage.html> → attention : **remplacez 3.0.4-SNAPSHOT par 3.0.3**
 4. Cobertura - <http://www.mojohaus.org/cobertura-maven-plugin/usage.html>



Remarque : il est possible qu'**IntelliJ colore en rouge le nom des plugins** dans le POM, ce n'est pas un problème et c'est typique lors de la première utilisation des plugins, cela disparaît en principe quelques instants après le premier lancement réussi des phases « clean test site » (cf. question 6).

5. Spécifiez deux configurations de lancement Maven (commande Run→Edit configurations, puis clic sur « + » (en haut à gauche) et sur « Maven ») pour :
6. le lancement des phases clean et package (tapez « clean package » dans le champ « Command line »)
7. le lancement des phases clean, test et site (tapez « clean test site » dans le champ « Command line »).

Testez vos 2 configurations en les lançant l'une après l'autre. Explorez le dossier « target » de votre projet pour constater le résultat des 2 commandes Maven. À l'aide d'un navigateur, explorez en détail le site généré par Maven (y compris les pages générées par les plugins).


Exercice 2 – Gestion du projet my-simple-stack avec Git et IntelliJ

1. Dans le menu VCS cliquez sur « Enable Version Control Integration ». Sélectionnez Git comme *version control system*. Contrôlez avec le terminal intégré d'IntelliJ (menu View→Tool Windows→Terminal ou raccourci clavier Alt+F12) que le dossier .git a bien été créé à la racine de votre projet. **Conseil :** sous Windows, pour utiliser Git Bash comme terminal dans IntelliJ, cliquez sur File→Settings→Tools→Terminal, puis spécifiez le chemin « "C:\Program Files\Git\bin\sh.exe" --login -i » ou « "C:\Git\bin\sh.exe" --login -i » dans le champ « Shell path ».
2. Dans IntelliJ, à la racine de votre projet supervisé à présent par Git, créez le fichier « .gitignore » permettant d'ignorer :
 - les fichiers ayant pour extension « .class »,
 - le dossier « target/ »,
 - le dossier « .idea/ » et les fichiers ayant pour extension « .iml ».
3. En utilisant la vue « Changes » d'IntelliJ, ajoutez l'ensemble des fichiers à l'index (cliquez droit sur l'item « Unversioned files » puis sur « add to VCS »). Effectuez le « commit » de vos fichiers après avoir renseigné le « commit message » (clic droit sur l'item « Default » puis sur « Commit changes »).
4. Créez sur votre compte GitHub un projet nommé « my-simple-stack ».
5. Via le terminal (Git Bash sous Windows), ajoutez à votre projet local le remote « origin » référençant le projet my-simple-stack.

Rappel : pour un **accès HTTPS** l'URL du dépôt GitHub commencera par « https://... ».

6. Effectuez la synchronisation du remote « origin » avec votre projet local (commande de menu VCS→Git→Push).



| | |
|--|-------------------------------------|
|  MASTER DÉVELOPPEMENT LOGICIEL | M1 Info DL |
| | Développement collaboratif, Qualité |

7. Écrivez dans le projet « my-simple-stack » un programme permettant d'implémenter l'interface SimpleStack. Vous introduirez les interfaces et classes nécessaires ainsi que les tests de telle sorte que la **couverture du code par les tests** soit de 100%.

```

public interface SimpleStack {
    /**
     * Tests if this stack is empty
     */
    public boolean isEmpty();

    /**
     * Returns the number of items in this stack.
     */
    public int getSize();

    /**
     * Pushes an item onto the top of this stack.
     * null item is allowed.
     */
    public void push(Item item);

    /**
     * Looks at the object at the top of this stack without removing it from the stack.
     */
    public Item peek() throws EmptyStackException;

    /**
     * Removes the object at the top of this stack and returns that object as the value of this function.
     * @throws EmptyStackException if this stack is empty.
     */
    public Item pop() throws EmptyStackException;
}

```

Note : à partir du code d'une classe (par exemple Item) IntelliJ peut vous aider à générer un squelette de classe de test (ItemTest). Il suffit de cliquer sur le nom de la classe dans le fichier Item.java, de taper Alt+Entrée, choisir Create Test, puis JUnit 4, et cocher les méthodes à tester.



Cette œuvre est mise à disposition selon les termes de la [Licence Creative Commons Paternité - Pas d'Utilisation Commerciale 3.0 France](https://creativecommons.org/licenses/by-nc/3.0/fr/).