

Manuel d'intallation MyNeighTool

Sommaire

1. [Prérequis](#)
2. [Importer le projet depuis github](#)
3. [Configurer le projet](#)
4. [Lancer le serveur](#)

1. Prérequis

Tout d'abord, l'installation doit avoir les logiciels suivants d'installés sur son ordinateur :

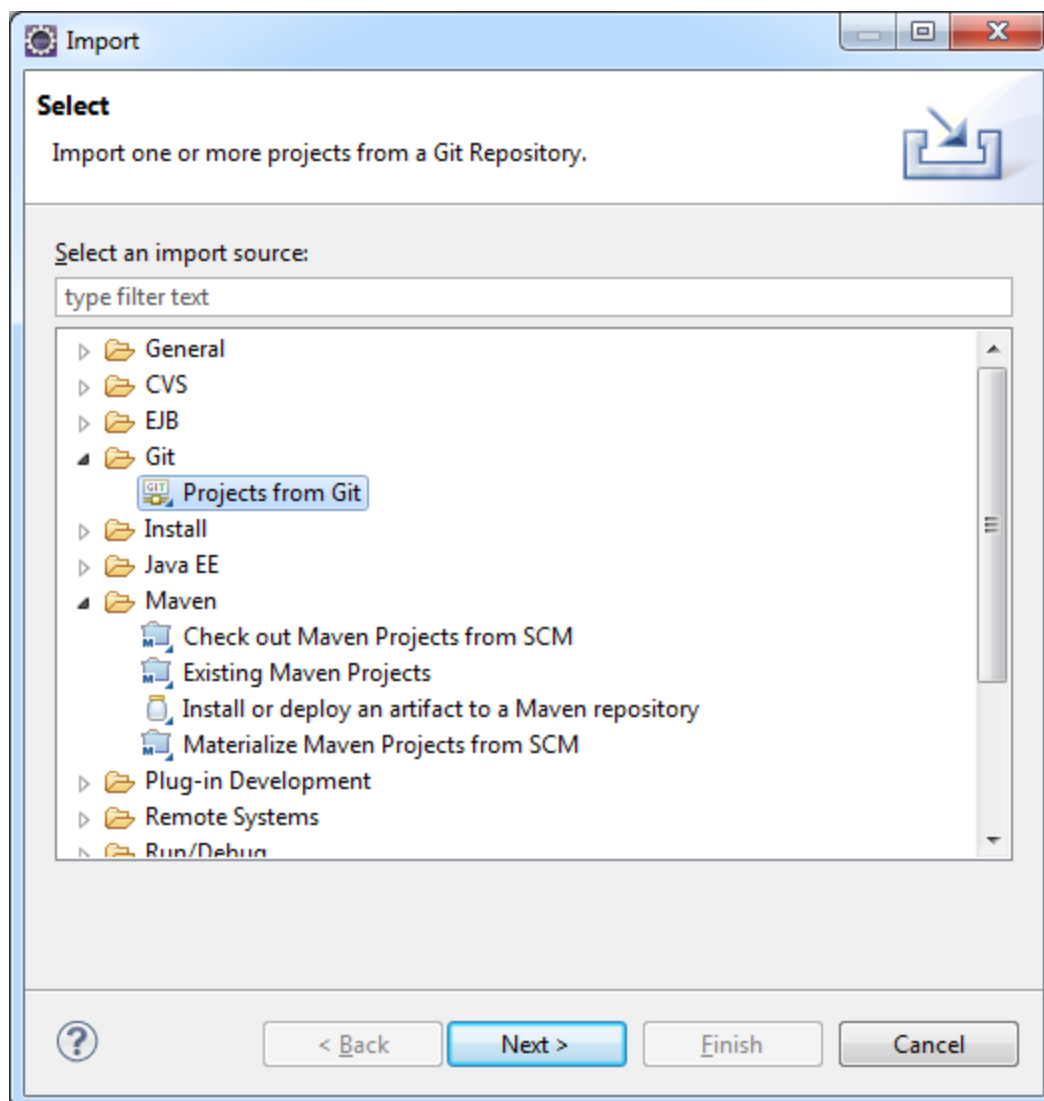
- Eclipse (J2EE de préférence pour que Maven et eGit soient déjà installés)
- Le Plugin Maven (<http://download.eclipse.org/technology/m2e/releases>)
- Le Plugin eGit (<http://download.eclipse.org/egit/updates>)
- Une plateforme de développement Web telle que WampServer (<http://www.wampserver.com/>) (Pas nécessaire si on travail sur un ordinateur du cremi)

2. Importer le projet depuis github

En utilisant eGit de Eclipse, vous pouvez importer le projet depuis github qui se trouve à l'adresse suivante : <https://github.com/MyNeighTool/Backend.git>.

Pour se faire, faites un clique droit dans la vue *Package Explorer* de eclipse et cliquez sur *import*.

Sélectionnez ensuite Git/Project from Git :



Cliquez ensuite sur *Clone url* et entrez le lien vers le projet.

Import Projects from Git

Source Git Repository
Enter the location of the source repository.

Location

URI:

Host:

Repository path:

Connection

Protocol:

Port:

Authentication

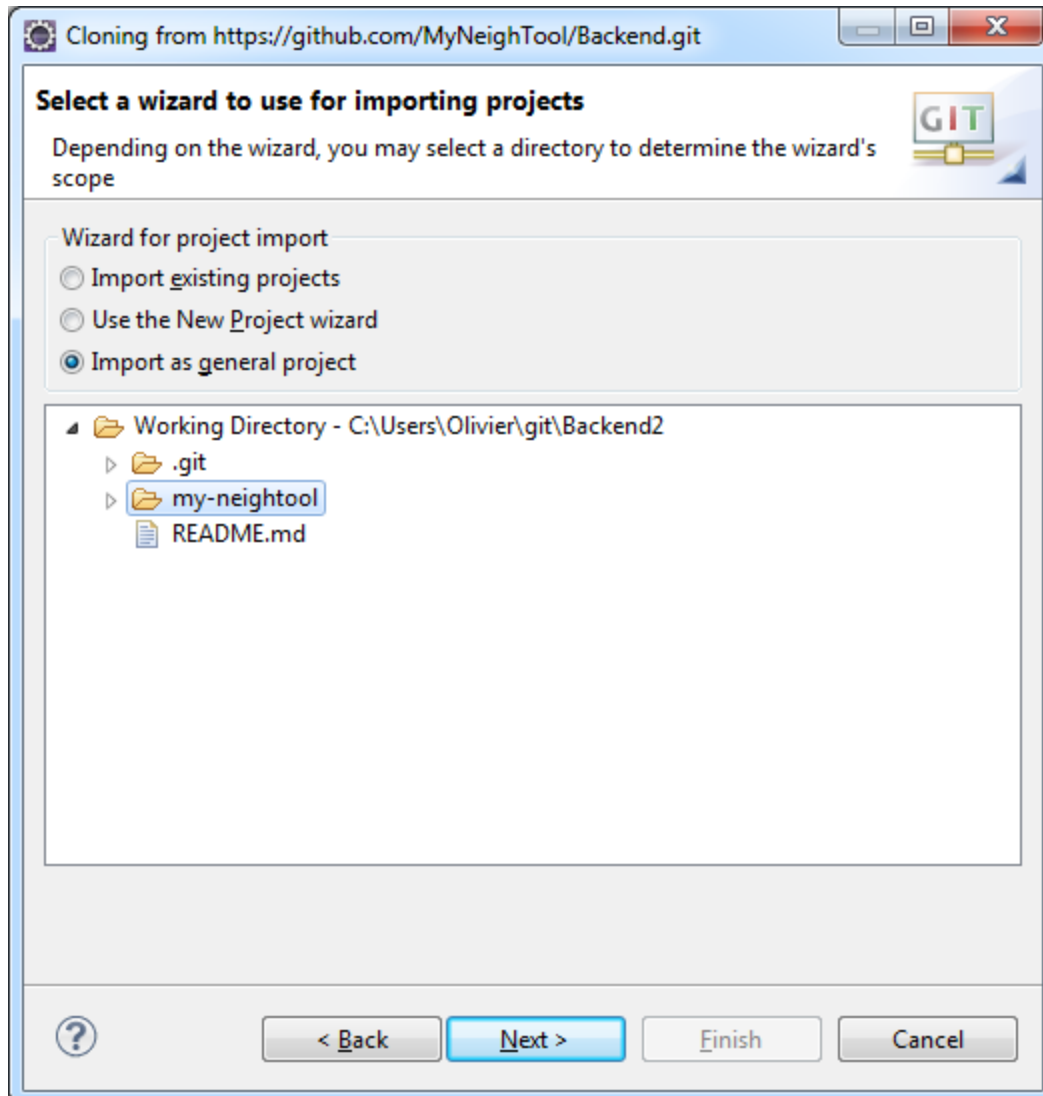
User:

Password:

Store in Secure Store ☐

? < Back Next > Finish Cancel

Cliquez ensuite sur Next et importez my-neightool en temps que *general project*



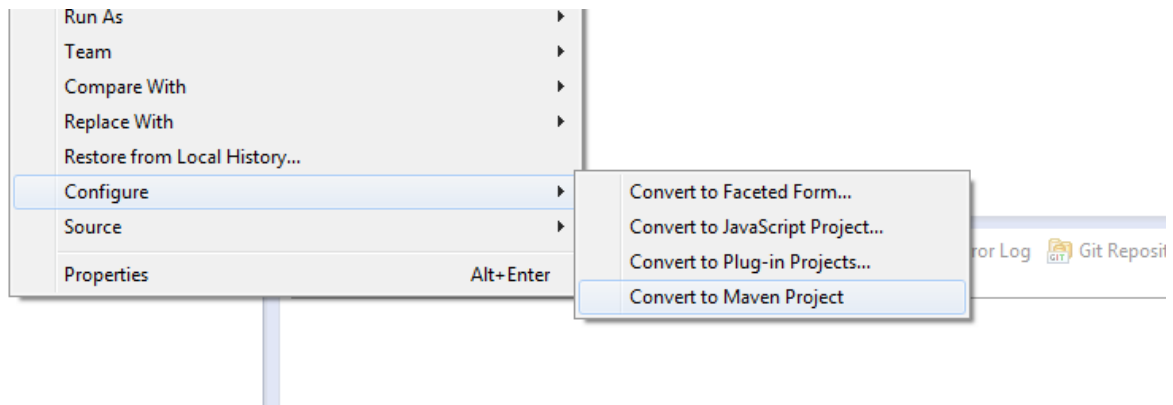
Si la petite fleche pour descendre dans l'arborescence ne s'affiche pas, alors cliquez sur Back et revenez en cliquant sur Next.

Ensuite, cliquez sur Next jusqu'à pouvoir cliquer sur Finish.

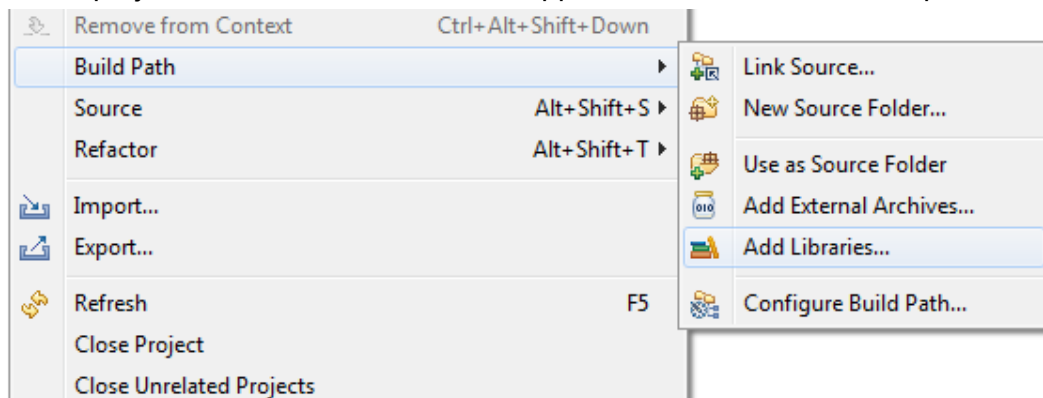
Le projet est alors importé dans votre workspace.

3. Configurer le projet

Ensuite, faire un clic droit sur le projet, aller dans *Configure*, et convertir le projet en projet Maven.



Quand le projet sera converti, des erreurs apparaîtront. Il faudra alors importer la librairie JUnit4.

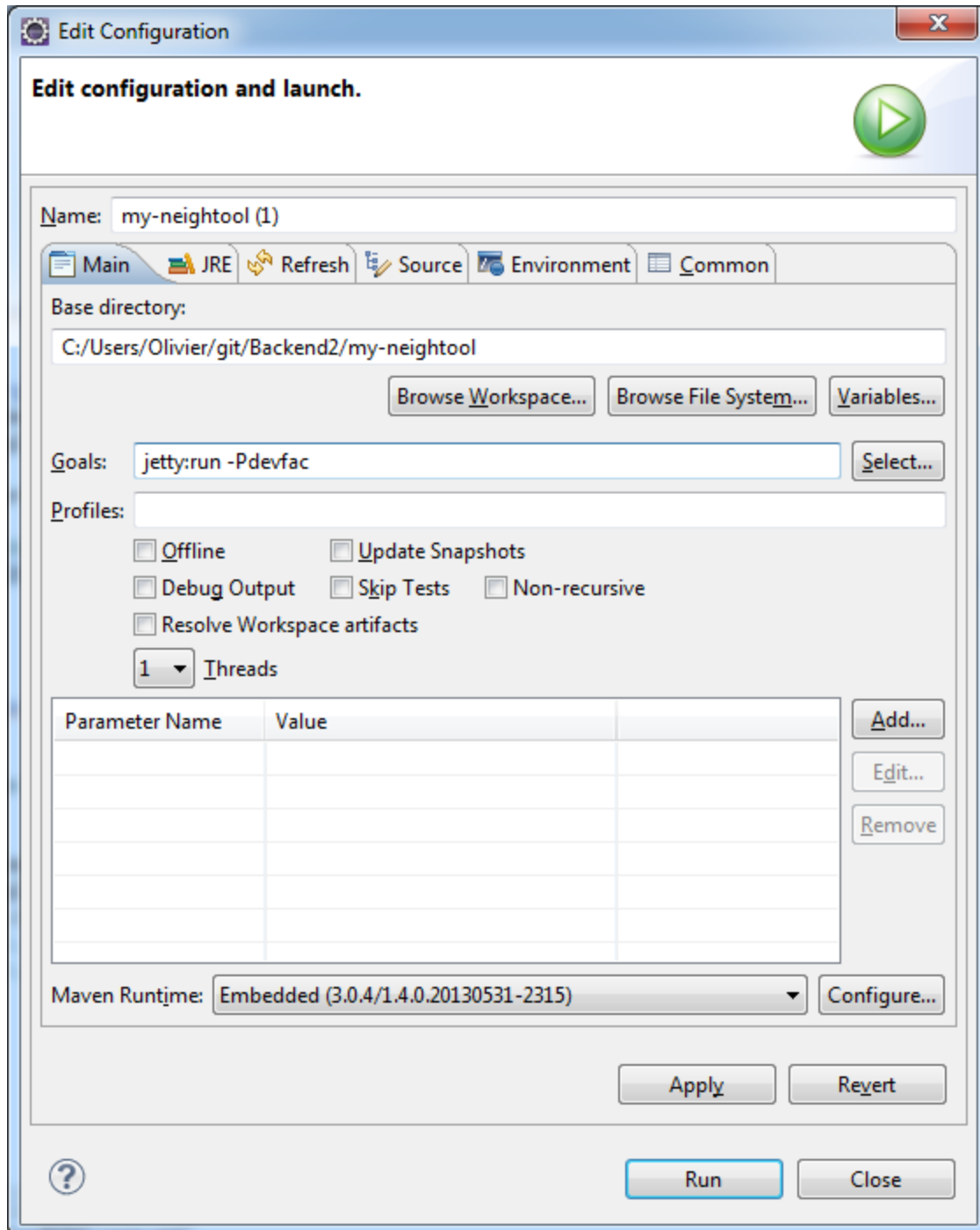


Ensuite, éditer le fichier pom.xml en allant dans le profil devfac si vous êtes sur un ordinateur du cremi en utilisant les informations du service emi (<https://services.emi.u-bordeaux1.fr/dbserver/phpMyAdmin/>). Sinon, laissez le fichier pom tel qu'il est.

```
<profiles>
  <profile>
    <!-- pour dev à la fac avec la bd emi , rajouter -Pdevfac après le jetty:run -->
    <id>devfac</id>
    <properties>
      <!-- <jdbc.dialect>org.hibernate.dialect.H2Dialect</jdbc.dialect> -->
      <jdbc.dialect>org.hibernate.dialect.MySQLDialect</jdbc.dialect>
      <jdbc.driver>com.mysql.jdbc.Driver</jdbc.driver>
      <jdbc.url>
        jdbc:mysql://dbserver:3306/obraik<!-- mettre ici le nom de bd -->
      </jdbc.url>
      <jdbc.user>obraik</jdbc.user> <!-- pensez a mettre votre login -->
      <jdbc.password>motd'passe</jdbc.password> <!-- pensez a mettre votre password -->
    </properties>
  </profile>
  <profile>
    <!-- pour dev chez soit avec par exemple wampserver, ici par default -->
    <id>devhome</id>
    <activation>
      <activeByDefault>true</activeByDefault>
    </activation>
    <properties>
      <jdbc.dialect>org.hibernate.dialect.MySQLDialect</jdbc.dialect>
      <jdbc.driver>com.mysql.jdbc.Driver</jdbc.driver>
      <jdbc.url>jdbc:mysql://localhost:3306/neighborhood</jdbc.url>
      <jdbc.user>root</jdbc.user>
      <jdbc.password></jdbc.password>
    </properties>
  </profile>
</profiles>
```

4. Lancer le serveur

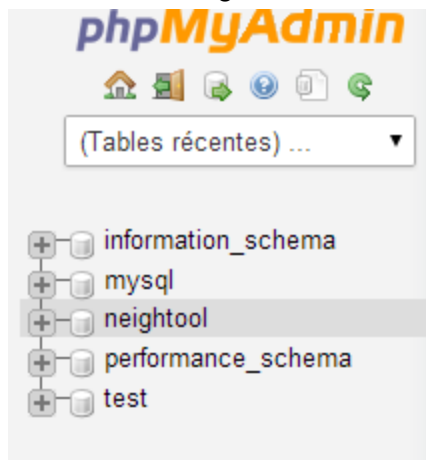
Si vous développez sur un ordinateur du cremi, lancez le projet maven avec la commande `jetty:run -Pdevfac`.



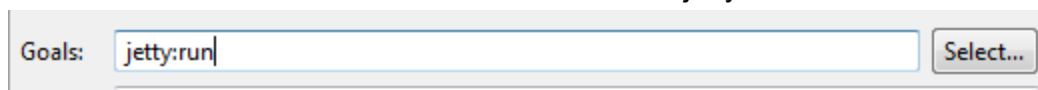
Sinon, lancez d'abord WampServer, et attendez que l'icône devienne verte comme ceci :



Une fois wampServer lancé, allez dans PhpMyAdmin, rajoutez une base de donnée ayant comme nom neightool.



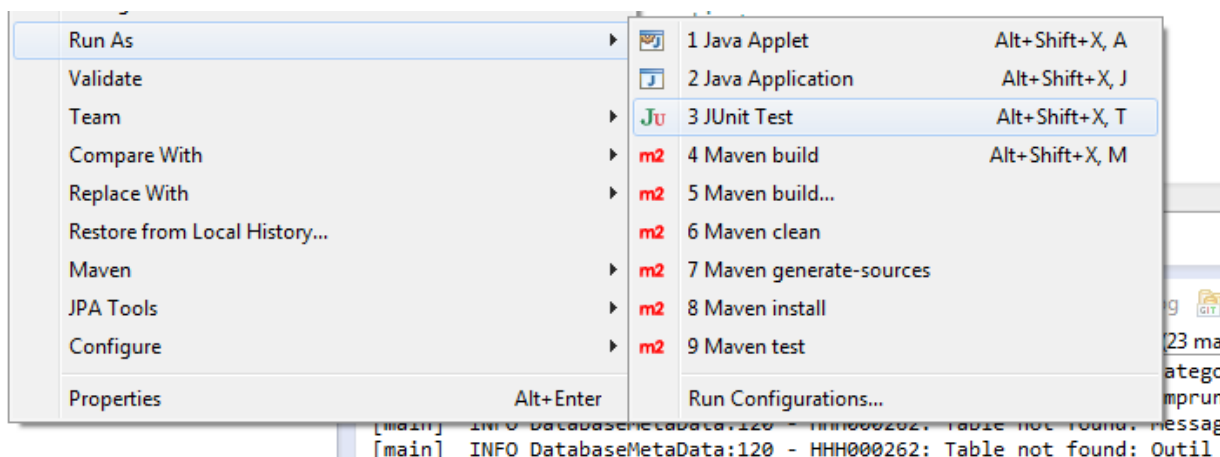
Lancez ensuite le build Maven avec la commande jetty:run.



Attendez ensuite que la console affiche Started Jetty server


```
[main] INFO DatabaseMetaData:120 - HHH000262: Table not found
[main] INFO DatabaseMetaData:120 - HHH000262: Table not found
[main] INFO DatabaseMetaData:120 - HHH000262: Table not found
[main] INFO DatabaseMetaData:120 - HHH000262: Table not found
[main] INFO log:88 - Logging to org.slf4j.impl.Log4jLoggerAda
[main] DEBUG log:51 - Cross-origin filter configuration: allow
2014-03-23 13:24:10.635:INFO:oejs.AbstractConnector:Started Se
[INFO] Started Jetty Server
```

Une fois le server lancé, il va falloir créer les tables initiales de la base de données en lançant les tests JUnit.










Normalement, les test devraient tous s'effectuer correctement et donc s'afficher en vert comme ceci :

Finished after 9,02 seconds

Runs: 31/31  Errors: 0  Failures: 0



- ▷  com.ped.myneightool.TestAdresse [Runner: JUnit 4] (1,96
- ▷  com.ped.myneightool.TestCategorie [Runner: JUnit 4] (0,
- ▷  com.ped.myneightool.TestOutil [Runner: JUnit 4] (3,120 s
- ▷  com.ped.myneightool.TestEmprunt [Runner: JUnit 4] (0,3
- ▷  com.ped.myneightool.TestConnexion [Runner: JUnit 4] (0,
- ▷  com.ped.myneightool.TestUtilisateur [Runner: JUnit 4] (0,
- ▷  com.ped.myneightool.TestMessage [Runner: JUnit 4] (1,4

Vous pouvez maintenant tester l'application web en utilisant l'url <http://localhost:8080>.