TP JEE (0)
Introduction à MAVEN

1 Introduction à Apache Maven

Les projets Java (surtout JEE) ont la particularité de dépendre de beaucoup (trop ?) de librairies externes. Ainsi pour construire une application JEE il faut télécharger des JAR ("Java ARchive", fichier d'archive contenant des classes java compilées), les installer et les renseigner lors de la construction du logiciel.

Cette étape est complexe : outre la partie installation, il faut pouvoir renseigner à Eclipse où sont ces fichiers JAR. Cela va constituer le *buildpath* de l'application. Malheureusement ce *buildpath* diffère selon les machines des développeurs de l'équipe : */Users/Jean/mesJARS*, */home/Yves/ProgJava/MesJars*. Par la suite chaque développeur doit maintenir et gérer sa propre collection de librairies et sa propre configuration. Ces étapes sont souvent tellement complexes qu'elles sont en général laissées à la charge de l'architecte logiciel.

Apache Maven est un système très populaire de gestion de construction pour les projets Java, conçu pour supprimer les tâches difficiles du processus de build.

A partir d'une configuration Maven créée et gérée par un architecte, il est possible de construire des applications complexes sur des types de configurations différentes et avec des IDE différents (Eclipse, Netbeans, ...).

Dans les autres TPs les fichiers de configuration Maven vous seront donnés, mais afin de les comprendre il est nécessaire de faire une petite initiation. C'est le but de ce TP. De plus, nous verrons comment créer un projet JEE simple. La compréhension de cette étape est cruciale pour commencer le développement en JEE.

1.1 Création d'un JAR simple

Exercice

1. Créez un nouveau projet de type *Maven* appelé *MonPremierProjet* en vous inspirant de la ligne de commande de la Figure 1.
2. Commentez les arguments de la ligne de commande précédente. A quoi servent ces arguments ?
3. Parcourez le dossier qui a été créé. Que contient-il ? Qu'est-ce que le fichier pom.xml ? Où est la classe contenant la fonction main ?
4. A quoi sert la partie du fichier pom.xml contenant la chaîne de caractères "junit"' ?
5. Compilez le projet avec la commande mvn compile.

6. A quoi sert le dossier test ? Lancez les tests avec la commande mvn test.
7. Créez un JAR avec la commande mvn package. Où se trouve-t-il ?

```
mvn archetype:generate -DarchetypeGroupId=org.apache.maven.
    archetypes -DgroupId=com.mycompany.app -DartifactId=my-app
```

FIGURE 1 – Création d'un projet Maven. La commande va créer un projet avec une architecture classique (répondez par défaut aux questions). (file=samples/commandGenerateJarSimple)

1.2 Les plugins maven : création d'un *Runnable Jar*

La commande Maven *mvn package* nous a permis de créer un JAR qui contient la classe App.java. Nous allons utiliser les plugins maven pour transformer ce JAR en *Runnable JAR*. Un Runnable JAR est un JAR qui peut être exécuté. Afin de l'exécuter, un fichier *Manifest* doit être créé. Il sert à informer la JVM où se trouve la classe principale (celle qui possède une fonction main). Nous allons ici utiliser les plugins de type "build" de Maven.

Exercice

1. Modifiez le pom.xml afin d'intégrer l'extrait de source suivant :

```
<project>
....
 <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-shade-plugin</artifactId>
                <version>1.5</version>
                <executions>
                    <execution>
                        <phase>package</phase>
                        <goals>
                            <goal>shade</goal>
                        </goals>
                        <configuration>
                            <transformers>
                                <transformer implementation=
                                    "org.apache.maven.plugins
                                    .shade.resource.
                                    ManifestResourceTransformer
                                    ">
```

```
                                        <mainClass>com.mycompany
                                            .app.App</mainClass>
                                    </transformer>
                                </transformers>
                            </configuration>
                        </execution>
                    </executions>
                </plugin>
            </plugins>
        </build>
        ....
    </project>
```

2. Dans le nouveau pom.xml, à quoi sert la balise <mainClass> ? Et la balise <phase> ?
3. Compilez le nouveau JAR et lancez-le.

2 Un projet WEB

Dans cette partie, nous allons créer notre premier projet Web Java. Un projet web est comme en PHP un projet qui permet d'être exécuté sur un serveur. Nous allons voir comment en créer un, d'abord à la main, puis en utilisant Maven.

2.1 À la main

Exercice

1. Création de la structure hiérarchique du projet :

```
mkdir MonProjetWebMain
cd MonProjetWebMain
mkdir webapp
cd webapp
touch index.jsp
mkdir WEB-INF
cd WEB-INF/
touch web.xml
```

2. Modifiez le descripteur de déploiement web.xml.

```
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
  <display-name>Mon Appli Web</display-name>
    <welcome-file-list>
```

```
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

3. On modifie notre première JSP (JavaServer Page) : index.jsp

```
<html>
<body>
<h2>Hello World! </h2>
</body>
</html>
```

4. Créez un fichier WAR avec la commande jar -cvf webapp.war * (Depuis le dossier webapp).
5. Qu'est-ce qu'un fichier WAR ?

Déployer et tester votre fichier WAR

1. Lancez une instance de Tomcat (./bin/startup.sh) (shutdown.sh pour l'éteindre).
2. Trouvez le login/mot de passe de tomcat dans le fichier ./conf/tomcat-users.xml.
3. Rendez-vous sur la page http ://localhost :8080 puis, dans la partie Manager App.
4. Déployez votre fichier WAR et testez votre site web.
5. Qu'est ce que Tomcat ?

2.2 Avec Maven

Exercice :

1. Créez un projet Maven avec la commande suivante :

```
mvn archetype:generate -DarchetypeGroupId=org.apache.maven.
    archetypes -DarchetypeArtifactId=maven-archetype-webapp -
    DgroupId=org.iut.jee -DartifactId=
    MonPremierWebProjectMaven -Dversion=1.0-SNAPSHOT
```

2. Quelle est la différence entre ce pom.xml et le précédent ?
3. Configurez le fichier pom.xml pour avoir un profil pour votre serveur Tomcat. Inspirez vous du fichier samples/pom-mavenweb.xml. Attention aux balises *finalName* et *artifactId*
4. Commentez les différentes parties de ce fichier. À quoi servent les balises *repositories* et *profiles*. Quelle est l'utilité des différents plugins ?
5. Déployez et testez votre projet.

2.3 Création d'une Servlet

1. Générez un projet Eclipse avec la commande :[1]

```
mvn eclipse:eclipse -Dwtpversion=2.0
```

2. Ouvrez Eclipse et importez le projet généré (via "importer un projet existant dans l'espace de travail").
3. Dans Eclipse créez un nouveau dossier de sources (src/main/java).
4. Dans ce même dossier de sources, créez un nouveau package (org.iut.jee).
5. Pour finir, créez une nouvelle Servlet *HelloWorld.java*. Dans le menu de création de fichiers d'Eclipse, il y a une catégorie spéciale pour les Servlets.
6. Si ce n'est pas déjà fait, ajoutez l'annotation @WebServlet comme le montre l'extrait de code suivant.

```
@WebServlet("/hello")
public class HelloWorld extends HttpServlet {
        public HelloWorld() {}

        protected void doGet(HttpServletRequest request,
           HttpServletResponse response) throws
           ServletException, IOException {}

        protected void doPost(HttpServletRequest request,
           HttpServletResponse response) throws
           ServletException, IOException {}
}
```

7. Qu'est ce qu'une Servlet ? à quoi sert l'annotation *@WebServlet("/hello")* ?
8. Commentez et expliquez le rôle des différentes méthodes de la Servlet.
9. Copiez-collez le code suivant dans la méthode doGet de votre servlet :

```
PrintWriter out = response.getWriter();
out.println( "Hello !" );
out.flush();
out.close();
```

10. Dans */src/main/webapp/WEB-INF/*, il existe un fichier web.xml. À quoi sert-il ?
11. Supprimez-le. Expliquez pourquoi cette suppression est nécessaire.
12. Créez un nouveau WAR avec Maven, et déployez-le sur Tomcat avec la commande mvn tomcat:deploy.
    ATTENTION : Dans le cas d'un redéploiement, il faut utiliser la commande mvn tomcat:redeploy.
13. Testez la nouvelle servlet.

1. Note : si vous travaillez à la maison, attention à ce que la variable d'environnement JAVA_HOME pointe bien vers le dossier de votre JVM. Ce chemin dépend de votre système.

3 Intégration de Maven et Eclipse

Après avoir visionné la vidéo : /net/Bibliothèque/JEE/2011-2012/Maven2Eclipse.mov : ajoutez le serveur Tomcat dans Eclipse, puis déployez votre application depuis Eclipse.

1. Comment débugger une application depuis Eclipse ?

4 La culture Java EE (ou : avoir une bonne note à l'examen)

Répondez aux questions suivantes en vous aidant de votre navigateur favori :

1. Qu'est-ce que Maven ? Qu'est ce que ANT ? Quel est le lien entre ces deux outils ? Quel est leur équivalent en C++ (ou autres langages).
2. Quel est le nom du fichier de configuration Maven ?
3. Comment compiler un projet Maven ?
4. Comment générer un projet Eclipse depuis Maven ?
5. Qu'est-ce que Tomcat ?
6. Qu'est-ce qu'un fichier WAR ?
7. Qu'est-ce qu'un fichier JSP ?
8. Qu'est-ce qu'une Servlet ?