

Maven par la pratique



Qui suis-je ?

Rossi Oddet

Consultant IT chez SQLI
Manager JCertif Lab

Twitter : @rossioddet

Blog: <http://blog.rodde.com>



Maven c'est quoi ?



Votre outil de gestion de projet !



Que peut-on concrètement faire avec ?



Quelques services rendus à votre projet

Créez !

à l'aide de modèles

Compilez !

Gérer les dépendances, la version du JDK, ...

Testez !

Exécutez des tests unitaires, d'intégrations, ...

Versionnez !

Gérer les versions de votre projet

Assemblez !

JAR, WAR, EAR, ZIP, ...

Archivez !

dans un dépôt local ou distant

Qualifiez !

Vérifier la qualité de vos projets

Reportez !

Générer/envoyer divers rapports

Déployez !

dans un serveur d'application local, distant, ...



Maven, toute une philosophie

“convention plutôt que configuration”

“standard plutôt qu’outil”

“même modèle pour tous les projets”

“mvn install pour tout le monde”



Installer Maven

- Rendez-vous à la page de téléchargement : <http://maven.apache.org/download.html>
- Décompressez dans le répertoire de votre choix le fichier apache-maven-3.0.4-bin.zip
- Mettre le répertoire [MAVEN_INSTALL]/bin dans la variable Path
- Exécutez la commande : `mvn -version`



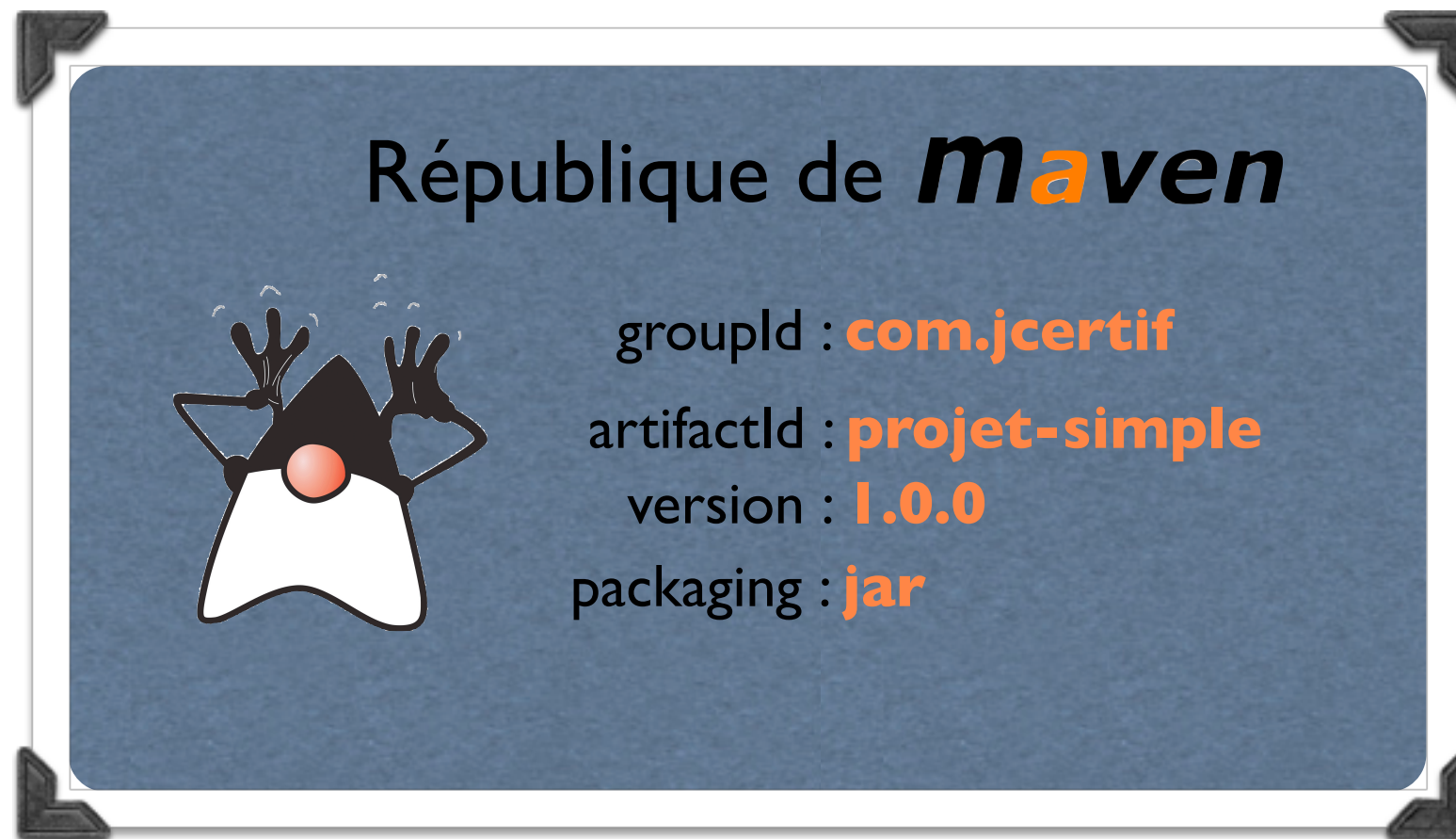
Le fichier settings.xml

Unique fichier XML de configuration

“convention plutôt que configuration”



La carte d'identité d'un projet



L'archivage d'un projet

<Répertoire archive> / <groupid> / <artifactId> / <version> / <artifactId>-<version>.<packaging>

Par exemple :

.m2/repository/com/jcertif/projet-simple/1.0.0/mon-projet-simple-1.0.0.jar



La structure par défaut d'un projet

```
mon-projet
  /pom.xml
  /src
    /main
      /java
      /resources
      /webapp
    /test
      /java
      /resources
  /target
```



TP I : Créer un projet simple

```
mvn archetype:generate -DgroupId=com.jcertif -DartifactId=projet-simple -Dversion=1.0.0 -  
DinteractiveMode=false
```



Les fichiers générés

projet-simple

/pom.xml

/src

/main

/java

/com

/jcertif

/App.java

/test

/java

/com

/jcertif

/AppTest.java



```
<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.jcertif</groupId>
  <artifactId>projet-simple</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>projet-simple</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>
```

Fichier pom.xml pour
configurer votre projet



Classe App.java

```
package com.jcertif;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Hello World!" );
    }
}
```



Classe AppTest.java

```
package com.jcertif;
```

```
import junit.framework.Test;  
import junit.framework.TestCase;  
import junit.framework.TestSuite;
```

```
public class AppTest extends TestCase {  
    public AppTest( String testName ) { super( testName );}
```

```
    public static Test suite() { return new TestSuite( AppTest.class );}
```

```
    /**
```

```
     * Rigorous Test :-)
```

```
    */
```

```
    public void testApp() {  
        assertTrue( true );  
    }
```

```
}
```

pom.xml

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

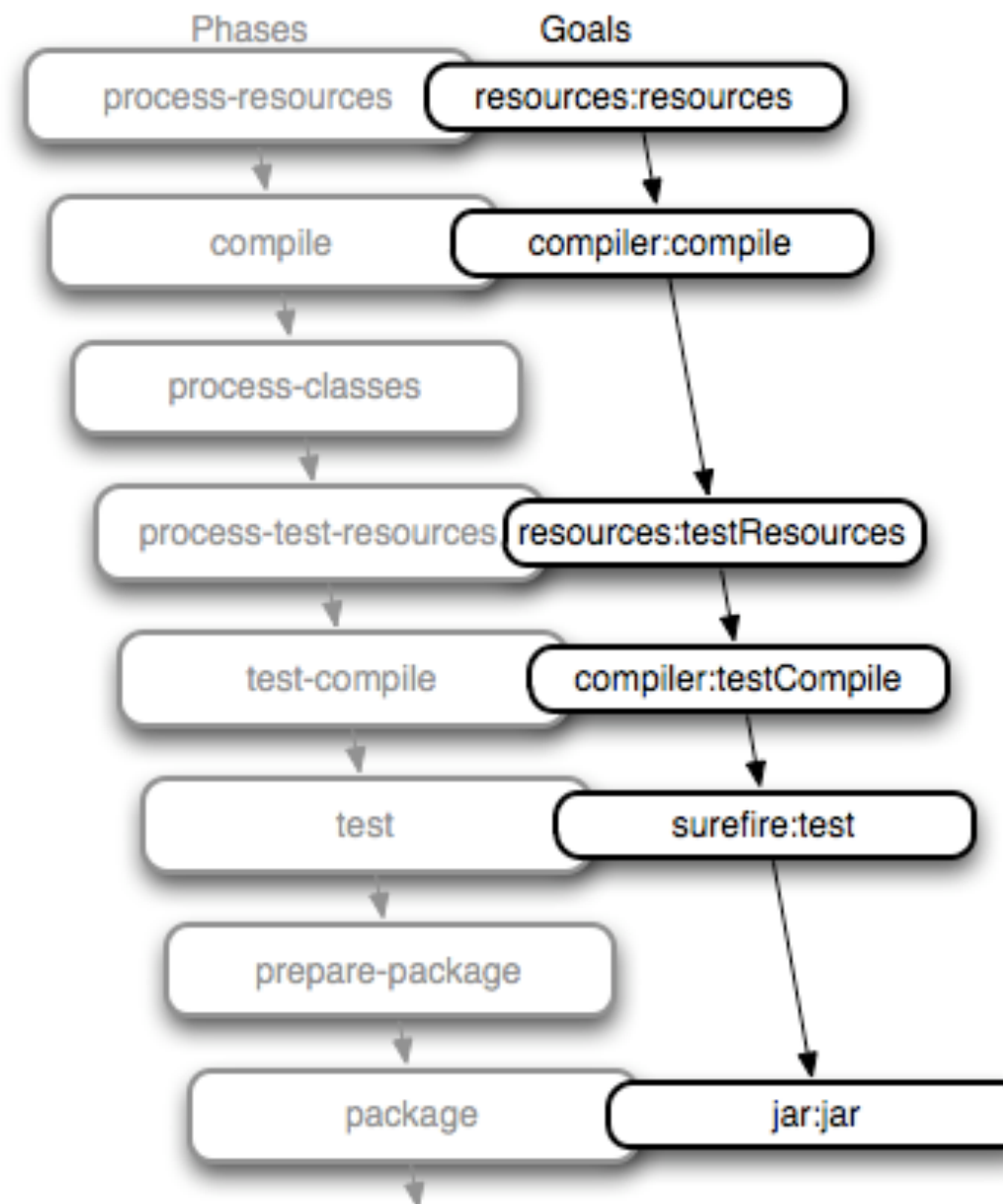


Construisez votre projet

- Se positionner dans le répertoire “mon-projet-simple”
- Exécutez la commande : `mvn install`



Cycle vie - mvn install



Note: There are more phases than shown above, this is a partial list



Fichiers générés par mvn install

```
projet-simple
  /target
    /projet-simple-1.0-SNAPSHOT.jar
  /classes
    /com
      /jcertif
        /App.class
  /maven-archiver
    /pom.properties
  /surefire-reports
    /com.jcertif.AppTest.txt
    /TEST-com.jcertif.AppTest.xml
  /test-classes
    /com
      /jcertif
        /AppTest.class
```



Exécutez l'application


```
java -cp target/projet-simple-1.0-SNAPSHOT.jar com.jcertif.App
```



Modifier la classe App.java

```
package com.jcertif;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        JFrame window = new JFrame();
    }
}
```



Exécutez : mvn install

```
[INFO] -----  
[ERROR] COMPILATION ERROR :  
[INFO] -----  
[ERROR] /dev/jcertif/projet-simple/src/main/java/com/jcertif/App.java:[11,8] cannot find  
symbol  
symbol : class JFrame  
location: class com.jcertif.App  
[ERROR] /dev/jcertif/projet-simple/src/main/java/com/jcertif/App.java:[11,28] cannot find  
symbol  
symbol : class JFrame  
location: class com.jcertif.App  
[INFO] 2 errors  
[INFO] -----  
[INFO] -----  
[INFO] BUILD FAILURE  
[INFO] -----  
[INFO] Total time: 2.072s  
[INFO] Finished at: Sun Aug 19 14:47:03 CEST 2012  
[INFO] Final Memory: 8M/81M  
[INFO] -----
```



Corriger la classe

Les ressources du JDK
sont accessibles sans
ajout de dépendance



```
package com.jcertif;

import javax.swing.JFrame;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        JFrame window = new JFrame();
    }
}
```



Exécutez : mvn install

```
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 3.345s
[INFO] Finished at: Sun Aug 19 15:09:01 CEST 2012
[INFO] Final Memory: 9M/81M
[INFO] -----
```



Modifier la classe AppTest.java

```
package com.jcertif;

import junit.framework.Test;
import junit.framework.TestCase;
import junit.framework.TestSuite;

/**
 * Unit test for simple App.
 */
public class AppTest extends TestCase {
    public AppTest( String testName ) { super( testName );}
    public static Test suite() { return new TestSuite( AppTest.class );}

    /**
     * Rigorous Test :-)
     */
    public void testApp()
    {
        assertTrue( false );
    }
}
```

Mise en échec du test



Exécutez mvn install

```
-----
TESTS
-----
```

Running com.jcertif.AppTest

Tests run: 1, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 0.019 sec <<< FAILURE!

Results :

Failed tests: testApp(com.jcertif.AppTest)

Tests run: 1, Failures: 1, Errors: 0, Skipped: 0

```
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 2.358s
[INFO] Finished at: Sun Aug 19 15:25:19 CEST 2012
[INFO] Final Memory: 9M/81M
[INFO] -----
```



Exécutez

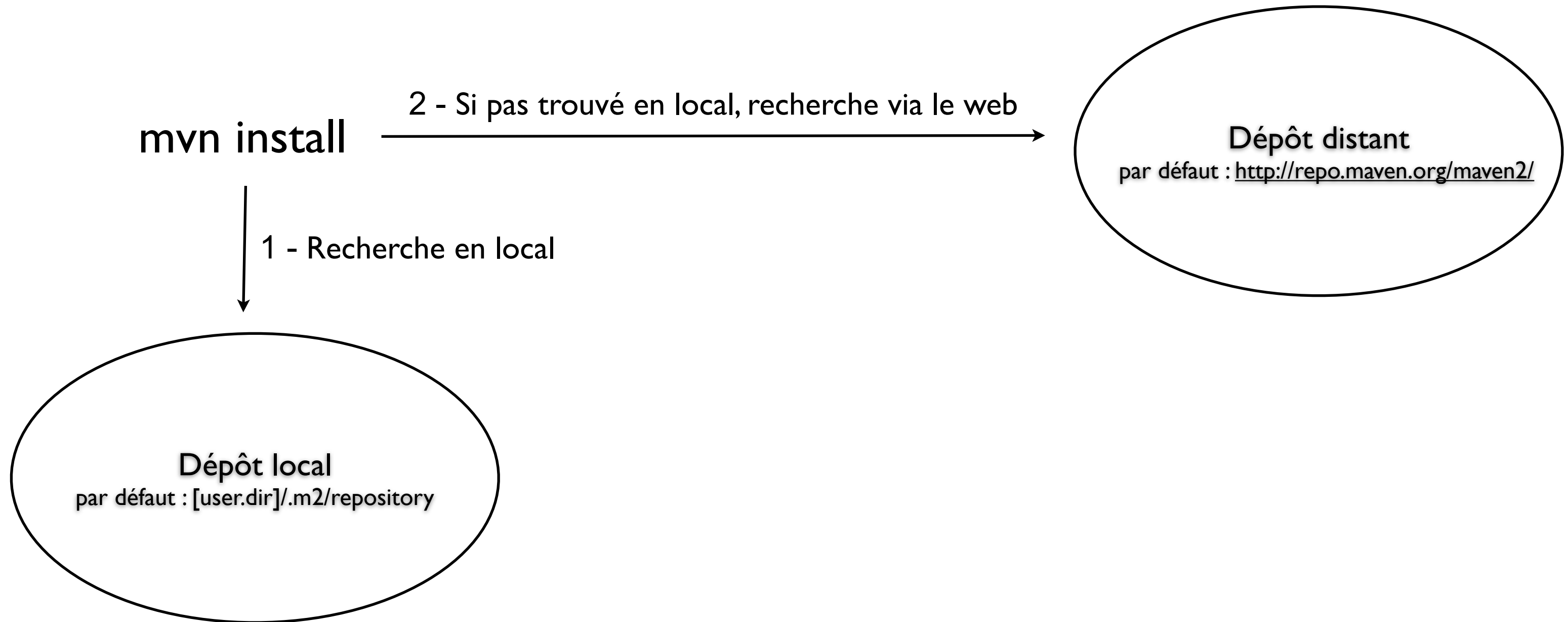
mvn install -Dmaven.test.skip

```
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 1.787s  
[INFO] Finished at: Sun Aug 19 15:27:25 CEST 2012  
[INFO] Final Memory: 5M/81M  
[INFO] -----
```

Pas d'exécution des tests



Où récupérer les dépendances ?

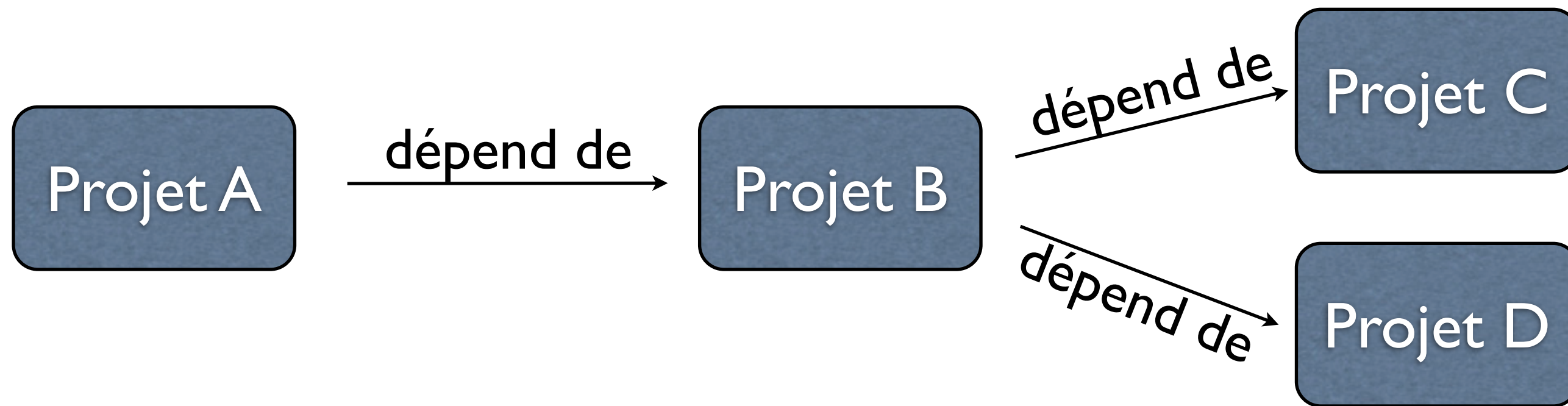


Différents types de dépendance

- **compile (par défaut)** : indispensable à la compilation
- **runtime** : inutile à la compilation mais indispensable à l'exécution
- **test** : utile uniquement à la compilation et l'exécution des tests
- **provided** : indispensable à la compilation, dépendance fournie par l'environnement d'exécution.
- **system** : dépendance à récupérer en local, hors dépôt Maven => à ne pas utiliser dans la mesure du possible



Transitivité des dépendances



=> Le projet A peut utiliser les classes des projets B, C et D



TP 2 : Gérer les dépendances

Objectif :

- utiliser la bibliothèque commons-lang
- visualiser/analyser les dépendances
- exclusion de bibliothèque



Modifier la classe App.java

Utilisation de la classe
RandomStringUtils de la bibliothèque
commons-lang

```
package com.jcertif;

import org.apache.commons.lang.RandomStringUtils;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        String random = RandomStringUtils.randomNumeric(10);
        System.out.println("Chaîne = " + random);
    }
}
```



mvn install

```
[INFO] -----  
[ERROR] COMPILATION ERROR :  
[INFO] -----  
[ERROR] /dev/jcertif/projet-simple/src/main/java/com/jcertif/App.java:[3,25] package  
org.apache.commons does not exist  
[ERROR] /dev/jcertif/projet-simple/src/main/java/com/jcertif/App.java:[13,24] cannot find symbol  
symbol : variable RandomStringUtils  
location: class com.jcertif.App  
[INFO] 2 errors  
[INFO] -----  
[INFO] -----  
[INFO] BUILD FAILURE  
[INFO] -----  
[INFO] Total time: 1.508s  
[INFO] Finished at: Sun Aug 19 17:04:31 CEST 2012  
[INFO] Final Memory: 8M/81M  
[INFO] -----
```



Ajouter la dépendance commons-lang

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>commons-lang</groupId>
    <artifactId>commons-lang</artifactId>
    <version>2.6</version>
  </dependency>
</dependencies>
```



mvn install

```
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 3.771s
[INFO] Finished at: Sun Aug 19 17:33:30 CEST 2012
[INFO] Final Memory: 9M/81M
[INFO] -----
```



Ajouter une dépendance vers Hibernate-core

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>commons-lang</groupId>
    <artifactId>commons-lang</artifactId>
    <version>2.6</version>
  </dependency>
  <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>4.1.2</version>
  </dependency>
</dependencies>
```



mvn dependency:tree

```
[INFO]
[INFO] --- maven-dependency-plugin:2.1:tree (default-cli) @ projet-simple ---
[INFO] com.jcertif:projet-simple:jar:1.0-SNAPSHOT
[INFO] +- junit:junit:jar:3.8.1:test
[INFO] +- commons-lang:commons-lang:jar:2.6:compile
[INFO] \- org.hibernate:hibernate-core:jar:4.1.2:compile
[INFO]     +- antlr:antlr:jar:2.7.7:compile
[INFO]     +- org.jboss.logging:jboss-logging:jar:3.1.0.GA:compile
[INFO]     +- org.jboss.spec.javax.transaction:jboss-transaction-api_1.1_spec:jar:1.0.0.Final:compile
[INFO]     +- dom4j:dom4j:jar:1.6.1:compile
[INFO]     +- org.hibernate.javax.persistence:hibernate-jpa-2.0-api:jar:1.0.1.Final:compile
[INFO]     +- org.javassist:javassist:jar:3.15.0-GA:compile
[INFO]     \- org.hibernate.common:hibernate-commons-annotations:jar:4.0.1.Final:compile
```



Exclure la bibliothèque dom4j

```
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-core</artifactId>
  <version>4.1.2</version>
  <exclusions>
    <exclusion>
      <groupId>dom4j</groupId>
      <artifactId>dom4j</artifactId>
    </exclusion>
  </exclusions>
</dependency>
```



mvn dependency:tree

```
[INFO]
[INFO] --- maven-dependency-plugin:2.1:tree (default-cli) @ projet-simple ---
[INFO] com.jcertif:projet-simple:jar:1.0-SNAPSHOT
[INFO] +- junit:junit:jar:3.8.1:test
[INFO] +- commons-lang:commons-lang:jar:2.6:compile
[INFO] \- org.hibernate:hibernate-core:jar:4.1.2:compile
[INFO]     +- antlr:antlr:jar:2.7.7:compile
[INFO]     +- org.jboss.logging:jboss-logging:jar:3.1.0.GA:compile
[INFO]     +- org.jboss.spec.javax.transaction:jboss-transaction-api_1.1_spec:jar:1.0.0.Final:compile
[INFO]     +- org.hibernate.javax.persistence:hibernate-jpa-2.0-api:jar:1.0.1.Final:compile
[INFO]     +- org.javassist:javassist:jar:3.15.0-GA:compile
[INFO]     \- org.hibernate.common:hibernate-commons-annotations:jar:4.0.1.Final:compile
```

pas de dépendance vers dom4j



mvn dependency:analyze

```
[INFO]
[INFO] <<< maven-dependency-plugin:2.1:analyze (default-cli) @ projet-simple <<<
[INFO]
[INFO] --- maven-dependency-plugin:2.1:analyze (default-cli) @ projet-simple ---
[WARNING] Unused declared dependencies found:
[WARNING]    org.hibernate:hibernate-core:jar:4.1.2:compile
```



TP 3 : Utiliser les plugins Maven

Illustration avec le plugin exec-maven-plugin



Modifier le fichier pom.xml

```
</dependencies>
<build>
  <plugins>
    <plugin>
      <groupId>org.codehaus.mojo</groupId>
      <artifactId>exec-maven-plugin</artifactId>
      <version>1.1</version>
      <configuration>
        <mainClass>com.jcertif.App</mainClass>
      </configuration>
      <executions>
        <execution>
          <phase>integration-test</phase>
          <goals>
            <goal>java</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
</project>
```



mvn install

```
[INFO]
[INFO] >>> exec-maven-plugin:1.1:java (default) @ projet-simple >>>
[INFO]
[INFO] <<< exec-maven-plugin:1.1:java (default) @ projet-simple <<<
[INFO]
[INFO] --- exec-maven-plugin:1.1:java (default) @ projet-simple ---
Chaine = 1840905304
[INFO]
```



TP 4 : Utiliser les ressources

Objectif : créer un fichier de propriétés et l'utiliser dans son application



Créer un fichier de paramétrage

```
projet-simple  
    /src  
        /main  
            /resources  
                /param.properties
```



Ajouter la propriété carac.count

param.properties

carac.count=4



Utiliser la propriété carac.count dans App.java

```
package com.jcertif;

import org.apache.commons.lang.RandomStringUtils;
import java.util.ResourceBundle;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        String caracCount = ResourceBundle.getBundle("params").getString("carac.count");
        String random = RandomStringUtils.randomNumeric(Integer.valueOf(caracCount));
        System.out.println("Chaîne = " + random);
    }
}
```



mvn install

```
[INFO]  
[INFO] >>> exec-maven-plugin:1.1:java (default) @ projet-simple >>>  
[INFO]  
[INFO] <<< exec-maven-plugin:1.1:java (default) @ projet-simple <<<  
[INFO]  
[INFO] --- exec-maven-plugin:1.1:java (default) @ projet-simple ---  
Chaine = 0166
```



Modifier la propriété carac.count

param.properties

carac.count=20



mvn install

```
[INFO]
[INFO] >>> exec-maven-plugin:1.1:java (default) @ projet-simple >>>
[INFO]
[INFO] <<< exec-maven-plugin:1.1:java (default) @ projet-simple <<<
[INFO]
[INFO] --- exec-maven-plugin:1.1:java (default) @ projet-simple ---
Chaine = 92760895216065926312
```



TP 5 : Utiliser des variables

Objectif : variabiliser la construction d'un projet



Compléter le fichier params.properties

param.properties

```
carac.count=4  
projet.nom=${project.artifactId}  
projet.version=${project.version}
```



Modifier la classe App.java

```
package com.jcertif;

import org.apache.commons.lang.RandomStringUtils;
import java.util.ResourceBundle;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args ) {
        String caracCount = ResourceBundle.getBundle("params").getString("carac.count");
        String nomProjet = ResourceBundle.getBundle("params").getString("projet.nom");
        String versionProjet=ResourceBundle.getBundle("params").getString("projet.version");
        System.out.println("#####");
        System.out.println("Nom projet : " + nomProjet);
        System.out.println("Version projet : " + versionProjet);
        System.out.println("#####");
        String random = RandomStringUtils.randomNumeric(Integer.valueOf(caracCount));
        System.out.println("Chaine = " + random);
    }
}
```



mvn install

```
INFO]
[INFO] >>> exec-maven-plugin:1.1:java (default) @ projet-simple >>>
[INFO]
[INFO] <<< exec-maven-plugin:1.1:java (default) @ projet-simple <<<
[INFO]
[INFO] --- exec-maven-plugin:1.1:java (default) @ projet-simple ---
#####
Nom projet : ${project.artifactId}
Version projet : ${project.version}
#####
Chaine = 53577921925530242406
```



Activer la mise à jour des variables dans les ressources

```
</dependencies>
<build>
  <resources>
    <resource>
      <directory>src/main/resources</directory>
      <filtering>true</filtering>
    </resource>
  </resources>
</build>
<plugins>
  <plugin>
    <groupId>org.codehaus.mojo</groupId>
```



mvn install

```
[INFO]
[INFO] >>> exec-maven-plugin:1.1:java (default) @ projet-simple >>>
[INFO]
[INFO] <<< exec-maven-plugin:1.1:java (default) @ projet-simple <<<
[INFO]
[INFO] --- exec-maven-plugin:1.1:java (default) @ projet-simple ---
#####
Nom projet : projet-simple
Version projet : 1.0-SNAPSHOT
#####
Chaine = 78559487063744308397
```



Créer la propriété carac.count dans le fichier pom.xml

```
</dependencies>
<properties>
  <carac.count>13</carac.count>
</properties>
<build>
  <resources>
```



Utiliser la variable `${carac.count}`

`param.properties`

```
carac.count=${carac.count}  
projet.nom=${project.artifactId}  
projet.version=${project.version}
```



mvn install

```
[INFO]
[INFO] >>> exec-maven-plugin:1.1:java (default) @ projet-simple >>>
[INFO]
[INFO] <<< exec-maven-plugin:1.1:java (default) @ projet-simple <<<
[INFO]
[INFO] --- exec-maven-plugin:1.1:java (default) @ projet-simple ---
#####
Nom projet : projet-simple
Version projet : 1.0-SNAPSHOT
#####
Chaine = 1778566838937
```



TP 6 : Utiliser les profils

Exécuter l'application uniquement dans un profil



Supprimer l'utilisation du plugin exec-maven-plugin

```
<build>  
  <resources>  
    <resource>  
      <directory>src/main/resources</directory>  
      <filtering>true</filtering>  
    </resource>  
  </resources>  
</build>
```



Créer le profil “exec-appli”

```
<profiles>
  <profile>
    <id>exec-appli</id>
    <build>
      <plugins>
        <plugin>
          <groupId>org.codehaus.mojo</groupId>
          <artifactId>exec-maven-plugin</artifactId>
          <version>1.1</version>
          <configuration>
            <mainClass>com.jcertif.App</mainClass>
          </configuration>
          <executions>
            <execution>
              <phase>integration-test</phase>
              <goals>
                <goal>java</goal>
              </goals>
            </execution>
          </executions>
        </plugin>
      </plugins>
    </build>
  </profile>
</profiles>
```



mvn install

=> Pas d'exécution de l'application



mvn install -Pexec-appli

```
[INFO]
[INFO] >>> exec-maven-plugin:1.1:java (default) @ projet-simple >>>
[INFO]
[INFO] <<< exec-maven-plugin:1.1:java (default) @ projet-simple <<<
[INFO]
[INFO] --- exec-maven-plugin:1.1:java (default) @ projet-simple ---
#####
Nom projet : projet-simple
Version projet : 1.0-SNAPSHOT
#####
Chaine = 6180915411835
```



Une valeur de propriété par profil

```
<profiles>
  <profile>
    <id>exec-appli</id>
    <properties>
      <carac.count>2</carac.count>
    </properties>
    <build>
      <plugins>
        <plugin>
          <groupId>org.codehaus.mojo</groupId>
```



mvn install -Pexec-appli

```
[INFO]
[INFO] >>> exec-maven-plugin:1.1:java (default) @ projet-simple >>>
[INFO]
[INFO] <<< exec-maven-plugin:1.1:java (default) @ projet-simple <<<
[INFO]
[INFO] --- exec-maven-plugin:1.1:java (default) @ projet-simple ---
#####
Nom projet : projet-simple
Version projet : 1.0-SNAPSHOT
#####
Chaine = 70
```



Activer un profil par défaut

```
<profiles>
  <profile>
    <id>exec-appli</id>
    <activation>
      <activeByDefault>true</activeByDefault>
    </activation>
    <properties>
      <carac.count>2</carac.count>
    </properties>
    <build>
      <plugins>
```



mvn install

```
[INFO]
[INFO] >>> exec-maven-plugin:1.1:java (default) @ projet-simple >>>
[INFO]
[INFO] <<< exec-maven-plugin:1.1:java (default) @ projet-simple <<<
[INFO]
[INFO] --- exec-maven-plugin:1.1:java (default) @ projet-simple ---
#####
Nom projet : projet-simple
Version projet : 1.0-SNAPSHOT
#####
Chaine = 10
```



TP 4 : Projet multimodules

Objectif : créer un ensemble de projet interdépendant



Créer un projet “parent”

```
mvn archetype:generate -DgroupId=com.jcertif -DartifactId=projet-parent -Dversion=1.0.0 -  
DinteractiveMode=false
```



Supprimer le répertoire /src et Modifier le 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.jcertif</groupId>
  <artifactId>projet-parent</artifactId>
  <packaging>pom</packaging>
  <version>1.0.0</version>
  <name>projet-parent</name>
  <url>http://maven.apache.org</url>

</project>
```



Créer deux projets dans le répertoire projet-parent

```
mvn archetype:generate -DgroupId=com.jcertif -DartifactId=projet-simple-dao -Dversion=1.0.0 -DinteractiveMode=false
```

```
mvn archetype:generate -DgroupId=com.jcertif -DartifactId=projet-simple-service -Dversion=1.0.0 -DinteractiveMode=false
```



projet-parent : pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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.jcertif</groupId>
  <artifactId>projet-parent</artifactId>
  <packaging>pom</packaging>
  <version>1.0.0</version>
  <name>projet-parent</name>
  <url>http://maven.apache.org</url>

  <modules>
    <module>projet-simple-dao</module>
    <module>projet-simple-service</module>
  </modules>
</project>
```



projet-dao : pom.xml

```
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd" xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <artifactId>projet-parent</artifactId>
    <groupId>com.jcertif</groupId>
    <version>1.0.0</version>
  </parent>
  <groupId>com.jcertif</groupId>
  <artifactId>projet-simple-dao</artifactId>
  <version>1.0.0</version>
  <name>projet-simple-dao</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>
```



projet-parent : mvn install

```
INFO] -----  
[INFO] Reactor Summary:  
[INFO]  
[INFO] projet-parent ..... SUCCESS [0.515s]  
[INFO] projet-simple-dao ..... SUCCESS [3.455s]  
[INFO] projet-simple-service ..... SUCCESS [0.564s]  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 4.669s  
[INFO] Finished at: Sun Aug 19 22:24:40 CEST 2012  
[INFO] Final Memory: 9M/81M  
[INFO] -----
```



projet-simple-dao : mvn install

```
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.074s
[INFO] Finished at: Sun Aug 19 22:26:40 CEST 2012
[INFO] Final Memory: 4M/81M
[INFO] -----
```



“Management” des dépendances dans projet-parent:pom.xml

```
<modules>
  <module>projet-simple-dao</module>
  <module>projet-simple-service</module>
</modules>

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



Plus besoin de préciser la version dans les modules

```
<groupId>com.jcertif</groupId>
<artifactId>projet-simple-dao</artifactId>
<version>1.0.0</version>
<name>projet-simple-dao</name>
<url>http://maven.apache.org</url>
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
  </dependency>
</dependencies>
```



mvn install

```
[INFO] -----
[INFO] Reactor Summary:
[INFO]
[INFO] projet-parent ..... SUCCESS [0.409s]
[INFO] projet-simple-dao ..... SUCCESS [1.466s]
[INFO] projet-simple-service ..... SUCCESS [0.529s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.544s
[INFO] Finished at: Sun Aug 19 22:34:32 CEST 2012
[INFO] Final Memory: 4M/81M
[INFO] -----
```



“Management” des plugins dans projet-parent : pom.xml

```
<build>
  <pluginManagement>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>2.1</version>
        <configuration>
          <source>1.6</source>
          <target>1.6</target>
          <encoding>UTF-8</encoding>
        </configuration>
      </plugin>
      <plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>exec-maven-plugin</artifactId>
        <version>1.1</version>
      </plugin>
    </plugins>
  </pluginManagement>
</build>
```



Utilisation d'un plugin dans le module projet-simple-dao

la version du plugin
n'est plus
indispensable

```
<build>
  <plugins>
    <plugin>
      <groupId>org.codehaus.mojo</groupId>
      <artifactId>exec-maven-plugin</artifactId>
      <configuration>
        <mainClass>com.jcertif.App</mainClass>
      </configuration>
      <executions>
        <execution>
          <phase>integration-test</phase>
          <goals>
            <goal>java</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```



projet-parent : mvn install

```
[INFO]
[INFO] >>> exec-maven-plugin:1.1:java (default) @ projet-simple-dao >>>
[INFO]
[INFO] <<< exec-maven-plugin:1.1:java (default) @ projet-simple-dao <<<
[INFO]
[INFO] --- exec-maven-plugin:1.1:java (default) @ projet-simple-dao ---
Hello World!
```



Maven & les IDE



NetBeans



IntelliJIDEA



Générer le site de votre projet

Exécutez la commande : `mvn site`



Ouvrer le fichier target/index.html

avec un navigateur

projet-parent

Last Published: 2012-08-19 | Version: 1.0.0

Modules

[projet-simple-dao](#)

[projet-simple-service](#)

Project Documentation

▼ Project Information

[About](#)

[Plugin Management](#)

[Distribution Management](#)

[Dependency Information](#)

[Dependency Convergence](#)

[Source Repository](#)

[Mailing Lists](#)

[Issue Tracking](#)

[Continuous Integration](#)

[Project Plugins](#)

[Project License](#)

[Project Modules](#)


[Dependency Management](#)

[Project Team](#)

[Project Summary](#)

[Dependencies](#)

Built by:



Project Information

This document provides an overview of the various documents and links that are part of this project's automatically generated by [Maven](#) on behalf of the project.

Overview

Document	Description
About	There is currently no description associated with this project.
Plugin Management	This document lists the plugins that are defined through pluginManagement.
Distribution Management	This document provides informations on the distribution management of this p
Dependency Information	report.dependency-info.description
Dependency Convergence	This document presents the convergence of dependency versions across the e
Source Repository	This is a link to the online source repository that can be viewed via a web bro
Mailing Lists	This document provides subscription and archive information for this project's
Issue Tracking	This is a link to the issue management system for this project. Issues (bugs, f queried using this link.
Continuous Integration	This is a link to the definitions of all continuous integration processes that buil
Project Plugins	This document lists the build plugins and the report plugins used by this projec
Project License	This is a link to the definitions of project licenses.
Project Modules	This document lists the modules (sub-projects) of this project.
Dependency Management	This document lists the dependencies that are defined through dependencyMar



Merci !

