Maven Cheatsheet

Basic operations with maven you need during a work day

Klaus Franz Eckenfellner
eckobar@gmail.com
http://eckobar.wordpress.com

Version 0.2.0

Tasks

create project

create java project

```
mvn archetype:create -DgroupId=org.yourcompany.project -DartifactId=application
```

create web project

```
mvn archetype:create -DgroupId=org.yourcompany.project -DartifactId=application
  -DarchetypeArtifactId=maven-archetype-webapp
```

basic project tasks

clean project: will delete target directory

```
mvn clean
```

validate project: validate the project is correct and all necessary information is available

```
mvn validate
```

compile project: compile source code, classes stored in target/classes

```
mvn compile
```

test project: run tests using a suitable unit testing framework

```
mvn test
```

package project: take the compiled code and package it in its distributable format, such as a JAR / WAR

```
mvn package
```

verify project: run any checks to verify the package is valid and meets quality criteria

```
mvn verify
```

install project: install the package into the local repository, for use as a dependency in other projects locally

```
mvn install
```

deploy project: done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects

```
mvn deploy
```

deploy-file: can be used for deploying a external jar file to repository

```
mvn deploy:deploy-file -Dfile=/path/to/jar/file -DrepositoryId=repos-server -Durl=http
  ://repos.company.org/test -DgroupId=javax -DartifactId=mail -Dpackaging=jar
  -Dversion=1.0.1
```

eclipse integration

create metainformation: meta files for eclipse are created, can be used for project import

```
mvn eclipse:eclipse
```

create metainformation with wtp: same like create metainformation + WTP plugin infos

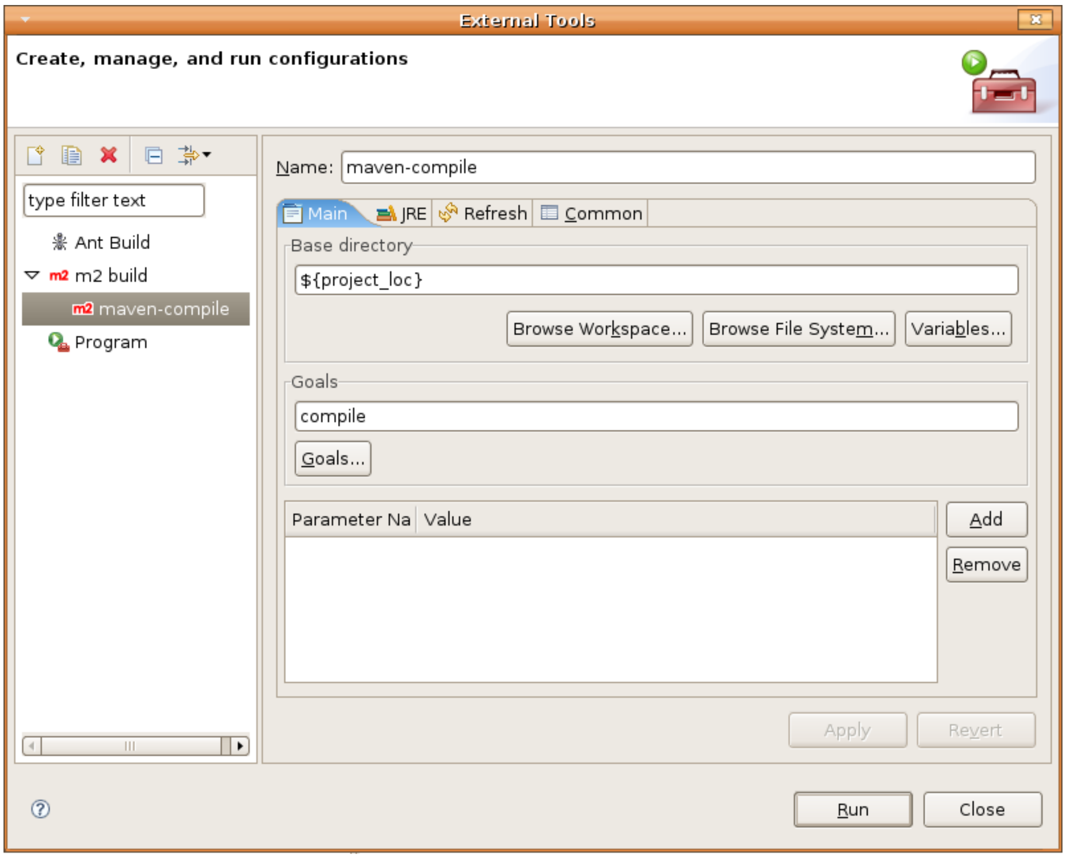
```
mvn -Dwtpversion=1.5 eclipse:eclipse
```

tell eclipse where local repository is located

```
mvn -Declipse.workspace=/path/to/workspace eclipse:add-maven-repo
```

run maven tasks in maven (maven eclipse plugin must be installed)

```
add maven as external tool – in the dialog there should be a category called 'm2 build'
 – create new and enter informations – you need to specify every goal seperate.
```



Name: maven-compile

Base directory: ${project_loc}

Goals: compile

Illustration 1: defined maven goals in eclipse

release project

prepare release: informations about versions number are collected

```
mvn release:prepare
```

clean release: rollback to snapshot versions

```
mvn release:clean
```

perform release: deploy project to remote repository and make tag in version control system. username and password for version control system are taken from server informations in ~/.m2/settings.xml. serverID is same like defined in deploymentServer ... this behaviour is not whished

```
mvn release:perform
```

perform release with username and password for authentication on version control system

```
mvn release:perform -Dusername=foo -Dpassword=bar
```

webproject special tasks

create war file, same like mvn package

```
mvn war:war
```

Build an exploded web application into ${maven.war.src}. This allows you to mount it in your application server, and you only need to run it again for dependency and class changes, not JSP changes. This goal will not clean old dependencies - due to the dangers involved in automating this for your source tree, you must do that yourself.

```
mvn war:inplace
```

delete all artifacts created by war plugin

```
mvn war:clean
```

tomcat integration

deploy webproject to tomcat

```
mvn tomcat:deploy
```

redeploy webproject, didn't worked in my test environment

```
mvn tomcat:redeploy
```

undeploy webproject

```
mvn tomcat:undeploy
```

stop context on tomcat

```
mvn tomcat:stop
```

start context on tomcat

```
mvn tomcat:start
```

Configuration

example 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>org.yourcompany.project</groupId>
      <artifactId>application</artifactId>
      <packaging>pom</packaging>
      <version>1.0-SNAPSHOT</version>
      <name>application</name>
      <repositories>
            <repository>
                  <id>repo-server</id>
                  <url>http://192.168.1.147/mvn-repos/sun/</url>
                  <snapshots>
                        <enabled>true</enabled>
                  </snapshots>
            </repository>
            <repository>
                  <id>repo-server</id>
                  <url>http://192.168.1.147/mvn-repos/apache/</url>
                  <snapshots>
                        <enabled>true</enabled>
                  </snapshots>
            </repository>
            <repository>
                  <id>repo-server</id>
                  <url>http://192.168.1.147/mvn-repos/plotcommons/</url>
                  <snapshots>
                        <enabled>true</enabled>
                  </snapshots>
            </repository>
      </repositories>
      <distributionManagement>
            <repository>
                  <id>repo-server</id>
                  <url>scp://192.168.1.147/home/www/mvn-repos/test</url>
            </repository>
      </distributionManagement>
      <scm>
            <connection>scm:svn:http://192.168.1.147
/svn/project/trunk/applications/test</connection>

      </scm>
      <modules>
            <module>project-utils</module>
            <module>project-gui</module>
      </modules>
      <dependencies>
            <dependency>
                  <groupId>junit</groupId>
                  <artifactId>junit</artifactId>
                  <version>3.8.1</version>
                  <scope>test</scope>
            </dependency>
      </dependencies>
      <build>
        <plugins>
                <plugin>
                        <groupId>org.apache.maven.plugins</groupId>
                        <artifactId>maven-compiler-plugin</artifactId>
                        <configuration>
                                <source>1.5</source>
                                <target>1.5</target>
                        </configuration>
                </plugin>
        </plugins>
  </build>

</project>
```

settings.xml

```
<settings>
      <localRepository>/home/klaecken/.m2/repository/</localRepository>
      <servers>
            <server>
                  <id>repo-server</id>
                  <username>deployer</username>
                  <privateKey>/home/klaecken/.ssh/id_dsa</privateKey>
                  <directoryPermissions>770</directoryPermissions>
            </server>
            <server>
                  <id>local-tomcat</id>
                  <username>admin</username>
                  <password>admin</password>
            </server>
      </servers>
</settings>
```

Special Hints

filepermissions

in the ~/.m2/settings.xml it is possible to set a filePermissions value which defines which permissions to set when deploying to remote repositories. the wagon-ssh-provider doesn't set the permissions correct, the best workaround is to define no filepermissions, then the default 644 is set, otherwise the value is randomly.

for during goal release

the goal release makes a process fork of you running shell, therefore be carefull to set JAVA_HOME and binaries mvn & java & javac in you PATH, otherwise you will get a “mvn command not found” - Error. the environment variables JAVA_HOME and PATH(including mvn / java / javac) must be set for the whole system, setting it in you current shell is not sufficient.

Weblinks

http://maven.apache.org/

Weblinks

http://maven.apache.org/continuum/
http://mojo.codehaus.org/