

## Erstes Projekt mit Maven

Zunächst muss man sichergehen das Maven auf dem PC installiert ist, man downloadt eine zip, dann entzippen und dann den bin Ordner in den Pfad verschieben.

Man kann die Installation überprüfen, mit der Kommandozeile: `mvn -v`

Hat es funktioniert dann werden einige Informationen zur OS Version, zum Java home Verzeichnis und zum Maven Home Verzeichnis.

Danach erstellt man eine pom.xml Datei. In ihr sind Informationen, wie der Projektname, die Version und Abhängigkeiten zu externen Bibliotheken unter anderem definiert.

Mit dem Befehl: `mvn compile` startet man Maven und dieses führt den Kompilierungsbefehl aus.

Die kompilierten .class Dateien findet man im target/classes Verzeichnis.

Mit dem Befehl: `mvn package` wird der Java Code kompiliert, tests durchgeführt und der Code wird in eine .JAR Datei verpackt. Der Name der Datei wird in der pom.xml im <artifactId> tag definiert.

Maven stellt einen Ablageort für Abhängigkeiten bereit, normalerweise heißt dieser (m2/repository) im Home Verzeichnis. Mit dem Befehl: `mvn install` kompiliert, testet und verpackt den Code und kopiert dieses Paket dann in das Verzeichnis. Somit kann dieses in einem anderen Projekt referenziert werden.

### Eigene Abhängigkeiten deklarieren

Will man auf externe libraries zugreifen, so definiert man erst welche und importiert diese in das eigene Projekt. Beim kompilieren würde jetzt erstmal ein Fehler erscheinen, man muss zunächst noch in der pom.xml die Abhängigkeit eintragen. Am besten macht man dies mit jeder neuen dependency ganz unten in der File kurz bevor das <project> geschlossen wird. Folgendermaßen wird eine Abhängigkeit eingetragen.

```
<dependencies>
  <dependency>
    <groupId>joda-time</groupId> //Die Gruppe der die Abhängigkeit gehört
    <artifactId>joda-time</artifactId> //Die benötigte Bibliothek
    <version>2.2</version> //Die Version der Bibliothek benötigt wird
    <scope>provided || test</scope> // 'provided', dabei werden benötigte dependencies durch einen Container bereitgestellt. 'test', Abhängigkeiten, die bei kompilieren und beim testing genutzt werden, aber nicht beim bauen oder ausführen des runtime code.
  </dependency>
</dependencies>
```

### Einen Test machen

Wir fügen eine GreeterTest Klasse hinzu und eine weitere dependency.

`mvn test` damit wird dann der Test durchgeführt