3ÜbunGszettel MAVEN

**Allgemeine Anmerkungen**:

* Nicht vergessen alle Aufgaben so zu dokumentieren, dass alles gut nachvollziehbar ist (Notizen, Screenshots, ...)
* Teilweise wurden bewusst ältere Materialien verwendet, weil diese sehr verständlich angelegt sind. Bitte aber bei der verwendeten Software (JDK, Maven, Maven plug-ins, Maven archetypes, Maven dependencies, ...) immer die aktuellsten Versionen verwenden (außer es gibt echte Inkompatibilitäten)!

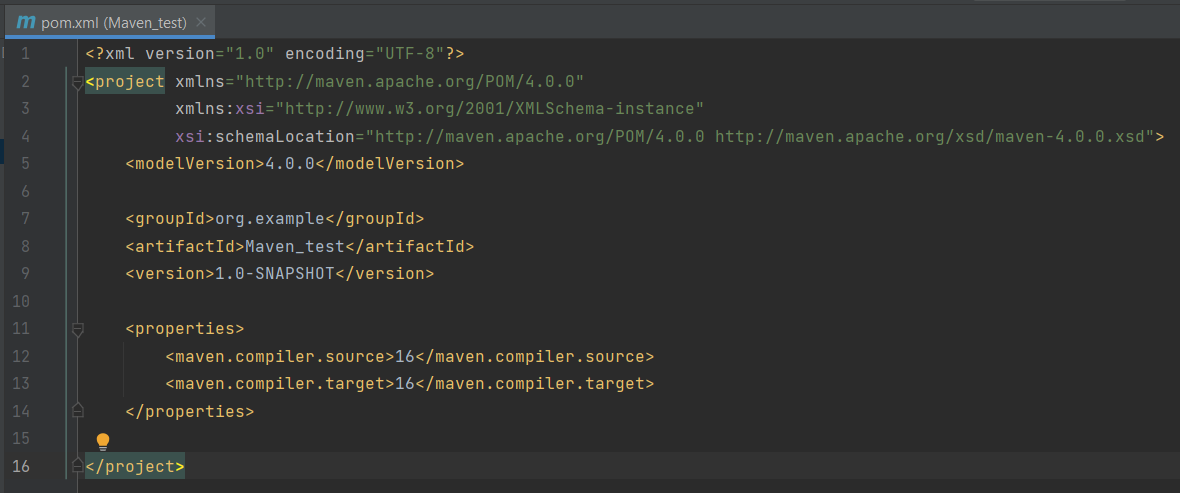
# Aufgabe 1: Maven in 5 Minutes

Mache das Tutorial <https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html> durch (alles ausprobieren, Sachen kreativ variieren etc.)

Maven ist bei mir bei IntelliJ dabei, habe es nicht separat installiert. Daher funktionieren die „mvn“-Befehle nicht. Habe Maven dann separat installiert, dann hat nichts mehr funktioniert…

# Aufgabe 2: IntelliJ und Maven

Lerne, wie Maven in IntelliJ verwendet wird: <https://www.youtube.com/watch?v=pt3uB0sd5kY>



Ein Bild, das Text, Screenshot, Bildschirm enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Über IntelliJ ein Standard-Maven-Programm angelegt…

# Aufgabe 3: Apache Maven Tutorial: Einführung in Build-Prozess & Abhängigkeiten für Anfänger ohne Vorkenntnisse

Vertiefe dein Wissen rund um Maven und verwende Maven wie im folgenden Tutorial (alles am eigenen Rechner mitmachen, weitere Dependencies reinholen, verschiedene Maven-Goals ausführen, etc.):

<https://youtu.be/ExKq23bNABk>

Maven löst 2 Probleme:

* Dependency Management (benötigte Dependencies und Plugins werden automatisch heruntergeladen)
* Reproduzierbarer Build (unabhängig von der Entwicklungsumgebung)

In pom.xml (Project Object Modell) wird das Maven-Projekt konfiguriert.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Jede Phase führt auch die Phasen darüber aus. Wenn „test“ ausgewählt wird, wird zuerst validiert, dann kompiliert und zu Schluss getestet.

|  |  |
| --- | --- |
| **Phase** | **Funktion** |
| validate | validiert, ob Projekt korrekt ist und ob alle notwendigen Informationen vorhanden sind |
| compile | kompiliert den Code des Projekts |
| test | testet den kompilierten Code mit einem passenden Test-Framework. |
| package | nimmt den kompilierten Code und packt ihn in ein verteilbares Format (z.B. jar) |
| verify | führt Checks von Integrations-Tests durch, um zu versichern, dass die Qualitätskriterien erfüllt werden. |
| install | installiert das Paket auf das lokale Repo, um das Repo als Dependency in anderen Projekten verwenden zu können. |
| deploy | kopiert lokales Repo auf Remote-Repo, damit es andere auch verwenden können |

# Aufgabe 4: Java FX, Easiest Setup, IntelliJ + Maven

Schaue dir am Beispiel JavaFX 12 praktisch an, wie schnell man mit Maven Dependencies für ein IntelliJ-Projekt managen kann:

<https://www.youtube.com/watch?v=4vd-RE0X5Lg>

Video angeschaut. Bei CryptoWallet und kleineren eigenen Projekten schon mal gemacht.

**Alternativ** dazu kannst du folgende Anleitung befolgen, um dein erstes IntelliJ-Projekt für JavaFX 12 mit Maven zu erstellen:

1. JDK 14 installieren
2. JDK 14\bin-Ordner im Systempfad verfügbar machen (javac -version auf der Kommandozeile muss JDK 14 anzeigen)
3. JAVA\_HOME-Variable auf JKD 14 einstellen (echo %JAVA\_HOME% muss auf JDK14-Installationsordner zeigen)
4. Folgende Anleitung durchmachen: <https://openjfx.io/openjfx-docs/#IDE-Intellij>
5. Gestartet wird das Programm dann mit folgenden MavenGoals (IntelliJ kann eine eigene Built-In Maven-Installation verwenden, oder man kann auch eine eigene auf dem System installierte angeben 🡪 siehe File 🡪 Settings 🡪 Build,Execution,Deployment 🡪 BuildTools🡪 Maven):
   1. Plugins 🡪 Compiler 🡪 compiler:compile
   2. Plugins 🡪 JavaFX 🡪 javafx:run
6. Oder alternativ auf der Kommandozeile mit Maven direkt: mvn clean javafx:run