Einführung in

Jan Speckamp

Inhalt

- Automation Build Tools
- Situation in Java
- Maven
  - Aufbau
  - Lifecycle
- Alternativen
- Interaktive Demo

Hinweis: Maven lernt man nicht durch Folien

Automation Build Tool

- Automatisierung von Vorgängen ist weit verbreitetes Konzept
  - Produktivitätssteigerung
  - Qualitätssicherung
  - Reduzierung des Bus-Faktors
  - Geld & Zeitersparnis

→ Trifft auch auf Softwareentwicklung zu

- Ziele
  - Wiederholbarkeit/Reproduzierbarkeit
  - Kombination von zahlreichen Arbeitsschritten zu einem Gesamtpaket
  - Einfach zu betreiben, gut dokumentiert

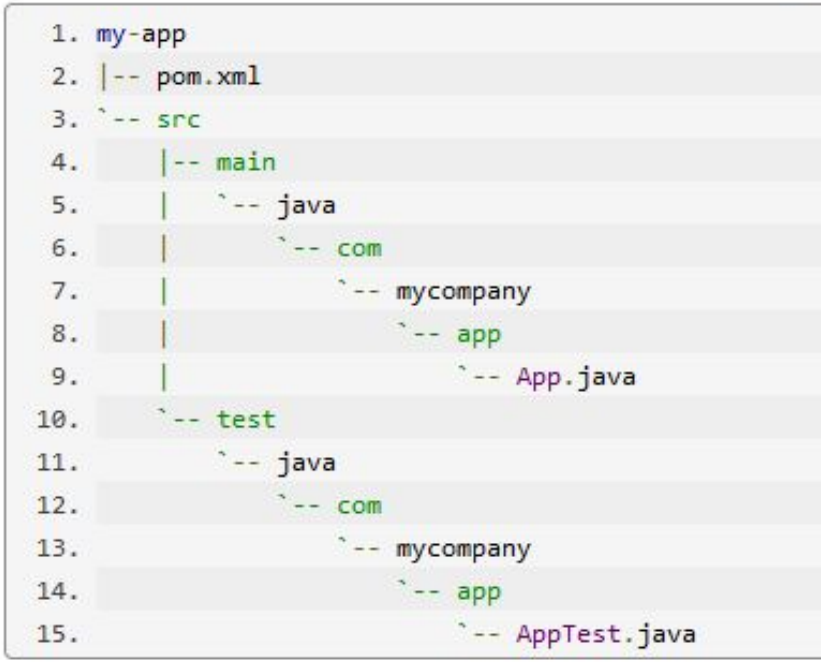
Situation in Java

- Große Software mit vielen Klassen
- Zahlreiche Abhängigkeiten (Dependencies)
  - Verschiedene Quellen
  - Verschiedene Versionen
  - Abhängigkeiten haben Abhängigkeiten (die im Konflikt mit eigenen Abhängigkeiten stehen)
- Tests
- Zahlreiche verschiedene Tasks müssen erledigt werden
  - Dokumentation generieren
  - Lizenzenprüfung
  - Codestilprüfung
  - Automatisierte Bugsuche
  - Kompilieren
  - Paketieren
  - etc.

Maven

- Automation Build Tool
- Initiiert in 2004, mittlerweile in Version 3.x
- Build Tool
  - Ausführen von Tests
  - Generierung von Code
  - Paketieren der Software als .jar, .war, etc.
- Dependency Management Tool
  - Automatisches Herunterladen/Einbinden von Abhängigkeiten
- Documentation Tool
  - Automatisches Generieren von Dokumentation
  - Dokumentiert alle Abhängigkeiten
- Sehr gute Integration in IDEs

Aufbau eines Maven Projektes



```
my-app
|-- pom.xml
`-- src
    |-- main
    |   `-- java
    |       `-- com
    |           `-- mycompany
    |               `-- app
    |                   `-- App.java
    `-- test
        `-- java
            `-- com
                `-- mycompany
                    `-- app
                        `-- AppTest.java
```

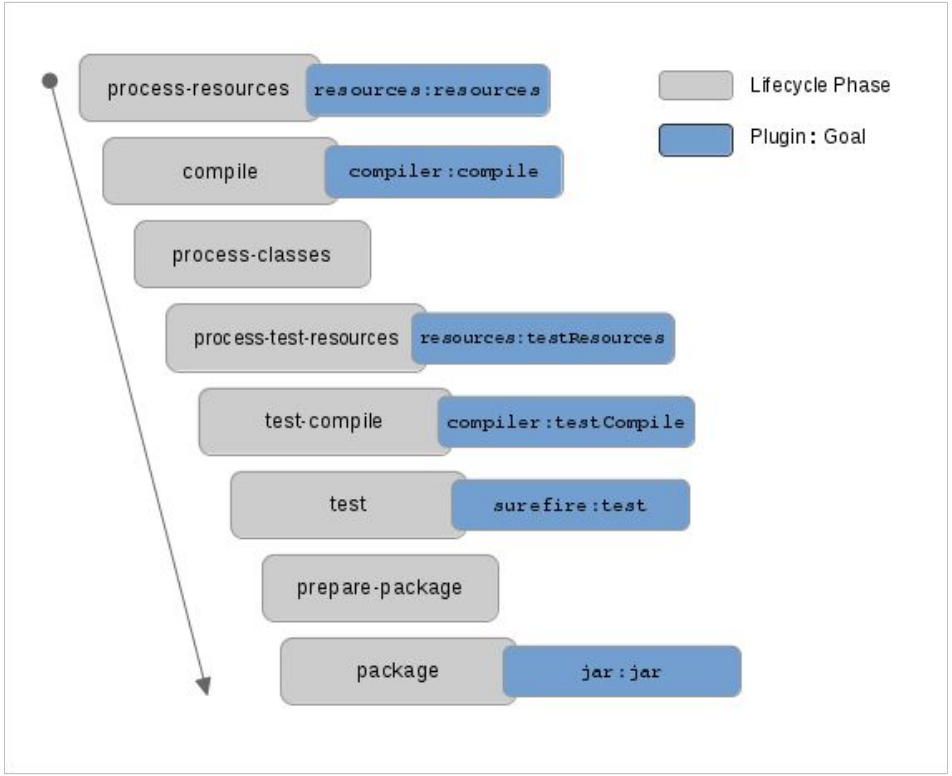
pom.xml

- Zentrales Dokument im obersten Verzeichnis (root)
- Definiert das gesamte Projekt
  - Name
  - Autoren
  - Versionen
  - Abhängigkeiten
  - Lizenzinformationen
  - Build-Prozess
  - etc.

```
<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/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>org.notgroupb</groupId>
        <artifactId>workerHygon</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>Worker of Hygon data</name>

        <organization>
                <name>not-group-b</name>
        </organization>

        <description>Indicates flooding for Hygon Data</description>

        <properties>
                <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
                <kafka.version>1.1.0</kafka.version>
                <slf4j.version>1.7.7</slf4j.version>
                <log4j.version>1.2.17</log4j.version>
        </properties>
```

Maven Lifecycle

- Bau der Software is linearer Prozess
- Verschiedene Phasen werden nacheinander ausgeführt
- Plugins können sich in die Phasen “einklinken” und eigene Arbeit ausführen
  - Lizenzen prüfen
  - Codestil prüfen
  - etc.



Quelle: https://www.codetab.org/tutorial/apache-maven/plugins/maven-lifecycle-goals/

Alternativen

Interaktive Demo

1. Installation von Maven
2. Erzeugen eines Maven Projektes
    a. mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.4
3. Bauen des Projektes
4. Starten der Applikation