

Toolunterstützung

Es gibt verschiedenste Tools zur Unterstützung der Konfigurationsprozesse.

Hier werden einige Open Source Tools vorgestellt, die weit verbreitet sind:

- Subversion, GIT Versionskontrolle
- Maven Build Prozess
- Hudson Continuous Integration
- Redmine Kollaboration



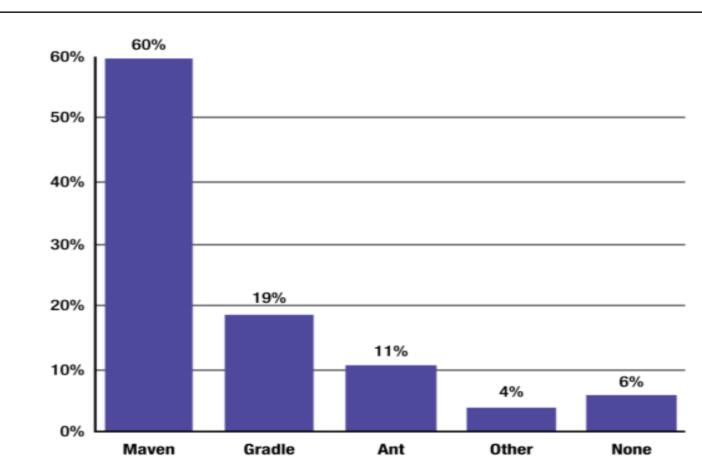
Maven Quellen

- Maven Apache Projekt Seite <u>https://maven.apache.org/</u>
- Maven by Example <u>https://books.sonatype.com/mvnex-book/pdf/mvnex-pdf.pdf</u>
- Maven The Complete Reference <u>https://books.sonatype.com/mvnref-book/pdf/mvnref-pdf.pdf</u>
- Maven Guides: http://maven.apache.org/guides/
- G. Popp: Konfiguration Management, d.punkt Verlag,
 4. Auflage, 2013
- B. Varanasi, Introducing Maven, Second Edition, Apress, Salt Lake City, 2019



V

Vergleich der Verbreitung von build Tools



Quelle: B. Varanasi, Introducing Maven, Second Edition, Apress, Salt Lake City, 2019



Maven

- Ebenfalls (wie svn) ein Top Level Projekt der Apache Software Foundation (https://maven.apache.org/)
- Aktuelle Release Version (April 2022): 3.8.5
- Idee: Modellbasierter, deklarativer Ansatz zur Buildautomatisierung
- Maven in 5 min: <u>http://maven.apache.org/guides/getting-</u> started/maven-in-five-minutes.html



Maven, was ist das?

Quelle: Maven by Example:

Maven is a project management tool which encompasses a **project object model**, a set of **standards**, a **project lifecycle**, a **dependency management** system, and logic for executing plugin goals at defined phases in a lifecycle. When you use Maven, you **describe** your project using a well-defined project object model, Maven can then apply cross-cutting logic from a set of shared (or custom) plugins.



Maven, was ist das?

Aus http://maven.apache.org/guides/getting-started/index.html

In a nutshell Maven is an attempt to apply **patterns** to a project's build infrastructure in order to promote comprehension and productivity by providing a clear path in the use of **best practices**.



Maven, was ist das?

Aus http://maven.apache.org/guides/getting-started/index.html

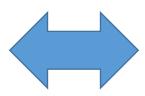
Maven is essentially a project management and comprehension tool and as such provides a way to help with managing:

- Builds
- Documentation
- Reporting
- Dependencies
- SCMs
- Releases
- Distribution



Ant vs Maven

Ant: Toolbox



Maven: Anwendung von Patterns



Convention over configuration

Historie:

- Jedes Projekt besitzt eine eigene Ablagestruktur.
- Jede Bibliothek wird einzeln eingebunden.
- In jedem Projekt wird der build Prozess neu definiert.

Stattdessen in Maven: Unterstützung vernünftiger default Werte. z.B. für Ablage der Sourcen, der Tests, der Kompilate etc.



Ein einheitliches Interface

Vor Maven:

- Eigenes build system für jedes Projekt
 - Eigene source code Analyse Tools mit spezieller Einbindung in den Prozess.
 - Eigene Unit Test Frameworks mit spezieller Einbindung in den Prozess.
 - **-** . . .
- Ineffizient, hoher Aufwand der Integration der Aktivitäten und Tools

Mit Maven: Einigung auf ein einheitliches Interface um Projekte zu bauen. (nicht so sehr einheitliches Tool).



Ein einheitliches Interface

Maven beantwortet folgende Fragen

- Was ist nötig, um das Projekt zu bauen?
- Welche Libraries muss ich downloaden?
- Wo lege ich die Libraries ab?

Früher sehr projektspezifische Antworten

mit Maven standardisiert.



Wiederverwendung der build Logik durch Maven Plugins

- Der build Prozess ist deklariert und wird durch Plugins implementiert.
- Änderungen in den Plugins oder Unterstützung neuer Frameworks durch die alten oder ggfs neue Plugins ändern nicht das Build System.
- D.h. der deklarierte Prozess kann für verschiedenste Plugins verwendet werden.



Konzeptionelles Modell eines "Projekts"

Die Definition eines konzeptionellen Modells für ein Projekt und die Verwendung eindeutiger Projektkoordinaten ermöglicht:

- Dependency Management
- Remote Repositories
- Wiederverwendung der Build Logik
- Tool Integration/Vereinheitlichung
- Einfache Suche nach Artefakten



Maven - Funktionsumfang

Drei Kategorien

- 1. Durchführung des Build Prozesses.
- 2. Verwaltung der Abhängigkeiten von externen Bibliotheken.
- 3. Erstellen der Projektdokumentation.



Maven Installation

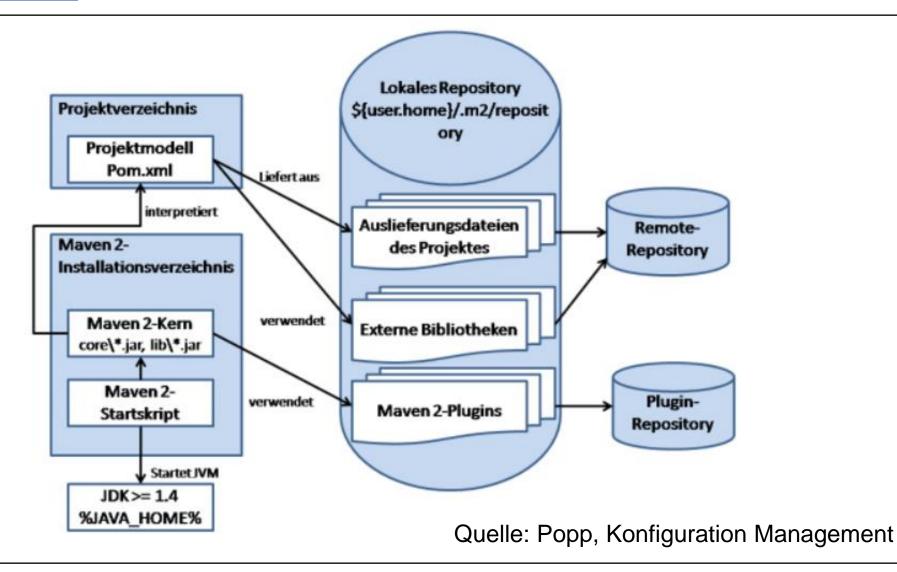
Siehe Maven by example:

- Download
- Entpacken
- Environment Variablen setzen (Windows)
- Fertig

Geringe Größe: Alle Plugins und libs werden bei Bedarf geladen.



Maven - Architektur



Prof. Dr. Michael Bulenda S. 16



Maven Konfiguration

Drei Levels der Konfiguration:

- Project im POM.xml
- Installation (z.B. Pfade)
- User: in \${user.home}/.m2/settings.xml:

Beispiele für User Settings:

- Lokales Repository: in \${user.home}/.m2/settings.xml:
- Proxy: in \${user.home}/.m2/settings.xml
- Security settings (pwd für Zugriff auf ein Repository)



Maven Projekt -einfaches Beispiel

- Quelle: Maven by Example, Kapitel 3, Verwendung des Archetype Plugin, Bauen eines "Hello World" Projekts
- Ziel: Verständnis der wesentlichen Konzepte
 - Plugins und Goals
 - Build lifecycle
 - Repositories
 - Dependency management
 - POM



Aufrufe von der Kommandozeile:

- 1. Generieren der Projektstruktur: mvn archetype:generate
 - 1. Selektieren des default archetypes
 - 2. Eingabe der Projektkoordinaten
- → Projektstruktur ist angelegt.



2. Bauen der Applikation

- mvn install (im selben directory wie das pom)
- →Es werden alle Projektphasen bis zu der Projektphase install der Reihe nach durchlaufen:
 - **→**Kompilieren
 - **→**Testen
 - → Paketieren
 - →Installieren



3. Laufen lassen

Zugriff auf das generierte jar file im target Ordner

java -cp target/simple-1.0-SNAPSHOT.jar org.bulenda.simplemavenproject.App



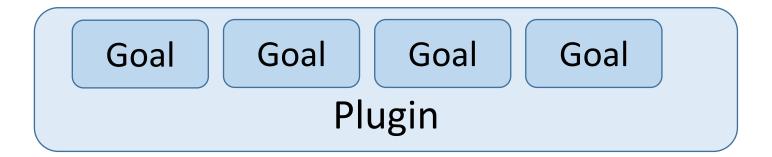
Was sieht man an diesem Beispiel? → Core Concepts

- Aufruf von maven Plugins mvn <plugin>:<goal>
- Maven Standard Projektstruktur
- Maven POM
- Maven build Phases
- Maven Repository
- Weiteres: Dependency Management



Maven Plugins und Goals

Ein Maven Plugin ist eine Kollektion von goals



Plugins für

- Erzeugen von jar Files
- Kompilieren des Source Codes
- Durchführen von Unit Tests
- Etc.

Maven delegiert die ganze Arbeit an Plugins!



Goals

Goals sind die Tasks, die die Aufgaben durchführen. Sie können konfiguriert werden, um vom default abzuweichen.

Der Kern von Maven hat wenig mit den spezifischen Tasks des Projekt Builds zu tun. Diese Aufgaben übernehmen die Plugins.



Aufruf von Plugin Goals und Konfiguration.

Aufruf eines Goals

- → maven sucht im lokalen repository nach dem Plugin.
 - 1. Ist es nicht vorhanden, lädt es das Plugin aus dem Remote Repository.
- 2. Das Goal des Plugins wird ausgeführt.



Aufruf und Konfiguration von plugins

Konfiguration von Plugins – Bsp: Der Java

Compiler soll Java 5.0 akzeptieren.

→ im POM:

```
√uild>
 <plugins>
   <plugin>
     <groupId>org.apache.maven.plugins
     <artifactId>maven-compiler-plugin</artifactId>
     <version>2.5.1
     <configuration>
       <source>1.5</source>
       <target>1.5</target>
     </configuration>
   </plugin>
 </plugins>
</build>
```



Was sieht man an diesem Beispiel? → Core Concepts

- Aufruf von maven Plugins mvn <plugin>:<goal>
- Maven Standard Projektstruktur
- Maven POM
- Maven build Phases
- Maven Repository

Weiteres: Dependency Management



Maven Standard Projektstruktur

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



Abweichen von der Standard Projektstruktur

Bsp: Abweichender Pfad zu den Test Sourcen Konfiguration im POM:

```
ct>
     properties>
          <src.junit>src/junit</junit>
     </properties>
<build>
     <testSourceDirectory>${src.junit}</testSourceDirectory>
</build>
```



Was sieht man an diesem Beispiel? -> Core Concepts

- Aufruf von maven Plugins mvn <plugin>:<goal>
- Maven Standard Projektstruktur
- Maven POM
- Maven build Phases
- Maven Repository
- Weiteres: Dependency Management



Maven Koordinaten

- Pom.xml: Project Object Model: Eine deklarative Beschreibung des Projekts. Jedes Goal hat Zugriff auf die Information des POMs.
- Diese File ist der Dreh- und Angelpunkt.
- Im POM
 - wird das Verhalten der Plugins konfiguriert
 - Werden Abhängigkeiten konfiguriert
 - Informationen zum Projekt gehalten
 - **-** . . .



Maven Koordinaten

Unique Identifier für Projekt, dependency oder plugin:

- groupId
- artifactId
- version

- → Gruppe, Organisation, ...
- → Projekt innerhalb der Organisation
- → Spezifisches Release des Projekts

Zusätzliche Koordinate:

packaging

→Projekttyp

Bsp:

<groupId>org.bulenda.simplemavenproject</groupId>

<artifactId>simple</artifactId>

<packaging>jar</packaging>

<version>1.0-SNAPSHOT</version>

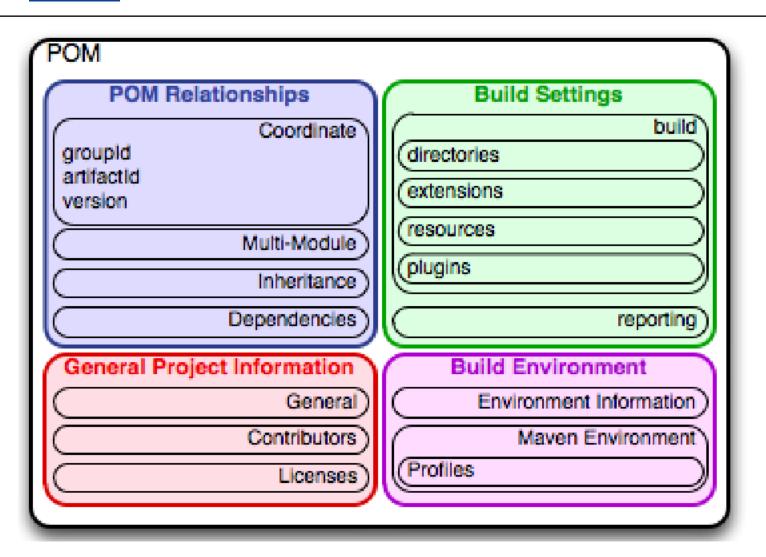


Logischer Aufbau des POM





POM



Quelle: Maven: The Complete reference



POM des einfachen Projekts

Projektkoordinaten und packaging Format

Prof. Dr. Michael Bulenda



POM des einfachen Projekts

Beschreibung des Projekts

```
....
<name>simple</name>
<url>http://maven.apache.org</url>
...
```



POM des einfachen Projekts

Projekt Dependencies

```
<dependencies>
 <dependency>
  <groupId>junit
  <artifactId>junit</artifactId>
  <version>4.11</version>
  <scope>test</scope>
 </dependency>
</dependencies>
```



Super POM

■ → Darstellen des effektiv wirksamen POMs

mvn help:effective-pom

Super POM unter

http://maven.apache.org/ref/3.8.1/maven-model-builder/super-pom.html



Einfachstes Maven Projekt

Was sieht man an diesem Beispiel? Concepts

- Aufruf von maven Plugins mvn <plugin>:<goal>
- Maven Standard Projektstruktur
- Maven POM
- Maven build Phases
- Maven Repository
- Weiteres: Dependency Management



Maven build Phases

Maven basiert auf dem zentralen Konzept des **build lifecycle.**

Siehe oben:

Aufruf *mvn install*

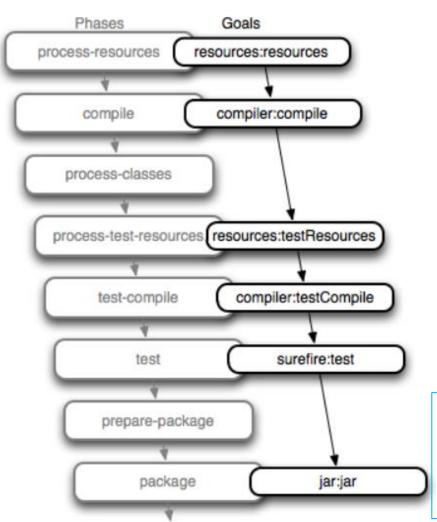
Maven Lifecycle Phase

Es gibt drei eingebaute Lifecycles:

- Default Project deployment
- Clean Project cleaning
- Site Project Documentation



Maven Life Cycle Phases



Ausschnitt! Es gibt mehr Phasen als die gezeigten!

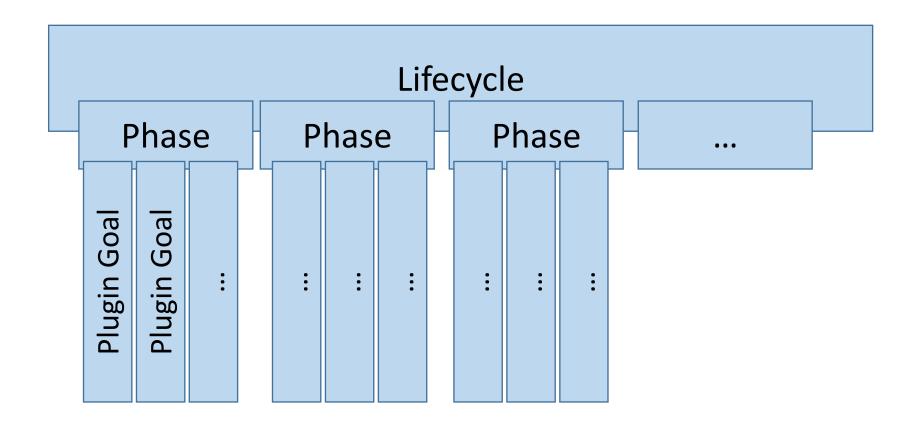
Bei einem Aufruf einer Phase werden alle Phasen bis zur aufgerufenen durchlaufen und alle verbundenen Goals aufgerufen.

Komplette Liste unter

http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle_html#Lifecycle_Reference



Lifecycle - phases - goals



Eine Phase kann ein oder mehr goals haben



Lifecycle - phases - goals

Wie bestimme ich, welche Goals für mein Projekt durchgeführt werden sollen?

1. Bestimmung aus dem packaging (Lifecycle mapping pro package type)

Werte siehe unter

http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html#Lifecycle Reference



Lifecycle – phases - goals

Wie bestimme ich, welche Goals für mein Projekt durchgeführt werden sollen?

- 2. Konfiguration von Plugins für eine Phase Plugin goals werden einer Phase hinzugefügt und dadurch zusätzlich zu den bereits konfigurierten ausgeführt.
 - → entsprechender Eintrag in die POM



Konfiguration eines zusätzliche Goals – Bsp

```
<plugin>
  <groupId>com.mycompany.example
  <artifactId>display-maven-plugin</artifactId>
  <version>1.0</version>
  <executions>
    <execution>
      <phase>process-test-resources</phase>
      <goals>
       <goal>time</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

Quelle: http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html



Maven Lifecycle

- → Der Aufruf *mvn install* war identisch mit den sequentiellen Aufrufen
- →mvn <plugin>:<goal>

- mvn resources:resources
- mvn compiler:compile
- mvn resources:testResources
- mvn surefire:test
- mvn jar:jar
- mvn install:install



Plugin Namen

Plugin gekennzeichnet durch groupid, artifactId, version

Bsp:

mvn compiler:compile



Plugin Kurzname, kompletter Name:

org.apache.maven.plugins:maven-compiler-plugin:3.8.1

mvn sucht unter dieser groupld

mvn nutzt dieses Namensschema zur Auflösung Neueste Version



Einfachstes Maven Projekt

Was sieht man an diesem Beispiel? -> Core Concepts

- Aufruf von maven Plugins mvn <plugin>:<qoal>
- Maven Standard Projektstruktur
- Maven POM
- Maven build Phases
- - Maven Repository
 - Weiteres: Dependency Management



Maven Repositories

Erste Verwendung von Maven

- → es wird eine Unmenge an Files geladen
 - 1. Plugins
 - 2. Libraries

Siehe Folie zur Maven Architektur:

Diese Files werden aus den **Remote Repositories** in das lokale Repository geladen.

Unter Windows in C:\Users\<user>\.m2\repository



Maven repositories

Default remote repositories:
 http://repol.maven.org/maven2
 von hier erfolgt der download der plugins und dependencies.

Suchoberfläche über das repository: http://search.maven.org/



Maven repositories

Maven repository: Sammlung von Projekt Artefakten in Form einer Ordner Struktur nach den Maven Koordinaten.

Ein Artefakt wird unter folgendem Pfad gespeichert:

```
<repository- root>/<groupId>
/<artifactId>/<version>/<artifactId>-
<version>.<packaging>
```

Blick ins lokale Repository → dort liegt auch das erzeugte jar



Lokales Repository

- Das mvn install Kommando hat dafür gesorgt, dass das erzeugte Artefakt im lokalen repository gespeichert wird.
- Maven sieht immer erst im lokalen repository nach, ob ein benötigtes Artefakt oder Plugin vorhanden ist, erst danach wird ein remote Repository zugegriffen.
- Das lokale Repository wird verwendet, um Abhängigkeiten zwischen lokalen Projekten zu verwalten.



Einfachstes Maven Projekt

Was sieht man an diesem Beispiel? Concepts

- Aufruf von maven Plugins mvn <plugin>:<goal>
- Maven Standard Projektstruktur
- Maven POM
- Maven build Phases
- Maven Repository
- Weiteres: Dependency Management



Dependencies im POM

Einfaches Beispiel:

```
<dependencies>
 <dependency>
                                       Eindeutige Identifikation
  <groupId>JUnit
                                       der Abhängigkeit
  <artifactId>JUnit</artifactId>
  <version>4.8.1
  <scope>test</scope>
                                       Definition der
                                       Verwendung
 </dependency>
</dependencies>
```



Suche der Dependencies

Projekt ist von einem Artefakt abhängig

■ Suche im lokalen repository mit Hilfe der Koordinaten des Artefakts

• → falls nicht gefunden → Suche im remote Repository und laden ins lokale Repository.



Transitive Abhängigkeiten

Stellen Sie sich vor, Sie benötigen eine externe Bibliothek in Form eines jar-Files.

Diese Bibliothek benötigt weitere Bibliotheken in bestimmten Versionen → Sie müssen nach und nach alle diese Abhängigkeiten Ihrem Projekt hinzufügen.

Ändert sich die Version Ihres primären Jars, geht das von vorne los.

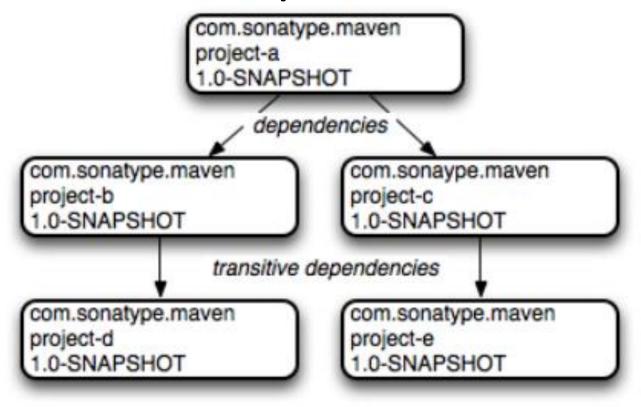
Diese jars verwalten Sie in Ihrer Sourceverwaltung.

Dadurch wird das Projekt sehr groß.



Maven dependency management

Maven löst transitive Dependencies auf!



Quelle: Maven by Example



Dependency Management

Transitive Dependencies

Dependency Scope

■ siehe http://maven.apache.org/guides/introductio n/introduction-to-dependency-mechanism.html



Einfachstes Maven Projekt

Was sieht man an diesem Beispiel? Concepts

- Aufruf von maven Plugins mvn <plugin>:<goal>
- Maven Standard Projektstruktur
- Maven POM
- Maven build Phases
- Maven Repository



Weiteres: Site Generation



Site Generation

mvn site → Generierung einer Projekt Website unter /target/site

simple

Last Published: 2015-06-06 | Version: 1.0-SNAPSHOT simple

Project Documentation

▼ Project Information
Dependencies
Dependency
Convergence
Dependency
Information
About

Plugin Management Project Plugins Project Summary



About simple

There is currently no description associated with this project.

Copyright © 2015. All Rights Reserved.



Fazit - Was haben wir gemacht?

- Projektstruktur angelegt (mvn archetype:generate)
- BuildProzess durchlaufen (mvn install)
 - Kompilieren
 - Tests kompilieren
 - Tests durchführen
 - Software paketieren
 - Software ausliefern
- Website generiert (mvn site)

Wenige Befehle, Verwendung von Convention over Configuration, Dependency management, Standard Phasen, Standard Plugins



Zusammenfassung

Kernkonzepte

- Plugins und Goals
- Lifecycle
- Koordinaten
- Repositories
- Dependency Management
- Site Generation und Reporting



Anwendung der Konzepte in einem komplexeren Beispiel

Wetter Projekt aus maven by example

Aufgabe:

- Eingabe der PLZ über command line,
- Abfragen des Yahoo! Wetter feeds,
- Darstellen der Antwort.



Maven – ein etwas komplexeres Bsp

- → neue dependencies
- Compiler Konform zu Java 5 → Konfig des Compiler plugins
- 3. Zusätzliche Project Informationen → POM ergänzen



Maven – ein etwas komplexeres Bsp

- 4. Resourcen hinzufügen (um log4J zu konfigurieren und für velocity ein Template zur Verfügung zu stellen)
- 5. Programm laufen lassen (mit dem exec Plugin des Mojo Projekts)
- 6. Unit Tests schreiben
- 7. Test Scope Dependencies hinzufügen
- 8. Unit Test Resources hinzufügen
- 9. Unit Tests ausführen
- 10.Applikation mit allen Abhängigkeiten packen



Behandlung mehrerer Projekte

■ Bisher: Fokus auf ein einzelnes Projekt

 Jetzt: Mehrere Projekte in Abhängigkeit voneinander



Zwei Möglichkeiten

Vererbung von Konfigurationen im POM

Module innerhalb eines Projekts



Vererbung von Konfigurationen im POM

Es werden folgende Elemente in den POMs vererbt:

- dependencies
- developers and contributors
- plugin lists (including reports)
- plugin executions with matching ids
- plugin configuration
- resources



Vererbung von Konfigurationen im POM

Szenario:

Projektstruktur:

```
.
|-- my-module
| `-- pom.xml Projekt POM

`-- pom.xml Parent POM
```



Vererbung von Konfigurationen im POM

Projekt POM

```
ct>
  <parent>
    <groupId>com.mycompany.app</groupId>
    <artifactId>my-app</artifactId>
    <version>1</version>
  </parent>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mycompany.app</groupId>
  <artifactId>my-module</artifactId>
  <version>1</version>
</project>
```

Verweis auf das Parent POM



Module innerhalb eines Projekts/Projekt Aggregation

Zu tun:

- 1. Parent POM: Packaging auf "POM" ändern.
- 2. Parent POM: Spezifizieren der Directories der Module



Bsp:

Projektabhängigkeiten

Module innerhalb eines Projekts/Projekt Aggregation

```
Dir Struktur

--- my-module
--- pom.xml
--- pom.xml
```

```
Parent POM
ct>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1</version>
  <packaging>pom</packaging>
  <modules>
    <module>my-module</module>
  </modules>
</project>
```



Mehrere Projekte gleichzeitig – Bsp

Dir Struktur

pom.xml my-app - pom.xml +- src +- main +- java +- my-webapp - pom.xml +- src +- main +- webapp

Parent POM

```
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>com.mycompany.app</groupId>
 <artifactId>app</artifactId>
 <version>1.0-SNAPSHOT
 <packaging>pom</packaging>
 <modules>
   <module>my-app</module>
   <module>my-webapp</module>
 </modules>
</project>
```