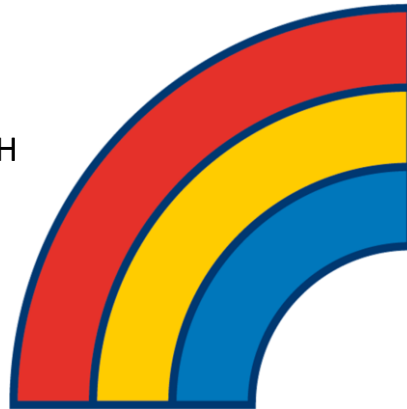


Automatischer Build mit Maven 2

Stefan Scheidt
OPITZ CONSULTING GmbH



Ihr Referent

Stefan Scheidt

- Senior Architekt bei der OPITZ CONSULTING GmbH
- Seit über 10 Jahren im Oracle- und Java-Umfeld tätig
- Schwerpunkte: Leichtgewichtiges Enterprise-Java, Spring
- Autor für Javamagazin, JavaSPEKTRUM, OBJEKTSpektrum
- Referent auf JAX, OOP, DOAG, ...
- Email: stefan.scheidt@opitz-consulting.com

Agenda

- Einführung
- Konzepte
- Dependency Management
- Multimodul-Projekte
- IDEs & Continuous Integration
- Ausblick auf Maven 3



(Keine) Motivation

In diesem Vortrag keine Motivation für Build-Automatisierung.

Aber ein paar Ziele:

- Continuous Integration
- Automatisiertes Testen
- Code-Qualität



Was ist Maven?

- „Software Project Management Tool“
- Deklaratives Build-System
 - Convention over Configuration
 - Wiederverwendung durch Plugins
 - Dependency Management
- Tool für technisches Projekt-Reporting



Was ist Maven? (2)

- Open-Source-Projekt bei Apache
 - Entstanden 2002 für diverse Apache-Projekte
 - Ziel: Vereinheitlichen von
 - Build
 - Verteilung
 - Erstellen einer Projekt-Webseite
- (Don't repeat yourself!)



Das „Project Object Model“ (POM)

- Beschreibt ein Projekt für Maven...
 - „Artefakt-Koordinaten“
 - Informationen für die Webseite
 - Was enthält das Projekt?
 - Welche Abhängigkeiten hat das Projekt?
- ...als XSD-basierte XML-Datei



(Nahezu) minimales POM

```
<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>de.opitzconsulting</groupId>
  <artifactId>hellomaven</artifactId>
  <version>1.0.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>Hello Maven</name>
  <description>Simple Maven Project</description>
</project>
```



Convention over Configuration

- Diese Projektbeschreibung wird durch Voreinstellungen ergänzt, ...
- ... die in „Super POM“ definiert sind http://maven.apache.org/pom.html#The_Super_POM
- Dadurch kann man jetzt schon kompilieren, (JUnit-)Tests ausführen, JAR bauen, JAR verteilen, Projekt-Website generieren...



Standard-Verzeichnisstruktur

```
/src/main/java  
/src/main/resources  
/src/main/webapp  
/src/test/java  
/src/test/resources  
/target/classes  
/target/test-classes
```



Maven Plugins

- Im Kern ist Maven eine Laufzeitumgebung für Plugins
- Diese stellen die eigentlichen Build-Funktionen als „Goals“ zur Verfügung
- Das „Super POM“ macht die gebräuchlichsten Plugins verfügbar (compile, „test“, jar, war, ...)
- Weitere können in Projekt-POM registriert werden



Beispiel: Jetty Plugin

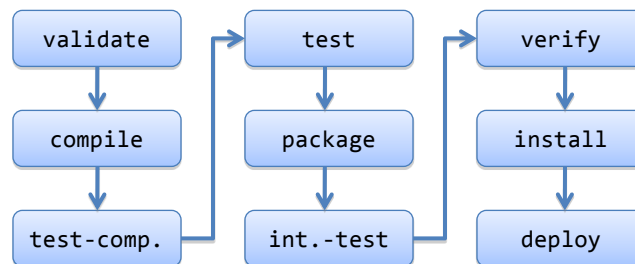
```
<!-- ... -->
<build>
  <plugins>
    <plugin>
      <groupId>org.mortbay.jetty</groupId>
      <artifactId>maven-jetty-plugin</artifactId>
    </plugin>
  </plugins>
</build>
<!-- ... -->
```

...und „mvn jetty:run“ startet Jetty



Build Lifecycle

- Sequenz von „Build-Phasen“
- Eingebaute Lifecycle: „clean“, „default“, „site“
- Auszug aus Default Lifecycle:



Build Lifecycle (2)

- Plugin Goals können an Lifecycle-Phasen gebunden werden
- Binding wird durch Packaging Type ausgewählt
- Maven gibt bereits Bindungen vor
- Plugins können
 - neue Bindungen definieren
 - Bindungen und Lifecycle anpassen
 - neue Packaging Types definieren



Beispiel: Binding für jar/war

Lifecycle Phase	Plugin:Goal
compile	compiler:compile
test-compile	compiler:testCompile
test	surefire:test
package	jar:jar (war:war)

Auszug aus dem Lifecycle Binding
für Packaging Type jar und war

Beispiel: Custom Binding (1)

```
<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>sql-maven-plugin</artifactId>
  <!-- ... -->
  <executions>
    <execution>
      <id>create-schema-before-test</id>
      <!-- ... -->
    </execution>
    <execution>
      <id>drop-schema-after-test</id>
      <!-- ... -->
    </execution>
  </executions>
</plugin>
```

Auszug aus einem POM: Konfiguration für SQL Plugin mit zwei Executions

Beispiel: Custom Binding (2)

```
<execution>
  <id>create-schema-before-test</id>
  <phase>pre-integration-test</phase>
  <goals>
    <goal>execute</goal>
  </goals>
  <configuration>
    <srcFiles>
      <srcFile>
        src/main/sql/schema-create.sql
      </srcFile>
    </srcFiles>
  </configuration>
</execution>
```

Auszug aus einem POM: Execution-Konfiguration für die Phase „pre-integration-test“



OPITZ CONSULTING

Dependency Management

- Dependencies beschreiben Abhängigkeiten zu Java-Bibliotheken (JARs)
- werden über „Artefakt-Koordinaten“ spezifiziert
- können durch „Scopes“ (compile, test, runtime, provided, ...) qualifiziert werden
- werden bei Bedarf aus einem „Remote Repository“ heruntergeladen



OPITZ CONSULTING

Beispiel: Dependency zu JUnit 4.4

```
<dependencies>
  <!-- JUnit -->
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.4</version>
    <scope>test</scope>
  </dependency>
  <!-- ... -->
</dependencies>
```

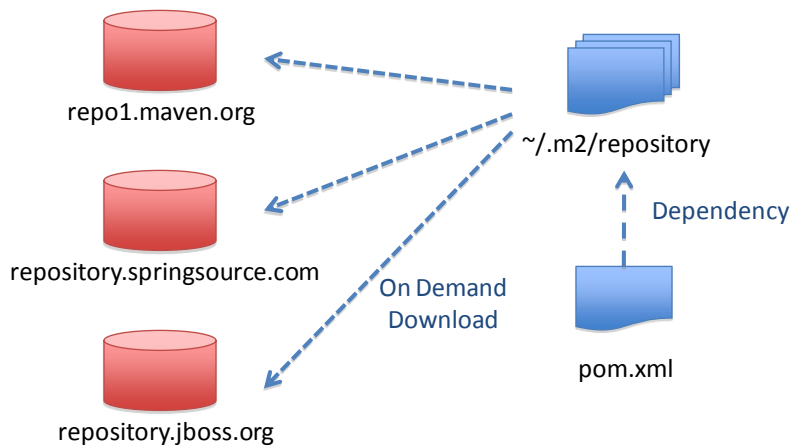


Maven Repositories

- Stellen Artefakte bereit
- Lokal: im Filesystem („~/ .m2/repository“)
- Remote: als „Web Service“ zum Download
- Default Remote Repository:
<http://repo1.maven.org/maven2>
- Weitere Repositories können in Projekt-POM registriert werden



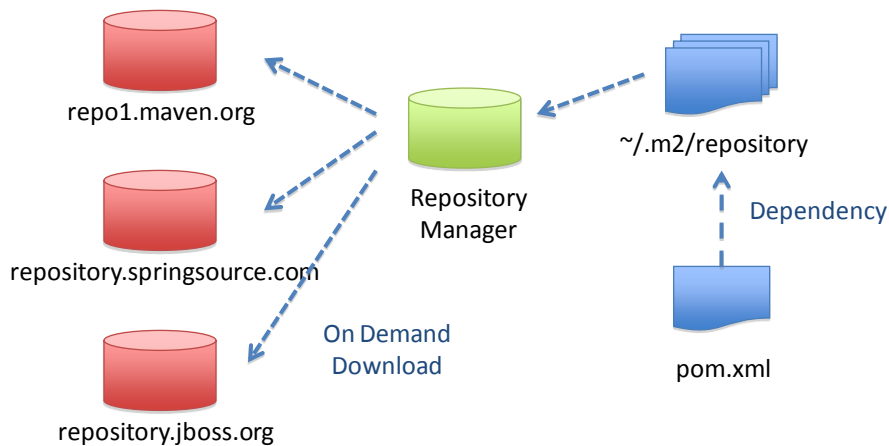
Maven Repositories (2)



Repository Manager

- Stellt Remote Repository Service bereit
- Zweck:
 - Zentraler Proxy für benötigte Remote Repositories
 - Bereitstellung nicht öffentlich verfügbarer JARs
 - Kontrolle über verwendete JARs
 - Verteilung selbsterstellter Artefakte

Repository Manager (2)



Repository Manager (3)

- Empfehlung: Repository Manager einsetzen!
 - Unternehmensweit
 - Ggf. gestaffelt: Unternehmen & pro Projekt
- R.M.-Implementierungen:
 - Sonatype Nexus
<http://nexus.sonatype.org/>
 - JFrog Artifactory
<http://www.jfrog.org/products.php>

POMs und Vererbung

- POMs können von einem Parent POM erben
- Wurzel ist das „Super POM“
- Ein Parent POM kann vererben:
 - Plugin-Konfigurationen
 - Dependencies
 - vieles mehr...
- Häufig in Verbindung mit Multimodul-Projekten

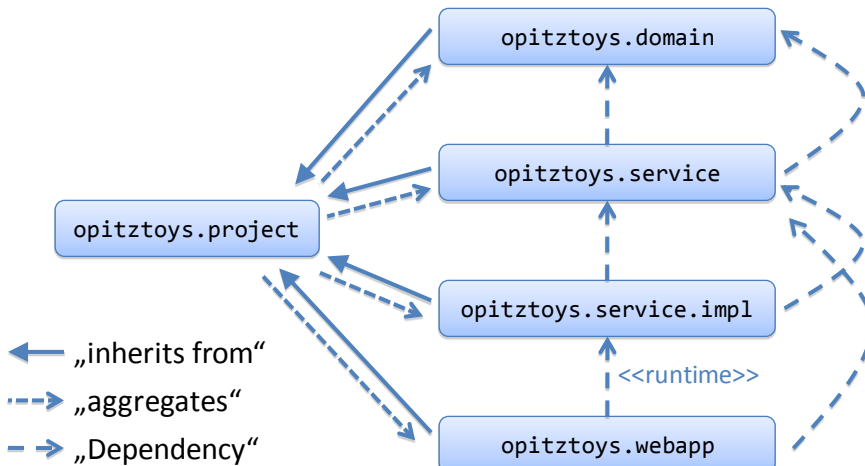


Multimodul-Projekte

- Ein Maven-Projekt kann nur ein Artefakt produzieren
- Maven forciert dadurch die Aufteilung eines Projekts auf mehrere Sub-Projekte
- Diese können durch ein Multimodul-Projekt zusammengefasst und gemeinsam gebaut werden
- Multimodul-POM ist meist auch Parent POM



Beispiel: Multimodul-Projekt



Einschub: Maven und OSGi

- Auf den ersten Blick passen Maven und OSGi gut zusammen
- Die Dependency-Konzepte unterscheiden sich aber grundlegend... Und: „Wer ist der Boss“?
- Es gibt aber Integrationssupport
 - Maven-Bundle-Plugin <http://tinyurl.com/66q93b>
 - Pax Construct <http://www.ops4j.org/projects/pax/construct>
 - SpringSource Bundlor <http://www.springframework.org/bundlor>

Maven und IDEs

- Eclipse
 - eclipse: Maven Plugin für Eclipse
 - m2eclipse: Eclipse Plugin für Maven
 - Eclipse IAM: Integration von eclipse.org
- NetBeans
 - Ab Version 6.7 direktes Ausführen von Maven
- IntelliJ
 - Build-In-Support für Import von Maven-Projekten



Maven und Continuous Integration

Support durch diverse CI-Server

- CruiseControl
- Continuum
- Hudson
- TeamCity
- Bamboo
- ...



Site-Generierung

- Webseite mit „Projektsteckbrief“
- Einbetten von JavaDoc und Projektdoku.
- Einbetten von Berichten durch Reporting-Plugins: Checkstyle, JUnit/TestNG, Corbatura, PMD, FindBugs, JDepend, ...
- Beispiele: Siehe Open-Source-Projekte, die Maven verwenden...



Maven 3

- Umfangreiches Refactoring der Codebase
- Neue API für Artifact Resolution
- Bessere Unterstützung für IDE-Integration
- Kompatibilität mit Maven 2 wird durch Integrationstests sichergestellt
- Siehe auch <http://www.sonatype.com/events/meetup0309/jason-on-maven3>



Fazit und Bewertung

- Standardisierung zahlt sich bei einer großen Zahl von Projekten aus
- Schnelle Ergebnisse bei Standard-Anforderungen
- Für komplexere Anpassungen ist jedoch umfangreiche Einarbeitung nötig
- Passt evtl. nicht bei sehr individuellen Anforderungen (bzw. sehr großer Aufwand)



Links

- Projekt-Seite
<http://maven.apache.org/>
- Maven The Definitive Guide
<http://www.sonatype.com/books/maven-book/reference/>
- DZone Refcard
<http://refcardz.dzone.com/refcardz/apache-maven-2>
- Maven How Tos
<http://www.sonatype.com/people/2009/04/summary-of-maven-how-tos/>
- Nexus
<http://nexus.sonatype.org/>
- Artifactory
<http://www.jfrog.org/products.php>
- m2eclipse
<http://m2eclipse.codehaus.org/>



Vielen Dank für Ihre Aufmerksamkeit!



Kontakt:

stefan.scheidt@opitz-consulting.com