

# Maven2

---

***maven*** als *Enabler* für *Continuous Integration*  
(*Maven2 Einführung und Best Practices*)

2007.05.24

Dipl.Ing.(FH) Martin Ahrer

# Maven2

---

"Die Tätigkeiten von Softwareingenieuren sind geprägt von einem großen Anteil an Routinearbeiten. Deren konsequente Vernachlässigung gefährdet die Projektqualität, deren konsequente Durchführung gefährdet die Agilität eines Teams.

Softwareentwicklung wird immer mehr zu einem „industriellen Prozess“ und verlangt nach einem hohen Maß an Automatisierung.

Maven2 offeriert ein Modell zur Standardisierung und Vereinfachung von Build-Prozessen. Mit Maven2 wird „Continuous Integration“ zur Selbstverständlichkeit und seine Integration in die neuen Generationen von Entwicklungsumgebungen bestätigt die zunehmende Popularität von Maven2.

Während dieser Präsentation werden die Schlüsselfeatures von Maven2 gezeigt und „Best-Practices“ diskutiert.

In einer Live-Demo wird der Build-Prozess für ein Webservice realisiert und vom Continuous Integration Server Continuum ausgeführt. Es erfolgt weiter ein automatisiertes "Deployment" auf einem Apache Tomcat Web Container und die Ausführung von Integrationstests mittels SoapUI."

# Vortragender

---

## Qualifikationen

- Softwareentwickler seit 1990 (Java seit 1998)
- Selbständige Tätigkeit seit 2004

## Maven

- Erfolgreicher Einsatz von Maven1 und Maven2 in großen Projekten
- Maven-Plugin Entwicklung

## Open Source Projekt »Maven Station«

- <http://www.martinahrer.at/oss/m2/maven-best-practice/index.html>

# Umfrage

---

**Wer kennt (arbeitet mit) Ant ?**

**Wer kennt (arbeitet mit) Maven ?**



# Agenda

---

## ***Continuous Integration und Agilität***

Maven Prinzipien

Projekt Objekt Modell (POM)

Projektstrukturen

Buildprozess

Reporting

Continuous Integration Demo

# Continuous Integration – Martin Fowler

---

## Maintain a Single Source Repository

- SVN, CVS, Perforce, ClearCase, ...

## Automate the Build

- Maven, Ant, ...

## Make Your Build Self-Testing

- Xtreme Programming, Test Driven Development

## Everyone Commits Every Day

- Avoid errors conflicting with other developers

# Continuous Integration – Martin Fowler

---

## Every Commit Should Build the Mainline on an Integration Machine

- Cruise Control, Continuum, ...
- Continuous Build versus Nightly Build

## Keep the Build Fast

- Unit Testing versus Integration Testing!
- Mock Objects (EasyMock)

# Continuous Integration – Martin Fowler

---

## Test in a Clone of the Production Environment

- Maven Cargo Plugin

## Make it Easy for Anyone to Get the Latest Executable

- Artifact Repository

## Everyone can see what's happening

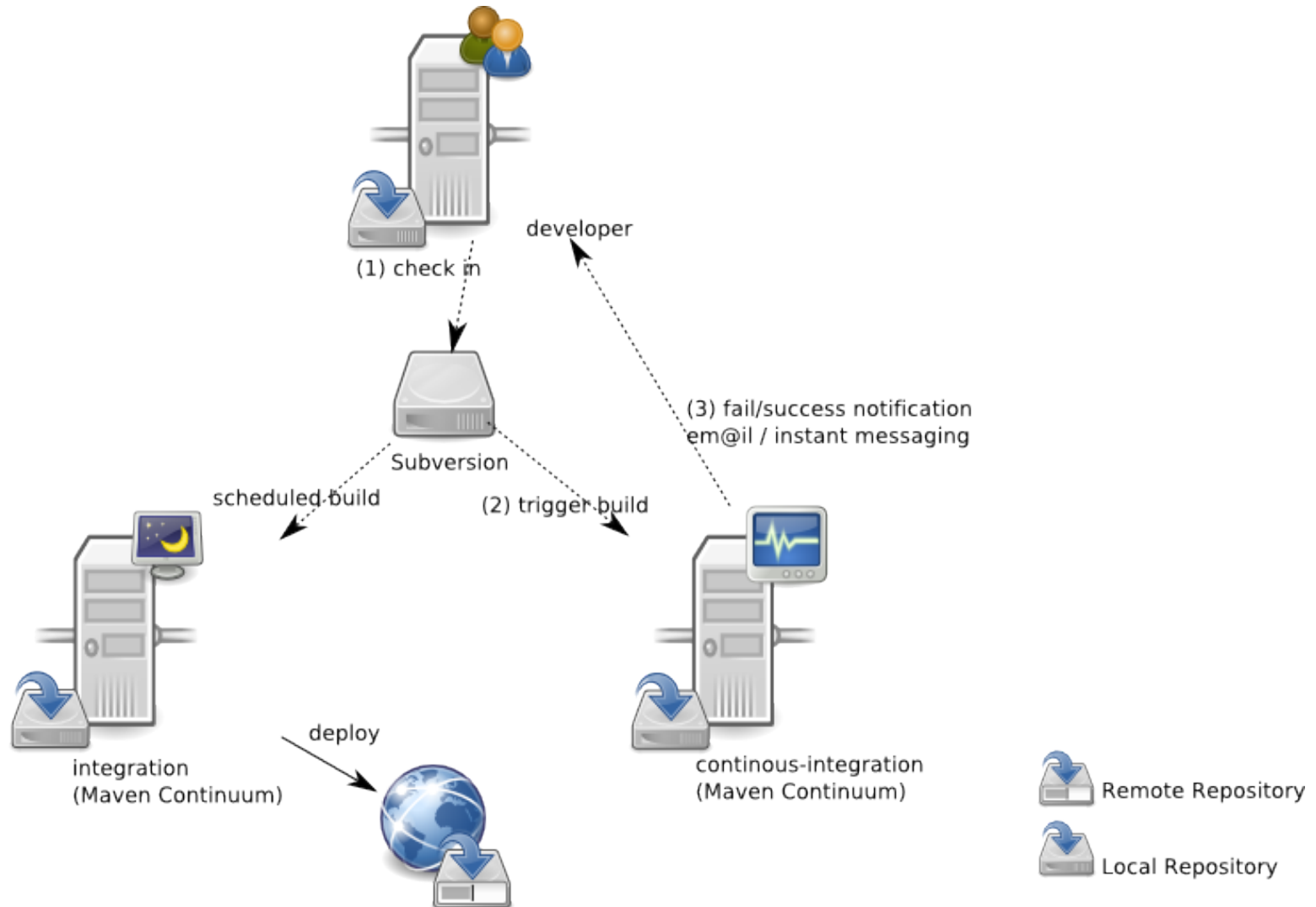
- Reporting

## Automate Deployment

- Maven Cargo Plugin

Quelle: <http://www.martinfowler.com/articles/continuousIntegration.html>

# Continuous Integration Zyklus



# Agilität?

**Irgendwo**  
hatte ich ja  
noch ein  
**TODO** im Code  
vermerkt...

Der **Checkout**  
meiner Web  
App dauert  
heute wieder  
**ewig!!!**

Woher bekomme  
ich nun alle JAR  
von denen  
hibernate.jar  
**abhängig** ist ???

Heute ist ein  
**Release** fällig,  
das wird eine  
lange Nacht



Welche  
**Version** von  
hibernate.jar  
benötige ich???

Ich habe jetzt sicher  
keine Zeit um  
dieses coole  
**FindBugs Tool** zu  
konfigurieren...

Welche **Dateien**  
benötige ich für ein  
**neues Web**  
**Projekt???**

Bei **mir**  
funktioniert alles!

# Agenda

---

Continuous Integration und Agilität

***Maven Prinzipien***

*Projekt Objekt Modell (POM)*

Projektstrukturen

Buildprozess

Reporting

Continuous Integration Demo

# Maven Prinzipien (1)

---

*„Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.“*

- „Making the build process easy“
- „Providing a uniform build system“
- „Providing quality project information“
- „Providing guidelines for best practices development“
- „Allowing transparent migration to new features“

# Maven Prinzipien (2)

---

## Wiederverwendung von Build Logik

- Plugins für isolierte Aufgaben
- Deklarative Ausführung (Project Object Model)
- Definierter „Build Life Cycle“ koordiniert die Ausführung von Plugins während eines Build

## Kohärente Organisation von Abhängigkeiten

- Build Tasks (Plugins)
- Libraries (Dependencies)

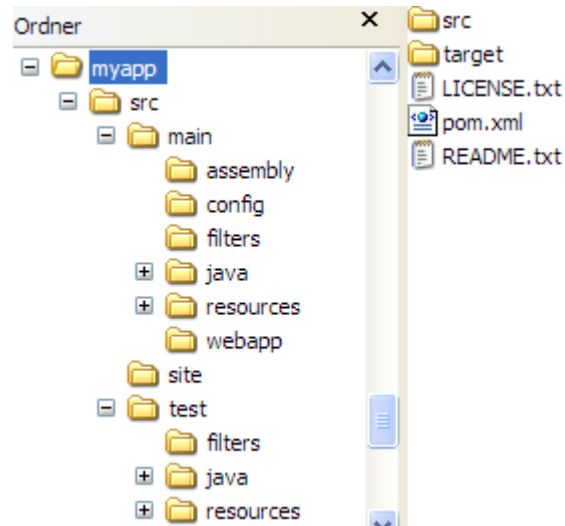
## Reproduzierbarkeit von Builds

## Keine Artefakte lokal im Projekt (SCM)!!!

# Maven Prinzipien (3)

## Projekt erzeugt exakt ein Artefakt (z.B. WAR) „Convention over Configuration“

- Standard Namenskonventionen
- Standard Verzeichnislayout



|                    |                               |
|--------------------|-------------------------------|
| src/main/java      | Application/Library sources   |
| src/main/resources | Application/Library resources |
| src/main/filters   | Resource filter files         |
| src/main/assembly  | Assembly descriptors          |
| src/main/config    | Configuration files           |
| src/main/webapp    | Web application sources       |
| src/test/java      | Test sources                  |
| src/test/resources | Test resources                |
| src/test/filters   | Test resource filter files    |
| src/site           | Site                          |
| LICENSE.txt        | Project's license             |
| README.txt         | Project's readme              |

# 5-Minuten Test (1)

---

## Maven Installation (maven-2.0.6.zip)

## JAR Projekt neu anlegen mittels Maven Archetype Plugin

- Auswahl der Archetype Plugin Koordinaten
- Festlegung Projekt Koordinaten

```
mvn archetype:create
-DarchetypeGroupId=at.martinahrer
-DarchetypeArtifactId=maven-jar-archetype
-DarchetypeVersion=1.0
-DgroupId=mydomain
-DartifactId=myapp
-Dversion=1.0-SNAPSHOT
-DpackageName=mypackage
```

# 5-Minuten Test (2)

## Artefakt (JAR Datei) bauen

```
$javac App.java  
$javac AppTest.java  
$java junit.textui.TestRunner  
$jar cvf myapp-1.0-SNAPSHOT.jar ...
```

```
$mvn package
```

## Projekt Site generieren

- Projektinformation
  - Libraries, ...
- Projekt Reports
  - JavaDoc, ...
- Projekt Dokumentation

```
$mvn site
```

The screenshot shows a web browser window titled 'myapplication - Project Summary - Mozilla Firefox'. The page content includes:

- Project Information** table:

| Field       | Value   |
|-------------|---|
| Name        | myapplication   |
| Description | myapplication   |
| Homepage    | <a href="http://www.martinahrer.at/maven-jar-project/myapplication">http://www.martinahrer.at/maven-jar-project/myapplication</a> |
- Project Organization** table:

| Field | Value   |
|-------|---|
| Name  | Martin Ahrer  |
| URL   | <a href="http://www.martinahrer.at">http://www.martinahrer.at</a> |
- Build Information** table:

| Field      | Value         |
|------------|---------------|
| GroupId    | mydomain      |
| ArtifactId | myapplication |
| Version    | 1.0-SNAPSHOT  |
| Type       | jar           |

# 5-Minuten Test (4)

## myapp Projekt Deskriptor

```
<?xml version="1.0" encoding="UTF-8"?>
<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>

  <parent>
    <groupId>at.martinahrer</groupId>
    <artifactId>maven-jar-project</artifactId>
    <version>1.0</version>
  </parent>

  <groupId>mydomain</groupId>
  <artifactId>myapp</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>myapp</name>
  <description>myapp</description>
</project>
```

# Agenda

---

Continuous Integration und Agilität

Maven Prinzipien

***Projekt Objekt Modell (POM)***

Projektstrukturen

Buildprozess

Reporting

Continuous Integration Demo

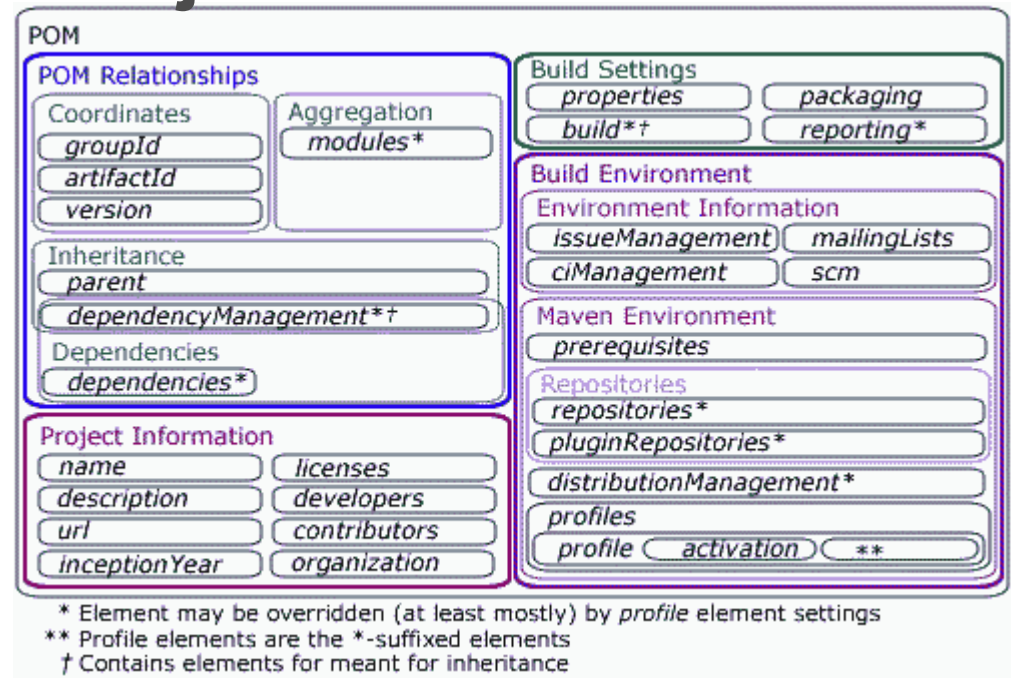
# Projekt Deskriptor - POM (1)

XML Datei beschreibt Struktur, Eigenschaften und Buildprozess eines Projekts.

- pom.xml

## Deskriptor Struktur

- Projektstrukturen
- Projektinformation
- Build Prozess
- Build Umgebung



## Deskriptor ist dynamisch

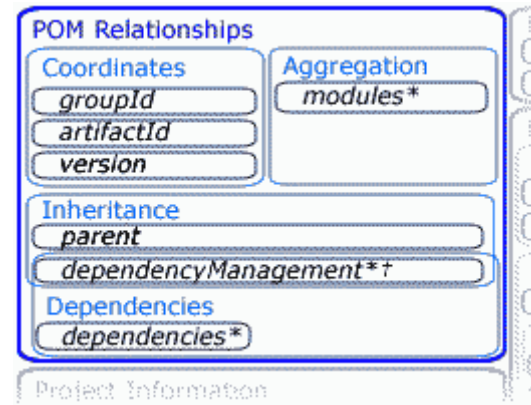
Quelle: <http://www.javaworld.com/javaworld/jw-05-2006/jw-0529-maven.html>

- Ersetzung von Variablen

# Projekt Deskriptor - POM (2)

## Maven Koordinaten

- groupId
- artifactId
- version
- packaging
- (classifier)



Quelle:  
<http://www.javaworld.com/javaworld/jw-05-2006/jw-0529-maven.html>

## Identifikation eines Artefakts im Repository! SNAPSHOT Version !!!

```
$M2_REPO/mydomain/myapp/1.0-SNAPSHOT/myapp-1.0-SNAPSHOT.jar
```

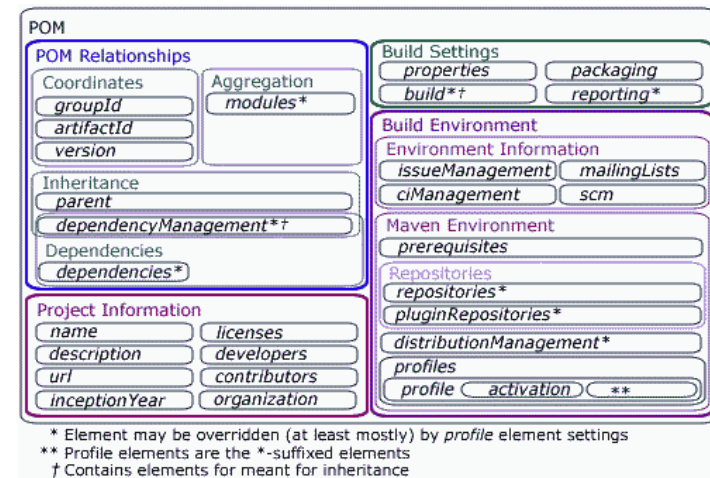
# Projekt Deskriptor - POM (4)

## Problem: Komplexität von Projekten und Build Prozessen -> komplexer Projektdeskriptor

- Woran ist der Entwickler interessiert („Separation of Concerns“)
  - Compilieren (Classpathdefinition)
  - JUnit Test, Test Coverage
  - QA Metrik Reports
  - ...

## Lösung:

- Generalisierung (Vererbung)
- Aggregation
- Profile



Quelle:

<http://www.javaworld.com/javaworld/jw-05-2006/jw-0529-maven.html>

# Agenda

---

Continuous Integration und Agilität

Projekt Objekt Modell (POM)

***Projektstrukturen***

Buildprozess

Reporting

Continuous Integration Demo

# Projektstrukturen – Dependencies (1)

---

## Dependency beschreibt Abhängigkeit von einem Artefakt und enthält

- Maven Koordinaten des abhängigen Artefakts
- Type (**jar**, war, ear, ...)
- Scope (**compile**, provided, runtime, test, system)

## Wozu?

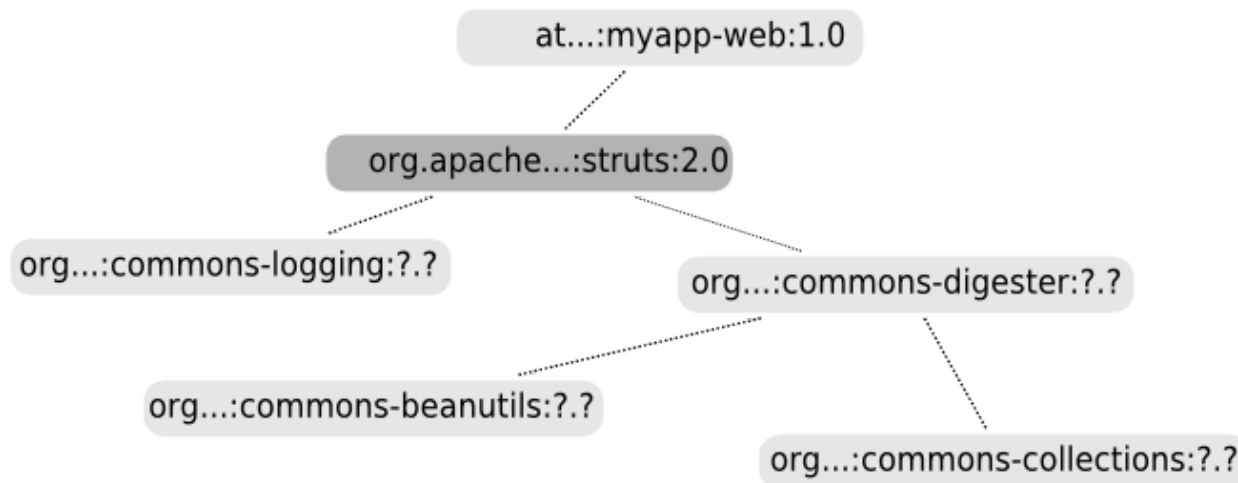
- Classpathdefinitionen
- Komponenten für Assemblies (Installer!)

## JSR 277 – Java Module System (Java7)

# Projektstrukturen – Dependencies (2)

## Transitive Abhängigkeiten erlauben kompakte Beschreibung von Abhängigkeitsregeln

```
<dependencies>
  <dependency>
    <groupId>org.apache.struts</groupId>
    <artifactId>struts</artifactId>
    <version>2.0</version>
  </dependency>
</dependencies>
```



# Projektstrukturen – Dependencies (3)

---

## Scope limitiert Gültigkeit einer Dependency

```
<dependencies>
  <dependency>
    <groupId>org.junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.0</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

# Projektstrukturen – Dependencies (4)

---

## Compile

- this is the default scope, used if none is specified. Compile dependencies are available in all classpaths.

## System

- this scope is similar to provided except that you have to provide the JAR which contains it explicitly. The artifact is always available and is not looked up in a repository

## Runtime

- this scope indicates that the dependency is not required for compilation, but is for execution. It is in the runtime and test classpaths, but not the compile classpath.

# Projektstrukturen – Dependencies (5)

---

## Provided

- this is much like compile, but indicates you expect the JDK or a container to provide it. It is only available on the compilation classpath, and is not transitive

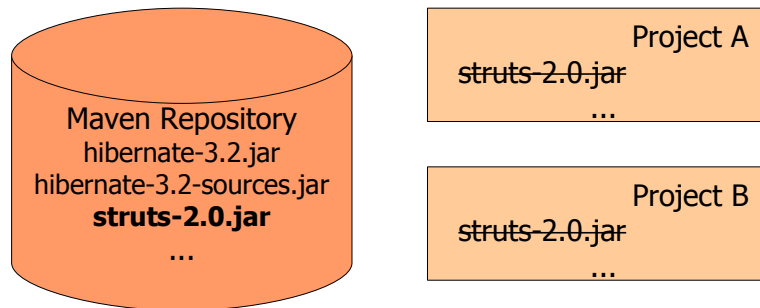
## Test

- this scope indicates that the dependency is not required for normal use of the application, and is only available for the test compilation and execution phases.

# Projektstrukturen – Repositories (1)

Fowler - „Make it easy for anyone to get the latest...“

Maven Prinzip - Keine Artefakte im SCM (Projekt)



## Artefakte archiviert in zentralen Repositories

- Binäre Build Artefakte (Dependencies) (\*.jar, \*.war, ...)
- Artefaktmetadaten
- Sources (Debugging!)

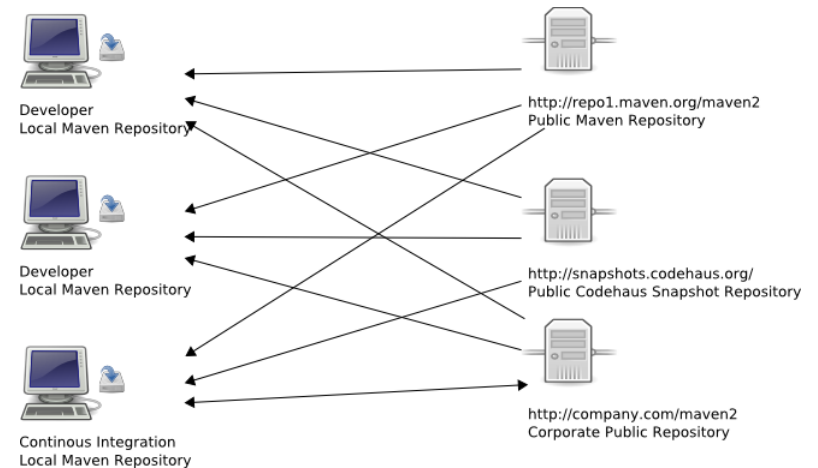
# Projektstrukturen – Repositories (2)

## Remote Repository

- Extern (<http://>, WebDAV)
- Intern (<file://>)

## Local Repository

- Arbeitet als lokaler Artefakt Cache
- Ermöglicht „sharing“ von Artefakten über Projektgrenzen
- Artefakte werden automatisch von einem Remote Repository in das Local Repository übertragen und aktualisiert (SNAPSHOT Mechanismus)



# Projektstrukturen – Repositories (3)

## Local Repository Konfiguration

- Default Repository: `~/.m2/repository`
- Konfiguration in `~/.m2/settings.xml`

```
<?xml version="1.0"?>
<settings 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/settings-1.0.0.xsd">

  <localRepository>D:/Projects/.m2/repository</localRepository>
</settings>
```

# Projektstrukturen – Repositories (4)

## Remote Repository Konfiguration

- Default Repository: <http://repo1.maven.org/maven2>
- Konfiguration im Projektdeskriptor (pom.xml)

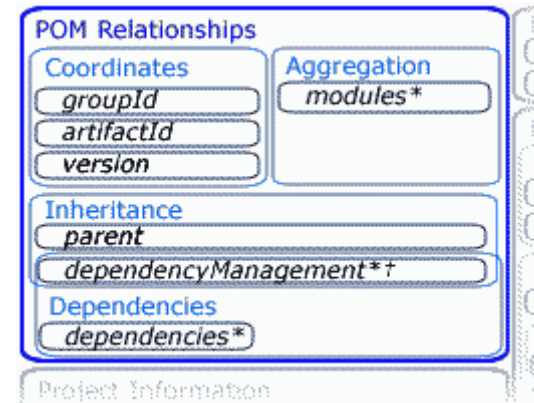
```
<pluginRepositories>
  <pluginRepository>
    <id>codehaus snapshot repository</id>
    <url>http://snapshots.repository.codehaus.org/</url>
  </pluginRepository>
</pluginRepositories>

<repositories>
  <repository>
    <id>codehaus snapshot repository</id>
    <url>http://snapshots.repository.codehaus.org/</url>
  </repository>
</repositories>
```

# Projektstrukturen – Generalisierung (1)

## Vererbung von

- Dependencies
- Build Plugins
- ...



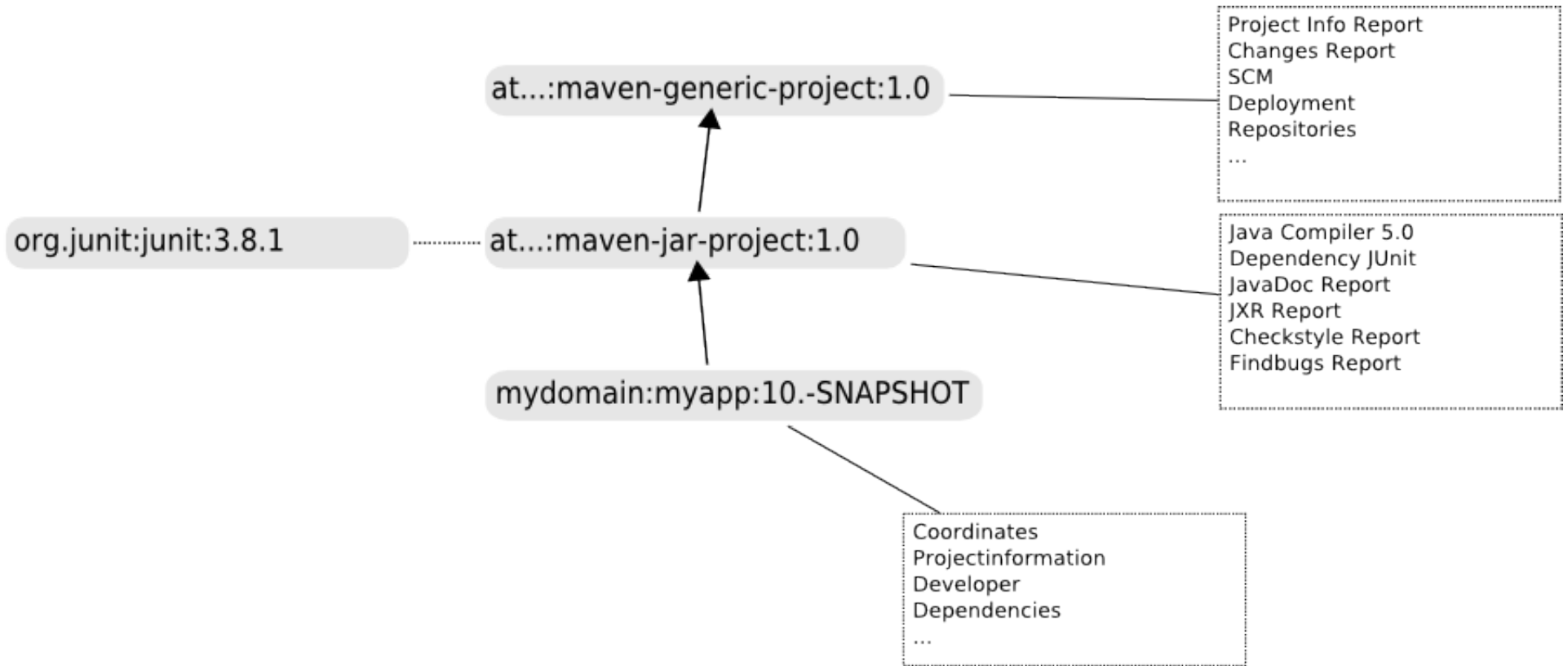
Quelle:  
<http://www.javaworld.com/javaworld/jw-05-2006/jw-0529-maven.html>

## Super POM

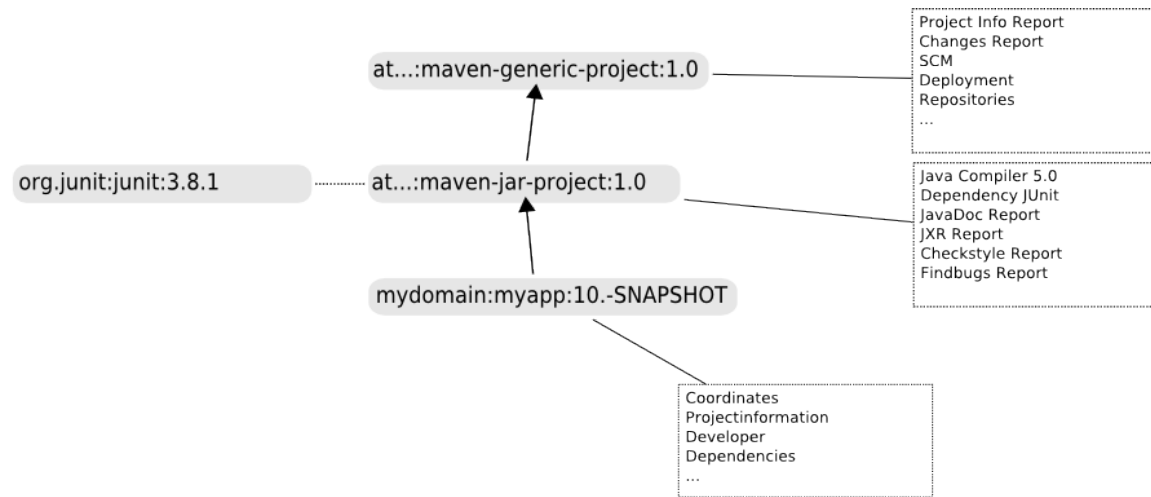
- Implementiert Prinzip „*Convention Over Configuration*“
- Defaultkonfiguration
- Implizite Basis aller Projekte

```
<parent>
  <artifactId>maven-jar-project</artifactId>
  <groupId>at.martinahrer</groupId>
  <version>1.0</version>
</parent>
```

# Projektstrukturen – Generalisierung (2)



# Projektstrukturen – Generalisierung (3)

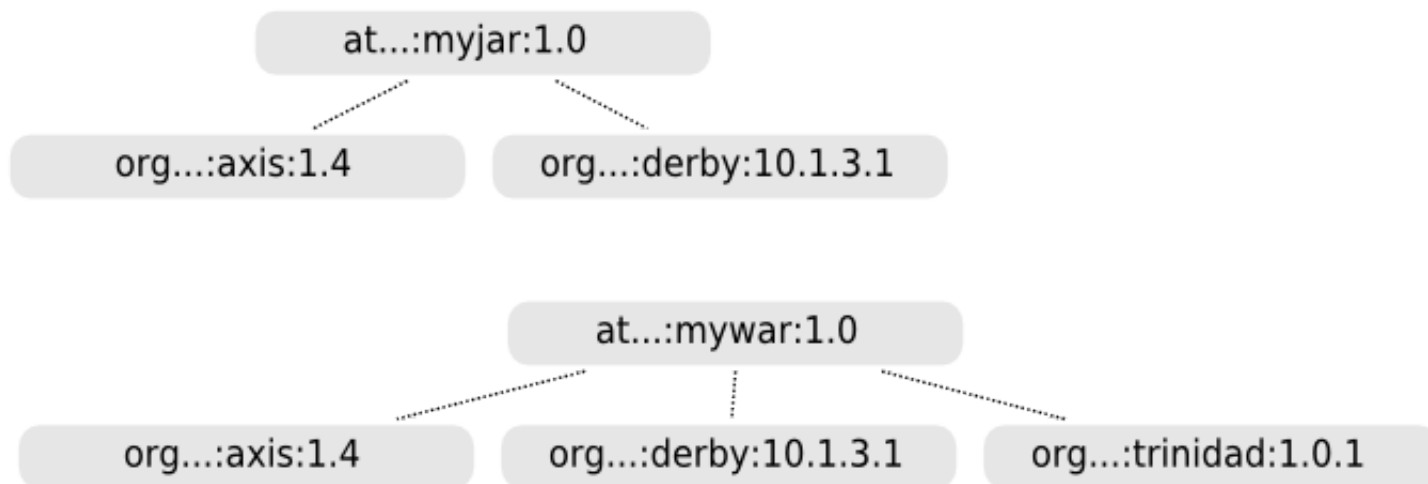


```
$ mvn dependency:resolve
[INFO] Scanning for projects...
[INFO] Searching repository for plugin with prefix: 'dependency'.
WAGON_VERSION: 1.0-beta-1
[INFO] -----
[INFO] Building myapp
[INFO]   task-segment: [dependency:resolve]
[INFO] -----
[INFO] [dependency:resolve]
[INFO] The following files have been resolved:
[INFO]   junit:junit:jar:3.8.1 (scope = test)
```

# Projektstrukturen – Generalisierung (4)

## Problem: Gemeinsam benutzte Libraries innerhalb einer komplexen Projektstruktur

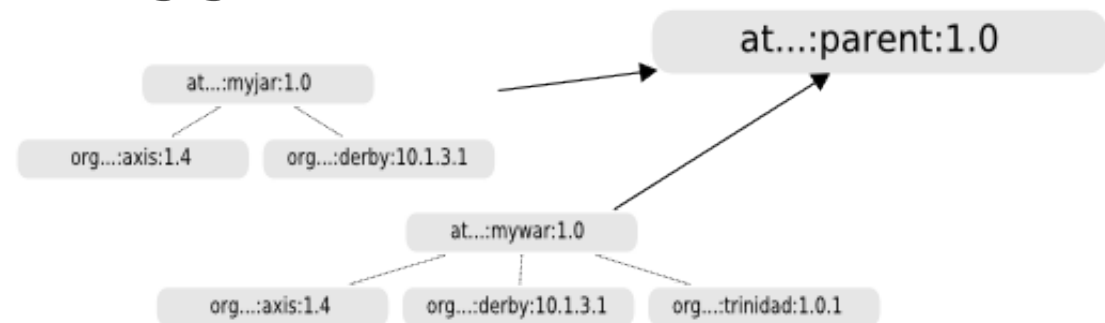
- Gewährleistung konsistenter Versionen aller Libraries in allen Projekten



# Projektstrukturen – Generalisierung (5)

## Lösung: Dependency Management

- Erzeugt keine Abhängigkeit
- Konfiguriert eine Abhängigkeit



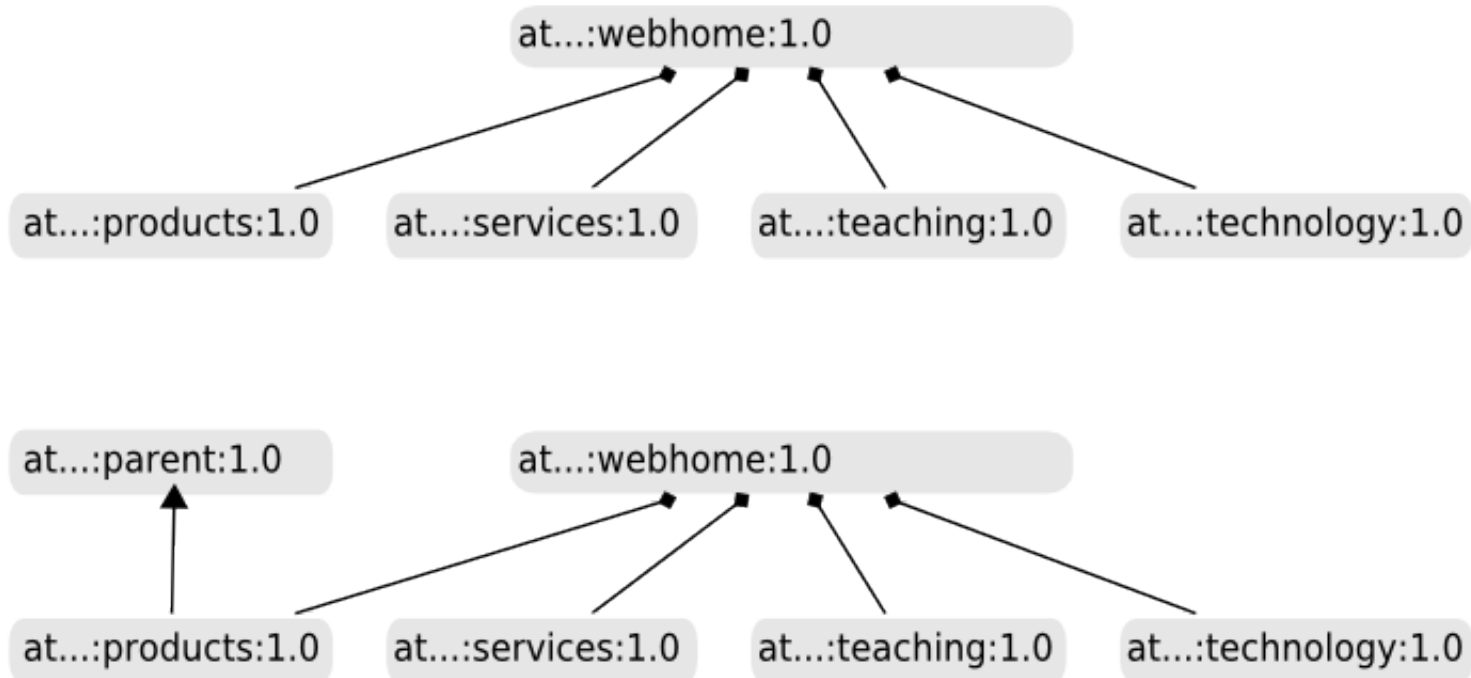
```
<!-- setup in parent POM -->
<dependencyManagement>
  <dependency>
    <groupId>org.apache.axis</groupId>
    <artifactId>axis</artifactId>
    <version>1.4</version>
  </dependency>
</dependencyManagement>
```

```
<!-- setup in child POM -->
<dependencies>
  <dependency>
    <groupId>org.apache.axis</groupId>
    <artifactId>axis</artifactId>
  </dependency>
</dependencies>
```

# Projektstrukturen – Aggregation (1)

## Multimodulprojekt

- Modul erbt vom Aggregator
- Modul kann von anderem Projekt erben!



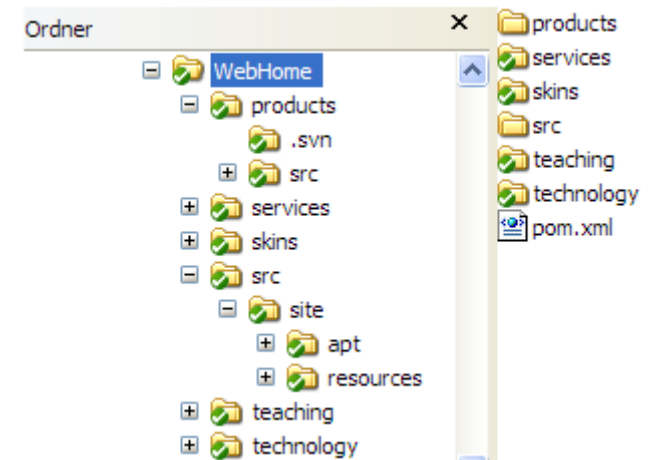
# Projektstrukturen – Aggregation (2)

## Buildprozeß bearbeitet Aggregator und Module

- Builddurchführung Aggregator + Module

## Modul liegt in gleichnamigem Unterverzeichnis

```
<project >
  <modelVersion>4.0.0</modelVersion>
  <groupId>at.martinahrer.webhome</groupId>
  <artifactId>main</artifactId>
  <version>1.0.3-SNAPSHOT</version>
  <packaging>pom</packaging>
  <modules>
    <module>products</module>
    <module>services</module>
    <module>teaching</module>
    <module>technology</module>
  </modules>
</project >
```



# Agenda

---

Continuous Integration und Agilität

Projekt Objekt Modell (POM)

Projektstrukturen

***Buildprozess***

Reporting

Continuous Integration Demo

# Build Life Cycle (1)

---

## Fowler - „Automate the Build“

### Elementares Konzept: „Build Life Cycle“

- Der Prozeß der Erstellung und Verteilung eines Artefakts ist klar definiert (er ist immer gleich!)

### Build Phase

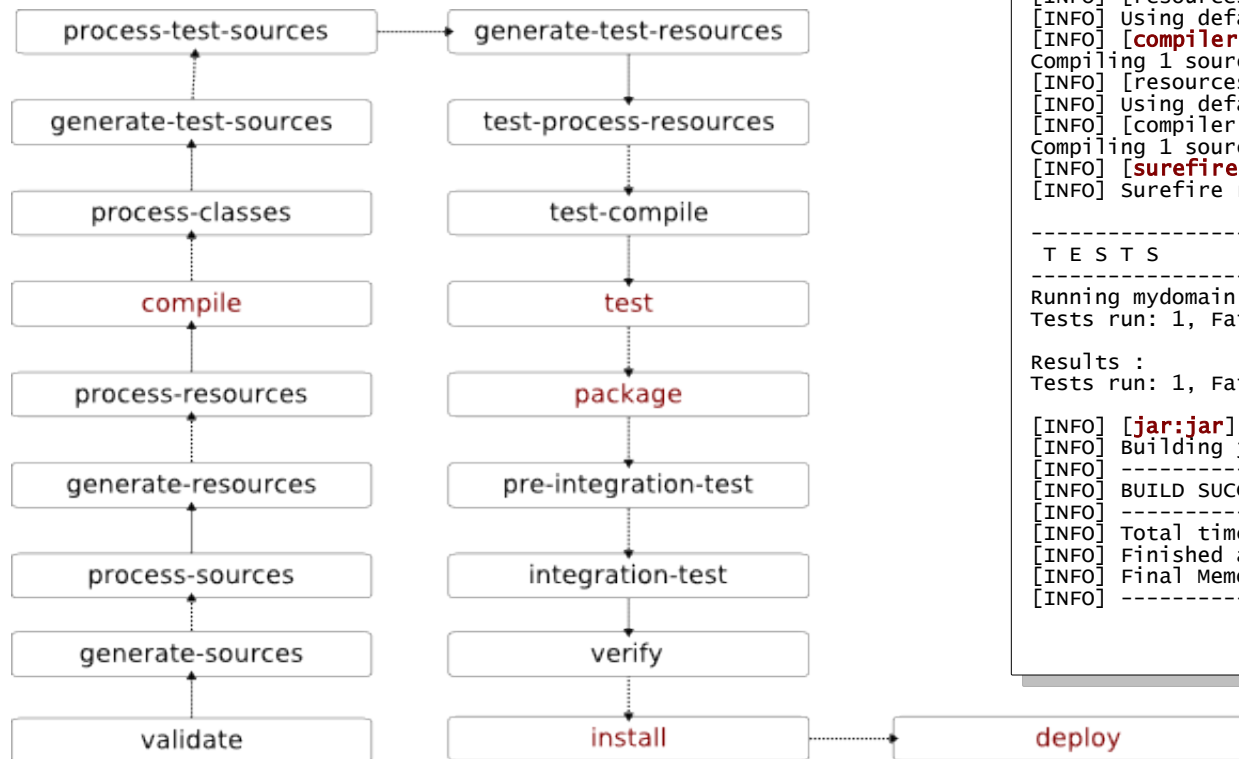
- Abstrakter Task: compile, package, ...
- Teil einer Sequenz von auszuführenden Build Tasks

### Build Plugin

- Konkreter Task: javac, jar, apt, ...

# Build Life Cycle (2)

## Ausführung von 1-n Plugins je Phase



### \$mvn package

```
...
[INFO] Building myapplication
[INFO]   task-segment: [package]
[INFO] -----
[INFO] [resources:resources]
[INFO] Using default encoding to copy filtered r
[INFO] [compiler:compile]
[INFO] Compiling 1 source file to D:\Projects\eclipse-w
[INFO] [resources:testResources]
[INFO] Using default encoding to copy filtered r
[INFO] [compiler:testCompile]
[INFO] Compiling 1 source file to D:\Projects\eclipse-w
[INFO] [surefire:test]
[INFO] Surefire report directory: D:\Projects\ec
-----
T E S T S
-----
Running mydomain.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

Results :
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] [jar:jar]
[INFO] Building jar: D:\Projects\eclipse-workspa
[INFO] -----
[INFO] BUILD SUCCESSFUL
[INFO] -----
[INFO] Total time: 4 seconds
[INFO] Finished at: Tue Jan 16 15:28:09 CET 2007
[INFO] Final Memory: 6M/14M
[INFO] -----
```

# Build Life Cycle (3)

## Plugin-Phase Bindung

- Packaging
- Projektspezifische Plugin Konfiguration (pom.xml)
- Plugin Default Phase (Plugin Implementation)

```
<project>
...
  <groupId>mydomain</groupId>
  <artifactId>myapp</artifactId>
  <version>1.0</version>
  <packaging>jar</packaging>
...
</project>
```

|                        |                         |
|------------------------|-------------------------|
| process-resources      | resources:resources     |
| compile                | compiler:compile        |
| process-test-resources | resources:testResources |
| test-compile           | compiler:testCompile    |
| test                   | surefire:test           |
| package                | jar:jar                 |
| install                | install:install         |
| deploy                 | deploy:deploy           |

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

# Build Life Cycle (4)

## Plugin Konfiguration

- Plugin (Artefakt) Koordinaten
- Pluginspezifische Konfigurationsparameter

## Plugin Aktivierung veranlasst automatischen Download/Update aus Remote Repository

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

# Plugin Konfiguration (1)

## Executions binden ein Plugin Goal und eine Plugin Konfiguration an eine Phase

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-changes-plugin</artifactId>
      <executions>
        <execution>
          <phase>generate-sources</phase>
          <goals>
            <goal>announcement-generate</goal>
          </goals>
          <id>announcement-generate</id>
        </execution>
      </executions>
      <configuration>
        <smtpHost>mail.yourhost.com</smtpHost>
      </configuration>
    </plugin>
  </plugins>
</build>
```

# Plugin Konfiguration (2)

---

**compile jar war ear site** assembly clover  
archetype **ejb surefire deploy install** verifier rar  
pmd jxr ant **release** eclipse idea cargo changes  
changelog doap dependency... ∞

## Plugin Portale

- <http://mojo.codehaus.org/>
- <http://maven.apache.org/plugins/index.html>
- <http://sourceforge.net>

# Resources

---

## Nicht compilierte Dateien

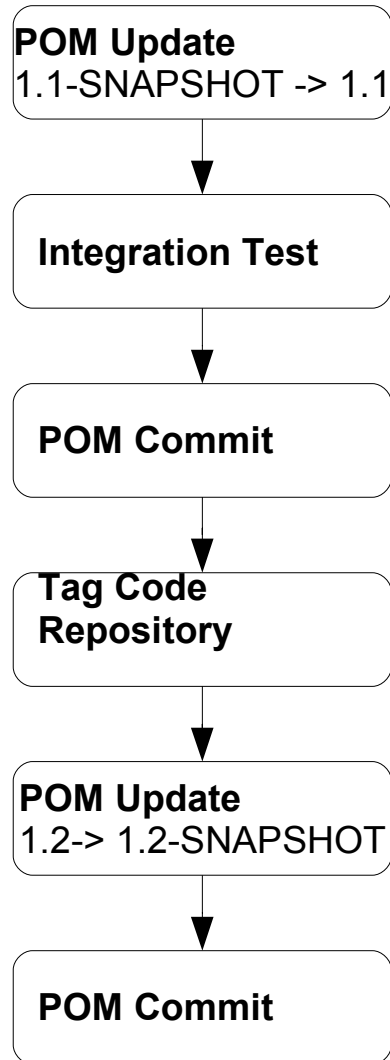
- Konfigurationsdateien (\*.properties, \*.xml, ...)
- Im Artefakt (z.B. Jar) verpackt

## Ersetzung von Platzhaltern (Filter)

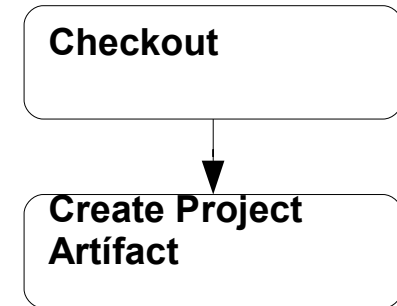
- POM Properties
- System Properties
- Kommandline Properties (Java VM Parameter -D)

# Release Plugin

## Prepare



## Perform



# Release Plugin

```
$ mvn release:prepare
```

```
$ mvn release:perform
```

```
...  
[INFO] [release:prepare]  
[INFO] Verifying that there are no local modifications...  
...  
[INFO] Checking dependencies and plugins for snapshots ...  
What is the release version for "project"?(at...:main) 1.0.3: :  
...  
What is SCM release tag or label? project-1.0.3: :  
What is the new development version? 1.0.4-SNAPSHOT: :  
...  
[INFO] Executing: mvn clean integration-test --no-plugin-updates  
[INFO] Checking in modified POMs...  
[INFO] Tagging release with the label project-1.0.3...  
[INFO] Checking in modified POMs...  
...
```

# Agenda

---

Continuous Integration und Agilität

Projekt Objekt Modell (POM)

Projektstrukturen

Buildprozess

***Reporting***

Continuous Integration Demo

# Reporting und Projekt Dokumentation (1)

Fowler - „Everyone can see what's happening“

Projekt Web Site präsentiert Projekt

Dokumentformate

- xdoc (xhtml)
- apt (Wiki Style)
- fml (FAQ markup language)

I18N Unterstützung

The screenshot shows a Mozilla Firefox browser window displaying the Project Summary page for 'myapplication'. The page is titled 'myapplication - Project Summary' and includes a navigation menu on the left with options like 'Project Information', 'Project Documentation', and 'Project Summary'. The main content area is divided into three sections: 'Project Information', 'Project Organization', and 'Build Information', each containing a table of key-value pairs.

| Field       | Value   |
|-------------|---|
| Name        | myapplication   |
| Description | myapplication   |
| Homepage    | <a href="http://www.martinahrer.at/maven-jar-project/myapplication">http://www.martinahrer.at/maven-jar-project/myapplication</a> |

| Field | Value   |
|-------|---|
| Name  | Martin Ahrer  |
| URL   | <a href="http://www.martinahrer.at">http://www.martinahrer.at</a> |

| Field      | Value         |
|------------|---------------|
| GroupId    | mydomain      |
| ArtifactId | myapplication |
| Version    | 1.0-SNAPSHOT  |
| Type       | jar           |

# Reporting und Projekt Dokumentation (2)

---

## Projekt Reports

- Projektdeskriptor
- Code Analysen
- ...

## Technische Dokumentationen

- Projekt Dokumentation
- Konfigurationsanleitungen
- Handbücher
- ...

## Aggregation einzelner Site Dokumente (Doxia)

- Gesamtdokument (RTF, DocBook, PDF, Tex, ...)

# Reporting und Projekt Dokumentation (3)

## APT Format

### XML Schema Definitions

As the Maven2 project currently lacks XML Schema definitions for XML input files of various plugins we provide them here until they are officially made available by the Maven2 team. So this is just to make life easier until this happens ;)

```
* {{{xsd/changes.xsd}changes.xsd}}
```

```
* {{{#}site.xsd}}
```

```
* ...
```

```
[]
```

[Note:] Maven is not going to validate your XML file against these XSD but you can use them in your favourite IDE to enable features...

# Reporting und Projekt Dokumentation (4)

## Import von Codefragmenten in Site Dokument

...aus dem Deskriptor ausgelesen und mittels Ant Property <<<dependency.test.classpath>>> als Classpath bereit stehen. Die Verwendung des <<<usescope>>> Attributs bewirkt dabei, dass auch Abhängigkeiten mit Scope <<<test>>> mit berücksichtigt werden, dass also JUnit in den Classpath aufgenommen wird.

```
%{snippet|id=init-test|url=file:///d:.\build.xml}
```

Es liegt klar auf der Hand dass es sinnvoll ist, ausschließlich mit ...

```
<!-- START SNIPPET: init-test -->
<artifact:pom id="maven.project" file="${basedir}/pom.xml" />
<artifact:dependencies pathid="dependency.test.classpath"
pomrefid="maven.project" usescope="test" />

<target name="init-test">
  <mkdir dir="${target}/test-classes" />
</target>
<!-- END SNIPPET: init-test -->
```

# Reporting und Projekt Dokumentation (5)

## Site Deskriptor (site.xml)

- Navigations Menü
- Links
- Look and Feel (Skin)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<project name="myapp">
  <body>
    <links>
      <item name="Martin Ahrer" href="http://www.martinahrer.at"/>
    </links>
    <menu ref="reports" inherit="bottom"/>
  </body>
  <skin>
    <groupId>org.apache.maven.skins</groupId>
    <artifactId>maven-stylus-skin</artifactId>
    <version>1.0</version>
  </skin>
</project>
```

# Reporting Plugins (1)

---

**site project-info** changelog changes findbugs  
checkstyle **surefire-report** pmd jdepend javancss  
**clover javadoc taglist jxr jalopy ... ∞**

- Code Metriken
- Coding Style Check
- Quellcode Dokumentation
- Test Abdeckung
- ...

<http://maven.apache.org/plugins/index.html>

# Reporting Plugins (2)

## maven-site-plugin

- Erzeugt Projekt Web Site
- Steuert Report Plugins

```
<build>
  <plugins>
    <plugin>
      <artifactId>maven-site-plugin</artifactId>
      <configuration>
        <locales>en</locales>
        <outputEncoding>ISO-8859-1</outputEncoding>
      </configuration>
    </plugin>
  </plugins>
</build>
```

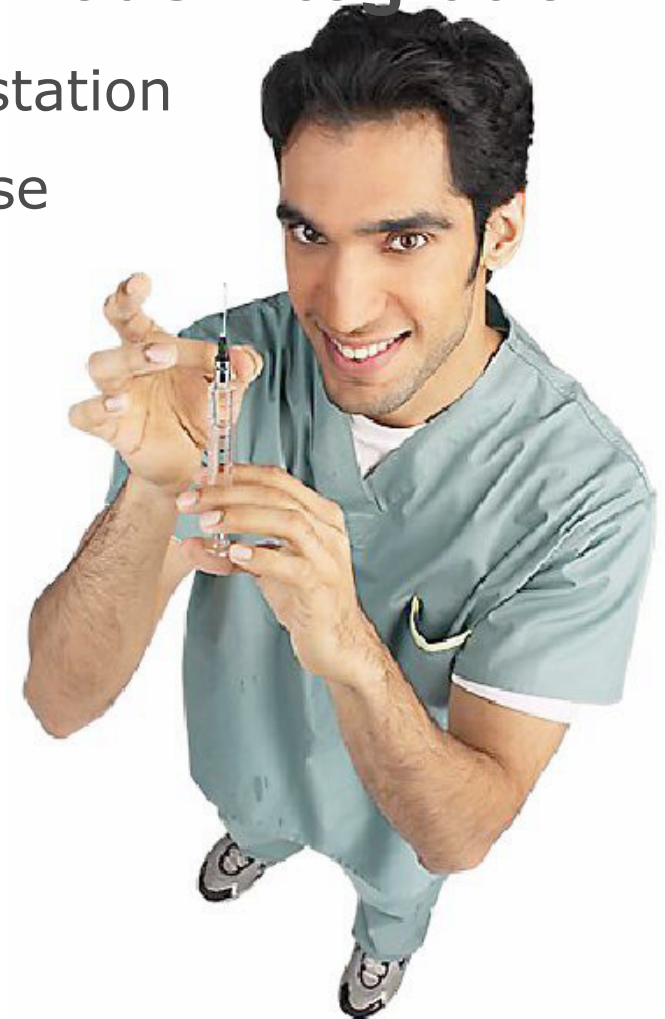
```
<reporting>
  <plugins>
    <plugin>
      <groupId>org.codehaus.mojo</groupId>
      <artifactId>surefire-report-maven-plugin</artifactId>
    </plugin>
  </plugins>
</reporting>
```

...

# Reporting

## Reporting ist die Basis für Continuous Integration

- Intensiv-Information statt Intensivstation
- Tägliche / Stündliche Projekt Analyse
  - Code Analyse
  - Qualitätsprüfung



# Agenda

---

Continuous Integration und Agilität

Projekt Objekt Modell (POM)

Projektstrukturen

Buildprozess

Reporting

***Continuous Integration Demo***

# Tools (1)

## Continuum Build Management Console

- Scheduling
- Log Viewer
- Build Trigger



Current User: Martin Ahrer (admin) - Edit Details - Logout

Continuum | Maven | Apache

### Continuum

About

Show Project Groups

### Add Project

Maven 2.0.x Project

Maven 1.x Project

Ant Project

Shell Project

### Administration

Schedules

Configuration

Appearance

Users

### Legend

Build Now



Build History



Build In Progress



### Project Group Summary

Members

Build Definitions

Notifiers

### Project Group

Project Group Name: CI Demo

Project Group Id: ci-demo

Description: Continuous Integration Demo

### Project Group Actions

Build

Edit

Delete

Release

Add New Project

Add

### Member Projects

|   | Project Name                      | Version | Build    | Group   |   |   |   |   |   |   |
|---|-----------------------------------|---------|----------|---------|---|---|---|---|---|---|
| ✖ | <a href="#">maven-demo-ws-war</a> | 0.1     |          | CI Demo | 🌸 | 📄 | 📄 | 📄 | 📄 | ✖ |
| 🟢 | <a href="#">maven-demo-jar</a>    | 0.1     | <u>1</u> | CI Demo | 🌸 | 📄 | 📄 | 📄 | 📄 | ✖ |

## Tools (2)

# Fowler - „Test in a Clone of the Production Environment“, „Automate Deployment“

## Codehaus Cargo

- Java API, Ant, Maven

## Integrationstest

1. Container Download und Installation
2. Deployment
3. Testausführung



# Tools (3)

---

## SoapUI

- Open Source Projekt
- Web Service Integration Test

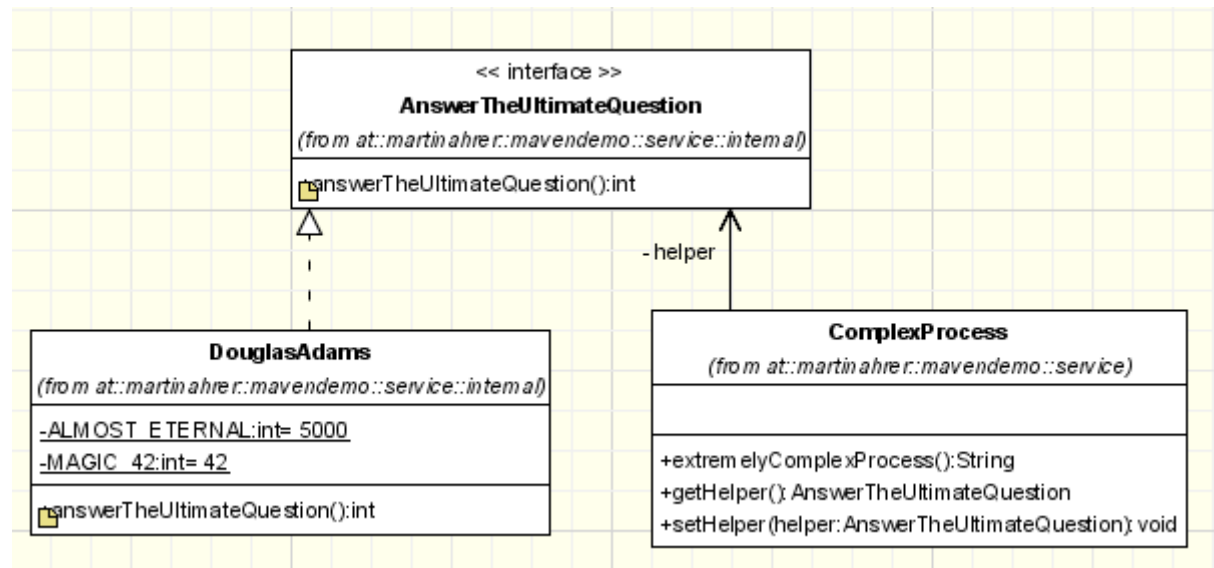
## EasyMock

- Integration Test versus Unit Test

# Demoprojekte

## maven-demo-jar

- JAR Artefakt
- Business Logik
- JUnit Test
- EasyMock Test

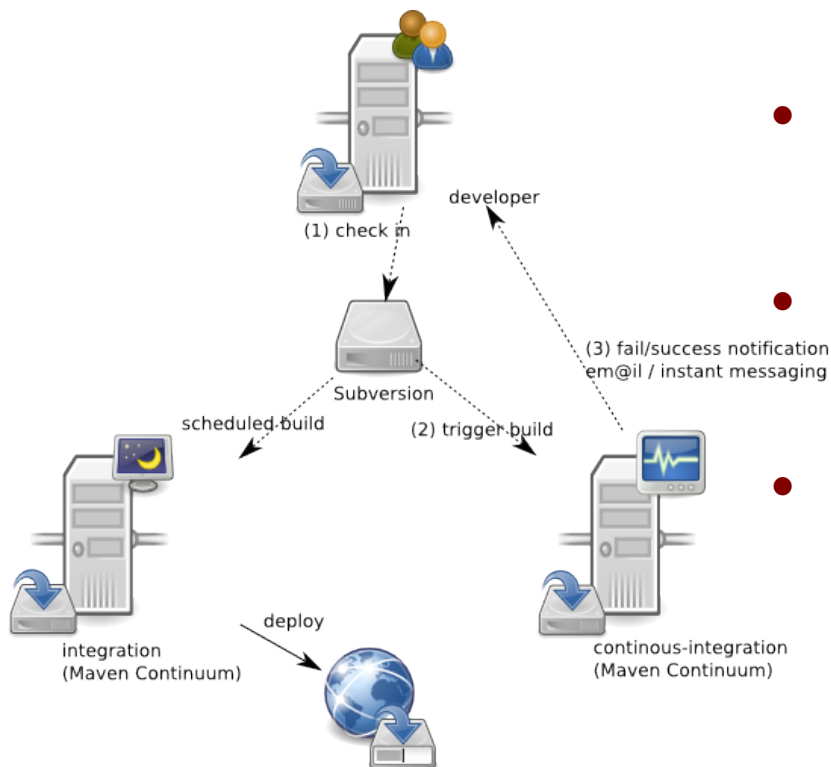


## maven-demo-ws-war

- Axis Web Service Implementation (WAR Artefakt)
- Axis Deployment Deskriptor
- SoapUI Web Service Testdefinition

## Continuous Integration Setup

- Build Prozess
  - Maven-2.0.6
  - Cargo
- Build Management
  - Continuum-1.1-alpha - Tomcat 5.5.x
- Remote Repository
  - WebDAV - Tomcat 5.5.x
- Messaging
  - Apache James Mail Server 2.3.1



# Zusammenfassung

---

**Einfacher Einsatz bewährter Build Werkzeuge**

**Effizientes Management von Abhängigkeiten**

**Unterstützung bei der Einführung eines klaren Buildprozess.**

**Qualitätsverbesserung durch Automatisierung**

**Maven2 ist komplex! Aber Möglichkeit einer rollenorientierten Prozessdefinition!**

# Fragen und Antworten

---

???

**Frage:** Gibt es kommerzielle Unterstützung zu Maven?

**Antwort:** Ja!

**Frage:** Wo gibt es weitere Infos zu dieser Präsentation?

**Antwort:** <http://www.martinahrer.at/oss/m2/maven-best-practice/>