SoftwareEntwicklung ▸ Beratung ▸ Schulung

Der Weisheit letzter Schluss:

Maven 2 in der Java Entwicklung

Web Site:
www.soebes.de
Blog:
blog.soebes.de
Email:
info@soebes.de

Dipl.Ing.(FH) Karl Heinz Marbaise

Agenda

1. Was ist Maven?
2. Features von Maven
3. Grundkonzepte
4. Beispiel
5. Vor- und Nachteile
6. Informationsquellen

2. Installation

- Einfach das Archiv für Windows/Unix von den Maven Seiten [1] herunter laden.
- Auspacken und in den Path aufnehmen.

2. Was ist Maven?

- Build Management Werkzeug
  - Ursprünglich zur Vereinfachung des Build-Prozesses in “Jakarta Turbine” entwickelt.
- Build- und Deployment Werkzeug
- “Software project management and comprehension tool”.

2. Was ist Maven? Features

- Projektübergreifende Vereinheitlichung des Build-Prozesses.
  - Wenn man ein Projekt mit Maven kennt, kennt man alle Projekte mit Maven.
- Einheitlicher Build-Lifecycle

2. Was ist Maven? Features

- Management von komplexen Abhängigkeiten
  - Einfache- als auch transitive Abhängigkeiten [4]
    - Verwendung von log4j-1.2.13.jar
    - Verwendung von Tika-0.3.0.jar, das wiederrum poi-3.5-beta.jar verwendet etc.

2. Was ist Maven? Features

- Erzeugung einer Projekt Site
  - Abhängigkeiten, Reports z.B. Qualitätrelevante Information
    - Unterstützung von Change Logs aus dem Versionskontrollsystem
    - Cross-Referenz der Quellen
    - Mailing Listen
    - Anhängigkeitslisten
    - Unit Test Reporting inkl. Code-Coverage

2. Was ist Maven? Features

- Veröffentlichung der Builds in Repositories (zentral, eigene) und Zugriff auf auf Zentrale oder eigene Repositories.

2. Was ist Maven? Features

- Die “Best-Practice” für Java Projekte
  - Getrennte Aufbewahrung von Test-Code, Produktiv Code, JavaDoc etc.
  - Namenskonvention zur Benenung der Test-Cases
  - Eigenes Test-Setup

2. Was ist Maven? Features

- Erweiterbarkeit durch Plugin-Konzept
  - Es existiert schon eine sehr große Menge von Plugin's [2]
    - EAR, install, License Checks, RPM, ejb, war, clover, changes,...
- Plattformunabhängigkeit
  - Java only

3. Grundkonzepte Prämisse

- Die Prämisse von Maven:

  “Convention over Configuration”

  - Das bedeutet, dass möglichst viel per Konvention festgelegt wird. Nur bei einer eventuellen notwendigen Abweichung ist eine Konfiguration nötig.

3. Grundkonzepte Artefakte

- Identifikation von Artefakten
  - **groupId**
    - Gruppe, Firma, Organisation typischerweise die Domain der Firma, Organisation etc.
  - **artifactId**
    - Name des Artefaktes. Eindeutig innerhalb der groupId

3. Grundkonzepte Artefakte

- Identifikation von Artefakten
  - **version**
    - Version des Artefaktes (z.B. 1.3.2)
  - **classifier**
    - Ergänzung des Artefaktnamens. Typische Anwendung z.B. jdk14, bin, src etc.

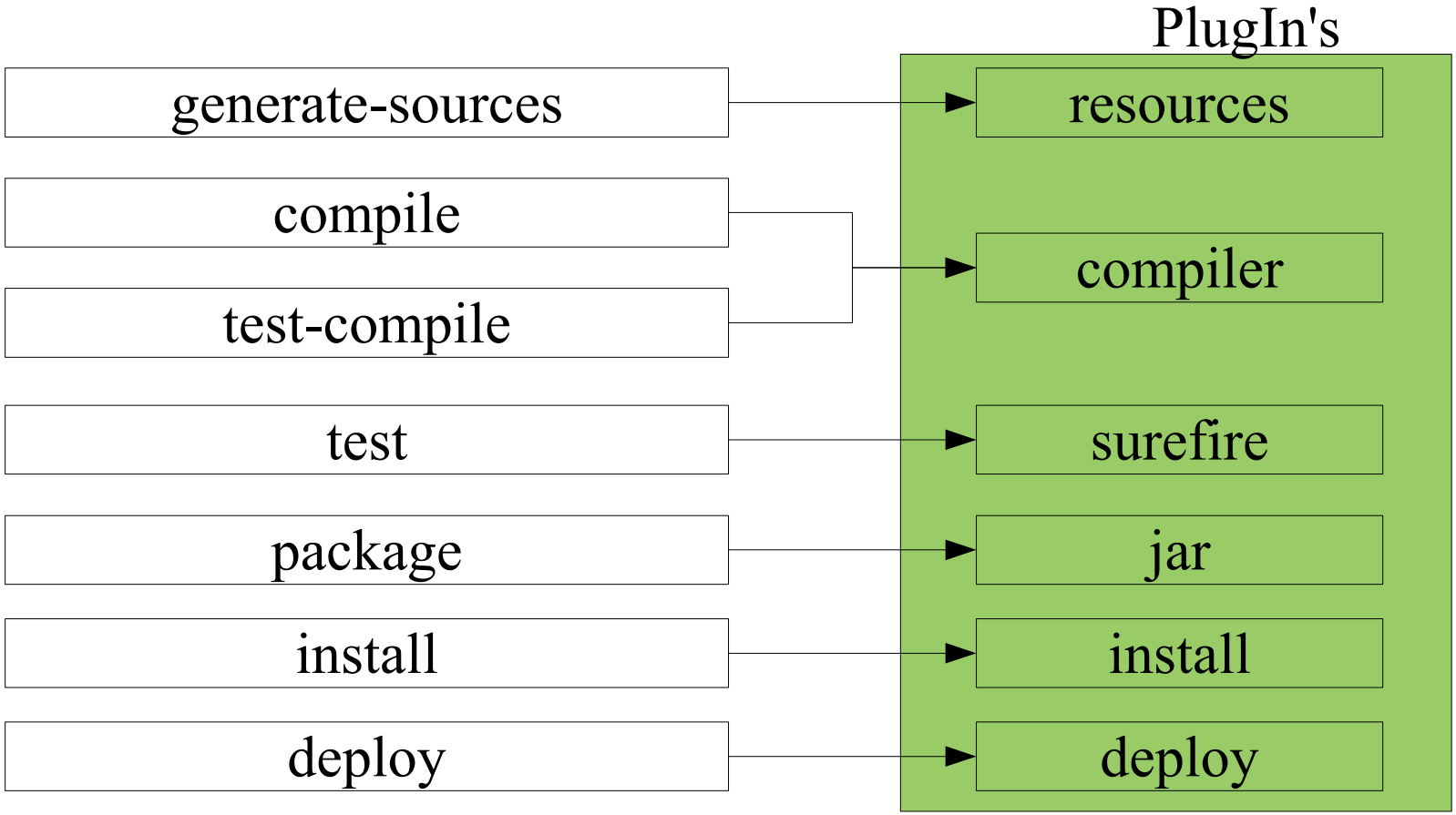
3. Grundkonzepte Lifecycle

- Default Lifecycle (vereinfacht) [3]

Phasen

- validate – Validiert das Projekt
- compile – Kompiliert den Code
- test – Testen des Code per Unit Tests
- package - Erzeugen des Packages
- verify – Prüfung von Q-Merkmalen
- install – Installation in lokales Repos
- deploy – Installation in zentrales Repos.

3. Grundkonzepte
Lifecycle Build Phases



3. Grundkonzepte Beispiel Aufrufe

- Lifecylce

mvn lifecycle

mvn compile
mvn test
mvn install

3. Grundkonzepte Beispiel Aufrufe

- Goal

mvn plugin:goal

mvn compiler:compile
mvn compiler:testCompile
mvn jar:jar

3. Grundkonzepte Beispiel Aufrufe

- Kombinationen

  mvn clean package

  mvn deploy site site:deploy

  mvn release:prepare

  mvn release:perform

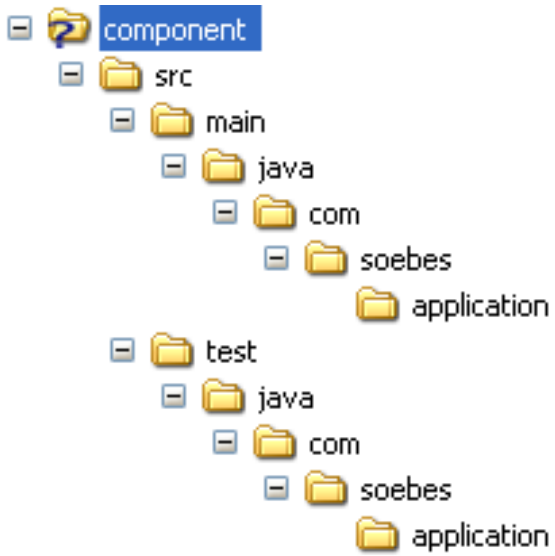
3. Grundkonzepte Beispiel

- Erzeugen eines typischen Layouts [5]

```
mvn archetype:generate \
-DinteractiveMode=false \
-DgroupId=com.soebes.application \
-DartifactId=component
```

3. Grundkonzepte Beispiel

- Erzeugen eines typischen Layouts




```
.\pom.xml
.\src
.\src\main
.\src\main\java
.\src\main\java\com
.\src\main\java\com\soebes
.\src\main\java\com\soebes\application
.\src\main\java\com\soebes\application\App.java
.\src\test
.\src\test\java
.\src\test\java\com
.\src\test\java\com\soebes
.\src\test\java\com\soebes\application
.\src\test\java\com\soebes\application\AppTest.java
```

3. Grundkonzepte
Project Object Model

```
<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>com.soebes.application</groupId>
    <artifactId>component</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <name>component</name>
    <url>http://maven.apache.org</url>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

3. Grundkonzepte
Project Object Model

```
<project .......

    <!-- Basics -->
    <groupId>...</groupId>
    <artifactId>...</artifactId>
    <version>...</version>
    <packaging>...</packaging>
    <dependencies>...</dependencies>
    <parent>...</parent>
    <dependencyManagement>...</dependencyManagement>
    <modules>...</modules>
    <properties>...</properties>
    .
    .
    .
```

Definition der Abhängigkeiten, eventuelle Parents, Module, properties.

3. Grundkonzepte
Project Object Model

```
<project .......

    <!-- Basics -->...

    <!-- Build -->
    <build>...</build>
    <reporting>..</reporting>
    .
    .
    .
```

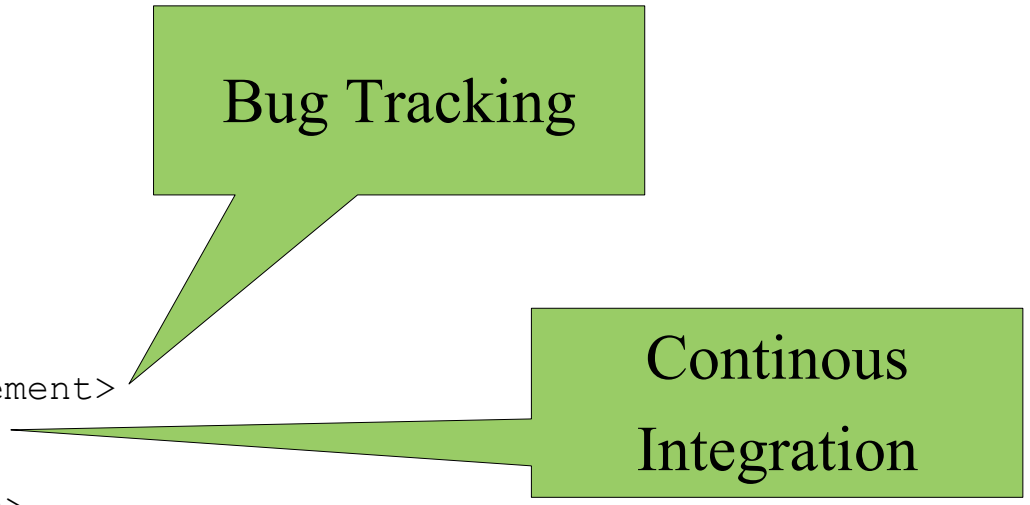
Im **Build** werden die Konfigurationsangaben für PlugIn's abgelegt. Im Bereich **Reporting** werden definitionen und Konfigurationen für das Reporting abgelegt.

3. Grundkonzepte
Project Object Model

```
<project .......

    <!-- Basics -->...

    <!-- Build -->...

    <!-- Project -->
    <name>...</name>
    <url>...</url>
    <inceptionYear>...</inceptionYear>
    <licenses>...</licenses>
    <organization>...</organization>
    <developers>...</developers>
    <contributors>...</contributors>
    <mailingLists>...</mailingLists>
    .
    .
    .
```

3. Grundkonzepte
Project Object Model

```
<project .......

    <!-- Basics -->...

    <!-- Build -->...

    <!-- Project -->...

    <!-- Environment -->
    <issueManagement>...</issueManagement>
    <ciManagement>...</ciManagement>
    <scm>...</scm>
    <prerequisites>...</prerequisites>
    <repositories>...</repositories>
    <pluginRepositories>...</pluginRepositories>
    <distributionManagement>...</distributionManagement>
    <profiles>...</profiles>

</project>
```



3. Grundkonzepte
Project Object Model

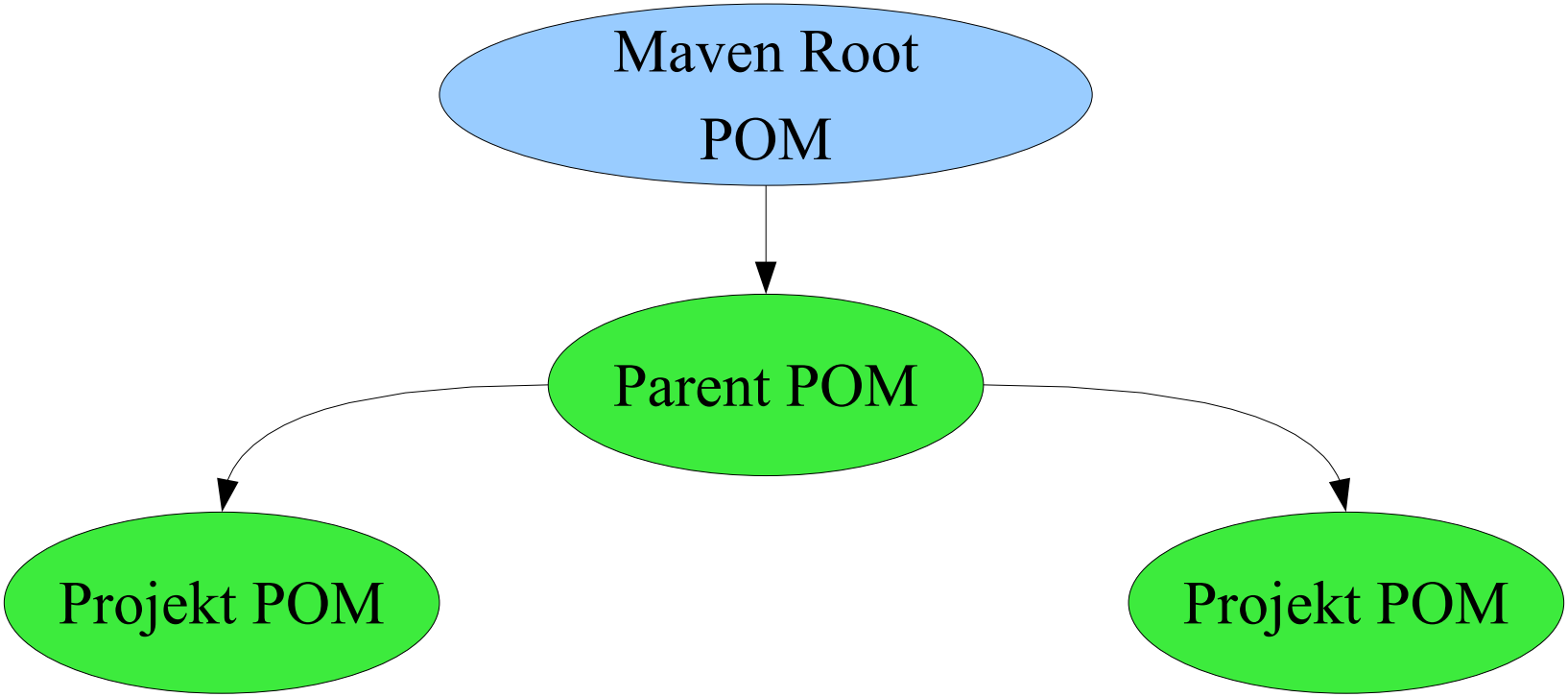
```
<project .......

    <!-- Basics -->...

    <!-- Build -->...

    <!-- Project -->...

    <!-- Environment -->
    <issueManagement>...</issueManagement>
    <ciManagement>...</ciManagement>
    <scm>...</scm>
    <prerequisites>...</prerequisites>
    <repositories>...</repositories>
    <pluginRepositories>...</pluginRepositories>
    <distributionManagement>...</distributionManagement>
    <profiles>...</profiles>

</project>
```

Versionskontrolle

Verteilung
Release/Site

3. Grundkonzepte
Project Object Model

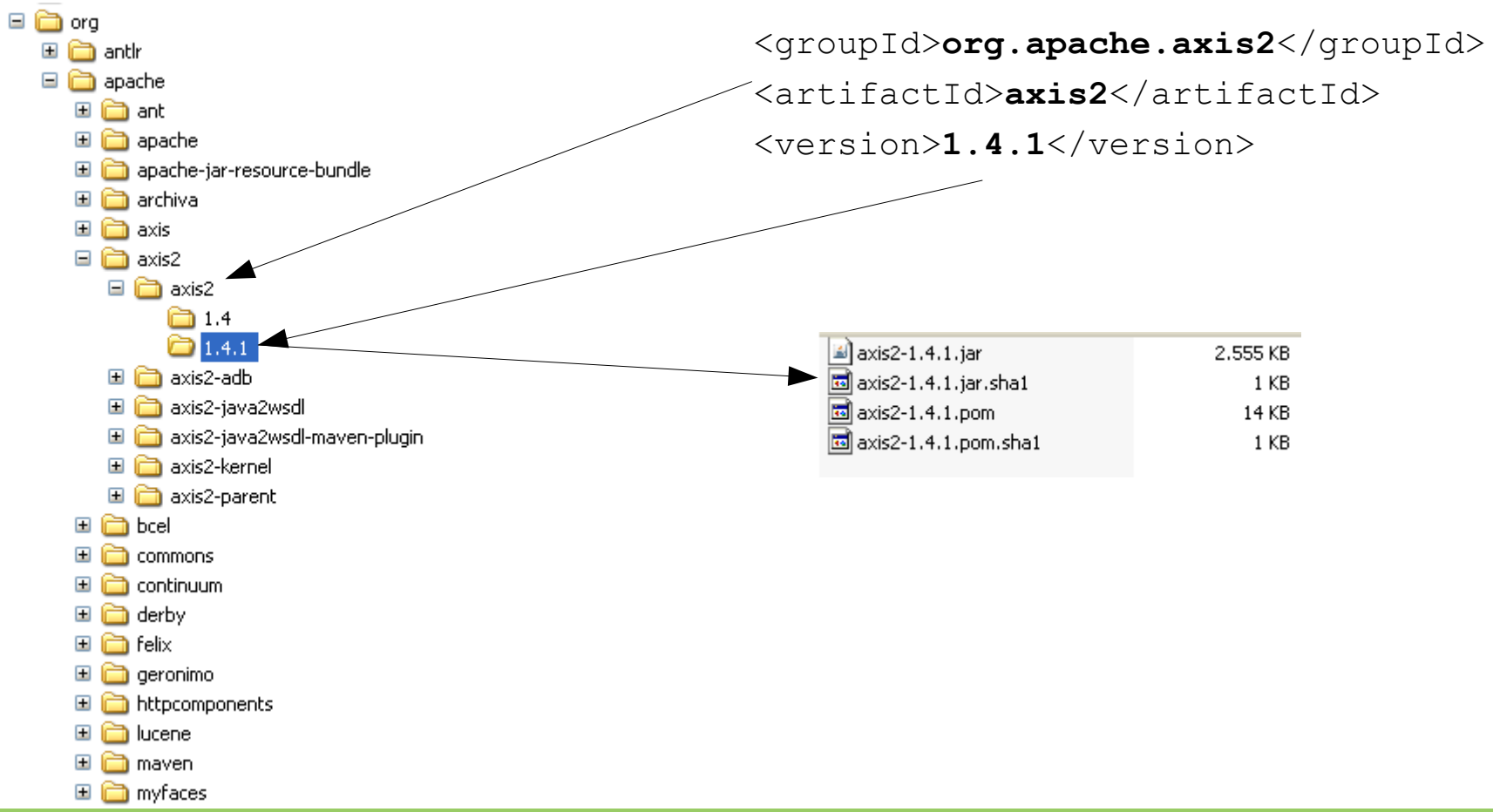
- POM's sind hierarschich gliederbar



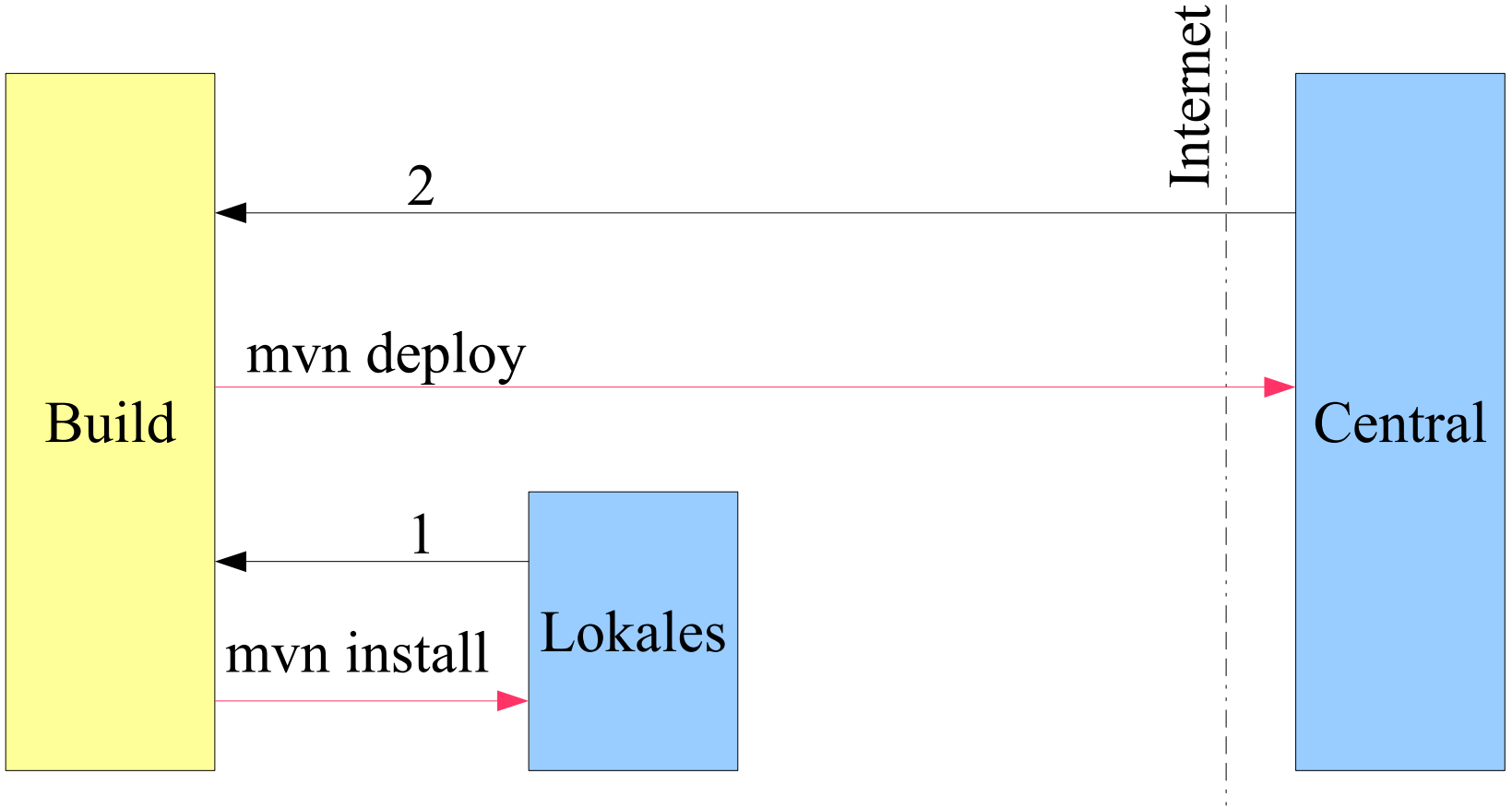
3. Grundkonzepte
Project Object Model

- Folgende Elemente werden vererbt:
  - Abhängigkeiten
  - Entwickler-, Contributoren und Mailing-Listen
  - PlugIn Liste (inkl. reports)
  - PlugIn Ausführungen mit passenden Id's.
  - PlugIn Konfiguration
  - Resourcen

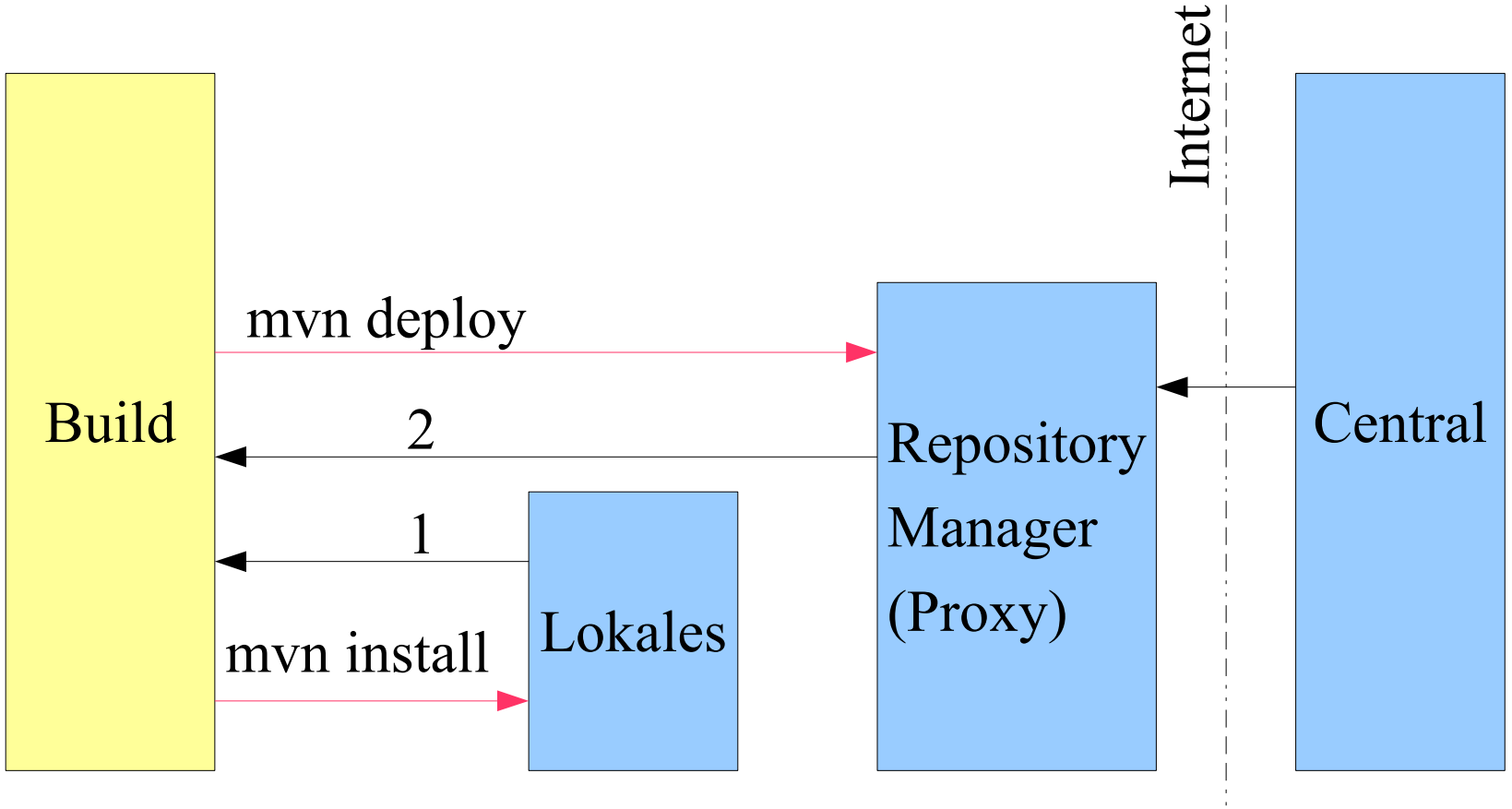
3. Grundkonzepte Repositories



3. Grundkonzepte Repositories (Einfach)



3. Grundkonzepte Repositories (Real)



3. Grundkonzepte Konfiguration

- Benutzerspezifische Anpassungen
  - .m2/settings.xml (User home)
- Globale Konfiguration
  - settings.xml im Maven Programmverzeichnis (conf)
- Profiles in den Projekten
  - profiles.xml

3. Grundkonzepte Module

- Es ist einfach möglich und sehr oft sinnvoll Module zu definieren.
  - Typische Teilung von Projekten
    - z.B.
      - Client
      - Server
      - WSDL
      - WAR
      - CORE
      - Integration Test

4. Beispiele

- Beispiel für einfache POM's ohne Module
- Beispiel für komplexeres Setup mit Modulen und Vererbung

4. Vor- und Nachteile

- Vorteile siehe Features:
  - Vereinfachung der Einarbeitung
  - Wiederverwendung von Komponenten
  - Einfache Nutzung von CI Systemen wie z.B. Hudson, Continuum etc.

4. Vor- und Nachteile

- “Nachteile”:
  - Internet Verbindung notwendig
  - Installation Repository Manager Notwendig
  - Backup des “internen” Repositories bzw. Versionierung
  - Einige Phasen nicht vorhanden:
    - prepare-package erst mit Maven >= 2.1

4. Vor- und Nachteile

- Nachteile:
  - Integrations-Test Phase derzeit nur über ein eigenes Module realisierbar.

Informationquellen

- [1] Homepage of Maven 2
  - http://maven.apache.org
- [2] Maven 2 PlugIn's
  - http://maven.apache.org/plugins/
- [3] Maven 2 Build Lifecycles
  - introduction-to-the-lifecycle.html
- [4] Maven 2 Dependey Mechanism
  - depdency-mechanism.html

Informationquellen

- [5] Archetype Liste
- Mailing Liste (Homepage von Maven)
- Bücher:
  - Maven: The Definitve Guide
  - Better Builds with Maven

Questions?

linuxtag2009@soebes.de

- Thank you for your attention.

Copyright / Urheberecht

Der Vortrag unterliegt der:

GNU Free Documentation License
Version 1.3, 3 November 2008