

Orientation in Objects

Build Management, Teil 2:

Apache Maven

mehr als nur Ant Facelifting

Kristian Köhler
Steffen Schluff

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Version: 1.0

Apache Maven Project
http://maven.apache.org/

Gliederung

Orientation in Objects

- Einführung
- Kleines Maven Wörterbuch
- Maven im Unternehmenseinsatz
- Was bringt die Zukunft?

Gliederung

- **Einführung**
- Kleines Maven Wörterbuch
- Maven im Unternehmenseinsatz
- Was bringt die Zukunft?

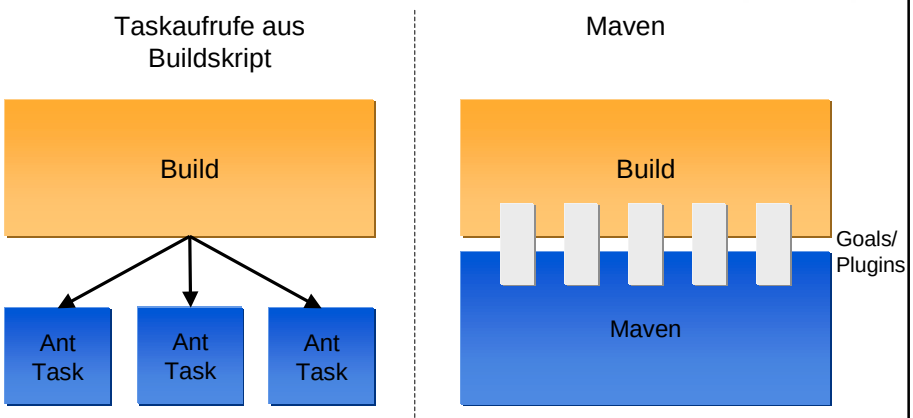
Maven Überblick

- Apache Toplevel Projekt
  - http://maven.apache.org
- Entstehung innerhalb Jakarta Turbine Projekt (2001)
  - viele gleichartige Unterprojekte mit Abhängigkeiten zueinander
- Aktuelle stabile Version: 1.0.2 (7. Dezember 2004)
  - Version 1.1 aus CVS verfügbar
- Maven 2.0 ist angekündigt

Welche Ziele verfolgt Maven?

- Buildprozeß einfach(er) gestalten
  - „Abschottung von Details“
- Einheitliches Buildsystem zur Verfügung stellen
  - Aufbau ist bei allen Projekten gleich
- Einfaches Erstellen von Projektdokumentation
  - implizite Projektinformationen werden aufbereitet
- Gängige Vorgehensweisen werden forciert
  - Bewährte Arbeitsweisen sind integriert
  - Anbindung bestehender Infrastruktur

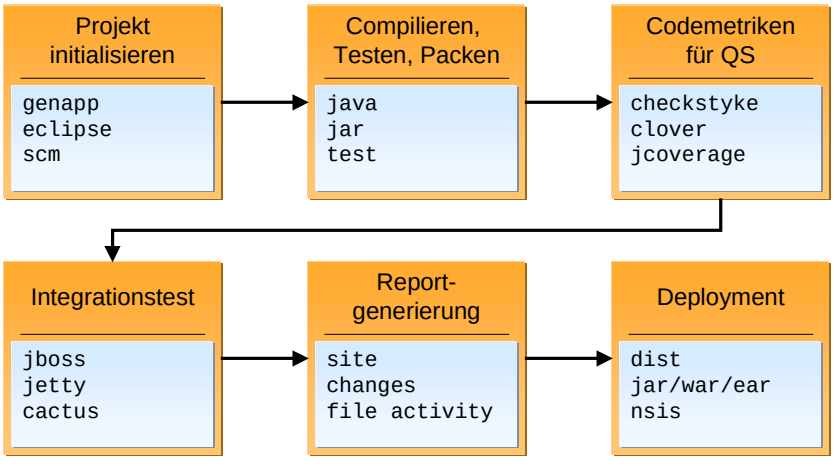
Maven versus Ant



Analogie zu Containerarchitektur

- Inversion of Control Ansatz - Projekt ist „passiv“
  - Maven beschränkt sich auf Projektbeschreibung
  - Bei Ant direkte Befehle: z. B. parametrisierter Aufruf des Compilers
- Maven bietet „Containerdienste“ durch Plugins an
  - Projekt weiß nicht, welche Plugins verfügbar sind
  - Dienste können hinzugefügt und aktualisiert werden
  - Dienste können voneinander abhängig sein
- Maven deckt kompletten Projekt-Lebenszyklus ab
  - Plugin Unterstützung von Projektbeginn bis Deployment

Lebenszyklus eines Maven Projekts



Gliederung

- Einführung
- **Kleines Maven Wörterbuch**
- Maven im Unternehmenseinsatz
- Was bringt die Zukunft?

Zentrale Begriffe

- Project Object Model (POM)
  - Metadaten zur vollständigen Beschreibung des Projektes
- Plugins
  - stellen eigentliche Funktionalität zur Verfügung
- Repository
  - zentrale strukturierte Ablage von benötigten Daten (z. B. Bibliotheken)

Project Object Model (POM)

- Beschrieben durch XML Datei „project.xml“ im Projektverzeichnis
- Enthält Allgemeines zum Projekt
  - Name, Organisation, Sourcecode Repository, etc.
- Beinhaltet Abhängigkeiten des Projektes
  - Sämtliche benötigten Bibliotheken, Plugins, Taglibs, etc.
- Beschreibung der „Buildumgebung“ des Projektes
  - Verzeichnislayout für Sourcecode, Testcode, Ressourcen, etc.
- Angaben zur Reportgenerierung

Plugins und Goals

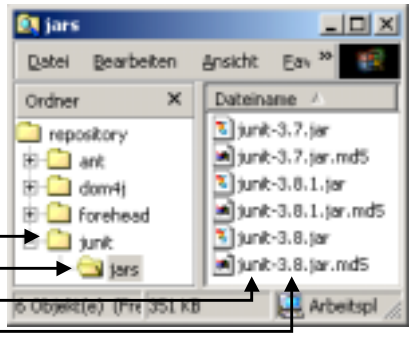
- Eigentliche Funktionalität von Maven in Plugins enthalten
  - java, jar, eclipse, test, etc.
- Plugins stellen Goals zur Verfügung
  - Werten Metadaten des Projektes aus, z. B. Verzeichnislayout
  - Entsprechen vorkonfigurierten Targets in Ant
- Änderung des Standardverhaltens über Property Dateien
- Sind in Jelly geschrieben
  - Durch optionale Javafunktionalität erweiterbar

Repositories und Artefakte (1)

- Ein Projekt erzeugt jeweils ein Artefakt
  - Mögliche Artefakte: JARs, Plugins, Binary distribution, etc.
- Projekte benötigen selbst fremde Artefakte
- Artefakte werden projektübergreifend in Repository abgelegt
  - Artefakte müssen versioniert sein
  - Unterscheidung Release-Version und „SNAPSHOT“
- Entbindet von projektbezogener Ablage in SCM
  - Artefakte meist große nicht „eigenversionierte“ Binärdateien

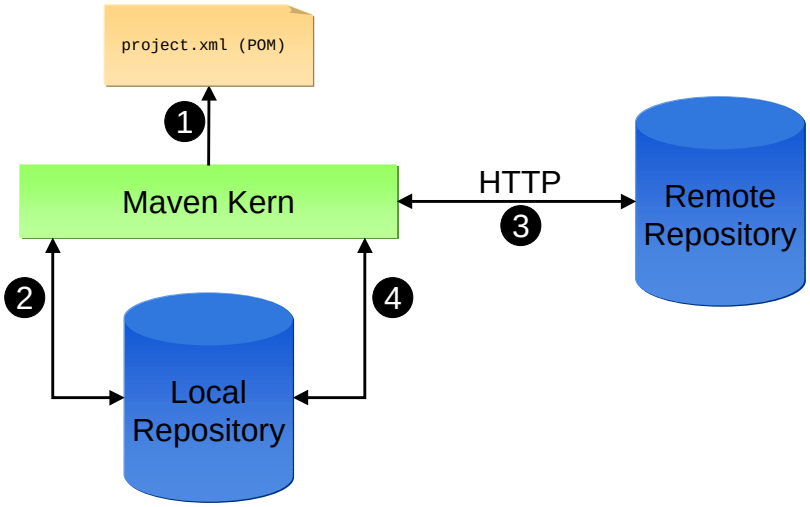
Repositories und Artefakte (2)

- Jedes Artefakt ist eindeutig über Bezeichner adressierbar
  - GroupId
  - Type
  - ArtifactId
  - Version

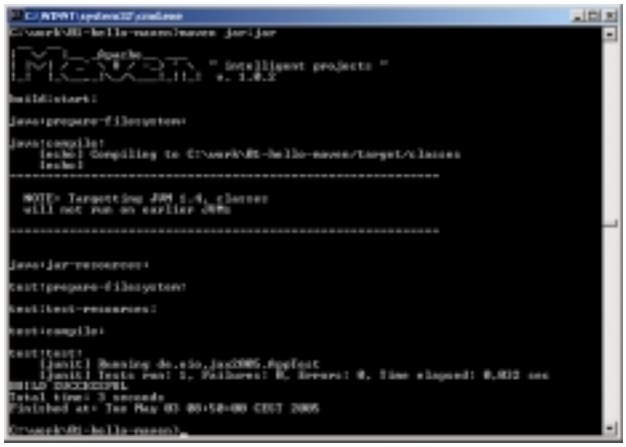


- Unterscheidung lokale und entfernte Repositories
  - Zentrale gepflegte Repositories verfügbar z. B. ibiblio.org

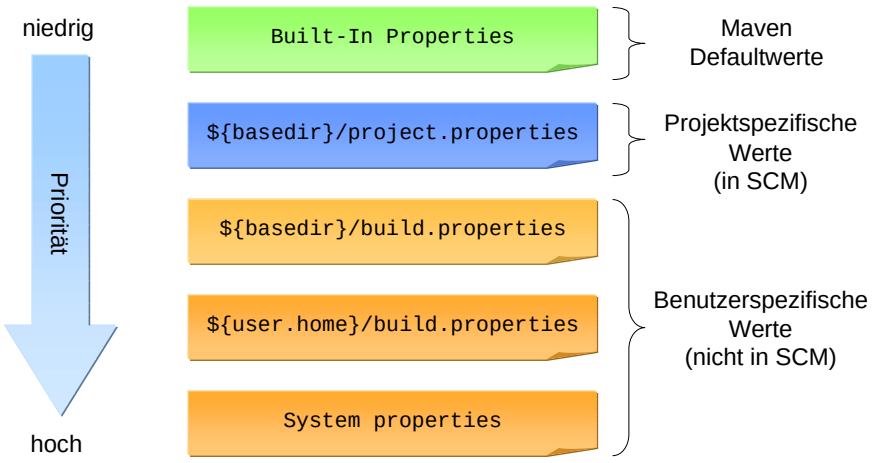
Maven Repositories



First Look - erstes Beispielprojekt



Property Verarbeitung



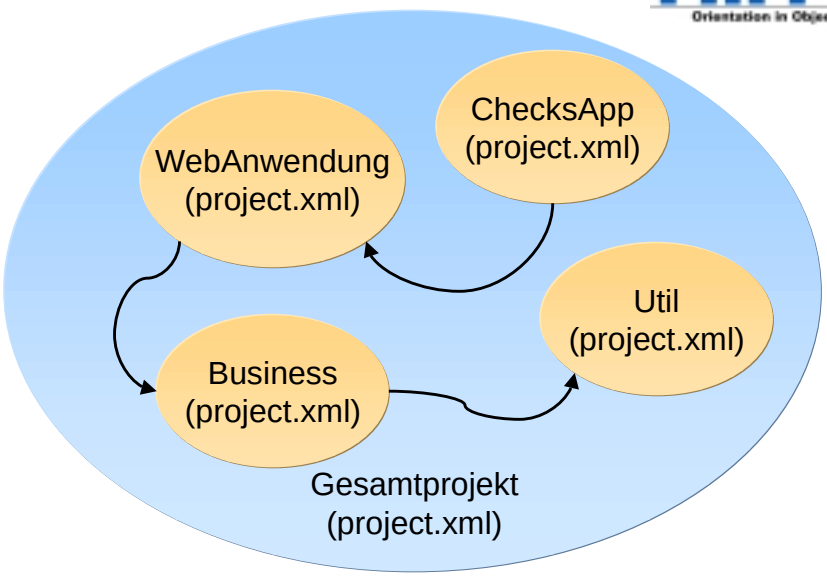
Gliederung

- Einführung
- Kleines Maven Wörterbuch
- **Maven im Unternehmenseinsatz**
- Was bringt die Zukunft?

Multiprojekt Motivation

- Komplexe Projekte benötigen logische Unterteilung
  - Separation of concerns
  - Util, Common, EJB-Schicht, Web-Schicht, etc.
- Unterprojekte teilen viel Information
  - Jedes Unterprojekt erzeugt eigenes Artefakt
- Sicht auf Gesamtprojekt soll bestehen bleiben
  - Einzelne Artefakte werden zu gesamt Artefakt assembliert
  - Integrationstests, Reportgenerierung, Distribution

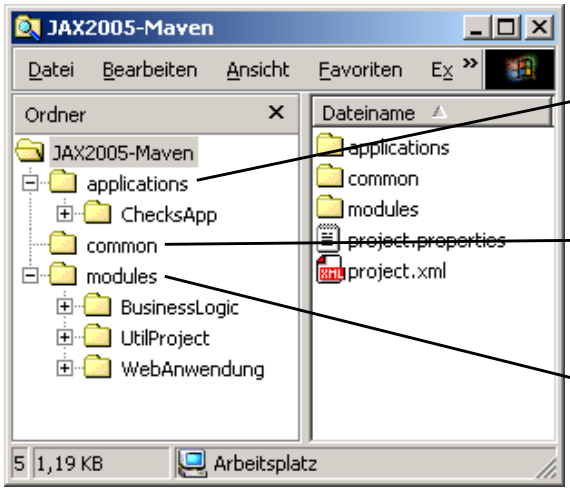
Komplexes Projekt besteht aus Teilprojekten



Multiproject Plugin

- Einheitliche Sicht auf Gesamtprojekt
- Lädt Project Object Models (POM) aus Teilprojekten
  - Auswahl der Teilprojekte über Properties
  - Default: */project.xml
- Berechnet Reihenfolge der Abarbeitung der Teilprojekte
  - Aus Abhängigkeiten der Teilprojekten zueinander
- Aus Sicht des Teilprojektes Einbindung transparent
  - Abhängigkeiten zu Teilprojekten oder anderen Artefakten gleichwertig

Typisches J2EE Projekt

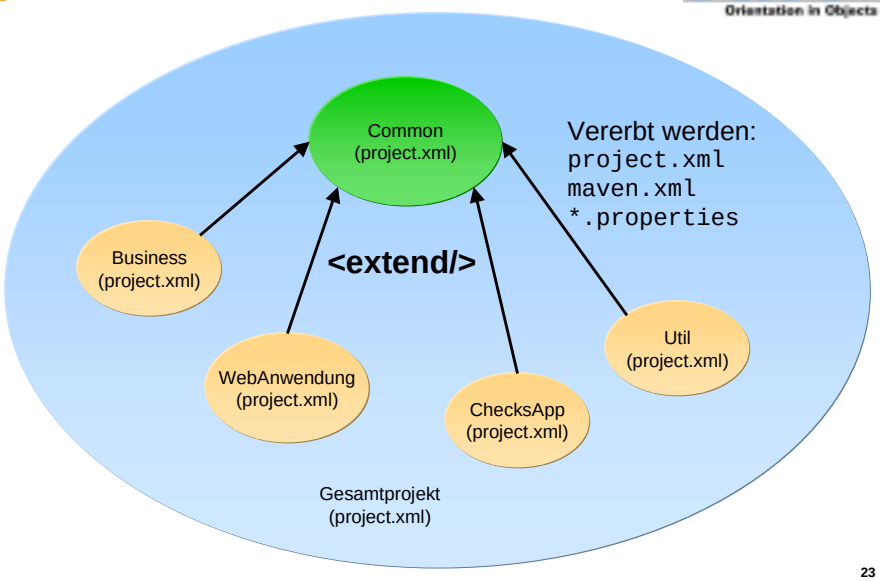


Applications
Assemblierungsinfos, die aus „*Modules*“die eigentlichen Anwendungen zusammen stellen.

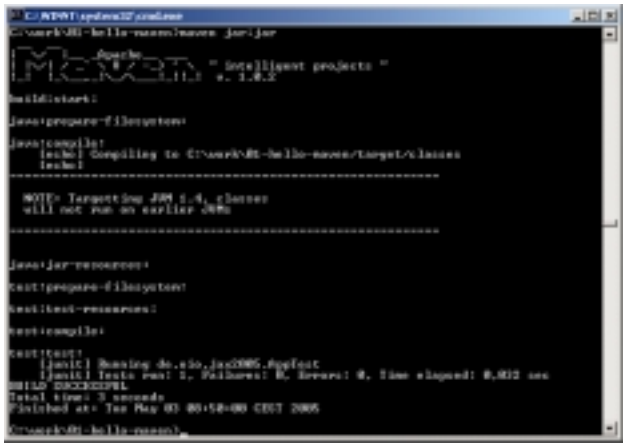
Common
gemeinsam benutzte Dateien und Konfigurationen.
z. B. „Global POM“

Modules
Teilprojekte. Logische Einheiten. Produzieren einzelne Artefakte

Vererbung bei Multiproject



Demo II Multiproject



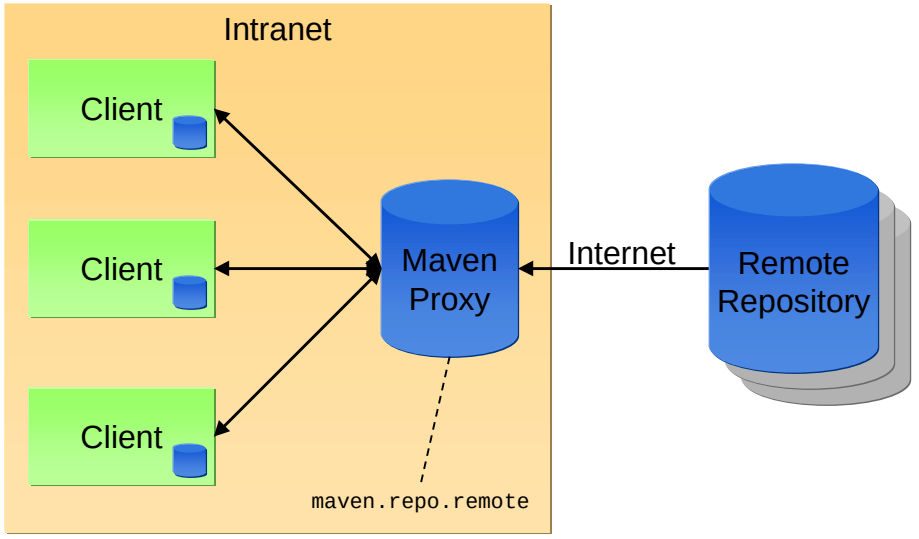
Anpassungen mittels maven.xml

- Erweiterungsmöglichkeit des Buildvorgangs
  - basierend auf XML Skriptsprache Jelly
  - einfache Einbindung von Ant tasks
- Möglichkeit der Vor- und Nachverarbeitung von Goals
  - Definition über Hooks („preGoal“ und „postGoal“)
- Definition eigener Goals möglich
  - Verwendung bestehender Goals über „prereqs“ bzw. „attainGoal“
  - Dokumentierbar über „description“ Attribut für „maven --usage“
- „*You should try not to have a maven.xml file.*“
  - http://maven.apache.org/using/bestpractices.html

Stolpersteine eines Unternehmenseinsatzes

- Ständiger Zugriff auf externes Repository
  - Zeitverzögerung, Downloadvolumen, etc.
- Verteilung Kommerzielle Bibliotheken
  - nicht in externen Repository verfügbar
- Einbindung in Continuous Integration
  - Maven Continuum, CruiseControl
- SNAPSHOT Abhängigkeiten bei Auslieferungen
- „One-Click build“ und Archivierung

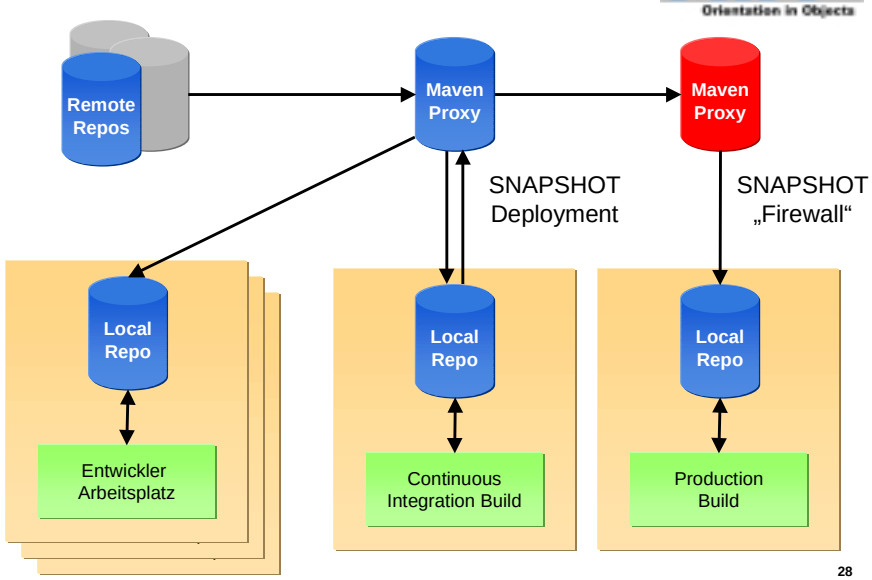
Maven Proxy



27

Apache Maven - mehr als Ant Facelifting

Mehrere Repositories



28

Apache Maven - mehr als Ant Facelifting

Gliederung

- Einführung
- Kleines Maven Wörterbuch
- Maven im Unternehmenseinsatz
- **Was bringt die Zukunft?**

Maven 2

- Vollständige Neuimplementierung
  - Version alpha 1 als Download verfügbar
  - Release Version geplant für August 2005
  - http://maven.apache.org/maven2
- Schwerpunkte sind Geschwindigkeit und Benutzerfreundlichkeit
  - Neue Architektur, schneller und weniger Speicherverbrauch
  - Maven soll einfacher und konsistenter werden
- „Unfortunately, to reach these goals - we've had to sacrifice backwards compatibility to do this“
  - http://maven.apache.org/maven2/about.html

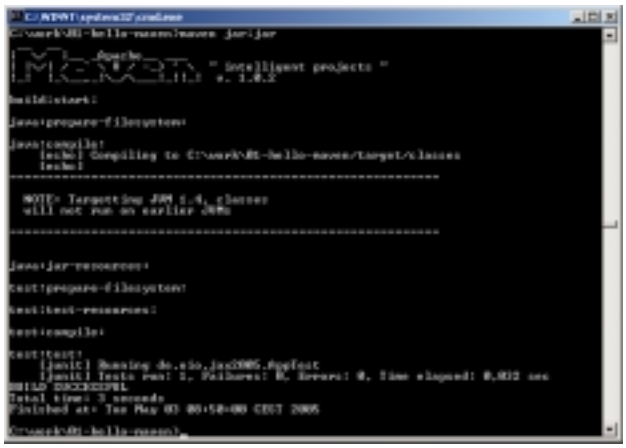
Was bringt die Zukunft? (I)

- Multiproject bereits eingebaut
  - Gebräuchlichster Anwendungsfall in J2EE Projekten
- Verzicht auf properties-Dateien
  - Konfiguration geschieht vollständig im POM
- Verzicht auf „maven.xml“ und Jelly
  - Plugins zukünftig in Java und Skriptsprache Marmalade
- Bessere Behandlung von SNAPSHOTs
  - Konfigurierbares Update Verhalten

Was bringt die Zukunft? (II)

- Verbesserter Dependency Mechanismus
  - Verkettete Abhängigkeitsverwaltung
  - Sichtbarkeit abhängiger Artefakte („compile“, „runtime“, „test“)
  - Versionierte POM Vererbung
- Änderungen im Repository Layout
  - POMs werden mitverwaltet
  - Versionen bilden sich auf Verzeichnisse ab
  - Pläne einer Verwaltungsanwendung mit Suchfunktionalität
- Kleiner und schneller
  - As always

Demo III Maven 2

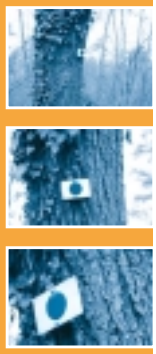


Wer baut mit Maven?

- Turbine
  - http://jakarta.apache.org/turbine/index.html
- Apache Geronimo
  - http://geronimo.apache.org
- Jetspeed
  - http://portals.apache.org/jetspeed-1/
- ... und bald auch Sie?

Fazit

- Contra
  - Dokumentation vorhanden aber unübersichtlich
  - erste „echte“ Probleme können zu Frustration führen
  - noch mehr Skriptsprachen - Jelly bzw. Marmalade
  - Maven 2 bereits absehbar
- Pro
  - Bewährte Vorgehensweisen werden forciert
  - Einheitlicher Verzeichnisaufbau bei allen Projekten
  - Konsistente Namen für Aufgaben: „compile“, „clean“, „war“, etc.
  - Deklarative Projektbeschreibung anstelle immer gleicher Buildskripte
  - Automatisches Herunterladen von abhängigen Artefakten
  - Hohe Produktivität erreichbar



Fragen ?

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Version: 1.0

Orientation in Objects

Vielen Dank für Ihre Aufmerksamkeit !

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Version: 1.0