Tracing von (Open-Source) Projekten mit Maven

Tammo van Lessen, Steffen Pingel

Fachgruppe IT Projekt-Management

Fakultät für Elektrotechnik und Informatik
Universität Stuttgart

inf.misc, 04.02.2004

1. Project Tracing
2. Basics
3. Plugins
4. Advanced Stuff
5. Fazit

Teil I

Project Tracing

Project Tracing

Das Ziel von Project Tracing ist die Verfolgung bestimmter Projektparameter:

Wo steht mein Projekt in diesem Moment?

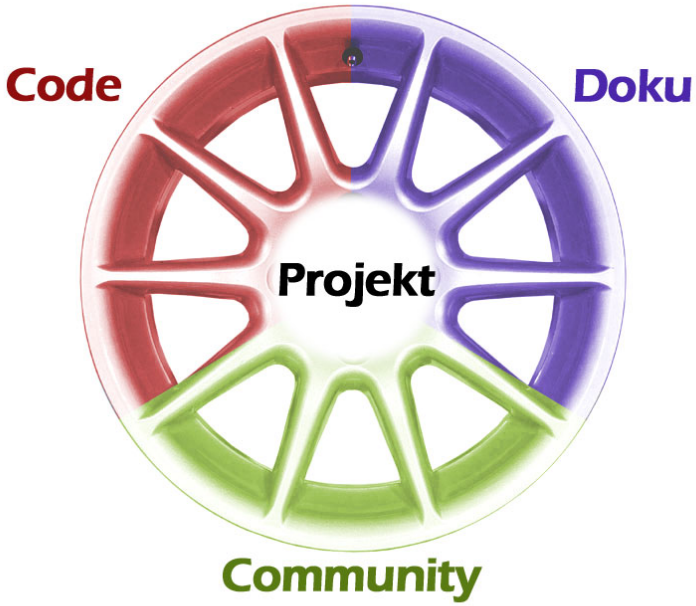
- BWL-Sicht: Zeit, Qualität, Kosten, Funktions-Umfang
- Entwickler-Sicht: Velocity, Dokumentation, Metriken

The Apache Way of Life

Ein Projekt als Hülle für:

- Community
- Dokumentation
- Code

Alle Elemente zusammen machen erfolgreiche Projekte möglich.



Tracing von Open-Source Projekten

- Einheitliche Dokumentation
- Architektur sichtbar machen
- *Wer* hat als letztes *was*, *wo* gemacht?
- Qualitätskontrolle durch Source-Metriken
- Überblick über die verwendeten Bibliotheken

Das alles, und noch viel mehr: Mit **Apache Maven** in den Buildprozess integriert.

Teil II

Maven Basics

Apache Maven

- Build-Werzeug mit Projektverständnis
- In Java entwickelt
- Gestartet von Jason van Zyl als Subprojekt von Apache Turbine
- Inzwischen Toplevel-Projekt bei der ASF
- Basiert auf Jelly (XML Scripting Engine)
- Plugin-Architektur

Aktuelle Version ist 1.0-rc1 (Stand: Februar 2004)

Installation von Maven

1. Download von http://maven.apache.org/start/download.html
2. Entpacken (z.B. nach /usr/share/maven-1.0rc1)
3. Umgebungsvariablen anpassen
   - `$MAVEN_HOME=/usr/share/maven-1.0rc1`
   - `$PATH` erweitern um `$MAVEN_HOME/bin`
4. (Optional) Lokales Repository erstellen
   - `$MAVEN_HOME/bin/install_repo.sh $HOME/.maven/repository`

Neues Projekt anlegen

Es gibt 2 Möglichkeiten, mit Maven zu beginnen:

- Manuell
- Automatisch

Projekt anlegen lassen

vanto@marvin > maven genapp

Demonstration

Neues Projekt anlegen

Es gibt 2 Möglichkeiten, mit Maven zu beginnen:

- Manuell
- Automatisch

Projekt anlegen lassen

vanto@marvin > maven genapp

Demonstration

Verzeichnisstruktur

Maven wünscht sich eine besondere Projektgliederung

```
statcvs-xml
  + src
    + java
    + test
  + xdocs

  - maven.xml
  - project.xml
  - project.properties
```

Projekt Object Model (POM)

Das POM wird in der Datei project.xml im Projektverzeichnis definiert und enthält Projekt-Metadaten:

- Projektname, -version, -beschreibung, -lizenz
- URL der Projekt-Homepage, des Issue-Trackers und der Mailinglisten
- CVS-Repository (Release Tags, Branches)
- Projekt-Team Mitglieder
- Abhängigkeiten (Bibliotheken)
- (Re-)source-Verzeichnisse
- Liste der Projekt-Berichte

maven.xml

Enthält alle Projekt spezifischen Build-Targets, sogenannte **Goals**.

- Goals sind in Jelly-XML definierte Funktionen
- Jedes Goal hat ein Pre- und Post-Goal
- Start von der Konsole
- Start aus anderen Goals heraus

Beispiel

```
<goal name="site-update" description="Updates the web site">
    <cvs command="-q update -Pd"/>
    <attainGoal name="clean"/>
    <attainGoal name="site:deploy"/>
</goal>
```

Plugin-Architektur

Everything is a plugin

Plugins sind konfigurierbare Goal-Bibliotheken.

- Jedes Plugin besteht aus einer plugin.jelly Datei
- plugin.properties setzt die Plugin-Defaultwerte
- maven.xml wird durch alle plugin.jelly Dateien ergänzt

Konfigurierbarkeit

(Fast) alle Plugin-Parameter lassen sich anpassen.

- Definition in Property-Files
- Vererbung der Properties
  1. `plugin.properties`
  2. `$maven.home/bin/driver.properties`
  3. `$project.home/project.properties`
  4. `$project.home/build.properties`
  5. `$user.home/build.properties`

Lokale Anpassungen nur in den letzten beiden!

Repositories

Maven übernimmt die Verwaltung aller im Projekt verwendeten Bibliotheken.

- Automatischer Download aus dem remote Repository in das lokale Repository
- Ergänzung des Classpaths

Dokumentation

Sämtliche Projektdokumentation erfolgt mit **xdocs**

- Erweitertes (X)HTML
- Muss well-formed sein
- Kann jedoch nicht validiert werden
- Ermöglicht einfache Gliederung

Beispiel

```
<document>
  <properties>
    <title>Overview</title>
  </properties>
  <body>
    <section name="Overview">
      <p>The project goal is to develop a graphical user interface
         for programming a CPLD. The software is written in C++ and
         based on QT.</p>
      ...
```

Buildprozess starten

Java Plugin

java:compile	Kompiliert die im POM angegebenen Sourcen

Jar Plugin

jar:jar	Erzeugt aus den im POM angegebenen Sourcen und Resourcen ein Jar (z.B. statcvs-xml-1.0.jar)
jar:snapshot	Erzeugt ein Snapshot-Jar ohne Version, dafür mit Datum (z.B. statcvs-xml-20040204.140000.jar)
jar:install	(Erzeugt und) Deponiert das Jar im lokalen Repository

Alle Jar-Dateien werden im ./targets Verzeichnis abgelegt.

Release erstellen

Distribution Plugin

dist	Kompiliert die Sourcen und erzeugt die Source- und Binary-Distributionen (z.B. statcvs-xml-1.0.tar.gz)
dist:build-src	Erzeugt eine Source-Distribution
dist:build-bin	Erzeugt eine Binary-Distribution

Alle Ergebnisse werden im ./targets/distributions Verzeichnis abgelegt.

Dokumentation erstellen und deployen

XDoc Plugin

xdoc	Erzeugt aus den xdoc-Dateien HTML-Dokumente

Site Plugin

site	Generiert die Reports der aktivierten Plugins
site:deploy	Generiert und kopiert die Projektdokumentation per ssh auf den im POM angegebenen Webserver
site:fsdeploy	wie site:deploy, allerdings lokal im Filesystem

Teil III

Maven Plugins

Core Plugin

Die Core Plugins werden mit Maven ausgeliefert.

- *Clean*
- *Jar*
- *Java*
- License
- Multi-Project
- Plugin
- Test
- Touchstone
- Touchstone Partner

Optionale Plugins

Die Optionalen Plugins werden bei Bedarf automatisch installiert. Auf der Maven Seite sind 77 Optionale Plugins gelistet (Stand: 25. Januar 2004).

- Dist
- Genapp
- LinkCheck
- Release
- Site
- XDoc
- ...

Die nachfolgend vorgestellten Source Code Plugins sind Java spezifisch.

Source Code Dokumentation

JavaDoc Plugin

- Generiert JavaDoc Dokumentation

Konfiguration (project.properties)

```
maven.javadoc.links
  = http://java.sun.com/j2se/1.4/docs/api/
```

JXR Plugin

- Generiert Querverbunde Sourcen
- Auch als *Cross-Reference* oder *XRef* bekannt

Style Guide Checking

CheckStyle Plugin

- Prüft den Source Code anhand von Regeln
- Bericht enthält Links auf Fehlerzeile im XRef
- Konfiguration über eine Properties Datei
- Mit Maven 1.0-rc1 ausgelieferte Version benutzt veraltetes CheckStyle 2.1 (CVS Version benutzt bereits 3.3)

Konfiguration (project.properties)

```
maven.checkstyle.format = (sun|turbine|avalon)
maven.checkstyle.properties = mystyle.properties
```

Style Guide Checking

Jalopy Plugin

- Formatiert Source Code anhand von Regeln
- Transformiert unter anderem
  - Geschweifte Klammern
  - Leerzeichen und -zeilen
  - Einrückungen
  - Zeilenumbrüche
- Kann JavaDoc Kommentare, Kopf- und Fusszeilen generieren

Konfiguration (project.properties)

```
maven.jalopy.style = mystyle.xml
```

Source Code Metriken

PMD Plugin

- Prüft den Source Code anhand von Regeln auf potientielle Fehler
- Findet unter anderem
  - Unbenutzte Variablen
  - Leere catch-Blöcke
  - Leere if-Anweisungen,
  - Duplizierte import Anweisungen
  - Nicht benutzte private Methoden
  - Cyclomatic Complexity
- Anpassung über vordefinierte Regeldateien

Konfiguration (project.properties)

```
maven.pmd.rulesetfiles
  = $plugin.resources/rulesets/strings.xml,
    $plugin.resources/rulesets/junit.xml, ...
```

Source Code Metriken

JDepend Plugin

- Prüft die Entwurfs Qualität von Source Code anhand von Regeln
- Misst die
  - Erweiterbarkeit
  - Wiederverwenbarkeit
  - Wartbarkeit
- Zeigt Metriken pro Packet an

Je nach Anzahl der Pakete und Fehler kann der Berichte sehr gross werden (z.B. XNap mit ca. 65.000 SLOC: 742 kb).

Test und Überdeckung

JUnit Report Plugin

- Erstellt einen Bericht über die Ausführung der Testfälle

Clover Plugin

- Misst die Testüberdeckung
- Automatische Instrumentierung des Source Codes
- Messung der Anweisungs- und Zweigüberdeckung
- HTML Ausgabe im Frameset ähnlich JavaDoc
- Frameset nicht in die Projekt Seite integriert
- Clover ist nicht frei
- Kostenlose Lizenz für Open-Source Projekte auf Anfrage

Konfiguration (project.properties)

```
maven.junit.fork = true
```

Testüberdeckung

JCoverage Plugin

- Funktionalität äquivalent zu Clover
- GPL Version verfügbar
- Unbrauchbare Standardkonfiguration
- Fehler im aktuellen Maven 1.0-rc1 Release

Konfiguration (project.properties)

```
maven.jcoverage.report.template = jcoverage
```

GCover Plugin

- Formatiert die Ausgabe von gcov als HTML Frameset
- gcov kann Testüberdeckung für gcc Kompilate auswerten
- Bisher nicht in Maven integriert

Commit Log Plugins

ChangeLog Plugin

- Zeigt den SCM Commit Log als Bericht an
- Fügt Links auf die jeweiligen Revisionen in ViewCVS ein
- Unterstützt mehrere Soure Control Management Werkzeuge
  - CVS
  - Perforce
  - StarTeam
  - IBM Rational ClearCase
  - Subversion

CVS Plugins

Developer Activity Plugin

- Zeigt zu jedem Entwickler
  - Anzahl Commits
  - Anzahl der geänderten Dateien
- Unterstützt nur CVS

File Activity Plugin

- Zeigt zu jeder Datei die Anzahl der Änderungen
- Unterstützt nur CVS

CVS Plugins

StatCvs Plugin

- Erzeugt Metriken aus dem CVS Log
  - Anzahl der Lines of Code pro Entwickler
  - Commit Log (ähnlich dem ChangeLog Plugin)
  - Anzahl der Dateien
  - Grösse der Dateien
  - Grösse der Module
  - Aktivität in den Modulen
- Graphische Darstellung mit Charts
- Basiert auf StatCvs-XML
- Unterstützt nur CVS

Standard Plugins

License Plugin

- Zeigt die Lizenz an

Konfiguration (project.properties)

```
maven.license.licenseFile = COPYING
```

Changes Plugin

- Zeigt eine Versions Historie an
- Konfiguration über eine XML Datei

Beispiel

```
xsltproc changes.xsl xdocs/changes.xml | fold -s
  | perl -n -i -e 'print "    " if $_ =~ /^[A-Za-z]/
                   && $_ !~ /^Version/;' -e 'print $_' > CHANGES
```

Noch mehr Plugins

Ant Plugin [ant]

- Erzeugt eine build.xml Datei für Ant
- Hilfreich für Builds, wenn Maven nicht verfügbar ist

Eclipse Plugin [eclipse]

- Generierung von Eclipse Projekt Dateien
- Berücksichtigung aller Abhängigkeiten
- Anlegen der MAVEN_HOME Variable in Eclipse notwendig

SourceForge Plugin [sourceforge:deploy-dist]

- Uploaded und Released (!) Dateien auf SourceForge

Noch mehr Plugins

Console Plugin [console]

- Schnelles sukzessives Aufrufen von Goals über eine Konsole
- Unzureichende Speicherfreigabe führt nach mehrmaligem Goal Aufruf zu OutOfMemory Fehlern

Plugin Plugin [plugin:plugin] [plugin:install]

- Deployment von Maven Plugins
- Maven ist sein eigenes Build Tool

UberJar Plugin [uberjar]

- Erzeugt ein Jar mit allen Dependencies

Teil IV

Beyond the Basics

Command Line Optionen

Aufruf

```
maven [options] [goal [goal2 [goal3] ...]]
```

Eine Übersicht der wichtigsten Parameter:

- **-g** Eine Liste aller Goals anzeigen.
- **-o** Offline Modus. Nur Goals ausführen, die keine Internet Verbindung benötigen.
- **-p projekt-datei** Notwendig für Aufrufe aus Nicht-Projekt-Verzeichnissen (z.B. in einem Cron Job).

Konfiguration

Persönliche Einstellungen können in $HOME/build.properties vorgenommen werden.

$HOME/build.properties

```
maven.repo.local = /usr/local/maven-1.0-
rc1/repository
maven.home.local = /usr/local/maven-1.0-rc1
java.compiler = modern
maven.username = nick
```

Jelly Stylesheet Library (JSL)

- In JSL sind Code und Layout bunt durcheinander gemischt
- xnap.jsl: Ein Beispiel für ein 3 Column Layout
- Ausführen von PHP Skripten aus XDoc Seiten mit SSI

faq.xml

```
<section name="FAQ">
<p><include file="faq.php"/></p>
```

xnap.jsl

```
<jsl:template match="include" trim="false">
 <x:set var="_file" select="string(@file)"/>
 <jsl:comment>#include virtual="$_file"</jsl:comment>
</jsl:template>
```

Teil V

Fazit

Zusammenfassung

Maven ist ein **brauchbares** Tool

- Sehr einfache Erstellung von Projektdokumentation
- Sehr einfache Erhebung von Projekt- und Prozess-Metriken
- Sehr einfaches Building und Releasing, Deploying
- Automatische Auflösung von Abhängigkeiten

Manches ist noch **nicht ausgereift**

- Lange Startzeiten (auf marvin ca. 4s)
- Viele Reports sind Java spezifisch
- Anpassung der Webseiten schwierig
- Keine I18n Unterstützung
- Generierte Webseiten enthalten viele Leerzeilen
- Aufruf aus "falschen" Verzeichnissen führt zu *sehr* wirren Meldungen

Resourcen

Vortrag

- http://xnap.org/talks/
- http://statcvs-xml.berlios.de/
- http://poa.berlios.de/gcover-project/

Maven

- http://maven.apache.org/
- http://maven.apache.org/reference/plugins/index.html
- http://jakarta.apache.org/commons/jelly/
- http://maven-plugins.sourceforge.net/
- http://wiki.codehaus.org/maven/

Vielen Dank!