ANT

Kurzvortrag von Manuel Schulze

mschulze@inf.fu-berlin.de

ANT – Überblick

- Teilprojekt der **Apache Software Foundation** [1]
- ANT ist Opensource
- Build-Tool ähnlich wie **make** (?) jedoch voll auf Java zugeschnitten
- Warum dann ein neues Build-Tool?
  - Plattform unabhängig
  - Im Vergleich zu make sind Build-Files einfacher zu handhaben
  - Kein eigenes Dateiformat sondern XML basiert

ANT – Funktionen (Ausschnitt)

- ANT steht den Makefiles trotz Plattformunabhängigkeit in nichts nach

- Beliebig durch Java Code erweiterbar

- ANT kann:
  - Java Code compilieren
  - Dokumentation aus dem Quellcode erstellen (Javadoc und andere)
  - Jars und komplette Releases erzeugen
  - Archive erzeugen/entpacken (zip, gzip)
  - Die gängigsten CVS (?) Operationen ausführen
  - Dateioperation (Kopieren, Verschieben, Zugriffsrechte ändern) durchführen
  - SQL Operationen an Datenbanken abschicken
  - Sounds nach Beendigung des Build-Vorgangs abspielen :)
  - ...

- Für einen kompletten Überblick ANT Dokumentation lesen [2]

ANT – Installation

- Gute Nachricht: ANT ist in Eclipse [3] mit installiertem SDK bereits enthalten

- Consolen Nutzer müssen etwas mehr tun:
  - Vorraussetzung:
    - Installiertes JDK1.2 (Java 2)
    - Umgebungsvariable **JAVA_HOME** auf Java Installation gesetzt
    - **JAVA_HOME/bin** befindet sich im Pfad
  - Aktuelle Version (1.5.4) von ANT herunterladen [4]
  - Datei in ein beliebiges Verzeichnis entpacken
  - Umgebungsvariable **ANT_HOME** auf dieses Verzeichnis setzen
  - Verzeichnis **ANT_HOME/bin** in den Pfad aufnehmen

- Bemerkung: Einige ältere ANT Befehle arbeiten mit dem JDK1.1 besser als mit dem JDK1.2

ANT – Konfiguration (1)

- ANT selbst muss nicht konfiguriert werden
- Aufgabe des Benutzers ist es ein ANT Build-File für sein Projekt zu erstellen
- Wenn Dateiname **build.xml**, dann findet ANT diese Datei automatisch sonst Aufruf:

```
ant -buildfile <buildfile>
```

- Ein Build-File besteht aus einer Menge von **Properties** und **Targets**
- Ein Target stellt eine Aktion, die man auf das Projekt ausführen können soll, dar
- Targets können von anderen Targets abhängig sein

ANT – Konfiguration (2)

- Build-File wird eingeleitet mit:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

- Die Targets und Properties befinden sich alle zwischen:

```
<project name="Projektname" default="Default Target"
basedir="Projekt Basis Verzeichnis">
...
</project>
```

- Die Attribute sind:
  - **name** – gibt dem Projekt einen Namen
  - **default** – das Target, das ausgeführt wird, wenn kein anderes angegeben wird
  - **basedir** – das Wurzelverzeichnis des Projekts (normalerweise .)

ANT – Konfiguration (3)

- Mit Properties kann man häufig benutzte Strings definieren (z.B. Verzeichnisse, Namen)

- Definition wie folgt:

```
<property name=”Ein Name” value=”Ein Wert”/>
```

- z.B.

```
<property name=“group“ value=“04“/>
```

  Assoziiert die Gruppennummer „04“ mit dem Namen „group“

- Zugriff auf Properties wie folgt:

```
${Property Name}
```

- z.B.

```
${group}
```

  Führt dazu, das an der entsprechenden Stelle „04“ eingefügt wird

ANT – Konfiguration (4)

- Die Targets beschreiben die eigentlichen Aktionen auf dem Projekt

- Alles was zwischen:

```
<target name="Target Name">
...
</target>
```

steht wird von ANT interpretiert und ausgeführt (die erwähnten **Tasks**)

- Ein Beispiel:

```
<target name="echo">
   <echo>
   Hallo Welt
   </echo>
</target>
```

Würde „Hallo Welt“ auf den Bildschirm schreiben

- Targets werden auf der Kommandozeile wie folgt aufgerufen:

```
ant [-buildfile <Build File>] <Target Name>
```

- Lässt man `<Target Name>` weg, wird das **Default Target** ausgeführt

ANT – Konfiguration (5)

- Abhängigkeiten zu anderen Targets werden mit dem Attribut:
  `depends="Irgendein anderes Target"`
  bekannt gemacht

- Beispiel:

```
<target name="hallo">
   <echo>
   Hallo
   </echo>
</target>

<target name="echo" depends="hallo">
   <echo>
   Welt
   </echo>
</target>
```

- Zuerst werden alle Abhängigkeiten aufgelöst und anschliessend das eigentliche Target ausgeführt

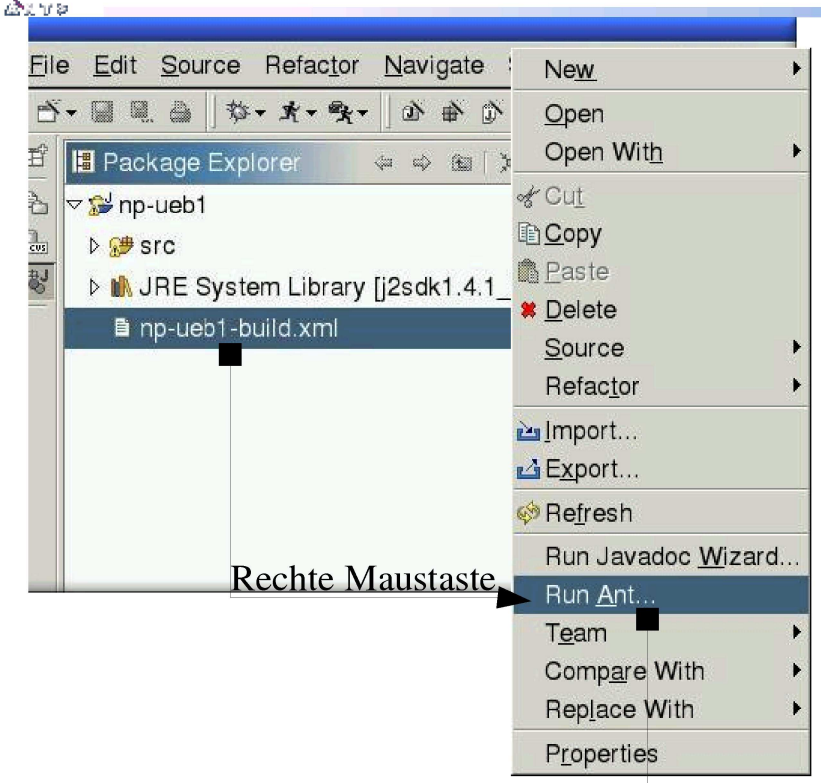
- Also erst „Hallo“ dann „Welt“

ANT – Ein grösseres Beispiel

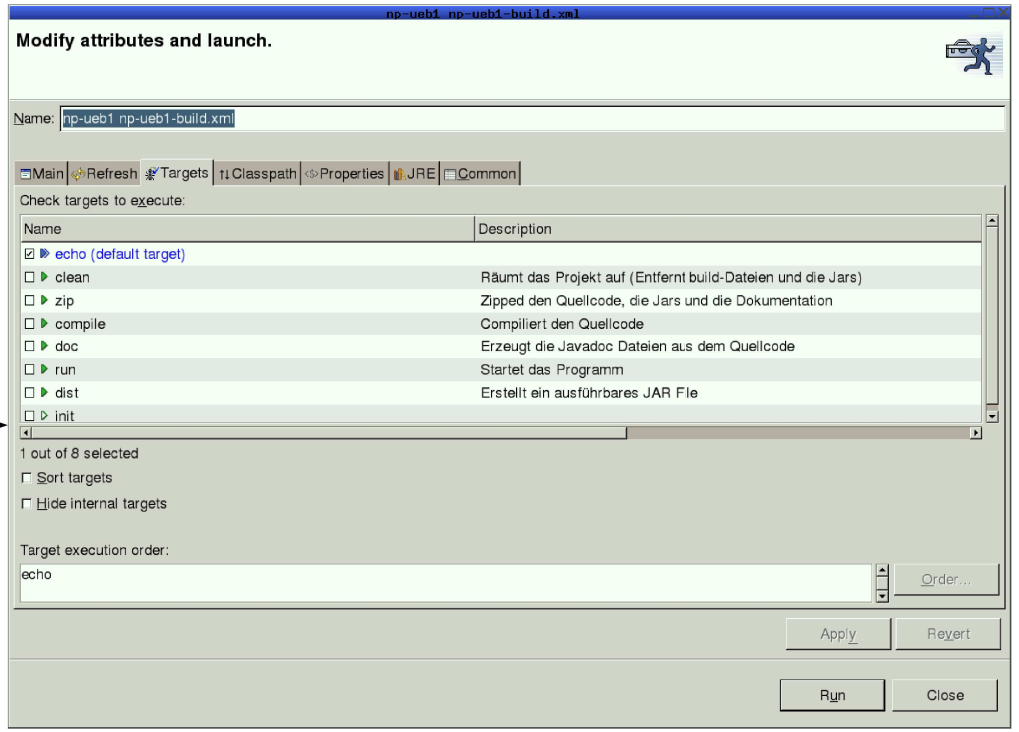
```
<property name="src" value="src"/>
<property name="build" value="build"/>
<property name="dist" value="dist"/>

<target name="compile" depends="init">
   <javac srcdir="${src}" destdir="${build}" debug="on">
      <classpath refid="project.class.path"/>
   </javac>
</target>

<target name="dist" depends="compile">
   <mkdir dir="${dist}"/>
   <jar jarfile="${dist}/${project}-${group}.jar"
basedir="${build}">
      <manifest>
         <attribute name="Main-Class" value="${exec}"/>
      </manifest>
   </jar>
</target>
```

- Der Aufruf von `ant dist` würde dazu führen das:
  - der Quellcode aus **${src}** nach **${build}** kompiliert wird (<javac/> Task)
  - aus den Kompilaten ein Jar-Archiv mit der zugehörigen Manifest Datei nach **${dist}** erzeugt wird (<jar/> Task)

ANT – Verwendung in Eclipse



- Build-File ist Teil des Projekts
- Klick auf **Run Ant...** öffnet das unten dargestellte Fenster
- Dort ANT Targets wählen und mit **RUN** ausführen lassen



Quellen und Links

[1]www.apache.org - Apache Software Foundation

[2]http://ant.apache.org/manual/index.html - Das ANT Manual

[3]http://www.eclipse.org - Homepage der Eclipse IDE

[4]http://ant.apache.org - ANT Homepage