

Apache Ant - Another Neat Tool

Hausarbeit in Open Source Software

Andy Stolzer
mail@stolzer.net

Matrikelnummer 701007

Augsburg, 2. Juli 2005

Inhaltsverzeichnis

1	Einführung	3
2	Installation	5
2.1	Dateien installieren	5
2.2	Umgebungsvariablen setzen	5
2.3	Ant kompilieren	6
3	Das Buildfile	7
3.1	Projekte (project)	9
3.2	Ziele (target)	9
3.3	Tasks	10
3.4	Properties	10
	Literatur	11

1 Einführung



Ant ist Open Source, startete als Teil des Jakarta-Projekts und ist nun ein Apache-Top-Level-Projekt. Ant ist ein Akronym und steht für “Another Neat Tool” (engl. für “Noch ein hübsches Werkzeug”) und entwickelt wurde die erste Version von James Duncan Davidson der 1999 ein Tool wie make für Java benötigte, als er die erste J2EE-Referenz-Implementierung entwickelte. Davidson gilt weiterhin als Vater von Jakarta Tomcat. Der Name “Ant” steht auch dafür, dass es, vergleichbar mit Ameisen, als relativ kleines Programm, Großes leisten kann.

Während bei Make die Steuerung durch Makefiles in einem besonderen (manchmal merkwürdigen) Format erfolgt, geschieht dies bei Ant durch den Buildfile in einem speziellen XML-Dateiformat. Der Defaultname für den Buildfile ist build.xml. Man kann aber ebenso wie bei Make mittels der Option -f eine Datei mit besonderem Namen oder Pfadangabe aufrufen.

Voraussetzung zur Installation von Ant ist eine vorhergehende Installation der Java-Entwicklungsumgebung (bisher kann das jede beliebige Version sein, ab dem nächsten Release werden nur noch die Versionen ab jdk1.2 unterstützt). Die Installation kann grundsätzlich in jedem beliebigen Verzeichnis erfolgen. Voraussetzung für die Verwendung ist einerseits, dass sich das Startskript ant (oder ant.bat) im Systempfad (PATH) findet und andererseits, dass die Umgebungsvariable JAVA_HOME auf das Installationsverzeichnis der Java-Entwicklungsumgebung

1 Einführung

zeigt.

Ant hat einen ähnlichen Aufrufmechanismus wie Make. Man kann optional den Namen der steuernden Datei angeben oder sich einfach auf den voreingestellten Namen (Makefile oder build.xml) beziehen. Ant kennt wie Make das Konzept von Targets, d.h. die explizite Angabe, welche Aktionen auszuführen wird. Ebenso gibt es für jeden Buildfile ein Defaulttarget, das intern festgelegt ist, und dessen Angabe unterbleiben kann. Genauso wie bei Make können bei Bedarf interne Variable in der Aufrufzeile definiert werden.

2 Installation

Eine aktuelle Version von ANT kann unter <http://ant.apache.org> bezogen werden.

Für die meisten Situationen sollte eine bereits compilierte Version genügen.

2.1 Dateien installieren

```
1 ant
2   +--- bin // Startskripte
3   |
4   +--- lib // Ant Jar Dateien und notwendige Abhängigkeiten
5   |
6   +--- docs // Dokumentation
7   |     +--- ant2 // Kurze Beschreibung der Anforderungen
8   |     |
9   |     +--- images // Bilder der HTML Dokumentation
10  |     |
11  |     +--- manual // Dokumentation
12  |
13  +--- etc
```

Listing 2.1: Struktur der Binaries

Die Verzeichnisse **bin** und **lib** werden für die Installation von Ant benötigt.

2.2 Umgebungsvariablen setzen

```
1 set ANT_HOME=c:\ant
2 set JAVA_HOME=c:\jdk1.5.0
3 set PATH=%PATH%;\%ANT_HOME%\bin
```

Listing 2.2: Windows

2 Installation

```
1 export ANT_HOME=/usr/local/ant
2 export JAVA_HOME=/usr/local/jdk-1.5.0
3 export PATH=${PATH}:${ANT_HOME}/bin
```

Listing 2.3: Unix (bash)

```
1 setenv ANT_HOME /usr/local/ant
2 setenv JAVA_HOME /usr/local/jdk-1.5.0
3 set path=( $path $ANT_HOME/bin )
```

Listing 2.4: Unix (csh)

2.3 Ant kompilieren

```
4 build install
```

Listing 2.5: Windows

```
1 build.sh install
```

Listing 2.6: Unix

3 Das Buildfile

Ant's Buildfile wird in der XML geschrieben. Jedes Buildfile beinhaltet ein **project** und mindestens ein **default** Target. Jedes Target beinhaltet einen Task, welcher mit einem id-Attribut belegt werden kann. Dieses kann dann referenziert werden.

```
1 <?xml version="1.0"?>
2 <project name="AlgoPlan" default="install" basedir=". ">
3
4 <!-- Folders -->
5 <property name="src" value="{basedir}/src"/>
6 <property name="src-bigclient" value="{src}/bigclient"/>
7 <property name="src-guibigclient" value="{src}/guibigclient"/>
8 <property name="src-logic" value="{src}/logic"/>
9 <property name="src-resources" value="{src}/resources"/>
10 <property name="src-server" value="{src}/server"/>
11 <property name="src-tools" value="{src}/tools"/>
12 <property name="dest" value="{basedir}/build/">
13
14 <!-- Delete existing server class files -->
15 <target name="clean">
16   <delete>
17     <fileset dir="{dest}" includes="*.*"/>
18   </delete>
19 </target>
20
21 <!-- Create build folder -->
22 <target name="prepare">
23   <mkdir dir="build"/>
24   <mkdir dir="build\run"/>
25 </target>
26
27 <!-- Build logic, resources and tools -->
```

3 Das Buildfile

```
28 <target name="build" depends="prepare">
29   <javac srcdir="${src-resources}" destdir="${dest}" deprecation="on
   "
30     debug="on" optimize="on" classpath="${dest}"/>
31   <javac srcdir="${src-logic}" destdir="${dest}" deprecation="on"
32     debug="on" optimize="on" classpath="${dest}"/>
33   <javac srcdir="${src-tools}" destdir="${dest}" deprecation="on"
34     debug="on" optimize="on" classpath="${dest}"/>
35 </target>
36
37 <!-- Build the client application -->
38 <target name="build-bigclient" depends="prepare">
39   <javac srcdir="${src-guibigclient}" destdir="${dest}" deprecation=
   "on"
40     debug="on" optimize="on" classpath="${dest}"/>
41   <javac srcdir="${src-bigclient}" destdir="${dest}" deprecation="on
   "
42     debug="on" optimize="on" classpath="${dest}"/>
43 </target>
44
45 <!-- Build the server application -->
46 <target name="build-server" depends="prepare">
47   <javac srcdir="${src-server}" destdir="${dest}" deprecation="on"
48     debug="on" optimize="on" classpath="${dest}"/>
49   <rmic base="${dest}" stubversion="1.2" classname="server.
   ServerImpl"/>
50 </target>
51
52 <target name="cp">
53   <copy todir="${dest}">
54     <fileset dir="${src}" includes="*.policy,*.properties,*.xml,*.
   gif,*.bat"/>
55   </copy>
56   <copy todir="${dest}\run">
57     <fileset dir="${src}\run" includes="*.bat"/>
58   </copy>
59 </target>
60
61 <target name="run">
62   <exec dir="" executable="C:\Programme\Internet Explorer\iexplore.
```

```

    exe">
63     <arg line="G:\MyDiplomarbeit\Anwendung\AlgoPlan\build\run"/>
64 </exec>
65 </target>
66
67 <target name="show">
68     <exec dir="" executable="explorer.exe">
69         <arg line="/e,/root,G:\MyDiplomarbeit\Anwendung\AlgoPlan"/>
70     </exec>
71 </target>
72
73 <target name="install" depends="prepare,build,build-server,build-
    bigclient,cp"/>
74
75 <target name="start" depends="install,run"/>
76 </project>

```

Listing 3.1: Beispiel build.xml

3.1 Projekte (project)

Das Tag `project` hat 3 Attribute:

- **name** Der Name des Projektes
- **default** Das Standard-Ziel
- **basedir** Das Basisverzeichnis aus dem alle Funktionen aufgerufen werden

Optional kann eine Beschreibung im Tag **description** hinzugefügt werden.

3.2 Ziele (target)

Mit dem Attribut **depends** kann ein Ziel von von anderen Zielen abhängig gemacht werden. Im Listing “Beispiel Target” hängt das Target `build-server` vom Target `prepare` ab, welches als erstes ausgeführt wird. Die Abhängigkeit eines Zielles ist nicht auf einen Wert beschränkt. So könnte man zum Beispiel folgenden Ausdruck verwenden `depends="a,b,c,d"`.

```
1 <target name="build-server" depends="prepare"/>
```

Die Ausführung eines Targets kann bedingt werden. Hierzu werden die Attribute **if** und **unless** eingesetzt. Folgende Attribute sind für Ziele zulässig:

- name
- depends
- if
- unless
- description

3.3 Tasks

Ein Task ist ein Stück Code der ausgeführt werden kann. Im Beispiel build.xml wurde zum Beispiel der Java Compiler eingebunden.

```
1 <javac srcdir="${src-logic}" destdir="${dest}" deprecation="on"  
2     debug="on" optimize="on" classpath="${dest}"/>
```

3.4 Properties

Properties können in die build.xml geschrieben werden oder in eine externe Datei ausgelagert werden. Ein Property wird im Tag property definiert und hat die Attribute name und value, z.B.

```
1 <property name="src-tools" value="${src}/tools"/>
```

Der Zugriff auf eine Property erfolgt über

```
1 ${src-tools}
```

Ausgelagerte Property-Dateien können mit

```
1 <property file="property.properties">
```

eingebunden werden.

Literaturverzeichnis

[1] Apache. *Apache Ant 1.6.5 Manual*.