Introduktion til ant

Denne artikel beskriver Apache ant, som er et værktøj til at builde Java applikationer med.

Den beskriver nogle af de mest brugte ant tasks.

Den forudsætter solidt kendskab til Java og lidt erfaring med build af større systemer.

Skrevet den **09. Feb 2009** af **arne_v** I kategorien **Programmering / Java** | ★★☆☆☆

Historie:
V1.0 - 13/01/2004 - original
V1.1 - 31/01/2004 - forbedret formatering
V1.2 - 07/04/2004 - udpensle synopsis lidt
V1.3 - 25/07/2004 - uddybe og forklare bedre udfra feedback

Hvad er ant

Apache ant er Java verdenens svar på make.

Hvis du kender make, så kender du princippet.

For dem som ikke kender make, så er det en løsning på problemet, hvor man skal have bygget et lidt større projekt og det kræver 10, 100 eller måske endda 1000 kommandoer. Det er jo håbløst at taste manuelt. Og laver man et script som udfører alle kommandoer hver gang, så kan det tage mange timer at køre. Make er intelligent d.v.s. at den indeholder instruktioner til at kunne udføre alle kommandoer, men den nøjes med at udføre dem som er nødvendige (den checker det ved at sammenligne tids stempler på filer).

Java bruger ant til det samme. Man definerer hvordan projektet skal builds i en XML fil. Man kører ant. Ant finder selv ud af hvilke dele af projektet der skal rebuildes og gør det. ant understøtter alle de ting man har brug for i almindelig Java og i J2EE. Det er efterhånden almindeligt at større program pakker i Java verdenen kommer med ant tasks så pakken kan bruges i ant scripts.

Hvorfor ant

Alle udviklings miløer (JBuilder, Eclipse etc.) kommer med funktionalitet til at builde etc.. Hvorfor ikke bruge det i.s.f. at lære et nyt værktøj som ant ?

Det er der flere grunde til:

1) den slags udviklings miljøer understøtter normalt kun de mest basale compile, jar og kør funktionalitet hvorimod ant understøtter en masse ekstra

2) med ant er man uafhængig af udviklings miljøet

3) ant XML filen er fremragende dokumentation af hvordan builden foregår
    hvorimod udviklings miljøets build typisk er skjult en i binær projekt
    fil i proprietært format

ant er efterånden blevet standard i de fleste større Java udviklings projekter.

Og derfor bør man også kende ant hvis man vil arbejde som java programmør.

(mange udviklings miljøer er også begyndt at understøtte ant)

Brug
Apache ant kan hentes her:
  http://ant.apache.org/

Installationen består i at:
  - unzippe
  - definere en environment variabel ANT_HOME til at pege på roden
  - få bin directoriet i PATH

Makefile i ant hedder build.xml og er som navnet antyder
i XML.

Jeg vil i det efterfølgende kort beskrive nogle af de mest
gængse tasks i ant.

Jeg vil kun berøre en ganske lille del af de næsten uendelige
muligheder der er i ant.

Men når man først er kommet igang med ant har man sjældent
problemer med at finde det man skal bruge i ant's dokumentation.

Simpelt eksempel
Lad os starte med et meget simpelt eksempel:
  - vi har nogle Java klasser i src\minpakke*.java
  - vi vil gerne have buildet en jar fil af den oversatte kode
  - vi vil helst ikke have .class filerne ned i src men have dem i
    et separat directory

Problemet kan selvfølgelig godt løses med et gammeldags build script
(build.bat eller build.sh), men det kan godt være lidt kringlet
at få den lavet rigtigt med komplicerede directory strukturer
og brug af pakker. Og build script er naturligvis også platform
specifikke, hvilket ikke er i javas ånd.

En ant build.xml til problemet kan se ud som:

```
<project name="simple" default="build">
    <property name="src.dir" value="src"/>
    <property name="bin.dir" value="bin"/>
    <property name="jar.fil" value="minutil.jar"/>
    <target name="compile">
        <javac srcdir="${src.dir}" destdir="${bin.dir}"/>
```

```
    </target>
    <target name="build" depends="compile>
        <jar destfile="${jar.fil}" basedir="${bin.dir}"/>
    </target>
</project>
```

Denne XML fil består af:
- 3 properties som definerer symbolske navne når man definerer en
  property "src.dir" så kan man senere i XML filen referere til
  ${src.dir} og påd en måde bliver filen nemmere at vedligeholde
- et target ved navn compile som kalder javac
- et target ved navn build som afhænger af compile og kalder jar

ant compile

vil:
  - compile alt fra src\minpakke*.java til bin\minpakke*.class

ant build

eller bare (fordi build er default):

ant

vil:
  - compile alt fra src\minpakke*.java til bin\minpakke*.class
  - pakke bin\minpakke*.class i minutil.jar

Det er faktisk ret simpelt.

Mere komplekst eksempel
Ovenstående var et J2SE eksempel.

Man har endnu mere nytte af ant til J2EE.

Jeg vil vise og kommentere en build.xml jeg faktisk har brugt.

Hvis du ikke kender alle J2EE begreberne så fortvivl ikke - man kan godt få et indblik i styrken i ant uden at kende betydningen af det hele.

Hvis du skal bruge ant til J2EE kan du sakse en del fra dette eksempel.

```
<project name="useradmin" default="allbuild">
    <!--
      Definer jar/war/ear/zip filer.
    -->
    <property name="ejbjarname" value="useradmin-ejb.jar"/>
    <property name="warname" value="useradmin.war"/>
    <property name="earname" value="useradmin.ear"/>
    <property name="ejbzipname" value="ejb-doc.zip"/>
```

```
    <property name="webzipname" value="web-doc.zip"/>
    <!--
      Definer directories.
    -->
    <property name="ejbsrcdir" value="ejbsrc"/>
    <property name="ejbbuilddir" value="ejbbuild"/>
    <property name="ejbdescripdir" value="ejbdescrip"/>
    <property name="ejbdocdir" value="ejbdoc"/>
    <property name="websrcdir" value="websrc"/>
    <property name="webbuilddir" value="webbuild"/>
    <property name="webdescripdir" value="webdescrip"/>
    <property name="weblibdir" value="weblib"/>
    <property name="webdocdir" value="webdoc"/>
    <property name="appdescripdir" value="appdescrip"/>
    <property name="testsrcdir" value="testsrc"/>
    <property name="testbuilddir" value="testbuild"/>
    <property name="jspdir" value="jsp"/>
    <property name="xmldir" value="xml"/>
    <property name="propsdir" value="props"/>
    <!--
      Definer en utility jar fil som vi skal bruge.
    -->
    <property name="utillib" value="../Util/util.jar"/>
    <!--
      Load en ekstern (standard Java) properties fil som deles af mange
build.xml
      med placering af diverse fælles jar filer: j2ee.jar, log4j.jar etc.
    -->
    <property file="../ant-global.properties"/>
    <!--
      Compile EJB's.
    -->
    <target name="ejbcompile">
        <javac classpath="${log4jlib};${j2eelib}" srcdir="${ejbsrcdir}"
destdir="${ejbbuilddir}"/>
    </target>
    <!--
      Lav EJB jar file.
    -->
    <target name="ejbbuild" depends="ejbcompile">
        <jar jarfile="${ejbjarname}">
            <fileset dir="${ejbbuilddir}"/>
            <metainf dir="${ejbdescripdir}"/>
        </jar>
    </target>
    <!--
      Compile servlets og beans.
    -->
    <target name="webcompile">
        <javac classpath="${ejbjarname};${log4jlib};${j2eelib}"
srcdir="${websrcdir}" destdir="${webbuilddir}"/>
    </target>
    <!--
      Kopier en utility jar fil som vi skal have med i web applikationens lib.
    -->
```

```
    <target name="weblibgen">
        <copy file="${utillib}" todir="${weblibdir}"/>
    </target>
    <!--
      Lav war fil.
    -->
    <target name="webbuild" depends="webcompile,weblibgen">
        <war warfile="${warname}" webxml="${webdescripdir}/web.xml">
            <classes dir="${webbuilddir}"/>
            <classes dir="${propsdir}"/>
            <lib dir="" includes="${ejbjarname}"/>
            <lib dir="${weblibdir}"/>
            <fileset dir="${jspdir}"/>
            <webinf dir="${xmldir}"/>
        </war>
    </target>
    <!--
      Lav ear fil.
    -->
    <target name="allbuild" depends="ejbbuild,webbuild">
        <ear earfile="${earname}" appxml="${appdescripdir}/application.xml">
            <fileset dir="" includes="${ejbjarname},${warname}"/>
        </ear>
    </target>
    <!--
      Deploy (bare en copy) til JBoss.
    -->
    <target name="deploy" depends="allbuild">
        <copy file="${earname}" todir="${deploydir}"/>
    </target>
    <!--
      Compile JUnit test cases.
    -->
    <target name="testcompile" depends="ejbbuild">
        <javac classpath="${ejbjarname};${junitlib};${j2eelib}"
srcdir="${testsrcdir}" destdir="${testbuilddir}"/>
    </target>
    <!--
      Run JUnit test cases.
      Vigtigt: brug altid fork="on" ellers virker det ikke
    -->
    <target name="testrun" depends="testcompile">
        <junit fork="on">
            <classpath
path="${testbuilddir};${ejbjarname};${log4jlib};${jbosslib};${j2eelib}"/>
            <formatter type="plain" usefile="false"/>
            <test name="test.AllTests"/>
        </junit>
    </target>
    <!--
      Generer JavaDoc for EJB's.
    -->
    <target name="ejbgendoc">
        <javadoc classpath="${log4jlib};${j2eelib}"
packagenames="dk.vajhoej.*" sourcepath="${ejbsrcdir}" destdir="${ejbdocdir}"/>
```

```
    </target>
    <!--
      Zip JavaDoc for EJB's.
    -->
    <target name="ejbzipdoc" depends="ejbgendoc">
        <zip zipfile="${ejbzipname}" basedir="${ejbdocdir}"/>
    </target>
    <!--
      Generer JavaDoc for servlets og beans.
    -->
    <target name="webgendoc">
        <javadoc classpath="${ejbjarname};${log4jlib};${j2eelib}"
packagenames="dk.vajhoej.*" sourcepath="${websrcdir}" destdir="${webdocdir}"/>
    </target>
    <!--
      Zip JavaDoc for servlets og beans.
    -->
    <target name="webzipdoc" depends="webgendoc">
        <zip zipfile="${webzipname}" basedir="${webdocdir}"/>
    </target>
    <target name="document" depends="ejbzipdoc,webzipdoc"/>
</project>
```

Videre med ant

Når først du kommer igang med ant lærer du hurtigt at slå ant tasks op
i dokumentationen (følger med når man installerer).

Jeg har allerede nævnt følgende tasks:

javac - compile
java - kør
javadoc - generer dokumentation
jar - jar filer
zip - zip filer
copy - kopier filer
junit - junit tests
war - war filer (J2EE)
ear - ear filer (J2EE)

Af andre vigtige tasks som komme rmed ant vil jeg nævne:

exec - kør eksterne programmer/Scripts
delete - slet filer
rmic - kør RMI compiler
touch - marker filer som ændrede

Af kende program pakker som kommer med specielle ant tasks vil jeg nævne:
- xdoclet
- Apache Axis
- de fleste JDO implementationer

Kommentar af simonvalter d. 01. Aug 2004 | 1

Jeg kan ikke helt forstå kritiken, hvert target har en fin forklaring og man kan hurtigt se hvilken task der skal bruges. for en forklaring af hver attribut kan man læse dokumentationen. Men ok jeg er nok skadet af at jeg bruger mere end en kilde til at lære om noget.

Kommentar af kube d. 15. Jan 2004 | 2

Jeg fik ikke det mindste ud af denne artikel.

Kommentar af touel d. 06. Mar 2005 | 3

Kommentar af mikkelbm d. 04. Dec 2004 | 4

Jeg forstår heller ikke kritikken af artiklen.
Fra synopsis:
"Den forudsætter solidt kendskab til Java og lidt erfaring med build af større systemer"
Måske derfor de andre ikke forstår det der bliver beskrevet.
Skal dog lige siges at jeg ikke har set versionerne før 1.3

Kommentar af william_munny d. 27. Mar 2004 | 5

Kunne godt bruge noget mere i opsætningen af ant, men ellers fin artikel.

Kommentar af ttn- d. 30. Jul 2004 | 6

For lidt forklaring af, hvad der sker. Virker ellers som et glimrende værktøj.

Kommentar af meho_tarevci d. 14. Sep 2005 | 7

Kommentar af alister_crowley d. 31. Mar 2005 | 8