Apache ANT

Introducere

Apache Ant este o unealta open-source de dezvoltare a software-ului scrisa in intregime in Java. Este o unealta similara utilitarului **make** utilizat la dezvoltarea aplicatiilor scrise in limbajul C.

Spre deosebire de make care invata sa execute noi actiuni prin intermediul comenzilor shell-ului (command.com, bash), Ant se poate extinde scriind noi clase Java. Fisierele de configurare se bazeaza pe XML. Fiecare operatie este executata de un obiect ce implementeaza o interfata Task specifica.

Pentru a executa comenzi shell, independent de sistemul de operare instalat pe calculatorul unde ruleaza programul, Ant pune la dispozitie operatia **<exec>**.

Distributia Ant este utilizata cu success pe mai multe platforme, de la Linux, Solaris, Windows pana la Novell Netware6 sau MacOS X. Versiunea compilata contine ultima versiune a parserului XML Apache Xerces2.

Versiunea curenta necesita prezenta unei variante de JDK (>1.1) instalata pe calculatorul gazda. Unele operatii nu functioneaza decat cu Java 1.2 sau o versiune ulterioara a acesteia.

Observatie. JDK de la Microsoft nu este compatibil in intregime cu Ant.

Instalare

Pentru a putea lucra acest program trebuie sa realizam instalarea utilitarului precum si realizarea configurarii mediului sistemului de operare.

Pasi ce trebuie urmati in acest sens sunt urmatorii:

1) Fisierul ce contine distributia programului Ant poate fi descarcat de la adresa http://ant.apache.org. Dupa ce a fost descarcat, se alege un director unde va fi decompresata arhiva (fie o arhiva zip, fie o arhiva tar.gz).

Observatie. Daca a fost luata arhiva cu fisierele sursa ale proiectului, pentru a putea rula programul, mai intai trebuie sa compilam intregul proiect!

1. Dowloadati distributia programului ANT de la adresa http://ant.apache.org.

   Observatie: Daca a fost luata arhiva cu fisierele sursa ale proiectului, pentru a putea rula programul, mai intai trebuie sa compilam intregul proiect!

2. Dezarhivati programul intr-un director ales de catre dumneavoastra (de preferinta un director nou).

   Exemplu: d**:\ant**

3. Initializati variabila de mediu ANT_HOME cu adresa absoluta catre directorul de instalare.

   Exemplu: **set ANT_HOME=d:\ant**

4. Verificati faptul ca variabila de mediu JAVA_HOME este initializata cu adresa absoluta catre o distributie Java prezenta pe calculator.

Exemplu: **set JAVA_HOME=c:\jdk1.4.2**

5. Includeti in variabila de mediu PATH si calea catre ANT_HOME\bin
Exemplu: **set PATH=%PATH%:%ANT_HOME%\bin**

Observatie: Pentru ca setarile sa ramana permanente se recomanda adaugarea acestor comenzi in fisierul `autoexec.bat` pe sistemele de operare Windows 9x sau setarea acestor variabile ca *Environment Variables* la nivelul sistemului Windows 2000, XP, 2003.

Observatie. Pentru a verifica daca programul Ant a fost instalat/configurat corect este suficient ca din linia de comanda sa scrieti

ant [ENTER].

Elemente de Baza

Pentru a folosi Ant in scopul compilarii automatizate a distributiei unui produs este nevoie de un fisier de configurare *build.xml*, fisier ce trebuie sa respecte in primul rand sintaxa limbajului XML si apoi sa respecte modul de formare pentru a putea fi inteles de catre Ant.

Sintaxa unui fisier de build

Fisierele Ant de build contin un proiect si cel putin un target (default). Target-urile contin elemente de tip task. Fiecare element de tip task dintr-un fisier de build poate avea un atribut `id` prin intemediul caruia poate fi referit ulterior. Valoarea acestui identificator trebuie sa fie unica.

Proiecte

Un *project* are urmatoarele trei atribute:

Atribut	Descriere	Obligatoriu
name	numele proiectului.	Nu
default	numele target-ului default utilizat atunci cand la rulare nu e furnizat nici un target.	Da
basedir	Directorul de baza relativ la care sunt calculate toate caile. Acest atribut poate fi suprascris setand proprietatea "basedir" mai inainte. Daca acesta a fost facuta atunci acest atribut poate lipsi. Daca nici atributul nici porprietatea nu a fost setata atunci se va considera directorul parinte al fisierului.	Nu

Optional, se poate furniza o descriere a proiectului poate in interiorul tagului `<description>`.

Fiecare proiect defineste unu sau mai multe *target-uri*. Un target este format dintr-o multime de *task-uri* care pot fi executate. La rularea programului Ant, putem selecta target-ul/target-urile care dorim sa le executam. Daca nu este specificat nici un target, atunci se utilizeaza targetul default al proiectului.

Target-uri

Sintaxa:

```
<target name="targetName" depends="lista targeturi" if=”nume propritate 1” unless=”nume propritate 2” description=”Descriere target”/>
```

Un *target* are urmatoarele atribute:

Atribut	Descriere	Obligatoriu
`Name`	numele target-ului	Da
`Depends`	lista de nume de targeturi de care depinde target-ul curent separate prin virgula.	Nu
`If`	numele proprietatii care trebuie sa fie setata pentru ca targetul sa se execute.	Nu
`unless`	numele proprietatii care nu trebuie sa fie setata pentru ca targetul sa se execute.	Nu
`description`	o scurta descriere a activitatii realizate de target.	Nu

Un nume de target valid este format din orice combinatie alfanumerica de caractere ce respecta encodingul fiserului XML. Cu toate ca este permis ca numele sa fie vid sirul "" , sa contina vigula "," sau spatiu " " este de dorit sa se evite folosirea acestora pentru a se evita confuzii.

Target-urile al caror nume incep cu minus (de exemplu `"-restart")` sunt valide, insa se utilizeaza atunci cand acele targeturi nu pot fi apelate direct din linia de comanda.

Un target poate sa depinda de alte target-uri. De exeplu putem avea un target pentru realizarea compilari si unul pentru crearea distributiei proiectului. Insa nu potem realiza distributia proiectului daca nu a fost compilat in prelabil. Astfel, spunem ca target-ul de distribuie *depinde de* targetul de compilare. Ant poate sa resolve astfel de dependente.

Astfel, pentru a specifica ca un target depinde de unu sau mai multe target-uri utilizam atributul `depends` pentru a specifica *ordinea* in care trebuie sa se execute target-urile ce trebuie sa se execute.

Ant incerca sa execute target-urile specificate in atributul `depends` in ordinea in care apar (de la stanga la dreapta). Trebuie sa tinem seama ca este posibil ca un target sa se execute mai devreme decat este specificat in lista daca un alt target depinde de el:

```
<target name="A"/>
<target name="B" depends="A"/>
<target name="C" depends="B"/>
<target name="D" depends="C,B,A"/>
```

Sa presupunem ca dorim sa executam targetul D. In conformitate cu valorile atributului `depends`, trebuie sa se execute mai intai targetul C, apoi B si apoi A. Gresit! C depinde B, si B depinde A, astfel A este executat primul, apoi B, apoi C, iar in final D.

Este posibil sa conditionam executia unui task de prezenta, `if`, (sau absenta `unless`) setarii unei variabile. Aceasta pemite, de exemplu, un mai bun control al procesului de build-uire in functie de starea sistemului (versiunea de java, versiunea de system de operare, definirea unor parametric din linia de comanda, etc.). Pentru a realiza acest lucru se utilizeaza atributul `if` (sau `unless`) cu numele proprietatior care trebuie luate in considerare.

Observatie: Ant va verifica numai daca acea proprietate este setata, si nu si ce valoare are. O prorietate care are ca valoare sirul vid "" este considarata definata .

Exemplu:

```
<target name="build-module-A" if="module-A-present"/>
<target name="build-own-fake-module-A" unless="module-A-
present"/>
```

In primul exemplu, daca proprietatea `module-A-present` a fost setata (indifferent cu ce valoare), targetul se va rula. In al doilea examplu, daca prorietatea `module-A-present` a fost setata (din nou, indifferent de valoare), targetul nu se va rula.

Task-uri

Un task este un fragment de cod (o actiune) care poate fi executat.

Un task poate avea mai multe atribute (sau arguments). Valoarea unui atribut poate face referire la o proprietietate definite in prealabil

Task-urile au urmatoarea stuctura:

```
<name attribute1="value1" attribute2="value2" ... />
```

unde *name* este numele taskului, *attributeN* este numele atributului N, iar *valueN* este valoarea pentru acest atribut.

Exista o serie de taskuri built-in (http://ant.apache.org/manual/coretasklist.html) (mkdir,copy, delete, javac) precum si o serie de taskuri optionale. De asemenea este posibil sa scriem noi propriile taskuri.

Propietati

Intr-un proiect se pot defini o multime de proprietati (variabile). Acestea se pot defini fie in interiorul fisierului de build utilizand tagul `property`, fie in afara Ant-ului. Sintaxa definirii unei variabile este urmatoarea:

```
<property name="numePropietate" value="valoare" />

<property name="numePropietate" location="cale" />
```

O propietate are un nume si o valoare; numele este case-sensitive. Valorile acestori propietati se pot utiliza ca attribute in cadrul taskurilor. Referirea la valoarea unei variabile se fac printr-o constructie de forma `${numeVariabila}`.

Propietati predefinite

In afara variabilelor definite de catre utilizator, exista o serie de variabile predefinite, majoritatea dintre ele avand nume identice cu cele utilizate in *System.getProperties()*.
Exemplu:

`basedir`	- calea absoluta catre directorul radacina al proiectului
`ant.file`	- calea absoluta a fisierului 'build.xml'
`ant.version`	- versiunea programului Ant
`ant.project.name`	- numele proiectului aflat in desfasurare
`ant.java.version`	- versiunea masinii virtuale Java – JVM - ce a fost detectata.

Exemplu:
Modul de utilizare al variabilelor predefinite este identic cu modul de utilizare al variabilelor definite de catre utilzator.
Exemplu:

```
<?xml version="1.0"?>
<project name="Test Default Variables" default="PrintVars" basedir=".">


<target name="PrintVars" description="Prints some system variables">
   <echo message = "Java version is ${ant.java.version}"/>
   <echo message = "Ant version is ${ant.version}"/>
</target>

</project>
```

Exemplu Buildfile

```
<project name="MyProject" default="dist" basedir=".">
    <description>
        simple example build file
    </description>
  <!-- set global properties for this build -->
  <property name="src" location="src"/>
  <property name="build" location="build"/>
  <property name="dist"  location="dist"/>

  <target name="init">
    <!-- Create the time stamp -->
    <tstamp/>
    <!-- Create the build directory structure used by compile -->
    <mkdir dir="${build}"/>
  </target>

  <target name="compile" depends="init"
        description="compile the source " >
    <!-- Compile the java code from ${src} into ${build} -->
    <javac srcdir="${src}" destdir="${build}"/>
  </target>
```

```
  <target name="dist" depends="compile"
        description="generate the distribution" >
    <!-- Create the distribution directory -->
    <mkdir dir="${dist}/lib"/>

    <!-- Put everything in ${build} into the MyProject-${DSTAMP}.jar file -
->
    <jar jarfile="${dist}/lib/MyProject-${DSTAMP}.jar" basedir="${build}"/>
  </target>

  <target name="clean"
        description="clean up" >
    <!-- Delete the ${build} and ${dist} directory trees -->
    <delete dir="${build}"/>
    <delete dir="${dist}"/>
  </target>
</project>
```