Structura unui proiect tipic

Un proiect, in viziunea Ant, trebuie sa contina trei subdirectoare de baza:

1. **src** – este directorul in care se vor afla toate sursele programului, inclusiv pachetele acestuia.
2. **classes** – este generat de obicei de catre Ant si contine clasele compilate ale proiectului, inclusive subdirectoare corespunzatoare eventualelor pachete (packages).
3. **doc/api** – este generat de obicei de catre Ant si va contine documentatia proiectului generata automat cu ajutorul programului JavaDoc.

De regula alaturi de aceste directore se mai gasesc:

4. **lib** – acest director contine librariile (jar-urile) necesare compilarii si rularii aplicatiei
5. **dist** – in acest director se va genera versiunea "pentru distributie" (binara) a aplicatiei.

Exemplu: Sa presupunem ca aplicatia noastra se afla in directorul "Biblioteca". Continutul minimal al acestuia este:

```
____Biblioteca      ___ classes
                   |___ doc _ api
                   |___ src
                   |___ lib
                   |___ dist
                   build.xml
```

```
<project name="Biblioteca" default="build" basedir=".">
```

name – numele proiectului

default – numele sarcinii (target) ce va fi executata in mod implicit daca ANT este apelat fara parametri

basedir – calea relativa catre directorul radacina al proiectului. "." Indica faptul ca orice task va fi executat in directorul radacina al proiectului.

La fel ca si utilitarul **make**, fisierul **build.xml** defineste mai multe sarcini ce pot fi incercate independent. Fiecare dintre aceste sarcini trebuie sa fie initializata cu un nume si o descriere.

Exemplu:

```
<target name="build" description="Compileaza toate fisierele sursa">
```

Observatie. Este indicat ca in interiorul fisierului de configurare sa se foloseasca comentarii XML de forma <!-- …. --> pentru a putea fi usor inteles. Aceste descrieri pot fi vizualizate in urma executarii comenzii:

ant –projecthelp

Dupa ce am definit o sarcina, trebuie specificata o lista de operatii (tasks) ce indica transformarile ce trebuie efectuate pentru indeplinirea cu succes a acesteia atunci cand este ceruta de catre **Ant**. Programul Ant contine un numar variat de operatii predefinite.

Observatie. In situatia in care o anumita operatie nu este definita in Ant, atunci i se ofera posibilitatea programatorului de a extinde una din clasele-operatie existente utilizand o multime de interfete.

Pentru a realiza o simpla compilare sunt necesare doua operatii: mai intai trebuie creat directorul “classes”. Acest lucru se realizeaza cu ajutorul operatiei predefinite ‘**mkdir**’:

```
<mkdir dir=”/classes”/>
```

Cel de-al doilea lucru ce trebuie efectuat este compilarea propriu-zisa. Ant contine o operatie predefinita pentru a compila fisierele sursa java. Din documentatie se poate observa multitudinea de optiuni disponibile pentru operatia ‘**javac**’.

Cel mai simplu mod de utilizare este de a specifica numai directoarele sursa si destinatie. In cadrul procesului de compilare se va intra recursiv in directorul sursa cat si in subdirectoarele acestuia.
Exemplu:

```
<javac srcdir=”/src” destdir=”/classes”/>
```

Recapituland, pana in acest moment fisierul nostru arata astfel:

```
<project name="Bilioteca" default="build" basedir=".">
   <target name="build" description="Compileaza toate fisierele sursa">
      <mkdir dir="/classes"/>
      <javac srcdir="/src" destdir="/classes"/>
   </target>
</project>
```

Exemplu:
Pentru a verifica ceea ce am realizat vom apela fie ‘ant’ fie ‘ant build’:

```
> ant build
Buildfile: build.xml

build:
      [mkdir] Create dir: C:\Biblioteca\classes

BUILD SUCCESSFUL
Total time: 3 seconds
```

Utilizarea variabilelor

De obicei atunci cand scriem un fisier de configurare mai complicat, suntem pusi in situatia sa refolosim nume de fisiere, nume de cai si valori in diferite locuri ale fisierului. Elementele comune pot fi definite la inceputul fisierului ‘**build.xml**’.
Exemplu:

```
<project name=”Bilioteca” default=”build” basedir=”.”>

   <property name="src" location="src"/>
   <property name="classes" location="classes"/>
   <property name="docs.api" location="doc/api"/>
```

Generarea “stampilelor de timp”

Exista situatii in care este necesar ca versiunea unor fisiere sa fie denumita pe baza datei curente sau a timpului current. Primul pas in directia generarii unei stampile de timp este sa initializam variabilele predefinite de timp si data.
Exemplu:

```
<tstamp/>
```

Aceasta comanda trebuie inclusa in interiorul unei operatii, deoarece in exteriorul contextului unei operatii nu se intampla nimic, si astfel o stampila de timp nu este relevanta. In urma executarii acestei comenzi, sunt initializate urmatoarele variabile (se specifica si formatul lor implicit):

DSTAMP – 'yyyyMMdd'
TSTAMP – 'hhmm'
TODAY – 'MMMM dd yyyy'
Formatul implicit poate fi modificat folosind elementul **<format>.**

Operatiuni uzuale

<javac>

Asa cum am aratat anterior, pentru a putea utiliza operatia '**javac**' configuratia minimala include initializarea directorului sursa cat si a directorului destinatie.

```
<target name="build" description="Compileaza toate fisierele sursa">
   <mkdir dir=""/>
   <javac srcdir="" destdir=""/>
</target>
```

Sunt situatii cand este necesar ca mai multe librarii jar sa fie adaugate la Classpath doar pentru momentul compilarii. Acest rezultat se poate obtine prin folosirea tagurilor '**classpath**' in combinatie cu '**fileset**'.

Exemplu: Sa presupunem ca vrem sa includem fisierul jar 'c:\parsers\xmlXerces.jar' precum si directorul 'c:\htmlparser\classes\'.

```
<target name="build" description="Compileaza toate fisierele sursa">
   <mkdir dir=""/>
   <javac srcdir="" destdir="">
      <classpath>
           <fileset file="c:\parsers\xmlXerces.jar"/>
           <fileset dir="c:\htmlparser\classes\"/>
      </classpath>
   </javac>
</target>
```

<clean>

Operatia 'clean' isi are radacinile in vechile conventii stabilite in cadrul fisierelor **makefile**: operatia presupunea stergerea fisierelor obiect pentru a forta o recompilare curata. Extrapoland aceasta idee, operatia **clean** va sterge directoarele cu clasele compilate precum si eventual fisierul jar obtinut in urma compilarii.

Este indicat ca fiecare proiect sa implementeze o operatie asemanatoare.

Exemplu:

```
<target name="clean" description="Sterge directoarele cu fisierele class">
   <delete dir=""/>
   <delete file=""/>
</target>
```

<jar>

Operatia '**jar**' are drept scop crearea de arhive jar. Atributul '**destfile**' indica locatia fisierului jar. In afara de specificarea informatiilor referitoare la sursa cat si la destinatie, este foarte indicat sa cream o dependenta: compilarea proiectului trebuie sa se efectueze inainte de crearea jar-ului.

Exemplu:

```
<target name="jar" depends="build" description="Creaza un fisier jar">
   <delete file=""/>
   <jar destfile="">
        <fileset dir = "">
            <exclude name="**/*_test.class"/>
        </fileset>
        <fileset file = "/icon.gif"/>
   </jar>
</target>
```

Proiectul contine si imagini ce trebuies incluse in fisierul jar. De asemenea, avem si cateva clase in cadrul proiectului care nu vrem sa fie incluse: este vorba de clase folosite pentru testare, de exemplu.

**/ este o expresie regulata a carei semnificatie este: orice subdirector identificat in cadrul directorului curent.

FileSet

Exemplu: Sa presupunem ca dorim sa copiem toate fisierele din directorul 'd:\student\an3\pm' in directorul 'd:\student\an3\temp'.

```
  <copy todir="d:\student\an3\temp">
     <fileset dir="d:\student\an3\pm">
        <include name="\student\an3\temp\*.gz"/>
     </fileset>
  </copy>
```

Nu se va copia nici un fisier deoarece calea specificata pe linia a treia este redundanta.

Corect este:

```
  <copy todir="d:\student\an3\temp">
     <fileset dir="d:\student\an3\pm">
        <include name="*.gz"/>
     </fileset>
  </copy>
```

Elemente avansate

In plus de operatiile relative la sistemul de fisiere cum ar fi compilarea, stergerea sau arhivarea, Ant stie sa realizeze sarcini complexe de , fiind pregatit pentru operatii in retea cum ar fi FTP, telnet, trimiterea de e-mail-uri, etc.

Pentru a pune la dispozitia utilizatorului functii de baza ale serviciului FTP, trebuie instalata libraria Jakarta Commons Net. Aceasta poate fi descarcata de la adresa http://jakarta.apache.org/commons/net/. Fisierul **commons-net-*x.x.x*.jar** (x.x.x este numarul versiunii cea mai recenta) poate fi pus in directorul $ANT_HOME\lib sau inclus in CLASSPATH.

<ftp>

Exemplu:

```
<target name="ftp" depends="jar" description="Pune fisierul jar si toate
fisierele .cfg pe server">
  <ftp action="put" server="192.168.28.33" remotedir="/home/myapp"
userid="bubu" password="secretpass" binary="yes">
     <fileset file=""/>
     <fileset dir="/">
        <include name="*.cfg"/>
     </fileset>
  </ftp>
</target>
```

Actiunea 'ftp' va fi dependenta de realizarea operatiei 'jar'. Definim tipul de transfer pentru date ca fiind 'put'. Initializam conexiunea pe binary deoarece fisierul nostru nu este de tip ASCII.