Apache Ant

Apache Ant predstavlja Java alat za izgradnju aplikacija. Osnovni koncept preuzet je od *make* alata kao dominantnog u okviru C/C++ platforme. Izvršni oblik Java aplikacija dobija se kao rezultat procesiranja skriptova kojima se definiše nizovi operacija (kao što su generisanje izvornog koda, kompajliranje, pakovanje u JAR arhive i sl). Neki od osnovnih nedostataka koji su sprečili *make* alat da bude široko prihvaćen na Java platformi su:

- komande u okviru *make* skriptova su orijentisane ka određenoj *shell* implementaciji,
- implementacije proširenja za komande su specifične sa stanovišta primenjenog operativnog sistema,
- *make* skriptovi su komplikovani i osetljivi na strukturu komandi (npr. jedan prazan karakter može da promeni značenje komande).

Apache Ant je sagrađen u cilju eliminisanja nedostataka *make* alata:

- upotreba *shell* komandi zamenjena je upotrebom XML konfiguracionih datoteka koje definišu nizove zadataka koji se izvršavaju u cilju izgradnje aplikacija,
- proširenja se definišu uz upotrebu Java klasa i
- obezbeđena je potpuna prenosivost uz oslonac na Java platformu.

Instalacija Apache Ant

U cilju instalacije *Apache Ant* alata potrebno je ispratiti sledeći niz koraka[1]:

1. Raspakovati *Ant* instalacioni arhiv na proizvoljnoj lokaciji na disku,
2. Dodati putanju do `bin` direktorijuma Ant instalacije u sistemsku promenjivu `PATH`,
3. Napraviti novu sistemsku promenjivu `ANT_HOME` i dodeliti joj vrednost koja predstavlja punu putanju do direktorijuma u kojem je *Ant* arhiva raspakovana i
4. Napraviti novu sistemsku promenjivu `JAVA_HOME` i dodeliti joj vrednost koja predstavlja punu putanju do direktorijuma u kojem je Java instalirana.

Uz pretpostavku da je izvršni oblik *Apache Ant* alata raspakovan u direktorijumu `D:\javatools\apache-ant-1.7.0`, a Java razvojna platforma instalisana u direktorijumu `C:\jdk1.5.0_12` potrebno je u okviru komandne linije izvršiti sledeći niz komandi (*Windows* platforma) kao bi *Ant* alat bio spreman za upotrebu[2]:

```
set ANT_HOME= D:\javatools\apache-ant-1.7.0
set JAVA_HOME= C:\jdk1.5.0_12
set PATH=%PATH%;%ANT_HOME%\bin
```

[1] Ukoliko se *Apache Ant* skriptovi koriste u okviru Eclipse okruženja posebna instalacija nije potrebna. Instalaciona verzija Eclipse okruženja se isporučuje zajedno sa integrisanim *Apache Ant* alatom.

[2] Efekat komandi koje su izvršene u okviru komandne linije je ograničen samo na trenutnu sesiju. Ukoliko želite da pomenute sistemske vrednosti budu perzistentne potrebno ih je dodati u prozoru Environment Variables (Windows platforma; Control Panel>System>Advanced>Environment Variables) ili u datoteku /etc/profile (Linux platforma).

Apache Ant skriptovi

Apache Ant skriptovi se definišu u obliku XML datoteka. Podrazumevani naziv skripta je *build.xml* a moguće je definisati i drugačije ime. Svaka skript datoteka se sastoji od jednog **projekta** (*project*). Projekat se sastoji od jednog ili više **ciljeva** (*target*), a ciljevi se sastoje od jednog ili više **zadataka** (*task*). Ciljevi predstavljaju logičke particije svih zadataka u okviru projekta. Svaki od ciljeva je kandidat za početnu tačku izvrašavanja skripta. Zadaci predstavljaju atomične poslove (npr. generisanje izvornog koda, kompajliranje, kreiranje arhiva ali i slanje elektronske pošte, pristup bazi podataka i sl).

```
<?xml version=”1.0”?>
<project name=”primer” default=”C” basedir=”.”>
      <target name=”A”/>
      <target name=”B” depends=”A”/>
      <target name=”C” depends=”B”/>
      <target name=”D” depends=”C, B, A”/>
</project>
```

Primer 1 – Jednostavan Ant skript

Sa primera 1 može da se uoči da XML element `<project>` definiše i tri atributa. Vrednost atributa *name* predstavlja ime projekta, atribut *default* definiše podrazumevani početni cilj (ukoliko se eksplicitno ne navede koji je početni cilj) dok vrednost atributa *basedir* predstavlja osnovni direktorijum u odnosu na koji se izvode sve kalkulacije vezane za direktorijumske putanje (ukoliko se ne navede njegova vrednost uzima se u obzir roditeljski direktorijum u kojem se nalazi skript). Od ova tri atributa jedino je *default* obavezan.

Ciljevi mogu da budu međusobno zavisni. Vrednost atributa *depends* specificira od kojih ciljeva je posmatrani cilj zavisan. Njegova vrednost predstavlja listu imena ciljeva odvojenu zarezima. Na ovaj način se definiše redosled izvršavanja ciljeva. Ant pokušava da izvrši ciljeve u zavisnost od redosleda u kojem se pojavljuju u okviru vrednosti *depends* atributa (sleva na desno).

Ukoliko se pokuša sa izvrašanjem skripta u okviru primera 1 izlaz na konzoli će da bude sledeći (Windows platforma):

```
E:\>ant
Buildfile: build.xml

A:
B:
C:

BUILD SUCCESSFUL
Total time: 0 seconds
E:\>
```

Prilikom pokretanja izvršavanja skripta iz komandne linije nije naveden početni cilj. *Ant* će na osnovu vrednost *default* atributa elementa `<project>` da ustanovi da treba da krene sa izvršavanjem od cilja C. Ali cilj C zavisi od cilja B (vrednost atributa *depends*) pa bi trebalo prvo

da se izvrše zadaci u okviru cilja B (koji je za sada prazan, kao i svi ostali ciljevi). Istom logikom dolazimo do zaključka da pre B mora da se izvrši cilj A. Pošto cilj A ne zavisi od drugih ciljeva proces rezolucije terminira. Izvršavanje ciljeva je u obrnutom redosledu u odnosu na proces rezolucije. Znači prvo će da se izvrši cilj A, pa za njim cilj B, pa posle njega cilj C. Pošto cilj D nije pronašao svoje mesto u okviru procesa rezolucije neće biti izvršen.

Ukoliko se pak navede početni cilj prilikom pokretanja skripta u okviru primera 1 izlaz na konzoli će da bude sledeći:

```
E:\>ant D
Buildfile: build.xml

A:
B:
C:
D:

BUILD SUCCESSFUL
Total time: 0 seconds
E:\>
```

Proces rezolucije u ovom slučaju počinje od cilja D jer je tako eksplicitno navedeno u komandnoj liniji. Cilj D zavisi od sva tri cilja, a prema već spomenutom pravilu u proces rezolucije krećemo s leva na desno. Primećujemo da se u okviru procesa rezolucije u ovom slučaju više puta javljaju ciljevi B i A, a da je izlaz na konzoli pokazao da se oni izvršavaju samo jednom. Ovaj primer oslikava još jedno od pravila *Ant* skriptova. Cilj se uvek izvršava samo jednom iako više atributa može da ga referencira (implicitno, putem procesa rezolucije ili eksplicitno).

Konstante

Jedan od načina da se parametrizuje izvršavanje projekta je upotreba **konstanti** (*properties*). Konstante mogu da se definišu u skript datoteci upotrebom <*property*> zadatka (koji se izvršava pre bilo kog cilja) ili pak izvan skripta (mogu da se proslede skriptu kroz komandu liniju tokom pokretanja izvršavanja skripta ili u obliku datoteke konstanti - *property file*). Konstante se sastoje od imena i vrednosti. *Ant* propisuje da je ime osetljivo na veličinu karaktera (*case sensitive*). Vrednosti konstanti su dostupne zadacima putem operatora ${ime konstante} i određuju se u toku izvršavanja skripta (*run-time*).

Postoji čitav niz podrazumevanih konstanti kao što su:

- `basedir` – apsolutna putanja do osnovnog direktorijuma projekta (uzima se u obzir vrednost atributa *basedir* elementa <*project*>),
- `ant.file` – apsolutna putanja do skript datoteke,
- `ant.version` – verzija Ant alata,
- `ant.project.name` – ime projekta koji se trenutno izvršava (uzima se u obzir vrednost *name* atributa <*project*> elementa) i
- `ant.java.version` – verzija JVM u okviru koje se izvršava Ant skript.

Izvršavanje ciljeva može da se odloži ako neka konstanta nije definisana. Uvode se atributi *if* i *unless* u okviru `<target>` elementa koji se koriste na sledeći način:

```
<target name="A" if="test-property"/>
<target name="A" unless="test-property"/>
```

Prvi cilj se izvršava samo u slučaju da je definisana konstanta `test-propert` (vrednost nije bitna), dok se drugi cilj izvršava u slučaju kada konstanta nije definisana.

Zadaci

Zadatak reprezentuje konkretnu atomičnu jedinicu procesiranja. Predstavlja se odgovarajućim XML fragmentom. Atributi zadatka mogu da referenciraju konstante. Evaluacija vrednosti konstanti se odvija pre izvršavanja zadatka. Postoji skup ugrađenih zadataka, dok je pisanje novih zadataka jednostavna operacija (implementacija odgovarajućih Java klasa).

```
<target name="compile" depends="generate">
      <mkdir dir="${bin.dir}"/>
      <javac destdir="${bin.dir}" debug="on">
            <src path="${ejb.jar}"/>
      </javac>
</target>
```

Primer 2 – Zadatak koji definiše kompajliranje izvornog koda

Primer 2 prikazuje cilj koji se sastoji od dva zadatka. Prvi zadatak je definisan elementom `<mkdir>` i omogućuje da se kreira novi direktorijum. Naziv novog direktorijuma određuje vrednost atributa *dir*. Drugi zadatak ima za cilj da prevede u izvršni oblik sav izvorni kod koji se nalazi u folderu na koji pokazuje vrednost konstante *ejb.jar* (definisano elementom `<src>` u okviru `<javac>` elementa). Prevedene datoteke se smeštaju u direktorijum koji je definisan atributom *destdir*[3].

Putanje

Liste direktorijumskih putanja kao i putanje do Java biblioteka i klasa (*classpath*) mogu da se specificiraju uz oslonac na `<path>`, odnosno, `<classpath>` zadatke. Pri tome je moguće koristiti separatore ; (Windows konvencija) ili : (Linux konvencija). *Ant* je sposoban da prevede odgovarajući separator u konkretan oblik u zavisnosti od operativnog sistema na kojem se izvršava.

```
<classpath>
      <pathelement path="${classpath}">
      <pathelement location="lib\helper.jar">
</classpath>
```

Primer 3 – Definisanje classpath vrednosti

[3] Struktura i sadržaj elemenata koji definišu odgovarajuće zadatke je promenjiv. Konsultovati *Apache Ant* dokumentaciju za tačne definicije zadataka.

Atribut *location* elementa `<pathelement>` sa primera 3 specificira jednu datoteku ili direktorijum (čija putanja je relativna u odnosnu na vrednost *basedir* atributa) dok atribut *path* definiše niz lokacija koje su odvojene separatorima : ili ;.

Kao dodatak prethodno spomenutom zadatku `<path>` mogu da se koriste i zadaci `<dirset>`, `<fileset>` i `<filelist>` koji se poput `<pathelement>` element navode unutar `<classpath>`, odnosno, `<path>` elemenata.

```
<classpath id="classpath1">
      <pathelement path="${classpath}">
      <pathelement location="${classes}">
      <fileset dir="lib">
            <include name="**/*.jar">
      </fileset>
      <dirset dir="${build.dir}">
            <include name="apps/**/classes"/>
            <exclude name="apps/**/*Test*"/>
      </dirset>
</classpath>
```

Primer 4 – Definisanje classpath vrednosti upotrebom `<fileset>` i `<dirset>` elemenata

Primer 4 predstavlja zadatak koji definiše putanju do Java biblioteka i klasa koja sadrži vrednost konstante `classpath`, sve biblioteke iz `lib` direktorijuma, direktorijum `classes` kao i sve `classes` direktorijume koji se nalaza unutar `apps` direktorijuma na bilo kojoj dubini hijerarhije (**), osim onih koji imaju vrednost `Test` u svom imenu.

Ukoliko nam je želja da koristimo istu vrednost za listu direktorijuma i Java biblioteka u okviru više zadataka moguće je definisati elemente `<path>` i `<classpath>` na istom nivou kao i ciljeve i dodeliti vrednost atributu *id*. Ista vrednost može da se upotrebi kao referenca u odgovarajućem zadatku. Primer 5 prikazuje cilj koji u okviru sebe definiše zadatak koji se bavi kompajliranjem Java koda pri čemu kao vrednost classpath-a referencira odgovarajući `<classpath>` sa primera 4.

```
<target name="compile" depends="generate">
      <javac destdir="${bin.dir}" debug="on" classpathref="classpath1">
            <src path="${ejb.jar}"/>
      </javac>
</target>
```

Primer 5 – Referenciranje `<classpath>` definicije na osnovu vrednosti atributa *id*