

# Apache Ant – Atomski mrav u programiranju

Programiranje je proces koji je svima nama dobro poznat. Uglavnom se sastoji od tačnog, nedvosmislenog predstavljanja sopstvenih ideja u strogo formalnom obliku nekog programskog jezika. Programeri se trude da savladaju što naprednije i što izražajnije programske jezike kako bi na što lakši i prirodni način iskazali svoje ideje. Medjutim, pored kreativnog dela, programiranje u sebi sadrži i jedan manje zanimljiv deo koji je uvek pravio muku programerima. Medju njima najpoznatiji je proces kompajliranja aplikacije. Potrebno je da se izvrše određena podešavanja sistema i da se koriste određeni argumenti kompajlera kako bi se ceo proces uspešno završio. Medjutim, kompajliranje predstavlja samo jedan deo celokupnog ciklusa u kome se programiranje odvija. Pored njega postoje i druge aktivnosti koje znaju da zadaju glavobolju programeru. Različita podešavanja raznih servera, pokretanje različitih testova i kreiranje test okruženja, upload-ovanje na server sa source kodom i sl. Svaki programer obavlja nekoliko poslova pre početka programiranja, zatim nekoliko pre i posle svakog konkretnog kompajliranja i pokretanja aplikacije i nekoliko poslova koji se obavljaju s' vremena na vreme.

Prilikom razvoja tipične aplikacije skoro svaki java programer se sreće sa nekoliko standardnih koraka:

- Preuzimanje programskog koda sa servera za čuvanje koda (CVS, Subversion i sl.)
- Priprema procesa generisanja (build-ovanja) projekta
- Kompajliranje source koda
- Završavanje procesa build-ovanja (kopiranje svih potrebnih fajlova (slika, property fajlova i sl.)
- Pokretanje testova (JUnit i sl.)
- Pokretanje sour
- Pravljenje javad
- Pravljenje jar da
- Instaliranje proj (web ili aplikaci

Ovo je uprošćen programa. Svaki o od nekoliko oper. za pripremu koji potrebno obrisati verziju programa postoji, na odgovarajuće direktorijume, staviti na classpath sve zavisne jar datoteke bez

kod kojih se ne bi uspešno kompajliralo i sl. Za rešavanje ovog problema postoji nekoliko načina:

Prvi način je da napravimo .bat ili .sh skripte koje bi sadržale

naredbe komandne linije za svaki ovaj korak. Medjutim, za svaki korak ne mora da postoji komandno-linijski program tako da je ovo delimično rešenje koje je pritom zavisno od sistema. Ukoliko promenimo sistem morali bismo da vršimo prepravljavanje ovih skripti.

Drugo rešenje je da koristimo neko naprednije razvojno okruženje (Eclipse, Netbeans i sl.) da bismo automatizovali proces. Ipak, ovo je isto samo delimično rešenje pošto razvojna okruženja imaju ograničen skup najčešće korišćenih operacija koje se mogu automatizovati, a ujedno razvoj softvera postaje zavistan od razvojnog okruženja što je definitivno nešto što bi trebalo da izbegnemo.

Treće rešenje podrazumeva korišćenje nekog alata za automatizaciju build procesa. Na ovaj način bismo obezbedili portabilnost našeg projekta sa sistema na sistem i njegovu nezavisnost od razvojnog okruženja. U Java svetu de-facto standardni alat za automatizaciju build procesa je Ant, open source alat koji je razvila Apache Software Fondacija.

Ant je dobitnik velikog broja nagrada što govori o kvalitetu ovog programa. Medju njima su najznačajnije: Java Pro 2003 Readers Choice Award for Most Valuable Java Deployment Technology, Java Developers Journal Editors Choice Award, JavaWorld Editors' Choice Award for Most Useful Java Community-Developed Technology 2002 i 2003 i Software Development magazine's 2002 Productivity Award.

Ovaj alat se bazira na korišćenju standardnih XML datoteka u kojima se navode definicije svakog potrebnog koraka, a pozivanje odgovarajuće aktivnosti je vrlo jednostavno, bilo kroz komandnu liniju ili kroz razvojno okruženje.

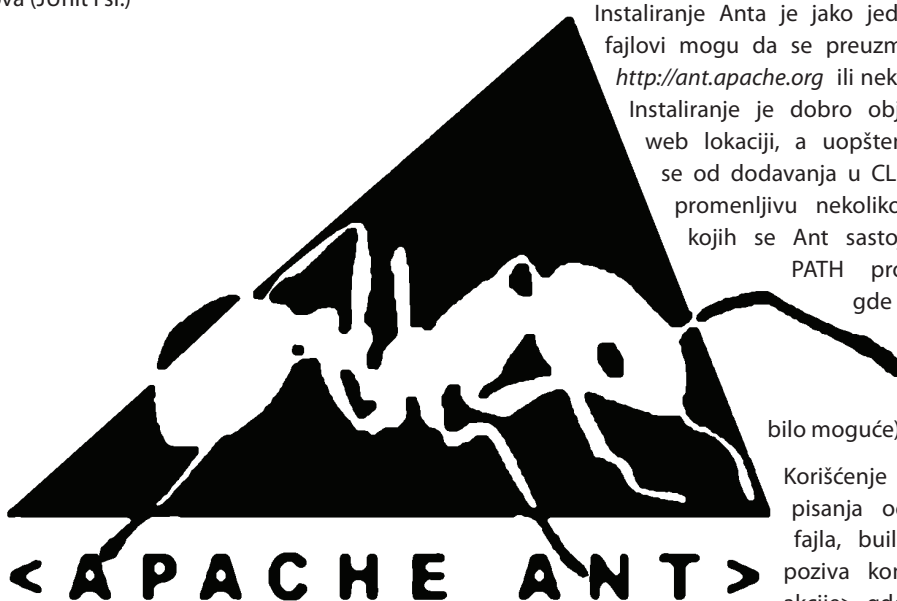
Instaliranje Anta je jako jednostavno. Potrebni fajlovi mogu da se preuzmu sa web lokacije <http://ant.apache.org> ili nekog lokalnog mirora.

Instaliranje je dobro objašnjeno na samoj web lokaciji, a uopšteno gledano sastoji se od dodavanja u CLASSPATH sistemsku promenljivu nekoliko jar datoteka od kojih se Ant sastoji i dodavanja na PATH promenljivu lokacije gde je Ant instaliran (da bi korišćenje komande ant u komandnoj liniji bilo moguće).

Korišćenje alata se sastoji od pisanja odgovarajućeg XML fajla, build.xml, a zatim se poziva komanda ant <naziv akcije> gde je <naziv akcije>

konkretna akcija koju želimo da izvršimo. Tipičan build.xml fajl se sastoji od nekoliko delova.

Glavni XML tag Ant je <project> tag. On sadrži nekoliko



atributa. Među njima najznačajniji je basedir atribut pomoću koga navodimo direktorijum u kome se nalazi projekat. Uobičajena praksa je da se build.xml fajl nalazi u osnovnom direktorijumu projekta, tako da se basedir atribut podesi na ''što označava da je osnovni direktorijum projekta tamo gde je build.xml fajl.

Svaki projekat se sastoji od nekoliko target-a. Svaki target ima nekoliko atributa od kojih je samo jedan obavezan, a to je name atribut kojim definišemo ime targeta da bismo kasnije mogli da ga pozovemo iz komandne linije. Za svaki target možemo atributom depends da definišemo listu targeta od kojih on zavisi i koji će biti pozvani pre njegovog izvršenja (na taj način možemo da definišemo targete makroe koji ne sadrže nikakvu konkretnu operaciju već samo zavise od nekoliko drugih target-a)

Svaki target se dalje sastoji od nekoliko task-ova kojima izvršavamo željenu akciju. U Antu postoje ugrađeni taskovi za veliki broj akcija (javac, java, jar, javadoc, za pravljenje kompresovanih datoteka, rada sa fajl sistemom, logovanje, j2ee servisima i sl.) ali je i moguće pravljenje novog taska pomoću odgovarajuće java klase. Na taj način je veliki broj third party kompanija napravilo za svoje proizvode odgovarajuće Ant task java klase koje se veoma lako integrišu u sam Ant (korišćenjem taskdef taska) a zatim koristiti kao i sve druge taskove.

Pored targeta, Ant project tag može da sadrži i property-je. Property bi bio pandan promenljivoj u programskim jezicima. Na primer, definisanje property-ja build-a kojim se određuje kako će se zvati direktorijum u koji želimo da smeštamo kompajlirane java klase. Ukoliko želimo naknadno da promenimo ime direktorijuma dovoljno je da promenimo vrednost property-ja i svi taskovi će automatski koristiti novu vrednost.

Postoje i path-like strukture koje se definišu tagom <path>. One mogu biti korisne za definisanje classpath i sličnih promenljivih koje treba da sadrže neku listu datoteka (npr. listu svih jar biblioteka iz nekog direktorijuma). Da ne bi svaki put kada dodamo novi jar fajl menjali classpath, možemo samo da definišemo path-like strukturu tako da sadrži sve jar datoteke nakon čega više ne moramo da vodimo računa o njima.

Sledeći primer predstavlja jedan tipični Ant build.xml fajl. Za kompajliranje, sve što je potrebno je da unesemo komandu ant compile, a za pravljenje jar datoteke ant dist. Ovo značajno olakšava proces razvoja softvera, što omogućava da se programer posveti samom problemu aplikacije. Takođe, sva današnja moderna okruženja za razvoj java aplikacija imaju odličnu integraciju za Ant, tako da je moguće recimo uvesti build.xml fajl u Eclipse, a zatim dvostrukim klikom izvršiti željeni task.

Sama dokumentacija na Antovoj web lokaciji je odlična i jako lako može da se nađe opis svakog taska koji je ugrađen u Ant. Postoje i tipični primeri korišćenja za većinu taskova tako da je jako lako dodati svom projektu neki task za novu funkcionalnost. Za odgovarajuće tipove aplikacija (npr. Struts web aplikacije) postoje već gotovi Ant build.xml fajlovi koji sadrže sve karakteristične taskove i sve što je potrebno je da se dodaju taskovi specifični za konkretnu aplikaciju.

```
<project name="Seminarski" default="compile"
  basedir=".">
  <description>
    Seminarski iz Projektovanja Programa
  </description>

  <!-- vrednosti koje koristimo na velikom broju
  mesta -->
  <property name="src" location="src"/>
  <property name="build" location="build"/>
  <property name="lib" location="lib"/>
  <property name="dist" location="dist"/>
  <property name="mainClass"
    location="seminarski.Main"/>

  <!-- definisanje path-a koji sadrzi sve iz
  ${build} i ${lib} direktorijuma -->
  <path id="klase.path">
    <pathelement location="${build}"/>
    <fileset dir="${lib}">
      <include name="**/*.jar"/>
    </fileset>
  </path>
  <target name="compile"
    description="Kompajliranje samog
      programa iz ${src} u
      ${build}" >
    <!-- Kreiranje direktorijuma u koji se
    smestaju .class file-ovi -->
    <mkdir dir="${build}"/>
    <javac srcdir="${src}" destdir="${build}" >
      <classpath refid="klase.path"/>
    </javac>
  </target>

  <target name="run" depends="compile"
    description="Pokretanje samog programa
      iz ${build}" >
    <java classname="${mainClass}">
      <classpath refid="klase.path"/>
    </java>
  </target>

  <target name="dist" depends="compile"
    description="kreiraj jar file" >
    <!-- Kreiraj direktorijum lib u build
    direktorijumu i stavi sve iz
    ${build} seminarski.jar -->
    <mkdir dir="${dist}/lib"/>
    <jar jarfile="${dist}/lib/seminarski.jar"
      basedir="${build}"/>
  </target>

  <target name="clean"
    description="obrisi direktorijume" >
    <!-- Obrisi ${build} i ${dist} direktorijume -->
    <delete dir="${build}"/>
    <delete dir="${dist}"/>
  </target>

  <!-- makro target da izvrshimo sve naredbe
  odjednom. (run i dist pozivaju compile
  tako da je ne moramo navesti)-->
  <target name="all" depends="clean,run,dist" />
</project>
```

Joe Ottinger-a, donedavni glavni uredni Java Developers Journala, je Ant opisao na sledeći način: „Ant is the hammer of the Java world: without it, civilization might have progressed, but much more slowly than it has. Ant is one of the most useful build tools I have ever had the pleasure to use.“ Ant je zaista jedan sjajan alat koji bi trebalo da se nalazi pri ruci svakom java programeru kako bi proces programiranja učinio što prijatnijim i kreativnijim i omogućio usmerenje misli na prave stvari.

Vitimir KOVANOVIĆ  
kovanovic@gmail.com