Instrumente de build

George Popa

ThoughtWorks®

Instrumente De Build

Build automation este procesul de creare automată a componentelor software pornind de la codul sursă și dependențe.

Fazele uzuale ale procesului de build:

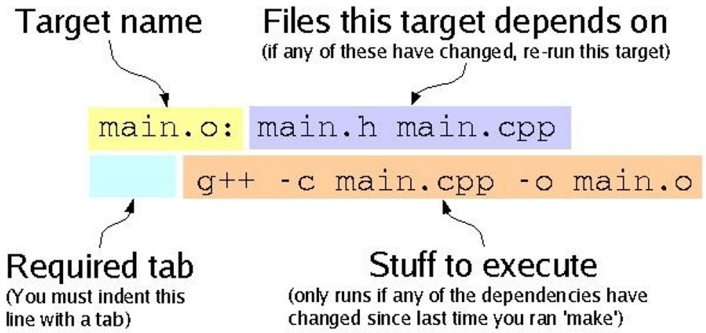
- Instalare dependențe
- Compilare cod sursă în cod binar
- Împachetare cod binar
- Rulare aplicație
- Rulare teste automate
- Revenire la starea inițială

Instrumente De Build

- GNU Make - Utilizat pe sisteme UNIX, de regulă pentru limbajele C/C++, 1976 GNU MAKE
- Apache Ant - Instrument de build Java, 2000 APACHE ANT
- Apache Maven - Instrument de build, raportare și documentare a proiectului, utilizând plug-ins. MAVEN
- Gradle - Instrument de build cu spectru larg, utilizat în general pentru proiecte Java & Scala, C/C++ și Android (default build tool). GRADLE

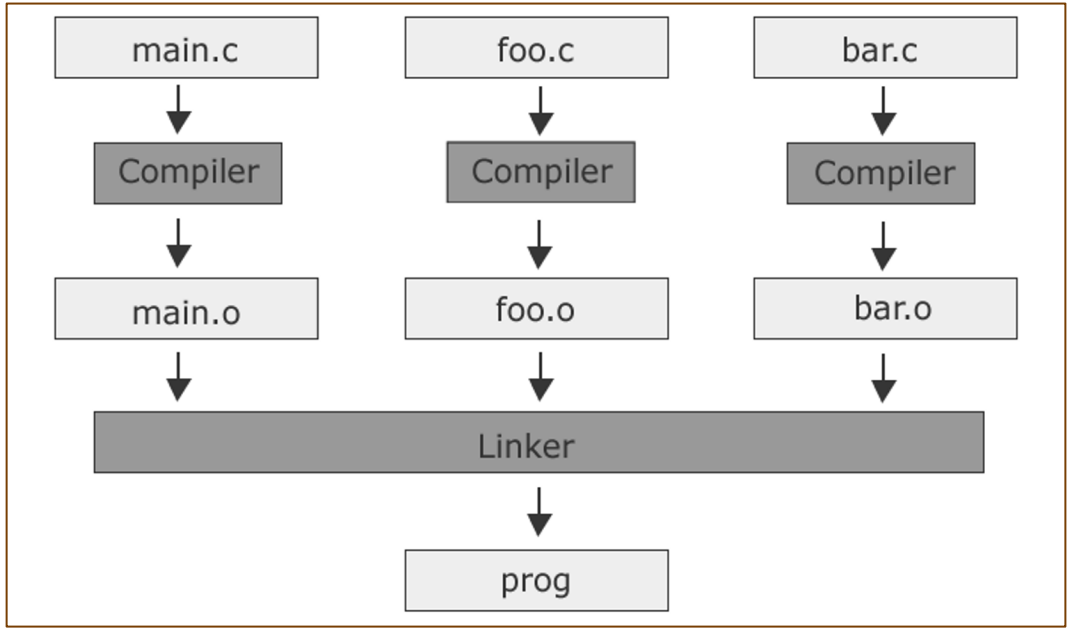
GNU Make

- Instrument de automatizare a build-ului pentru aplicații C/C++
- Utilitarul `make` citește fișiere de tip Makefile pentru execuție rules
- Fiecare rule conține un target, o lista de dependențe si o secvența de comenzi de executie:



Make File

```
prog: main.o foo.o bar.o
        gcc -o prog main.o foo.o bar.o
main.o: main.c
        gcc -c main.c
foo.o: foo.c
        gcc -c foo.c
bar.o: bar.c
        gcc -c bar.c
```



Make File

```
all: compile run

compile:
        gcc -Wall -o calculator calculator.c add.c substract.c

run:
        ./calculator

clean:
        rm -f calculator
```

- make compile
- make run
- make clean
- make

Apache Ant

- Similar cu make în multe aspecte
- Build.xml reprezintă fișierul principal de definiție a *target*-urilor.
- Fără suport inițial pentru *target*-uri uzuale, ci totul este definit de utilizatori, uneori în fișiere uriașe.
- Managementul dependențelor a fost implementat ulterior în proiectul Apache Ivy.

```
<project>
    <target name="clean">
        <delete dir="classes" />
    </target>

    <target name="compile" depends="clean">
        <mkdir dir="classes" />
        <javac srcdir="src" destdir="classes" />
    </target>

    <target name="jar" depends="compile">
        <mkdir dir="jar" />
        <jar destfile="jar/HelloWorld.jar" basedir="classes">
            <manifest>
                <attribute name="Main-Class"
                  value="antExample.HelloWorld" />
            </manifest>
        </jar>
    </target>

    <target name="run" depends="jar">
        <java jar="jar/HelloWorld.jar" fork="true" />
    </target>
</project>
```

Apache Maven

- **Evoluția naturală a Ant, instrument de build și management al dependențelor**
- **pom.xml reprezintă fișierul principal de definiție a *phase*-urilor.**
- **Comenzile maven uzuale sunt deja built-in, nu trebuiesc definite și se bazează pe plug-ins.**
- **Există o gamă largă de plug-ins dezvoltate pentru fiecare etapă de build: https://maven.apache.org/plugins/**

```
<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance”
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
     http://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <groupId>baeldung</groupId>
   <artifactId>mavenExample</artifactId>
   <version>0.0.1-SNAPSHOT</version>
   <description>Maven example</description>

   <dependencies>
      <dependency>
         <groupId>junit</groupId>
         <artifactId>junit</artifactId>
         <version>4.12</version>
         <scope>test</scope>
      </dependency>
   </dependencies>
</project>
```

Gradle - Open Source Build Automation

- **Construit utilizând conceptele Apache Ant și Apache Maven**
- **Folosește limbajul Groovy sau Kotlin, înlocuind fișierele XML (build.xml, pom.xml) - cu scopul de a ușura utilizatorilor umani înțelegerea build.gradle**
- **Compatibilitate cu fisierele Ant build.xml**
  - **ant.importBuild 'build.xml'**
  - **gradle ant.targ**
- **Oferă suport pentru migrare de la Maven**
  - **gradle init**

```
apply plugin: 'java'

repositories {
    mavenCentral()
}

jar {
    baseName = 'gradleExample'
    version = '0.0.1-SNAPSHOT'
}

dependencies {
    testImplementation 'junit:junit:4.12'
}
```

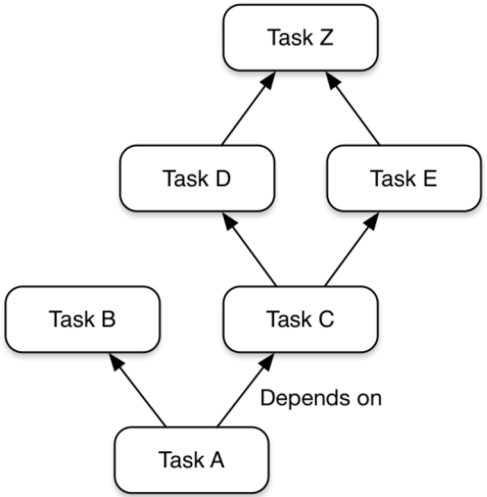
Gradle - Open Source Build Automation

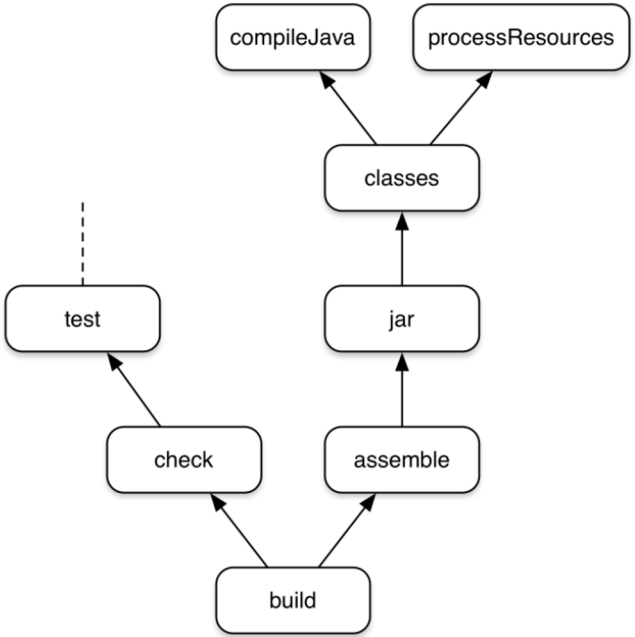
- Task-urile uzuale sunt deja definite in plug-ins:
  - Language plugins (Java, Groovy, Scala, Antrl) - executate în JVM
  - Incubating language plugins (Assembler, C, C++, Objective-C, Windows-resources)
  - Integration plugins (application, ear, jetty, maven, war)
  - IDE plugins (eclipse, idea, sonar)
  - Third party plugins
- Documentație: http://gradle.org/getting-started-gradle-java/

Gradle - Open Source Build Automation

- **Deși rămâne folosit cu preponderență în JVM, Gradle suportă orice tip de proiect software**
- **Folosește sistemul local de fișiere sau repository Maven si Ivy pentru managementul dependențelor**
- **Task-urile Gradle sunt executate folosind grafuri aciclice dirijate, optimizând astfel execuția**

Gradle Tasks Graph





Gradle -q viewLifecycle

Let’s have fun!

Prerequisites:

- **Java 8 SDK**
- **Gradle**

Execute inside new folder:

- **gradle init**
- **gradle tasks**
- **gradle clean build**
- **gradle clean install**
- **Docs:**
  - **https://docs.gradle.org**
  - **https://www.jetbrains.com/help/idea/gradle.html**