Modern build eszközök

Pataki Norbert

ELTE Informatikai Kar,
Programozási Nyelvek és
Fordítóprogramok Tanszék
patakino@elte.hu

Projekt eszközök

Tulajdonságok

- Platform-függetlenség
- XML-alapú szintaxis
- Absztrakció: dependency
- Paradigma: nem feltétlenül imperatív

Operációs rendszer parancsok

- Törlés: `rm`/`del`
- Másolás: `cp`/`copy`
- ...

Tulajdonságok

- Imperatív
- Jellemzően: Java
- `build.xml`
- `http://ant.apache.org/`

build.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="projname"
         default="deftarget"
         basedir=".">

  ...

</project>
```

target

```
<project ...>

  <property name="jarname" value="projeszk.jar" />


  <target name="compile" depends="prepare">

    ...
  </target>

</project>
```

- Command line:
  `ant` **`compile`**

target

```
<project ...>

  <target name="prepare">
    <mkdir dir="classes"/>
  </target>

</project>
```

target

```
<project ...>

  <target name="compile" depends="prepare">
    <javac srcdir="src"
           destdir="classes"/>
  </target>

</project>
```

delete task

```
<project ...>

  <target name="clean">
    <delete>
      <fileset dir="classes" includes="*"/>
    </delete>
    <delete dir="classes"/>
  </target>

</project>
```

delete task

```
<project ...>

  <target name="clean" failonerror=false>
    <delete dir="classes"/>
  </target>

</project>
```

target

```
<project ...>

  <target name="jar" depends="compile">
    <jar destfile="${jarname}">
      <fileset dir="classes">
        <include name="*.class"/>
      </fileset>
      <manifest>
        <attribute name="Main-Class" value="Main"/>
      </manifest>
    </jar>
  </target>

</project>
```

target

```
<target name="compile" depends="compwsdl,init">
  <javac destdir="build/classes" debug="on">
    <src path="src/java"/>
    <src path="build/java"/>
    <include name="**/*.java"/>
    <exclude name="com/comp/xyz/applet/*.java"/>
    <classpath>
      <fileset dir="lib">
        <include name="*.jar"/>
      </fileset>
    </classpath>
  </javac>
</target>
```

File-ok másolása

```
<mkdir dir="build/war/WEB-INF/lib"/>
<copy todir="build/war/WEB-INF/lib">
  <fileset dir="lib">
     <include name="*.jar"/>
     <exclude name="servlet-api.jar"/>
     <exclude name="catalina-ant.jar"/>
     <exclude name="el-api.jar"/>
  </fileset>
</copy>
```

JVM indítás

```
<target name="test">
    <java classname="com.comp.foo.TestClient"
          jvmargs="-Xdebug server=y,suspend=n">
        <classpath>
            <fileset dir="lib">
                <include name="*.jar"/>
            </fileset>
        </classpath>
    </java>
</target>
```

Dokumentáció generálás

```
<target name="doc">
   <tstamp>
     <format property="timestamp"
             pattern="d.M.yyyy"
             locale="en"/>
   </tstamp>
   <mkdir dir="doc"/>
  <javadoc sourcepath="src"
           windowtitle="Project documentation"
           destdir="doc">
    <header><b>Project eszkozok</b></header>
    <footer>Javadocs compiled ${timestamp}></footer>
    <bottom>Copyright &#169; Norbert Pataki</bottom>
    <fileset dir="src/" includes="**/*.java" />
  </javadoc>
</target>
```

Target későbbi hívása

- A dependency-ket automatikusan meghívja, de néha a target végén kell egy másik target-et alkalmazni (pl. más build-elési beállítások miatt):

```
<antcall target="targetname"/>
```

Maven. Tulajdonságok

- Szoftver projekt menedzsment eszköz
- Projekt build, tesztek futtatása, függőségek kezelése, dokumentáció kezelése
- Csomagkezelés: Függőségek automatikus letöltése
- A build folyamat deklaratív leírása
- Fix, előre definiált könyvtárszerkezet
- Jellemzően Java, de plugin-ok segítségével más nyelveket is tud kezelni
- `pom.xml`
- `http://maven.apache.org/index.html`

Projekt

- Project Object Model
- A projektet egyértelműen azonosítja a projekt csoportjának, azonosítójának és verziójának hármasa (group, artifact id, version)
- Egy projekt több modulra bontható, a modulokat külön is tudjuk kezelni.

pom.xml

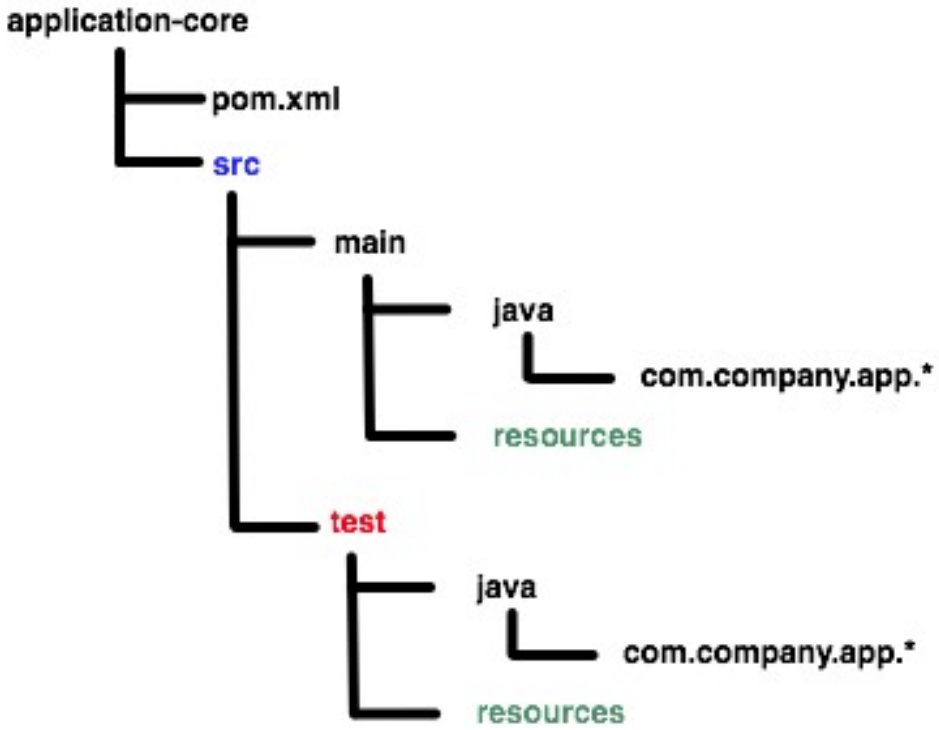
Itt adjuk meg a projekt lényeges információit:

- groupId, artifactId, version megadása
- hogyan fordítsunk
- mi legyen a fordítás eredménye
- tesztesetek
- függőségek

pom.xml

```
<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>com.mycompany.software</groupId>
  <artifactId>app</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>pom</packaging>

    ...

</project>
```

Könyvtár szerkezet



Projekt fordítás fázisai

- validate - validate the project is correct and all necessary information is available
- compile – compile the source code of the project
- test – test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
- package – take the compiled code and package it in its distributable format, such as a JAR.
- integration-test – process and deploy the package if necessary into an environment where integration tests can be run

Projekt fordítás fázisai

- verify – run any checks to verify the package is valid and meets quality criteria
- install – install the package into the local repository, for use as a dependency in other projects locally
- deploy - done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.
- Pontosabb lista: http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html
- Command line:

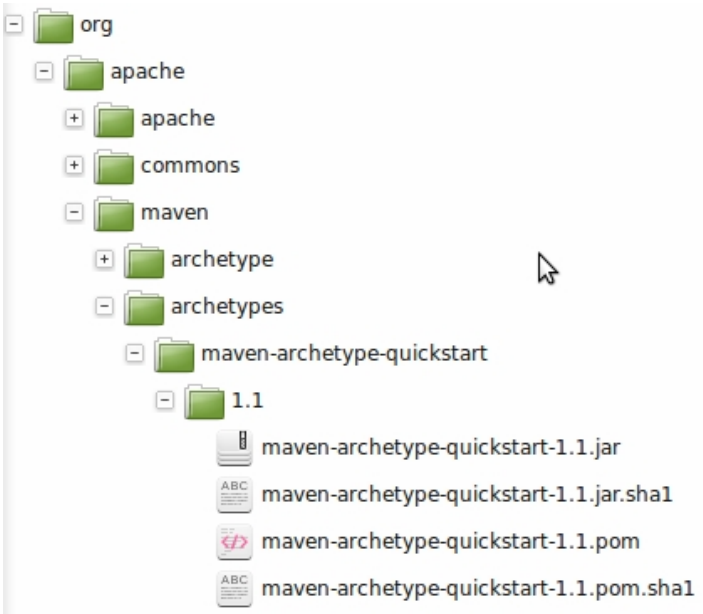
```
mvn clean install
```

Célok (goals)

- A fordítási fázisok célokból épülnek fel
- A cél egy olyan taszk, ami a projekt fordításával illetve menedzselésével kapcsolatos.
- Egy maven parancsnál több cél is megadható, ezek végrehajtásának sorrendje függ a felsorolástól és attól, hogy melyik fázishoz köthetőek.
- Célok-at (goal) lehet definiálni (a pom.xml-ben)

Repository

- Centrális: `http://repo.maven.apache.org`
- Lokális: saját gépünkön található maven repository
  - `/.m2/repository`



Repository

- Első build-eléskor letölti a függőségeket, plugin-okat, majd a lokális repository-ba tárolja el
- jelentősen megnöveli a hálózati forgalmat, lassítja a build-elési folyamatot
- Maven konfiguráció megváltoztatása, hogy az általunk megadott repo-t/mirror-t használja:
  `/.m2/settings.xml`

Buildelés eredményei

- Fordítás során keletkezik egy `target` könyvtár, ebbe kerülnek a fordítás során létrehozott fájlok.
- kimenet, pl a `my-app-1.0-SNAPSHOT.jar`
- `classes` könyvtár – osztályok, amelyek a fordítás során keletkeztek (de nem a teszt-osztályok)
- `test-classes` – osztályok, amelyek a teszt-forrásfájlokból keletkeznek
- `maven-archiver` – egy `pom.properties` file, ami leírja a projektet azonosító hármast (groupId, artifactId, version)
- `surefire-reports` – ide kerülnek a tesztelés eredményei

Project hierarchiák

```
<project ...>

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.mycompany.app</groupId>
  <artifactId>parent-app</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>pom</packaging>
  <!-- alprojektek felsorolása-->
  <modules>
    <module>first-child-app</module>
    <module>second-child-app</module>
  </modules>
</project>
```

Project hierarchiák

```
<project ...>

  <modelVersion>4.0.0</modelVersion>

  <parent>
    <groupId>com.mycompany.app</groupId>
    <artifactId>parent-app</artifactId>
    <version>1.0-SNAPSHOT</version>
  </parent>

  <groupId>com.mycompany.app</groupId>
  <artifactId>first-child-app</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>war</packaging>
    ...
</project>
```

Plugin-ok

- A maven funkcionalitása az alap dolgokra korlátozódik, bár különböző plugin-ok használatával bármit meg lehet tenni: pl C++, LaTeXfordítás, ant build, javadoc.
- Mindenki írhat magának külön plugin-t.

PI. javadoc plugin

```
<project ...>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-javadoc-plugin</artifactId>
        <version>2.8.1</version>
        <configuration>
          ...
        </configuration>
      </plugin>
    </plugins>
  </build>
  ...
</project>
```

Függőségek megadása

```
<project ...>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

- groupId + artifactId + version azonosítja a projektet, amitől függünk
- scope megadja, hogy melyik életciklusban használjuk a meghatározott függőséget

Függőségek scope-ja

- A leggyakrabban használt scope-ok:
  - compile – Ez a default, ha nincs megadva. Fordításhoz szükséges függőségek.
  - runtime – Ez azt jelzi, hogy a függőség futásidőben szükséges, fordításkor nem.
  - test – Jelzi, hogy a függőség nem szükséges a normál működéshez, de a teszt-fordításhoz és a tesztek futtatásához kell.