Apache Maven

Jeszenszky Péter
Debreceni Egyetem, Informatikai Kar
jeszenszky.peter@inf.unideb.hu

Utolsó módosítás: 2018. március 8.

Apache Maven

- Projektkezelő eszköz (*software project management and comprehension tool*), mely támogatja például az alábbiakat:
  - Projekt összeállítás (*build*)
  - Projektinformációk megjelenítése a weben
  - Kiadáskezelés (*release management*)
  - Disztribúció (*distribution*)
  - Függőségkezelés (*dependency management*)

Jellemzők (1)

- Konvenciók előtérbe helyezése az egyedi beállításokkal szemben (*convention over configuration*).
  - Például szabványos könyvtárszerkezet meghatározása.
- Projekt életciklusok és életciklus fázisok meghatározása.
- Jellegét tekintve deklaratív.
- Moduláris és kiterjeszthető felépítés.
  - Minden funkció megvalósítása bővítményekkel történik.

Jellemzők (2)

- Noha a gyakorlatban főleg Java projektekhez használják, más programozási nyelvek esetén is használható, például:
  - C/C++:
    - nar-maven-plugin http://maven-nar.github.io/
  - Kotlin:
    - kotlin-maven-plugin https://kotlinlang.org/docs/reference/using-maven.html
  - Scala:
    - scala-maven-plugin http://davidb.github.io/scala-maven-plugin/

Fejlesztés

- Programozási nyelv: Java
- Szabad és nyílt forrású: az Apache License v2 hatálya alatt terjesztik.
- A jelenleg aktuális stabil verzió a 3.5.3 számú (kiadás dátuma: 2018. március 8.). https://maven.apache.org/docs/history.html
  - Eltérések vannak a 2.*x* és 3.*x* verziók között.

Telepítés

- Az Apache Maven használatához JDK szükséges, JRE nem elegendő!
  - A JDK 7-es vagy későbbi kiadása szükséges.
    - Innen tölthető le az Oracle JDK: http://www.oracle.com/technetwork/java/javase/downloads/
  - Fontos, hogy megfelelően be legyen állítva a `JAVA_HOME` környezeti változó is!
- Letöltés: http://maven.apache.org/download.html
- A használatba vételhez a szoftvert tartalmazó archív állomány kibontása után csupán a `PATH` környezeti változót kell beállítani.

Telepítés (Linux) (1)

- Ha például az `/opt/apache-maven-3.5.3` könyvtár alá bontottuk ki a szoftvert tartalmazó állományt, akkor az alábbi környezeti változó beállítás szükséges:
  - `export PATH=/opt/apache-maven-3.5.3/bin:$PATH`
- Tipp: a beállítások elvégzéséhez hozzuk létre az `/etc/profile.d/maven.sh` állományt a fenti tartalommal.

Telepítés (Linux) (2)

- Az Apache Maven az SDKMAN! eszközzel is telepíthető, melyhez az alábbi parancsot kell végrehajtani:

  sdk install maven

- Az SDKMAN! telepítéséről lásd: http://sdkman.io/install.html

Telepítés (Windows)

- Ha például a `C:\Program Files\apache-maven-3.5.3` könyvtár alá bontottuk ki a szoftvert tartalmazó állományt, akkor az alábbi beállítás szükséges:
  - Adjuk hozzá a `PATH` környezeti változó értékéhez a `C:\Program Files\apache-maven-3.5.3\bin` könyvtárat.

Telepítés sikerességének ellenőrzése

- Hajtsuk végre a parancsértelmezőben az alábbi ekvivalens parancsok valamelyikét:

```
mvn --version
mvn -v
```

  - Sikeres telepítés esetén a program az alábbit írja a kimenetre:

```
Apache Maven 3.5.3 (3383c37e1f9e9b3bc3df5050c29c8aff9f295297;
    2018-02-24T20:49:05+01:00)
Maven home: /opt/apache-maven-3.5.3
Java version: 9.0.4, vendor: Oracle Corporation
Java home: /home/jeszy/.sdkman/candidates/java/9.0.4-oracle
Default locale: hu_HU, platform encoding: UTF-8
OS name: "linux", version: "3.13.0-37-generic", arch: "amd64", family: "unix"
```

IDE integráció

- **Eclipse**: m2eclipse http://www.sonatype.org/m2eclipse http://eclipse.org/m2e/
  - Update site: http://download.eclipse.org/technology/m2e/releases/
  - Az Eclipse IDE for Java Developers Indigo kiadása már tartalmazza a m2eclipse bővítményt, így külön telepítése nem szükséges.
- **IntelliJ IDEA**: beépített Apache Maven támogatás. https://www.jetbrains.com/help/idea/maven.html
- **Netbeans**: a 6.7 verziótól kezdve beépített Apache Maven támogatás. http://wiki.netbeans.org/Maven

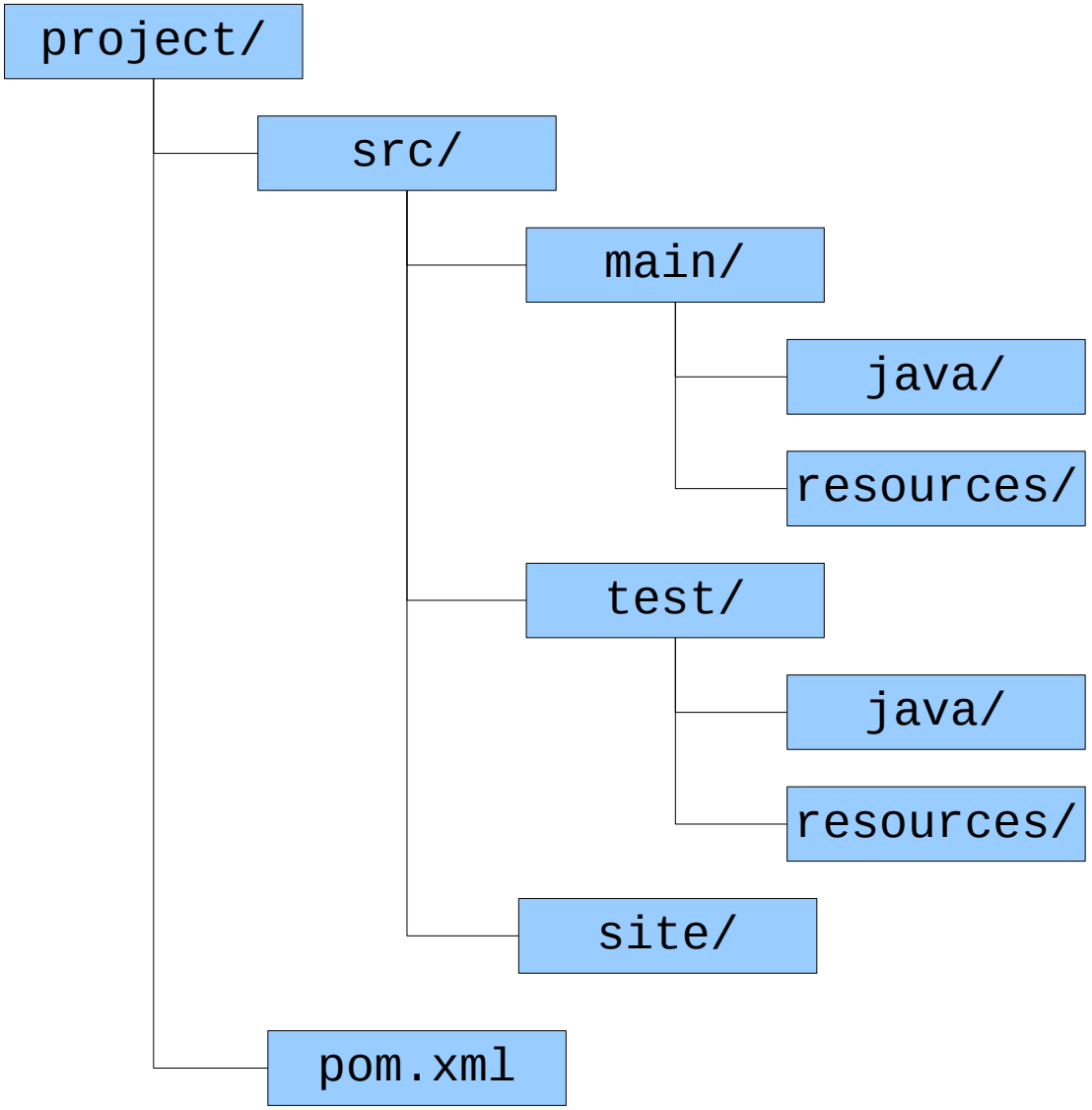
További információk

- Szabadon, Creative Commons licenc hatálya alatt elérhető könyvek http://www.sonatype.org/nexus/resources/resources-book-links-and-downloads/
  - *Maven by Example*
  - *Maven: The Complete Reference*
  - *Repository Management with Nexus*
- Levelezési listák: http://maven.apache.org/mail-lists.html

Projekt könyvtárszerkezet (1)

- Szabványos könyvtárszerkezet meghatározása a projektek számára.
  - Lásd: *Introduction to the Standard Directory Layout* http://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html

Projekt könyvtárszerkezet (2)



Használat (1)

- A használat módjáról és a megadható parancssori opciókról az `mvn --help` vagy `mvn -h` parancsok végrehajtásával kaphatunk leírást.
- Parancssori argumentumként megadható életciklus fázis (például `mvn package`) és *előtag*`:`*cél* formában bővítmény-cél (például `mvn site:run`).
  - Tetszőleges sok ilyen argumentum adható.
  - A végrehajtáshoz paramétereket rendszertulajdonságokkal adhatunk meg `-D`*név*`=`*érték* formában.

Használat (2)

- Bővítmény-cél megadható *groupId*:*artifactId*:*verzió*:*cél* formában is.
  - Akkor lehet szükséges így hivatkozni egy bővítmény-célt, ha a bővítmény adott számú verzióját kell használni, vagy a Maven nem tudja, hogy az előtag melyik bővítményhez tartozik.
    - Például: `org.apache.maven.plugins:maven-archetype-plugin:3.0.1:generate`

Maven Help Plugin (1)

- Bővítmény, mellyel információk kérdezhetők le bővítményekről, a futtató környezetről és a projektekről.
  - További információk: http://maven.apache.org/plugins/maven-help-plugin/
- A bővítmény használatáról ad leírást az `mvn help:help` parancs.
  - Részletesebb leírást kapunk, ha megadjuk a `-Ddetail=true` opciót is.
  - Csak az adott célról kapunk leírást, ha megadjuk a `-Dgoal=`*cél* opciót is.
    - Például: `mvn help:help -Dgoal=describe -Ddetail=true`

Maven Help Plugin (2)

- A `describe` cél bővítményekről, életciklusokról és életciklus fázisokról szolgáltat információkat.
  - Példák a használatra:
    - `mvn help:describe -Dplugin=org.apache.maven.plugins:maven-jar-plugin:3.0.2`
    - `mvn help:describe -Dplugin=jar`
    - `mvn help:describe -Dplugin=jar -Dgoal=sign`
    - `mvn help:describe -Dplugin=jar -Dgoal=sign -Ddetail=true`
    - `mvn help:describe -Dcmd=jar:jar -Ddetail=true`
    - `mvn help:describe -Dcmd=clean`

Maven Help Plugin (3)

- Az `effective-pom` cél az úgynevezett effektív POM-ot írja a kimenetre.
  - Az `mvn help:effective-pom` parancs végrehajtásához `pom.xml` állomány szükséges az aktuális könyvtárban.
    - Ha ez hiányzik, akkor a `-f` vagy `--file` opcióval kell megadni a POM elérési útvonalát, mint például:
      `mvn help:effective-pom -f ../project/pom.xml`

Maven Archetype Plugin (1)

- Lehetővé teszi projektek létrehozását sablon alapján.
  - Az `mvn archetype:generate` parancsot végrehajtva interaktív módon hozható létre projekt.
    - Ki kell választani egy sablont, majd meg kell adni a Maven koordinátákat és a csomagolás módját.
- További információk: http://maven.apache.org/archetype/maven-archetype-plugin/

Maven Archetype Plugin (2)

- A bővítmény 2.1 verziójától a sablon kiválasztása során minta alapján történő szűrés lehetséges.
  - A szűréshez a minta megadható a `-Dfilter=`*minta* opcióval is.
    - Példa: `mvn archetype:generate -Dfilter=android`

Maven Archetype Plugin (3)

- Néhány hasznos sablon:
  - `org.apache.maven.archetypes:maven-archetype-plugin`
  - `org.apache.maven.archetypes:maven-archetype-site`
  - `com.zenjava:javafx-basic-archetype`
  - `org.wildfly.archetype:wildfly-javaee7-webapp-archetype`
  - …

settings.xml (1)

- Projekt-független beállításokat tartalmazó konfigurációs állomány.
  - Az összes felhasználó számára globális beállításokat szolgáltat az `$M2_HOME/conf/settings.xml` állomány.
  - A globális beállítások felülírásához a felhasználók elhelyezhetnek egy saját `settings.xml` állományt a HOME könyvtáruk `.m2` alkönyvtárában.
    - Linux rendszerekben tehát `~/.m2/settings.xml` az állomány elérési útvonala.
- XML séma: http://maven.apache.org/xsd/settings-1.0.0.xsd

settings.xml (2)

- A beállítások megjelenítésére szolgál a Maven Help Plugin `effective-settings` célja.
  - Az `mvn help:effective-settings` parancs a globális és a felhasználói beállítások összefésülésének eredményét írja a kimenetre.
- Tipp: saját `settings.xml` állomány létrehozásához használjuk sablonként a globálisat.
  - Linux környezetben az alábbi módon másolhatjuk az állományt a megfelelő könyvtárba:

```
cp $M2_HOME/conf/settings.xml ~/.m2
```

Alapfogalmak

- Termék (*artifact*)
- Projekt objektum modell (POM – *Project Object Model*)
- Szuper-POM (*super POM*)
- Effektív POM (*effective POM*)
- Maven koordináták (*Maven coordinates*)
- Bővítmény (*plugin*), bővítmény-cél (*plugin goal*)
- Távoli és lokális tároló (*remote/local repository*)
- Életciklus (*lifecycle*), életciklus fázis (*lifecycle phase*)

Termék (artifact)

- A projektek termékeként előállított állományokat nevezik így.
  - Egy projektben tipikusan egy termék készül (például egy `jar` csomagolású projektben egyetlen JAR állomány).
    - Több kimeneti állomány esetén használható ezek megkülönböztetésére a `classifier` Maven koordináta.
  - Publikálásuk tárolókban történik, mely lehetővé teszi a más projektekhez történő felhasználást.

Projekt objektum modell (POM)

- Egy projekt deklaratív leírását tartalmazó XML dokumentum (`pom.xml`).
  - Metaadatokat és konfigurációs beállításokat tartalmaz.
- XML séma: http://maven.apache.org/xsd/maven-4.0.0.xsd
- A projektek között szülő-gyerek kapcsolatok definiálhatóak.
  - A gyerek projekt megörökli a szülőhöz tartozó POM beállításait, melyeket felülírhat.

Minimális POM

```
<project xmlns="http://maven.apache.org/POM/4.0.0">
   <modelVersion>4.0.0</modelVersion>
   <groupId>hu.unideb.inf.maven</groupId>
   <artifactId>maven-hello</artifactId>
   <version>1.0</version>
</project>
```

Szuper-POM

- Ha egy projektnek nincs explicit módon megadott szülője, akkor az úgynevezett szuper-POM beállításait örökli.
  - A szuper-POM a Maven által alapértelmezésben használt POM, mely számos beállításhoz szolgáltat alapértelmezett értékeket.
- A 3.*x*.*y* verziók esetén az installáció `lib/` alkönyvtára alatt található `maven-model-builder-3.`*x*`.`*y*`.jar` állomány tartalmazza `pom-4.0.0.xml` néven.

Effektív POM

- A projekthez tartozó POM, a felmenő ági projektekhez tartozó POM-ok és a szuper-POM kombinációja.
  - A futás során a projekthez ténylegesen felhasználásra kerülő beállításokat szolgáltatja.
- Az mvn help:effective-pom parancs jeleníti meg.

Maven koordináták (1)

- Minden termék azonosítása 3 vagy 4 komponensből álló azonosítókkal történik
  - Kötelező komponensek:
    - **`groupId`**: csoportazonosító, melynél gyakori a fordított domain-nevek használata (például: `org.apache.maven.plugins`), de nem kizárólagos (például: `junit`)
    - **`artifactId`**: projektnév (például `maven-assembly-plugin`, `junit`)
    - **`version`**: verziószám (például: `1.0`, `1.0-SNAPSHOT`)
  - Opcionális komponens:
    - **`classifier`**: „osztályozó” (lásd később)

Maven koordináták (2)

- A projekt POM-jában megadott `groupId`, `artifactId` és `version` elemek határozzák meg a kimenetként előállított állományok koordinátáit.
  - Explicit módon megadott szülő esetén a gyerek projekt a koordinátákat is örökli.
    - Ilyenkor tipikus a `groupId` és `version` átvétele, valamint az `artifactId` felülírása.
- A Maven koordinátákat gyakran *groupId*`:`*artifactId*`:`*version* formában írják (például: `junit:junit:4.12`).

Maven koordináták (3)

- Lehetővé teszik a függőségként történő hivatkozást, mint például:

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

Csomagolás

- A `packaging` elemben adható meg a csomagolás módja, jelenleg támogatott:
  - `pom`
  - `jar` (alapértelmezés)
  - `maven-plugin`
  - `ejb`
  - `war`
  - `ear`
  - `rar`
  - `par`

Bővítmények (1)

- Szinte minden funkciót bővítmények nyújtanak.
  - A bővítmények egy-egy funkciót megvalósító úgynevezett célokat szolgáltatnak.
- A bővítmények is termékek, melyekre a Maven koordinátákkal lehet hivatkozni.
  - Példa a POM-ban történő hivatkozásra:

```
<plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-site-plugin</artifactId>
   <version>3.7</version>
</plugin>
```

- Minden bővítményhez tartozik egy olyan előtag, mely lehetővé teszi az egyes célokra *előtag*`:`*cél* formában történő hivatkozást, mint például `site:site`.

Bővítmények (2)

- Névkonvenció:
  - A hivatalos, azaz az Apache Maven projektben fejlesztett bővítmények neve `maven-`*xyz*`-plugin` formájú, ahol *xyz* az előtag.
    - Más bővítményeknél tilos ezt a mintát követni.
  - Más bővítményeknél *xyz*`-maven-plugin` az ajánlott forma, ahol *xyz* az előtag.
- Az előtagot a bővítmények határozhatják meg a róluk metaadatokat szolgáltató `plugin.xml` állományukban.

Bővítmények (3)

- A Maven alapértelmezésben csak az `org.apache.maven.plugins` és az `org.codehaus.mojo` csoportokba tartozó bővítmények céljaira való hivatkozásokat teszi lehetővé előtagok révén.
  - Lásd a `maven-metadata-central.xml` állományokat a `$HOME/.m2/repository/org/apache/maven/plugins` és a `$HOME/.m2/repository/org/codehaus/mojo` könyvtárakban.
  - Lásd: *Introduction to Plugin Prefix Resolution* http://maven.apache.org/guides/introduction/introduction-to-plugin-prefix-mapping.html

Tárolók (1)

- A termékek, köztük a bővítmények elérése tárolókból történik, amelyeknek két fajtája van:
  - Távoli tárolók tipikusan a weben érhetők el, például HTTP vagy HTTPS protokollon keresztül.
    - Központi Maven tároló (Central Repository) https://repo.maven.apache.org/maven2
  - A lokális tároló a távoli tárolókból a felhasználó számára lokális használatra letöltött termékeket tartalmazza az állományrendszerben, valamint az `mvn install` paranccsal lokálisan telepített termékeket.
    - Gyorsítótár szerepét tölti be.
    - A felhasználó HOME könyvtárában található (Linux rendszerekben az `~/.m2/repository/` alkönyvtárban).
- A távoli és lokális tárolók azonos felépítésűek.

Tárolók (2)

- A tárolókban a csoportazonosító leképezése egy könyvtárszerkezetre.
  - Például: `org.apache.maven.plugins` → `/org/apache/maven/plugins/`
    - A könyvtárszerkezetben további alkönyvtárak, melyek neve az `artifactId` és `version` komponensek értékével egyezik meg (például: `junit:junit:4.12` → `/junit/junit/4.12/`).
- A Maven 3.*x* verziókban a bővítményekhez külön tárolók használata.

Tárolók (3)

- Szoftverek tárolók üzemeltetéséhez (*repository managemenet software*):
  - Szabad és nyílt forrású szoftverek:
    - *Apache Archiva* (Apache License v2) http://archiva.apache.org/
    - *Artifactory Open Source* (GNU GPL v3) http://www.jfrog.com/open-source/
    - *Nexus OSS* (Eclipse Public License v1.0) https://www.sonatype.com/download-oss-sonatype
  - Nem szabad szoftverek:
    - *Artifactory* http://www.jfrog.com/
    - *Nexus Repository Pro* https://www.sonatype.com/nexus-repository-sonatype

Életciklusok

- Egy életciklus jól meghatározott életciklus fázisok egy sorozatát jelenti.
  - Minden életciklus fázist egy egyedi név azonosít.
  - A fázisokhoz bővítmény-célokat lehet hozzárendelni, a hozzárendelést kötésnek nevezik.
- Az életciklus fázisok végrehajtása a hozzájuk tartozó bővítmény-célok végrehajtását jelenti.
  - Adott fázis végrehajtása maga után vonja valamennyi, a sorrendben azt megelőző fázis végrehajtását.
- Három szabványos életciklus: `clean`, `default`, `site`
  - A csomagolás módjától függően a fázisokhoz alapértelmezésben hozzárendeltek bizonyos célok.

Életciklusok: a clean életciklus

- A `clean` életciklus az alábbi három életciklus fázist tartalmazza:
  - `(1) pre-clean`
  - `(2) clean`
  - `(3) post-clean`
- A `clean` életciklus fázishoz alapértelmezésben a `clean:clean` cél van hozzákötve.
  - A cél végrehajtásának eredményeként törlésre kerülnek a projekt munkakönyvtárából az összeállítás során a Maven által létrehozott állományok.
- Lásd: *Lifecycles Reference* http://maven.apache.org/ref/current/maven-core/lifecycles.html

Életciklusok: a site életciklus

- A `site` életciklus az alábbi négy életciklus fázist tartalmazza:
  - `(1) pre-site`
  - `(2) site`
  - `(3) post-site`
  - `(4) site-deploy`
- A `site` életciklus fázishoz alapértelmezésben a `site:site` cél, a `site-deploy` életciklus fázishoz pedig a `site:deploy` cél van hozzákötve.
- Lásd: *Lifecycles Reference* http://maven.apache.org/ref/current/maven-core/lifecycles.html

Életciklusok: a default életciklus (1)

(1) validate
(2) initialize
(3) generate-sources
(4) process-sources
(5) generate-resources
(6) process-resources
(7) compile
(8) process-classes
(9) generate-test-sources
(10) process-test-sources
(11) generate-test-resources
(12) process-test-resources
(13) test-compile
(14) process-test-classes
(15) test
(16) prepare-package
(17) package
(18) pre-integration-test
(19) integration-test
(20) post-integration-test
(21) verify
(22) install
(23) deploy

Életciklusok: a default életciklus (2)

- Az alapértelmezett kötések `ejb`, `jar`, `rar` és `war` csomagolás esetén:
  - Lásd: *Plugin Bindings for default Lifecycle Reference* http://maven.apache.org/ref/current/maven-core/default-bindings.html

`process-resources`	`resources:resources`
`compile`	`compiler:compile`
`process-test-resources`	`resources:testResources`
`test-compile`	`compiler:testCompile`
`test`	`surefire:test`
`package`	`ejb:ejb / jar:jar / rar:rar / war:war`
`install`	`install:install`
`deploy`	`deploy:deploy`

Hivatkozás tulajdonságokra (1)

- A `${`*x*`}` formájú hivatkozások helyettesítése a POM-ban.
  - `${env.`*név*`}` formájú hivatkozások helyettesítése a megfelelő nevű környezeti változó értékével.
    - Például `${env.PATH}` a `PATH` környezeti változó értékét szolgáltatja.
  - A hivatkozásban megadható Java rendszertulajdonság neve.
    - Például: `${java.home}`, `${line.separator}`
  - `${project.`*x*`}` formájú hivatkozások helyettesítése a POM megfelelő elemének értékével. Csak egyszerű típusú elemekhez használható!
    - Például: `${project.groupId}`, `${project.artifactId}`, `${project.url}`, `${project.build.outputDirectory}`
  - `${settings.`*x*`}` formájú hivatkozások helyettesítése a `settings.xml` állomány megfelelő elemének értékével.

Hivatkozás tulajdonságokra (2)

- Ilyen módon hivatkozható bármely a `properties` elemben definiált tulajdonság.
  - Például:

```
<properties>
   <company.name>unideb</company.name>
</properties>
...
${company.name}
```

POM referencia (1)

- `artifactId`: a Maven koordináták projektnév komponensét tartalmazza

```
<artifactId>commons-lang3</artifactId>
```

- `build`: a projekt összeállítási beállításait tartalmazza

```
<build>
    <directory>${project.basedir}/target</directory>
    <outputDirectory>
        ${project.build.directory}/classes
    </outputDirectory>
    <finalName>
        ${project.artifactId}-${project.version}
    </finalName>
    ...
</build>
```

- `ciManagement`: a projektben használt continuous integration (CI) rendszerről tartalmaz információkat

```
<ciManagement>
    <system>continuum</system>
    <url>http://vmbuild.apache.org/</url>
</ciManagement>
```

POM referencia (2)

- `contributors`: a fejlesztőkön kívüli további közreműködőkről tartalmaz információkat

```
<contributors>
   <contributor>
      <name>Naoki Nose</name>
      <email>ikkoan@mail.goo.ne.jp</email>
      <roles>
         <role>Japanese translator</role>
      </roles>
   </contributor>
   ...
</contributors>
```

POM referencia (3)

- `dependencies`: a projekt függőségeit tartalmazza

```
<dependencies>
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-jdk14</artifactId>
        <version>1.7.25</version>
        <scope>runtime</scope>
    </dependency>
    ...
</dependencies>
```

- `dependencyManagement`: alapértelmezett verziószámokat szolgáltat függőségek kezeléséhez

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>1.7.25</version>
        </dependency>
        ...
    </dependencies>
</dependencyManagement>
```

POM referencia (4)

- `description`: a projekt leírását tartalmazza

```
<description>JUnit is a regression testing
   framework. It is used by the developer who
   implements unit tests in Java.
</description>
```

- `developers`: a projekt fejlesztőiről tartalmaz információkat

```
<developers>
   <developer>
      <id>jeszy</id>
      <name>Péter Jeszenszky</name>
      <email>jeszenszky.peter@inf.unideb.hu</email>
      <url>http://www.inf.unideb.hu/~jeszy/</url>
      <roles>
         <role>developer</role>
      </roles>
   </developer>
   ...
</developers>
```

POM referencia (5)

- `distributionManagement`: információkat tartalmaz annak a webszervernek illetve távoli tárolónak az eléréséhez, ahová a projektben előállított webhely és termékek telepítése történik

```
<distributionManagement>
   <site>
      <id>morse</id>
<url>scp://jeszy@arato.inf.unideb.hu/home/jeszenszky.peter/public_html/project/</url>
   </site>
</distributionManagement>
```

POM referencia (6)

- `groupId`: a Maven koordináták csoportazonosító komponensét tartalmazza

```
<groupId>org.apache.maven.plugins</groupId>
```

- `inceptionYear`: a projekt indulásának évét tartalmazza

```
<inceptionYear>2010</inceptionYear>
```

- `issueManagement`: a projekthez használt hibakövető rendszerről tartalmaz információkat

```
<issueManagement>
   <system>Bugzilla</system>
   <url>https://issues.apache.org/bugzilla/</url>
</issueManagement>
```

POM referencia (7)

- `licenses`: a projekthez használt szoftverlicencekről tartalmaz információkat

```
<licenses>
    <license>
        <name>GNU General Public License v3.0</name>
        <url>http://www.gnu.org/copyleft/gpl.html</url>
    </license>
    ...
</licenses>
```

- `mailingLists:` a projekt levelezési listáiról tartalmaz információkat

```
<mailingLists>
    <mailingList>
        <name>User Mailing List</name>
        <subscribe>user-subscribe@tika.apache.org</subscribe>
        <unsubscribe>user-unsubscribe@tika.apache.org</unsubscribe>
        <post>user@tika.apache.org</post>
        <archive>http://mail-archives.apache.org/mod_mbox/tika-
user/</archive>
    </mailingList>
    ...
</mailingLists>
```

POM referencia (8)

- `modules`: többmodulos projektek esetén az egyes modulok könyvtárainak relatív elérési útvonalait tartalmazza

```
<modules>
   <module>client</module>
   <module>library</module>
</modules>
```

- `name`: a projekt nevét tartalmazza „emberi fogyasztásra” szánt formában

```
<name>JUnit</name>
```

- `organization`: a projektet tulajdonló szervezet nevét és weboldalának címét tartalmazza

```
 <organization>
   <name>Faculty of Informatics, University of
      Debrecen</name>
   <url>http://www.inf.unideb.hu/</url>
</organization>
```

POM referencia (9)

- `packaging`: a projekt csomagolásának módját tartalmazza

```
<packaging>jar</packaging>
```

- `parent`: a szülő Maven koordinátáit tartalmazza

```
<parent>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-parent</artifactId>
    <version>1.7.25</version>
</parent>
```

- `pluginRepositories`: a projektben bővítményekhez használt távoli tárolókról szolgáltat információkat

```
<pluginRepositories>
    <pluginRepository>
        <id>central</id>
        <name>Central Repository</name>
        <url>https://repo.maven.apache.org/maven2</url>
        <snapshots>
            <enabled>false</enabled>
        </snapshots>
    </pluginRepository>
    ...
</pluginRepositories>
```

POM referencia (10)

- `prerequisites`: követelményeket tartalmaz az összeállítási környezetre (jelenleg csak a Maven minimális verziószáma adható meg)

```
<prerequisites>
   <maven>3.0</maven>
</prerequisites>
```

- `profiles`: összeállítási profilokat tartalmaz (lásd később)
- `properties`: Maven tulajdonságok értékeit tartalmazza

```
<properties>
   <project.build.sourceEncoding>UTF-8
      </project.build.sourceEncoding>
   <project.reporting.outputEncoding>UTF-8
      </project.reporting.outputEncoding>
   ...
</properties>
```

POM referencia (11)

- `reporting`: jelentéskészítő-bővítmények és a jelentéskészítés beállításainak megadására szolgáló elem

```
<reporting>
  <plugins>
    <plugin>
  <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-javadoc-
plugin</artifactId>
      <version>3.0.0</version>
    </plugin>
    ...
  </plugins>
</reporting>
```

POM referencia (12)

- `repositories`: a projektben függőségekhez használt távoli tárolókról szolgáltat információkat

```
<repositories>
    <repository>
        <id>maven-restlet</id>
        <name>Restlet repository</name>
        <url>https://maven.restlet.com/</url>
    </repository>
    ...
</repositories>
```

POM referencia (13)

- `scm`: a projekthez használt verziókezelő rendszer eléréséhez tartalmaz információkat

```
<scm>
    <connection>
        scm:git:http://git-wip-us.apache.org/repos/asf/wicket.git
    </connection>
    <developerConnection>
        scm:git:https://git-wip-
us.apache.org/repos/asf/wicket.git
    </developerConnection>
    <url>http://git-wip-us.apache.org/repos/asf/wicket/repo?
p=wicket.git</url>
    <tag>wicket-7.2.0</tag>
</scm>
```

- `url`: a projekt weboldalának címét tartalmazza

```
<url>http://wicket.apache.org</url>
```

- `version`: a Maven koordináták verziószám komponensét tartalmazza

```
<version>3.0</version>
```

Függőségek kezelése

- A Maven a Maven Artifact Resolver könyvtárat használja a függőségek kezeléséhez. https://maven.apache.org/resolver/

Függőségek megadása (1)

```
<dependencies>
    <dependency>
        <groupId>groupId</groupId>
        <artifactId>artifactId</artifactId>
        <version>version</version>
        <classifier>classifier</classifier>
        <type>type</type>
        <optional>false|true</optional>
        <scope>compile|provided|runtime|system|test</scope>
        <systemPath>elérési útvonal</systemPath>
        <exclusions>
            <exclusion>
                <groupId>groupId</groupId>
                <artifactId>artifactId</artifactId>
            </exclusion>
            ...
        </exclusions>
    </dependency>
    ...
</dependencies>
```

Függőségek megadása (2)

- `groupId`, `artifactId`, `version`, `classifier`: a függőség Maven koordinátáit tartalmazzák
- `type`: a csomagolás módját tartalmazza (alapértelmezés: `jar`)
- `optional`: a függőség opcionális-e (alapértelmezés: `false`)

Függőségek megadása (3)

- `scope`: a függőség hatáskörét tartalmazza
  - Lehetővé teszi a különböző összeállítási folyamatokhoz (például fordítás, tesztelés) szükséges classpath meghatározását és a tranzitivitás korlátozását, lehetséges értékei:
    - **`compile`**: minden classpath tartalmazni fogja a függőséget, a függő projekteknek is függősége lesz (ez az alapértelmezés)
    - **`provided`**: azt jelzi, hogy a függőséget a futtató környezet (például JDK) biztosítja, a függőséget csak a fordításhoz használt classpath tartalmazza, nem tranzitív
    - **`runtime`**: azt jelzi, hogy a függőség csak a végrehajtáshoz szükséges (a programtesztek végrehajtásánál is rendelkezésre áll)
    - **`system`**: azt jelzi, hogy a függőséget nem tároló tartalmazza, hanem a felhasználó biztosítja
    - **`test`**: azt jelzi, hogy függőség csak a programtesztek fordításához és végrehajtásához szükséges

Függőségek megadása (4)

- `systemPath`: `system` hatáskörű függőség esetén adható meg csak, de ekkor kötelező
  - A függőség abszolút elérési útvonalát tartalmazza, mint például:

```
<systemPath>${java.home}/lib/jfxrt.jar</systemPath>
```

- `exclusions`: a kizárandó függőségek megadására szolgál

Függőségek megadása (5)

- Példa `system` hatáskörű függőségre: JavaFX használata JDK7 esetén.

```
<dependency>
   <groupId>com.oracle</groupId>
   <artifactId>javafx</artifactId>
   <version>2.2</version>
   <scope>system</scope>
   <systemPath>${java.home}/lib/jfxrt.jar</systemPath>
</dependency>
```

Verziószámok (1)

- *p*`.`*q*`.`*r*`-`*s* alakú verziószámok használata, ahol
  - *p* főverzió (*major version*),
  - *q* alverzió (*minor version*),
  - *r* inkrementális verzió (*incremental version*),
  - *s* build szám (*build number*) vagy minősítő (*qualifier*).
- Minősítők: `alpha`/`a`, `beta`/`b`, `milestone`/`m`, `rc`/`cr`, `snapshot`, ‹*üres sztring*›/`final`/`ga`, `sp`
  - Felsorolás a rendezésnek megfelelő sorrendben (növekvő sorrend).

Verziószámok (2)

- Példa verziószámokra: `1.2`, `4.8.2`, `1.6.0-alpha2`, `1.0-beta9`
- Rendezés értelmezése a verziószámokon (kiterjesztés a szabványos alaktól eltérő formájú verziószámokra is).
  - A rendezés komponensenként történik, számok esetén numerikus rendezés.
  - Példa verziószámok rendezésére:
    - `1.0` < `1.5` < `1.10` < `1.10.1` < `2.0`
    - `1.0-alpha1` < `1.0-beta1` < `1.0-beta2` < `1.0-rc1` < `1.0` < `1.0-sp1`

Verziószámok (3)

- Verziószámok összehasonlításához használjuk a következő parancsot:
  - Linux: `java -jar $M2_HOME/lib/maven-artifact-*.jar`
  - Windows: `java -jar %M2_HOME%\lib\maven-artifact-*.jar`
- A programnak két verziószámot kell megadni parancssor argumentumként.

Verziószámok (4)

- Lásd: org.apache.maven.artifact.versioning.ComparableVersion
  - https://maven.apache.org/ref/3.5.3/maven-artifact/apidocs/org/apache/maven/artifact/versioning/ComparableVersion.html
  - https://github.com/apache/maven/blob/master/maven-artifact/src/main/java/org/apache/maven/artifact/versioning/ComparableVersion.java

Verziótartományok (1)

- Függőségekben verziószám helyett megadható verziótartomány.
  - Az alábbi formák mindegyike támogatott: `(a,b)`, `(a,b]`, `[a,b)`, `[a,b]`
    - A matematikában az intervallumoknál használt jelölés átvétele.
    - Elhagyható az alsó és felső határ, előbbire az alapértelmezés negatív végtelen, utóbbira pozitív végtelen.
  - Megadható tartományok egy vessző karakterekkel elválasztott listája is (a tartományok unióját jelent).
    - Például: `(,1.0),(1.0,)`

Verziótartományok (2)

- Például az alábbi függőség esetén a JUnit bármely olyan verziója elfogadható, melynek *v* verziószámára teljesül, hogy $3.8 \leq v < 4.0$

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>[3.8,4.0)</version>
  <scope>test</scope>
</dependency>
```

Verziótartományok (3)

- Ha egy függőségben a `version` elemben nem tartomány jelenik meg, hanem csak egyetlen verziószám, akkor a Maven azt csupán ajánlásnak tekinti, melyet szükség esetén tetszőleges verzióval helyettesíthet.
  - Adott verzió kényszerítése az alábbi módon lehetséges:

```
<dependency>
   <groupId>junit</groupId>
   <artifactId>junit</artifactId>
   <version>[4.12]</version>
   <scope>test</scope>
</dependency>
```

Tranzitív függőségek (1)

- Ha *B* függősége *A*-nak, *C* pedig *B*-nek, akkor azt mondjuk, hogy *C* tranzitív függősége *A*-nak.
- A Maven automatikusan kezeli a tranzitív függőségeket.
  - Képes a tranzitív függőségek kapcsán felmerülő konfliktusok kezelésére.

Tranzitív függőségek (2)

- Az alábbi táblázat szemlélteti a függőségek tranzitív öröklődését.
  - Az *A* projekt egy a bal oldali oszlopban feltüntetett hatáskörű *B* függőségének egy a felső sorban feltüntetett hatáskörű *C* függősége a sor és oszlop metszéspontjában szereplő hatáskörű függősége egyben *A*-nak is.

	compile	**provided**	**runtime**	**test**
compile	compile	–	runtime	–
provided	provided	–	provided	–
runtime	runtime	–	runtime	–
test	test	–	test	–

Tranzitív függőségek (3)

```
<project xmlns=
"http://maven.apache.org/POM/4.0.0">
 <modelVersion>4.0.0</modelVersion>
 <groupId>my</groupId>
 <artifactId>project-A</artifactId>
 <packaging>jar</packaging>
 <version>1.0</version>
 <dependencies>
  <dependency>
   <groupId>my</groupId>
   <artifactId>project-B</artifactId>
   <version>1.0</version>
   <scope>compile</scope>
  </dependency>
 </dependencies>
</project>
```

```
<project xmlns=
"http://maven.apache.org/POM/4.0.0">
 <modelVersion>4.0.0</modelVersion>
 <groupId>my</groupId>
 <artifactId>project-B</artifactId>
 <packaging>jar</packaging>
 <version>1.0</version>
 <dependencies>
  <dependency>
   <groupId>org.slf4j</groupId>
   <artifactId>slf4j-jdk14</artifactId>
   <version>1.7.25</version>
   <scope>runtime</scope>
  </dependency>
  ...
 </dependencies>
</project>
```

- A fenti példában az `slf4j-jdk14` a `project-A` projektnek is implicit módon `runtime` hatáskörű függősége.

Tranzitív függőségek kizárása (1)

- Tranzitív függőségek kizárásához használjuk az `exclusions` elemet.
  - Konfliktus esetén szükséges lehet, de hasznos felesleges függőségek kizárásához is.

Tranzitív függőségek kizárása (2)

- Példa felesleges függőség kizárására:

```
<dependencies>
   <dependency>
      <groupId>org.seleniumhq.selenium</groupId>
      <artifactId>selenium</artifactId>
      <version>2.0b1</version>
      <exclusions>
         <exclusion>
            <groupId>org.testng</groupId>
            <artifactId>testng</artifactId>
         </exclusion>
      </exclusions>
   </dependency>
<dependencies>
```

  - A selenium `2.0b1` verziójához a JUnit és a TestNG is függőség volt, ha a JUnit-ot használjuk, akkor felesleges a TestNG.
  - Forrás: http://seleniumhq.wordpress.com/2011/01/25/2-0b1-and-maven/

Tranzitív függőségek kizárása (3)

- Függőség összes tranzitív függőségének kizárása:

```
<dependency>
  ...
  <exclusions>
    <exclusion>
      <groupId>*</groupId>
      <artifactId>*</artifactId>
    <exclusion>
  </exclusions>
</dependency>
```

Opcionális függőségek (1)

- A függő projektek számára nem tranzitív függőséget jelentek.
  - Például olyan alternatív függőségek megadása esetén használják őket, amelyek közül csak az egyik szükséges.

Opcionális függőségek (2)

- Példa: *JSR 353: Java API for JSON Processing* https://jcp.org/en/jsr/detail?id=353
  - Nem része a Java SE platformnak, a Java EE 7-nek azonban igen (lásd a `javax.json` csomagot és alcsomagjait).
    - Java SE alkalmazásokhoz a központi tárolóból elérhető `javax.json:javax.json-api:1.1.2` termék szolgáltatja.
  - Használatához az API egy implementációja szükséges, két alternatív implementáció:
    - Az Oracle referencia implementációja (`org.glassfish:javax.json`) https://jsonp.java.net/
    - Genson (`com.owlike:genson`) http://owlike.github.io/genson/

Opcionális függőségek (3)

- Az alábbi projektnek mindkét implementáció opcionális függősége:

```
<project xmlns="http://maven.apache.org/POM/4.0.0">
    <modelVersion>4.0.0</modelVersion>
    <groupId>hu.unideb.inf.maven</groupId>
    <artifactId>json-processor</artifactId>
    <packaging>jar</packaging>
    <version>1.0</version>
    <dependencies>
        <dependency>
            <groupId>org.glassfish</groupId>
            <artifactId>javax.json</artifactId>
            <version>1.1.2</version>
            <optional>true</optional>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>com.owlike</groupId>
            <artifactId>genson</artifactId>
            <version>1.4</version>
            <optional>true</optional>
            <scope>runtime</scope>
        </dependency>
        ...
```

Opcionális függőségek (4)

- (folytatás az előző oldalról):

```
...
        <dependency>
            <groupId>javax.json</groupId>
            <artifactId>javax.json-api</artifactId>
            <version>1.1.2</version>
            <scope>compile</scope>
        </dependency>
    </dependencies>
```

Opcionális függőségek (5)

- Azonban az opcionális függőségek nem öröklődnek tranzitív módon, ha az előbbi projekt jelenik meg függőségként, explicit módon fel kel tüntetni függőségként a két implementáció valamelyikét is!

```
<dependencies>
   <dependency>
      <groupId>hu.unideb.inf.maven</groupId>
      <artifactId>json-processor</artifactId>
      <version>1.0</version>
      <scope>compile</scope>
   </dependency>
   <dependencies>
   <dependency>
      <groupId>com.owlike</groupId>
      <artifactId>genson</artifactId>
      <version>1.4</version>
      <scope>runtime</scope>
   </dependency>
</dependencies>
```

Alapértelmezett verziószámok szolgáltatása függőségekhez (1)

- A felső szintű `dependencyManagement` elem egyetlen `dependencies` elemet tartalmazhat.
  - A benne hivatkozott termékek a felső szintű `dependencies` elemtől eltérően nem lesznek automatikusan a projekt függőségei!
  - A `dependency` elemek itt csupán alapértelmezett verziószámokat szolgáltatnak termékekhez, melyek lehetővé teszik, hogy a projektben és gyermek projektekben verziószám megadása nélkül lehessen függőségként hivatkozni rájuk.
    - Megadható hatáskör is a függőséghez.

Alapértelmezett verziószámok szolgáltatása függőségekhez (2)

- Példa:

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

Alapértelmezett verziószámok szolgáltatása függőségekhez (3)

- Példa (folytatás):
  - Ilyenkor a projektben és a gyerek projektekben a termékre függőségként való hivatkozásnál elhagyható a verziószám és a hatáskör is, mindkettőt a `dependencyManagement` elem szolgáltatja:

```
<dependencies>
   <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
   </dependency>
   ...
</dependencies>
```

Snapshot verziók

- Verziószámok végén a `-SNAPSHOT` karakterlánccal jelezhető, hogy a projekt aktív fejleszés alatt áll.
  - Például: `1.0-SNAPSHOT`
- A termék tárolóba való telepítésekor a `SNAPSHOT` karakterlánc kifejtése az aktuális rendszeridővel (UTC idő használata).
  - Például közép-európai idő szerint 2018. január 3-án 21:58:34-kor a fenti verziószám esetén a helyettesítés eredménye az `1.0-20180103.205834-`*`N`* verziószám.
  - *N* értéke 1-ről indul, minden új *snapshot* verzió esetén eggyel nő.

Snapshot és release termékek (1)

- A *snapshot* verziószámokkal ellátott termékeket *snapshot* termékeknek nevezik.
  - A fejlesztés adott pillanatbeli állapotát tükrözik.
  - Csak fejlesztés közben használatosak.
  - Az újabb és újabb *snapshot* verziók hamar elavulttá teszik őket.
- A többi terméket *release* terméknek nevezik.
  - Ezek stabilnak tekinthető termékek.
  - Hosszabb időn keresztül használatosak.

Snapshot és release termékek (2)

- Általában külön tárolókat használnak a *snapshot* és a *release* termékek publikálásához.
- De akár ugyanaz a tároló szolgáltathat *snapshot* és *release* termékeket is.
  - Lásd a `repositories/repository` és a `pluginRepositories/pluginRepository` elemekben rendelkezésre álló `releases` és `snapshots` elemeket.

Termékek feltöltése távoli tárolóba (1)

- A `deploy` életciklus fázisban kerülnek feltöltésre a termékek a beállításokban adott távoli tárolóba.
- A POM `distributionManagement` elemében megadható `repository` és `snapshotRepository` elemek szolgáltatják a távoli tároló eléréséhez a beállításokat.
  - A `repository` elem a *release* termékek tárolóját, a `snapshotRepository` elem pedig értelemszerűen a *snapshot* termékek tárolóját adja meg.

Termékek feltöltése távoli tárolóba (2)

- A `repository` és `snapshotRepository` elemek használata:

```
<distributionManagement>
  <repository>
    <id>azonosító</id>
    <name>név</name>
    <url>URI</url>
    <layout>default|legacy</layout>
    <uniqueVersion>true|false</uniqueVersion>
  </repository>
  <snapshotRepository>
    <id>azonosító</id>
    <name>név</name>
    <url>URI</url>
    <layout>default|legacy</layout>
    <uniqueVersion>true|false</uniqueVersion>
  </snapshotRepository>
</distributionManagement>
```

Termékek feltöltése távoli tárolóba (3)

- A `repository` és `snapshotRepository` elemekben megadható elemek:
  - `id`: a tároló egyedi azonosítója
  - `name`: a tároló ember számára olvasható neve
  - `url`: URI a tároló eléréséhez
  - `layout`: a tároló kialakítása
    - `default`: a Maven 2.*x* és 3.*x* számú verziói által használt kialakítás (ez az alapértelmezés)
    - `legacy`: a Maven 1.*x* számú verzói által használt kialakítás
  - `uniqueVersion`: *snapshot* verzió esetén egy egyedi verziószám kerüljön-e előállításra az aktuális rendszeridő felhasználásával (alapértelmezés: `true`)

Tároló beállítások (1)

- Függőségeket szolgáltató tárolók eléréséhez a felső szintű `repositories` elemben adhatóak meg beállítások:

```
<repositories>
  <repository>
    <id>azonosító</id>
    <name>név</name>
    <url>URI</url>
    <layout>default|legacy</layout>
```

Tároló beállítások (2)

(folytatás az előző oldalról)

```
      <releases>
         <checksumPolicy>fail|ignore|warn</checksumPolicy>
         <enabled>false|true</enabled>
         <updatePolicy>always|daily|interval:N|
            never</updatePolicy>
      </releases>
      <snapshots>
         <checksumPolicy>fail|ignore|warn</checksumPolicy>
         <enabled>false|true</enabled>
         <updatePolicy>always|daily|interval:N|
            never</updatePolicy>
      </snapshots>
   </repository>
   ...
</repositories>
```

Tároló beállítások (3)

- A `repository` elemben megadható elemek:
  - `id`: a tároló egyedi azonosítója
  - `name`: a tároló ember számára olvasható neve
  - `url`: URI a tároló eléréséhez
  - `layout`: a tároló kialakítása
    - `default`: a Maven 2.*x* és 3.*x* számú verziói által használt kialakítás (ez az alapértelmezés)
    - `legacy`: a Maven 1.*x* számú verzói által használt kialakítás
  - `releases`: *release* termékek letöltésére vonatkozó előírások
  - `snapshots`: *snapshot* termékek letöltésére vonatkozó előírások

Tároló beállítások (4)

- A `releases` és `snapshots` elemben megadható elemek:
  - `checksumPolicy`: hogyan történjen az ellenőrző összeg hibák kezelése (a tárolók minden termékhez nyilvántartanak egy MD5 és/vagy egy SHA-1 ellenőrző összeget)
    - `fail`: hiba
    - `ignore`: figyelmen kívül hagyás
    - `warn`: figyelmeztetés (ez az alapértelmezés)
  - `enabled`: engedélyezett-e a megfelelő típusú (*snapshot* vagy *release*) termékek letöltése a tárolóból (alapértelmezés: `true`)

Tároló beállítások (5)

- A `releases` és `snapshots` elemben megadható elemek (folytatás):
  - `updatePolicy`: milyen gyakran történjen a tárolóból frissítés
    - `always`: a Maven minden egyes futtatásakor
    - `daily`: naponta egyszer (ez az alapértelmezés)
    - `interval:`*N* (ahol *N* egész szám): *N* percenként
    - `never`: soha

Tároló beállítások (6)

- Bővítményeket szolgáltató tárolókhoz a `pluginRepositories` felső szintű elemet kell használni:

```
<pluginRepositories>
  <pluginRepository>
    ...
  </pluginRepository>
  ...
</pluginRepositories>
```

  - Az elemben megadható `pluginRepository` elemek tartalma megegyezik a `repository` elemekével.

Függés snapshot termékektől (1)

- Beállítható, hogy *snapshot* terméktől való függés esetén automatikusan a távoli tárolóban rendelkezésre álló legkésőbbi *snapshot* verzió kerüljön felhasználásra.
  - A beállítás a `repository` és a `pluginRepository` elemekben rendelkezésre álló `snapshots` elemben történik.

```
<repository>
   ...
   <snapshots>
      <enabled>true</enabled>
      <updatePolicy>frissítési stratégia</updatePolicy>
   </snapshots>
   ...
</repository>
```

- Ne feledjük, hogy csak fejlesztés közben szabad *snapshot* termékektől függeni!

Függés snapshot termékektől (2)

- Ha olyan *snapshot* termékre történik hivatkozás függőségként, mely nem áll rendelkezésre a lokális tárolóban, akkor a távoli tárolóból mindig automatikusan a legkésőbbi *snapshot* verzió kerül letöltésre.
- Ha egy termék legalább egy *snapshot* verziója a lokális tárolóban van, akkor megállapításra kerül, hogy a távoli tároló tartalmaz-e későbbi *snapshot* verziót.
  - Ha igen, akkor a legkésőbbi *snapshot* verzió letöltése a távoli tárolóból a lokális tárolóba.
- Az `updatePolicy` elemmel szabályozható, hogy mikor forduljon a Maven a távoli tárolóhoz újabb *snapshot* verziókért.

Függés snapshot termékektől (3)

- Az `updatePolicy` elem lehetséges értékeinek jelentése:
  - **`always`**: a Maven minden futtatáskor ellenőrzi a távoli tárolót
  - **`daily`**: a Maven minden nap az első futtatáskor ellenőrzi a távoli tárolót
  - **`interval`**:*N* (ahol *N* egész szám): a Maven akkor ellenőrzi a tárolót, ha *N* perc telt el a legutóbbi ellenőrzés óta
  - **`never`**: nincs ellenőrzés

Függés release termékektől (1)

- Release termékek kezeléséhez a *snapshot* termékeknél tárgyalt módon adható meg az `updatePolicy` elem:

```
<repository>
   ...
   <releases>
      <enabled>true</enabled>
      <updatePolicy>frissítési stratégia</updatePolicy>
   </releases>
   ...
</repository>
```

Függés release termékektől (2)

- Az `updatePolicy` beállítás *release* termékek esetére való értelmezéséhez az alábbiakat kell megjegyezni:
  - Minden *release* termék csak egyszer kerül letöltésre a távoli tárolóból a lokális tárolóba!
    - Egy *release* termék akkor sem lesz újra letöltve, ha a távoli tárolóban felülírásra került.
- A `never`-től különböző `updatePolicy` beállítás például verziótartományok használata esetén eredményezheti a tárolóból egy *release* termék későbbi verzióinak letöltését.

Állomány kézi telepítése a lokális tárolóba

- Az alábbi parancs végrehajtásával lehetséges:

```
mvn install:install-file \
   -Dfile=path \
   -DgroupId=groupId \
   -DartifactId=artifactId \
   -Dversion=version \
   -Dpackaging=packaging \
   -DgeneratePom=true
```

  - Például olyan JAR állományok esetén használjuk, amelyek nem állnak rendelkezésre egyetlen elérhető távoli tárolóban sem.

Öröklés (1)

- Olyan projekt lehet szülő, melynél a csomagolás módja `pom`:

```
<project xmlns=
      "http://maven.apache.org/POM/4.0.0">
  <modelVersion>4.0.0</modelVersion>
  <groupId>hu.unideb.inf.maven</groupId>
  <artifactId>parent</artifactId>
  <packaging>pom</packaging>
  <version>1.0</version>
  ...
</project>
```

Öröklés (2)

- Szülő projekt megadása gyerek projektben:

```
<project xmlns=
   "http://maven.apache.org/POM/4.0.0">
   <modelVersion>4.0.0</modelVersion>
   <parent>
      <groupId>hu.unideb.inf.maven</groupId>
      <artifactId>parent</artifactId>
      <version>1.0</version>
   </parent>
   <artifactId>child</artifactId>
   <packaging>jar</packaging>
   ...
</project>
```

Öröklés (3)

- A gyerek projekt a szülő projekthez tartozó POM-ból automatikusan örököl bizonyos beállításokat az effektív POM előállítása során.
  - Bizonyos elemek csak akkor lesznek átvéve a szülő POM-ból, ha azok a gyerek POM-ban nincsenek explicit módon megadva.
    - Így történik például a `ciManagement`, `contributors`, `developers`, `groupId`, `issueManagement`, `licenses`, `mailingLists`, `organization`, `url` és `version` elemek kezelése.
  - Bizonyos elemek esetén a tartalom kombinálása történik, ha a szülő és a gyerek POM-ban is szerepelnek.
    - Így történik például a `plugins`, `scm` és `repositories` elemek kezelése.

Többmodulos projektek (1)

- A többmodulos projektek, más néven aggregátor projektek moduloknak nevezett projektekből állnak.
  - A többmodulos projektek esetén a csomagolás módja kötelezően `pom`.
    - A modulok csomagolása már tetszőleges lehet, ezek is lehetnek akár többmodulos projektek.
  - A modulok felsorolása a `modules` elemben történik.
    - Az egyes `module` elemek a modulok könyvtárainak relatív elérési útvonalát tartalmazzák.
      - Az aggregátor projekt általában alkönyvtárként tartalmazza a modulokat.

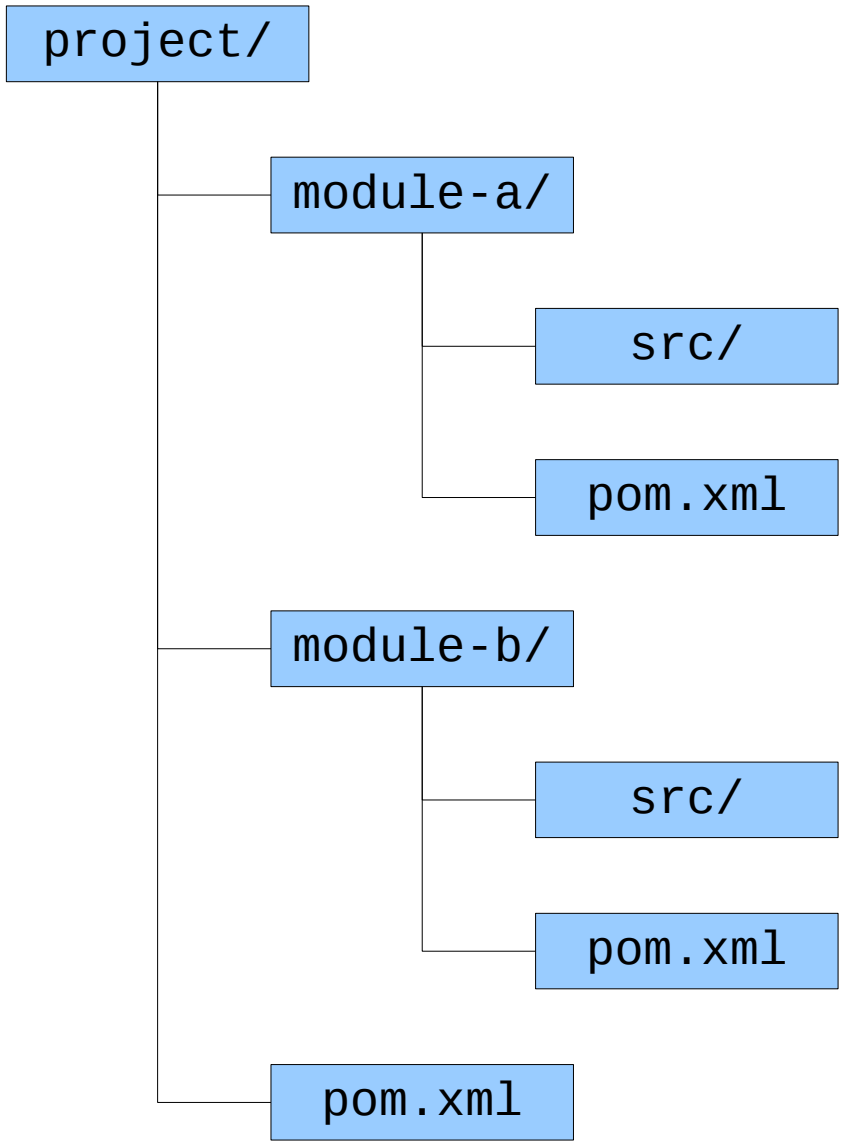
Többmodulos projektek (2)

- Ha az aggregátor projekt főkönyvtárában kezdeményezzük életciklus fázisok vagy bővítmény-célok végrehajtását, akkor a végrehajtás minden egyes modulban megtörténik.
  - A Maven automatikusan meghatározza a modulok sorrendjét (a modulok függhetnek egymástól).

Többmodulos projektek (3)

- Az aggregátor projekt és a modulok szülő-gyerek kapcsolatban lehetnek egymással, ilyenkor öröklés is történik.
  - Ez azonban nem kötelező!
  - Egy többmodulos projekt tipikusan alapbeállításokat szolgáltat a POM-ban a modulok számára.

Többmodulos projekt szervezése



Az aggregátor projekt POM-ja:

```
<project xmlns=
  "http://maven.apache.org/POM/4.0.0">
  <modelVersion>4.0.0</modelVersion>
  <groupId>my</groupId>
  <artifactId>project</artifactId>
  <packaging>pom</packaging>
  <version>1.0</version>
  <modules>
    <module>module-a</module>
    <module>module-b</module>
  </modules>
  ...
</project>
```

Többmodulos projekt létrehozása: m2eclipse

1. Hozzuk létre az aggregátor projektet: *File* → *New* → *Project…* → *Maven* → *Maven Project*
   - Legyen bekapcsolva a *Create a simple project (skip archetype selection)* checkbox!
   - A csomagolás módjának `pom`-ot válasszunk a *Packaging* mezőben.
   - Törölhető az aggregátor projektben felesleges, de automatikusan létrehozott `src/` alkönyvtár.
2. Egy modul létrehozásához válasszuk ezt: *File* → *New* → *Project…* → *Maven* → *Maven Module*.
   - A *Parent Project* mezőben adható meg/választható ki, hogy melyik projekt modulja legyen.
   - A modul automatikusan gyermeke is lesz az aggregátor projektnek.

Többmodulos projekt létrehozása: NetBeans

1. Hozzuk létre az aggregátor projektet: *File* → *New Project*
   - Válasszuk a *POM Project* opciót a *Maven* kategóriából.
2. Egy modul létrehozásához a teendők:
   - A *Projects* panelen kattintsunk az egér jobb gombjával az aggregátor projekt neve alatt a *Modules*-ra és válasszuk a *Create New Module…* pontot.
   - A modul automatikusan gyermeke is lesz az aggregátor projektnek.

Profilok (1)

- A profilok a POM olyan opcionális beállításokat tartalmazó részei, amelyek csak aktiválás esetén kerülnek felhasználásra.
  - Lehetővé teszik a POM futásidejű módosítását.
  - Hasznosak például a projekt eltérő környezetekben történő használata esetén.
    - Különböző környezetekhez szolgáltathatnak testreszabott beállításokat.

Profilok (2)

- Megadásuk az alábbi módon történhet a POM-ban:

```
<profiles>
   <profile>
      <id>azonosító</id>
      <activation>aktiválási feltétel(ek)</activation>
      profil-specifikus beállítások
   </profile>
   <profile>
      <id>azonosító</id>
      <activation>aktiválási feltétel(ek)</activation>
      profil-specifikus beállítások
   </profile>
   ...
</profiles>
```

Profilok (3)

- Profilokban beállítások megadásához használható elemek:
  - `build`
  - `dependencies`
  - `dependencyManagement`
  - `distributionManagement`
  - `modules`
  - `pluginRepositories`
  - `properties`
  - `reporting`
  - `repositories`

Profilok (4)

- A Maven Help Plugin szolgáltat információkat a profilokról.
  - Az `mvn help:all-profiles` parancs az összes rendelkezésre álló profilt, az `mvn help:active-profiles` parancs pedig az összes aktív profilt jeleníti meg.

Profil aktiválás

- Profil aktiválása történhet a felhasználó explicit kérésére és meghatározott feltételek teljesülése esetén automatikusan.
  - Automatikus aktiválás történhet az alábbiak alapján:
    - Rendszertulajdonságok és környezeti változók értéke
    - Operációs rendszer
    - JDK verziószám
    - Állományok létezése és hiánya
  - Az automatikus aktiválás feltételeinek megadására szolgálnak az `activation` elemben a `file`, `jdk`, `os` és `property` elemek.
    - Ha közülük több is megjelenik egy `activation` elemben, akkor bármelyik feltételeinek teljesülése aktiválást eredményez (logikai vagy kapcsolat).

Profil aktiválás: explicit

- Profilok aktiválásához használjuk a `-P` vagy `--activate-profiles` parancssori opciót, amely után profilok azonosítóit kell megadni (egynél több profil esetén `','` karakterekkel elválasztva).
  - Ha egy profil azonosítója elé a `'!'` vagy a `'-'` karaktert írjuk, akkor az a profil kikapcsolását jelenti.
    - Figyelem: a `'!'` karakternek a Bash parancsértelmezőben speciális jelentése van, ezért megfelelően le kell védeni!
  - Példa (a `profile-1` profil aktiválása és a `profile-2` profil kikapcsolása):

```
mvn help:active-profiles -P profile-1,-profile-2
```

Profil aktiválás: alapértelmezetten aktív profil

- Az alábbi módon megadott profil alapértelmezetten aktív:

```
<profile>
   <id>default</id>
   <activation>
      <activeByDefault>true</activeByDefault>
   </activation>
   ...
</profile>
```

  - Explicit aktiválás és nem alapértelmezetten aktív profil(ok) automatikus aktiválása esetén az ilyen profilok ki lesznek kapcsolva!
    - Kivéve akkor, ha explicit módon kérjük az aktiválásukat.

Profil aktiválás: rendszertulajdonságok

- Aktiválás akkor, ha a `debug` rendszertulajdonság be van állítva, értéke tetszőleges:

```
<activation>
   <property>
      <name>debug</name>
   </property>
</activation>
```

- Aktiválás akkor, ha a `debug` rendszertulajdonság nincs beállítva:

```
<activation>
   <property>
      <name>!debug</name>
   </property>
</activation>
```

- Aktiválás akkor, ha az `environment.type` rendszertulajdonság értéke `production`:

```
<activation>
   <property>
      <name>environment.type</name>
      <value>production</value>
   </property>
</activation>
```

Profil aktiválás: környezeti változók

- Aktiválás akkor, ha a `DEBUG` környezeti változó be van állítva, értéke tetszőleges:

```
<activation>
   <property>
      <name>env.DEBUG</name>
   </property>
</activation>
```

- Aktiválás akkor, ha az `ENV` környezeti változó értéke `test`:

```
<activation>
   <property>
      <name>env.ENV</name>
      <value>test</value>
   </property>
</activation>
```

- Aktiválás akkor, ha a `DEBUG` környezeti változó nincs beállítva:

```
<activation>
<property>
   <name>!env.DEBUG</name>
</property>
</activation>
```

- Aktiválás akkor, ha az `ENV` környezeti változó értéke nem `test`:

```
<activation>
   <property>
      <name>env.ENV</name>
      <value>!test</value>
   </property>
</activation>
```

Profil aktiválás: operációs rendszer specifikus (1)

- Használjuk az os elemet:

```
<activation>
   <os>
      <arch>...</arch>
      <name>...</name>
      <family>...</family>
      <version>...</version>
   </os>
</activation>
```

  - `arch`: operációs rendszer architektúra (például `amd64`, `x86`, …)
  - `name`: operációs rendszer neve (például `linux`, `windows xp`, …)
  - `family`: operációs rendszer-család (`mac`, `unix`, `windows`)
  - `version`: az operációs rendszer verziószáma (pontos verziószám, verziótartomány nem használható)
- Negáció kifejezéséhez az `arch`, `name`, `family` és `version` elemekben is használható a `'!'` karakter (például `<family>!windows</family>`).

Profil aktiválás: operációs rendszer specifikus (2)

- Példa:
  - ```
    <activation>
      <os>
        <name>linux</name>
        <arch>amd64</arch>
      </os>
    </activation>
    ```
  - ```
    <activation>
      <os>
        <name>windows xp</name>
        <version>5.1</version>
      </os>
    </activation>
    ```

Profil aktiválás: JDK

- A `jdk` elemben megadható verziószám kezdőszelet vagy verziótartomány (aktiválás akkor, ha a JDK verziószáma ennek megfelelő).
  - Verziószám kezdőszelet esetén negáció, ha az első karakter `'!'`.
- Példa:

```
<profile>
  <id>jdk6</id>
  <activation>
    <jdk>1.6</jdk>
  </activation>
  ...
</profile>
```

- Példa:

```
<profile>
  <id>pre-jdk6</id>
  <activation>
    <jdk>[,1.6)</jdk>
  </activation>
  ...
</profile>
```

Profil aktiválás: állományok (1)

- A `file` elem segítségével adott állományok létezéséhez és/vagy hiányához köthető az aktiválás.

```
<activation>
  <file>
    <exists>...</exists>
    <missing>...</missing>
  </file>
</activation>
```

  - Az `exists` és `missing` elemekben egy állomány elérési útvonalát kell megadni.

Profil aktiválás: állományok (2)

- Példa:
  - ```
    <activation>
       <file>
          <exists>${user.home}/.myTool/license.txt</exists>
       </file>
    </activation>
    ```
  - ```
    <activation>
       <file>
          <missing>${basedir}/.git</missing>
       </file>
    </activation>
    ```

Bővítmények használata a POM-ban (1)

```
<build>
   <plugins>
      <plugin>
         <groupId>groupId</groupId>
         <artifactId>artifactId</artifactId>
         <version>version</version>
         <configuration>beállítások</configuration>
         <dependencies>függőségek</dependencies>
         <executions>cél végrehajtások</executions>
         <extensions>false|true</extensions>
         <inherited>false|true</inherited>
      </plugin>
      ...
   </plugins>
```

Bővítmények használata a POM-ban (2)

- A `plugin` elem:
  - `groupId`, `artifactId`, `version`: a bővítmény Maven koordinátái
  - `configuration`: konfigurációs paramétereket tartalmaz a célok végrehajtásához
    - Az XML séma a tartalomra nem tesz semmilyen megszorítást.
  - `dependencies`: a bővítményhez szükséges függőségeket tartalmazza
    - A függőségek megadása a korábban tárgyalt formában történik.

Bővítmények használata a POM-ban (3)

- A `plugin` elem (folytatás):
  - `executions`: lehetővé teszi bővítmény-célok végrehajtásának hozzárendelését életciklus fázisokhoz, így az összeállítási folyamat testreszabását (részletesen lásd később)
  - `extensions`: (alapértelmezés: `false`)
  - `inherited`: azt jelzi, hogy öröklés során át kell-e venni a bővítmény beállításait (alapértelmezés: `true`)

Bővítmények használata a POM-ban (4)

- Az `executions` elem:

```
<plugin>
   <groupId>...</groupId>
   <artifactId>...</artifactId>
   <version>...</version>
   <executions>
      <execution>
         <id>azonosító</id>
         <goals>
            <goal>cél1</goal>
            ...
            <goal>céln</goal>
         </goals>
         <phase>életciklus fázis</phase>
         <inherited>false|true</inherited>
         <configuration>beállítások</configuration>
      </execution>
      ...
   </executions>
   ...
</plugin>
```

Bővítmények használata a POM-ban (5)

- Az `execution` elem:
  - **`id`**: azonosító az elem számára
  - **`goals/goal`**: a végrehajtandó bővítmény-célok neveit tartalmazzák
  - **`phase`**: az életciklus fázis neve, amelyhez hozzá kell rendelni a célok végrehajtását
  - **`inherited`**: azt jelzi, hogy öröklés során át kell-e venni az `execution` elemet (alapértelmezés: `true`)
  - **`configuration`**: konfigurációs paramétereket tartalmaz a `goal` elemekben felsorolt célok végrehajtásához
    - Általa finomítható a `plugin/configuration` elemben megadott konfiguráció.

Bővítmények használata a POM-ban (6)

- Példa az `executions` elem használatára:

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-antrun-plugin</artifactId>
            <version>1.8</version>
            <executions>
                <execution>
                    <id>confirm-clean</id>
                    <phase>pre-clean</phase>
                    <goals>
                        <goal>run</goal>
                    </goals>
                    <configuration>
                        <target>
                            <echo>Cleaning up...</echo>
                        </target>
                    </configuration>
                </execution>
        </executions>
    </plugin>
    ...
```

Bővítmények használata a POM-ban (7)

- Egy bővítmény-célhoz tartozhat egy alapértelmezett életciklus fázis, ekkor az `execution` elemben nem szükséges megadni a `phase` elemet.
  - Ha nincs alapértelmezett életciklus fázis, akkor a `phase` elem hiányában a bővítmény-cél nem kerül végrehajtásra!

Bővítmények használata a POM-ban (8)

- Például a `maven-enforcer-plugin` bővítmény `enforce` célja alapértelmezésben a `validate` életciklus fázishoz van hozzákötve

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-enforcer-plugin</artifactId>
            <version>3.0.0-M1</version>
            <executions>
                <execution>
                    <id>enforce-java-version</id>
                    <goals><goal>enforce</goal></goals>
                    <configuration>
                        <rules>
                            <requireJavaVersion>
                                <version>1.9</version>
                            </requireJavaVersion>
                        </rules>
                    </configuration>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

Webhely készítése (1)

- A `reporting` elemben kell megadni azokat a jelentéskészítő-bővítményeket, melyek által előállított jelentések automatikusan a webhely részei lesznek:

```
<reporting>
   <outputDirectory>elérési útvonal</outputDirectory>
   <plugins>
      jelentéskészítő-bővítmények felsorolása (plugin elemek)
   </plugins>
   <excludeDefaults>false|true</excludeDefaults>
</reporting>
```

  - `outputDirectory`: a kimeneti könyvtár elérési útvonala (alapértelmezés: `${project.build.directory}/site`).
  - `excludeDefaults`: az alapértelmezésben előállításra kerülő jelentések kizárása (alapértelmezés: `false`).

Webhely készítése (2)

- A Maven 3 az alábbi módon is lehetővé teszi a jelentéskészítő-bővítmények megadását:

```
<build>
   <plugins>
      ...
      <plugin>
         <groupId>org.apache.maven.plugins</groupId>
         <artifactId>maven-site-plugin</artifactId>
         <version>3.7</version>
         <configuration>
            <reportPlugins>
               jelentéskészítő-bővítmények felsorolása (plugin
               elemek)
            </reportPlugins>
         </configuration>
      </plugin>
      ...
   </plugins>
</build>
```

Webhely készítése (3)

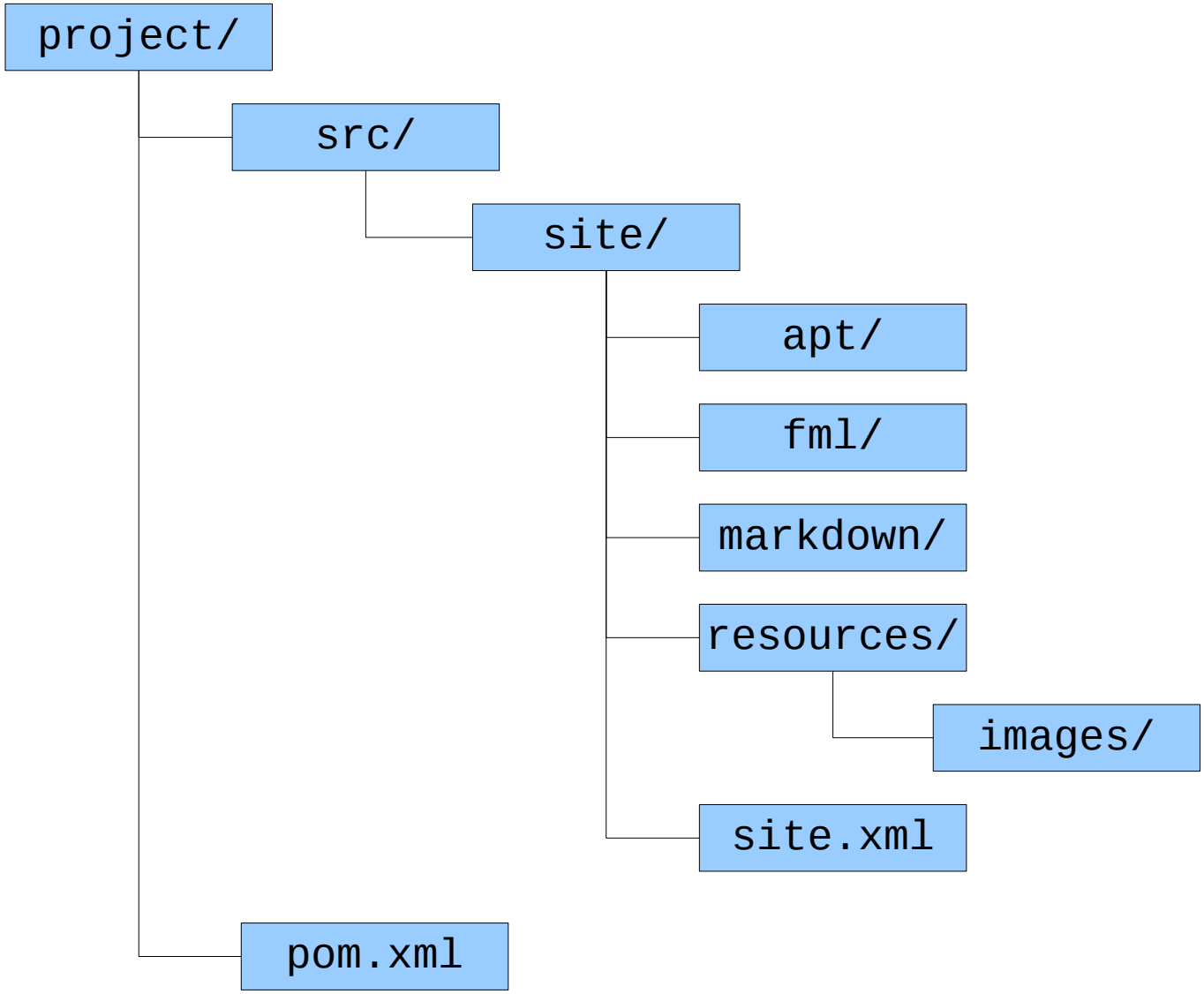
- Többmodulos projekt esetén az `mvn site` parancs helyett a webhely előállításához az `mvn site site:stage` parancsot kell végrehajtani.
  - Ilyenkor az eredmény a `${basedir}/target/staging/` könyvtárban jön létre!
  - A működéshez az alábbiakat is meg kell adni a POM-ban:

```
<distributionManagement>
   <site>
      <id>website</id>
      <url>file:///tmp/fake.com/</url>
   </site>
</distributionManagement>
```

Webhely testreszabása (1)

- A webhely testreszabásához a `${basedir}/src/site/` könyvtárban kell elhelyezni a megfelelő állományokat.
  - A `site.xml` (site descriptor) állományban változtatható meg a webhely megjelenésének felépítése.
    - A formátumhoz az alábbi XML sémát kell használni: http://maven.apache.org/xsd/decoration-1.7.0.xsd
  - A könyvtár alatt speciális alkönyvtárak helyezhetők el, amelyek a webhelyhez szolgáltatnak tartalmat.
    - Speciális formátumok használata, amelyekből automatikusan HTML oldalak jönnek létre.

Webhely testreszabása (2)



Webhely testreszabása (3)

- Formátumok: http://maven.apache.org/doxia/references/
  - APT (Almost Plain Text) http://maven.apache.org/doxia/references/apt-format.html
  - FML (FAQ Markup Language) http://maven.apache.org/doxia/references/fml-format.html
  - Markdown http://daringfireball.net/projects/markdown/