Apache Maven

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

Utolsó módosítás: 2023. február 17.

Apache Maven

- Egy projektkezelő eszköz (*software project management and comprehension tool*) a következő célkitűzésekkel:
  - Az összeállítási folyamat megkönnyítése
  - Egységes rendszer biztosítása az összeállításhoz
  - Minőségi projekt információk szolgáltatása
  - Irányelvek és legjobb gyakorlatok szolgáltatása a fejlesztéshez
- Lásd: https://maven.apache.org/what-is-maven.html

Lehetőségek

- A főbb lehetőségek közé tartoznak a következők:
  - Egységes rendszer projektek összeállításához
  - Függőségkezelés
  - Disztribúció közzététel
  - Jelentéskészítés és webhely létrehozás
- Lásd: *Feature Summary* https://maven.apache.org/maven-features.html

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++:
    - Native Maven Plugin
      https://www.mojohaus.org/maven-native/native-maven-plugin/
      https://github.com/mojohaus/maven-native
  - Kotlin:
    - kotlin-maven-plugin https://kotlinlang.org/docs/maven.html
  - Scala:
    - scala-maven-plugin http://davidb.github.io/scala-maven-plugin/
      https://github.com/davidB/scala-maven-plugin

Történet

- A Maven eredeti szerzője Jason van Zyl: https://github.com/jvanzyl
  - Lásd: *History of Maven* https://maven.apache.org/background/history-of-maven.html

Fejlesztés (1)

- Programozási nyelv: Java
- Szabad és nyílt forrású: az *Apache License 2.0* hatálya alatt terjesztik
- A jelenleg aktuális stabil verzió a 3.9.0 számú (kiadás dátuma: 2023. január 31.)
  - Lásd: *Maven Releases History* https://maven.apache.org/docs/history.html
- Tároló: https://github.com/apache/maven

Fejlesztés (2)

- A következő fő verzió (Maven 4.0) jelenleg fejlesztés alatt áll.
- Webhely: https://maven.apache.org/ref/4-LATEST/
- További információk:
  - Maarten Mulders, Martin Kanters. *What's New in Maven 4*. November 30, 2020. https://maarten.mulders.it/2020/11/whats-new-in-maven-4/
  - Hervé Boutemy. *From Maven 3 to Maven 5*. December 21, 2021. https://www.javaadvent.com/2021/12/from-maven-3-to-maven-5.html

Ipari felhasználások

- Az Apache Maven-nel összeállított projektek:
  - *Apache Log4j 2* https://github.com/apache/logging-log4j2
  - *Apache Spark* https://github.com/apache/spark
  - *Eclipse GlassFish* https://github.com/eclipse-ee4j/glassfish
  - *Eclipse Jetty* https://github.com/eclipse/jetty.project
  - *Jenkins* https://github.com/jenkinsci/jenkins
  - *Gson* https://github.com/google/gson
  - *Guava* https://github.com/google/guava
  - *WildFly Application Server* https://github.com/wildfly/wildfly
  - …

Telepítés

- Az Apache Maven használatához JDK szükséges, JRE nem elegendő!
  - A JDK 7-es vagy későbbi kiadása szükséges.
  - Fontos, hogy megfelelően be legyen állítva a `JAVA_HOME` környezeti változó is!
- Letöltés: https://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-3.8.7` könyvtár alá bontottuk ki a szoftvert tartalmazó állományt, akkor az alábbi környezeti változó beállítás szükséges:
  - `export PATH=/opt/apache-maven-3.9.0/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: https://sdkman.io/install

Telepítés (Windows)

- Ha például a `C:\Program Files\apache-maven-3.8.7` 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-3.9.0\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 3.9.0 (9b58d2bad23a66be161c4664ef21ce219c2c8584)
Maven home: /home/jeszy/.sdkman/candidates/maven/current
Java version: 19.0.2, vendor: Oracle Corporation,
    runtime: /home/jeszy/.sdkman/candidates/java/19.0.2-open
Default locale: hu_HU, platform encoding: UTF-8
OS name: "linux", version: "5.1.5-050105-generic", arch: "amd64", family: "unix"
```

IDE integráció (1)

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

IDE integráció (2)

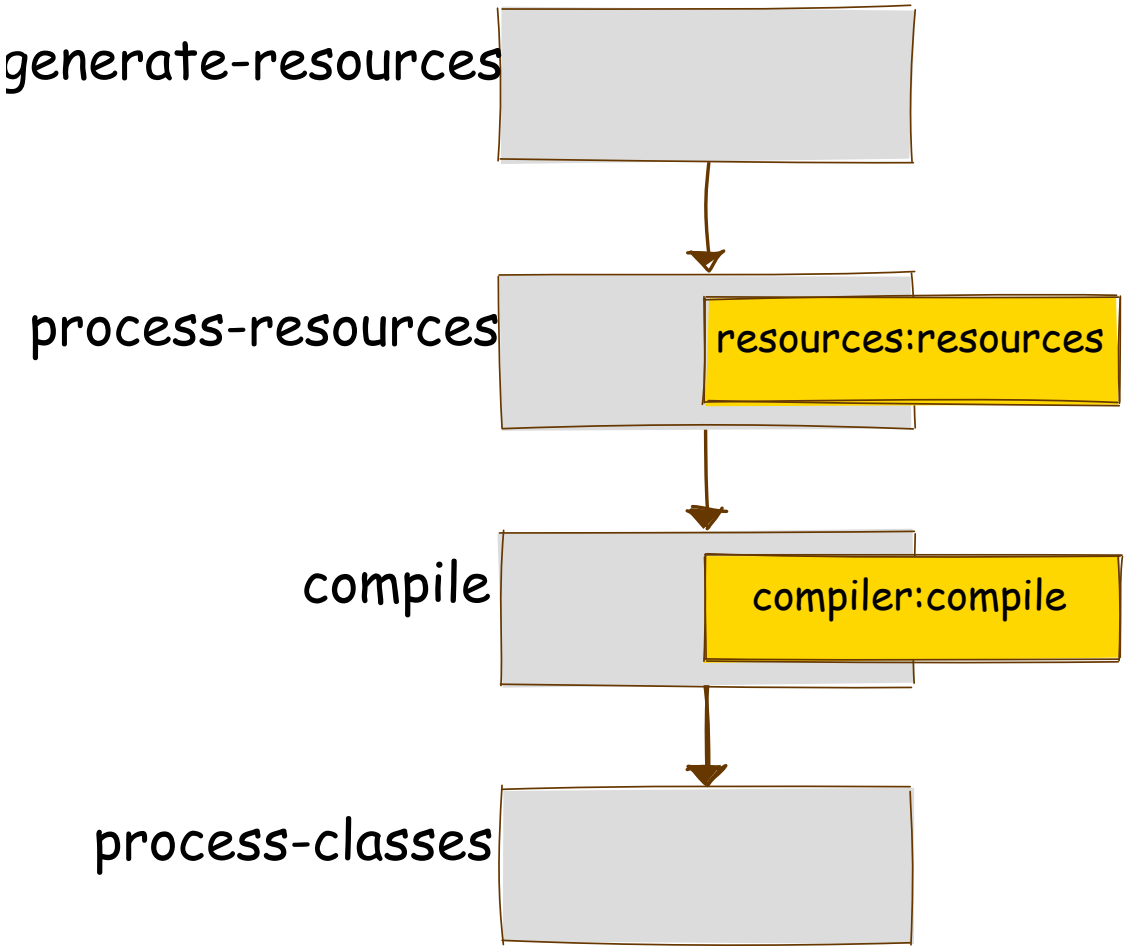
- **Visual Studio Code**: a *Maven for Java* kiterjesztés nyújt Maven támogatást.
  - Lásd:
    - *Maven for Java* https://marketplace.visualstudio.com/items?itemName=vscjava.vscode-maven

További információk

- Hivatalos dokumentáció: https://maven.apache.org/guides/
- Levelezési listák: https://maven.apache.org/mailing-lists.html

A Maven működése

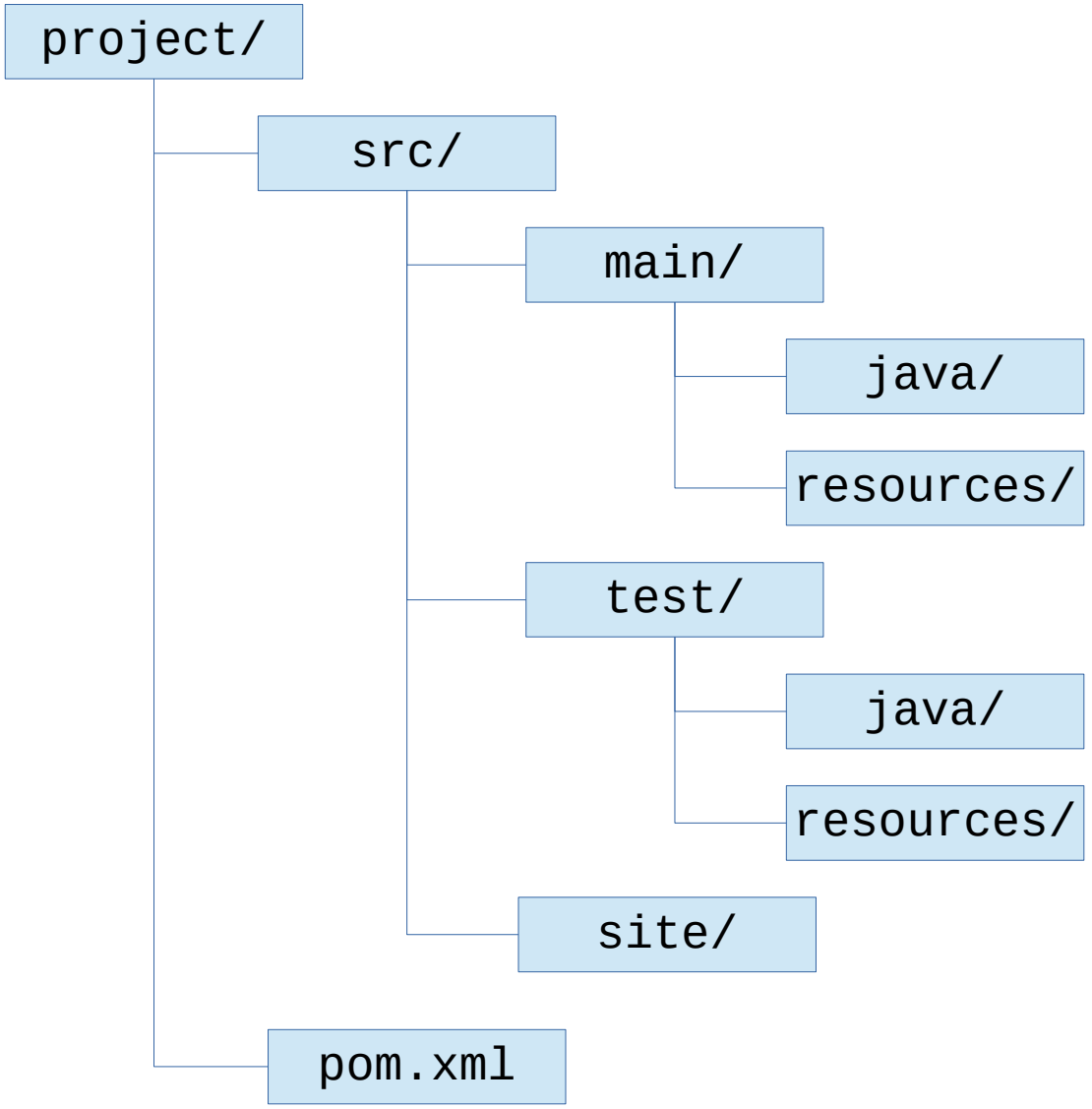
- A `default` életciklus egy kis részlete:



Projekt könyvtárszerkezet (1)

- Szabványos könyvtárszerkezet meghatározása a projektek számára.
  - Lásd: *Introduction to the Standard Directory Layout* https://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.
  - Példa:
    ```
    mvn org.codehaus.mojo:versions-maven-plugin:2.14.2:help
    ```

settings.xml (1)

- Projekt-független beállításokat tartalmazó konfigurációs állomány.
  - Az összes felhasználó számára globális beállításokat szolgáltat a `${maven.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.

settings.xml (2)

- Referencia: https://maven.apache.org/ref/current/maven-settings/settings.html
- XML séma: https://maven.apache.org/xsd/settings-1.2.0.xsd

settings.xml (3)

- 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)

- Egy projekt által előállított állomány, mely annak végső termékének tekinthető.
  - Egy projektben általában egy termék készül (például egy `jar` csomagolású projektben egyetlen JAR állomány).
    - A `classifier` POM elem szolgál az egy projekt által létrehozott termékek megkülönböztetésére.
- Tárolókban kerülnek közzétételre, mely lehetővé teszi a más projektekhez függőségként történő felhasználásukat.

Projekt objektum modell (POM) (1)

- Egy projekt deklaratív leírását tartalmazó XML dokumentum (`pom.xml`).
  - Metaadatokat és konfigurációs beállításokat tartalmaz.
- Egy életciklus fázis vagy bővítmény-cél végrehajtásakor a Maven alapértelmezésben az aktuális könyvtárban keresi a POM-ot.
  - A POM elérési útvonala `-f` vagy `--file` opcióval adható meg.
- A projektek között szülő-gyerek kapcsolatok definiálhatóak.
  - A gyerek projekt megörökli a szülőhöz tartozó POM beállításait, melyeket felülírhat.

Projekt objektum modell (POM) (2)

- XML séma:
  http://maven.apache.org/xsd/maven-4.0.0.xsd
- Dokumentáció:
  - *POM Reference*
    https://maven.apache.org/pom.html
  - https://maven.apache.org/ref/current/maven-model/maven.html

Minimális POM (1)

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

Minimális POM (2)

- A JDK 9-től kezdve további információkat is meg kell adni a fordításhoz, amint alább látható:

```
<project xmlns="http://maven.apache.org/POM/4.0.0">
  <modelVersion>4.0.0</modelVersion>
  <groupId>hu.unideb.inf</groupId>
  <artifactId>maven-hello</artifactId>
  <version>1.0</version>
  <properties>
    <maven.compiler.source>17</maven.compiler.source>
    <maven.compiler.target>17</maven.compiler.target>
   </properties>
</project>
```

Szuper-POM

- A szuper-POM a Maven által alapértelmezésben használt POM.
- Ha egy projektnek nincs explicit módon megadott szülője, akkor az úgynevezett szuper-POM beállításait örökli.
- 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 projektet a Maven koordinátái azonosítanak, mely a következő 3 komponensből áll:
  - **`groupId`**: csoportazonosító, melynél gyakori a fordított domain-nevek használata (például `org.apache.maven.plugins`, `com.google.guava`), de nem kizárólagos (például `commons-io`, `junit`)
  - **`artifactId`**: projektnév (például `maven-site-plugin`, `guava`)
  - **`version`**: a projekt verziószáma (például `1.0`, `1.0-SNAPSHOT`)

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 (példa: `org.jsoup:jsoup:1.15.3`).

Maven koordináták (3)

- Lehetővé teszik a függőségként történő hivatkozást, mint például:

```
<dependency>
  <groupId>org.jsoup</groupId>
  <artifactId>jsoup</artifactId>
  <version>1.15.3</version>
  <scope>compile</scope>
</dependency>
```

Csomagolás

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

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>3.12.1</version>
      </plugin>
      ```
- A rendelkezésre álló bővítmények listája: *Available Plugins* https://maven.apache.org/plugins/
- Minden bővítményhez tartozik egy olyan előtag, mely lehetővé teszi az egyes célokra *előtag*`:`*cél* formában történő hivatkozást, mint például `site:deploy`.
      ```

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* https://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 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 a `.m2` alkönyvtárban (Linux rendszerekben a `~/.m2/repository/` alkönyvtárban).
- A távoli és lokális tárolók azonos felépítésűek.
- Lásd: *Introduction to Repositories* https://maven.apache.org/guides/introduction/introduction-to-repositories.html

Tárolók (2)

- A tárolókban a csoportazonosító leképezése egy könyvtárszerkezetre.
  - Példa: `org.apache.maven.plugins` → `/org/apache/maven/plugins/`
    - A könyvtárszerkezetben további alkönyvtárak, melyek neve az `artifactId` és `version` komponensek értékével egyezik meg (példa: `org.jsoup:jsoup:1.15.3` → `/org/jsoup/jsoup/1.15.3`).
- A Maven 3.*x* verziói külön tárolókat tudnak használni a függőségekhez és a bővítményekhez.

Tárolók (3)

- Szoftverek tárolók üzemeltetéséhez (*repository managemenet software*):
  - Szabad és nyílt forrású szoftverek:
    - *Apache Archiva* (licenc: Apache License 2.0) https://archiva.apache.org/ https://github.com/apache/archiva
    - *Artifactory Open Source* (licenc: GNU GPL v3) https://jfrog.com/community/download-artifactory-ce/
    - *Nexus Repository OSS* (licenc: Eclipse Public License v1.0) https://www.sonatype.com/products/repository-oss-download
  - Nem szabad szoftverek:
    - *Artifactory* https://jfrog.com/artifactory/
    - *Nexus Repository* https://www.sonatype.com/products/repository-pro

Maven központi tároló

- Webhely: https://repo.maven.apache.org/maven2/
- Keresés: https://search.maven.org/
  https://central.sonatype.dev/
  - Alternatíva: https://javalibs.com/
- Statisztikák:
  - https://search.maven.org/stats
  - *How many artifacts are in Maven Central Repository?*
    https://javalibs.com/charts/central

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 **kötés**nek 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.
  - Egy fázishoz kötött célok abban a sorrendben kerülnek végrehajtásra, amelyben a POM-ban deklarálására kerülnek.
- Három szabványos életciklus: `clean`, `default`, `site`
  - A csomagolás módjától függően a fázisokhoz alapértelmezésben hozzárendeltek bizonyos célok.
- Lásd: *Introduction to the Build Lifecycle* https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html

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*
  https://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* https://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* https://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 `${env.PATH}` a `PATH` környezeti változó értékét szolgáltatja.
  - A hivatkozásban megadható Java rendszertulajdonság neve.
    - Példa: `${java.home}`, `${line.separator}`
  - `${project.`*x*`}` formájú hivatkozások helyettesítése a POM megfelelő elemének értékével. Csak egyszerű típusú elemekhez használható!
    - Példa: `${project.groupId}`, `${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.
  - Példa:

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

Függőségek kezelése

- A Maven a Maven Artifact Resolver könyvtárat használja a függőségek kezeléséhez.
  - Lásd: 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>path</systemPath>
    <exclusions>
      <exclusion>
        <groupId>groupId</groupId>
        <artifactId>artifactId</artifactId>
      </exclusion>
      ...
    </exclusions>
  </dependency>
  ...
</dependencies>
```

Függőségek megadása (2)

- **`groupId`**, **`artifactId`**, **`version`**: a függőség Maven koordinátáit tartalmazzák
- **`classifier`**: az egy projekt által létrehozott termékek megkülönböztetésére szolgál
  - Tipikus értéke például a `javadoc` és `sources`.
- **`type`**: a függőség típusát tartalmazza (alapértelmezés: `jar`)
  - A típus meghatározza a termék állománynév kiterjesztését és csomagolását, valamint (opcionálisan) az osztályozót is.
  - Lásd: *Default Artifact Handlers Reference* https://maven.apache.org/ref/current/maven-core/artifact-handlers.html
- **`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* tartalmazza a függőséget, a függő projekteknek is függősége lesz (ez az alapértelmezés)
  - **`provided`**: a függőséget a futtató környezet (például a JDK) biztosítja, csak a fordításhoz használt *classpath* tartalmazza, nem tranzitív
  - **`runtime`**: a függőség csak a végrehajtáshoz szükséges (a programtesztek végrehajtásánál is rendelkezésre áll)
  - **`system`**: a függőséget nem egy tároló szolgáltatja, hanem a lokális állományrendszerben található
  - **`test`**: a függőség csak a programtesztek fordításához és végrehajtásához áll rendelkezésre, nem tranzitív
  - **`import`**: kizárólag `pom` típusú függőségekhez adható meg a `dependencyManagement` részben, egy ilyen függőség kicserélésére kerül a POM-ja `dependencyManagement` részének függőségeire
- Lásd: *Introduction to the Dependency Mechanism – Dependency Scope* https://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html#Dependency_Scope

Függőségek megadása (4)

- **`systemPath`**: `system` hatáskörű függőséghez megengedett és 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)

- A verziószámok *p*`.`*q*`.`*r*`-`*s* alakúak, ahol
  - *p* a főverzió (*major version*),
  - *q* az 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).
  - A 2019. szeptemberében kiadott 3.6.2 verziótól kezdve a `release` minősítő is használható az ‹*üres sztring*›/`final`/`ga` megfelelőjeként.

Verziószámok (2)

- Példa verziószámokra:
  - `1.2`
  - `4.8.2`
  - `1.6.0-alpha2`
  - `1.0-beta9`
- A verziószámok komponensekre történő bontása a `'.'` és `'-'` karaktereknél, valamint a számjegyek és betűk közötti átmeneteknél.

Verziószámok (3)

- Rendezés értelmezése a verziószámokon (kiterjesztés a szabványos alaktól eltérő formájú verziószámokra is).
  - A rendezés komponensenként történik, balról jobbra haladva.
    - A csak számjegyekből áll komponensek rendezése numerikusan történik.
  - 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 (4)

- 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 argumentumokként.
- Lásd: *POM Reference – Version Order Testing* - https://maven.apache.org/pom.html#Version_Order_Testing

Verziószámok (5)

- Lásd még:
  - *POM Reference – Version Order Specification*
    https://maven.apache.org/pom.html#Version_Order_Specification
  - `org.apache.maven.artifact.versioning.ComparableVersion`
    https://maven.apache.org/ref/current/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

Verzió követelmények (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).
    - Példa: `(,1.0),(1.0,)`

Verzió követelmények (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>
```

Verzió követelmények (3)

- Ha egy függőséghez a `version` elemben egyetlen verziószám szerepel, 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>[4.13.2]</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.
- Lásd: *Introduction to the Dependency Mechanism – Transitive Dependencies*
  https://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html#Transitive_Dependencies

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)

- Az alábbi példában a `hsqldb` termék a `project-A` projektnek is implicit módon `runtime` hatáskörű függősége.

```
<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.hsqldb</groupId>
   <artifactId>hsqldb</artifactId>
   <version>2.7.1</version>
   <scope>runtime</scope>
  </dependency>
 </dependencies>
 ...
</project>
```

Tranzitív függőségek kizárása (1)

- Tranzitív függőségek kizárására szolgál az `exclusions` elem.
  - Konfliktus esetén szükséges lehet, de hasznos felesleges függőségek kizárásához is.
- Lásd: *Optional Dependencies and Dependency Exclusions*
  https://maven.apache.org/guides/introduction/introduction-to-optional-and-excludes-dependencies.html

Tranzitív függőségek kizárása (2)

- Példa:
  - Az Apache HttpClient programkönyvtár alapértelmezésben az Apache Commons Logging programkönyvtárat használja naplózáshoz. A következő POM részlet azt szemlélteti, hogyan helyettesíthető a Commons Logging az SLF4J-vel.
    - Lásd: https://hc.apache.org/httpcomponents-client-4.5.x/logging.html

Tranzitív függőségek kizárása (3)

- Példa: (folytatás)
  - A `httpclient` függőség hozzáadása a `commons-logging` függősége kizárásával:

```
<dependencies>
  <dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
    <version>4.5.14</version>
    <scope>compile</scope>
    <exclusions>
      <exclusion>
        <groupId>commons-logging</groupId>
        <artifactId>commons-logging</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
```

Tranzitív függőségek kizárása (4)

- Példa: (folytatás)
  - A `commons-logging`-ot helyettesítő függőségek hozzáadása:

```
  <dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>jcl-over-slf4j</artifactId>
    <version>2.0.6</version>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-classic</artifactId>
    <version>1.4.5</version>
    <scope>runtime</scope>
  </dependency>
  ...
</dependencies>
```

Tranzitív függőségek kizárása (5)

- Függőség összes tranzitív függőségének kizárása:

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

Alapértelmezések 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 alapértelmezéseket (alapértelmezett verziószámokat és/vagy hatásköröket) szolgáltatnak a megnevezett termékekhez, mely lehetővé teszi, hogy a projektben és a gyermek projektekben ezen információk megadása nélkül lehessen függőségként hivatkozni rájuk.
- Lásd: *Introduction to the Dependency Mechanism – Dependency Management* https://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html#Dependency_Management

Alapértelmezések szolgáltatása függőségekhez (2)

- Példa:

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.jsoup</groupId>
      <artifactId>jsoup</artifactId>
      <version>1.15.3</version>
      <scope>test</scope>
    </dependency>
    ...
  </dependencies>
</dependencyManagement>
```

Alapértelmezések 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>org.jsoup</groupId>
    <artifactId>jsoup</artifactId>
  </dependency>
  ...
</dependencies>
```

Snapshot verziók

- Verziószámok végén a `SNAPSHOT` utótaggal jelezhető, hogy a projekt aktív fejlesztés alatt áll.
  - Példa: `1.0-SNAPSHOT`, `SNAPSHOT`
- A termék távoli tárolóba való kihelyezésekor a `SNAPSHOT` utótag kifejtése az aktuális rendszeridővel (UTC idő használata).
  - Például közép-európai idő szerint 2022. január 28-án 21:58:34-kor a fenti verziószám esetén a helyettesítés eredménye az `1.0-20220128.205834-`*N* verziószám.
    - *N* értéke 1-ről indul, minden további kihelyezésnél 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 kihelyezéséhez.
- 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.

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</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</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`, `scm`, `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` 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.

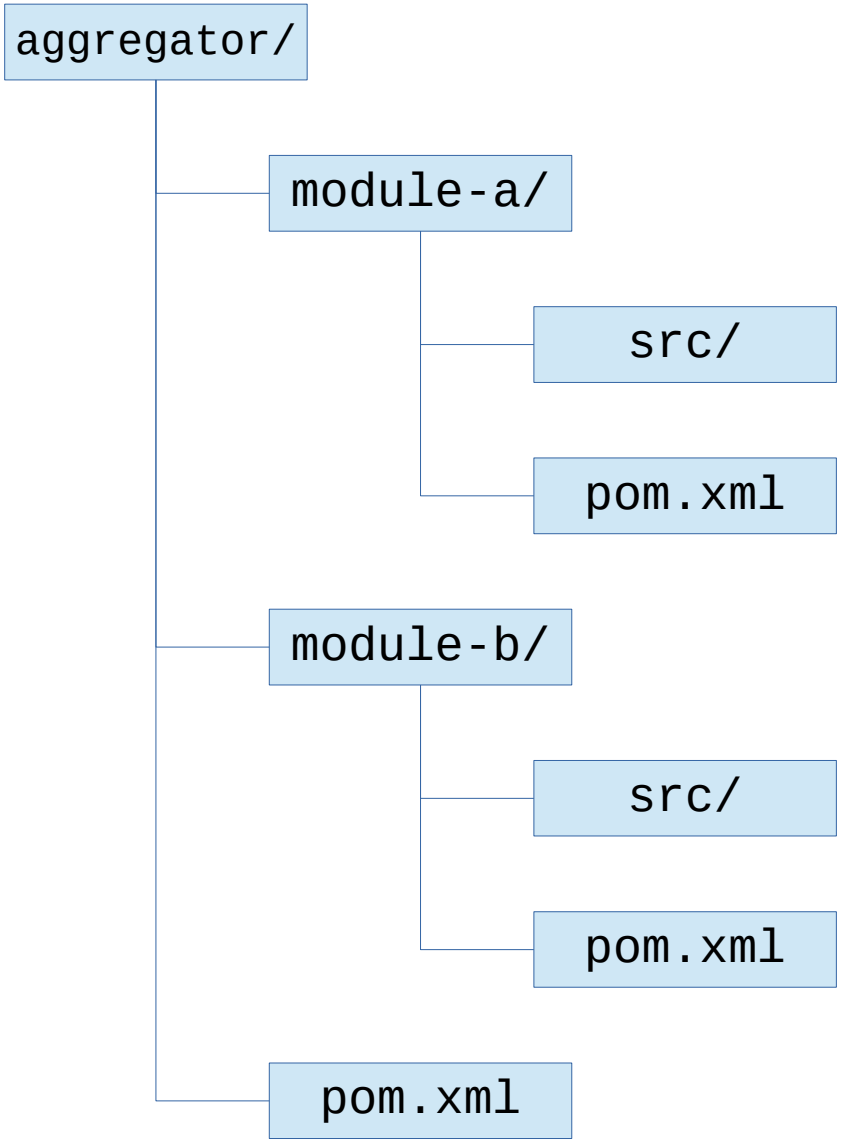
Többmodulos projektek (2)

- 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>hu.unideb.inf</groupId>
   <artifactId>aggregator</artifactId>
   <packaging>pom</packaging>
   <version>1.0</version>
   <modules>
     <module>module-a</module>
     <module>module-b</module>
   </modules>
   ...
 </project>
```

Többmodulos projekt létrehozása: m2eclipse

- 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.
- 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

- Hozzuk létre az aggregátor projektet: *File → New Project*
  - Válasszuk a *POM Project* opciót a *Java with Maven* kategóriából.
- 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.

Többmodulos projekt létrehozása: IntelliJ IDEA

- Hozzuk létre az aggregátor projektet: *File* → *New* → *Project…* → *Maven*
- Egy modul létrehozásához a teendők: *File* → *New* → *Module…*
  - Válasszuk ki az aggregátor projektet az *Add as a module to* mező melletti gombra kattintva.
  - Az aggregátor projekt kiválasztható a modul szülőjeként is a *Parent* mező melletti gombra kattintva.
- Lásd: *Maven – Configuring a multi-module Maven project* https://www.jetbrains.com/help/idea/maven-support.html#maven_multi_module

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)

- Az `os` elem szolgál profilok az operációs rendszer alapján történő aktiválására:
  -
```
<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 10`, …)
  - **`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>
    <family>mac</family>
  </os>
</activation>
```

Profil aktiválás: JDK

- A `jdk` elem szolgál profilok a JDK verziószáma alapján történő aktiválására.
  - Az elemben megadható verziószám kezdőszelet vagy verziótartomány is.
    - Verziószám kezdőszelet esetén negáció, ha az első karakter `'!'`.

- Példa:

```
<profile>
  <id>jdk8</id>
  <activation>
    <jdk>1.8</jdk>
  </activation>
  ...
</profile>
```

- Példa:

```
<profile>
  <id>pre-jdk8</id>
  <activation>
    <jdk>[,1.8)</jdk>
  </activation>
  ...
</profile>
```

Profil aktiválás: állományok (1)

- A `file` elem segítségével adott állományok létezéséhez és/vagy hiányához köthető az aktiválás.
  - Az `exists` és `missing` elemekben egy állomány elérési útvonalát kell megadni.

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

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 (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>
  ...
</build>
```

Bővítmények használata (2)

- A `plugin` elemben rendelkezésre álló elemek:
  - **`groupId`**, **`artifactId`**, **`version`**: a bővítmény Maven koordinátái
  - **configuration**: konfigurációs paramétereket tartalmaz a célok végrehajtásához
    - Az XML séma a tartalomra nem tesz semmilyen megszorítást.
    - Ezek a konfigurációs paraméterek valamennyi bővítmény-célra vonatkoznak.
  - **`dependencies`**: a bővítményhez szükséges függőségeket tartalmazza
    - A függőségek megadása a korábban tárgyalt formában történik.

Bővítmények használata (3)

- A `plugin` elemben rendelkezésre álló elemek (folytatás):
  - **executions**: lehetővé teszi bővítmény-célok végrehajtásának hozzákötését életciklus fázisokhoz, így az összeállítási folyamat testreszabását (részletesen lásd később)
  - **extensions**: azt jelzi, hogy be kell-t tölteni a bővítmény kiterjesztéseit (alapértelmezés: `false`)
  - **`inherited`**: azt jelzi, hogy öröklés során át kell-e venni a bővítmény beállításait (alapértelmezés: `true`)

Bővítmények használata (4)

- Az `executions` elem:

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

Bővítmények használata (5)

- Az `execution` elemben rendelkezésre álló elemek:
  - **`id`**: a végrehajtás egyedi azonosítója
  - **`phase`**: az életciklus fázis neve, melyhez hozzá kell kötni a cél(ok) végrehajtását
  - **`goals/goal`**: a végrehajtandó bővítmény-célok neveit tartalmazzák
  - **`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 (6)

- Példa az `executions` elem használatára:

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-shade-plugin</artifactId>
      <version>3.4.1</version>
      <executions>
        <execution>
          <phase>package</phase>
          <goals>
            <goal>shade</goal>
          </goals>
          <configuration>
            <minimizeJar>true</minimizeJar>
          </configuration>
        </execution>
      </executions>
    </plugin>
    ...
```

Bővítmények használata (7)

- Egy bővítmény-célhoz tartozhat egy alapértelmezett életciklus fázis, ekkor az `execution` elemben nem szükséges megadni a `phase` elemet.
  - Például a `lombok-maven-plugin` bővítmény `delombok` célja alapértelmezésben a `generate-sources` életciklus fázishoz van hozzákötve. (Lásd a következő oldalon.)
- Ha nincs alapértelmezett életciklus fázis, akkor a `phase` elem hiányában a bővítmény-cél nem kerül végrehajtásra!

Bővítmények használata (8)

```
<build>
  <plugins>
    <plugin>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok-maven-plugin</artifactId>
      <version>1.18.20.0</version>
      <executions>
        <execution>
          <goals>
            <goal>delombok</goal>
          </goals>
          <configuration>
            <verbose>true</verbose>
          </configuration>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

Bővítmények konfigurálása (1)

- Példa:
  - *Exec Maven Plugin*
    https://www.mojohaus.org/exec-maven-plugin/
    - Az `exec:java` céllal hajtható végre egy Java osztály az aktuális Java virtuális gépben.

Bővítmények konfigurálása (2)

- Konfigurációs beállítások megadása a `configuration` elemben:

```
<build>
  <plugins>
    <plugin>
      <groupId>org.codehaus.mojo</groupId>
      <artifactId>exec-maven-plugin</artifactId>
      <version>3.1.0</version>
      <configuration>
        <mainClass>pkg.Main<mainClass>
        <commandlineArgs>
          -o out.txt
        </commandlineArgs>
      </configuration>
    </plugin>
  </plugins>
</build>
```

Bővítmények konfigurálása (3)

- Konfigurációs beállítások megadása a `properties` elemben:

```
<properties>
  <exec.mainClass>pkg.Main</exec.mainClass>
  <exec.args>-o out.txt</exec.args>
</properties>
<build>
  <plugins>
    <plugin>
      <groupId>org.codehaus.mojo</groupId>
      <artifactId>exec-maven-plugin</artifactId>
      <version>3.1.0</version>
    </plugin>
  </plugins>
</build>
```

Bővítmények konfigurálása (4)

- Konfigurációs beállítások megadása a parancssorban:

```
$ mvn exec:java -Dexec.mainClass=pkg.Main \
  -Dexec.args="-o out.txt"
```

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 s*napshot* 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 elérési beállítások (1)

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

Tároló elérési beállítások (2)

```
<repositories>
  <repository>
    <id>azonosító</id>
    <name>név</name>
    <url>URI</url>
    <layout>default|legacy</layout>
    <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 elérési 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 elérési 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 elérési 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* nemnegatív egész szám): *N* percenként
    - **`never`**: soha

Tároló elérési beállítások (6)

- Bővítményeket szolgáltató tárolókhoz a `pluginRepositories` felső szintű elemet kell használni.
  - Az elemben megadható `pluginRepository` elemek tartalma megegyezik a `repository` elemekével.

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

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 `pluginRepository` elemekben rendelkezésre álló `snapshots` elemben történik.

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

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* nemnegatív 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.

Termékek 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, melyek nem állnak rendelkezésre egyetlen elérhető távoli tárolóban sem.

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>4.0.0-M5</version>
      <configuration>
        <reportPlugins>
          jelentéskészítő-bővítmények felsorolása (plugin elemek)
        </reportPlugins>
      </configuration>
    </plugin>
    ...
  </plugins>
</build>
```

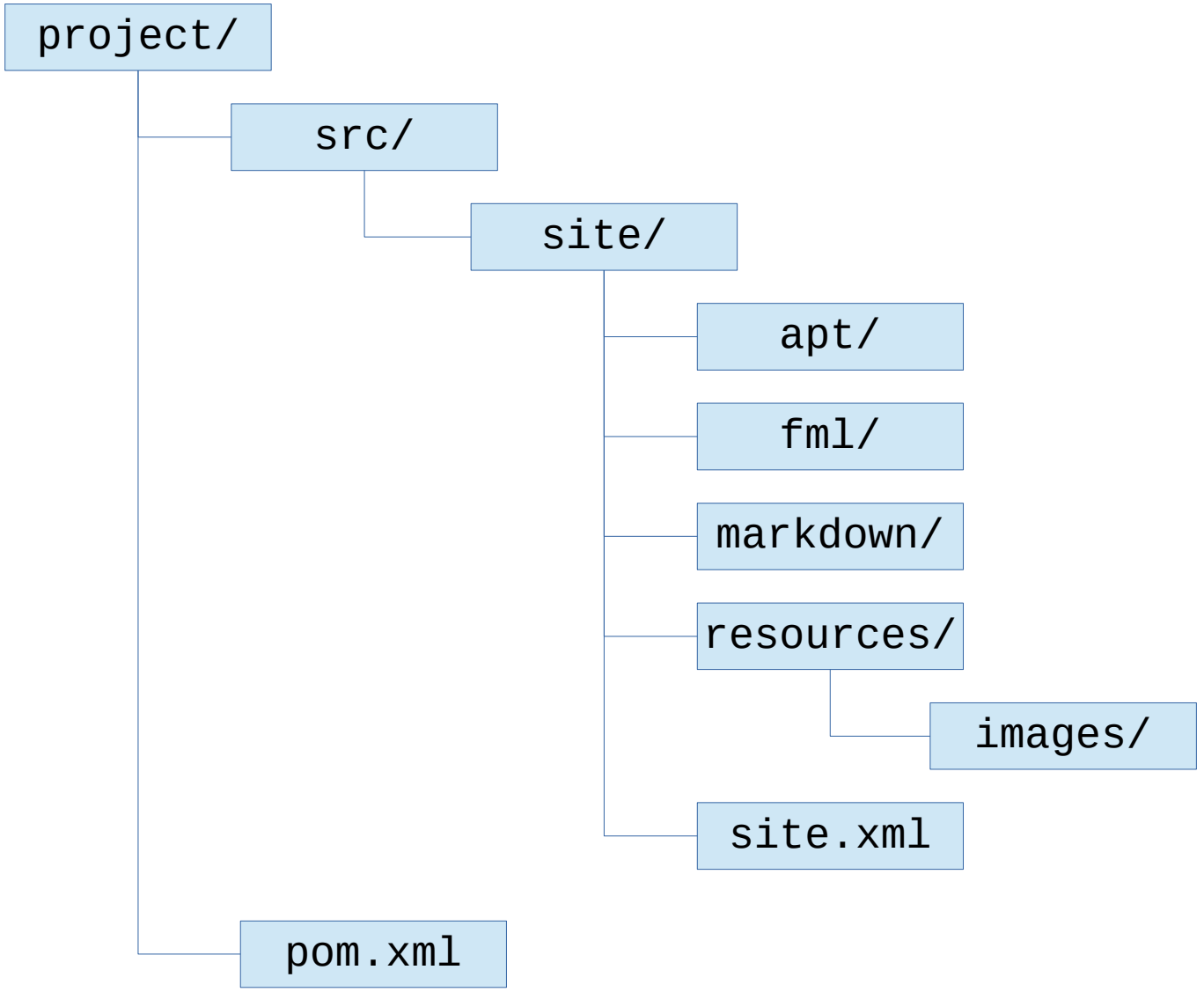
Webhely készítése (3)

- 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-1.8.0.xsd
  - A könyvtár alatt speciális alkönyvtárak helyezhetők el, melyek a webhelyhez szolgáltatnak tartalmat.
    - Speciális formátumok használata, amelyekből automatikusan HTML oldalak jönnek létre.

Webhely testreszabása (2)



Webhely testreszabása (3)

- Formátumok: https://maven.apache.org/doxia/references/
  - AsciiDoc https://asciidoc.org/
  - FML (FAQ Markup Language) https://maven.apache.org/doxia/references/fml-format.html
  - Markdown https://daringfireball.net/projects/markdown/

Maven Wrapper (1)

- Lehetővé teszi a fejleszők számára csomagoló szkriptek (`mvnw` és `mvnw.cmd`) elhelyezését a Maven projektekben a Maven telepítéséhez és futtatásához.
- Ez a funkcionalitás a Maven Wrapper Plugin-en keresztül érhető el, lásd a `wrapper:wrapper` célt.
- Webhely: https://maven.apache.org/wrapper/
- Tároló: https://github.com/apache/maven-wrapper

Maven Wrapper (2)

- Csomagoló szkriptek hozzáadása az aktuális projekthez (egy `pom.xml` állomány szükséges hozzá az aktuális könyvtárban):
  - `mvn wrapper:wrapper`
    `mvn wrapper:wrapper -Dmaven=3.9.0`
- Projekt összeállítása a csomagoló szkriptekkel:
  - `./mvnw clean compile # Unix-szerű rendszerek`
  - `mvnw.cmd clean compile # Windows rendszerek`

Apache Maven Daemon (1)

- Egy hosszú élettartamú háttérfolyamat (démon) használatával gyorsítja fel a Maven összeállítási folyamatot.
- Egy beágyazott Mavent tartalmaz, ezért nem szükséges azt külön telepíteni.
- Tároló: https://github.com/apache/maven-mvnd
- Telepítés (SDKMAN!): `sdk install mvnd`

Apache Maven Daemon (2)

- Az `mvnd` kliens egy, a GraalVM-mel létrehozott natív végrehajtható állomány, gyorsabban indul el és kevesebb memóriát használ egy hagyományos virtuális géppel történő elindításhoz képest.
- Egy démon példány az `mvnd` kliens több egymást követő kérését is kiszolgálhatja.
- Modulok összeállítása párhuzamosan történik.

Apache Maven Daemon (3)

- Használat:
  - Az `mvnd` ugyanazokat a parancssori opciókat fogadja el, mint a Maven, plusz néhány továbbit (például `--status` és `--stop`).
    - `mvnd clean compile`
      `mvnd --help`
      `mvnd --status`
      `mvnd package`
      `mvnd --stop`