Hello Maven

JSE vs. JEE, JEE vs Spring

Óbudai Egyetem, Java Enterprise Edition
Műszaki Informatika szak
Labor 2

Bedők Dávid
2017.09.18.
v0.1

Java project struktúra

Avagy hogyan szervezzük forrásainkat?

- **javac**
- **IDE** (Eclipse, IntellJ IDEA, ...)
- **build tool**

Java Build eszközök:

- batch files / bash scripts
- Apache ANT (+ Apache IVY)
- Apache Maven
- Gradle

Miket érdemes figyelembe venni?

- Lesznek **egység tesztjeink**?
  - külön `source folder` a teszteknek (**test** és **main**)
- Lesznek **erőforrásaink**?
  - külön `source folder` az erőforrásoknak (**resources**)
- Lesznek **nem java forrásaink** is?
  - külön `source folder` a forrásoknak (**java**, **scala**, **groovy**, stb.)

Java filozófia

- átláthatóság, egyértelműség
- **classpath** (cp) használata
- JAR, WAR, EAR, SAR*, APK** csomagolások alkalmazása

SAR*: JBoss specifikus
APK**: android specifikus

Java project struktúra

javac

Csak a "fantázia" szab határt a konfigurációnak (nincsenek szabályok)! `javac` programnak megadjuk az összes olyan könyvtárat, ahol forrásállományok találhatóak.

Könyvtár struktúra

`bin/`
`src1/` → **source folder**
`src2/` → **source folder**

Az `Application` osztály használja az `ImperialToMetricCalculator` osztály egy példányát. Mindkettő azonos csomagba fordul, ezért az alkalmazás nem importálja a kalkulátort, futás időben egy helyen lesznek.

`\hellotest`

```
1 > javac -d ./bin ./src1/hu/qwaevisz/demo/Application.java
      ./src2/hu/qwaevisz/demo/ImperialToMetricCalculator.java
2 > java -cp ./bin hu.qwaevisz.demo.Application
```

Java project struktúra

Eclipse IDE

Eclipse "Java project" alapértelmezett konfigurációja:

Könyvtár struktúra

```
bin/
src/ → source folder
```

Eclipse "Java project" egység tesztekkel konfigurációja:

Könyvtár struktúra

```
bin/
src/
    main/ → source folder
    test/ → source folder
```

A legtöbb IDE-ben ezek teljeskörűen konfigurálható elemek (Eclipse: `Project properties` | `Java Build Path` | `Source` tab)!

Maven alapértelmezett könyvtár konfigurációja:

Könyvtár struktúra

```
src/
    main/
        java/ → source folder
        resources/ → source folder
    test/
        java/ → source folder
        resources/ → source folder
```

Ezektől Maven-ben is szabadon el lehet térni, de nem érdemes. Ha élünk ezzel, akkor minimális konfigurációval el tudjuk kezdeni a munkát!

Megjegyzés: A `resources` könyvtárak bár tipikusan nem tartalmaznak lefordítandó forrást, ugyanúgy rajta vannak a `classpath`-on, hogy runtime a tartalmuk feldolgozható legyen.

JAR - Java ARchive

ZIP formátum, mely (Java) `byte code`-okat (`*.class`), konfigurációs állományokat (pl. `*.properties`, `*.xml`, stb.) illetve egy speciális metaadatokat (kulcs-érték párokat) tartalmazó **MANIFEST.MF** állományt tartalmaz.

Könyvtár struktúra

```
META-INF/
    MANIFEST.MF
hu/
    qwaevisz/
        demo/
            HelloWorld.class
            Lorem.class
log4j.xml
```

Struktúrája előre definiált, opcionálisan forrásállományokat (pl. `*.java`, `*.groovy`, stb.) is magával hordoz (azonos helyen a `byte code`-dal).

```
1 Manifest-Version: 1.0
2 Created-By: 1.7.0_67 (Oracle Corporation)
```

MANIFEST.MF

Executable JAR file

A `Main-Class` kulcsnak szerepelnie kell a `MANIFEST.MF` állományban, és a belépési pontként szolgáló osztály `full qualified` neve lesz az értéke.

```
1 Manifest-Version: 1.0
2 Created-By: 1.7.0_67 (Oracle Corporation)
3 Main-Class: hu.qwaevisz.demo.Application
```

`MANIFEST.MF`

```
1 > cd bin
2 > jar cvfe calculator.jar hu.qwaevisz.demo.Application
      hu/qwaevisz/demo/Application.class
      hu/qwaevisz/demo/ImperialToMetricCalculator.class
3 > cd ..
4 > java -jar bin/calculator.jar
```

create new archive
verbose
specify archive **f**ile name (2)
sepcify **e**ntry point (main class) (3)

Eclipse IDE for Java EE Developers

Letöltés: https://www.eclipse.org/downloads/
Verzió: **4.7.0**
Telepítés: unzip vagy installer
Integrált kiegészítők (plugins):

- Gradle
- Maven
- Git
- EclEmma Java Code Coverage
- ...

Magyar billentyűzet kiosztás esetén a "{" (Ctrl + B) használata: `Preferences | General | Keys | Skip all brakepoints` (Ctrl + Alt + B) → Unbind

Eclipse IDE alapvető kezelése: http://users.nik.uni-obuda.hu/bedok.david/jse.html

Továbbiak telepítése (`Help / Eclipse Marketplace`):

- **TestNG** (filter: `testng`)
  - http://beust.com/eclipse

Eclipse beállítása

Code Style Formatter

`Window` | `Preferences` (type: `formatter`)

- ▷ `Java` | `Code Style` | `Formatter`
  - New... / Import...: **uni-obuda-java-formatter**
    - Initialize: Eclipse [build-in]
    - Indentation | Indent | Statement within 'switch' body
    - Line Wrapping | General | Maximum line width: 160
    - Line Wrapping | Enum declaration
      * Policy: Wrap all elements, every element on a new line
      * Constants policy: Wrap all elements, every element on a new line + Force split
    - Comments | Line width | Maximum: 120

`\eclipse\uni-obuda-java-formatter.xml`

Eclipse beállítása

Save Actions

`Window` | `Preferences` (type: `save actions`)

- ▷ `Java` | `Editor` | `Save Actions`
  - Perform the selected actions on save
    - **Format source code** (all lines)
    - Organize imports
    - Additional actions - Configure
      - Code Organaizing: Remove trailing whitespaces
      - Code Style: Use blocks in if/while/for/do statements
      - Member Accesses: Use 'this' qualifier for field accesses: Always
      - Member Accesses: Use 'this' qualifier for method accesses: Always
      - Unnecessary Code: Remove unused imports

Letöltés: https://www.jetbrains.com/idea/

- **Kereskedelmi termék**
- Community verzió pl. JavaEE-t nem támogat, de ezen IDE támogatás nélkül is tökéletesen alkalmas professzionális munkára (akár JavaEE projektek kezelésére is)
- Néhol gyorsabb mint Eclipse
- Más billentyűkiosztás, átszokni nem könnyű, de van lehetőség más IDE kiosztásának használatára
- Beépített Maven/Gradle/Git plugin

Hello World

src | main | java | hu | qwaevisz | hello | Application.java

```
1  package hu.qwaevisz.hello;
2
3  public class Application {
4
5    public static void main(final String[] args) {
6      System.out.println("Hello World");
7    }
8
9    public int add(final int a, final int b) {
10     return a + b;
11   }
12
13 }
```

Application.java

[gradle|maven]\helloworld

TestNG

3rd party library

- http://testng.org/
- GitHub: https://github.com/cbeust/testng
- Verzió: **6.11**
- Artifactory URL:
  - `'org.testng:testng:6.11'`
  - group/groupId: `org.testng`
  - name/artifactId: `testng`
  - version: `6.11`

Egység teszt TestNG-vel

src | test | java | hu | qwaevisz | hello | ApplicationTest.java

```
1  package hu.qwaevisz.hello;
2
3  import org.testng.Assert;
4  import org.testng.annotations.Test;
5
6  public class ApplicationTest {
7
8    @Test
9    public void addNumbers() {
10     Application app = new Application();
11     Assert.assertEquals(app.add(2, 3), 5);
12   }
13
14 }
```

ApplicationTest.java

Apache Maven

Build tool

- https://maven.apache.org/
- Letöltés: https://maven.apache.org/download.cgi
- Verzió: **3.5.0**
- A szoftver teljes (fejlesztési) életciklusát támogatni kívánó eszköz.
- monorepo és multi-repo támogatás
- POM: Project Object Model
- Telepítés: unzip

Környezeti változók:

- **MAVEN_HOME** →`c:\apps\apache-maven-3.3.9`
- **Path** módosítása → `%Path%;%MAVEN_HOME%\bin`

A Maven elsődleges célja hogy teljeskörűen kezelje és összehangolja a fejlesztési folyamatokat, és mindezt a lehető legrövidebb (fejlesztési) idő befektetése mellett produkálja.

- A build folyamat könnyűvé tétele
- Egységes build rendszer kialakítása
- Minőségi projekt információk biztosítása
- Fejlesztési útmutatások biztosítása "best practice"-ek alapján
- Áttekinthető migráció biztosítása új lehetőségek bevezetésekor

A Maven egy **plugin execution framework**, minden munkát plugin-ek hajtanak végre.

Maven

```
1 > mvn --version
2 Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5;
      2015-11-10T17:41:47+01:00)
3 Maven home: c:\apps\apache-maven-3.3.9\bin\..
4 Java version: 1.8.0_102, vendor: Oracle Corporation
5 Java home: c:\apps\java\jdk1.8.0_102\jre
6 Default locale: en_US, platform encoding: Cp1252
7 OS name: "windows 7", version: "6.1", arch: "amd64", family: "dos"
```

validate a project korrektségének és a szükséges információk meglétének ellenőrzése

compile a project forrásának lefordítása

test a lefordított források **egység tesztelése** test framework alapján

package a lefordított forrás becsomagolása a szállítási formátumba (pl. `jar`)

verify **integrációs tesztek** futtatása és minőség ellenőrzési kritériumok vizsgálata

install a szállított `artifact` helyi `repository`-ba telepítése (így elérhetővé válik más helyi project-ek számára mint függőség)

deploy a szállított `artifact` távoli `repository`-ba telepítése (így elérhetővé válik más fejlesztők és project-ek számára)

clean a korábban elkészült `artifact`(ok) takarítása

site a project dokumentációjának generálása

A **phase**-ek valójában a hozzájuk tartozó **goal**-okat fogját futtatni, contextus alapján (pl. `packages` végrehajt egy `jar:jar` goal-t, ha a project típusa `jar`, és `war:war` goal-t, ha a project csomagolási típusa `war`).

Az **archetype**ok segítségével legenerálhatunk **blueprint** projekteket, mankókat, melyeket egyébként is elkészítenénk, mert mindenki így járna el (best practices).
Ezek nélkül a legegyszerűbb “hello world” példa is viszonylagosan sok "gépeléssel" járna.
Későbbiekben teljesen figyelmen kívül fogjuk hagyni az archetype-okat...

HelloWorld létrehozása archetype segítségével

maven\helloworld

```
1 > mvn archetype:generate -DgroupId=hu.qwaevisz.hello
    -DartifactId=hellomaven
    -DarchetypeArtifactId=maven-archetype-quickstart
    -DinteractiveMode=false
```

- **archetype** →Maven plugin
- **generate** →goal (a plugin-hez tartozik)

Létrejön a projekt struktúra és a `pom.xml`.

Hello Maven!

```
1  <project xmlns="http://maven.apache.org/POM/4.0.0"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
       http://maven.apache.org/maven-v4_0_0.xsd">
2    <modelVersion>4.0.0</modelVersion>
3    <groupId>hu.qwaevisz.hello</groupId>
4    <artifactId>hellomaven</artifactId>
5    <packaging>jar</packaging>
6    <version>1.0</version>
7    <name>Hello Maven</name>
8    <properties>
9      <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding
10   </properties>
11   <dependencies>
12     <dependency>
13       <groupId>org.testng</groupId>
14       <artifactId>testng</artifactId>
15       <version>6.11</version>
16       <scope>test</scope>
17     </dependency>
18   </dependencies>
19 </project>
```

<packaging>jar</packaging> →A projekt kimenete egy JAR állomány lesz.

```
1 > mvn clean package
```

Output: target/hellomaven-1.0.jar

```
1 > java -cp target/hellomaven-1.0.jar
    hu.qwaevisz.hello.Application
```

Maven Eclipse integráció

- Eclipse **m2e plugin**je felismeri a `maven` konfigurációs állományait, és annak megfelelően képes kezelni a projektet.
- Létezik Maven-hez is **Eclipse plugin**, mely legyártja az Eclipse specifikus állományokat, de ez a plugin már *deprecated*.

Eclipse Maven project

`File` | `Import...` | `Maven` | `Existing Maven Project`

- Project root directory: `\helloworld`

Mit honnan?

Az Eclipse Maven plugin (`m2e`) elsősorban a Maven projekt struktúra kezelése, a projektek importálása miatt fontos. Eclipse-ből futtatni Maven goal-okat már ízlés dolga (van aki minden ilyesmit Eclipse-ből szeret futtatni, van aki ezt külön terminal/command window-ban teszi meg).