Moduly v Javě, zavádění programů

Table of Contents

Struktura a zavádění javových programů

- struktura (třídy, balíky, moduly)
- zavádění (classloaders)
- další zdroje (resources)
- reflexe

Struktura a organizace programů

- Javový program sestává z alespoň jedné *třídy* s alespoň jednou *metodou*.
- Má-li být spustitelný z příkazové řádky, pak s metodou `main`.
- Ve spoustě nasazení - webové aplikace, aplikační kontejnery - není `main` třeba .
- Třídy se obvykle sdružují do *balíků* - deklarace `package xx.yy.zz` třeba `cz.muni.fi`.

Moduly

- Nově od Java 9 jsou balíky - přinejmenším v Core API - sdruženy do modulů (Modules).
- Moduly kromě tříd samotných mohou obsahovat další zdroje (*resources*) a obsahují popisovač (*module descriptor file*).
- Cílem je lepší znovupoužitelnost a potřeba u rozsáhlých programů lépe organizovat kód.

- V praxi se kromě organizace ve standardních knihovnách (Java Core API) moc nevyužívají.

Více info k modulům

- A Guide to Java 9 Modularity
- Java 9 Modules - Tutorial
- Java Modules

Účel modulů

- zabalit celou aplikaci nebo API jako samostatný modul
- umožnit sledování a zajištění závislostí mezi moduly
- napodobit "dohnat" to, co už jazyky jako JavaScript nebo Python mají
- technicky jde o soubor JAR s celým modulem, tzn. popisovačem, třídami, příp. dalšími zdroji
- dříve bylo třeba umísťovat zdroje do kořenové úrovně projektu a ručně je spravovat
- nyní lze na jemnější úrovni modulů - podle toho, který zdroj je kde potřeba

Popisovač modulu

Název

název modulu (pojmenování modulů podobně jako u balíků, tj. tečky jsou povoleny, pomlčky nikoli; časté je pojmenování ve stylu projektu (`my.module`) nebo ve stylu Reverse-DNS jako u balíků (`com.baeldung.mymodule`)

Závislosti

seznam dalších modulů, na kterých tento modul závisí

Veřejné balíky

seznam všech balíků, které chceme mít přístupné mimo modul (ve výchozím nastavení jsou všechny balíky soukromé)

Nabízené služby

můžeme poskytovat implementace služeb (*services*), které mohou být využívány jinými moduly

Využívané služby

seznam služeb využívaných tímto modulem

Povolení pro reflexi

explicitně povoluje ostatním třídám používat reflexi pro přístup k soukromým (`private`) prvkům balíku (ve výchozím nastavení jsou všechny třídy nepřístupné pro reflexi)

Hlavní deklarace v popisovači

Export z modulu

- veřejné balíky, viditelné zvnějšku

```
module com.jenkov.mymodule {
    exports com.jenkov.mymodule;
    exports com.jenkov.mymodule.util;
}
```

je třeba exportovat skutečně každý balík, který chceme zviditelnit ven - tzn. i ten `···util`

Závislosti

- závislost na ostatních modulech

```
module com.jenkov.mymodule {
    requires com.jenkov.yourmodule;
    requires com.jenkov.yourmodule.util;
}
```

Cyklické závislosti nejsou přípustné: `module.a` závisí na `module.b`, který závisí na `module.a` NELZE.

Balíky ve více modulech

- NE, jeden balík smí být současně exportován v max jednom modulu.
- V jedné aplikaci se nesmí potkat balík exportovaný současně dvěma moduly, dvěma cestami
- a to ani v případě, že by ve skutečnosti jedna třída byla vždy jen v jednom modulu.

Nepovolený export

Ani uvedené dva exporty nejsou možné:

```
module cz.muni.fi.pb162.mod1 {
    exports cz.muni.fi.pb162.bank;
}
```

- balík `cz.muni.fi.pb162.bank` v tomto modulu obsahuje jen třídu `Bank`

```
module cz.muni.fi.pb162.mod2 {
```

```
    exports cz.muni.fi.pb162.bank;
}
```

- balík `cz.muni.fi.pb162.bank` v tomto modulu obsahuje jen třídu `Account`

Práce s moduly

- kompilace: `javac -d out --module-source-path src/main/java --module com.jenkov.mymodule`
- spuštění aplikace v modulu: `java --module-path out --module com.jenkov.mymodule/com.jenkov.mymodule.Main`
- vytvoření JAR archívu s modulem: `jar -c --file=out-jar/com-jenkov-mymodule.jar -C out/com.jenkov.mymodule .` (`-c` *create*, `--file` soubor s výsledným JAR, `-C` *change directory* změna adresáře před sbalením JARu)
- spuštění z JAR: `java --module-path out-jar -m com.jenkov.mymodule/com.jenkov.mymodule.Main`
- spuštění hlavní třídy (Main-Class) z JAR: `java -jar out-jar/com-jenkov-javafx.jar`

Balení aplikací se závislostmi

- Moduly umožňují pohodlně vytvořit spustitelný JAR se všemi závislostmi bez ručního jednotlivého "přibalování"
- Nástroj **jlink**
- `jlink --module-path "out;C:\Program Files\Java\jdk-9.0.4\jmods" --add-modules com.jenkov.mymodule --output out-standalone`

  `--module-path`

  : složka `jmods` obsahuje standardní Core API moduly a `out` je složka s našimi předkompilovanými moduly

  `--add-modules`

  : které moduly se mají do aplikace sbalit, v tomto případě `com.jenkov.mymodule`

  `--output`

  : cílová složka pro umístění JAR s aplikací

- Spuštění výsledného JAR: `java --module com.jenkov.mymodule/com.jenkov.mymodule.Main`