Architektura CRCE

KIV/SAR - semestrální práce

student: Radek VAIS, Zdeněk VALEŠ
mail: vaisr@students.zcu.cz, valesz@students.zcu.cz
datum: 1.1.2019

1 Zadání

CRCE[1] je komponentové úložiště vyvíjené v rámci výzkumu na Katedře informatiky a výpočetní techniky na Západočeské univerzitě v Plzni. Jeho hlavní vlastností je podpora kontroly kompatibility verzí různých OSGi komponent nebo webových api.

1.1 Motivace

Vývoj projektu probíhá již několik let a vystřídalo se na něm mnoho vývojářů, což způsobilo zanesení větších či menších architektonických dluhů. Projekt již můžeme klasifikovat jako velký. Existuje potřeba, aby střídající se vývojáři (např. v rámci bakalářských prací) snadno a rychle zprovoznili základní instalaci a získali tak prostředí, pro rychlý vývoj nových částí.

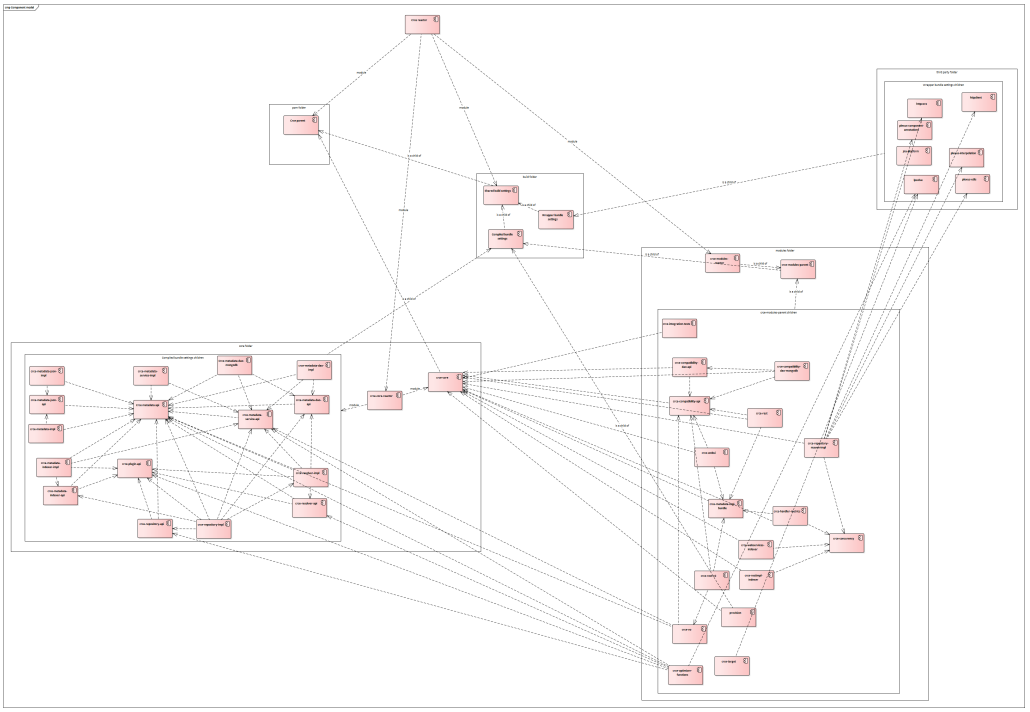
1.2 Cíle projektu

- Unifikovat proces sestavení aplikace a spuštění.
- Zdokumentovat současnou architekturu aplikace.
- Analyzovat závislosti modulů v projektu a ověřit, zda některé moduly mají být součástí jádra.
- Připravit Docker image pro vývojáře.

[1] Component Repository supporting Compatibility Evaluation

2 Analýza architektury

Celý projekt je rozdělen do čtyř hlavních modulů: sdílené nastavení sestavení, jádro, moduly a knihovny třetích stran. Ne zcela jasné je začlenění mechanismu pro nasazení a spuštění v modulu `CRCE modules` (jedná se o komponentu `provision`). Závislostí modulů na jádře pozorujeme dva typy. Prvním typem je závislost na agregačním modulu jádra, druhým je jmenovitá závislost na komponentách jádra. Tyto skutečnosti jsou patrné na celkovém diagramu na Obrázku 1.

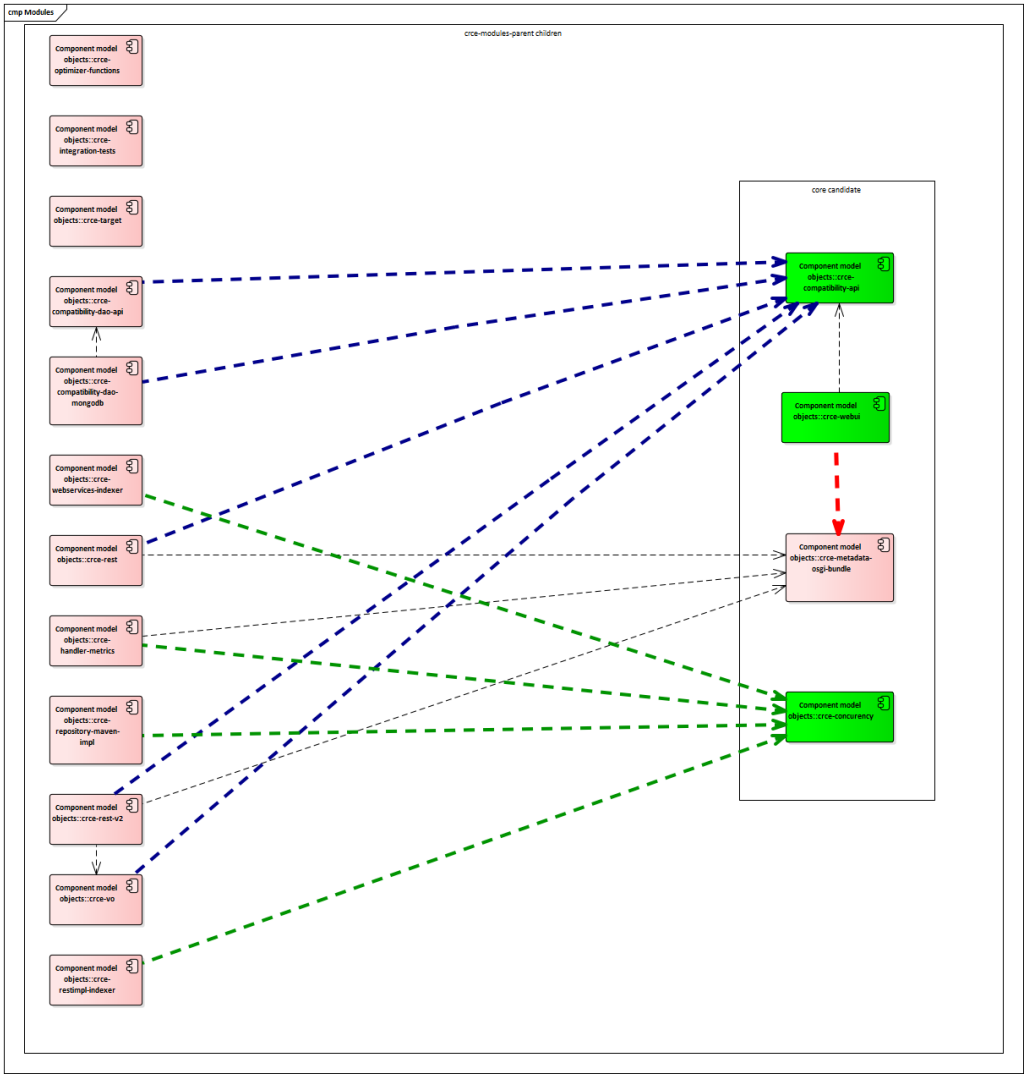


Obrázek 1: Diagram závislostí všech maven komponent projektu CRCE před provedenými úpravami.

Během analýzy modulu pro moduly CRCE jsme identifikovali několik kandidátů na přesun do jádra. Za kandidáty jsme volili takové komponenty, které využívá velká část ostatních modulů a zároveň tyto moduly mají závislosti do jádra. Komponenta webového rozhraní (`crce-webui`) tomuto pravidlu sice odporuje, představuje ale základní komponentu, bez které by úložiště nešlo použít. Přesouvané komponenty jsou: `crce-compatibility-api`, `crce-webui`, `crce-concurrency` (graficky znázorněno na Obrázku 2).

Po identifikaci kandidátů jsme z přesunu vyřadili komponentu metadata-osgi-bundle, kvůli nedostatečné obecnosti. Komponenta je schopna indexo-

vat pouze OSGI komponenty – přesun do jádra by byl možný, pokud by uměla indexovat obecné JAR. Komponenta `crce-webui` bohužel obsahuje silnou závislost na této komponentě, která brání přesunu `crce-webui` do jádra. Závislost spočívá v použití několika konstant, které jsou v definované v metadata-osgi-bundle. Navrhli jsme tedy přidání metody do rozhraní indexeru pro zjištění jaké elementy indexer zpracovává a silnou závislost jsme nahradili slabou závislostí.



Obrázek 2: Diagram původních závislostí uvnitř modulu moduly CRCE. Zeleně jsou označeny přesouvané komponenty. Červeně je označena nevhodná závislost.

3 Provedené změny

V projektu jsme provedli několik změn, které by měly vést k zjednodušení práce budoucích vývojářů. Pro přehlednost dodávám, že v této sekci pojem *modul* značí maven modul v kořenovém adresáři CRCE a pojem *komponenta* značí jeden z maven modulů obsahující konkrétní implementaci (*crce-webui*, *crce-metadata-api*, ...).

3.1 Nový modul pro sestavení

Do kořene CRCE byl přidán nový modul *deploy*, do kterého byla přesunuta komponenta *provision*, používaná ke spouštění CRCE. Jsou zde agregovány všechny komponenty z jádra a komponenty z modulu *modules*, které jsou zachycené v *crce-default-modules*.

Nově přidaná komponenta *crce-default-modules* tedy slouží jako agregátor komponent z modulu *modules*, které mají být použité ve výsledné distribuci.

V rámci přípravy modulu pro sestavení a spuštění byl přidán Docker image obsahující JDK 1.7 a Apache Felix verze 5.0.0.

3.2 Přesun vybraných modulů do jádra

Do jádra byly z modulu *modules* přesunuty komponenty: *crce-compatibility-api*, *crce-webui* a *crce-concurrency*. V případě *crce-webui* bylo nutné odstranit závislost na *crce-core* (jinak by se jednalo o kruhovou závislost) a nahradit ji jednotlivými závislostmi na vybraných komponentách z jádra.

V komponentě *crce-compatibility-api* bylo nutné přidat výchozí (prázdnou) implementaci rozhraní *Compatibility* a *Diff*. Pro tyto implementace byl přidán unit test.

Součástí přesunu komponent do jádra bylo přidání unit testu do *crce-metadata-osgi-bundle*, který dokazuje schopnost této komponenty indexovat pouze OSGI bundly a ne obecné JAR.

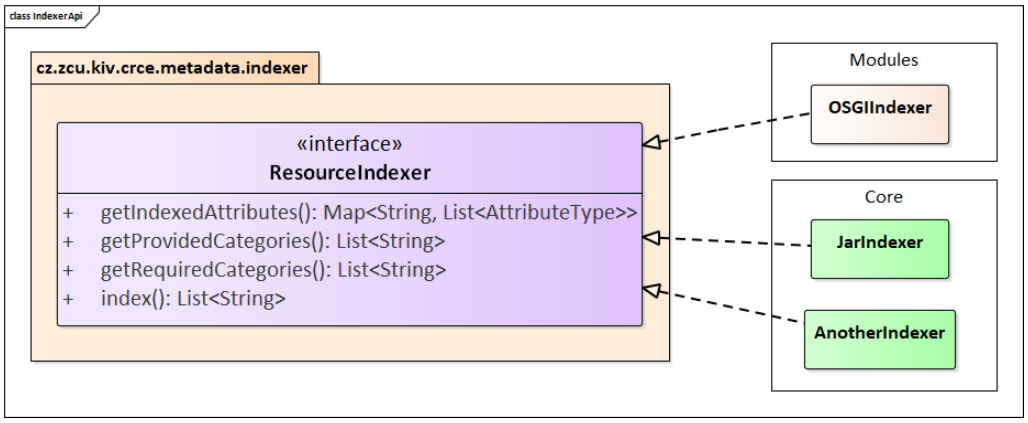
3.2.1 Nahrazení silné závislosti v komponentě crce-webui

Komponenta *crce-webui* obsahovala silnou závislost na komponentě *crce-metadata-osgi-bundle*. Tato závislost spočívala v použití několika konstant, označujících indexované elementy, definovaných v rozhraní *NsOsgiIdentity*. Konstanty byly použity ve třídě *ResourceServlet* k filtrování zobrazovaných OSGI bundlů.

Po dohodě s vyučujícím jsme tuto závislost nahradili slabou závislostí – konstanty stejného typu jsme dodefinovali i v komponentě *crce-webui*. Silná

závislost byla odstraněna, nicméně pokud se v budoucnu změní komponenta *crce-metadata-osgi-bundle*, bude nutné kontrolovat, zda změna neovlivnila *crce-webui* a případné nesrovnalosti opravit.

Aby bylo podobným závislostem v budoucnosti zamezeno, rozhraní indexeru bylo rozšířeno o metodu, která vrací elementy, jež je indexer schopen detekovat. V rámci této práce byla vytvořena defaultní implementace této vlastnosti v AbstractIndexer, která vrací pouze prázdný seznam. Projekt lze bezpečně přeložit, ale momentálně novou funkci rozhraní plně využívá jen OSGiIndexer.



Obrázek 3: Diagram ukazuje upravené rozhraní indexeru, které nově obsahuje metodu getIndexedAttributes, která slouží k získání seznamu vlastností, které daný indexer zpracovává. Dále naznačuje možnost umístit implementaci indexeru do modulu crce-core nebo crce-modules (fiktivní indexery JarIndexer a AnotherIndexer).

3.2.2 Načítání connection stringu

Součástí příprav pro dockerizaci bylo přidání konfigurace připojení k databázi MongoDB. Ke konfiguraci připojení lze nyní využít proměnnou prostředí *mongo_connection*, kde aplikace očekává MongoDB connection string[2]. V případě, že tato proměnná není nastavena, použije se výchozí hodnota (*mongodb://localhost:27017*, která nese připojení k databázi běžící na localhostu a standardním portu. Výhodou tohoto přístupu je možnost konfigurovat connection string v Dockerfile a při změně pouze znovu sestavit image.

[2]viz dokumentace: https://docs.mongodb.com/manual/reference/connection-string/

4 Dockerizace projektu

Do modulu *deploy* byl přidán Dockerfile, kterým je možné sestavit docker image obsahující spustitelnou verzi CRCE. Do takto spuštěného CRCE lze doinstalovávat nové OSGI komponenty a není nutné celý projekt znovu sestavovat.

Detaily o sestavení a spuštění docker image jsou v sekci 5

5 Uživatelská příručka

Provedené změny ovlivnily základní používání projektu při vývoji.

Pro sestavení na platformě Linux lze použít bash script *build.bash*[3], který automaticky spustí jednotlivé kroky sestavení pomocí nástroje Maven. Script vždy vykonává všechny kroky od začátku, v případě selhání je vhodné po opravě pokračovat v buildu manuálně.

V případě, že databáze neběží na localhostu, nebo standardním portu, je možné nastavit připojení skrze proměnnou prostředí *mongo_connection*. Tato proměnná by v případě použití měla obsahovat celý connection string. Pokud není nastavená, pro připojení k databázi bude použita výchozí hodnota: *mongodb://localhost:27017*.

5.1 Docker

K sestavení Docker image je možné použít Dockerfile, který se nachází v modulu *deploy*. Sestavení image je provedeno příkazem:

```
docker build . -t crce-docker
```

5.2 Spuštění docker image

Výsledný image obsahuje distribuci Apache Felix 5.0.0, JDK 1.7 a všechny OSGI bundly z CRCE. Spuštění lze provést příkazem:

```
docker run -it \
   -p 8080:8080 \
   --add-host mongoserver:172.17.0.1 \
   -v /felix/deploy:/felix/felix-framework-5.0.0/dist \
   crce-docker
```

[3]Teoreticky není nutná platforma Linux. Pro uživatele Windows by měl být dostačující bash interpret nebo Linux subsystem. Tato možnost nebyla testována.

Parametry *-it* spustí image s v interaktivním módu a přesměrují o něj STDIN. Tím je možné používat interaktivní GoGo shell. Pokud budou tyto parametry vynechány, GoGo shell nebude schopný detekovat žádný vstup a automaticky vypne běžící Felix.

Parametr *–add-host* slouží k přidání záznamu do /etc/hosts v kontejneru a image se pak může k databázi běžící na hostovacím stroji dostat pomocí hostname *mongoserver*.

IP 172.17.0.1 je adresa, kterou docker přiřazuje hostovacímu stroji. Do adresáře */felix/deploy* na hostovacím stroji je možné vložit OSGI bundly a ty následně nainstalovat a spustit skrze Gogo shell. K běžící instanci je možné přistoupit na adrese *http://localhost:8080/crce*.

Parametr *-v* slouží k propojení adresáře na hostovacím stroj (*/felix/deploy*) s adresářem uvnitř kontejneru (*/felix/felix-framework-5.0.0/dist*). Do tohoto adresáře je možné kopírovat OSGi bundly a instalovat je v runtime.

Před spuštěním docker image je třeba spustit instanci MongoDB. Není podstatné, zda bude spuštěna prostřednictvím Dockeru (např. image 3.4-jessie), lokálním stroji nebo na serveru. Jen je třeba před sestavením ověřit nastavení connection string. Není připravena konfigurace docker-compose, protože se nám v rámci konfigurace a testů nepodařilo zprovoznit GoGo shell při použití compose mechanismu, uživatel by byl ochuzen o diagnostické možnosti OSGi runtime.

5.3 Možné problémy při spuštění

Pokud není spuštěný docker daemon, nebo současný uživatel není ve skupině *docker*, může příkaz `docker run` vyžadovat spuštění pod rootem [4].

V případě, že by IP 172.17.0.1 nefungovala, je potřeba zjistit pod jakou adresou vidí docker hostovací stroj. Ve výchozím nastavení vytvoří docker síťový most `docker0` mezi imagem a hostovacím strojem. Detaily mostu lze zobrazit příkazem:

```
sudo ip addr show docker0
```

IP zobrazená ve výstupu příkazu je adresa, pod kterou docker container vidí hostovací stroj[5].

[4] Kroky po nainstalování dockeru: https://docs.docker.com/install/linux/linux-postinstall/

[5] Více detailů zde: https://stackoverflow.com/questions/24319662/from-inside-of-a-docker-container-how-do-i-connect-to-the-localhost-of-the-mach

6 Závěr

V rámci projektu jsme naplnili cíl usnadnit novým vývojářům spuštění výchozí aplikace, především přípravou docker image. Nepodařilo se nám zjednodušit proces sestavení celé aplikace na jeden Maven příkaz nicméně pro prostředí Linux jsme vytvořili script, který úkony nutné pro sestavení provede automaticky. Dále jsme přesunuli několik modulů do jádra CRCE z důvodu jejich provázanosti s ostatními nebo nutnosti pro používání.