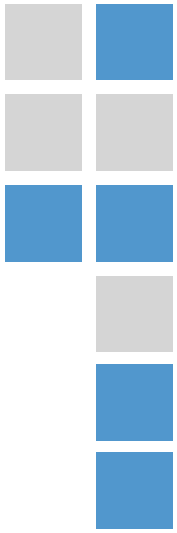

Podpora aplikační logiky v J2EE aplikačních rámcích

Tomáš Pitner, Petr Matulík
CESNET a MU v Brně
tomp@fi.muni.cz

Úvod, kontext

- **Kam se hodí Java2 Enterprise Edition?**
  - robustní, platformově nezávislá řešení podnikových aplikací různého rozsahu
- **Téma příspěvku:**
  - vrstva **aplikační logiky** v J2EE
  - především v **aplikačních rámcích** (frameworks):
    - pro J2EE je k dispozici mnoho a jsou rutinně využívány

Aplikační logika v rámcích

- Rámce podporují moderní programovací techniky a principy podporující aplikační logiku:
  - programování orientované na aspekty (**Aspect-Oriented Programming**, AOP) a
  - obrácení řízení (**Inversion of Control**, IoC)
- Tyto techniky budeme demonstrovat na:
  - jednoduchém rámci (**VRaptor**) a
  - komplexním (**Spring**)

Aplikační logika v OO prostředí

- **Kde? v metodách objektů (komponent)**
- **Jaká?**
  - **funkční, korektní, robustní**, *ale též*
  - **přehledná**,
  - **testovatelná**,
  - **udržovatelná**,
  - **znovupoužitelná**

Řízení aplikační logiky

- Moderní aplikace **odděluje jednotlivé vrstvy** a staví na **modelu MVC**:
  - model
  - view/pohled
  - controller/řadič
- Rámce z pohledu řízení používají:
  - *The Hollywood Principle*: “Do not call us, we will call you!”
  - o řízení toku se stará rámec =
  - dochází k **Inversion of Control**

Deklarativní řízení

- Tok výpočtu je řízen pravidly zachycenými v popisných souborech - obvykle XML.
- Příklad toku výpočtu - webová aplikace:
  - uživatel vyšle HTTP požadavek
  - řadič nasměruje dotaz na příslušný řetěz zpracování:
    - nejprve je získán patřičný **model**,
    - na něm jsou volány **metody aplikační logiky**
    - následně je na výsledek aplikován vhodný **pohled**, který se
  - dostane zpět klientovi.
- Toto řídí aplikační rámec, ne komponenta!

Injektáž závislostí

- **Injektáž závislostí** (Dependency Injection, DI) je jednou z podob IoC.
- Řeší automatizované „napojení“ instancí závisejících komponent.

- Závislosti nemusejí být definovány na úrovni tříd, ale rozhraní (komponenta nezávisí na konkrétní implementaci daného rozhraní.

Jak se DI řešila dříve?

- Dříve:
  - „natvrdo“ v kódu uvedená instanciace určité třídy nebo
  - použit „lookup“ mechanizmus na vyhledání implementace

- *Vše bylo přímo v kódu komponenty*
  - *tzn. žádné obrácení řízení!*

Injektáž závislostí (2)

- **Příklad (blíže viz rámec *Picocontainer*):**
  - **deklaruj rozhraní a komponenty:**

```
public interface Kissable {
    void kiss(Object kisser);
}
public class Boy implements Kissable {
    public void kiss(Object kisser) {
        System.out.println("I was kissed by
  " + kisser);
    }
}
```

Injektáž závislostí (3)

- **(pokračování):**

```
public class Girl {
    Kissable kissable;

    public Girl(Kissable kissable) {
        this.kissable = kissable;
    }
    public void kissSomeone() {
        kissable.kiss(this);
    }
}
```

Injektáž závislostí (4)

- Registruj komponenty v kontejneru:

```
MutablePicoContainer pico
  = new DefaultPicoContainer();

pico.registerComponentImplementation(
  Kissable.class, Boy.class);

pico.registerComponentImplementation(
  Girl.class);
```

Injektáž závislostí (5)

- Použij komponenty:

```
Girl girl = (Girl)
  pico.getComponentInstance(Girl.class);

girl.kissSomeone();
```

Typy injektáže závislostí

- Shrnutí, typy DI:
  - zde použita tzv. **Constructor Injection**
  - rámce jako Spring podporují ještě **Setter Injection** (metody setXXX na nastavení odkazu na komponentu, na níž závisí)
  - pokud je únosné, aby komponenta *implementovala předepsané rozhraní* (dané kontejnerem), lze uvažovat i o **Interface Injection** (vyhrazené metody na nastavení závislostí) - příliš svazuje s konkrétním rámcem

Příklady v konkrétních rámcích

- Rámec VRaptor:
  - jednoduchý rámec pro webové aplikace
  - http://vraptor.org

VRaptor - řízení toku

- Deklarativní popis (`vraptor.xml`) řízení toku:

```
<package name="chain">
  <chain name="productlist">
     <logic
      class="org.vraptor.example.chain.ProductLogic"
      method="listAll"/>
     <view>chain/productList.vm</view>
  </chain>
  <chain name="newproduct">
      <view>chain/productForm.vm</view>
  </chain>
  ...
</package>
```

Díky za pozornost!

dotazy nyní nebo kdykoli později na

<tomp@fi.muni.cz>