Advanced Java technologies: JBoss

Časť 2.

Contexts and Dependency Injection (CDI)

Enterprise JavaBeans 3.1

Jozef Hartinger

December 2012

redhat

Projekt

- https://community.jboss.org/wiki/AdvancedJavaEELabIntensive
- https://github.com/qa/pv243
- Zložka lesson02
- Tagy
  - Počiatočný cdi-00
- Dostupný online
  - http://lesson02-pv243.rhcloud.com

redhat

Úloha 1: CDI Beans

- Vytvorte CDI komponentu, ktorá bude s použitím komponenty **MathOperations** (injection) implementovať rozhranie **Factorial** a bude zdieľaná vrámci celej aplikácie.
- Riešenie overte pomocou testu **FactorialTest**

redhat

Úloha 1: CDI Beans

- Vytvorte CDI komponentu, ktorá bude s použitím komponenty **MathOperations** (injection) implementovať rozhranie **Factorial** a bude zdieľaná vrámci celej aplikácie.
- Riešenie overte pomocou testu **FactorialTest**
- Riešenie v tagu cdi-01

redhat

Úloha 2: View layer integration

- Pomocou JSF a Facelets vytvorte prezentačnú vrstvu pre Factorial komponentu (factorial.xhtml) pozostávajúcu z:
  - Input fieldu pre vstupné číslo
  - Tlačítka “Compute”
  - Tlačítka “Reset”
- Tipy:
  - EL použitý v JSF 2 podporuje parametrizované volanie metód (nie je nutné implementovať controller)
  - Výsledok výpočtu zobrazte pomocou Convertoru
    - <f:converter converterId="bigIntegerByteArray" />

redhat

Úloha 2: View layer integration

- Pomocou JSF a Facelets vytvorte prezentačnú vrstvu pre Factorial komponentu (factorial.xhtml) pozostávajúcu z:
  - Input fieldu pre vstupné číslo
  - Tlačítka “Compute”
  - Tlačítka “Reset”
- Tipy:
  - EL použitý v JSF 2 podporuje parametrizované volanie metód (nie je nutné implementovať controller)
  - Výsledok výpočtu zobrazte pomocou Convertoru
    - <f:converter converterId="bigIntegerByteArray" />
- Riešenie v tagu cdi-02

redhat

Úloha 3: Task parallelization

- Upravte komponentu **MathOperations** a vytvorte alternatívnu implementáciu rozhrania **Factorial** tak, aby výpočet prebiehal v dvoch vláknach (s použitím EJB asynchrónneho volania metód).
- Odlíšte vytvorenú implementáciu od predchádzajúcej s pomocou novo-vytvoreného kvalifikátora (**@Parallel**).
- Upravte **FactorialTest** a **factorial.xhtml** tak aby používali paralelnú implementáciu
- Tipy:
  - **@Asynchronous**
  - **@Future<BigInteger>**
  - **AsyncResult**

redhat

Úloha 3: Task parallelization

- Upravte komponentu **MathOperations** a vytvorte alternatívnu implementáciu rozhrania **Factorial** tak, aby výpočet prebiehal v dvoch vláknach (s použitím EJB asynchrónneho volania metód).
- Odlíšte vytvorenú implementáciu od predchádzajúcej s pomocou novo-vytvoreného kvalifikátora (**@Parallel**).
- Upravte **FactorialTest** a **factorial.xhtml** tak aby používali paralelnú implementáciu
- Tipy:
  - **@Asynchronous**
  - **@Future<BigInteger>**
  - **AsyncResult**
- Riešenie v tagu cdi-03

redhat

Úloha 4: Events

- Rozšírte komponentu ParallelFactorial tak, aby po každom dokončení výpočtu vyvolala udalosť **FactorialComputationFinished**
- Funkčnosť overte pomocou **FactorialTest.testEvent()** prípadne pomocou novo-vytvorenej komponenty, ktorá reaguje na udalosť a vypíše výsledok výpočtu na štandardný výstup
- Tipy:
  - **Event**
  - **@Observes**

redhat

Úloha 4: Events

- Rozšírte komponentu ParallelFactorial tak, aby po každom dokončení výpočtu vyvolala udalosť **FactorialComputationFinished**
- Funkčnosť overte pomocou **FactorialTest.testEvent()** prípadne pomocou novo-vytvorenej komponenty, ktorá reaguje na udalosť a vypíše výsledok výpočtu na štandardný výstup
- Tipy:
  - **Event**
  - **@Observes**
- Riešenie v tagu cdi-04

redhat

Úloha 5: Singletons

- Vytvorte komponentu zdieľanú vrámci celej aplikácie ktorá bude reagovať na udalosť **FactorialComputationFinished** a bude ukladať dvojice (číslo, číslo!).
- Komponenta bude poskytovať operácie:
  - **get(long number) : BigInteger**
  - **clear() : void**
- Nebude použitá thread-safe dátová štruktúra. Namiesto toho bude prístup k dátovej štruktúre chránený s pomocou read/write zámku (EJB)
- Tipy:
  - @Singleton
  - @Lock

redhat

Úloha 5: Singletons

- Vytvorte komponentu zdieľanú vrámci celej aplikácie ktorá bude reagovať na udalosť **FactorialComputationFinished** a bude ukladať dvojice (číslo, číslo!).
- Komponenta bude poskytovať operácie:
  - **get(long number) : BigInteger**
  - **clear() : void**
- Nebude použitá thread-safe dátová štruktúra. Namiesto toho bude prístup k dátovej štruktúre chránený s pomocou read/write zámku (EJB)
- Tipy:
  - @Singleton
  - @Lock
- Riešenie v tagu cdi-05

redhat

Úloha 6: Decorators

- Naimplementujte dekorator ktory bude obaľovať volanie **compute()** obidvoch implementácii **Factorial** rozhrania a bude vracať výsledky z cache v prípade, že budú dostupné. V opačnom prípade bude výpočet delegovaný na pôvodnú implementáciu.
- Tipy:
  - @Decorator
  - @Delegate
  - beans.xml

redhat

Úloha 6: Decorators

- Naimplementujte dekorator ktory bude obaľovať volanie **compute()** obidvoch implementácii **Factorial** rozhrania a bude vracať výsledky z cache v prípade, že budú dostupné. V opačnom prípade bude výpočet delegovaný na pôvodnú implementáciu.
- Tipy:
  - @Decorator
  - @Delegate
  - beans.xml
- Riešenie v tagu cdi-06

redhat

Úloha 7: Conversations

- Zoznámte sa s triedami v balíku **cz.muni.fi.pv243.lesson02.students** a preskúmajte akým spôsobom je sú použité konverzácie na realizáciu procesu registrácie študentov.

redhat

That's it!

- https://community.jboss.org/wiki/AdvancedJavaEELabIntensive
- https://github.com/qa/pv243
- http://lesson02-pv243.rhcloud.com

jharting@redhat.com | www.redhat.com