



Advanced Java technologies: JBoss

Časť 2.

Contexts and Dependency Injection (CDI)

Enterprise JavaBeans 3.1

Jozef Hartinger

11.3.2013

Projekt

- <https://community.jboss.org/wiki/AdvancedJavaEELabjaro2013>
- <https://github.com/qa/pv243>
- Zložka lesson02
- Tagy
 - Počiatočný cdi-00
- Dostupný online
 - <http://lesson02-pv243.rhcloud.com>

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**

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

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" />`

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

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**

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

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`**

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`

Úloha 5: Singletons

- Vytvorte komponentu zdieľanú v rámci 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

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

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

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

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.

That's it!

- <https://community.jboss.org/wiki/AdvancedJavaEELabjaro2013>
- <https://github.com/qa/pv243>
- <http://lesson02-pv243.rhcloud.com>



Questions?

jharting@redhat.com | www.redhat.com