Advanced Java technologies: JBoss

Část 2.

Contexts and Dependency Injection (CDI)

Enterprise JavaBeans

March 2016

redhat

Projekt

- https://developer.jboss.org/wiki/AdvancedJavaEELabFIMUNIJaro2016
- git clone git@github.com:qa/pv243-a4m36jee-2016-cdi-seminar.git
- Branches
  - Počáteční cdi-00

redhat

Úloha 1: CDI Beans

- Vytvořte CDI komponentu, která bude s použitím komponenty **MathOperations** (injection) implementovat rozhraní **Factorial** a bude sdílená v rámci celé aplikace.
- Řešení ověřte pomocí testu **FactorialTest**
  - **Arquillian**

redhat

Úloha 1: CDI Beans

- Vytvořte CDI komponentu, která bude s použitím komponenty **MathOperations** (injection) implementovat rozhraní **Factorial** a bude sdílená v rámci celé aplikace.
- Řešení ověřte pomocí testu **FactorialTest**
- Řešení v branch cdi-01

redhat

Úloha 2: View layer integration

- Propojte Factorial komponentu s view vrstvou
  - Doimplementujte třídu FactorialForm
    - name
    - scope
    - compute()

redhat

Úloha 2: View layer integration

- Propojte Factorial komponentu s view vrstvou
  - Doimplementujte třídu FactorialForm
    - name
    - scope
    - compute()
- Řešení v branch cdi-02

redhat

Úloha 3: Task parallelization

- Upravte komponentu **MathOperations** a vytvořte alternativu implementace rozhraní **Factorial** tak, aby výpočet probíhal v dvou vláknech (s použitím EJB asynchronního volaní metod).
- Odlište vytvořenou implementaci od předcházející s pomocí nově-vytvořeného kvalifikátoru (**@Parallel**).
- Upravte **FactorialTest** a **factorial.xhtml** tak aby používali paralelní implementaci
- Tipy:
  - **@Asynchronous**
  - **Future<BigInteger>**
  - **AsyncResult**

redhat

Úloha 3: Task parallelization

- Upravte komponentu **MathOperations** a vytvořte alternativu implementace rozhraní **Factorial** tak, aby výpočet probíhal v dvou vláknech (s použitím EJB asynchronního volaní metod).
- Odlište vytvořenou implementaci od předcházející s pomocí nově-vytvořeného kvalifikátoru (**@Parallel**).
- Upravte **FactorialTest** a **factorial.xhtml** tak aby používali paralelní implementaci
- Tipy:
  - **@Asynchronous**
  - **Future<BigInteger>**
  - **AsyncResult**
- Řešení v branch cdi-03

redhat

Úloha 4: Events

- Rozšiřte komponentu ParallelFactorial tak, aby po každém dokončení výpočtu vyvolala událost **FactorialComputationFinished**
- Funkčnost ověřte pomocí **FactorialTest.testEvent()** případně pomocí nově-vytvořené komponenty, která reaguje na událost a vypíše výsledek výpočtu na standartní výstup
- Tipy:
  - **Event<FactorialComputationFinished>**
  - **@Observes**

redhat

Úloha 4: Events

- Rozšiřte komponentu ParallelFactorial tak, aby po každém dokončení výpočtu vyvolala událost **FactorialComputationFinished**
- Funkčnost ověřte pomocí **FactorialTest.testEvent()** případně pomocí nově-vytvořené komponenty, která reaguje na událost a vypíše výsledek výpočtu na standartní výstup
- Tipy:
  - **Event<FactorialComputationFinished>**
  - **@Observes**
- Řešení v branch cdi-04

redhat

Úloha 5: Singletons

- Vytvořte komponentu sdílenou v rámci celé aplikace která bude reagovat na událost **FactorialComputationFinished** a bude ukládat dvojice (číslo, číslo!).
- Komponenta bude poskytovat operace:
  - **get(long number) : BigInteger**
  - **clear() : void**
- Nebude použitá thread-safe datová struktúra. Namísto toho bude přístup k datové struktúře chráněný s pomocí read/write zámku (EJB)
- Tipy:
  - @Singleton
  - @Lock

redhat

Úloha 5: Singletons

- Vytvořte komponentu sdílenou v rámci celé aplikace která bude reagovat na událost **FactorialComputationFinished** a bude ukládat dvojice (číslo, číslo!).
- Komponenta bude poskytovat operace:
  - **get(long number) : BigInteger**
  - **clear() : void**
- Nebude použitá thread-safe datová struktúra. Namísto toho bude přístup k datové struktúře chráněný s pomocí read/write zámku (EJB)
- Tipy:
  - @Singleton
  - @Lock
- Řešení v branch cdi-05

redhat

Úloha 6: Decorators

- Naimplementujte dekorátor který bude obalovat volaní **compute()** obou dvou implementací **Factorial** rozhraní a bude vracet výsledky z cache v případě, že budou dostupné. V opačném případě bude výpočet delegovaný na původní implementaci.
- Tipy:
  - @Decorator
  - @Dependent
  - @Delegate
  - @Priority

redhat

Úloha 6: Decorators

- Naimplementujte dekorátor který bude obalovat volaní **compute()** obou dvou implementací **Factorial** rozhraní a bude vracet výsledky z cache v případě, že budou dostupné. V opačném případě bude výpočet delegovaný na původní implementaci.
- Tipy:
  - @Decorator
  - @Dependent
  - @Delegate
  - @Priority
- Řešení v branchi cdi-06

redhat

Úloha 7: Conversations

- Seznamte se s třídami v balíčku **cz.muni.fi.pv243.lesson02.students** a prozkoumejte jakým způsobem jsou použité konverzace na realizaci procesu registrace studentů.

jpechane@redhat.com | www.redhat.com