PIISW, W08, IO, 2017/2018, semestr letni
Lista zadań nr 4

Michał Luzyna

26 marca 2018

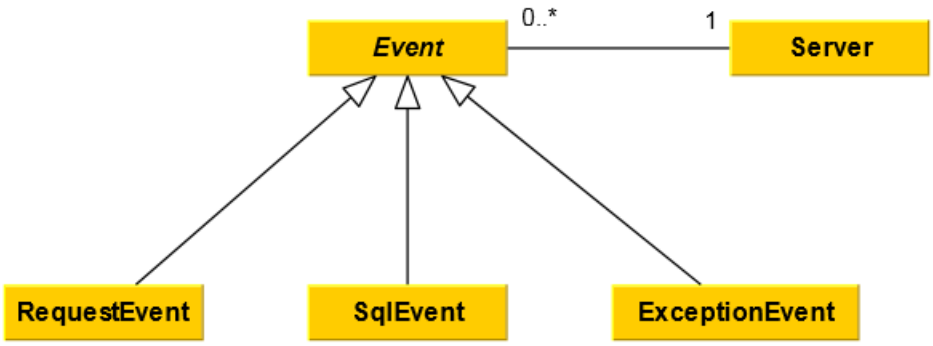
Wprowadzenie

Posiadamy system monitorowania zdarzeń na serwerach. Informacje o zdarzeniach muszą być zapisywane w relacyjnej bazie danych.

W celu realizacji zadania należy zapoznać się z dokumentacją:

- Spring data JPA
- JPQL reference

1. Utwórz nowe prywatne repozytorium o nazwie `imie-nazwisko-lista4` (w ramach organizacji `pwr-piisw`) oraz zaimportuj do niego zawartość repozytorium `https://github.com/pwr-piisw/jpa-starter`. Projekt zawarty w repozytorium został zbudowany w oparciu o Spring Boot i wykorzystuje JPA oraz bazę danych H2.
2. Skonfiguruj `travis-ci` dla tego repozytorium.
3. Zapoznaj się ze strukturą klas zaprezentowanych na diagramie z rysunku 1. Klasy zdarzeń (relacja dziedziczenia) zostały zmapowane z wykorzystaniem metody `TABLE_PER_CLASS`.
4. Zapoznaj się z mapowaniem relacji `Event` → `Server`.
5. Baza danych tworzona jest w momencie startu aplikacji/testów automatycznie dzięki ustawieniu w pliku `application.properties`: `spring.jpa.hibernate.ddl-auto=create`
6. Zapoznaj się ze skryptem `data-h2.sql` – skrypt ten jest uruchamiany przy starcie testów i ma za zadanie załadowanie danych testowych do bazy.
7. Zapoznaj się klasami:
   (a) `ServerService` – interfejs usług dotyczących klasy `Server`
   (b) `ServerServiceImpl` – implementacja `ServerService`
   (c) `ServerRepository` – interfejs DAO dostępu do danych klasy `Server`. Zauważ, że interfejs ten nie posiada implementacji. Biblioteka `spring-data` automatycznie generuje kod odpowiedzialny za komunikację z bazą danych.
   (d) `ServerServiceTest` – demonstruje działanie klas
8. Każde zadanie musi posiadać implementację w teście o nazwie `TaskX` gdzie `X` to numer zadania. Dla każdego zadania został przygotowany szablon testu.



Rysunek 1: Diagram klas

.

Oceny

Punkty:	< 9	$9 - 10$	$11 - 12$	$13 - 14$	$15 - 16$	$17 - 18$
Ocena:	$2,0$	$3,0$	$3,5$	$4,0$	$4,5$	$5,0$

Zadania

1. (2 pkt) Zmodyfikuj klasę Server, dodaj *Optimistic Locking* oraz zaimplementuj aktualizację pola `lastUpdateDate` o aktualną datę przy każdym zapisie aktualizacji obiektu klasy.

   Wskazówka: wykorzystaj anotacje `javax.persistence.Version` oraz zaimplementuj *listener* z wykorzystaniem adnotacji `javax.persistence.PreUpdate`.

2. (3 pkt) Utwórz interfejs `EventRepository` rozszerzający `org.springframework.data.jpa.repository.JpaRepository`. Zadeklaruj metodę która dla zadanych parametrów `LocalDateTime start`, `LocalDateTime end`, `boolean toBeAnalyzed` zwróci wszystkie zdarzenia `Event` które posiadają czas zarejestrowania time taki że `start` $<$ `Event.time` $<$ `end` oraz flaga `Event.analysisRequired` $=$ `toBeAnalyzed`.

   Wskazówka: wystarczy nazwać metodę zgodnie z konwencją `findBy`, `Between`, `And`.

3. (2 pkt) Stronicowanie – polega na przekazywaniu danych w porcjach tzw. Stronach. Zadeklaruj w `EventRepository` metodę umożliwiającą pobieranie danych z wykorzystaniem stronicowania.

   Wskazówka: utwórz metodę o nazwie `findAll`, wykorzystaj `Page` oraz `Pageable`.

4. (3 pkt) Usuwanie danych można zrealizować poprzez pobranie obiektów encji oraz wywołanie dla nich metody `JpaRepository.delete`, istnieje jednak możliwość usuwania danych za pomocą tzw. *Bulk delete*. Zadeklaruj w `EventRepository` metodę usuwającą wszystkie zdarzenia `Event` gdzie `Event.time` $<$ `X`. Zadeklaruj `X` jako *named parameter*.

   Wskazówka: konieczne jest wykorzystanie adnotacji `@Modifying`, `@Query`, `@Param` (z pakietu `org.springframework.data`)

5. (3pkt) Modyfikację danych można zrealizować poprzez pobranie obiektów encji, a następnie modyfikacji ich stanu, istnieje jednak możliwość modyfikacji danych za pomocą tzw. *Bulk update*. Zadeklaruj w `EventRepository` metodę modyfikującą wszystkie zdarzenia

`Event` określonej podklasy w ten sposób, że atrybut `toBeAnalyzed` przyjmie wartość `'T'`, dla wszystkich zdarzeń spełniających warunek: `Event.duration` > `X`. Metoda powinna przyjmować następujące argumenty:

- `Class<? extends Event> clazz`
- `int minDuration`

Wskazówka: konieczne jest wykorzystanie adnotacji `@Modifying`, `@Query`, `@Param` (z pakietu `org.springframework.data`).

6. (2 pkt) Zadeklaruj w `EventRepository` metodę zwracającą listę obiektów typu `ServerStatistic` — ile zdarzeń zostało zarejestrowanych na poszczególnych serwerach.

   Wskazówka: konieczne jest wykorzystanie anotacji `@Query` (z pakietu `org.springframework.data`), tzw. JPQL Constructor Expressions oraz klauzuli `group by`.

7. (3 pkt) Utwórz test zwracający mock zamiast rzeczywistego obiektu. Wykorzystaj w tym celu `ServerService.findByName` oraz `ServerRepository`.

   Wskazówka: zadeklaruj w teście ServerRepository jako `@MockBean`, skorzystaj z `Mockito.when`, `Mockito.eq` oraz `Mockito.thenReturn`.