TESTY JEDNOSTKOWE I INTEGRACYJNE

Termin oddania zadań: 6 stycznia 2017, godz. 23:59

ZADANIE 6

Twoim zadaniem jest napisanie testów jednostkowych i integracyjnych do zadania 4 (Unit of Work). W tym celu użyj biblioteki jUnit (http://junit.org) i Mockito (http://mockito.org).

Pisząc testy jednostkowe pamiętaj o zasadach opisanych przez Uncle Boba w „Clean Code" i dobrych praktykach przedstawionych na wykładzie.

Testy jednostkowe powinny testować tylko tę klasę, do której testy piszemy. Niedopuszczalne są interakcje z innymi klasami – powinny one być zamockowane przy użyciu Mockito.

Pamiętaj o przetestowaniu wszystkich ścieżek w każdej metodzie – zarówno przypadki pozytywne jak i negatywne. W szczególności, dla wybranego repozytorium określ i napisz testy jednostkowe:

- dla metody add w przypadkach, gdy argument jest poprawny, argumentem jest null,
- dla metod update i remove w przypadkach, gdy argument jest poprawny, gdy argument jest obiektem, którego nie ma w bazie oraz gdy argumentem jest null.

Analogiczne testy (dla poprawnych i niepoprawnych argumentów z uwzględnieniem wartości null) określ i napisz dla pozostałych metod: withId, allOnPage, persistAdd, persistDelete, persistUpdate dla wybranego repozytorium.

Przetestuj również metody z klasy HsqlUnitOfWork:

- markAsNew dla przypadków, gdy Entity przekazane jako argument jest poprawne, istniało już w bazie, jest wartością null oraz przekazane UnitOfWorkRepository jest poprawne, jest wartością null, jest repozytorium dla innego typu Entity (np. HsqlOrderRepository dla Entity typu Address),
- markAsDeleted i markAsChanged dla przypadków, gdy Entity przekazane jako argument jest poprawne, istniało już w bazie, jest wartością null oraz przekazane UnitOfWorkRepository jest poprawne, jest wartością null, jest repozytorium dla innego typu Entity (np. HsqlOrderRepository dla Entity typu Address).

Testy integracyjne powinny testować interfejs używany przez użytkownika, więc możemy przyjąć, że chcemy przetestować metody z klasy HsqlUnitOfWork:

- saveChanges – w przypadku, gdy nie wykonaliśmy żadnej operacji (markAsNew, markAsDeleted, markAsChanged), wykonaliśmy najpierw błędną, a potem poprawną (np. update nieistniejącego obiektu w bazie a następnie zapis obiektu), wykonaliśmy dwie poprawne operacje,
- undo – w połączeniu z testami dla poprzedniej metody dla tych samych przypadków (np. wykonaliśmy poprawną operacje, undo, kolejną poprawną operację a następnie saveChanges)
- markAsNew, markAsDeleted, markAsChanged – podobne przypadki jak dla poprzednich metod w połączeniu z użyciem tych metod (undo, saveChanges) w różnych kolejnościach operacji (np. najpierw markAsDeleted a potem markAsChanged, najpierw markAsChanged a potem markAsDeleted itd.)

W przypadku testów integracyjnych nie mockujemy żadnych klas – zależy nam na interakcjach z innymi klasami – dzięki temu możemy pisać testy tylko dla metod z klasy HsqlUnitOfWork.

Przydatne linki:

- http://fruzenshtein.com/junit-and-mockito/ - jak używać Mockito z jUnit.