Programowanie Aplikacji Użytkowych: lab08 – Hibernate

Materiały

1. [Hibernate ORM website](https://hibernate.org/orm/documentation/5.4/)

2. [Hibernate ORM starting guide](https://docs.jboss.org/hibernate/orm/5.4/quickstart/html_single/)

3. [Hibernate ORM docs](https://docs.jboss.org/hibernate/orm/5.4/userguide/html_single/Hibernate_User_Guide.html)

Zadania

1. Stwórz bazę danych dla salonów, wykorzystując opis każdej z encji z lab05
    - Możliwe są dwa sposoby implementacji:
        a. Stworzenie konfiguracji w aplikacji za pomocą XML lub adnotacji i wygenerowanie bazy danych na tej podstawie.
        b. Zaimplementowanie bazy za pomocą SQL i wygenerowanie konfiguracji klas encyjnych.
    - Salon zawiera listę produktów. Widać zatem, że pomiędzy tymi encjami występuje relacja 1:n. Należy to uwzględnić w [odpowiedni dla ORM sposób] (https://docs.jboss.org/hibernate/orm/5.4/userguide/html_single/Hibernate_User_Guide.html#domain-model).
    - Relacje mogą być dwustronne, tj. pojazd będzie zawierał informację o salonie, w którym się znajduje. Jeżeli to konieczne, wykorzystaj ten mechanizm.
    - Kod odpowiedzialny za obsługę bazy powinien być odseparowany od reszty aplikacji. Dobrym miejscem na wywoływanie procedur związanych z BD będzie kontroler.
    - Long story short: Przechowywanie i pobieranie danych powinno być w bazie. Stwórz odpowiednie tabele w bazie oraz w kodzie źródłowym aplikacji stosowne klasy typu _Entity_.

2. Dodaj nową encję _Rating_, jako ocenę wystawioną produktowi lub salonowi. Encja powinna składać się z:
    a. Wartości oceny w skali 0-5.
    b. Produktu/salonowi, któremu wystawiona została ta ocena.
    c. Data wystawienia oceny.
    d. Opcjonalny (not null) opis produktu.
    e. Należy pamiętać również o odpowiednim kluczy głównym.
3. Zmodyfikuj metody tak, by odwoływały się do bazy danych a nie do utworzonej statycznie listy.
    - Metody filtrujące/przeszukujące powinny szukać odpowiedniego obiektu w bazie i zwracać wynik lub zbiór wyników.
    - Metody modyfikujące stan obiektów (dodawanie, modyfikacja, usuwanie) powinny wykorzystywać odpowiednie do tego metody z [session/entityManager] (https://docs.jboss.org/hibernate/orm/5.4/userguide/html_single/Hibernate_User_Guide.html#pc).

- Metody, które nie pracują na danych, mogą pozostać bez zmian.
- Long story short: Dokonaj refaktoryzacji kodu źródłowego aplikacji w taki sposób, by dane przechowywane i pobierane były z bazy. **Obsłuż wyrzucane wyjątki na poziomie UI**.

4. Zmień kod eksportujący dane do pliku CSV w taki sposób, aby dane były pobierane z bazy danych. Wykorzystaj w tym celu zapytania HQL.
5. Znajdź zastosowanie i wykorzystaj obiekt [**Criteria**] (https://docs.jboss.org/hibernate/orm/5.4/userguide/html_single/Hibernate_User_Guide.html#criteria)
   Podpowiedź: skorzystaj z grupowania (GROUP BY w SQLu). Wartość wyświetl w UI:
   - W tabeli
   - Jako dowolną kontrolkę, której wartość zmienia się po wybraniu magazynu.
6. Wyświetlaj w tabeli z samochodami/salonami liczbę wystawionych ocen i średnią z nich dla każdego z produktów (np. średnia ocen dla Mazdy MX5). Wylicz ją podobnie jak w zadaniu 5. Grupowanie wykonaj po nazwie.

Uwagi

1. Do implementacji obsługi bazodanowej można wykorzystać [DAO pattern] (Data Access Object):.
   - (https://en.wikipedia.org/wiki/Data_access_object)
   - Więcej informacji (https://stackoverflow.com/questions/19154202/data-access-object-dao-in-java).
   - Warto przejrzeć też (https://www.baeldung.com/java-dao-pattern)
   - Oraz koncepcję tego wzorca (https://www.geeksforgeeks.org/data-access-object-pattern/).
2. Dużym uproszczeniem zarządzania zależnościami będzie wykorzystanie menedżera automatyzacji budowania projektu [Apache Maven] (https://www.apache.org/).

Jest on domyślnie wbudowany w IntelliJ, więc nie ma konieczności jego dodatkowej instalacji i konfiguracji w systemie. Więcej informacji w źródłach:

- https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html
- https://www.tutorialspoint.com/maven/index.htm
- https://www.jetbrains.com/help/idea/maven-support.html

3. Obiekty encyjne (Entity) mogą się różnić od tych wyświetlanych. Ten problem można rozwiązać na różne sposoby, np.:
   - Dodając dodatkowe pola w klasie encyjnej, nie oznaczając ich adnotacjami lub w configu xml. Wówczas nie będą one uczestniczyć w operacjach bazodanowych.
   - Stworzenie nowego typu, opakowującego obiekt encyjny, nadając mu nowe wartości.

4. Powyższy punkt dotyczy wzorców projektowych. Kolejne ćwiczenie będzie związane ze Springiem, więc mogę w tym miejscu podać [książkę](https://www.amazon.in/Spring-Design-Patterns-Dinesh-Rajput/dp/1788299450/) jako dobre źródło informacji na ten temat. Kody źródłowe z tejże książki można znaleźć na [GitHubie](https://github.com/PacktPublishing/Spring5-Design-Patterns).

Przykładowe pytania teoretyczne

1. ORM vs plain SQL
2. Java Persistance API (JPA) vs Hibernate Native API.
3. SQL vs HQL
4. Typy relacji w bazach SQL i sposoby ich realizacji w Hibernate
5. Czym są transakcje w bazach danych?
6. Co oznacza adnotacja @Cascade?
7. Do czego służy sesja i SessionFactory?
8. Do czego służy obiekt Criteria?
9. Czym jest SQL Injection?