PAOiM: lab04 – Hibernate

Zadania

1. Stwórz bazę danych dla stadnin, 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.
    - Stadnina zawiera listę koni. 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. koń będzie zawierał informację o stadninie, 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. Zaimplementuj nowe funkcjonalności:
    - Wykorzystując **serializację**, napisz funkcjonalność pozwalającą na zapisanie i odczytanie stanu stadnin oraz zwierząt do pliku binarnego.
    - Wykorzystując wbudowany **pakiet wejścia/wyjścia** (https://docs.oracle.com/javase/7/docs/api/java/io/package-summary.html)), napisz funkcjonalność eksportującą oraz importującą stan wybranej stadniny i listy zwierząt do/z pliku CSV.
    - Zastanów się gdzie i jak można wykorzystać **refleksję i adnotacje**.
3. Dodaj nową encję _**Rating**_, jako ocenę wystawioną dla konia. Encja powinna składać się z:
    a. Wartości oceny w skali 0-5.
    b. koń, któremu wystawiona została ta ocena.
    c. Data wystawienia oceny.
    d. Opcjonalny (not null) opis oceny.
    e. Należy pamiętać również o odpowiednim kluczy głównym.

4. 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**.
5. Zmień kod eksportujący dane do pliku **CSV** w taki sposób, aby dane były pobierane z bazy danych. Wykorzystaj w tym celu zapytania **HQL**.
6. 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.
7. Wyświetlaj w tabeli z końmi liczbę wystawionych ocen i średnią z nich dla każdego ze zwierząt. Grupowanie wykonaj po nazwie.

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)
4. Funkcjonalność serializacji i zapisu do pliku jest _niezależna_ od UI, gdyż dotyczy wyłącznie obiektów modelu i działania logiki biznesowej. Wasza implementacja powinna być więc przenaszalna.
5. Proszę zwrócić uwagę na standard CSV (ang. _comma separated values_). Więcej informacji na [Wiki] (https://en.wikipedia.org/wiki/Comma-separated_values).

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 lab 4

1. Rola adnotacji w językach obiektowych oraz działanie _Retention policies_ na przykładzie Javy.
2. W jaki sposób wykonywana jest serializacja? Jak ją personalizować? Gdzie można ją wykorzystać?
3. ORM vs plain SQL
4. Java Persistance API (JPA) vs Hibernate Native API.
5. SQL vs HQL
6. Typy relacji w bazach SQL i sposoby ich realizacji w Hibernate
7. Czym są transakcje w bazach danych?
8. Co oznacza adnotacja @Cascade?
9. Do czego służy sesja i SessionFactory?
10. Do czego służy obiekt Criteria?
11. Czym jest SQL Injection?