Politechnika Świętokrzyska
Wydział Elektrotechniki, Automatyki i Informatyki

Zespół:
Prusicki Jakub

Narzędzia odwzorowania mapowania obiektowo - relacyjnego

Projekt zespołowy
na studiach stacjonarnych
o kierunku **Informatyka**
Stopień **II**

Opiekun projektu:
dr inż. Mariusz Bedla

Kielce, 2022

SPIS TREŚCI

1. Wstęp teoretyczny.

1.1. Adnotacje.

Adnotacje w języku Java to mechanizm metadanych, który pozwala programistom dodawać informacje o programie do kodu źródłowego. Adnotacje są zazwyczaj określane przez symbol “@”, po którym następuje nazwa adnotacji. Adnotacje mogą być stosowane do klas, metod, zmiennych, parametrów i pakietów. Adnotacje są konstrukcją, która pozwala na przekazywanie dodatkowych informacji na temat kodu. Informacje te mogą być wykorzystane później w kilku miejscach. Mówi się, że adnotacje służą do przekazywania metadanych. Innymi słowy przekazują one dane o kodzie źródłowym.

Wraz z Java 1.5 zostały wprowadzone adnotacje. Adnotacjami nazywamy dodatkowe informacje (metadane) umieszczone w kodzie programu źródłowego, które nie wpływają na działanie programu. Wielu początkujących programistów nie jest w stanie dostrzec ich zastosowania. Dopiero po zapoznaniu się z nimi i dłuższym programowaniu widać ich sens. Mam nadzieję, że po przeczytaniu tego artykułu ujrzysz możliwe zastosowania i potencjał ich wykorzystania. Możliwe wykorzystania:

- Metaprogramowanie
- Tworzenie własnych wskazówek
- Sterowanie komunikatami generowanymi przez kompilator

@Table

Adnotacja @Table pozwala na zmianę nazwy odwzorowanej tabeli. Domyślnie nazwa jest taka sama jak nazwa klasy. W Java dobrą praktyką jest zapisywanie nazwy klasy w liczbie pojedynczej z wielkiej litery. Natomiast w bazie danych tabele zapisuje się małymi znakami w liczbie mnogiej. Aby było to możliwe wykorzystujemy adnotacje @Table.

```
1  @Entity
2  @Table(name = "books")
3  public class Book {
4
5  }
```

Rys. 1.1 Przykład adnotacji @Table.

@Column

Analogicznie do @Table działa adnotacja @Column – jednak tyczy się ona kolumn. Konwencją w Java jest tworzenie nazw zmiennych zapisywanych notacją camelCase. W bazie danych nazwy zmiennych wielowyrazowe rozdzielane są znakiem podkreślenia/podłogi -> „_”.

```java
@Entity
@Table(name = "books")
public class Book {
    @Column(name = "publish_date")
    private LocalDate publishDate;
}
```

Rys. 1.2 Przykład adnotacji @Column.

@Enumerated

Domyślnie typ enum jest zapisywany w bazie danych w postaci liczbowej – według kolejności wartości zapisanej w tym typie. Adnotacja *@Enumerated* pozwala tym sterować. Możemy wówczas ustawić, że wartości będą zapisywane w postaci ciągu tekstowego.

```java
@Entity
@Table(name = "books")
public class Book {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String title;
    private String author;
    @Column(name = "publish_date")
    private LocalDate publishDate;

    @Enumerated(EnumType.ORDINAL) // default
    private BookType bookType;

    // geters + setters
}
```

Rys. 1.3 Przykład adnotacji @Enumerated.

Oczywiście trzeba poważnie rozważyć czy stosowanie typu enum w postaci łańcucha znaków w bazie danych ma sens. Wszystko zależy od potrzeb systemu. Trzeba pamiętać, że zapisywanie tekstowej wartości typu wyliczeniowego do kolumny zamiast liczby wiąże się ze znacznie zwiększona ilością wykorzystywanej przestrzeni dyskowej przez bazę danych. Zwłaszcza jeśli mamy w tabeli dużo rekordów[1].

1.2. ORM.

ORM to akronim od *Object-Relational Mapping*, a narzędzia ORM pozwalają na mapowanie obiektów w języku programowania na odpowiednie zapytania SQL i operacje na bazach danych. Oto kilka popularnych narzędzi ORM:

1. ***Hibernate*** - popularne narzędzie ORM dla języka Java, które oferuje obsługę różnych baz danych i ułatwia operacje na bazach danych.

2. ***Entity Framework*** - narzędzie ORM dla platformy .NET, które umożliwia pracę z bazami danych SQL Server, MySQL, Oracle, PostgreSQL i inne.

3. ***Django ORM*** - narzędzie ORM dla frameworka webowego Django w języku Python, które ułatwia interakcję z bazami danych SQL.

4. ***Sequelize*** - narzędzie ORM dla języka JavaScript, które umożliwia łatwe mapowanie obiektów na relacyjne bazy danych w środowisku Node.js.

5. ***SQLAlchemy*** - narzędzie ORM dla języka Python, które oferuje wiele zaawansowanych funkcji, takich jak automatyczne tworzenie tabel, obsługę transakcji i wiele innych frameworków i bibliotek ,które można wykorzystać w zależności od języka programowania i wymagań projektowych.

Myśląc o technologii ORM jak o pewnego rodzaju pomoście już nie tylko pomiędzy kolejnymi warstwami aplikacji, lecz także między aplikacją a bazą danych, można dodać do wniosku, że skoro istnieje interfejs, to powinna nastąpić znaczna redukcja kodu. Jak najbardziej można się zgodzić z tak postawionym sformułowaniem.

Zastosowanie technologii ORM, gwarantuje implementację podstawowych operacji, takich jak: dodawanie, odczyt, modyfikacja, usuwanie nazywanymi operacjami

typu CRUD (ang. create, read, update, delete). Bez wątpienia fakt ten wpływa na przyspieszenie procedury tworzenia kodu przez programistę, zwalniając go tym samym z obowiązku tworzenia procedur składowanych odpowiedzialnych za wspomniane operacje. W tym przypadku zastosowania gotowego rozwiązania nie wyklucza możliwości użycia niestandardowej procedury dla konkretnych warunków, przy czym istnieje także możliwości implementacji odpowiedniej funkcji z poziomu samej aplikacji. Wszystkie te cechy gwarantują elastyczność i dostosowanie rozwiązania do ustalonych wymagań. Podczas procesu projektowania, a później tworzenia relacyjnej bazy danych moĪna kierować się różnego rodzaju przesłankami, które mają wpływ na końcową postać schematu bazy danych. Czasem będzie to minimalizacja czasu dostępu do danych, innym razem oszczędność pamięci fizycznej lub w skrajnych przypadkach będą to błędy na poziomie projektowym. Jednak bez względu na przyczynę taki model rzadko w pełni odzwierciedla logikę, jaką będzie musiała realizować aplikacja. W takich przypadkach zastosowanie technologii *ORM* pozwoli utworzyć taki model biznesowy, jaki w rzeczywistości jest wymagany, co także usprawni dalszy proces rozwoju systemu oraz traktowania go jako zgodnego z założeniami. Ta cecha jest szczególnie istotna także z punktu widzenia procesu wytwarzania oprogramowania przy wzięciu pod uwagę faktu, że w pracy nad projektem zaangażowani są zarówno projektanci, mający na celu utworzenie jak najbardziej wydajnej struktury bazy danych, jak i programiści, którzy skupiają się na samej logice systemu[2].

1.3. HIBERNATE.

Hibernate to framework umożliwiający mapowanie obiektowo-relacyjne (ORM) w języku Java. Hibernate umożliwia programistom pracę na poziomie obiektów, a jednocześnie automatycznie mapuje je na poziom relacyjnej bazy danych. Aby skorzystać z Hibernate’a, należy najpierw skonfigurować jego ustawienia. Można to zrobić przy użyciu pliku konfiguracyjnego hibernate.cfg.xml, który definiuje m.in. sposób połączenia z bazą danych oraz mapowanie klas Java na tabele w bazie danych. W Hibernate'ie klasy Java nazywane są encjami, a ich pola odpowiadają kolumnom w tabelach bazy danych. Hibernate umożliwia także tworzenie zapytań przy użyciu języka HQL (Hibernate Query Language) oraz kryteriów. Hibernate posiada także wiele narzędzi umożliwiających m.in. tworzenie tabel w bazie danych na podstawie encji, generowanie kodu na podstawie istniejącej bazy danych, czy też narzędzia do zarządzania transakcjami. Do pracy z Hibernate'em często wykorzystuje się także dodatkowe narzędzia i biblioteki, takie jak

Spring Framework czy JPA (Java Persistence API), które umożliwiają łatwiejszą konfigurację oraz szybsze wystartowanie samego projektu.

Korzyści

- programista nie musi tworzyć struktury bazy danych dla swojej aplikacji.
- programista, nie musi wykonywać ręcznych zapytań do bazy danych;
- znacznie przyspiesza proces tworzenia aplikacji;
- czysty, schludny kod.

Wady

- jest dodatkową warstwą na JDBC i generuje on opóźnienie w czasie dostępu
- generowanie nadmiarowych zapytań – np. znany problem „hibernate n+1”.

Hit czy kit ?

- Obecnie większość firm wybiera stosowanie tego frameworka, ponieważ decydującym czynnikiem jest skrócenie czasu tworzenia rozwiązań informatycznych. Dodatkowo całe rozwiązanie programistyczne jest bardziej znacznie czytelne niż korzystanie z JDBC (które wymaga wielu klas szablonowych).
- Obecnie stosowana techniką jest korzystanie z Hibernate w miejscach gdzie tylko to jest możliwe, natomiast w przypadku gdzie ważny jest czas dostępu korzystać z JDBC.
- Dobrą praktyką jest staranie się by korzystać tylko z jednego rozwiązania albo ORM albo JDBC. Aplikacje tworzone w ten sposób są znacznie łatwiejsze w utrzymaniu.

2. Przykład innego rozwiązania.

LITERATURA

[1] Najważniejsze adnotacje hibernate
https://bykowski.pl/hibernate-6-najprzydatniejsze-adnotacje-przy-modelowaniu-klas-encji/
[2] Najważniejsze adnotacje hibernate
https://bykowski.pl/hibernate-6-najprzydatniejsze-adnotacje-przy-modelowaniu-klas-encji/