УДК 004.41
Ланевич Т. – аспірант
Тернопільський національний технічний університет імені Івана Пулюя

ФРЕЙМВОРКИ ORM ТА ПАТЕРНИ, ЩО В НИХ ВИКОРИСТОВУЮТЬСЯ

Науковий керівник: канд. техн. наук., доц. Боднарчук І.О.

Lanevych T. – postgraduate
Ternopil Ivan Puluj National Technical University

ORM FRAMEWORKS AND PATTERNS USED IN THEM

Supervisor: Assoc. Prof. Dr. Bodnarchuk I.

Ключові слова: Object-Relational Mapping, ORM, Hibernate, Django, CakePHP, Entity Framework.
Key words: Об'єктно-реляційне відображення, ORM, Hibernate, Django, CakePHP, Entity Framework.

ORM розшифровується як об'єктно-реляційне відображення. Це техніка, яка використовується при розробці програмного забезпечення для відображення об'єктів у додатку в таблиці реляційної бази даних (наприклад SQL). Основна ідея ORM полягає в тому, щоб абстрагуватися від баз даних і надати високорівневий, об'єктно-орієнтований інтерфейс для доступу до бази даних і маніпуляцій з нею. Це полегшує розробку та підтримку додатків, які використовують базу даних, оскільки код доступу до бази даних може генеруватися автоматично, а код програми може бути написаний у більш інтуїтивно зрозумілому та виразному об'єктно-орієнтованому стилі [2].

Деякі популярні ORM-фреймворки включають:

1. Hibernate (Java) – дозволяє розробникам писати стійкі класи даних, дотримуючись таких концепцій ООП, як наслідування, поліморфізм, асоціація, композиція. Hibernate має високу продуктивність і масштабованість.
2. Django ORM (Python) – інструмент для швидкого створення веб-додатків.
3. CakePHP (PHP) – надає два типи об'єктів: сховища, які дають доступ до колекції даних, і сутності, які представляють окремі записи даних.
4. Entity Framework (C#) – це маппер об'єктів і баз даних між декількома базами даних. Підтримує SQL, SQLite, MySQL, PostgreSQL та Azure Cosmos DB. [3]

Для реалізації об'єктно-реляційного відображення (ORM), можна використовувати кілька різних патернів:

1. Active Record – це простий і зручний у використанні патерн ORM, в якому кожен запис бази даних представлений відповідним об'єктом у додатку.
2. Data Mapper – відокремлює модель даних додатку від бази даних і надає спосіб зіставлення даних, що зберігаються в базі даних, з об'єктами в додатку і навпаки.
3. Table Data Gateway – це простий і ефективний патерн ORM, який відображає одну таблицю бази даних у відповідний клас у додатку.
4. Unit of Work – відстежує всі зміни, які були внесені до об'єктів у додатку, і гарантує, що база даних буде відповідно оновлена, коли транзакція буде зафіксована [2].

Переваги використання ORM:

1. Незалежність від бази даних. ORM дозволяє застосовувати один і той самий код у різних реляційних системах керування базами даних (СКБД). Як результат, компанії можуть переходити з однієї СКБД на іншу з меншими витратами часу та грошей. У свою чергу, розробники можуть ділитися кодом і використовувати його повторно, щоб перевірити, як програма працює з різними базами даних.

2. Швидша розробка. Завдяки ORM, розробникам не потрібно писати шаблонні SQL-команди. Вони можуть повторно використовувати код кілька разів. В результаті розробники стають більш продуктивними, особливо на початку проєкту, маючи більше часу, щоб зосередитися на бізнес-логіці, а не на створенні запитів.

3. Простіше написання та підтримка коду. Оскільки не потрібно писати SQL-запити всередині коду, він стає більш читабельним, зрозумілим, а отже, його легше підтримувати.

4. Підвищена безпека. ORM-системи мають вбудовані засоби захисту від атак SQL-ін'єкцій – поширеної зловмисної активності, коли хакер надсилає шкідливі запити, щоб змусити додаток виконувати небажані команди.

5. Розширені можливості. Багато ORM-інструментів мають широкий функціонал, який робить розробників ще більш продуктивними. Серед найбільш корисних функцій – автоматична міграція схеми бази даних та оптимізація запитів [1].

Недоліки використання ORM:

1. Проблеми з налагодженням та оптимізацією запитів. Рівень абстракції забезпечує одну з найбільших переваг ORM – можливість використовувати один і той самий код для різних баз даних. Але є й компроміс: розробники практично не мають контролю над тим, як генеруються запити і як додаток взаємодіє з базою даних під капотом. Це створює певні проблеми, коли справа доходить до виявлення та усунення низькорівневих помилок. Також немає можливості оптимізувати запити.

2. Нижча швидкість та продуктивність додатку. Через рівень абстракції та пов'язані з ним додаткові операції, програмне забезпечення з ORM може працювати повільніше, ніж з сирим SQL. Це особливо актуально, коли програма працює зі складними запитами до великих наборів даних, що містять мільйони записів.

3. Прив'язка до постачальника. Хоча ORM надає незалежність від баз даних, він, в той же час, прив'язує до певної бібліотеки, фреймворку або постачальника API.

4. Крута крива навчання. Звільняючи від вивчення SQL, ORM сама по собі не є найпростішою для розуміння технологією. Крива навчання може бути особливо важкою для розробників-початківців, які не дуже добре знайомі з концепціями ООП [1].

Література

1. Object-Relational Mapping Tools: Pros, Cons, and When to Use [Електронний ресурс] – Режим доступу до ресурсу: **https://www.altexsoft.com/blog/object-relational-mapping-tools/.**

2. ORM's patterns [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/nerd-for-tech/orms-patterns-78d626fa412b.

3. What is an ORM – The Meaning of Object Relational Mapping Database Tools [Електронний ресурс] – Режим доступу до ресурсу: https://www.freecodecamp.org/news/what-is-an-orm-the-meaning-of-object-relational-mapping-database-tools/.