# Лекция: Объектно-реляционное отображение (ORM) для программирования дронов

## Введение в ORM

Объектно-реляционное отображение (ORM) — это техника программирования, которая позволяет разработчикам работать с базами данных в объектно-ориентированном стиле. ORM предоставляет способ отображения объектов, используемых в приложении, на записи базы данных, позволяя взаимодействовать с данными, как если бы они были объектами. Это упрощает разработку и поддержание кода, так как разработчики могут работать с данными на уровне объектов, а не SQL-запросов.

## Основные концепции ORM

1. **Класс-сущность**: Объект, который отображается на строку в таблице базы данных. В контексте дронов, это может быть класс Drone или User.
2. **Маппер**: Класс, который управляет преобразованием между объектами и реляционными данными. Например, UserMapper отвечает за преобразование объекта User в SQL-запрос и обратно.
3. **Связь**: Способ отображения связей между объектами и таблицами базы данных (один к одному, один ко многим, многие ко многим).

## Пример использования ORM для программирования дронов

Рассмотрим пример с использованием базы данных SQLite, где мы управляем информацией о дронах. Мы будем использовать классы User и UserMapper для демонстрации основ ORM.

**1. Класс-сущность User**

class User:

def \_\_init\_\_(self, id, username, email):

self.id = id

self.username = username

self.email = email

Класс User представляет сущность пользователя. Он имеет три атрибута:

* id: Уникальный идентификатор пользователя.
* username: Имя пользователя.
* email: Электронная почта пользователя.

**2. Класс UserMapper**

class UserMapper:

def \_\_init\_\_(self, connection):

self.connection = connection

def find\_by\_id(self, user\_id):

# Метод для поиска пользователя по ID

cursor = self.connection.cursor()

cursor.execute("SELECT id, username, email FROM users WHERE id = ?", (user\_id,))

result = cursor.fetchone()

if result:

# Возвращаем объект User, если пользователь найден

return User(\*result)

return None

def save(self, user):

# Метод для сохранения нового пользователя

cursor = self.connection.cursor()

cursor.execute("INSERT INTO users (username, email) VALUES (?, ?)", (user.username, user.email))

self.connection.commit()

Класс UserMapper отвечает за управление сохранением и извлечением объектов User из базы данных. Он использует объект connection для выполнения SQL-запросов.

* find\_by\_id(self, user\_id): Метод ищет пользователя в таблице users по id и возвращает объект User.
* save(self, user): Метод сохраняет нового пользователя в базу данных, вставляя его данные в таблицу users.

**Пример кода**

# Подключение к базе данных SQLite

connection = sqlite3.connect('test.db')

user\_mapper = UserMapper(connection)

# Сохранение нового пользователя

new\_user = User(None, "john\_doe", "john@example.com")

user\_mapper.save(new\_user)

# Поиск пользователя по ID

found\_user = user\_mapper.find\_by\_id(1)

if found\_user:

print(f"Пользователь найден: {found\_user.username}, {found\_user.email}")

else:

print("Пользователь не найден")

connection.close()

В этом примере создается подключение к базе данных SQLite, создается новый пользователь и сохраняется в базу данных с использованием маппера. Затем происходит поиск пользователя по идентификатору.

## Преимущества ORM

1. **Скорость разработки**: ORM значительно ускоряет разработку, так как разработчикам не нужно писать большие объемы SQL-кода.
2. **Читабельность и поддерживаемость**: Код становится более читаемым и поддерживаемым, поскольку разработчики работают с объектами, а не с сырыми данными.
3. **Безопасность**: ORM помогает избежать SQL-инъекций, так как параметры запроса обрабатываются безопасно.

## Заключение

Объектно-реляционное отображение — мощный инструмент, который позволяет разработчикам работать с базами данных в объектно-ориентированном стиле. В контексте программирования дронов ORM позволяет управлять данными о дронах, пользователях и других сущностях, обеспечивая удобство и безопасность. Понимание и использование ORM может значительно упростить процесс разработки и улучшить качество кода.