УНИВЕРСИТЕТ СИНЕРГИЯ

Курс | Фреймворк Flask

ОСНОВЫ FLASK-SQLALCHEMY

Спикер | Немков Максим Юрьевич

СОДЕРЖАНИЕ ТЕМЫ

- Flask SqlAlchemy
- Flask-Migrate
- Связь таблиц «один ко многим»
- Связь таблиц «многие ко многим»

Определение понятия

FLASK-SQLALCHEMY

это расширение для Flask, которое позволяет легко и быстро работать с реляционными базами данных

ШАГИ ДЛЯ СОЗДАНИЯ МОДЕЛИ

01 Создать класс модели, который будет представлять таблицу в базе данных. Класс должен наследоваться от `db.Model`

```
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(64), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
```

ШАГИ ДЛЯ СОЗДАНИЯ МОДЕЛИ

02 Загрузить модель в приложение. Для этого потребуется прописать простой Python код в консоли

```
python
>>from main import app, db
>>app.app_context().push()
>>db.create_all()
>>quit()
```

КОМАНДЫ ДЛЯ РАБОТЫ С ДАННЫМИ

Здесь мы создаём нового пользователя с именем `Ivan`
и адресом электронной почты `ivan@sinergy.ru`,
добавляем его в базу данных и сохраняем изменения

```
user = User(username="Ivan",
email="ivan@sinergy.ru")
db.session.add(user)
db.session.commit()
```

ЗАПРОСЫ К БАЗЕ ДАННЫХ

Мы получаем всех пользователей из базы данных и выводим их идентификаторы, имена и адреса электронной почты

```
users = User.query.all()
for user in users:
    print(f"{user.id},
{user.username},
{user.email}")
```

Определение понятия

FLASK-MIGRATE

это расширение для Flask, которое позволяет легко и быстро управлять изменениями в схеме базы данных

Определение понятия

МИГРАЦИЯ

это файл, который содержит изменения
в схеме базы данных

ИНИЦИАЛИЗАЦИЯ МИГРАЦИИ

Экспортируем переменную, чтобы система поняла
наш основной файл, инициируем миграцию,
запускаем миграцию и обновляем базу данных

```
export FLASK_APP=main.py
flask db init
flask db migrate
flask db upgrade
```

Определение понятия

СВЯЗЬ «ОДИН КО МНОГИМ»

это тип связи между двумя таблицами, при котором одна запись в одной таблице может быть связана с несколькими записями в другой таблице. Это означает, что один объект (например, пользователь) может иметь несколько связанных объектов (например, сообщений)

СВЯЗЬ ТАБЛИЦ «ОДИН КО МНОГИМ»

В этом примере модель Post имеет поле user_id, которое
связано с полем id модели User через отношение ForeignKey.
Это позволяет одному пользователю иметь несколько сообщений

```
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(50))
    posts = db.relationship('Post',
    backref='author', lazy='dynamic')

    def __repr__(self):
        return f'<User {self.name}>'
```

```
class Post(db.Model):
    id = db.Column(db.Integer,
    primary_key=True)
    title = db.Column(db.String(100))
    content = db.Column(db.Text)
    user_id = db.Column(db.Integer,
    db.ForeignKey('user.id'))

    def __repr__(self):
        return f"<Post {self.title}>"
```

СВЯЗЬ ТАБЛИЦ «ОДИН КО МНОГИМ»

Создание новой записи в таблицу

```
post = Post(title="My first post", content="This is my first post.", user_id=1)
```

Определение понятия

СВЯЗЬ ТАБЛИЦ «МНОГИЕ КО МНОГИМ»

это тип связи между двумя таблицами, при котором одна запись в одной таблице может быть связана с несколькими записями в другой таблице и наоборот. Это означает, что один объект (например, пользователь) может иметь несколько связанных объектов (например, групп), и одна группа может иметь несколько пользователей

ПОДВЕДЕМ ИТОГИ

- Узнали про Flask SqlAlchemy
- Поработали с Flask-Migrate
- Изучили связь таблиц «один ко многим»
- Изучили связь таблиц «многие ко многим»