

Модуль 4 Занятие 7

CRUD с SQLAlchemy



flask shell

CRUD

Практика

+

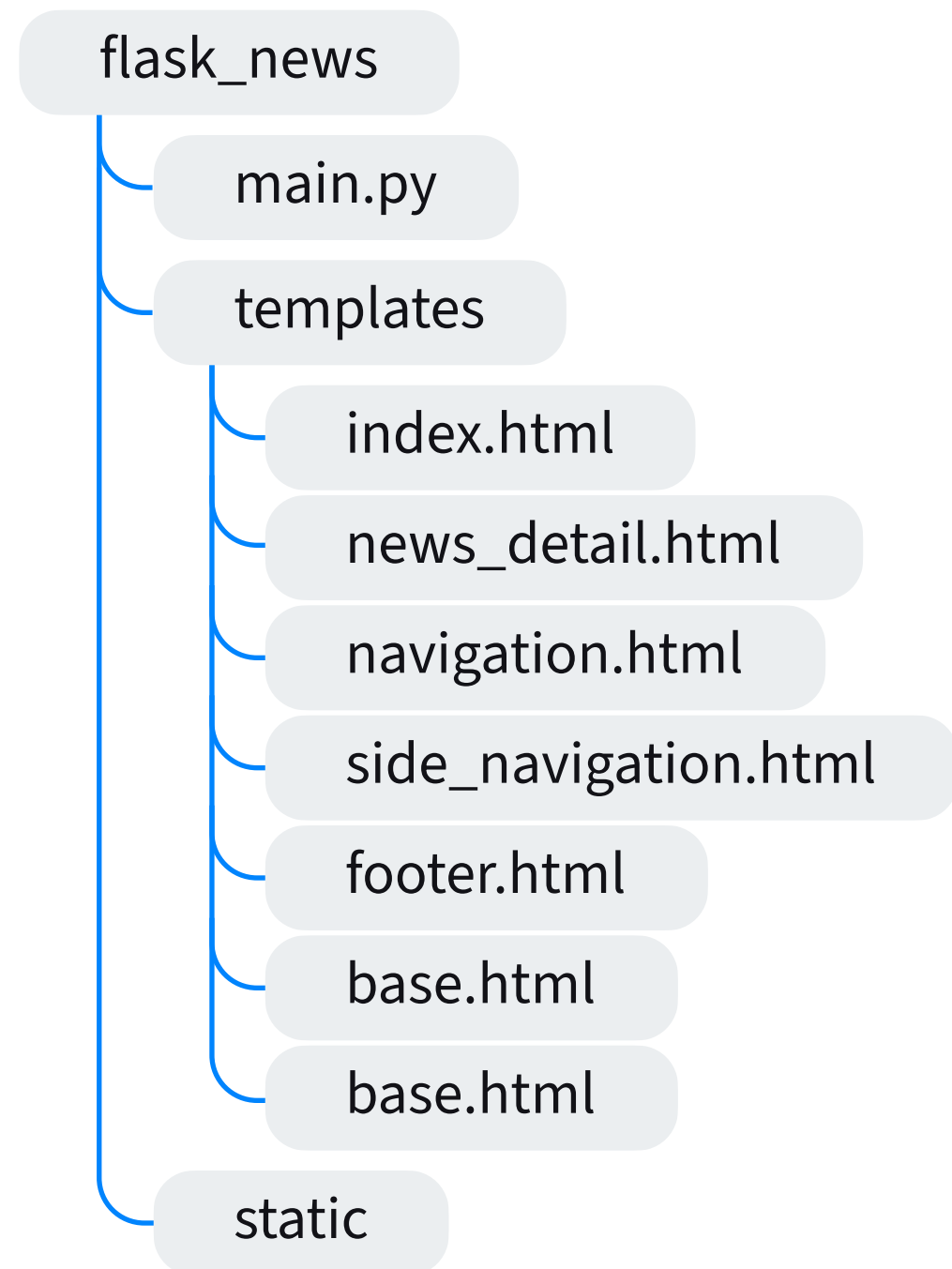
**Продолжаем создавать
новостной портал**

Повторение



Проверим структуру проекта flask_news и вспомним, на чем мы остановились.

Что нужно добавить в наш проект?



main.py

```
...
```

```
app = Flask(__name__)
```

```
...
```

```
db = SQLAlchemy(app)
```

```
class News(db.Model):
```

```
...
```

```
db.create_all()
```

```
class NewsForm(FlaskForm):
```

```
...
```

```
@app.route('/')
```

```
def index():
```

```
...
```

```
@app.route('/news_detail/<int:id>')
```

```
def news_detail(id):
```

```
...
```

```
@app.route('/add_news', methods=['GET', 'POST'])
```

```
def add_news():
```

```
...
```

← Импорты, настройки приложения,
подключение базы данных

← Модель News

← Создание базы данных и таблиц

← Форма добавления новостей

← Эндпоинт главной страницы
(вывод всех новостей)

← Эндпоинт отдельной новости

← Эндпоинт добавления новости

Создание связей

В реляционных базах данных могут быть следующие типы связей:



Один к одному



Один ко многим



Многие ко многим

Чтобы указать связь в дочерней модели с помощью объекта `ForeignKey`, необходимо добавить новое поле — внешний ключ на родительскую модель.

news

id	Integer
title	String(255)
text	Text
created_date	DateTime
category_id	Integer

category

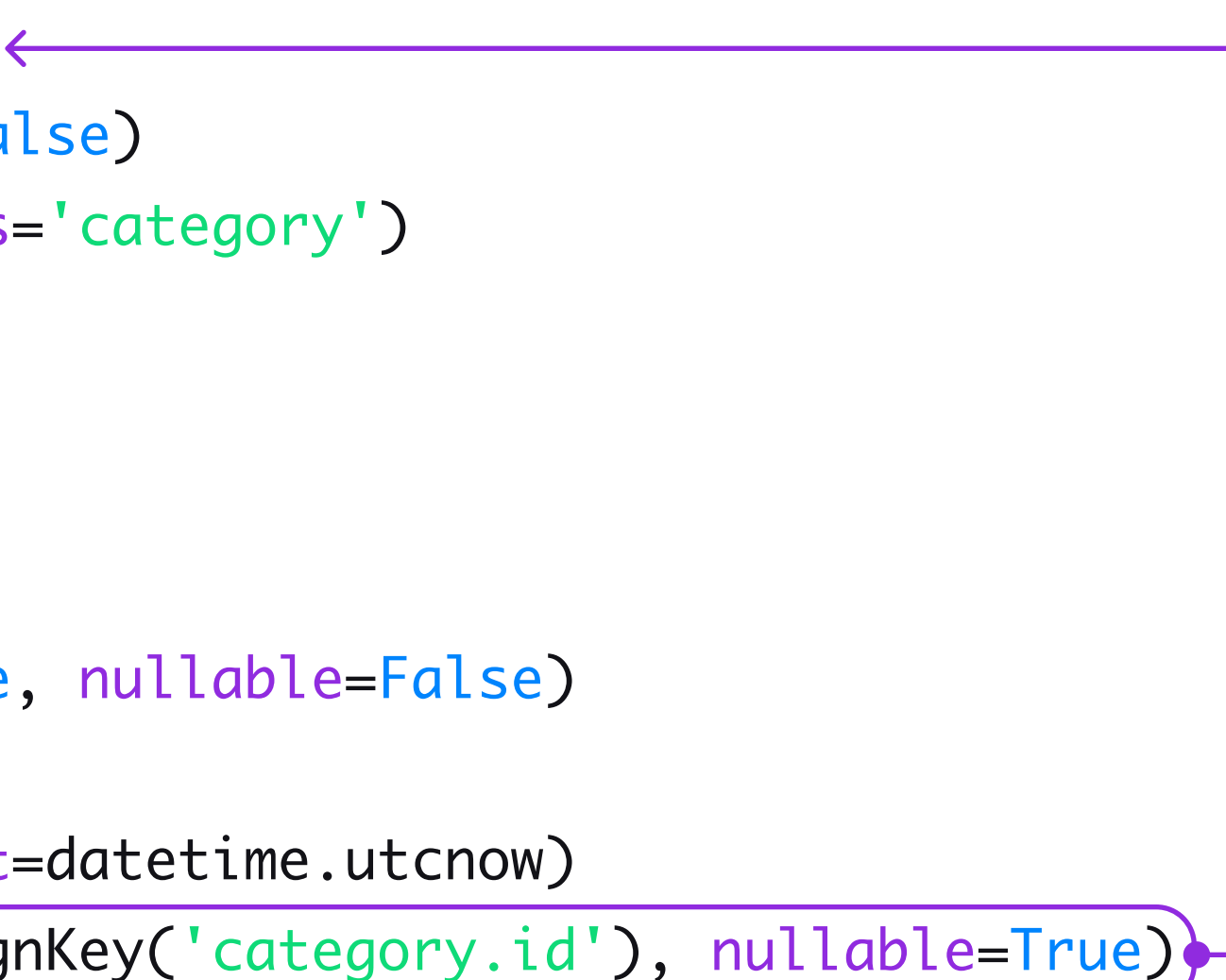
id	Integer
title	String(255)

Модель Category

Добавим модель Category:

```
class Category(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(255), nullable=False)
    news = db.relationship('News', back_populates='category')

class News(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(255), unique=True, nullable=False)
    text = db.Column(db.Text, nullable=False)
    created_date = db.Column(db.DateTime, default=datetime.utcnow)
    category_id = db.Column(db.Integer, db.ForeignKey('category.id'), nullable=True)
    category = db.relationship('Category', back_populates='news')
```



Модель Category

`db.relationship()` используется для добавления двунаправленной связи и позволит нам получить список всех новостей категории, используя синтаксис `Category.news` и наоборот, получать категорию новости, используя синтаксис `News.category`.

```
class Category(db.Model):  
    id = db.Column(db.Integer, primary_key=True)  
    title = db.Column(db.String(255), nullable=False)  
    news = db.relationship('News', back_populates='category')
```

```
class News(db.Model):  
    id = db.Column(db.Integer, primary_key=True)  
    title = db.Column(db.String(255), unique=True, nullable=False)  
    text = db.Column(db.Text, nullable=False)  
    created_date = db.Column(db.DateTime, default=datetime.utcnow)  
    category_id = db.Column(db.Integer, db.ForeignKey('category.id'), nullable=True)  
    category = db.relationship('Category', back_populates='news')
```

Flask shell

Откройте терминал, активируйте виртуальное окружение, перейдите в папку с проектом и выполните команды:

```
>>> flask --app main.py shell
```

```
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep 5 2022,  
14:08:36) [MSC v.1933 64 bit (AMD64)] on win32
```

```
App: main
```

```
Instance: C:\Users\User\flask_news\instance
```

```
>>> from main import db
```

```
>>> from main import News
```

```
>>> from main import Category
```

Запускаем интерпретатор Python
в контекст Flask приложения

Импортируем объект базы данных
и модели.

Для выхода используйте команду `exit()`.

CRUD

CRUD — это аббревиатура, обозначающая четыре действия при работе с базами данных:

Create

создание объекта в базе

Read

чтение объекта

Update

обновление или изменение объекта

Delete

удаление объекта

Прежде чем начать

- 1 Заполните базу данных тестовыми данными или воспользуйтесь уже готовой базой данных.
- 2 Переопределите метод `__repr__` у ваших моделей, например так:

```
class Category(db.Model):  
    ...  
  
    def __repr__(self):  
        return f'Category {self.id}: ({self.title})'  
  
class News(db.Model):  
  
    def __repr__(self):  
        return f'News {self.id}: ({self.title[:20]}...).'
```

Создание объекта

```
>>> news = News(title = 'Заголовок', text = 'Текст')
>>> news.title
'Заголовок'
```

```
>>>
```

```
>>> news.id
>>> news.created_date
```

```
>>>
```

```
>>> db.session.add(news)
>>> db.session.commit()
```

```
>>>
```

```
>>> news
News 10: (Заголовок...)
>>> news.id
10
>>> news.created_date
datetime.datetime(2023, 3, 31, 14, 27, 31, 535819)
```

```
>>> news.category
```

Создадим объект News() и сразу укажем значения атрибутов.

Или можно так:

```
>>> news = News()
>>> news.title = 'Заголовок'
>>> news.text = 'Текст'
```

Ничего не выводится, так как объекта еще нет в базе данных.

Чтобы добавить объект в базу данных, нужно добавить его в сессию и подтвердить изменения.

Теперь объект в базе данных.

Категория пуста, мы ее не указывали.

Чтение объектов

У моделей есть атрибут `query`, обращаясь к которому можно получить объект запроса `Query`, соответствующий всем доступным записям модели.

```
>>> News.query.all()
```

```
[News 1: (Слишком сложно и нич...), News 2: (Картины,
которые вам...), News 3: (Представление длиной...),
News 4: (Роботы-помощники...), News 5: (Как повесить
продукт...), News 6: (Лечение простуды...), News 7:
(Самый красивый пёс...), News 8: (Случай в зоопарке...),
News 9: (Удивительное событие...), News 10: (Заголовок...)]
```

Метод `all()` возвращает результаты, представленные этим `Query` объектом в виде списка.

```
>>>
```

```
>>> News.query.get(10)
```

```
News 10: (Заголовок...)
```

```
>>> News.query.get(11)
```

```
>>>
```

Метод `get(id)` возвращает один объект БД (или `None`), значение первичного ключа которого равно `id`. Новости с `id = 11` нет в базе данных – ничего не выводится.

Изменение объекта

```
>>> news = News.query.get(10)
>>> News 10: (Заголовок...)
>>> news.category
```

Получим объект с id = 10 — это наша добавленная новость. Категория пуста, мы ее не указывали.

```
>>>
>>> news.category_id = 6
>>> db.session.add(news)
>>> db.session.commit()
```

Изменим новость и укажем значение атрибута category_id = 6 — это наша добавленная категория «Образование». Чтобы обновить объект, нужно также добавить его в сессию и подтвердить изменения.

```
>>>
>>> news
News 10: (Заголовок...)
>>> news.category
Category 6: (Образование)
```

Теперь можно вывести категорию новости.

```
>>>
```

Удаление объекта

```
>>> category = Category.query.get(6)
>>> category
>>> Category 6: (Образование)
>>>
```

```
>>> db.session.delete(category)
>>> db.session.commit()
>>> Category.query.all()
[Category 1: (Спорт), Category 2:
(Технологии), Category 3: (Культура),
Category 4: (Наука), Category 5: (Юмор)]
>>>
```

```
>>> news = News.query.get(10)
>>> news.category
>>>
```

Для удаления объекта нужно использовать метод `delete()` объекта сессии и передать методу удаляемый объект. После этого нужно подтвердить изменения.

Атрибут `category_id` новости стал `None`, так как категория была удалена — ничего не выводится.

Другие методы

Кроме уже рассмотренных методов `all()` и `get()` объекта запроса `Query`, существует и множество других методов, позволяющих получать данные из базы данных, например:

`count()`

Возвращает количество объектов в запросе

`first()`

Возвращает первый объекта из запроса или `None` если записей нет

`filter(*criterion)`

Возвращает из `Query` объекты по заданным параметрам

`limit(limit)`

Ограничивает количество объектов результата

`offset(offset)`

Возвращает из `Query` объекты со смещением

`order_by(*criterion)`

Упорядочивает результат запроса

filter

```
>>> News.query.filter(News.id > 5).all()
```

```
[News 6: (Лечение простуды...), News 7: (Самый красивый  
пёс...), News 8: (Случай в зоопарке...), News 9:  
(Удивительное событие...)]
```

```
>>>
```

```
>>> News.query.filter(News.id > 5, News.category_id == 4).all()  
[News 6: (Лечение простуды...)]
```

```
>>>
```

```
>>> News.query.filter(News.title.contains("сложно")).all()  
[News 1: (Слишком сложно и нич...)]
```

```
>>>
```

```
>>> News.query.filter(News.category_id.in_([1, 3, 4])).all()  
[News 1: (Слишком сложно и нич...), News 2: (Картины, которые  
вам...), News 3: (Представление длиной...), News 6: (Лечение  
простуды...), News 8: (Случай в зоопарке...)]
```

```
>>>
```

Несколько критериев можно передать через запятую. В этом случае фильтр будет выполняться по обоим критериям.

Составлять условия можно с помощью различных операторов и выражений.

order_by

```
>>>News.query.order_by(News.title).all()

[News 5: (Как повысить продукт...), News 2: (Картины,
которые вам...), News 6: (Лечение простуды...), News 3:
(Представление длиной...), News 4: (Роботы-помощники...),
News 7: (Самый красивый пёс...), News 1: (Слишком сложно
и нич...), News 8: (Случай в зоопарке...), News 9:
(Удивительное событие...)]

>>>
```

```
>>> News.query.order_by(db.desc(News.title)).all()

[News 9: (Удивительное событие...), News 8: (Случай
в зоопарке...), News 1: (Слишком сложно и нич...), News 7:
(Самый красивый пёс...), News 4: (Роботы-помощники...),
News 3: (Представление длиной...), News 6: (Лечение
простуды...), News 2: (Картины, которые вам...), News 5:
(Как повысить продукт...)]

>>>
```

Для упорядочивания результата по убыванию нужно использовать функцию `db.desc()`.

limit и offset

```
>>> News.query.limit(3).all()
[News 1: (Слишком сложно и нич...), News 2:
(Картины, которые вам...), News 3: (Представление длиной...)]
>>>
>>> News.query.offset(3).limit(3).all()
[News 4: (Роботы-помощники...), News 5: (Как повысить
продукт...), News 6: (Лечение простуды...)]
>>>
```

limit и offset позволяют
ограничить количество
объектов результата
и выполнить смещение

ИТОГИ



CRUD — это аббревиатура, обозначающая четыре действия при работе с базами данных: **Create** — создание объекта в базе, **Read** — чтение объекта, **Update** — обновление или изменение объекта, **Delete** — удаление объекта.



Примеры:

Создание

```
>>> news = News(title =  
'Заголовок', text = 'Текст')  
>>> db.session.add(news)  
>>> db.session.commit()
```

Чтение

```
>>> News.query.all()  
>>> News.query.get(10)
```

Изменение

```
>>> news = News.query.get(10)  
>>> news.title = 'New title'  
>>> db.session.add(news)  
>>> db.session.commit()
```

Удаление

```
>>> news = News.query.get(10)  
>>> db.session.delete(news)  
>>> db.session.commit()
```