

Реализация MVC- приложения на Flask

Букин Д.Ю., ЗИВТ(2)

Одним словом о Flask - микрофреймворк

(А ещё он не MVC)

Пример Hello World



```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"


if __name__ == "__main__":
    app.run()
```

Маршрутизация - как это было...

```
class SimpleHTTPRequestHandler(BaseHTTPRequestHandler):  
  
    def do_GET(self):  
  
        import views.client as view  
  
        if self.path == '/':  
            self.send_response(200)  
            self.send_header('Content-Type', 'text/html, charset="utf-8"')  
            self.end_headers()  
  
            result = view.client_view.render_get_form()  
            result = bytes(result, 'utf-8')  
  
            self.wfile.write(result)
```

M – Model

ORM SQLAlchemy как
способ создать модель



```
# файл model.py
```

```
from flask_sqlalchemy import SQLAlchemy
```

```
db = SQLAlchemy()
```

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

```
    __tablename__ = "clients"
```

```
    id = db.Column(db.Integer, primary_key=True)
```

```
    first_name = db.Column(db.String(80), nullable=False)
```

```
    last_name = db.Column(db.String(80), nullable=False)
```

```
    email = db.Column(db.String(120), unique=True, nullable=False)
```

```
    city = db.Column(db.String(80), nullable=False)
```

```
    post_index = db.Column(db.Integer, nullable=False)
```

```
    address = db.Column(db.String(150), nullable=False)
```

```
    def __repr__(self):
```

```
        return f"{self.first_name} {self.last_name}"
```

Инициализация базы данных



```
# файл main.py

from model import db

app.config.from_pyfile('./config.py') # файл с переменными конфигурации

db.init_app(app)


@app.before_first_request
def create_tables():
    db.create_all()
```

C - Contoller

Blueprints, SQLAlchemy,
Jinja...

Blueprints – способ организации модульности приложения на Flask

В нашем случае – способ
изоляции контроллеров от
основного файла приложения

A decorative dark gray curved line starts from the bottom right corner and curves upwards and to the left, ending near the center of the right edge of the slide.

Регистрация Blueprint



```
# views/controllers.py

from flask import Blueprint

controller = Blueprint('controller', __name__, template_folder='templates')
```



```
# main.py

from views.controllers import controller

app.register_blueprint(controller)
```

Пример контроллера

```
from flask import render_template
from model import Client, db


@controller.route('/users/<int:id>/')
def show_user(id):
    user = db.session.query(Client).get(id)
    if user is None:
        return page_not_found(404)


    return render_template('layout.html', content='user.html', cl=user)

@controller.errorhandler(404)
def page_not_found(error):
    return render_template("404.html"), 404
```

Формы - WTForms

Создаём формы как
Python-объект

A decorative dark gray curved line starts from the bottom right corner and curves upwards and to the left, ending near the center of the bottom edge of the slide.



```
from flask_wtf import FlaskForm
from wtforms import StringField, SubmitField, IntegerField
from wtforms.validators import DataRequired, Email, NumberRange

class Form(FlaskForm):

    first_name = StringField("Имя: ", validators=[DataRequired()])
    last_name = StringField("Фамилия: ", validators=[DataRequired()])
    email = StringField("Email: ", validators=[Email()])
    city = StringField("Город: ", validators=[DataRequired()])
    post_index = IntegerField("Индекс: ", validators=[DataRequired(),
NumberRange(min = 100000, max = 999999)])
    address = StringField("Адрес: ", validators=[DataRequired()])
    submit = SubmitField("Отправить")
```

Файл шаблона формы

```
<form method="post">
  {{ form.csrf_token() }}

  <ul>
    {% for field in form if field.name != "csrf_token" %}
      <li>{{ field.label() }}
        {{ field }}

        {% for error in field.errors %}
          {{ error }}
        {% endfor %}
      </li>
    {% endfor %}
  </ul>
</form>
```

Но теперь форму нужно
обработать...

Продолжение C - Controllers

Начало...



```
from flask import render_template
from views.form import Form
from model import Client, db
import pyqrcode

@controller.route('/', methods=['get', 'post'])
def index():

    form = Form()

    if form.validate_on_submit():

        exists = db.session.query(Client).filter_by(email=form.email.data).first() is not None
        if exists:
            error = "E-mail уже зарегистрирован"
            return render_template('layout.html', content='error.html', error=error)

        .....
```


...продолжение

```
#.....

cl = Client(
    first_name = form.first_name.data,
    last_name = form.last_name.data,
    email = form.email.data,
    city = form.city.data,
    post_index = int(form.post_index.data),
    address = form.address.data
)

db.session.add(cl)
db.session.commit()

url = pyqrcode.create(f'https://FlaskMVC.danilabukin.repl.co/users/{cl.id}/')
url.svg(f'images/{cl.id}.svg', scale=8)

return render_template('layout.html', content='post.html', cl=cl)

return render_template('layout.html', content='form.html', form=form)
```

Оставшиеся контроллеры...

Раздача изображений



```
@controller.route('/images/<int:id>.svg/')
def show_image(id):
    try:
        file = open(f"images/{id}.svg").read()
    except Exception:
        return page_not_found(404)
    else:
        response = make_response(file)
        response.headers["Content-Type"] = 'image/svg+xml'

    return response
```

Получение всех пользователей...



```
@controller.route('/users/')  
def show_all_users():  
    users = db.session.query(Client).all()  
    return render_template('layout.html', content='allusers.html', users=users)
```

...и шаблон

```
<table border="1">
  <caption>Все пользователи</caption>
  <tr>
    <th>Номер</th>
    <th>Имя</th>
    <th>Фамилия</th>
    <th>E-mail</th>
    <th>Город</th>
    <th>Индекс</th>
    <th>Адрес</th>
  </tr>
  {% for user in users %}
    <tr>
      <td><a href="/users/{{ user.id }}">{{ user.id }}</a></td>
      <td>{{ user.first_name }}</td>
      <td>{{ user.last_name }}</td>
      <td>{{ user.email }}</td>
      <td>{{ user.city }}</td>
      <td>{{ user.post_index }}</td>
      <td>{{ user.address }}</td>
    </tr>
  {% endfor %}
</table>
```

Наш готовый main.py

```
from flask import Flask
from views.controllers import controller

from model import db

app = Flask(__name__)
app.register_blueprint(controller)
app.config.from_pyfile('./config.py')

db.init_app(app)

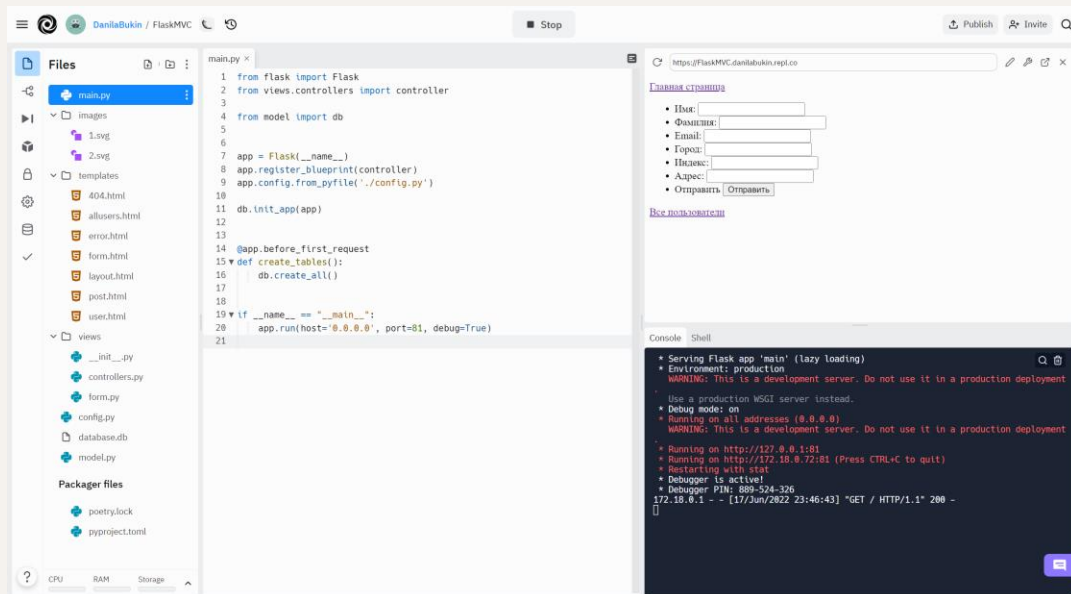
@app.before_first_request
def create_tables():
    db.create_all()

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=81, debug=True)
```

Сделаем выводы

- Самая мощная возможность Flask – маршрутизация
- Модульность – важнейшая концепция
- Мы бы могли использовать те же модули для приложения на ванильном Python
- Свобода в выборе архитектуры приложения требует бóльших усилий

P.S.



<https://replit.com/@DanilaBukin/FlaskMVC#main.py>