План лекций

- как работает интернет
- что такое web
- основы html+css+консоль разработчика
- основы js + простейший пример игры
- rest+soap
- обзор основных фрэймворков бэка и фронта
- поднятие простого бэка на джанге или ноде, отладка запросов с помощью Postman
- обзор современной команды разработки web-продуктов

План лабораторных работ

- Введение Познакомиться с курсом и узнать о том, на что способен предмет курса — микрофреймворк Flask.
- Быстрый старт с Flask Узнать, как создать простейшее Flask-приложение, готовое к запуску и ответу на запросы.
- Маршрутизация Познакомиться с концепцией маршрутизации и узнать, как Flask подходит к обработке маршрутов.
- Сложный роутинг и формирование URL Узнать про расширенные возможности системы маршрутизации и научиться получать URL для требуемых маршрутов.
- Шаблонизация HTML Узнать, как Flask формирует HTML-страницы с помощью механизма шаблонизации.
- Запросы Научиться работе с данными запросов: глаголами, заголовками, параметрами.
- Ответы сервера Научиться возвращать разного вида данные в ответ на запросы.
- Отладчик Werkzeug Узнать о ключевых особенностях встроенного во Flask отладчика Werkzeug.
- Запуск Flask в боевых условиях Узнать, чем отличается запуск на деве от запуска в проде. Деплой.

Введение

Зачем нужны фреймворки

+

- Повышение скорости и удобства разработки
- Сокращение затрат на создание приложения
- Оптимизация рабочего времени
- Написание чистого кода, не требующего существенного рефакторинга (переработки) в дальнейшем

–

- Проблемы с безопасностью
- Сложность освоения
- Необходимость переучиваться при смене продукта

Что такое Flask

- **Flask** — фреймворк для создания веб-приложений на языке программирования Python, использующий набор инструментов Werkzeug, а также шаблонизатор Jinja2

- **Flask** относится к категории так называемых микрофреймворков — минималистичных каркасов веб-приложений, сознательно предоставляющих лишь самые базовые возможности

- Поддерживается установка посредством пакетного менеджера PyPI, версия 1.0 совместима с Python 2.7, Python 3.3 и выше.

- Создатель и основной автор — австрийский программист Армин Ронахер, начал работу над проектом в 2010 году.

Что такое “микро” фреймворк

Что значит «микро»? «Микро» в фреймворке относится не только к простоте и небольшому размеру базы, но это также может означать тот факт, что он **не предлагает вам много проектных решений**. Несмотря на то, что Flask использует нечто подобное в виде шаблонизатора, мы не будем принимать подобные решения для вашего хранилища данных или других частей. Тем не менее, для нас термин «микро» **не означает, что вся реализация должна вписываться в один файл**. Одним из проектных решений во Flask является то, что **простые задачи должны быть простыми**; они не должны занимать много кода, и это не должно ограничивать вас. Поэтому мы сделали несколько вариантов дизайна, некоторые люди могут посчитать это удивительным и даже странным. Например, Flask использует локальные треды внутри объектов, так что вы не должны передавать объекты в пределах одного запроса от функции к функции, оставаясь в безопасном треде. Хоть это и очень простой подход, который позволяет сэкономить время, такое решение **может вызвать некоторые проблемы для слишком больших приложений**, поскольку изменения в этих локальных тредах-объектах могут произойти где угодно в этом треде. Для того, чтобы решить эти проблемы, мы не стали скрывать от вас локальные треды-объекты, вместо этого мы охватываем их и предоставляем вам много инструментов, чтобы сделать работу с ними настолько приятной, насколько это возможно. Во Flask многие вещи предварительно сконфигурированы, на основе общей базовой конфигурации. Например, шаблоны и статические файлы сохранены в подкаталогах в пределах исходного дерева. Вы также можете изменить это, но обычно этого не требуется. Основная причина почему Flask называется «микрофреймворком» — это **идея сохранить ядро простым, но расширяемым**. В нем нет абстрактного уровня базы данных, нет валидации форм или всего того, что уже есть в других библиотеках. Однако, Flask поддерживает расширения, которые могут добавить необходимую функциональность и имплементирует их так, как будто они уже были встроены изначально. В настоящее время уже есть расширения: формы валидации, поддержка закачки файлов, различные технологии аутентификации и многие другие.

Что нужно для лабораторных работ

- установить python (3.7+) https://www.python.org/downloads/
- установить pycharm community (другие IDE с поддержкой python) https://www.jetbrains.com/ru-ru/pycharm/download/#section=windows
- установить flask (latest) https://flask.palletsprojects.com/en/2.0.x/installation/
- создать аккаунт на https://github.com/
- изучить как пользоваться git https://losst.ru/kak-polzovatsya-github
- прислать списки групп с указанием git-а
- написать hello world на flask https://python-scripts.com/flask-vs-django
- закомитить hello world в git

Hello world

- Первая строка импортирует Flask
- Третья строка инициализирует переменную приложения, используя атрибут __name__
- Пятая строка содержит в себе чудеса Flask. @app.route – это декоратор Python. Он берет функцию снизу и модифицирует её. В данном случае, мы используем его для маршрутизации трафика из определенного URL в расположенной ниже функции. Используя различные вызовы @app.route, мы можем «спровоцировать» различные части кода, когда пользователь посещает разные части нашего приложения. В данном случае, у нас только один маршрутизатор «/», который является корнем по умолчанию в нашем приложении.
- В шестой строке, функция под названием ***hello*** не так уж важна. Вместо вызова этой функции из той или иной части нашего кода, она будет вызвана автоматически. Это хорошая практика для того, чтобы дать ей релевантное название.
- Седьмая стока возвращает строку нашему пользователю. Обычно мы рендерим шаблон или обрабатываем HTML, чтобы пользователь могу видеть аккуратно оформленную страницу, но и возврат строк Python также хорошо работает.
- Девятая строка – это обычный шаблон Python, используемый для того, чтобы убедиться в том, что мы не запускаем ничего в автоматическом режиме, если наш код был импортирован из другого скрипта Python.
- В десятой строке вызывается метод run() приложения, которое мы инициализировали в третьей строке. Это запускает сервер разработки для Flask и дает нам возможность посетить наше веб приложение с нашей локальной машины путем посещения localhost.

```
1  from flask import Flask
2
3  app = Flask(__name__)
4
5  @app.route("/")
6  def hello():
7      return "Hello, World!"
8
9  if __name__ == "__main__":
10     app.run()
```

Роутинг и формирование URL

Маршрутизация

- Роутеры строятся с помощью декораторов **route**

```
@app.route('/')
def index():
    return 'Index Page'
```

```
@app.route('/hello')
def hello():
    return 'Hello World'
```

Динамические роутеры

- Для добавления переменной части в URL можно пометить эти разделы, как <variable_name>
- Дополнительно преобразователь может быть определен путем указания правила <converter:variable_name>
- Имеются следующие конверторы

int	Принимает целые числа
float	то же самое, что int, только с плавающей точкой
path	похоже на то, что установлено по умолчанию, но принимает слэши

Динамические роутеры

Примеры динамических роутеров

```
@app.route('/user/<username>')
def show_user_profile(username):
    # show the user profile for that user
    return 'User %s' % username
```

```
@app.route('/post/<int:post_id>')
def show_post(post_id):
    # show the post with the given id, the id is an integer
    return 'Post %d' % post_id
```

Генерация URL

Функция **url_for()**

- может генерировать URL. Для создания URL, используйте функцию
- принимает имя функции в качестве первого аргумента, а также ряд ключевых аргументов, каждый из которых соответствует переменной части URL правила
- части неизвестной переменной добавляется к URL в качестве параметров запроса

https://pythonru.com/uroki/8-sozdanie-url-vo-flask#:~:text=Flask%20%D0%BC%D0%BE%D0%B6%D0%B5%D1%82%20%D0%B3%D0%B5%D0%BD%D0%B5%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D1%82%D1%8C%20URL%20%D1%81,%2F%20.

Генерация URL

Метод **test_request_context()** говорит Flask, как нужно обрабатывать запрос

```
>>> from flask import Flask, url_for
>>> app = Flask(__name__)
>>> @app.route('/')
... def index(): pass
...
>>> @app.route('/login')
... def login(): pass
...
>>> @app.route('/user/<username>')
... def profile(username): pass
...
>>> with app.test_request_context():
...  print url_for('index')
...  print url_for('login')
...  print url_for('login', next='/')
...  print url_for('profile', username='John Doe')
...
/
/login
/login?next=/
/user/John%20Doe
```

Генерация URL

- url_for() — одна из функций, которую можно использовать внутри html шаблона.
- Чтобы сгенерировать URL внутри шаблонов, нужно просто вызвать **url_for()** внутри фигурных скобок {{ … }}:

```
<a href="{{ url_for('books', genre='biography') }}">Books</a>
```

Вывод:

```
<a href="/books/biography/">Books</a>
```

Лабораторные 2,3

- Написать Hello World с помощью html шаблона
- Страница пользователя с динамическим url в двух версиях: внутри/вне html шаблона