

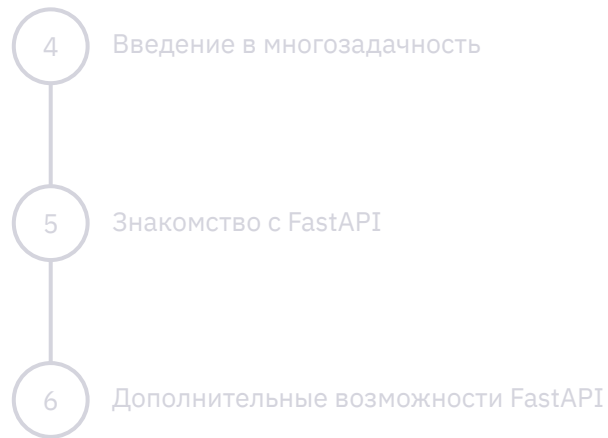
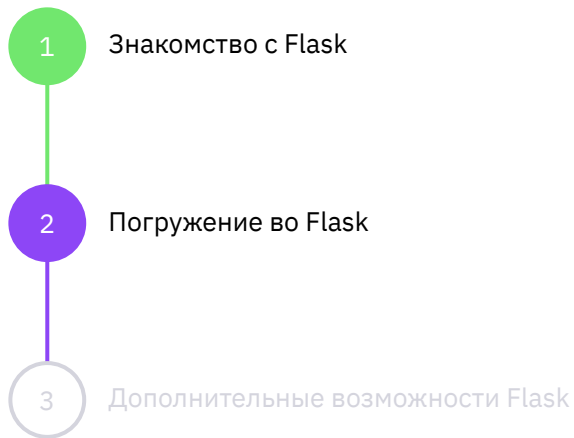
# Погружение во Flask

Урок 2





## План курса










# Содержание урока



## Что будет на уроке сегодня

-  Узнаем про экранирование пользовательских данных
-  Разберёмся с генерацией url адресов
-  Изучим обработку GET и POST запросов
-  Узнаем несколько полезных функций Flask
-  Разберёмся с cookie файлами и сессиями



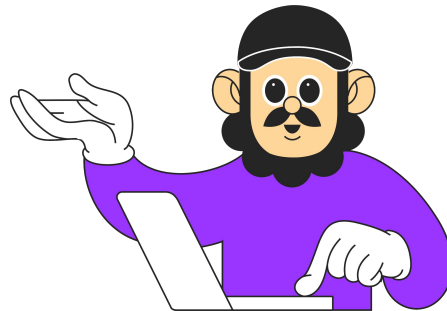
# Экранирование пользовательских данных



## Экранирование пользовательских данных

Запрос вида [http://127.0.0.1:5000/<script>alert\("I am hacker"\)</script>/](http://127.0.0.1:5000/<script>alert('I am hacker')</script>/) вполне может запустить скрипт. Используем экранирование:

```
from markupsafe import escape
```





# Генерация URL адресов



## Генерация URL адресов

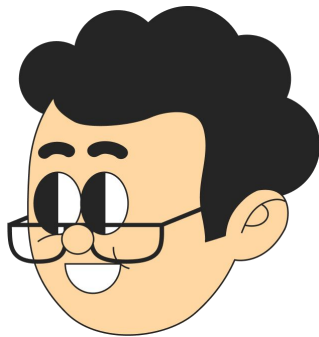
При формировании адреса используем функцию `url_for()` из модуля Flask:

```
url_for("test_url", num=42, data="new_data", pi=3.14515)
```

А в шаблонах используем `url_for` для указания пути к статике:

```

```







# Обработка запросов



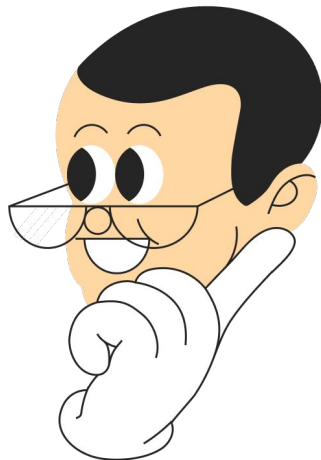
## Обработка GET запросов

Обработка GET запросов является поведением по умолчанию для представлений.

```
from flask import request
```

```
...  
    data = request.args.get('data')  
...
```

<http://127.0.0.1:5000/get/?name=alex&age=13&level=80>



## Обработка POST запросов

POST запросы используются для отправки данных на сервер. Обычно используют HTML форму:

```
<form action="/submit" method="post">  
  <input type="text" name="name" placeholder="Имя">  
  <input type="submit" value="Отправить">  
</form>
```





## Замена route на get и post

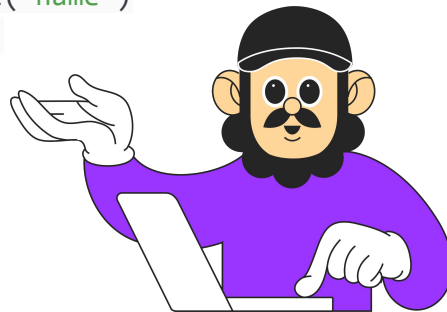
Функция-представление может быть разделена на две. Каждая обрабатывает свой вид запроса

### GET

```
@app.get('/submit')  
def submit_get():  
    return render_template('form.html')
```

### POST

```
@app.post('/submit')  
def submit_post():  
    name = request.form.get('name')  
    return f'Hello {name}!'
```



## Загрузка файлов через POST запрос

Обязательный минимум

- `enctype=multipart/form-data` в HTML форме
- `file = request.files.get('file')` при получении файла в POST view
- `file.save` для сохранения файла
- `from werkzeug.utils import secure_filename` - дополнительное преобразование имени для безопасности





# Несколько полезных функций



## Обработка ошибок



### Декоратор `errorhandler`

```
@app.errorhandler(404)
def page_not_found(e):
    ...
    return
render_template('404.html',
**context), 404
```



### Функция `abort`

```
if result is None:
    abort(404)
```



### Некоторые коды ошибок

- 400: Неверный запрос
- 401: Не авторизован
- 403: Доступ запрещен
- 404: Страница не найдена
- 500: Внутренняя ошибка сервера

## Перенаправления

Перенаправления в Framework Flask позволяют перенаправлять пользователя с одной страницы на другую

```
from flask import redirect, url_for  
...
```

```
@app.route('/redirect/')  
def redirect_to_index():  
    return redirect(url_for('index'))  
...
```







## Flash сообщения

Flash сообщения в Flask являются способом передачи информации между запросами

- Функция `flash()` принимает сообщение и категорию, к которой это сообщение относится, и сохраняет его во временном хранилище.
- `app.secret_key` обеспечивает работу flash в рамках сессии.
- Функция `get_flashed_messages` внутри шаблона возвращает все сообщения, переданные через flash и пока не отображённые.





# Хранение данных

## Работа с cookie файлами в Flask

**Cookie файлы** - это небольшие текстовые файлы, которые хранятся в браузере пользователя и используются для хранения информации о пользователе и его предпочтениях на сайте.

```
...  
response = make_response("Cookie установлен")  
response.set_cookie('username', 'admin')  
...  
name = request.cookies.get('username')  
...
```

Функция `make_response()` создаёт объект ответа. Если возвращать не результат её работы, функция неявно создаёт объект ответа.





## Сессии

Сессии в Flask являются способом сохранения данных между запросами. Это может быть полезно, например, для хранения информации о пользователе после авторизации или для сохранения состояния формы при перезагрузке страницы.

```
from flask import session
...
app.secret_key = '42'
...
session['username'] = request.form.get('username')
...
session.pop('username', None)
...
```








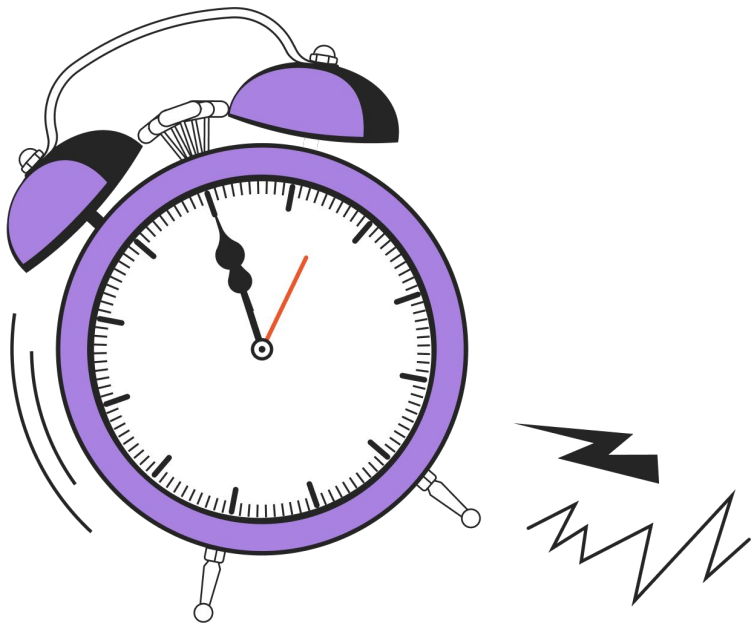


# Итоги занятия



## На этой лекции мы

-  Узнали про экранирование пользовательских данных
-  Разобрались с генерацией url адресов
-  Изучили обработку GET и POST запросов
-  Узнали несколько полезных функций Flask
-  Разобрались с cookie файлами и сессиями



## Задание

1. Для закрепления материалов лекции попробуйте самостоятельно набрать и запустить демонстрируемые примеры.



Спасибо за внимание