ДЕТСКАЯ ОНЛАЙН-ШКОЛА ПРОГРАММИРОВАНИЯ
HELLO WORLD

#ШПАРГАЛОЧКИ

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ PYTHON

Продвинутый уровень

Материалы подготовлены отделом методической разработки

Больше полезных материалов и общения в нашем комьюнити в Telegram: https://t.me/hw_school

Книга отзывов. Часть 2

GET-запрос – это такой запрос, при котором все данные на сервер передаются в строке запроса. Под строкой запроса имеется в виду адрес, по которому мы переходим в браузере.

Чтобы форма начала отправлять данные через GET-запрос, нужно указать ей атрибут **method**. А находящимся в ней полям ввода нужно указать **имя**:

```
<form method=“GET”>
    <input type=“email” name=“email”>
</form>
```

И тогда в поисковой строке после отправки формы мы увидим что-то вроде (знак процента и число - замена спец.символов, в нашем случае @):

http://127.0.0.1:8000/?email=example%40example.com

Теперь в **views.py** мы можем получить доступ к данным, отправленным в GET-запросе, через словарь **GET** из **request**:

email = request.GET.get('email')

Важно! Ключ в словаре **GET** совпадает со значением атрибута **name** у соответствующего поля ввода.

Для удобной работы с формами в Django есть два класса - Form и ModelForm. Для начала работы с ними нужно создать файл **forms.py** в папке приложения и импортировать нужную библиотеку:

```
from django import forms
```

Остается создать свой класс - наследник **Form** и прописать в нем необходимые поля:

```
class ReviewForm(forms.Form):
    name = forms.CharField(max_length=25)
    email = forms.EmailField()
    review = forms.CharField(widget=forms.TextArea)
```

Теперь в **views.py** нужно импортировать класс формы, создать его объект в функции-представлении и передать форму в шаблон:

```
form reviews.forms import ReviewForm

def review(request):
    form = ReviewForm()
    …
    return render(request, “index.html”, {... , ‘form’: form})
```

POST-запрос - более безопасный метод отправки данных. Данные из форм лучше отправлять именно им. Для этого нужно поменять атрибут формы, а еще - добавить в нее команду для защиты от подделки межсайтовых запросов:

```
<form method="POST">
    {% csrf-token %}
    …
</form>
```

Теперь мы не увидим данные в поисковой строке, но и в views.py получить их не сможем.

Поскольку теперь мы работаем и с GET, и с POST запросами, в представление нужно добавить проверку запроса пользователя, и обрабатывать его в зависимости от типа запроса:

```
if request.method == “GET”:
    # старый код
elif request.method == “POST”:
    form = ReviewForm(request.POST)
    if form.is_valid(): # проверка, валидна ли форма
        data = form.cleaned_data
        return redirect('reviews') # чтобы не было повторной отправки формы
    else:
        # то же, что и в GET
```

CSV (comma-separated values) - формат, позволяющий хранить данные в обычном текстовом файле с расширением **.csv**. Запись в файл:

```
if form.is_valid():
    data = form.cleaned_data
    name = data.get('name')
    ...
    with open('data.csv', 'a') as file:
        file.write(f'{name}|{email}|{review}')
```

Получение данных из файла:

```
with open('data.csv') as file:
    reviews = file.readlines() # в каждой строке - один отзыв
    if len(reviews) > 0:
        review_data_1 = reviews[0] # получаем первый отзыв
        review_data_1 = review_data_1.split('|') # разбиваем строку по разделителю |
        name_1 = review_data_1[0] # извлекаем первое значение (имя)
```