

## Разработка программного обеспечения на языке Python

[Обзорная панель](#) ▶ [Мои курсы](#) ▶ [Разработка ПО на языке Python](#) ▶ [Веб-программирование на Python](#) ▶

[Лекция 3. Шаблонизация в django](#)

### Лекция 3. Шаблонизация в django

Посмотрите видеоуроки и ответьте на контрольные вопросы после лекции

#### Формы в Django



#### Добавление формы в шаблон

index.html



00:00 / 04:25



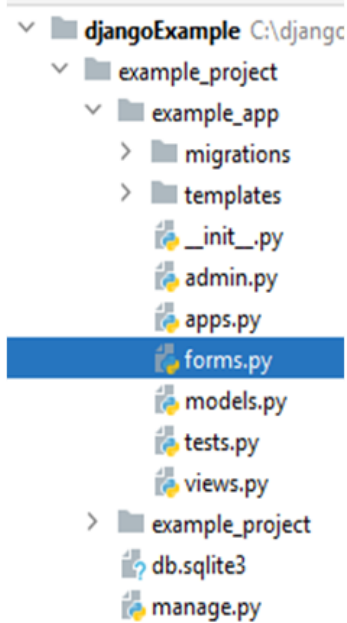
Рассмотрим способ отправления пользователем данных на сервер веб-приложения джанго.

В HTML форма - это набор элементов внутри тегов `<form>...</form>`, которые позволяют пользователю выполнять такие действия, как ввод текста, выбор опций, манипулирование объектами или элементами управления и так далее, а затем отправлять эту информацию обратно на сервер.

Форма обычно включает поля ввода данных и кнопку, при нажатии которой введенные данные отправляются на сервер. Django предоставляет ряд инструментов и библиотек, которые помогут создать формы, принимающие ввод от посетителей сайта, а затем обрабатывающие и отвечающие на него.

Формы могут быть описаны в шаблонизаторе django, html файлах. В данном случае они представляют собой набор полей внутри тега `<form>`, среди которых должен быть элемент кнопки, отвечающий за отправление данных на сервер. Кроме того, Django предоставляет специальные возможности по работе с формами, которые рассмотрим подробнее.

Во этом случае формы ввода данных описываются в виде классов. Классы размещаются внутри приложения, где они используются. Нередко они помещаются в отдельный файл, который называется, к примеру, `forms.py`.



Например, создадим в приложении новый файл **forms.py** и создадим в нем класс для формы с названием **UserForm**. Каждая форма определяется в виде отдельного класса, который расширяет класс **forms.Form**.

```
forms.py x
1  from django import forms
2
3
4  class UserForm(forms.Form):
5      name = forms.CharField()
6      age = forms.IntegerField()
7
```

В нашем примере он определяет два поля. Поле **name** представляет тип **CharField** и будет генерировать поле типа "text". Поле **age** представляет тип **IntegerField** и будет генерировать поле для ввода числа. То есть первое поле для ввода текста, а второе для ввода чисел.

Далее в файле **views.py** определим следующее представление. Здесь создаем объект класса формы и передаем его в шаблон **index.html** в виде переменной **form**.

```
views.py x
1  from django.shortcuts import render
2  from .forms import *
3
4
5  def index(request):
6      form = UserForm()
7      return render(request, "index.html", context={'form': form})
-
```

Далее добавим форму в виде переменной, переданной из контекста в шаблон **index.html**:

```
index.html x
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <title>Example project</title>
6 </head>
7 <body>
8 <form method="POST">
9 | {% csrf_token %}
10 <table>
11 | {{ form }}
12 </table>
13 <input type="submit" value="Отправить" >
14 </form>
15 </body>
16 </html>
```

Для создания формы здесь использован элемент html `<form>`. В начале формы помещен встроенный тег Django `{% csrf_token %}`, который позволяет защитить приложение от CSRF-атак, добавляя в форму в виде скрытого поля csrf-токен. Внизу формы определена кнопка для отправки данной формы на сервер. Из переменной `form` будет генерироваться код html, соответствующий описанию полей формы. Таким образом, в браузере будем видеть поля для ввода данных и кнопку для их отправления на сервер.

Представления обрабатывают **get и post запросы**.

Get – Клиент получает данные с сервера, часто в виде html страницы.

Post – Клиент отправляет данные с формы на сервер.

Теперь изменим представление в файле `views.py`. Поскольку в шаблоне форма по умолчанию будет отправляться на тот же адрес, то представление обрабатывает сразу два типа запросов GET и POST. Для определения типа запроса проверяем значение `request.method`.

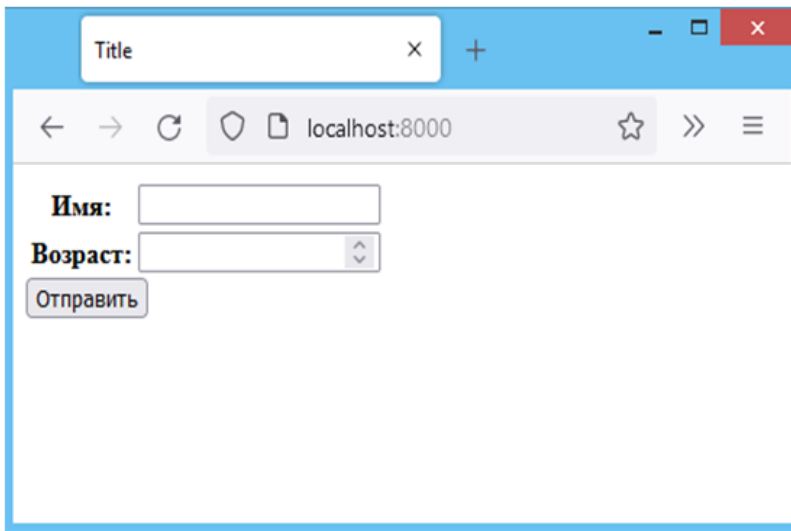
```
views.py x
1 from django.shortcuts import render
2 from django.http import HttpResponse, HttpResponseRedirect
3 from .forms import *
4
5
6 def index(request):
7     # POST запрос - пользователь отправляет данные с формы на сервер
8     if request.method == "POST":
9         name = request.POST.get("name") # получение значения поля name
10        age = request.POST.get("age") # получение значения поля age
11        return HttpResponse(f"<h2>Hello, {name}, {age} years</h2>")
12    else:
13        # GET запрос - пользователю получает страницу с формой
14        userform = UserForm()
15        return render(request, "index.html", {"form": userform})
16
```

Если запрос типа POST, то вначале создаем объект `UserForm`, заполним его данными, которые пришли в запросе через `request.POST`. То есть это и будут отправленные данные формы. Затем мы можем получить эти данные по отдельности для каждого поля формы. После этого отправляем пользователю сообщение через объект `HttpResponse`. В принципе тут можно было бы сделать переадресацию или использовать другой шаблон для генерации ответа.

Если запрос представляет тип GET, то просто отправляем форму для ввода данных.

Таким образом, при обращении к приложению мы вначале увидим форму ввода. Введем в нее некоторые данные. После нажатия на кнопку введенные данные в запросе POST опять же уйдут представлению `index`, которое обработает их и в ответ отправить пользователю сообщение с введенным именем.

После запуска приложения перейдем в веб браузер. Здесь видим, что на страницу была добавлена форма для ввода данных с указанными полями.



В этой теме были представлены сведения о формах. Мы узнали, как создать форму, и отправить с ее помощью данные клиента для дальнейшей обработки.

Настройка форм

ПРЕДЫДУЩИЙ ЭЛЕМЕНТ КУРСА

◀ [Задание 3. Создание своего первого сайта на Django](#)

Перейти на...

СЛЕДУЮЩИЙ ЭЛЕМЕНТ КУРСА

[Задание 4. Создание страниц веб-приложения](#) ▶

© 2010-2023 Центр обучающих систем  
Сибирского федерального университета, sfu-kras.ru

Разработано на платформе moodle  
Beta-version (3.9.1.5.w3)

[Политика конфиденциальности](#)

[Соглашение о Персональных данных](#)

[Политика допустимого использования](#)

Контакты +7(391) 206-27-05  
[info-ms@sfu-kras.ru](mailto:info-ms@sfu-kras.ru)

[Скачать мобильное приложение](#)

[Инструкции по работе в системе](#)