

Классы представлений, регистрация, оптимизация

Классы представлений: ListView, DetailView, CreateView

Основы ORM Django за час

Mixins - убираем дублирование кода

Постраничная навигация (пагинация)

Регистрация пользователей на сайте

Делаем авторизацию пользователей на сайте

Оптимизация сайта с Django Debug Toolbar

Включаем кэширование данных

Использование капчи captcha

Тонкая настройка админ панели

Поделиться

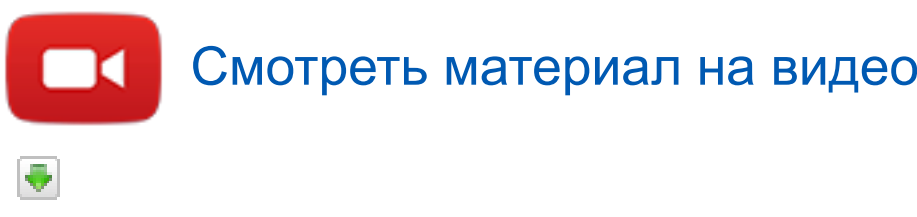


Наш канал



Главная → Django → Классы представлений, регистрация, оптимизация

## Использование капчи captcha



Архив проекта: [lesson-23-coolsite.zip](#)

На данный момент у нас с вами получился уже полноценный сайт, но одна страница все еще не сделана – это форма обратной связи. Давайте ее добавим. Для начала вместо функции contact пропишем следующий класс представления (в файле women/views.py):

```
class ContactFormView(DataMixin, FormView):
    form_class = ContactForm
    template_name = 'women/contact.html'
    success_url = reverse_lazy('home')

    def get_context_data(self, *, object_list=None, **kwargs):
        context = super().get_context_data(**kwargs)
        c_def = self.get_user_context(title="Обратная связь")
        return dict(list(context.items()) + list(c_def.items()))

    def form_valid(self, form):
        print(form.cleaned_data)
        return redirect('home')
```

Здесь все стандартно: определяем атрибуты form\_class, template\_name и success\_url, а также методы: get\_context\_data и form\_valid. Последний метод я прописал, чтобы продемонстрировать возможность обработки данных формы. Если пользователь все поля заполнил корректно, то будет вызван метод form\_valid(), в консоли увидим словарь с данными формы и выполнится перенаправление на главную страницу.

Далее, в файле women/forms.py пропишем класс формы:

```
class ContactForm(forms.Form):
    name = forms.CharField(label='Имя', max_length=255)
    email = forms.EmailField(label='Email')
    content = forms.CharField(widget=forms.Textarea(attrs={'cols': 60, 'rows': 10}))
```

Здесь тоже все так, как мы делали ранее, просто определены три поля для ввода.

Создадим шаблон contact.html с содержимым:

```
{% extends 'women/base.html' %}

{% block content %}
<h1>{{title}}</h1>

<form method="post">
    {% csrf_token %}

    <div class="form-error">{{ form.non_field_errors }}</div>

    {% for f in form %}
    <p><label class="form-label" for="{{ f.id_for_label }}">{{f.label}}: </label>{{ f }}</p>
    <div class="form-error">{{ f.errors }}</div>
    {% endfor %}

    <button type="submit">Отправить</button>

</form>

<p>{% endblock %}
```

И в файле women/urls.py свяжем маршрут contact с классом ContactFormView:

```
path('contact/', ContactFormView.as_view(), name='contact'),
```

Все, страница готова и полностью функциональна. Как вы понимаете, мы ее сделали не случайно. Следующим шагом добавим к этой форме так называемую капчу, то есть, графическую картинку с кодом, который нужно ввести, чтобы отправить данные формы на сервер. Это часто делают для защиты от ботов и на данный момент необходимый элемент всех открытых, публичных форм. Давайте это сделаем.

Вариантов для капчи множество и для фреймворка Django был разработан специальный модуль, который называется:

django-simple-captcha

Для его установки достаточно прописать команду:

```
pip install django-simple-captcha
```

и модуль готов к использованию. Ниже в описании модуля есть ссылка на документацию:

<https://django-simple-captcha.readthedocs.io/en/latest/>

где описывается как использовать этот пакет в своем приложении. Сделаем это. Вначале в файле settings.py пропишем этот модуль в списке установленных приложений:

```
INSTALLED_APPS = [
    ...
    'captcha',
    ...
]
```

Далее, по документации нам нужно выполнить миграцию:

```
python manage.py migrate
```

Затем, в корневой список маршрутов добавим строчку:

```
urlpatterns = [
    ...
    path('captcha/', include('captcha.urls')),
    ...
]
```

Все, привязка модуля к нашему приложению завершена и мы можем использовать ее в нашей форме. Для этого в файле women/forms.py импортируем класс для формирования поля капчи:

```
from captcha.fields import CaptchaField
```

и в классе ContactForm пропишем его:

```
captcha = CaptchaField()
```

По идее, это все, по умолчанию наша форма теперь имеет капчу и давайте посмотрим как все это будет работать. Переходим в браузер, обновляем страницу обратной связи и видим дополнительно еще и капчу. Соответственно, она будет меняться при каждом обновлении страницы.


Если нужен другой вид капчи, оформить ее в своих стилях, то для этого используется дополнительно класс CaptchaTextInput, пример представлен в документации:

<https://django-simple-captcha.readthedocs.io/en/latest/advanced.html>


Кроме того, если требуется поменять шрифт, или его размер, цвета и прочее, то можно прописывать соответствующие переменные (приведены там же) в файле settings.py.


Вот так в самом простом варианте можно подключать капчу к своим формам.

### Видео по теме





#1. Django - что это такое, порядок установки







#2. Модель MTV. Маршрутизация. Функции представления







#3. Маршрутизация, обработка исключений запросов,







#4. Определение моделей. Миграции: создание и выполнение







#5. CRUD - основы ORM по работе с моделями






#6. Шаблоны (templates). Начало





#7. Подкли статически: Фильтры ш



←

→

←

→