Django Form

В Django существует два вида форм:

1. Не связанные с моделью — родительским классом такой формы является класс Form

2. Связанные с моделью — родительскими класса такой формы являются классы Form и Model

Не связанные с моделью формы используются для сбора информации и их обработки, не требующей записи в базу данных, пример такой формы можно считать калькулятор на сайте для расчета объема отправляемой посылки

Связанные с моделью формы, зачастую применяются для заполнения базу дынных

Способы передачи данных из формы на сервер:

Для передачи данных из формы используются два метода HTTP: GET, POST

При передачи данных из формы с помощью метода GET, данные из формы генерируют URL запрос

При передачи данных из формы с помощью метода POST, данные из формы передаются в теле запроса в формате JSON

Описание формы:

Для описания форм, принято создавать отдельный файл: *forms.py*

Описание формы очень похоже на описание модели:

field_name = Widget(arguments)

Каждому типу данных при описании модели БД, соответствует 1 или несколько виджетов:

Родительским классом всех виджетов является класс **Widget**

Атрибуты виджетов:

attrs: dict — словарь HTML атрибутов для передачи стилей

Так же каждый виджет, в зависимости от типа, содержит свои атрибуты

Основные атрибуты класса Widget:

widgets: dict — Словарь, содержащий HTML атрибуты, необходим для передачи стилей

supports_microseconds: bool — атрибут, указание которого необходимо для виджетов даты и времени, указывает на отображение микросекунд

Дополнительные атрибуты класса ModelForm:

model — ссылка на модель с которой связана данная форма

fields: list[str] — список имен атрибутов, которые будут доступны в форме для заполнения

Обработка данных из формы:

При получении данных из формы, если метод отправки **POST**, то содержимое формы находится в атрибуте **POST** аргумента **request**

Для валидации данных, создается экземпляр формы, заполненный на основании атрибута **POST**

form = MyForm(request.POST)

Далее мы можем вызвать метод **is_valid()** для проверки данных в форме на соответствие ограничениям

Если форма не связана с моделью, мы можем дальше обрабатывать, производить какие-либо расчеты, так как атрибут **POST** является типом данных **QueryDict**, доставать значения из словаря можно только с помощью **get()**

Если форма связана с моделью и никакие обработки перед записью не нужны, мы можем вызвать метод **save()** для сохранения данных в БД

Отображение форм на HTML:

Все формы передаются на шаблон с помощью словаря контекта

Все формы отображаются в теге **<form></form>**

Открывающийся тег формы обязан принимать аргумент **method** со значением типа отправки данных (get, post)

Для работоспособности формы, внутри тега форм, обязательно указание в любом месте

{% csrf_token %} - данный токен обеспечивает безопасность данных, каждый раз он генерируется новый

Для отправки данных, обязательно наличие кнопки внутри формы

Для отображения формы достаточно вывести форму с помощью шаблонизатора
{{ form_name }}

Данный способ отображения выводит форму в строку, что не удобно для дальнейшей работы с ней в шаблоне, поэтому мы можем отрисовывать каждое поле формы отдельно в нужном месте указав имя атрибута

{{ form_name.field_name }}

Загрузка медиа файлов:

Для загрузки медиа файлов с помощью Админ панели или с помощью формы, необходимо создать папку **media** в корне проекта, а так же объявить 2 переменный внутри файла **settings.py**

MEDIA_ROOT = os.path.join(BASE_DIR, „media“) — полный путь до папки media

MEDIA_URL = „/media/“— URL, для использования в шаблонах

Для загрузки медиа файла, в модели описывается атрибут, хранящий в себе путь до файла:

пример: **ImageField**

Данный типы атрибутов, принимают аргумент **upload_to** — для указания папки загрузки, данный путь указывается относительно папки **media**

Отображение меди файла на шаблоне:

Для отображения медиа файла на шаблоне, у атрибута файла, необходимо вызвать атрибут **url**

{{ post.photo.url }}

Для корректной работы медиа файлов, необходимо подключить их к **urlpatterns** в главном файле диспетчеризации URL запросов

from django.conf import settings

from django.conf.urls.static import static

urlpatterns = [...]+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)