Материалы занятия

Курс: Web-разработка

Дисциплина: Создание web-приложений с использованием фреймворка Django

Тема занятия № 37: Модуль 20. Обработка выгруженных файлов

1. ПОДГОТОВКА ПОДСИСТЕМЫ ОБРАБОТКИ ВЫГРУЖЕННЫХ ФАЙЛОВ

Django предлагает удобные инструменты для обработки файлов, выгруженных посетителями: как высокоуровневые, в стиле "раз настроил — и забыл", так и низкоуровневые, для специфических случаев.

Подготовка подсистемы обработки ыыгруженных файлов

Чтобы успешно обрабатывать в своем сайте выгруженные посетителями файлы, следует выполнить некоторые подготовительные действия.

Настройка подсистемы обработки выгруженных файлов

Настройки этой подсистемы записываются в модуле settings.py пакета конфигурации. Вот наиболее интересные из них:

□ media url— префикс, добавляемый к интернет-адресу выгруженного файла.

Встретив в начале интернет-адреса этот префикс, Django поймет, что это выгруженный файл и его нужно передать для обработки подсистеме выгруженных файлов. По умолчанию — ’’пустая” строка;

□ media root— полный путь к папке, в которой будут храниться выгруженные файлы. По умолчанию — ’’пустая” строка.

Это единственные обязательные для указания параметры подсистемы выгруженных файлов. Вот пример их задания:

```
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'
```

Значение переменной base dir вычисляется в том же модуле settings.py и представляет собой полный путь к папке проекта;

□ file upload handlers— последовательность имен классов обработчиков выгрузки (обработчик выгрузки извлекает файл из отправленных посетителем данных и временно сохраняет его на диске серверного компьютера или в оперативной памяти). В Django доступны два класса обработчика выгрузки, объявленные в модуле django. Core, files .uploadhandler:

• memoryfiieupioadhandier— сохраняет выгруженный файл в оперативной памяти и задействуется, если размер файла не превышает 2,5 Мбайт (это значение настраивается в другом параметре);
• temporaryfiieupioadhandier— сохраняет выгруженный файл на диске серверного компьютера в папке для временных файлов. Задействуется, если размер выгруженного файла больше 2,5 Мбайт.
Значение по умолчанию — список с именами обоих классов, которые выбираются автоматически, в зависимости от размера выгруженного файла;
□ file upload max memory size — максимальный размер выгруженного файла, сохраняемого в оперативной памяти, в байтах. Если размер превышает это значение, то файл будет сохранен на диске. По умолчанию: 2621440 байтов (2,5 Мбайт);
□ file upload temp dir — полный путь к папке, в которой будут сохраняться файлы, размер которых превышает указанный в параметре file_upload_max_Memory size. Если задано значение None, то будет использована стандартная папка для хранения временных файлов в операционной системе. По умолчанию — None;
□ file upload permissions — числовой код прав доступа, даваемых выгруженным файлам. Если задано значение None, то права доступа даст сама операционная система — в большинстве случаев это будут права ообоо (владелец может читать и записывать файлы, его группа и остальные пользователи вообще не имеют доступа к файлам). По умолчанию — 0об44 (владелец может читать и записывать файлы, его группа и остальные пользователи — только читать их);
На заметку!
В версиях Django, предшествующих 3.0, параметр file upload permissions имел значение по умолчанию None.
□ file_upload_directory_permissions — числовой код прав доступа, даваемых папкам, которые создаются при сохранении выгруженных файлов. Если задано значение None, то права доступа даст сама операционная система, — в большинстве случаев это будут права ообоо (владелец может читать и записывать файлы в папках, его группа и остальные пользователи вообще не имеют доступа к папкам). По умолчанию — None;
□ default file storage— имя класса файлового хранилища, используемого по умолчанию, в виде строки (файловое хранилище обеспечивает сохранение файла в выделенной для этого папке, получение его интернет-адреса, параметров и пр.).
По умолчанию: ’’django.core, files. Storage. Filesystemstorage” (это единственное файловое хранилище, поставляемое в составе Django).

Указание маршрута для выгруженных файлов
Практически всегда посетители выгружают файлы на сайт для того, чтобы показать их другим посетителям. Следовательно, на веб-страницах сайта позже будут выводиться сами эти файлы или указывающие на них гиперссылки. Для того чтобы посетители смогли их просмотреть или загрузить, нужно создать соответствующий маршрут.
Маршрут, указывающий на выгруженный файл, записывается в списке уровня проекта, т. е. в модуле urls.py пакета конфигурации. Для его указания используется функция static о из модуля django.conf .urls.static, которая в этом случае вызывается в следующем формате:

```
static(<префикс>, document_root=<путь к папке с выгруженными файлами>)
```

Она создает маршрут, связывающий:
□ шаблонный путь формата:
<префикс, заданный в вызове функции>путь к выгруженному файлу>
□ контроллер-функцию serve () ИЗ модуля django.views.static, выдающую файл с заданным путем из указанной папки;
□ папку с путем, заданным в параметре document root, в которой хранятся выгруженные файлы.
В качестве результата возвращается список с единственным элементом — описанным ранее маршрутом. Если сайт работает в эксплуатационном режиме, то возвращается пустой список.
В нашем случае в качестве префикса следует указать значение параметра media url, а в качестве пути к папке — значение параметра media root настроек проекта (их можно получить из модуля settings.py пакета конфигурации):

```
from django.conf.urls.static import static
from django.conf import settings
. . .
urlpatterns = [
    . . .
]
urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Функция static о создает маршрут только при работе в отладочном режиме. После перевода сайта в эксплуатационный режим для формирования маршрута придется использовать другие средства.

2. ХРАНЕНИЕ ФАЙЛОВ В МОДЕЛЯХ

Предоставляемые Django высокоуровневые средства для обработки выгруженных файлов предполагают хранение таких файлов в полях моделей и подходят в большинстве случаев.

Типы полей модели, предназначенные для хранения файлов

Для хранения файлов в моделях Django предусматривает два типа полей, представляемые описанными далее классами из модуля django.db.models:
□ filefieid— файл любого типа. Фактически хранит путь к выгруженному файлу, указанный относительно папки, путь к которой задан в параметре media root настроек проекта.
Необязательный параметр max length указывает максимальную длину заносимого в поле пути в виде целого числа в символах (по умолчанию: юо).
Необязательный параметр upload to задает папку, в которой будет сохранен выгруженный файл и которая должна находиться в папке, чей путь задан в параметре media root. В качестве значения параметра можно указать:
• строку с путем, заданным относительно пути из параметра media root—файл будет выгружен во вложенную папку, находящуюся по этому пути:

```
archive = models.FileField(upload_to='archives/')
```

В формируемом пути можно использовать составные части текущих даты и времени: год, число, номер месяца и т. П., — вставив в строку с путем специальные символы, поддерживаемые функциями strftimeo и Strptimeo Python. Перечень этих специальных символов можно найти в документации по Python или на странице https://docs.python.Org/3/Library/datetime.htmltfstrftime-strptime-behavior. Пример:

```
archive = models.FileField(upload_to='archives/%Y/%m/%d/')
```

В результате выгруженные файлы будут сохраняться во вложенных папках с именами формата archives\<год>\<номер месяца>\<число>^ где под, номер месяца и число взяты из текущей даты.

Чтобы выгруженные файлы сохранялись непосредственно в папку из параметра media root, достаточно указать в параметре upload to ’’пустую” строку.

Это, кстати, его значение по умолчанию;

• функцию, возвращающую путь для сохранения файла, который включает и имя файла — файл будет сохранен во вложенной папке, расположенной по полученному от функции пути, под полученным именем. Функция должна принимать два параметра: текущую запись модели и изначальное имя выгруженного файла.

Этот способ задания пути сохранения можно использовать для сохранения файлов под какими-либо отвлеченными именами, сформированными на основе, например, текущей временной отметки. Вот пример такой функции:

```
from datetime import datetime
from os.path import splitext
def get_timestamp_path(instance, filename):
    return '%s%s' % (datetime.now().timestamp(),
                     splitext(filename)[1])
. . .
file = models.FileField(upload_to=get_timestamp_path)
```

□ imagefieid — графический файл. Фактически хранит путь к выгруженному файлу, указанный относительно папки из параметра media root настроек проекта.

Внимание!

Для успешной обработки полей типа imagefieid необходимо установить дополнительную библиотеку Pillow (если она не была установлена ранее). Сделать это можно подачей команды:

```
pip install pillow
```

Поддерживаются дополнительные параметры max iength и upload to, описанные ранее, а также следующие параметры:
• width fieid — имя поля модели, в которое будет записана ширина изображения из выгруженного файла. Если не указан, то ширина изображения нигде храниться не будет;
• height fieid —имя поля модели, в которое будет записана высота изображения из выгруженного файла. Если не указан, то высота изображения нигде храниться не будет.
Эти поля будут созданы самим Django. Можно указать создание как обоих полей, так и лишь одного из них (если зачем-то понадобится хранить только один размер изображения).
Приведен код модели, в которой присутствует поле типа imagefieid.

```
from django.db import models

class Img(models.Model):
    img = models.ImageField(verbose_name='Изображение',
                            upload_to=get_timestamp_path)
    desc = models.TextField(verbose_name='Описание')

    class Meta:
        verbose_name='Изображение'
        verbose_name_plural='Изображения'
```

Поля форм, валидаторы и элементы управления, служащие для указания файлов
По умолчанию поле модели filefieid представляется в форме полем типа Filefieid, а поле модели imagefield— полем формы imagefield. Оба ЭТИХ типа полей формы объявлены в модуле django. Forms:
□ filefieid — поле для ввода файла произвольного типа. Дополнительные параметры:
• max length— максимальная длина пути к файлу, заносимого в поле, в символах;
• aiiow empty file— если True, то к выгрузке будут допускаться даже ’’пустые” файлы (с нулевым размером), если False — только файлы с содержимым (ненулевого размера). По умолчанию — False;
□ imagefield— поле для ввода графического файла. Поддерживаются дополнительные параметры max length и aiiow_empty_fне, описанные ранее.
Помимо валидаторов, в полях этих типов можно использовать следующие, объявленные В модуле django.core.validators:
□ fiieextensionvaiidator — класс, проверяет, входит ли расширение сохраняемого в поле файла в список допустимых. Формат конструктора:

```
FileExtensionValidator(allowed_extensions=<допустимые расширения>[,
                       message=None][, code=None])
```

Он принимает следующие параметры:
• aiiowed extensions— последовательность, содержащая допустимые расширения файлов. Каждое расширение представляется в виде строки без начальной точки;
• message— строка с сообщением об ошибке. Если не указан, то выводится стандартное сообщение;

• code— код ошибки. Если не указан, то используется код по умолчанию "invalid_extension";
□ validate_image_file_extension — переменная, хранит экземпляр класса Fiieextensionvaiidator, настроенный считать допустимыми только расширения графических файлов. За основу берется список форматов файлов, поддерживаемых библиотекой Pillow.
Также поддерживаются дополнительные коды ошибок П "invalid" — Применительно К ПОЛЮ типа filefieid ИЛИ imagefield Сообщает, что задан неверный метод кодирования данных для формы. Следует указать метод muitipart/form-data, воспользовавшись атрибутом enctype тега <form>;
□ "missing” —файл по какой-то причине не был выгружен;
□ "empty" — выгруженный файл ’’пуст” (имеет нулевой размер);
□ "contradiction" — следует либо выбрать файл для выгрузки, либо установить флажок удаления файла из поля, но не одновременно;
□ "invalid extension" — расширение выбранного файла не входит в список допустимых;
□ "invalid image" — графический файл сохранен в неподдерживаемом формате или поврежден.
Для указания выгружаемых файлов применяются такие классы элементов управления ИЗ модуля django. Forms .widgets:
□ Fileinput —обычное поле ввода файла;
□ ciearabiefiieinput — поле ввода файла с возможностью очистки. Представляется как комбинация обычного поля ввода файла и флажка очистки, при установке которого сохраненный в поле файл удаляется.
Приведен код формы, связанной с моделью img и включающей поле для выгрузки графического изображения imagefieid.

```
from django import forms
from django.core import validators

from .models import Img

class ImgForm(forms.ModelForm):
    img = forms.ImageField(label='Изображение',
          validators=[validators.FileExtensionValidator(
          allowed_extensions=('gif', 'jpg', 'png'))],
          error_messages={
                'invalid_extension': 'Этот формат не поддерживается'})
    desc = forms.CharField(label='Описание',
                           widget=forms.widgets.Textarea())

    class Meta:
        model = Img
        fields = '__all__'
```

Обработка выгруженных файлов в контроллерах осуществляется так же, как обработка любых других данных, полученных от посетителя. Есть лишь два момента, которые нужно иметь в виду:

□ при выводе формы на экран необходимо указать для нее метод кодирования данных multipart/form-data, воспользовавшись атрибутом enctype тега <form>:

```
<form . . . enctype="multipart/form-data">
```

Если этого не сделать, то файл не будет выгружен;
□ при повторном создании формы конструктору ее класса вторым позиционным параметром следует передать значение атрибута files объекта запроса. Этот атрибут содержит словарь со всеми выгруженными из формы файлами.
Приведен код контроллера, сохраняющего выгруженный графический файл в модели. Для выгрузки файла используется форма imgform

```
from django.shortcuts import render, redirect

from .models import Img
from .forms import ImgForm

def add(request):
    if request.method == 'POST':
        form = ImgForm(request.POST, request.FILES)
        if form.is_valid():
            form.save()
            return redirect('testapp:index')
    else:
        form = ImgForm()
    context = {'form': form}
    return render(request, 'testapp/add.html', context)
```

Сохранение выгруженного файла выполняет сама модель при вызове метода save () связанной с ней формы. Нам самим заниматься этим не придется.
Если для выгрузки файла используется форма, не связанная с моделью, то нам понадобится самостоятельно занести выгруженный файл в нужное поле записи модели. Этот файл можно найти в элементе словаря, хранящегося в атрибуте Cleaned data объекта формы. Вот пример:

```
form = ImgNonModelForm(request.POST, request.FILES)
if form.is_valid():
    img = Img()
    img.img = form.cleaned_data['img']
    img.desc = form.cleaned_data['desc']
    img.save()
```

И в этом случае сохранение файла выполняется самой моделью.
Аналогичным способом можно сохранять сразу нескольких выгруженных файлов.
Сначала следует указать для поля ввода файла возможность выбора произвольного количества файлов, добавив в создающий его тег <input> атрибут без значения Multiple. Пример:

```
class ImgNonModelForm(forms.Form):
    img = forms.ImageField( . . .
       widget=forms.widgets.ClearableFileInput(attrs={'multiple': True}))
    . . .
```

Сложность в том, что элемент словаря из атрибута cieaned data хранит лишь один из выгруженных файлов. Чтобы получить все файлы, нужно обратиться непосредственно к словарю из атрибута files объекта запроса и вызвать у него метод Getiist(<элемент словаря>). В качестве результата он вернет последовательность файлов, хранящихся в указанном элементе. Вот пример:

```
form = ImgNonModelForm(request.POST, request.FILES)
if form.is_valid():
    for file in request.FILES.getlist('img'):
        img = Img()
        img.desc = form.cleaned_data['desc']
        img.img = file
        img.save()
```

Вывод выгруженных файлов
При обращении непосредственно к полю типа fiiefieid, хранящему выгруженный файл, мы получим экземпляр класса fieidfiie, содержащий различные сведения о выгруженном файле. Он поддерживает следующие атрибуты:
□ url — интернет-адрес файла:

```
{% for img in imgs %}
    <div><img src="{{ img.img.url }}"></div>
    <div><a href="{{ img.img.url }}">Загрузить картинку</a></div>
{% endfor %}
```

□ name — путь к файлу относительно папки, в которой он сохранен (путь к этой папке указывается в параметре media root настроек проекта);
□ size — размер файла в байтах.
При обращении К ПОЛЮ imagefieid МЫ получим экземпляр класса imagefieldfile.
Он является производным от класса fieidfiie, поддерживает все его атрибуты и добавляет два своих собственных:
□ width — ширина хранящегося в файле изображения в пикселах;
□ height — высота хранящегося в файле изображения в пикселах.

Удаление выгруженного файла
К сожалению, при удалении записи модели, в которой хранится выгруженный файл, сам этот файл удален не будет. Нам придется удалить его самостоятельно.
Для удаления файла применяется метод delete ([save=True]) класса fieldfile. Помимо этого, он очищает поле записи, в котором хранится файл. Необязательный параметр save указывает, сохранять запись модели после удаления файла (значение True, используемое по умолчанию) или нет (значение False).
Приведен код контроллера, удаляющего файл вместе с записью модели, в которой он хранится. Этот контроллер принимает ключ удаляемой записи С URL-параметром pk.

```
from django.shortcuts import redirect

def delete(request, pk):
    img = Img.objects.get(pk=pk)
    img.img.delete()
    img.delete()
    return redirect('testapp:index')
```

Можно реализовать удаление сохраненных файлов непосредственно в классе модели, переопределив метод delete (seif, *args, **kwargs):

```
class Img(models.Model):
    . . .
    def delete(self, *args, **kwargs):
        self.img.delete(save=False)
        super().delete(*args, **kwargs)
```

3. ХРАНЕНИЕ ПУТЕЙ К ФАЙЛАМ В МОДЕЛЯХ

Поле модели, представленное классом filepathfield ИЗ модуля django.db.models, служит для хранения пути к файлу (папке), существующему на диске серверного компьютера и хранящемуся в указанной папке. Отметим, что сохранить путь к несуществующему файлу (папке) в этом поле нельзя.

Конструктор класса filepathfield поддерживает следующие дополнительные параметры:

□ path— полный путь к папке. В поле могут сохраняться только пути к файлам или папкам, вложенным в нее.

Начиная с Django 3.0, в качестве значения этого параметра может быть указана функция, не принимающая параметров и возвращающая путь к папке;

□ match— регулярное выражение, записанное в виде строки. Если указано, то в поле могут быть сохранены только пути к файлам, имена которых (не пути целиком!) Совпадают с этим регулярным выражением. По умолчанию — None (в поле можно сохранить путь к любому файлу из заданной папки);

□ recursive — если True, то в поле можно сохранить путь не только к файлу, хранящемуся непосредственно в заданной папке, но и к любому файлу из вложенных в нее папок. Если False, то в поле можно сохранить только путь к файлу из указанной папки. По умолчанию — False;

□ aiiow files — если True, то в поле можно сохранять пути к файлам, хранящимся в указанной папке, если False — нельзя (по умолчанию — True);

□ aliow_folders — если True, то в поле можно сохранять пути к папкам, вложенным в указанную папку, если False--- нельзя (по умолчанию — False).

Внимание!

Допускается указание значения True только для одного из параметров: aiiow fiies ИЛИ allow_folders.

Для выбора пути к файлу (папке) служит поле формы класса filepathfieid из модуля django.forms. Конструктор поддерживает те же самые параметры path, match, Recursive, allow_files И allow_folders.

Для представления поля типа filepathfieid на странице применяется список (элемент управления Select).