

Урок №7

Работа с файлами

Содержание занятия

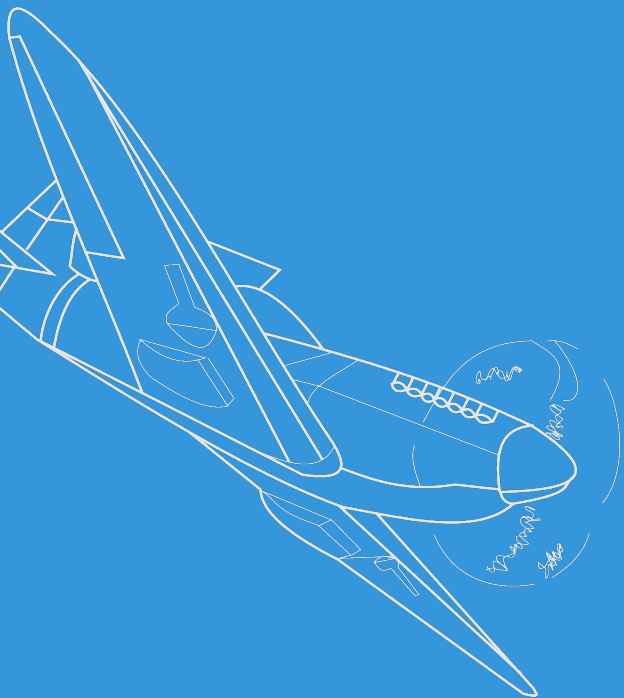
1. MIME типы;
2. Загрузка файлов из браузера;
3. Хранение файлов на локальном диске и в облаке.



Загрузка файлов из браузера



- Файлы прописанные в коде и в шаблонах, назовём их STATIC файлы (то, что добавляет разработчик);
- Файлы, который используются в коде, но известны только в процессе работы кода, назовем их MEDIA файлы (то, что добавляет пользователь);
- Пользователь загружает файл на страничке, frontend берёт содержимое и отправляет на бекэнд. Наше дело --- сохранить.



MIME-типы



MIME (Multipurpose Internet Mail Extension, Многоцелевые расширения почты Интернета) — спецификация для передачи по сети файлов различного типа: изображений, музыки, текстов, видео, архивов и др.

Основные MIME-типы



- application (внутренний формат прикладной программы). Примеры: json, xml, js, pdf.
- image (изображения). Примеры: jpeg, png, webp, gif.
- text (текст). Примеры: csv, markdown, plain, html, xml.
- multipart. Примеры: form-data, signed, encrypted.
- audio (аудио). Примеры: mpeg, aac, ogg.

Для чего нужно?

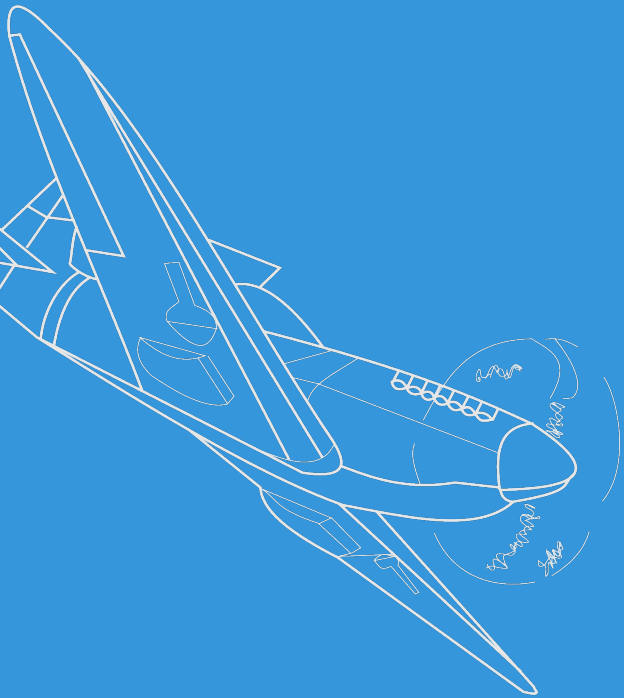


Загружать/отображать файлы определенного типа (вкладки фото/ видео/файлы в ВК, Телеграме и т.д.).

```
# Установить либу для определения mime типа.
```

```
pip3 install python-magic
```

```
def check_in_memory_mime(in_memory_file):  
    mime = magic.from_buffer(in_memory_file.read(),  
mime=True)  
    return mime
```

Хранение файлов на
локальном
хранилище и в облаке S3

Облачное хранилище Mail.Ru S3



S3 (Simple Storage Service) — онлайн-веб-служба, предоставляющая возможность для хранения и получения любого объёма данных, в любое время из любой точки сети, так называемый файловый хостинг.

S3 API — набор команд, которые «понимает» хранилище и выполняет в ответ некие действия (получение/запись файла).

Достоинства применения S3



- Высокая масштабируемость;
- Надёжность;
- Высокая скорость;
- SSL-соединение с хранилищем;
- Гарантирована на уровне 500 запросов в секунду;
- Недорогая инфраструктура хранения данных.

Виды хранилища Mail.Ru Cloud Solutions



- Icebox (для хранения редко используемых данных: архивов, резервных копий, журналов);
- Hotbox (для хранения часто используемых данных);

Django и Mail.Ru Cloud Solutions S3



Boto это набор средств разработки (SDK) от Amazon Web Services (AWS) для языка Python, позволяющая разработчикам писать программы для сервисов S3 или EC2 (Elastic Compute Cloud).

- Создать аккаунт для Cloud Storage (Облачное хранилище) и
- получить Access Key ID, Secret Key;
- Создать бакет.

Boto3



Используем boto3 библиотеку

установить библиотеку для работы с S3

```
pip3 install boto3
```

```
pip3 install django-storages
```

```
import boto3
```

```
session = boto3.session.Session()
```

```
s3_client = session.client(...)
```



Аргументы `session.client(...)`:

- `service_name='s3'`
- `endpoint_url='http://hb.bizmrg.com'`
- `aws_access_key_id='<your access key id>';`
- `aws_secret_access_key='<your secret key>'`

Можно передавать `aws_access_key_id` и `aws_secret_access_key` через перменное окружение, как-то так:

```
AWS_ACCESS_KEY_ID="<your acces key id>" AWS_SECRET_ACCESS_KEY="<your  
secret key>" python boto.py
```

Put/Get объект из S3



Успешно инициализировали `s3_client`.

```
s3_client.put_object(Bucket='<bucket name>', Key='<key>',  
Body=<content>)
```

- Одинаковые ключи по умолчанию перезаписываются;
- Решение 1: ключ --- хэш от названия файла и атрибутов пользователя;
- Решение 2: в Key можно указывать путь, например: `Key='dir1/key.txt'` ;
- В Body можно передать `BufferedReader` или содержимое файла (наш вариант).



Брать содержимое файла --- лишний трафик. URL на объект лёгкий;

- Решение: использовать `generate_presigned_url`, возвращающую валидный урл на файл.

```
url = s3_client.generate_presigned_url(  
    'get_object',  
    Params={  
        'Bucket': '<bucket name>',  
        'Key': '<key>',  
    },  
    ExpiresIn=3600  
)
```

А можно по-другому? Можно!



Можно настроить без вызова клиента s3 напрямую;

Нужно в settings.py переопределить DEFAULT_FILE_STORAGE и определить AWS_S3_ENDPOINT_URL, AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY, AWS_STORAGE_BUCKET_NAME.

Использование S3 хранилища

```
DEFAULT_FILE_STORAGE = 'storages.backends.s3boto3.S3Boto3Storage'
```

Использование локального хранилища

```
DEFAULT_FILE_STORAGE = 'django.core.files.storage.FileSystemStorage'
```

Файлы в модели (1)

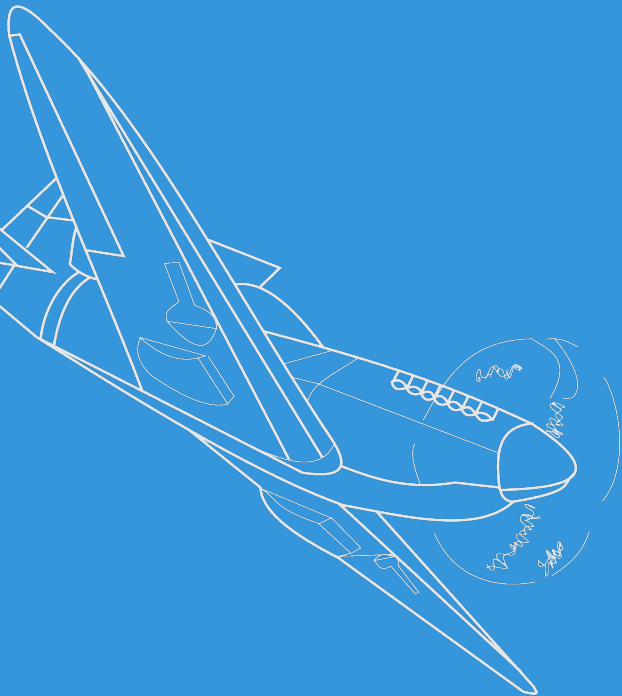


- Обработчики загрузки определены в настройке `FILE_UPLOAD_HANDLERS`;
- `MemoryFileUploadHandler` и `TemporaryFileUploadHandler` определяют обработку файлов по умолчанию в Django, загрузка небольших файлов в память и больших на диск.

Файлы в модели (2)



- Использовать внутри модели поле `models.FileField()`;
- Можно указать параметр `upload_to` --- папка, куда Django будет складывать файлы;
- Выбор места для хранения файлов задаётся в настройках `DEFAULT_FILE_STORAGE`;
- Можно для разных полей переопределить разные хранилища при помощи параметра `storage`.



Кеширование и контроль доступа к файлам в Nginx

Кеширование в nginx



- Не генерировать постоянно одни и те же скрипты;
- Приложение генерирует страницу один раз, и результат сохраняется в память;
- Периодически (time to live --- ttl) сохраненная версия будет удаляться и генерироваться новая;
- Ускорение сайта и экономию ресурсов.

Кеширование в nginx



```
http {  
    ...  
    proxy_cache_path /var/cache/nginx levels=1:2  
    keys_zone=all:32m max_size=1g;  
    ...  
}
```

```
mkdir /var/cache/nginx  
chown nginx:nginx /var/cache/nginx/
```

Кеширование в nginx



```
server { ...  
    proxy_cache all;  
    # Кешировать указанные коды ответов 10 минут  
    proxy_cache_valid 200 301 302 304 10m;  
    # Кешировать ошибки 1 минуту  
    proxy_cache_valid 404 502 503 1m;  
}
```


Контроль доступа файлов Nginx



- Пользователь каким-то образом получил урл или название файла;
- Все запросы на скачивание файлов передаются скрипту, который решает, как поступить: отправить пользователю какой-либо файл, или показать страницу access denied.

Контроль доступа файлов Nginx



- Новый location с internal;
- Использование заголовка X-Accel-Redirect и X-Accel-Expires в ответах от скриптов backend-сервера.

```
location /protected/ {  
    ...  
    internal;  
    ...  
}
```

Контроль доступа файлов Nginx



```
location /protected/ {
    set $s3_bucket 'upload_file';
    set $aws_access_key '<access key>';
    set $aws_secret_key '<secret key>';
    set $url_full "$1";
    set $string_to_sign
"$request_method\n\n\n\n\nx-amz-date:${now}\n/$bucket/$url_full";

    set_hmac_sha1 $aws_signature $aws_secret $string_to_sign;
    set_encode_base64 $aws_signature $aws_signature;
    proxy_http_version 1.1;
    proxy_set_header Connection "";
    proxy_set_header authorization "AWS $aws_access:$aws_signature";
    proxy_set_header Host "https://${s3_bucket}.hb.bizmrg.com";
}
```

Домашнее задание № 7



1. Реализовать метод API для загрузки файла (3 балла)
2. Использовать для хранения файла облачное S3 хранилище (3 балла)
3. Создать location в Nginx для раздачи загруженных файлов (3 балла)
4. Реализовать обработчик в приложении для проверки прав доступа к файлу (1 балла)

Рекомендуемая литература



[Mail.Ru Cloud Solutions](#)
[API django-storages: Amazon S3](#)

[Nginx: Официальный сайт](#)

[Как авторизоваться MRCS:](#)

[Создание подписей для запросов REST и их аутентификация](#)

Для саморазвития (опционально)

[Чтобы не набирать двумя пальчиками](#)

Спасибо за
внимание!

Антон Кухтичев



a.kukhtichev@mail.ru



[@toshunster](https://www.instagram.com/toshunster)