$\textbf{django}_{o}cr_{s}erver$

Release 2.0

Jan 27, 2021

Contents:

CHAPTER 1

Introduction

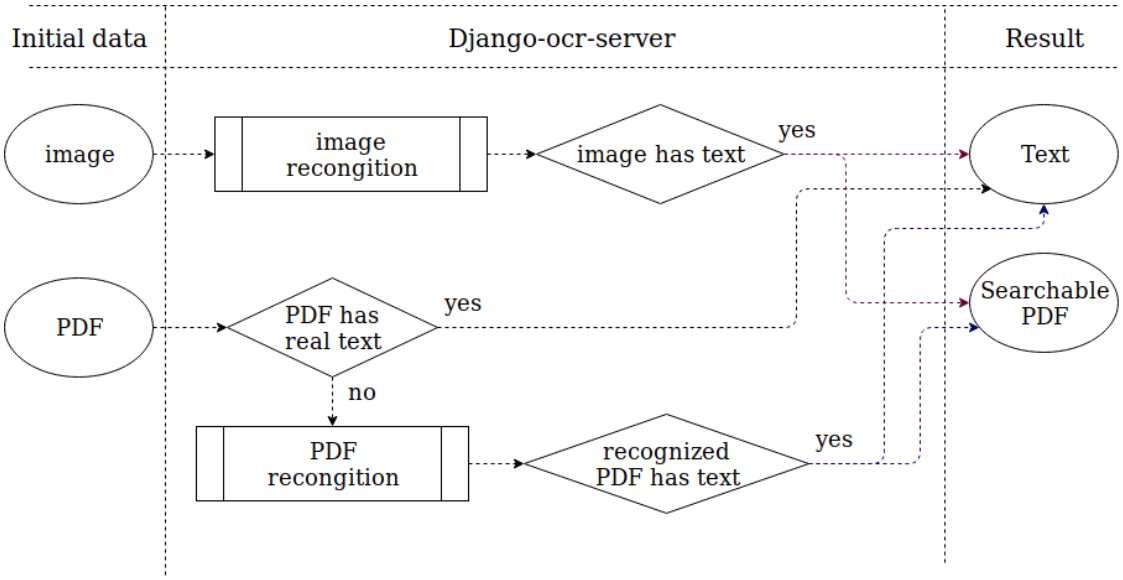
Django-ocr-server lets you recognize images and PDF. It is using tesseract for this. https://github.com/tesseract-ocr/tesseract

Django-ocr-server saves the result in the database. To prevent repeated recognition of the same file, it also saves the hash sum of the uploaded file. Therefore, when reloading an already existing file, the result returns immediately, bypassing the recognition process, which significantly reduces the load on the server.

If as a result of recognition a non-empty text is received, a searchable PDF is created.

For the searchable PDF is calculated hash sum too. Therefore, if you upload the created by Django-ocr-server searchable pdf to the server back, then this file will not be recognized, but the result will be immediately returned.

The server can process not only images, but PDF. At the same time, he analyzes, if the PDF already contains real text, this text will be used and the file will not be recognized, which reduces the load on the server and improves the quality of the output.



Storage of downloaded files and created searchable PDFs can be disabled in the settings.

For uploaded files and created searchable PDFs, and the processing results whole in the settings you can specify the lifetime after which the data will be automatically deleted.

To interact with Django-ocr-server you can use API or the admin interface.

CHAPTER 2

Installation

2.1 Linux Mint 19 (Ubuntu bionic)

Installing packages

```
$ sudo apt install g++  # need to build pdftotext
$ sudo apt install libpoppler-cpp-dev  # need to buid pdftotext
```

Installing tesseract

```
$ sudo apt install tesseract-ocr
$ sudo apt install tesseract-ocr-rus  # install languages you want
```

Installing ghostscript

```
$ sudo apt install ghostscript
```

Installing python3.7

```
$ sudo apt install python3.7
$ sudo apt install python3.7-dev
```

Installing pip

```
$ sudo apt install python-pip
```

Installing virtualenv

```
$ pip install --user virtualenv
$ echo 'PATH=~/.local/bin:$PATH' >> ~/.bashrc
$ source ~/.bashrc
```

Installing virtualenvwrapper

```
$ pip install --user setuptools
$ pip install --user wheel
$ pip install --user virtualenvwrapper
$ echo 'source ~/.local/bin/virtualenvwrapper.sh' >> ~/.bashrc
$ source ~/.bashrc
```

Creating virtualenv for django_ocr_server

```
$ mkvirtualenv django_ocr_server -p /usr/bin/python3.7
```

Installing django-ocr-server (on virtualenv django_ocr_server). It installs Django as a dependency.

```
$ pip install django-ocr-server
```

Create your Django project (on virtualenv django_ocr_server)

```
$ django-admin startproject ocr_server
```

Go to project directory

```
$ cd ocr_server
```

Edit ocr_server/settings.py

Add applications to INSTALLED_APPS

```
INSTALLED_APPS = [
    ...
    'rest_framework',
    'rest_framework.authtoken',
    'django_ocr_server',
    'rest_framework_swagger',
]
```

Edit ocr_server/urls.py

```
from django.contrib import admin
from django.urls import path, include
from rest_framework.documentation import include_docs_urls

admin.site.site_header = 'OCR Server Administration'
admin.site.site_title = 'Welcome to OCR Server Administration Portal'

urlpatterns = [
    path('admin/', admin.site.urls, ),
    path('docs/', include_docs_urls(title='OCR Server API')),
    path('', include('django_ocr_server.urls'), ),
]
```

Perform migrations (on virtualenv django_ocr_server)

```
$ python manage.py migrate
```

Create superuser (on virtualenv django_ocr_server)

```
$ python manage.py createsuperuser
```

Run server (on virtualenv django_ocr_server), than visit http://localhost:8000/

```
$ python manage.py runserver
```

2.2 Linux Mint 19 (Ubuntu bionic) automatic installation

Clone django_ocr_server from github

```
$ git clone https://github.com/shmakovpn/django_ocr_server.git
```

Run the installation script using sudo

```
$sudo {your_path}/django_ocr_server/install_ubuntu.sh
```

The script creates OS user named 'django_ocr_server', installs all needed packages. Creates the virtual environment. It installs django_ocr_server (from PyPI by default, but you can create the package from cloned repository, see the topic 'Creation a distribution package' how to do this). Then it creates the django project named 'ocr_server' in the home directory of 'django_ocr_server' OS user. After the script changes settings.py and urls.py is placed in ~django_ocr_server/ocr_server/ocr_server/. Finally it applies migrations and creates the superuser named 'admin' with the same password 'admin'.

Run server under OS user django_ocr_server, then change 'admin' password in the http://localhost:your_port/admin/ page.

```
$ sudo su
# su django_ocr_server
$ cd ~/ocr_server
$ workon django_ocr_server
$ python manage.py runserver
```

2.3 Centos 7

Install epel repository

```
$ sudo yum install epel-release
```

Install yum-utils

```
$ sudo yum install yum-utils
```

Install ghostscript (Interpreter for PostScript language & PDF needed for ocrmypdf)

```
$ sudo yum install ghostscript
```

Install wget (A utility for retrieving files using the HTTP or FTP protocols for download qpdf that needed for ocrmypdf)

```
$ sudo yum install wget
```

Install qpdf

```
$ cd /usr/local/src
$ wget https://github.com/qpdf/qpdf/releases/download/release-qpdf-9.1.0/
↪qpdf-9.1.0.tar.gz
```

(continues on next page)

(continued from previous page)

```
$ # TODO tar -zxvf qpdf-9.1.0.tar.gz
$ # TODO cd qpdf-9.1.0
$ # TODO ./Configure
$ # TODO make
$ # TODO make install
```

Install python 3.6

```
$ sudo yum install python36
$ sudo yum install python36-devel
```

Install gcc

```
$ sudo yum intall gcc
$ sudo yum install gcc-c++
```

Install poppler-cpp-devel (Development files for C++ wrapper for building pdftotext)

```
$ sudo yum install poppler-cpp-devel
```

Install tesseract

```
$ sudo yum-config-manager --add-repo https://download.opensuse.org/
↪repositories/home:/Alexander_Pozdnyakov/CentOS_7/
$ sudo bash -c "echo 'gpgcheck=0' >> /etc/yum.repos.d/download.opensuse.org_
↪repositories_home_Alexander_Pozdnyakov_CentOS_7*.repo"
$ sudo yum update
$ sudo yum install tesseract
$ sudo yum install tesseract-langpack-rus  # install a language pack you need
```

Install pip

```
$ sudo yum install python-pip
```

Install virtualenv

```
$ sudo pip install virtualenv
```

Create the virtual env for django_ocr_server

```
$ sudo virtualenv /var/www/ocr_server/venv -p /usr/bin/python3.6 --distribute
```

Give rights to the project folder to your user

```
$ sudo chown -R {your_user} /var/www/ocr_server/
```

Activate virtualenv

```
$ source /var/www/ocr_server/venv/bin/activate
```

Install postgresql 11 (The Postgresql version 9.2 that is installing in Centos 7 by default returns an error when applying migrations)

```
$ sudo rpm -Uvh https://yum.postgresql.org/11/redhat/rhel-7-x86_64/pgdg-
↪redhat-repo-latest.noarch.rpm
$ sudo yum install postgresql11-server
```

(continues on next page)

(continued from previous page)

```
$ sudo yum install postgresql-devel
$ sudo /usr/pgsql-11/bin/postgresql-11-setup initdb
```

Edit /var/lib/pgsql/11/data/pg_hba.conf

```
host    all             all             127.0.0.1/32            md5
host    all             all             ::1/128                 md5
```

```
$ sudo systemctl enable postgresql-11
$ sudo systemctl start postgresql-11
$ sudo -u postgres psql
```

Create the database and it's user

```
create database django_ocr_server encoding utf8;
create user django_ocr_server with password 'django_ocr_server';
alter database django_ocr_server owner to django_ocr_server;
alter user django_ocr_server createdb;  -- if you want to run tests
\q
```

Install python postgres database driver

```
$ pip install psycopg2-binary  # (on virtualenv django_ocr_server)
```

Installing django-ocr-server (on virtualenv django_ocr_server). It installs Django as a dependency

```
$ pip install django-ocr-server
```

Create django project (on virtualenv django_ocr_server)

```
$ cd /var/www/ocr_server
$ django-admin startproject ocr_server .
```

Edit ocr_server/settings.py

Add applications to INSTALLED_APPS

```
INSTALLED_APPS = [
    ...
    'rest_framework',
    'rest_framework.authtoken',
    'django_ocr_server',
    'rest_framework_swagger',
]
```

Configure database connection

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'django_ocr_server',
        'USER': 'django_ocr_server',
        'PASSWORD': 'django_ocr_server',
        'HOST': 'localhost',
        'PORT': '',
    }
}
```

Edit ocr_server/urls.py

```
from django.contrib import admin
from django.urls import path, include
from rest_framework.documentation import include_docs_urls

admin.site.site_header = 'OCR Server Administration'
admin.site.site_title = 'Welcome to OCR Server Administration Portal'

urlpatterns = [
    path('admin/', admin.site.urls, ),
    path('docs/', include_docs_urls(title='OCR Server API')),
    path('', include('django_ocr_server.urls'), ),
]
```

Apply migrations (on virtualenv django_ocr_server)

```
$ python manage.py migrate
```

Create superuser (on virtualenv django_ocr_server)

```
$ python manage.py createsuperuser
```

Run server (on virtualenv django_ocr_server), than visit http://localhost:8000/

```
$ python manage.py runserver
```

CHAPTER 3

Configuration

For changing your django_ocr_server behavior you can use several parameters in the settings.py of your django project.

`django_ocr_server.default_settings.`**`OCR_STORE_FILES = True`**
Store uploaded files (True) or not (False), default to True

`django_ocr_server.default_settings.`**`OCR_FILE_PREVIEW = True`**
Show file preview in admin (True) or not (False), default to True

`django_ocr_server.default_settings.`**`OCR_TESSERACT_LANG = 'rus+eng'`**
Sets priority of using languages, default to 'rus+eng'

`django_ocr_server.default_settings.`**`OCR_STORE_PDF = True`**
Generate and store recognized searchable PDF (True) or not (False), default to True

`django_ocr_server.default_settings.`**`OCR_STORE_FILES_DISABLED_LABEL = 'store_files_disab`**
The text of storeing uploaded files disabled label in the admin interface

`django_ocr_server.default_settings.`**`OCR_STORE_PDF_DISABLED_LABEL = 'store_pdf_disabled'`**
The text of storeing recognized PDF disabled label in the admin interface

`django_ocr_server.default_settings.`**`OCR_FILE_REMOVED_LABEL = 'file_removed'`**
The text of the label of *file removed* in the admin interface

`django_ocr_server.default_settings.`**`OCR_PDF_REMOVED_LABEL = 'pdf_removed'`**
The text of the label of *PDF removed* in the admin interface

`django_ocr_server.default_settings.`**`OCR_ALLOWED_FILE_TYPES = ['application/pdf', 'image`**
The types of file allowed to uploading

`django_ocr_server.default_settings.`**`OCR_FILES_UPLOAD_TO = '/home/docs/checkouts/readthe`**
The directory for saving uploaded files

`django_ocr_server.default_settings.`**`OCR_PDF_UPLOAD_TO = '/home/docs/checkouts/readthedo`**
The directory for storeing searchable PDFs

3.1 Time to live settings

`django_ocr_server.default_settings.`**`OCR_FILES_TTL = datetime.timedelta(0)`**
When current datetime will be grater then the datetime of file uploading plus this timedelta, the uploaded file will be removed. *timedelta(0)* means that **OCR_FILES_TTL** is disabled. Defaults to *timedelta(0)*.

`django_ocr_server.default_settings.`**`OCR_PDF_TTL = datetime.timedelta(0)`**
When current datetime will be grater then the datetime of creating recognized PDF plus this timedelta, the recognized PDF will be removed. *timedelta(0)* means that **OCR_PDF_TTL** is disabled. Defaults to *timedelta(0)*.

`django_ocr_server.default_settings.`**`OCR_TTL = datetime.timedelta(0)`**
When current datetime will be grater then the datetime of creating the model (OCRedFile) in the database plus this timedelta, the model in the database will be removed. *timedelta(0)* means that **OCR_TTL** is disabled. Defaults to *timedelta(0)*.

CHAPTER 4

Deploying to production

4.1 Linux Mint 19 (Ubuntu bionic)

Installing nginx

```
$ sudo apt install nginx
```

Installing uwsgi (on virtualenv django_ocr_server)

```
$ pip install uwsgi
```

Create {path_to_your_project}/uwsgi.ini

```
[uwsgi]
  chdir = {path_to_your_project}  # e.g. /home/shmakovpn/ocr_server
  module = {your_project}.wsgi  # e.g. ocr_server.wsgi
  home = {path_to_your_virtualenv}  # e.g. /home/shmakovpn/.virtualenvs/
↪django_ocr_server
  master = true
  processes = 10
  http = 127.0.0.1:8003
  vacuum = true
```

Create /etc/nginx/sites-available/django_ocr_server.conf

```
server {
    listen 80;  # choose port what you want
    server_name _;
    charset utf-8;
    client_max_body_size 75M;
    location /static/rest_framework_swagger {
        alias {path_to_your virtualenv}/lib/python3.6/site-packages/rest_
↪framework_swagger/static/rest_framework_swagger;
    }
```

(continues on next page)

(continued from previous page)

```
    location /static/rest_framework {
        alias {path_to_your virtualenv}/lib/python3.7/site-packages/rest_
↪framework/static/rest_framework;
    }
    location /static/admin {
        alias {path_to_your virtualenv}/lib/python3.7/site-packages/django/
↪contrib/admin/static/admin;
    }
    location / {
        proxy_pass http://127.0.0.1:8003;
    }
}
```

Enable the django_ocr_server site

```
$ sudo ln -s /etc/nginx/sites-available/django_ocr_server.conf /etc/nginx/
↪sites-enabled/
```

Remove the nginx default site

```
$ sudo rm /etc/nginx/sites-enabled/default
```

Create the systemd service unit /etc/systemd/system/django-ocr-server.service

```
[Unit]
  Description=uWSGI Django OCR Server
  After=syslog.service

[Service]
  User={your user}
  Group={your group}
  Environment="PATH={path_to_your_virtualenv}/bin:/usr/local/sbin:/usr/local/
↪bin:/usr/sbin:/usr/bin:/sbin:/bin"
  ExecStart={path_to_your_virtualenv}/bin/uwsgi --ini {path_to_your_project}/
↪uwsgi.ini
  RuntimeDirectory=uwsgi
  Restart=always
  KillSignal=SIGQUIT
  Type=notify
  StandardError=syslog
  NotifyAccess=all

[Install]
  WantedBy=multi-user.target
```

Reload systemd

```
$ sudo systemctl daemon-reload
```

Start the django-ocr-server service

```
$ sudo systemctl start django-ocr-server
```

Enable the django-ocr-server service to start automatically after server is booted

```
$ sudo systemclt enable django-ocr-server
```

Start nginx

```
$ sudo systemctl start nginx
```

Enable nginx service to start automatically after server is booted

```
$ sudo systemctl enable nginx
```

Go to http://{your_server}:80 You will be redirected to admin page

4.2 Centos 7

Installing nginx

```
$ sudo apt install nginx
```

Installing uwsgi (on virtualenv django_ocr_server)

```
$ pip install uwsgi
```

Create /var/www/ocr_server/uwsgi.ini

```
[uwsgi]
  chdir = /var/www/ocr_server
  module = ocr_server.wsgi
  home = /var/www/ocr_server/venv
  master = true
  processes = 10
  http = 127.0.0.1:8003
  vacuum = true
```

Create the systemd service unit /etc/systemd/system/django-ocr-server.service

```
[Unit]
  Description=uWSGI Django OCR Server
  After=syslog.service

[Service]
  User=nginx
  Group=nginx
  Environment="PATH=/var/www/ocr_server/venv/bin:/sbin:/bin:/usr/sbin:/usr/
↪bin"
  ExecStart=/var/www/ocr_server/venv/bin/uwsgi --ini /var/www/ocr_server/
↪uwsgi.ini
  RuntimeDirectory=uwsgi
  Restart=always
  KillSignal=SIGQUIT
  Type=notify
  StandardError=syslog
  NotifyAccess=all

[Install]
  WantedBy=multi-user.target
```

Reload systemd service

```
$ sudo systemctl daemon-reload
```

Chango user of /var/www/ocr_server to nginx

```
$ sudo chown -R nginx:nginx /var/www/ocr_server
```

Start Django-ocr-server service

```
$ sudo systemctl start django-ocr-service
```

Check that port is up

```
$ sudo netstat -anlpt \| grep 8003
```

you have to got something like this:

```
tcp        0      0 127.0.0.1:8003          0.0.0.0:*               LISTEN  ␣
↪   2825/uwsgi
```

Enable Django-ocr-server uwsgi service

```
$ sudo systemctl enable django-ocr-service
```

Edit /etc/nginx/nginx.conf

```
server {
    listen       80 default_server;
    listen       [::]:80 default_server;
    server_name  _;
    charset utf-8;
    client_max_body_size 75M;
    location /static/rest_framework_swagger {
        alias /var/www/ocr_server/venv/lib/python3.6/site-packages/rest_
↪framework_swagger/static/rest_framework_swagger;
    }
    location /static/rest_framework {
        alias /var/www/ocr_server/venv/lib/python3.6/site-packages/rest_
↪framework/static/rest_framework;
    }
    location /static/admin {
        alias /var/www/ocr_server/venv/lib/python3.6/site-packages/django/
↪contrib/admin/static/admin;
    }
    location / {
        proxy_pass http://127.0.0.1:8003;
    }
}
```

Configure SELinux

Django has a bug (https://code.djangoproject.com/ticket/29027#no1)

By default it stores uploading files size more than 2,5Mb to /tmp folder. A temp file gets 'system_u:object_r:httpd_tmp_t:s0' SELinux context. Then Django tries to copy this file to the uploading folder with its SELinux context using os.setxattr() from lib/python3.6/shutil.py. But it is a wrong behavior because in the uploading folder the SELinux context of a file have to be 'http_sys_rw_content_t'. To solve the problem we have to create another folder for temp files with 'http_sys_rw_content_t' for example /var/www/ocr_server/tmp. Then configure Django to store temp files to this folder.

```
$ sudo mkdir /var/www/ocr_server/tmp
$ sudo chown {your_user} /var/www/ocr_server/tmp
```

Change /var/www/ocr_server/ocr_server/settings.py

```
FILE_UPLOAD_TEMP_DIR = os.path.join(BASE_DIR, 'tmp')
```

Configure SELinux contexts

```
$ sudo semanage port -a -t http_port_t -p tcp 8003
$ sudo semanage fcontext -a -t httpd_sys_content_t '/var/www/ocr_server/
↪venv/lib/python3.6/site-packages/rest_framework_swagger/static/rest_
↪framework_swagger(/.*)?'
$ sudo semanage fcontext -a -t httpd_sys_content_t '/var/www/ocr_server/
↪venv/lib/python3.6/site-packages/rest_framework/static/rest_framework(/
↪.*)?'
$ sudo semanage fcontext -a -t httpd_sys_content_t '/var/www/ocr_server/
↪venv/lib/python3.6/site-packages/django/contrib/admin/static/admin(/.
↪*)?'
$ find /var/www/ocr_server/venv/lib/python3.6/site-packages/ | grep '\.so
↪' | grep -v '\.libs' | xargs -L1 sudo semanage fcontext -a -t httpd_
↪sys_script_exec_t
$ sudo semanage fcontext -a -t httpd_sys_script_exec_t '/var/www/ocr_
↪server/venv/bin(/.*)?'
$ sudo semanage fcontext -a -t httpd_sys_script_exec_t '/var/www/ocr_
↪server/venv/lib/python3.6/site-packages/psycopg2/.libs(/.*)?'
$ sudo semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/ocr_
↪server/django_ocr_server/upload(/.*)?'
$ sudo semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/ocr_
↪server/django_ocr_server/pdf(/.*)?'
$ sudo semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/ocr_
↪server/tmp(./*)?'
$ sudo restorecon -Rv /var/www/ocr_server
$ sudo setsebool -P httpd_can_network_connect_db 1
```

Start nginx service

```
$ sudo systemctl start nginx
```

Enable nginx service

```
$ sudo systemctl enable nginx
```

Configure firewall

```
$ sudo firewall-cmd --zone=public --add-service=http --permanent
$ sudo firewall-cmd --reload
```

Go to http://{your_server}:80
You will be redirected to admin page

CHAPTER 5

Usage examples

You can download all examples from https://github.com/shmakovpn/django_ocr_server/tree/master/usage_examples

5.1 curl

Use curl with '@' before the path of the uploading file

```
#!/usr/bin/env bash
curl -F "file=@example.png" localhost:8000/upload/
```

5.2 python

Use requests.post function

```
import requests


with open("example.png", 'rb') as fp:
    print(requests.post("http://localhost:8000/upload/",
                        files={'file': fp}, ).content)
```

5.3 perl

Use LWP::UserAgent and HTTP::Request::Common

```perl
#!/usr/bin/perl
use strict;
use warnings FATAL => 'all';
use LWP::UserAgent;
use HTTP::Request::Common;

my $ua = LWP::UserAgent->new;
my $url = "http://localhost:8000/upload/";
my $fname = "example.png";

my $req = POST($url,
    Content_Type => 'form-data',
    Content => [
        file => [ $fname ]
    ]);

my $response = $ua->request($req);

if ($response->is_success()) {
    print "OK: ", $response->content;
} else {
    print "Failed: ", $response->as_string;
}
```

5.4 php

Use CURLFile($file, $mime, $name)

```php
<?php
//Initialise the cURL var
$ch = curl_init();

//Get the response from cURL
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

//Set the Url
curl_setopt($ch, CURLOPT_URL, 'http://localhost:8000/upload/');

//Create a POST array with the file in it
$file='example.png';
$mime=getimagesize($file)['mime'];
$name=pathinfo($file)['basename'];
$postData = array(
    'file' => new CURLFile($file, $mime, $name),
);

curl_setopt($ch, CURLOPT_POSTFIELDS, $postData);

// Execute the request
$response = curl_exec(  $ch);
echo($response);

curl_close ($ch);

?>
```

CHAPTER 6

Running tests

Perform under you django_ocr_server virtual environment $python manage.py test django_ocr_server.tests

CHAPTER 7

API documentation

Django-ocr-server provides API documentation use restframework.documentation and swagger.
Visit http://localhost:8000/swagger and http://localhost:8000/docs/

CHAPTER 8

Management Commands

Run it to clean trash. It removes all uploaded files and PDFs that do not have related models in database.

```
$ python manage.py clean
```

Run it to remove models, uploaded files and PDFs, whose time to live (TTL) has expired.

```
$ python manage.py ttl
```

Create the user for API (return the AUTH-token).

```
$ python manage.py create_user username password
b2db7002e037a4edb25aed33b04b97e468970376
```

CHAPTER 9

Creation a distribution package

As mentioned earlier, the automatic installation script ‘install_ubuntu.sh’ uses the package from the PyPI repository by default. To change this behavior or if you need your own distribution package you can build it.

Run command

```
$cd path to cloned project from github
$python setup.py sdist
```

Look in ‘dist’ directory, there is your package was created.

Also you can continue automatic installation. The package will be used.

CHAPTER 10

Developer Guide

10.1 django_ocr_server/default_settings.py

The default settings of **django_ocr_server**.

```
"""
django_ocr_server/default_settings.py
++++++++++++++++++++++++++++++++++++

This file contains default settings for OCR Server

| Author: shmakovpn <shmakovpn@yandex.ru>
| Date: 2019-02-21/2019-03-29/2019-04-12/2021-01-19
"""
from typing import List
import os
from datetime import timedelta
from django.conf import settings

# configure Django's settings if not configured (need for Sphinx␣
↪autodoc)
if not settings.configured:
    settings.configure(
        BASE_DIR=os.path.dirname(os.path.dirname(os.path.abspath(__
↪file__))))

OCR_STORE_FILES: bool = True  #: Store uploaded files (True) or not␣
↪(False), default to True
OCR_FILE_PREVIEW: bool = True  #: Show file preview in admin (True)␣
↪or not (False), default to True
OCR_TESSERACT_LANG: str = 'rus+eng'  #: Sets priority of using␣
↪languages, default to 'rus+eng'
OCR_STORE_PDF: bool = True  #: Generate and store recognized␣
↪searchable PDF (True) or not (False), default to True
```

(continues on next page)

(continued from previous page)

```
OCR_STORE_FILES_DISABLED_LABEL: str = 'store_files_disabled'
"""The text of storeing uploaded files disabled label in the admin␣
↪interface"""

OCR_STORE_PDF_DISABLED_LABEL: str = 'store_pdf_disabled'
"""The text of storeing recognized PDF disabled label in the admin␣
↪interface"""

OCR_FILE_REMOVED_LABEL: str = 'file_removed'
"""The text of the label of *file removed* in the admin interface"""

OCR_PDF_REMOVED_LABEL: str = 'pdf_removed'
"""The text of the label of *PDF removed* in the admin interface"""

OCR_ALLOWED_FILE_TYPES: List[str] = [
    'application/pdf',
    'image/jpeg',
    'image/png',
    'image/bmp',
    'image/tiff',
]
"""The types of file allowed to uploading"""
"""
2019-10-22 shmakovpn. An error was found when trying to deploy␣
↪Django-OCR_Server using Apache
because usage of relative paths is a wrong way when Apache mod_wsgi␣
↪is using
https://modwsgi.readthedocs.io/en/develop/user-guides/application-
↪issues.html#application-working-directory
"""

OCR_FILES_UPLOAD_TO: str = os.path.join(settings.BASE_DIR, __package_
↪_,
                                        'upload')
"""The directory for saving uploaded files"""

OCR_PDF_UPLOAD_TO: str = os.path.join(settings.BASE_DIR, __package__,
↪ 'pdf')
"""The directory for storeing searchable PDFs"""

OCR_FILES_TTL: timedelta = timedelta(0)
"""
When current datetime will be grater then the datetime of file␣
↪uploading plus this timedelta,
the uploaded file will be removed.
*timedelta(0)* means that **OCR_FILES_TTL** is disabled.
Defaults to *timedelta(0)*.
"""

OCR_PDF_TTL: timedelta = timedelta(0)
"""
When current datetime will be grater then the datetime of creating␣
↪recognized PDF plus this timedelta,
the recognized PDF will be removed.
*timedelta(0)* means that **OCR_PDF_TTL** is disabled.
Defaults to *timedelta(0)*.
```

(continues on next page)

(continued from previous page)

```
"""

OCR_TTL: timedelta = timedelta(0)
"""
When current datetime will be grater then the datetime of creating␣
↪the model (OCRedFile) in the database plus this timedelta,
the model in the database will be removed.
*timedelta(0)* means that **OCR_TTL** is disabled.
Defaults to *timedelta(0)*.
"""
```

10.2 django_ocr_server/conf.py

The settings manager of **django_ocr_server**.

10.2.1 django_ocr_server/conf.py

The settings manager for **django_ocr_server**.

Usage:

```
from django_ocr_server.conf import ocr_settings
# Next line will print a value of **OCR_TESSERACT_LANG**
# using the variable from the Django's *settings.py* file
# if the variable is set there.
# Or the default value of **OCR_TESSERACT_LANG** from
# *django_ocr_server/default_settings.py* otherwise.
print(ocr_settings.OCR_TESSERACT_LANG)
```

Author: shmakovpn <shmakovpn@yandex.ru>
Date: 2021-01-20

class django_ocr_server.conf.**DjangoOcrSettings**
The settings manager of **django_ocr_server**

django_ocr_server.conf.**ocr_settings = <django_ocr_server.conf.DjangoOcrSettings obj**
The instance of settings manager of **django_ocr_server**

CHAPTER 11

Indices and tables

- genindex
- modindex

Python Module Index

d

Index

P

R

S

T

U