[OpenCV 활용 Django 기반 Webapp 만들기]

0. 지난 실습에서 만들었던 Python 가상환경을 삭제하고 다시 시작합니다.

C:₩**work_django₩django_mldl** 폴더 안의 [**django_env**] 폴더를 삭제해주세요.

(연습을 위한 것으로, 삭제하지 않고 그대로 유지하여도 실습에는 무방합니다.)

1. 장고 프로젝트 폴더 접근 & Python 가상환경 생성 및 activate

(cmd 관리자권한으로 실행)

cd .. 명령어로 c drive까지 나가기

1-1) c₩work_django₩**django_mldl 폴더가 <u>없는</u> 경우** (지난 수업에 참석 X 시)

mkdir work_django -> c₩work_django 폴더가 생깁니다. (이미 폴더가 있을 경우 Skip)

cd work_django

mkdir django_mldl -> c₩work_django₩django_mldl 폴더가 생깁니다. (이미 폴더가 있을 경우 Skip)

cd django_mldl

1-2) c₩work_django₩**django_mldl 폴더가 <u>있는</u> 경우** (지난 수업에 참석 O 시)

cd work_django

cd django_mldl

2) **가상환경 생성 및 Django 라이브러리 설치**

pip install virtualenv==16.7.7 -> 가상 환경을 만들어주는 라이브러리입니다.

virtualenv django_env -> django_env 라는 이름으로 가상 환경을 새로이 만듭니다.

django_env₩Scripts₩activate -> Scripts 폴더 내의 activate 파일을 실행해 가상 환경을 활성화합니다.

2. Django & OpenCV 설치하기 (활성화된 env 내에 설치)

pip install django==2.2.6

pip install opencv-python==4.1.1.26

pip install pillow==5.4.1

pip freeze -> 가상 환경에 설치된 라이브러리들의 리스트를 검토할 수 있습니다.

3. 장고 프로젝트 생성 (C:₩work_django₩django_mldl)

: project는 하나의 **웹사이트**에 해당합니다.

django-admin startproject cv_project -> **cv_project** 폴더가 생깁니다.

cd cv_project

Atom에서 프로젝트 폴더 열기

: **File -> Add Project Folder...** 에서 django_mldl > **cv_project** 폴더 선택하기 (django_mldl 폴더가 아닙니다.)

C:₩work_django₩django_mldl₩**cv_project₩cv_project** -> **settings.py 파일 수정 & 추가**

LANGUAGE_CODE = 'en-us' -> **'ko-kr'**

TIME_ZONE = 'UTC' -> **'Asia/Seoul'**

STATIC_ROOT = os.path.join(BASE_DIR, 'static')

3. 장고 app 생성 (C:₩work_django₩django_mldl₩cv_project)

: app 은 하나의 **웹사이트 내에 있는 기능들**에 해당합니다. (**회원가입 기능, 상품 관련 기능** 등)

python manage.py startapp opencv_webapp -> opencv_webapp 폴더가 생깁니다.

C:₩work_django₩django_mldl₩**cv_project₩cv_project** -> **settings.py 파일 수정**

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

장고 프로젝트 Server 실행 & 웹브라우저에서 확인

python manage.py runserver (포트번호 변경을 원할 시 : python manage.py runserver 8888)

브라우저에서 **127.0.0.1:8000** 접속

4. index page 역할을 할 첫번째 url/view/template set 만들기

* **cv_project₩cv_project₩urls.py** 파일의 상단 주석에 적힌 **URL Configuration** 방식 중 [**Including another URLconf**] 방식을 활용하는 방법입니다.

* 기존에 cv_project 폴더 내의 urls.py가 모두 처리하던 **전체 URL들을 App 단위로 나눠 app 폴더 내의 urls.py에게 권한을 위임**하는 과정입니다.

- 프로젝트 urls.py에 app의 urls.py를 미리 **include**
- app(opencv_webapp)의 urls.py를 생성
- app의 urls.py에 views.py에서 만들 함수를 미리 URL과 함께 등록하기
- app의 views.py에 실제 함수(first_view) 만들기
- views.py에 만든 함수(first_view)의 실행을 통해 사용자에게 보여줄 Template 파일(first_view.html) 만들기

4-1) C:₩work_django₩django_mldl**₩cv_project₩cv_project**

-> urls.py 파일 수정하여 opencv_webapp₩urls.py include하기

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    # 아래의 URL Pattern을 opencv_webapp/urls.py가 handling
    path('', include('opencv_webapp.urls')),
]
```

4-2) C:₩work_django₩django_mldl₩cv_project₩opencv_webapp -> **urls.py 파일 생성** 후 아래 내용 작성

```
from django.urls import path
from . import views # 같은 폴더 내의 views.py를 import

app_name = 'opencv_webapp'

urlpatterns = [
    path('', views.first_view, name='first_view'),
]
```

4-3) C:₩work_django₩django_mldl₩cv_project₩opencv_webapp -> **views.py 파일 수정**

```
from django.shortcuts import render

def first_view(request):
    return render(request, 'opencv_webapp/first_view.html', {})
```

4-4) ₩cv_project₩opencv_webapp₩templates₩opencv_webapp₩ -> 경로 유의하여 **first_view.html 파일 생성**

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Opencv-Django</title>
  </head>
  <body>
```

```
    Hello world!
  </body>
</html>
```

4-5) 장고 프로젝트 Server 실행 & 웹브라우저에서 확인

python manage.py runserver

브라우저에서 접속 @ **127.0.0.1:8000**

5. 이미지 파일 업로드를 처리하기 위한 페이지를 위한 url/view/template + form set 만들기 (DB 저장 X, 이미지 파일 업로드를 이해하기 위한 실습)

5-1) **업로드된 데이터(이미지 파일 자체)**를 담기 위한 **media 디렉토리 설정**하기

C:₩work_django₩django_mldl**₩cv_project₩cv_project** -> **settings.py** 파일 최하단에 아래 2줄 추가

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

5-2) C:₩work_django₩django_mldl**₩cv_project₩opencv_webapp** -> **urls.py 파일 수정**

```
from django.urls import path
from . import views # 같은 폴더 내의 views.py를 import
```

from django.conf import settings

from django.conf.urls.static import static

app_name = 'opencv_webapp'

urlpatterns = [

path('', views.first_view, name='first_view'),

path('simple_upload/', views.simple_upload, name='simple_upload'),

]

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

5-3) C:₩work_django₩django_mldl**₩cv_project₩opencv_webapp** -> **forms.py 파일 생성** 후 아래 입력

* 아래 코드에서 활용되는 forms.**ImageField**는 forms.**FileField**의 모든 속성과 기능들을 동일하게 갖고 있으며, 이에 더해 업로드된 객체가 이미지 파일이 맞는지를 검증해줍니다. 또한 이미지 파일의 저장 경로뿐만 아니라 가로 및 세로 사이즈를 함께 자동으로 저장하게끔 할 수 있으며, 이는 추후 이미지 크기를 기준으로 한 검색 혹은 정렬 기능 구현 시 유용하게 활용될 수 있습니다.

from django import forms

class SimpleUploadForm(forms.Form):

title = forms.CharField(max_length=50)

ImageField Inherits all attributes and methods from FileField, but also validates that the uploaded object is a valid image.

file = forms.FileField()

image = forms.ImageField()

5-4) C:₩work_django₩django_mldl**₩cv_project₩opencv_webapp** -> **views.py 파일 수정**

```
from django.shortcuts import render, redirect
from .forms import SimpleUploadForm
from django.core.files.storage import FileSystemStorage

...
def simple_upload(request):
    if request.method == 'POST':
        # print(request.POST) : <QueryDict: {'csrfmiddlewaretoken': ['~~~'], 'title': ['upload_1']}>
        # print(request.FILES) : <MultiValueDict: {'image': [<InMemoryUploadedFile: ses.jpg (image/jpeg)>]}>
        # 비어있는 Form에 사용자가 업로드한 데이터를 넣고 검증합니다.
        form = SimpleUploadForm(request.POST, request.FILES)

        if form.is_valid():
            myfile = request.FILES['image'] # 'ses.jpg'
            fs = FileSystemStorage()
            filename = fs.save(myfile.name, myfile) # 경로명을 포함한 파일명 & 파일 객체
            # 업로드된 이미지 파일의 URL을 얻어내 Template에게 전달
            uploaded_file_url = fs.url(filename) # '/media/ses.jpg'

            context = {'form': form, 'uploaded_file_url': uploaded_file_url} # filled form
            return render(request, 'opencv_webapp/simple_upload.html', context)

    else: # request.method == 'GET' (DjangoBasic 실습과 유사한 방식입니다.)
        form = SimpleUploadForm()
        context = {'form': form} # empty form
        return render(request, 'opencv_webapp/simple_upload.html', context)
```

5-5) ₩cv_project₩opencv_webapp₩templates₩opencv_webapp₩

-> 경로 유의하여 **simple_upload.html 파일 생성** 후 아래 내용 작성

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Opencv-Django</title>
  </head>
  <body>
    <h1>Upload your image file.</h1>

    <form method="POST" enctype="multipart/form-data">
      {% csrf_token %}
      {{ form.as_p }}
      <button type="submit">Upload</button>
    </form>

    {% if uploaded_file_url %}
      <img src="{{ uploaded_file_url }}">
      <p>Uploaded file @ : <a href="{{ uploaded_file_url }}">{{ uploaded_file_url }}</a></p>
    {% endif %}

    <p><a href="{% url 'opencv_webapp:simple_upload' %}">Return to home</a></p>
  </body>
</html>
```

5-6) 장고 프로젝트 Server 실행 & 웹브라우저에서 확인

python manage.py runserver

브라우저에서 접속 @ 127.0.0.1:8000**/simple_upload**

6. OpenCV 로 Face detection 구현하기

: 1) 이미지 파일을 업로드받아 DB에 저장하기

1-1) 업로드된 이미지 파일을 저장하고 관리하기 위한 **Model class 선언**하기

* 강의안 슬라이드 **58p**에서 먼저 만들려는 DB Table의 구조를 살펴보고 돌아와주세요.

C:₩work_django₩django_mldl₩**cv_project₩opencv_webapp** -> **models.py 파일 수정**

```
class ImageUploadModel(models.Model):
    # blank=True : Form에서 빈 채로 저장되는 것을 허용 (views.py에서 활용한 .is_valid() 함수가 검증 진행 시)
    description = models.CharField(max_length=255, blank=True)
    # upload_to : 저장될 파일의 경로를 지정 (ex. 'images/2020/02/21/test_image.jpg')
    document = models.ImageField(upload_to='images/%Y/%m/%d')
    # auto_now_add : 자동으로 저장되는 시점을 기준으로 현재 시간을 세팅
    uploaded_at = models.DateTimeField(auto_now_add=True)
```

6-2) 앞서 만든 **ImageUploadModel 을 활용한 ModelForm** 만들기

* 5-3에서 만들었던 SimpleUploadForm은 사용자로부터 이미지 파일을 업로드받아 프로젝트의 media 디렉토리에 저장하는 역할까지만 수행합니다. 이제부터 만들 ImageUploadForm은 사용자로부터 이미지 파일을 업로드받아 Database에 저장을 마친 후, 저장이 완료된 이미지 파일을 대상으로 Face detection까지 적용한다는 차이점이 있습니다. SimpleUploadForm을 활용할 때에는 models.py에 별도의 Class를 작성하지 않았었다는 것을 떠올려보시면 좋습니다.

C:₩work_django₩django_mldl₩**cv_project₩opencv_webapp** -> **forms.py 파일**에 아래 내용 추가

```
from django import forms
from .models import ImageUploadModel

class ImageUploadForm(forms.ModelForm):
    # Form을 통해 받아들여야 할 데이터가 명시되어 있는 메타 데이터 (DB 테이블을 연결)
    class Meta:
        model = ImageUploadModel
        # Form을 통해 사용자로부터 입력 받으려는 Model Class의 field 리스트
        fields = ('description', 'document', ) # uploaded_at
```

6-3) ImageUploadForm을 통해 **업로드된 이미지를 처리하기 위한 detect_face 함수**를 views.py에 정의하고 urls.py 연결하기

6-3-1) C:₩work_django₩django_mldl₩**cv_project₩cv_project** -> **settings.py 파일** 최하단에 아래 1줄 추가

* **'.'**은 프로젝트 디렉토리(C:₩work_django₩django_mldl₩cv_project) 자체를 의미합니다.

```
MEDIA_ROOT_URL = '.'
```

6-3-2) C:₩work_django₩django_mldl₩**cv_project₩opencv_webapp** -> **urls.py 파일 수정**

```
...
urlpatterns = [
    ...,
    path('detect_face/', views.detect_face, name='detect_face'),
]
```

6-3-3) C:₩work_django₩django_mldl₩**cv_project₩opencv_webapp** -> **views.py 파일 수정**

```
from django.shortcuts import render, redirect
from .forms import SimpleUploadForm, ImageUploadForm
from django.core.files.storage import FileSystemStorage
from django.conf import settings

...
def detect_face(request):
    if request.method == 'POST' :
        # 비어있는 Form에 사용자가 업로드한 데이터를 넣고 검증합니다.
        form = ImageUploadForm(request.POST, request.FILES)
        if form.is_valid():
            # Form에 채워진 데이터를 DB에 실제로 저장하기 전에 변경하거나 추가로 다른 데이터를 추가할 수 있음
            # post는 save() 후 DB에 저장된 ImageUploadModel 클래스 객체 자체를 갖고 있게 됨 (record에 해당)
            post = form.save(commit=False)
            post.save() # DB에 실제로 Form 객체('form')에 채워져 있는 데이터를 저장

            imageURL = settings.MEDIA_URL + form.instance.document.name
            # print(form.instance, form.instance.document.name, form.instance.document.url)
            # cv_detect_face(settings.MEDIA_ROOT_URL + imageURL) # 추후 구현 예정

            return render(request, 'opencv_webapp/detect_face.html', {'form':form, 'post':post})

    else:
        form = ImageUploadForm() # empty form
        return render(request, 'opencv_webapp/detect_face.html', {'form':form})
```

6-4) ImageUploadForm을 통해 사용자로부터 이미지 파일을 업로드받기 위한 **detect_face.html** 만들기

: ₩cv_project₩opencv_webapp₩templates₩opencv_webapp₩ -> 경로 유의하여 **detect_face.html** 파일 만들기

(아래 내용을 구글드라이브의 문서로 올려두었습니다.)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Opencv-Django</title>
  </head>
  <body>

    <form method="POST" enctype="multipart/form-data">
      {% csrf_token %}
      {{ form.as_p }}
      <button type="submit">Upload</button>
    </form>

    {% if post %}
      <!-- post.document : ImageUploadModel Class에 선언되어 있는 "document"에 해당 -->
      <img src="{{ post.document.url }}">
      <p>Uploaded file @ : <a href="{{ post.document.url }}">{{ post.document }}</a></p>
    {% endif %}

    <p><a href="{% url 'opencv_webapp:detect_face' %}">Return to home</a></p>

  </body>
</html>
```

6-5) **DB migrate** & 장고 프로젝트 Server 실행 후 결과 확인 (cmd)

Ctrl + C

python manage.py **makemigrations**

python manage.py **migrate**

python manage.py **runserver**

브라우저에서 접속 @ 127.0.0.1:8000**/detect_face**

7. OpenCV 로 Face detection 구현하기

: 2) 업로드받은 이미지를 대상으로 Face detection 적용하기

7-1) Face detection 을 위한 **opencv 활용 Python 함수** 구현

C:₩work_django₩django_mldl₩**cv_project₩opencv_webapp** -> **cv_functions.py** 파일 생성 후 아래 내용 작성

(아래 내용을 구글드라이브의 문서로 올려두었습니다.)

```
from django.conf import settings

import numpy as np

import cv2
```

```
def cv_detect_face(path):
    img = cv2.imread(path, 1)

    if (type(img) is np.ndarray):
        print(img.shape) # 세로, 가로, 채널

        # Haar-based Cascade Classifier : AdaBoost 기반 머신러닝 물체 인식 모델
        # 이미지에서 눈, 얼굴 등의 부위를 찾는데 주로 이용
        # 이미 학습된 모델을 OpenCV 에서 제공 (http://j.mp/2qIxrxX)
        baseUrl = settings.MEDIA_ROOT_URL + settings.MEDIA_URL
        face_cascade = cv2.CascadeClassifier(baseUrl+'haarcascade_frontalface_default.xml')
        eye_cascade = cv2.CascadeClassifier(baseUrl+'haarcascade_eye.xml')

        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        # detectMultiScale(Original img, ScaleFactor, minNeighbor) : See http://j.mp/2SxjtKR
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)
        for (x, y, w, h) in faces:
            cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
            roi_gray = gray[y:y+h, x:x+w]
            roi_color = img[y:y+h, x:x+w]
            eyes = eye_cascade.detectMultiScale(roi_gray)
            for (ex, ey, ew, eh) in eyes:
                cv2.rectangle(roi_color, (ex, ey), (ex+ew, ey+eh), (0, 255, 0), 2)
        cv2.imwrite(path, img)

    else:
        print('Error occurred within cv_detect_face!')
        print(path)
```

7-2) 위 cv_functions.py에서 불러와 활용하고 있는 **얼굴/눈 검출을 위한 학습된 모델 파일** 복사해 넣기

- 1) **haarcascade_frontalface_default.xml**
- 2) **haarcascade_eye.xml**
- 위 2개의 파일을 아래 media 폴더 안에 복사하기

  -> C:₩work_django₩django_mldl**₩cv_project₩media₩**

7-3) 구현을 마친 cv_functions.py의 cv_detect_face 함수를 실제로 view.py 에서 활용할 수 있도록 주석 해제

C:₩work_django₩django_mldl**₩cv_project₩opencv_webapp** -> **views.py 파일 수정**

```
...
from .cv_functions import cv_detect_face
...
def detect_face(request):
        ...
        cv_detect_face(settings.MEDIA_ROOT_URL + imageURL) # 추후 구현 예정
        ...
```

7-4) 장고 프로젝트 Server 실행 후 결과 확인 (cmd)

python manage.py runserver

브라우저에서 접속 @ 127.0.0.1:8000**/detect_face**

8. 구현을 마친 Face detector 개선하기 (자동 이미지 리사이징)

8-1) C:₩work_django₩django_mldl**₩cv_project₩opencv_webapp** -> **cv_functions.py** 파일 수정하기

(아래 내용을 구글드라이브의 문서로 올려두었습니다.)

...

```
def cv_detect_face(path):
    img = cv2.imread(path, 1)

    if (type(img) is np.ndarray):
        print(img.shape) # 세로, 가로, 채널

        resize_needed = False
        if img.shape[1] > 640: # ex) 가로(img.shape[1])가 1280일 경우,
            resize_needed = True
            new_w = img.shape[1] * (640.0 / img.shape[1]) # 1280 * (640/1280) = 1280 * 0.5
            new_h = img.shape[0] * (640.0 / img.shape[1]) # 기존 세로 * (640/1280) = 기존 세로 * 0.5
        elif img.shape[0] > 480: # ex) 세로(img.shape[0])가 960일 경우,
            resize_needed = True
            new_w = img.shape[1] * (480.0 / img.shape[0]) # 기존 가로 * (480/960) = 기존 가로 * 0.5
            new_h = img.shape[0] * (480.0 / img.shape[0]) # 960 * (480/960) = 960 * 0.5

        if resize_needed == True:
            img = cv2.resize(img, (int(new_w), int(new_h)))

        baseUrl = settings.MEDIA_ROOT_URL + settings.MEDIA_URL
        ...
```

8-2) 장고 프로젝트에서 **큰 이미지 업로드 하여 detection 실행** 후 출력된 이미지 저장하여 크기 확인해보기

python manage.py runserver

브라우저에서 접속 @ 127.0.0.1:8000**/detect_face**

9. 관리자 페이지에서 ImageUploadModel 모델을 관리할 수 있도록 변경하기

9-1) C:₩work_django₩django_mldl**₩cv_project₩opencv_webapp** -> **admin.py** 파일 수정하기

```
from django.contrib import admin
from .models import ImageUploadModel

class image_upload_Admin(admin.ModelAdmin):
    list_display = ('description', 'document', )

admin.site.register(ImageUploadModel, image_upload_Admin)
```

9-2) 관리자 계정 생성하고 장고 프로젝트 Server 실행 후 결과 확인 (cmd)

python manage.py **createsuperuser**

python manage.py runserver

브라우저에서 접속 @ 127.0.0.1:8000**/admin**

-> 위 createsuperuser로 생성한 ID & 비밀번호로 로그인하여 데이터 살펴보기

10. 장고 서비스 배포하기 (with Python anywhere)

* Python Anywhere의 경우 **무료 계정은 파일 저장소 용량이 약 512MB**로 제한되어 있습니다. 따라서 OpenCV와 Numpy를 포함한 필수 라이브러리들의 설치가 거의 불가능합니다. 그러므로 본 파트는 전체적인 웹서비스 배포의 흐름을 이해하는데 초점을 맞추고, 추후 필요할 경우 유료 계정으로 전환 후 배운 내용을 적용하도록 합니다.

10-1) 파이썬 가상환경(django_env) 내에 **설치된 library 목록을 담은 "requirements.txt" 파일 생성**하기

: C:₩work_django₩django_mldl**₩cv_project** 폴더 안에 아래 내용을 따라 "requirements.txt" 파일을 만듭니다.

@ cmd, **Ctrl + C 로 서버 종료** 후 아래 1줄 입력

$ **pip freeze > requirements.txt**

-> 만들어진 requirements.txt 파일을 확인합니다. (cv_project 프로젝트 폴더 안에 위치해있어야 합니다.)

10-2) **".gitignore" 파일 생성**하기

: **Atom에서** C:₩work_django₩django_mldl**₩cv_project** 폴더 안에 "**.gitignore**" 라는 이름의 파일을 만들고 아래의 내용을 작성한 후 저장합니다. (3번째 항목인 "__pycache__"에서 _(밑줄)이 양 옆에 2개씩 있음을 유의합니다.)

* 아래 목록에 해당하는 파일 혹은 폴더는 github에 push할 때 자동으로 제외되어 업로드되지 않습니다.

```
*.pyc
*~
__pycache__
django_env
db.sqlite3
/static
/media/images
.DS_Store
```

10-3) Github 에 소스코드 업로드

*** git bash 에서 403 에러 (permission 관련 에러) 발생 시 : git bash 터미널 계정 변경 방법 @ http://j.mp/2CN1TtT**

- **https://github.com** 접속 후 로그인

- 우측 상단 **New repository** 클릭 후 새로운 저장소 만들기 (저장소 이름 : **django_opencv**)

- **https://git-scm.com/downloads** 접속하여 Git for windows 설치

- **Git bash** 실행

- **git config --list** 입력하여 **user.name & user.email** 입력 여부 확인

- 입력되지 않았을 경우 아래 2줄 실행하여 새로이 입력 후 **git config --list** 입력하여 다시 체크

- **git config --global user.name 사용자명**

- **git config --global user.email 메일주소**

- cd .. 및 cd 폴더명 명령어로 다음 경로까지 이동

  C:₩work_django₩django_mldl₩cv_project

- 생성된 github repository 에 로컬 저장소 (로컬 폴더) 내의 파일 업로드

git init

git status

git add .

git commit -m "first commit"

git remote add origin https://github.com/repositivator/django_opencv.git <- "github.com/" 뒤의 사용자 계정명('repositivator')이 여러분의 Github 계정명으로 변경되어야 합니다. (저장소 생성 후 복사해서 가져오세요.)

git push -u origin master

- **Github repository 웹페이지 새로고침하여** 업로드된 파일 확인

10-4) Python anywhere 회원가입 & web app 추가하기

- **https://www.pythonanywhere.com/** 접속하여 회원가입 & 로그인 (여기서는 "**djangocv1**" 이라는 계정명을 예시로 활용하겠습니다. 각자 자신의 계정을 만들어주세요!)

- **Web 탭** 들어가기

- '**Add a new web app**' 클릭

- **Manual configuration (including virtualenvs)**

- **Python 3.7**

10-5) Python anywhere에 장고 프로젝트 소스코드 업로드하기

- https://www.pythonanywhere.com/ 접속

- **Consoles 탭** 들어가기

- **Bash** 클릭

git clone https://github.com/repositivator/django_opencv.git <- ".git" 앞부분에 **여러분의 github 저장소 링크**를 적어주세요.

ls

virtualenv --python=python3.7 django_env

source django_env/bin/activate

cd django_opencv

~~pip install django==2.2.6~~

~~pip install opencv-python==4.1.1.26~~

~~pip install pillow==5.4.1~~

pip install -r requirements.txt

10-6) **업로드된 settings.py 코드 수정하기** - **(1) Secret key 생성하기**

: **cmd** 혹은 **Python anywhere bash** 등에서 아래 파이썬 코드 복붙하여 실행하기

python <- python shell 실행

import string

import random

chars = ''.join([string.ascii_letters, string.digits, string.punctuation]).replace('₩'', '').replace('"', '').replace('₩₩', '')

SECRET_KEY = ''.join([random.SystemRandom().choice(chars) for i in range(50)])

print(SECRET_KEY)

-> **출력된 str 값 복사**해두기 (ex : +:p}y@nL&|0ouVEVBhl<GO0m,1=:d)O;(3%;*_~47X!&#L*j=t)

10-7) **업로드된 settings.py 코드 수정하기** - (2) 위 **복사해 둔 새로운 Secret key를 적용하고 기타 세팅 진행**

- 새 탭에서 https://www.pythonanywhere.com/ 접속
- **Files 탭** 들어가기
- **django_opencv/cv_project** 의 **settings.py** 클릭
- 열어둔 **settings.py** 에서 아래와 같이 수정

...

SECRET_KEY = '+:p}y@nL&|0ouVEVBhl<GO0m,1=:d)O;(3%;*_~47X!&#L*j=t' # 위에서 미리 복사해 둔 str 값

DEBUG = False

ALLOWED_HOSTS = ['djangocv1.pythonanywhere.com'] # djangocv1 : PythonAnywhere 계정명

...

...

STATIC_URL = '/static/' # 동일

STATIC_ROOT = os.path.join(BASE_DIR, 'static') # 동일

...

MEDIA_URL = '/media/' # 동일

MEDIA_ROOT = os.path.join(BASE_DIR, 'media') # 동일

MEDIA_ROOT_URL = '**/home/djangocv1/django_opencv**' # 수정 (djangocv1 : PythonAnywhere 계정명)

...

-> 상단 **Save** 버튼 클릭

10-8) **Virtualenv 경로 설정 (Web 탭)**

- 새 탭에서 https://www.pythonanywhere.com/ 접속
- **Web 탭** 들어가기
- **Virtualenv** 파트의 "**Enter path to a virtualenv, if desired**" 클릭
- 입력 : **/home/<PythonAnywhere계정명>/django_env** (ex. /home/djangocv1/django_env)

10-9) Static files 중 **media files 경로 설정 (Web 탭)**

- **Static files** 파트의 "**Enter URL**" 클릭
- 입력 : **/media/** <- settings.py의 [MEDIA_URL = '/media/'] 에 해당
- **Static files** 의 "**Enter path**" 클릭
- 입력 : **/home/<PythonAnywhere계정명>/django_opencv/media** (ex. /home/djangocv1/django_opencv/media)

: "django_opencv"는 github 저장소명에 해당

10-10) **WSGI configuration file 수정 (Web 탭)**

- "**Code:**" 파트의 "**WSGI configuration file:**"에 적혀 있는 파일명 클릭
- **전체 선택 & 삭제**
- 아래 내용 붙여넣기 후 수정 진행 & 저장 (PythonAnywhere계정명을 수정해주셔야 합니다!)

```
import os
import sys

path = '/home/<PythonAnywhere계정명>/django_opencv'  # "django_opencv"은 github 저장소명에 해당
if path not in sys.path:
    sys.path.append(path)

os.environ['DJANGO_SETTINGS_MODULE'] = 'cv_project.settings' # "cv_project"은 settings.py 파일 위치

from django.core.wsgi import get_wsgi_application
from django.contrib.staticfiles.handlers import StaticFilesHandler
application = StaticFilesHandler(get_wsgi_application())
```

10-11) Static 파일들을 모으고 관리자 계정 생성하기

: Consoles 탭에서 열어두었던 **Bash**에서 아래 명령어 실행

```
python manage.py collectstatic
python manage.py migrate
python manage.py createsuperuser (PythonAnywhere 계정명 추천)
```

10-12) 변경사항 반영 후 웹서비스 접속하기

- https://www.pythonanywhere.com/ 접속
- **Web 탭** 들어가기
- **Reload** 버튼 클릭
- Reload 완료 후 웹브라우저에서 웹서비스 접속해서 테스트 진행 (URL 뒤에 /detect_face 붙이기)

  ex) http://djangocv1.pythonanywhere.com**/detect_face/**

11. Recommended further works

- [3. Django intermediate] 실습의 16번 파트 이후를 참고하여 **직접 웹사이트를 꾸며보세요**.
- BeautifulSoup & Selenium, Scikit-learn & Tensorflow 등을 활용하여 **여러 기능을 새로운 App으로 추가해보세요**.
- Heroku, MS Azure, AWS Elastic Beanstalk 등 **여러 배포 서비스가 있습니다**. 끌리는 것을 골라 도전해보세요.