Django 서버 구축 가이드

좋은징조 - 이무송

환경 구축

가상환경 구축

웹 서비스 형식으로 서비스를 제공하기 위해서 django framework를 사용한다. Django 웹 서버를 구축하기 위해서 python3 기반의 가상환경을 만든다.

```
$python3 -m venv env
```

가상환경을 적용하기 위해선 다음과 같은 명령어를 수행해 준다 linux 기준으로 명령어는 다음과 같다

```
$source env/bin/activate
(env)$
```

명령 프롬프트 제일 좌측에 괄호로 자신의 가상환경 이름이 나온다면 가상환경 적용이 완료된 것이다.

패키지 설치 Django 웹 서버를 구축하고 원활한 서비스를 수행하기 위해선 다음과 같은 패키지가 필요하다

- django - Django 웹 서버를 구축하기 위한 필수 패키지이다 이 프로젝트에선 2.0.0 버전을 사용한다.
- django-sslserver - MediaDevice를 사용하기 위해서는 django web-server가 ssl이 적용이 되어야한다. 따라서 원활한 서비스와 개발을 위해서 꼭 필요한 패키지이다.

패키지 설치는 다음과 같은 명령어를 입력한다

```
(env)$pip install --upgrade pip
(env)$pip install django~=2.0.0
(env)$pip install django-sslserver
```

웹 서버 구축

프로젝트 생성 Django framework 를 온전히 받았다면 프로젝트를 생성해야 한다. 프로젝트란, django 웹 서버를 구동하기 위한 엔진이다.

프로젝트 생성을 위해 아래의 명령어를 사용한다

```
(env)$django-admin startproject mysite
```

mysite는 임의로 만든 프로젝트 명이고 목적에 따라 프로젝트 이름을 바꾸는 것이 좋다 프로젝트를 생성하면 다음과 같은 파일과 디렉토리가 생성이 된다.

```
mysite
├── manage.py
└── mysite
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py
```

환경을 구축하기 위한 핵심 파일만을 설명하겠다

- manage.py - 서버의 시작과 db적용을 담당한다.
- setting.py - 서버의 정적파일 디렉토리 설정, 허용 host, 설치할 앱 목록, 시간 설정을 담당한다.
- urls.py - 서버에 접속할때 제공하는 url을 설정할 수 있는 곳이다.

앱 생성 Django 서버를 구축할때 실제 서비스를 제공하는 프로그램이다. 앱 생성을 하려면 프로젝트 디렉토리(mysite)안의 아래의 명령어를 사용한다

```
(env)$python manage.py startapp myapp
```

프로젝트 생성과 마찬가지로 myapp 은 임의로 설정한 app이름이고, 프로그램의 목적에 맞게 설정하는 것이 좋다.

앱을 생성하면 앱 이름의 디렉토리와 다음과 같은 파일이 생성된다.

```
myapp
├── __init__.py
├── admin.py
├── apps.py
├── migrations
|   └── __init__.py
├── models.py
├── tests.py
└── views.py
```

환경을 구축하는데의 핵심 파일의 이름과 역할은 다음과 같다. 원활한 환경을 위해서 추가적으로 디렉토리와 파일을 생성해주자 앱 폴더(my app) 에서 다음과 같은 명령어를 입력하자

```
(env)$touch urls.py
(env)$mkdir static
(env)$mkdir templates
(env)$cd templates
(env)$mkdir myapp
(env)$cd ..
(env)$cd static
(env)$mkdir js
```

- models.py - django에서 사용할 데이터를 정의한다. 각 데이터는 object로 되어있고 migration을 통해 db에 구축할 수 있다.
- views.py - urls.py 를 통해 클라이언트가 request 패킷을 전송하면 실제 기능을 수행하는 모듈들을 정의하는 파일이다.
- urls.py - 프로젝트에서 소개했던 것과 비슷하지만 실제 클라이언트가 url을 요청하면 views.py 로 처리를 넘기는 역할을 한다.

- templates 폴더 - 실제 웹 페이지(html)를 보여주기 위해 사용하는 페이지를 모아 놓은 디렉토리이다. 하위 폴더로 myapp을 생성 했는데, 폴더 이름은 임의로 지정해서 각각 목적에 맞게 따로 관리할 수 있다.
- static 폴더 - js 나 css 이미지과 같은 정적 파일을 관리한다. js폴더를 생성했는데 templates 폴더와 같이 목적에 맞는 폴더 명을 적어 각 파일별로 관리 할 수 있다.

서비스 환경 구축

Django 웹 서버가 원할한 서비스를 제공하기 위해선, 몇 가지 설정이 필요하다.

settings.py 설정 프로젝트 폴더에 settings.py 에서 다음과 같이 설정해야 한다.

시간 설정 TIME_ZONE 변수가 UTC로 되어있을 것이다. 다음과 같이 한국 시간대로 변경해주자

```
TIME_ZONE = 'Asia/Seoul'
```

앱 등록 웹 서버가 우리가 생성한 앱의 기능을 수행하기 위해서 앱 등록 절차가 반드시 필요하다 앱 등록은 INSTALLED_APPS 리스트에 다음과 같이 추가하자

```
INSTALLED_APPS = [
...(생략)...
'myapp',
'sslserver',
]
```

myapp 밑에 sslserver를 추가해 주는 이유는 ssl을 적용시켜주기 위해 아까 django-sslserver 패키지를 설치를 하고 사용하기 위해서 적어주었다.

static_root 추가 웹 서버가 원할하게 static 자원을 잘 찾아갈 수 있게 static_root를 설정해줘야한다. static_url 밑에 다음과 같은 줄을 추가하자

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

외부 호스트 연결 허용 웹 서비스가 외부에서 원할히 요청하기 위해서 ALLOWED_HOSTS 변수를 수정할 필요가 있다. 만약 외부에서 요청하고 싶다면 ALLOWED_HOSTS를 다음과 같이 설정하면 된다.

```
ALLOWED_HOSTS = ['*']
```

서비스 제공하기

본격적으로 django에서 서비스를 제공하기 위해서 다음과 같은 설정이 필요하다 이 문서에선 사이트에 접속하면 main_page.html 을 반환해 주는 서비스를 제공 하는 예를 들어 보겠다.

1. 프로젝트 urls.py에서 앱(myapp)의 url을 포함시키기
2. 앱 url을 요청할 때 views.py 에서 해당 url을 처리
3. views.py 에서 main_page.html을 랜더링 처리
4. Templates 밑의 폴더에 main_page.html 작성
5. 서버 migrate 후 서버 실행

프로젝트 url 설정 mysite/urls.py에서 myapp urls.py를 포함 시켜야 myapp기능이 원활하게 작동을한다. 추가해주기 위해서 다음과 같이 urlpatterns를 수정해주자

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

urlpatterns = [
        path('',include('myapp.urls')),
]
```

myapp urls.py 설정 웹 서버에 접속할 때 바로 main_page.html을 띄어주기 위해 myapp/urls.py를 다음과 같이 추가해주자

```
from django.urls import path,include
from . import views
urlpatterns = [
        path('',views.main_process,name='main_process'),
]
```

path의 첫 번째 매개변수를 빈 문자열을 넣으면 웹 사이트에 접속할 때 main_process가 동작을 하게 된다.

views.py 설정 요청받은 url을 처리하기 위해서 views.py 안의 main_process함수를 작성해야 한다. 다음과 같이 작성 해준다.

```
def main_process(request):
    return render(request,'myapp/main_page.html',{})
```

url을 요청하면 main_page.html을 반환하는 코드이다.

main_page.html 작성 templates 폴더 밑에 myapp 폴더 안에 main_page.html을 작성 해준다.

```
<html>
    <head>
        <meta http-equiv="content-type" content="text/html; charset=utf-8">

        <title>main page</title>

    </head>
    <body>
        <h1>this is main page</h1>
    </body>
</html>
```

서버 migrate 후 동작시키기 서버를 최초로 구동하면 migrate작업을 해줘야하는데 하는 방법은 다음과 같다 프로젝트 폴더 안에 다음과 같은 명령어를 입력하자

```
(env)$python manage.py makemigrations myapp
(env)$python manage.py migrate
```

에러가 없으면 성공한 것이다.

서버를 동작하면 보통 runserver command를 사용하는데 ssl을 적용하기 위해서 다음과 같이 서버를 동작시킨다

```
(env)$python manage.py runsslserver
```

https://127.0.0.1:8000 을 접속하면 보안 연결이 되는지 확인하고 사이트 접속을 누르고 main_page.html이 나오는지 확인하자

정적 파일 제공

js나 img파일을 서버에서 제공하기 위해서 다음과 같은 과정이 필요하다

1. static 폴더를 생성해 목적 별로 폴더를 생성한다
2. 폴더 안에 파일을 넣는다
3. Html 파일에서 경로를 넣어준다

test js파일 생성 Static/js/test.js 파일을 다음과 같이 작성하자

```
window.alert('load completed!');
```

html 코드 수정 main_page.html밑에 다음 줄을 추가하자

```
{% load static %}
<html>
...

<script src='static/js/test.js'></script>
```

서버가 동작되어있다면 다시 시작할 필요없이 새로고침을 하여 확인하자