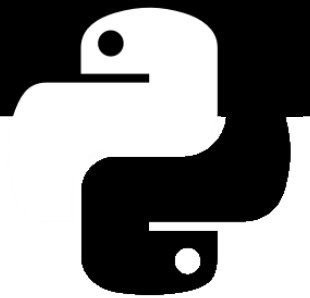
Title(“Django & Data structure basic training with Python”)

정리



2018.11.19

학교 / 순천향대학교
학과 / 정보보호학과
동아리 / 사이버물리시스템보안(CPSS)
발표 / 김준원 (github.com/ruskonert)

Agenda

0x1 / 장고 서버를 위한 환경 구축

0x2 / 장고 주요 기능

0x3 / NULL

Agenda

0x1 / 장고 서버를 위한 환경 구축

0x2 / 장고 주요 기능

0x3 / NULL

1 〉장고 서버를 위한 환경 구축

Virtual Environment (venv) 구축 (python3 기준)

- 패키지의 충돌 방지 및 프로젝트 별로 관리를 효율적으로 할 수 있도록 도와주는 모듈
- 윈도우 운영체제에서는 python 모듈로 기본 탑재되어 별도의 설치가 필요하지 않음

파이썬 모듈을 사용하는 방법

```
python -m [모듈명] [모듈 작동에 필요한 추가적인 인자값] …
```

venv 모듈을 이용한 Virtual Environment 구축 방법 및 가상 인터프리터 활성화 (# : 주석)

```
python -m venv [폴더명]
cd [폴더명]/Scripts
activate.bat # cmd 창에서 실행하는 경우
.\activate.ps1 # Powershell에서 실행하는 경우
```

Linux (python 3 이상)

```
sudo apt-get install python3-pip # python3 pip는 리눅스에선 별도로 설치해야 함
sudo python3 -m pip install venv # venv는 리눅스에선 별도로 설치해야 함
python3 -m venv [폴더명]
cd [폴더명]/bin
source activate
```

1 〉장고 서버를 위한 환경 구축

django 패키지 설치 (python 실행시 항상 venv 활성화 상태)

파이썬 pip 모듈을 이용한 패키지 설치

```
> python -m pip install --upgrade pip # pip 모듈 업그레이드
> python -m pip install django # django 설치
> python -m django --version # Django 인스톨 여부 확인 (버전 확인)
2.1.2
```

Linux

```
> python3 -m pip install --upgrade pip
> python3 -m pip install Django
> python3 -m django -version
2.1.2
```

Next Is …

0x1 / 장고 서버를 위한 환경 구축

0x2 / 장고 주요 기능

0x3 / NULL

2> 장고 주요 기능

장고 명령어 실행하기 (django-admin == manage.py)

- django-admin - 프로젝트 초기 설정 및 인스턴트 생성
- manage.py - 장고가 가지고 있는 기능(서버 시작, 앱 생성, 마이그레이션 등…)을 수행함

장고 프로젝트 뼈대 만들기 (startproject)

- 장고 서버를 돌리기 위한 기본적인 프레임워크를 만드는 명령어
- django-admin을 이용하여 프레임워크 생성 가능

장고 프레임워크 생성을 위한 명령어

```
> django-admin startproject [프로젝트명]
# 명령어 실행 후 디렉토리 구조
프로젝트명/
    manage.py # 기능 수행
    프로젝트명/
        __init__.py
        settings.py # 프로젝트 설정을 위한 파이썬 파일
        urls.py # 어플리케이션의 url 설정을 위한 파이썬 파일
        wsgi.py
```

2> 장고 주요 기능

장고 서버 오픈하기 (runserver)

- 서버를 오픈하기 위한 명령어
- 한 포트에 하나의 서버만 오픈할 수 있음
- 장고 프로젝트 내 파이썬 파일 중 문법 오류, 컴파일 오류 시 서버가 정상적으로 열리지 않음

장고 서버 오픈을 위한 명령어 (리눅스는 python이 아닌 python3로 사용함)

```
> python manage.py runserver [server_ip:port] [--configurations] …
…
Django version 2.1.3, using settings ‘your_project_name.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

장고 서버 오픈을 위한 참고 사항

- runserver 뒤에 바인딩 IP와 포트를 지정하여 서버를 외부에 오픈할 수 있으며 포트를 지정하여 전형적인 웹 서버처럼 사용할 수 있음 (예시: 80번 포트(HTTP))
- **--noreload, --nothreading** 옵션 인자를 붙여서 소스 코드가 바뀔 때 마다 실시간으로 리로드할 수 있음

2> 장고 주요 기능

장고 어플리케이션(앱) 만들기 (startapp)

- 앱(application): 특정한 기능을 수행하는 페이지 또는 웹 서비스 (게시물, 회원가입 서비스, 로그인, 홈페이지, …)
- 앱을 통해 서브 URL을 나누며 클라이언트가 URL를 통해 웹 페이지에 접근할 때, 각각의 URL에 대해서 웹 서비스를 매칭할지 또는 어떻게 페이지를 표시해줄 지에 대한 디자인을 작성함

장고 앱 생성을 위한 명령어 (리눅스는 python이 아닌 python3로 사용함)

```
> python manage.py startapp [앱이름]
# 명령어 실행 후 디렉토리 구조
[앱이름]/
    __init__.py
    admin.py    # 관리자 페이지를 위한 파일, 생성되지 않을 수도 있음
    apps.py     # 어플리케이션 이름 및 기본 속성 설정
    migrations/ # 모델 작성 후 데이터베이스에 뼈대를 만들기 위한 재료
        __init__.py
    models.py   # 데이터베이스에 값을 저장하기 위한 데이터 설계
    tests.py    # 어플리케이션 테스트를 위하 파일, 생성되지 않을 수도 있음
    views.py    # 페이지 응답을 위한 파일
    urls.py     # URL 지정을 위한 파일, 생성되지 않을 수도 있음
```

2> 장고 주요 기능

장고 뷰 만들기

- 뷰(view): 클라이언트에게 페이지를 보여주기 위한 메소드의 집합
- 어떤 어플리케이션에 대한 페이지 형태 지정 및 출력 전달을 담당하는 요소
- views.py에 정의되어 있는 메소드를 통해서 클라이언트에게 페이지 소스를 송신하거나 전달함

뷰(view)를 위한 메소드의 기본 형태

- 메소드 이름은 특별한 룰이 존재하지 않지만, 인자 값을 최소 1개 이상 받는 형태여야 함
- 첫번째 인자 값에는 GET, POST같은 메소드 방식, 클라이언트와 관련된 값 등을 넘겨 받음

```
ex) def index(request) (O)
    def example(request_value) (O)
    def blob(req, other_value) (O)
    def index() (X)
```

HttpResponse(ex) 함수

- Django.http 모듈에 있는 함수
- 클라이언트에게 전달하고자 하는 H문자열을 보내주는 함수
- **ex** 인자에 문자열, HTML 소스 코드로 이루어진 문자열 등 모든 문자열에 대한 값을 클라이언트로 송신해줄 수 있음

2> 장고 주요 기능

장고 뷰 만들기

- HttpResponse을 이용한 예시

```
# 파일 위치 : 앱이름/views.py

from Django.http import HttpResponse # HttpResponse 메소드는 Django.http 모듈에 정의되어 있음


def my_function(request):
    return HttpResponse("Hello, world!")

def blob(req):
    return HttpResponse("blob :)")
```

2> 장고 주요 기능

뷰 URL 연결해주기

- 뷰(view)에 정의해둔 메소드를 실행하여 클라이언트에게 소스를 보내기 위해서 각각의 메소드에 대해서 URL를 매칭해야 할 필요가 있음
- 어플리케이션 폴더에 존재하는 urls.py를 통해 정의할 수 있으며, urlpattern 변수를 만들어서 URL과 메소드를 맵핑시킬 수 있음

URL 맵핑을 위한 메소드 (path)

- django.urls 모듈에 정의되어 있는 path 함수를 이용하여 맵핑할 수 있음
- **path(url, method, name)**
  - url: 맵핑하고자 하는 url
  - method_name: url로 접속했을 때 실행되는 메소드
  - name: 이름 (url를 통해 인자를 넘겨받을 때, 넘겨받는 값을 지정해주기 위해 유용하게 사용됨)

```
# 파일 위치 : 앱이름/urls.py

from django.urls import path # path 메소드를 사용하기 위해 모듈 가져오기

from . import views # views 모듈이 urls.py와 같은 디렉토리에 있으므로 . 을 이용해 디렉토리 지정

urlpatterns = [
    path('test/', views.my_function, name='my_function_name'),
    path('blob/', views.blob, name='blob_function'),
]
```

2> 장고 주요 기능

뷰 URL를 어플리케이션 URL에 연결해주기

- 프로젝트 디렉토리에 존재하는 urls.py에 어플리케이션과 관련한 url을 맵핑함
- include 함수를 통해 어플리케이션 url 맵핑에 대한 정보를 담고 있는 파일의 경로를 기재하여 연결
  메인 어플리케이션 url에 연결시킴

```
# 파일 위치 : 프로젝트폴더/urls.py

from django.urls import include, path # include 메소드 또한 django.urls 모듈에 정의되어 있음

urlpatterns = [
    path('myapplication/', include('어플리케이션명.urls')),
]
```

예제 소스의 URL 최종 매칭 결과 (서버 명령어 기준 - python manage.py runserver 8081)

http://localhost:8081/myapplication/ → 페이지 존재하지 않음 (404 에러 발생, '' 에 대한 url 맵핑이 없음)
http://localhost:8081/myapplication/test/ → 어플리케이션명/views.py 내 my_function 함수의 반환값을 받음
http://localhost:8081/myapplication/blob/ → 어플리케이션명/views.py 내 blob 함수의 반환값을 받음
http://localhost:8081/myapplication/other/ → 404 에러 ('other'에 대한 url 맵밍이 없음)
http://localhost:8081/myapplication2/ → 404 에러

2> 장고 주요 기능

데이터베이스에 저장하기 위한 모델 작성하기

- 데이터베이스(Database): 웹 서버에 사용되는 데이터를 담는 스토리지
- 게시물 데이터, 회원 정보 등 웹 서비스를 위한 데이터를 저장하기 위해 필요함
- 장고에서는 데이터베이스에 어떠한 정보를 담을 때, 그 정보의 형태를 잡아 주기 위한 모델(model)을 만들어야 하며, 필드를 만들기 위해서는 XXXXField 메소드를 통해 타입을 정의하여 설계함
- 어플리케이션 내 models.py에 클래스를 만들어서 정보에 대한 모델을 작성할 수 있음

- CharField - 문자열을 담기 위한 데이터 필드(길이 제한 존재)
- IntegerField - 숫자값을 담기 위한 데이터 필드
- TextField - 문자열을 담기 위한 데이터 필드(길이 제한 없음)
- DateField - 날짜를 담기 위한 데이터 필드(예시: yyyy-mm-dd-HH:mm:ss)

```
# 파일 위치 : 앱이름/models.py
from django.db import model

class MyModel(models.Model):
    FieldNumber = models.CharField(max_length=200)
    FieldInteger = models.IntegerField()
    FieldDate = models.DateField( ... )
```

2> 장고 주요 기능

작성한 모델 → 데이터베이스 테이블(형태)로 변환 (makemigration)

- 작성한 모델을 데이터베이스에 호환되는 데이터 형식으로 바꿔 데이터베이스에 등록함
- 모델이 등록된 이후에 만들어진 객체를 저장하거나 수정 및 삭제를 할 수 있음
- 모델을 사용하기 전 settings.py에 INSTALLED_APPS 리스트에 어플리케이션 Config 클래스를 추가 (apps.py 내 Config 클래스)하여 어플리케이션 활성화함

```
# 파일 위치 : 앱이름/models.py
from django.db import models

class MyModel(models.Model):
    FieldNumber = models.CharField(max_length=200)
    FieldInteger = models.IntegerField()
    FieldDate = models.DateField( ... )

# 파일 위치: 앱이름/apps.py (manage.py에 의해 자동 생성됨)
from django.apps import AppConfig

class 앱이름Config(AppConfig):
name = '앱이름'
```

```
# 파일 위치 : 프로젝트이름/settings.py

INSTALLED_APPS = [
'django.contrib.admin',
'django.contrib.auth',
'django.contrib.contenttypes',
'django.contrib.sessions',
'django.contrib.messages',
'django.contrib.staticfiles',
'앱이름.apps.앱이름config', # 자신이 만든 config 클래스 추가
]


…
```

```
> python manage.py makemigration <앱 이름>
 - Create model MyModel
…
```

2> 장고 주요 기능

적용 안된 마이그레이션 생성 최종 반영하기 (migrate)

- makemigration 명령어를 통해 생성된 마이그레이션 모델을 장고 서버에 최종 반영함
- 모델이 변경될 때 migrate 명령어를 통해서 변경 사항을 적용할 수 있음

```
> python manage.py migrate <앱 이름>
# 명령어를 실행하였을 때 사용자마다 다르게 나올 수 있음
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, 앱
Running migrations:
  Rendering model states… DONE
  Applying 앱.0001_initial... OK
```

2> 장고 주요 기능

웹 페이지 소스 코드 전송 및 URL 매핑

- HTML 소스 코드에 장고 스크립트 문을 삽입하여 파이썬에서 동작하는 변수 값에 매칭할 수 있으며 동적인 소스 코드를 설계할 수 있음
- template.get_template() 메소드를 이용해 렌더링 하고자 하는 html 소스를 가져오며, render() 메소드를 통해 장고 스크립트를 거쳐 나온 최종 소스 코드를 클라이언트에게 전송할수 있음
- 어플리케이션 폴더 내 templates 폴더에 있는 파일을 이용하여 페이지를 렌더링함

장고만 가지고 있는 HTML 스크립트

- {% … %} - 장고 스크립트 (for, if, load …)
- {{ … }} - 변수값을 참고하기 위한 스크립트

2> 장고 주요 기능

웹 페이지 소스 코드 전송 및 URL 매핑

- 계정 정보를 담은 리스트를 이용하여 HTML 렌더링 예시

```
# 파일 위치 : 앱이름/templates/index.html
!<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Main page</title>
</head>
<body>
{% if account_list % }
<ul>
{% for account in account_list %}
<li>
<a>계정이름 : {{ account.id }}, 점수: {{ account.score }}</a>
</li>
{% endfor %}
</ul>
{% else %}
<p>계정 리스트가 존재하지 않습니다.</p>
{% endif %}
</body>
</html>
```

```
# 파일 위치 : 앱이름/views.py

from django.http import HttpResponse
from django.template import loader

from .models import Account


def index_function(request):
informations = []
informations.append(Account(id="Account1",score=100))
informations.append(Account(id="Account22",score=96))
informations.append(Account(id="Account333",score=80))
template = loader.get_template('index.html')
context = {
'account_list': informations,
}
return HttpResponse(template.render(context, request))
```

2> 장고 주요 기능

웹 페이지 소스 코드 전송 및 URL 매핑

- 계정 정보를 담은 리스트를 이용하여 HTML 렌더링 예시

```
# 파일 위치 : 앱이름/models.py

from django.db import models
class Account(model.Models):
    id = models.CharField(max_length=256)
    score = models.IntegerField()
```

```
# 파일 위치 : 앱이름/urls.py

from django.urls import path
from . import views
urlpatterns = [
path('suburl', views.index_function),
]
```

```
# 파일 위치 : 프로젝트이름/urls.py

from django.urls import path, include

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

```
# python manage.py runserver 8081
# GET http://localhost:8081/test/suburl
# Result:

계정이름 : Account1, 점수: 100
계정이름 : Account2, 점수: 96
계정이름 : Account3, 점수: 80
```

수고하셨습니다!

궁금하신 내용은 ruskonert@gmail.com로 이메일을 보내주시면 아는 한도
내에 성심껏 답변해드리겠습니다.

해당 자료는 Django documentation
(https://docs.djangoproject.com/ko/2.1/intro/)
에서 참고하여 만들었습니다.

자료는 https://github.com/CPSSOpenSource/CPSS_Basic_PPT에서도 보실 수 있습니다.