

Python web FrameWork 털어보기

2023.08.31



코치 세션의 내용은 모두 정답인 방법이 아닙니다.
실습코치들의 경험과 방법을 바탕으로
교육생분들의 더 원활한 프로젝트를 위해 제공해드립니다.

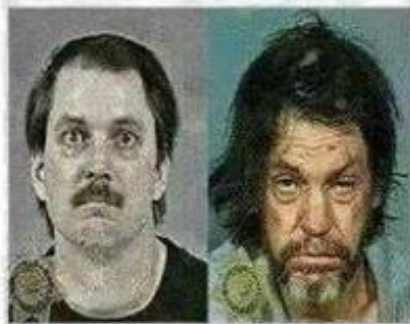


Python을 왜 씀?

BEFORE AND AFTER CODING



c++



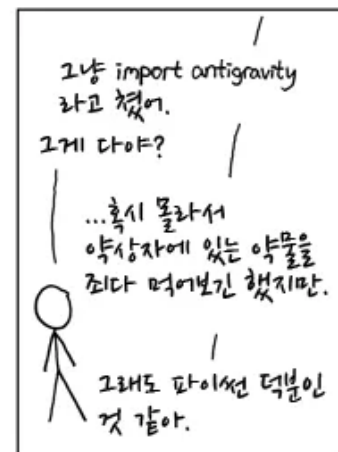
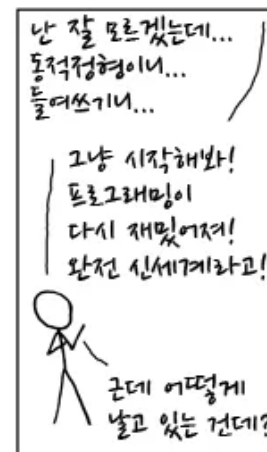
javascript



java



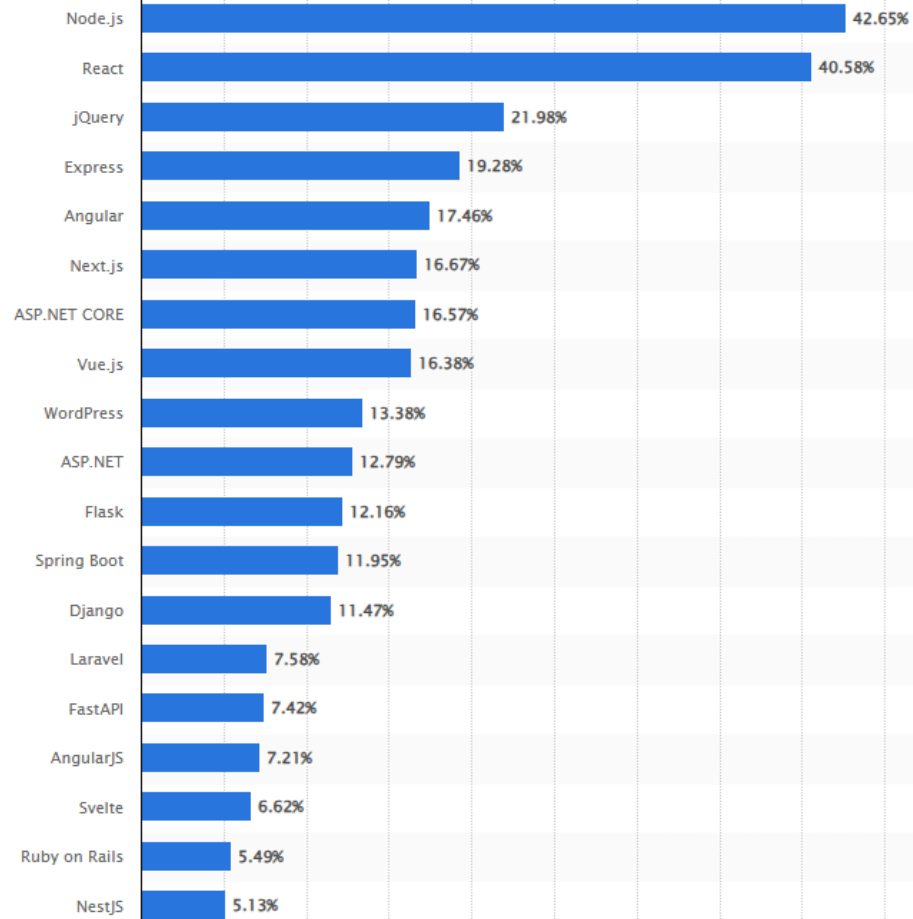
python



Python을 왜 씀?

Technology & Telecommunications > Software

Most used web frameworks among developers worldwide, as of 2023



DOWNLOAD



Source

- [Show sources information](#)
- [Show publisher information](#)
- [Use Ask Statista Research Service](#)

Release date

June 2023

Region

Worldwide

Survey time period

May 8, 2023 to May 19, 2023

Number of respondents

71,802 respondents

Special properties

Software developers

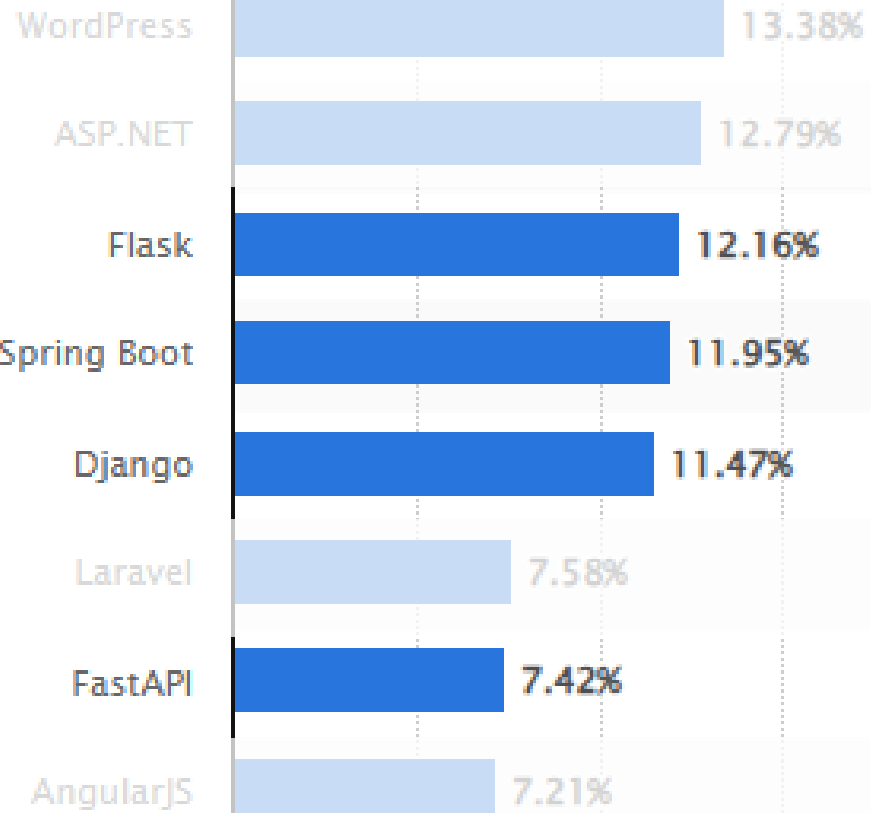
Method of interview

Online survey

Supplementary notes

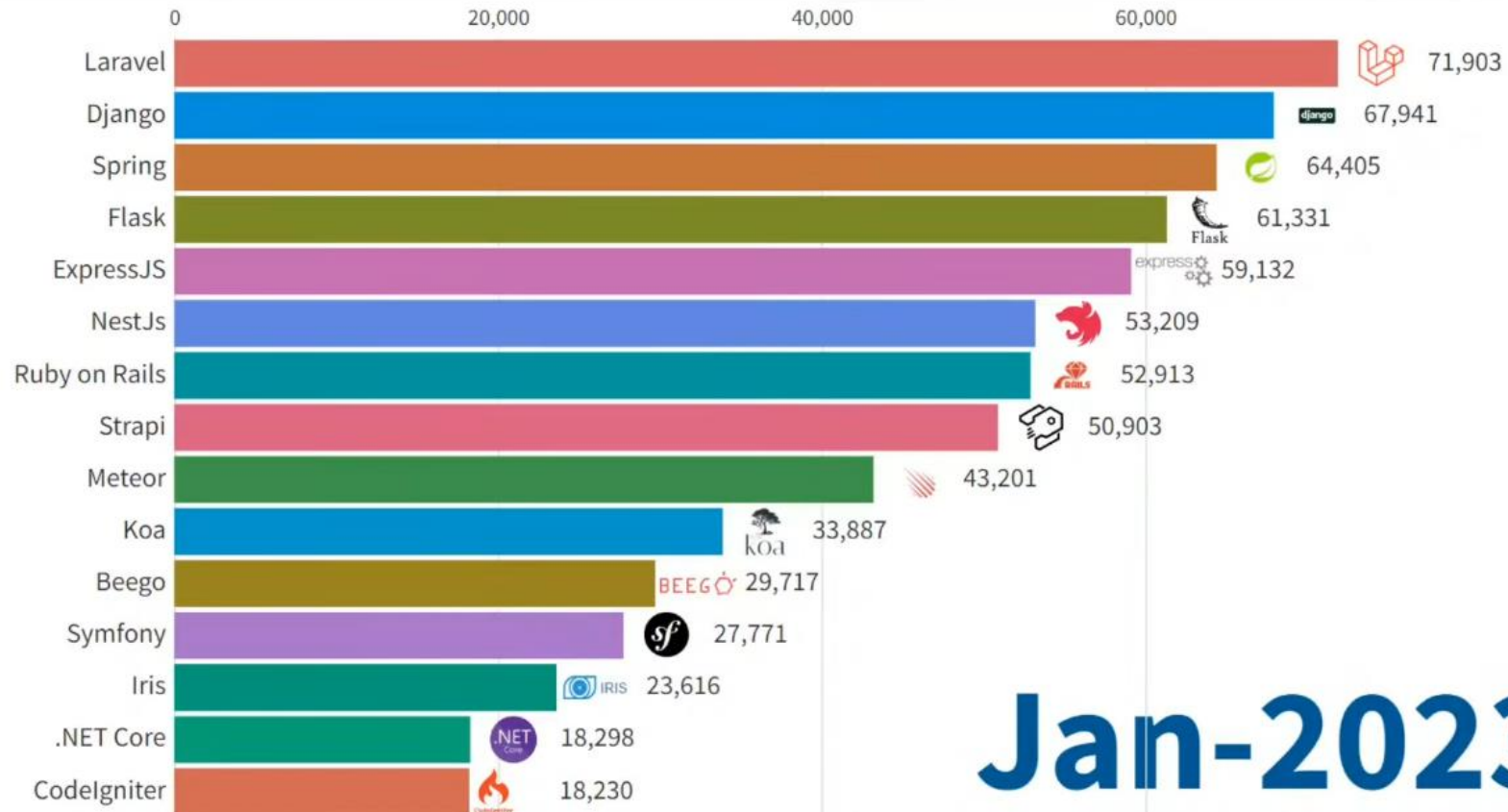
Multiple responses were possible.

Python을 왜 씀?



Python을 왜 씀?

Most Popular Backend Frameworks



Jan-2023

Python을 왜 씀?



네 생각엔 백엔드 프레임워크 중에 어떤 게 가장 좋은 것 같아?



저의 지식이 2021년 9월까지 업데이트된 것이라서 현재 가장 좋은 백엔드 프레임워크에 대한 최신 정보를 제공해드릴 수는 없습니다. 하지만 2021년까지의 시점에서는 다양한 백엔드 프레임워크들이 많이 사용되었습니다. 어떤 프레임워크가 가장 좋은지는 프로젝트의 요구사항, 개발자의 선호도, 커뮤니티 지원 등에 따라 다를 수 있습니다.

이외에도 많은 다양한 백엔드 프레임워크들이 존재하며, 어떤 프레임워크가 가장 좋은지 판단하기 위해서는 프로젝트의 목표와 요구사항, 개발 팀의 기술 스택과 선호도, 커뮤니티 지원, 문서화, 보안 등을 고려하는 것이 중요합니다. 최신 동향과 실제 사용자의 경험을 참고하여 결정하는 것이 좋습니다.

Python을 왜 씀?



일반적으로 많이 사용되는 백엔드 프레임워크로는 다음과 같은 것들이 있습니다.

1. **Django**: 파이썬 기반의 프레임워크로, 빠른 개발과 강력한 보안 기능을 제공합니다. 대규모 웹 어플리케이션을 개발할 수 있고, 다양한 기능을 내장하고 있어서 생산성이 높습니다.
2. **Ruby on Rails**: 루비 기반의 프레임워크로, 개발자 친화적인 구조와 코드 생성 기능으로 빠른 개발을 지원합니다. 컨벤션 오버 설정 원칙을 따라 개발자들이 일관된 코드를 작성할 수 있습니다.
3. **Spring Boot**: 자바 기반의 프레임워크로, 기업용 애플리케이션을 개발하기 위한 많은 기능과 라이브러리를 제공합니다. 스프링 생태계의 다양한 모듈과 통합이 가능합니다.
4. **Express.js**: Node.js 기반의 경량 백엔드 프레임워크로, 빠른 속도와 간결한 코드 작성을 지원하며, JavaScript로 서버를 구축하는 데 적합합니다.
5. **Laravel**: PHP 기반의 프레임워크로, 심플한 문법과 다양한 기능을 제공합니다. 초보자부터 전문가까지 다양한 개발자에게 유용할 수 있습니다.
6. **ASP.NET Core**: C# 기반의 프레임워크로, 크로스 플랫폼 지원과 높은 성능을 갖추고 있습니다. 마이크로서비스 아키텍처를 구축하는 데도 유용합니다.

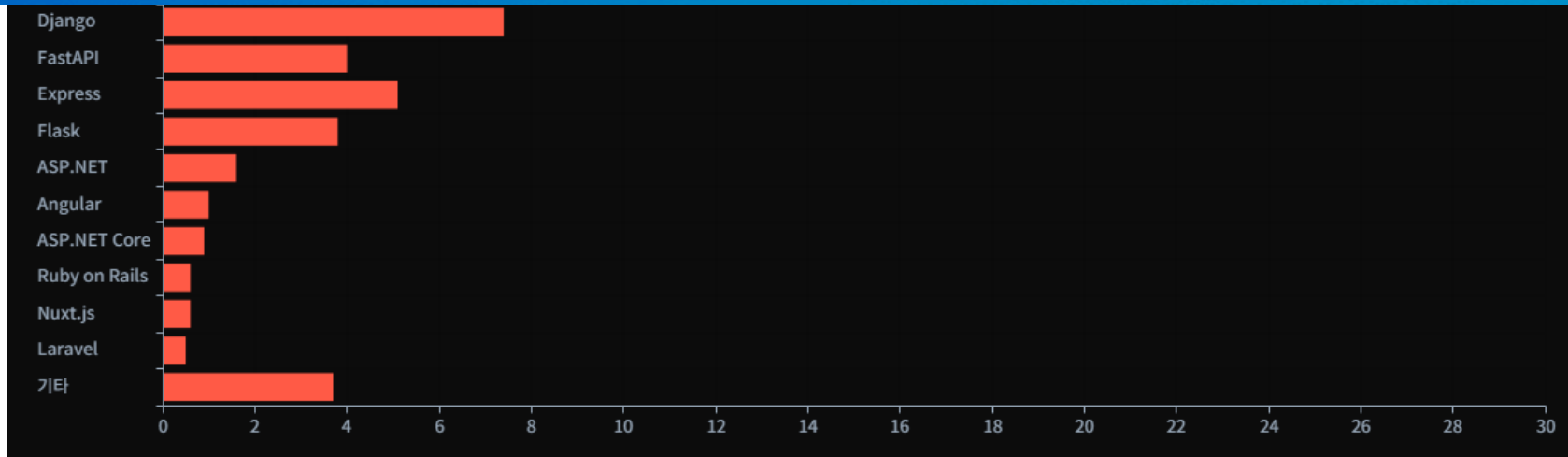
Python을 왜 씀?

주로 사용하는 웹 프레임워크 또는 라이브러리는 무엇인가요?

전자정부 표준프레임워크 v4.1.0 배포안내

E GOVERNMENT STANDARD FRAMEWORK

실행환경 OSS 업그레이드, Eclipse 2022-03 (4.23.0) 적용
공통컴포넌트 기능 추가 및 개선
(DBMS 골디락스 & 모바일 운전면허증 연계 추가)



Python을 왜 씀?



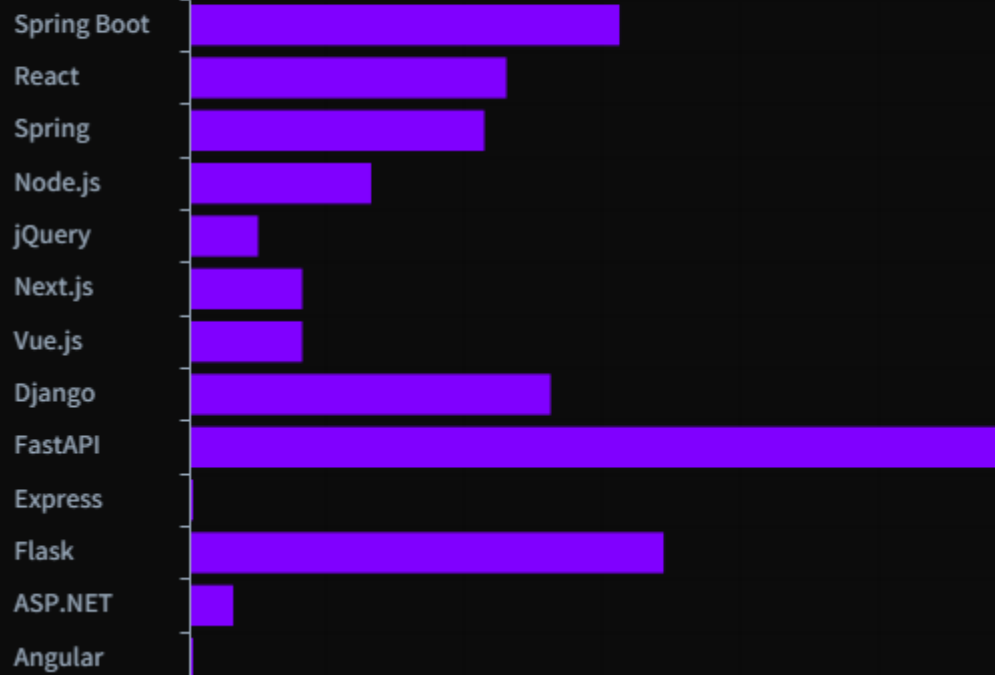
서버/백엔드/풀스택

프론트엔드

머신러닝/AI/데이터분석

앱개발

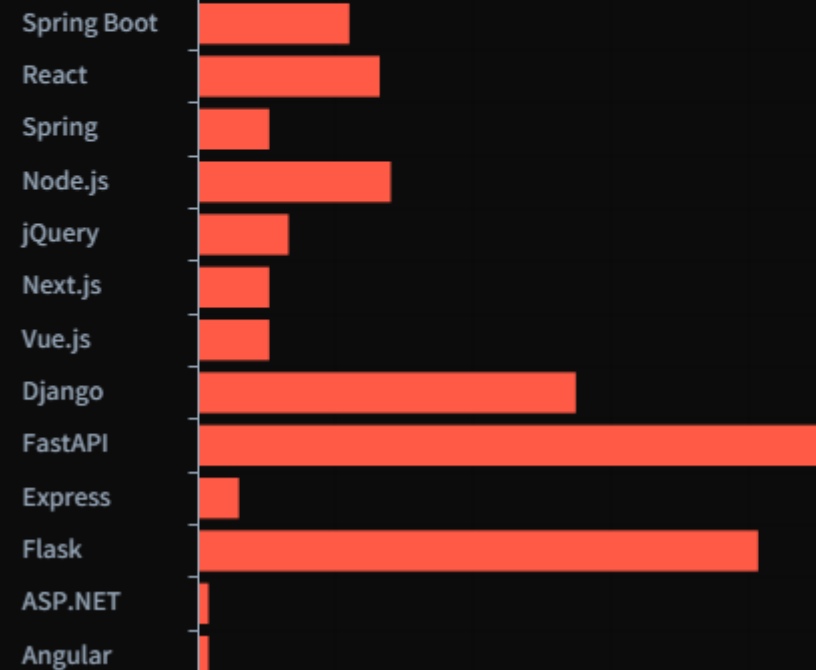
데이터 엔지니어



서버/백엔드/풀스택

프론트엔드

머신러닝/AI/데이터분석



Python을 왜 씀?



ChatGPT API



TensorFlow.js

Python을 왜 씀?

 FastAPI

django



Flask



Django VS Flask





Flask vs Django:
2023년에 사용할 Python 웹 프레임워크는 무엇입니까?
(hackr.io)

Django VS Flask



Django	Flask
강력한 기본 기능(배터리 포함) 내장 템플릿, 내장 ORM, 관리자...	미니멀한 구조로 필요한 것만 선택 가능
학습 곡선이 다소 가파름	시작하기는 쉬우나, 잘 쓰기는 힘들
프로젝트 기본 구조가 복잡함	복잡한 어플리케이션에선 추가 구현 필요
공통된 패턴과 컨벤션을 제공	자유로운 패턴
3.0 버전부터 비동기 처리 기본 제공	기본적으로 동기 처리, 비동기 확장 가능
대규모 어플리케이션에 적합	작은 프로젝트 및 프로토타이핑 적합
높은 보안성	보안 및 권한 부여 기능 부족



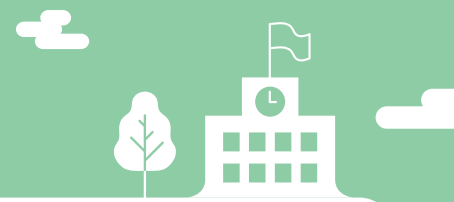
**DB의 User 테이블의 id가 1인 row를 들고 와서
view_count column에 1을 추가하고
결과를 반환하는 POST요청 API**

Django VS Flask

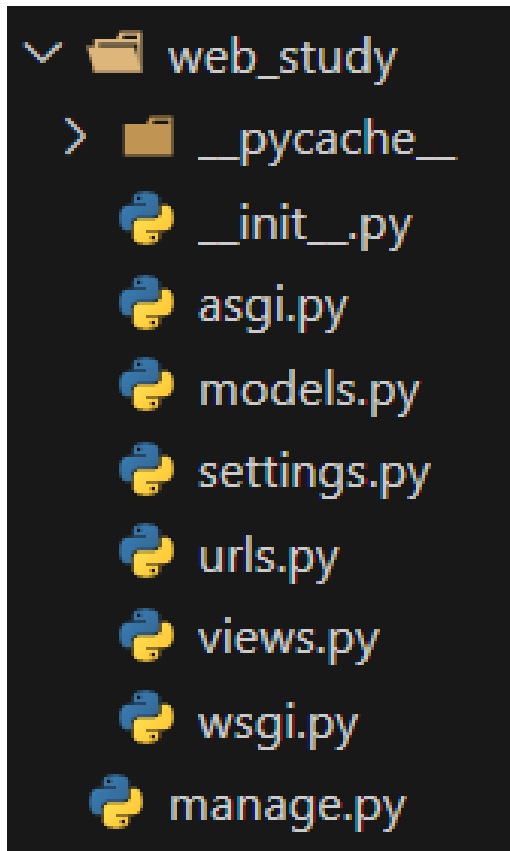


django

Django VS Flask



- 폴더구조



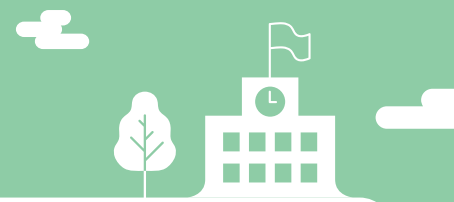
- settings.py

```
INSTALLED_APPS = [  
    'web_study',  
  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',
```

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.mysql',  
        'NAME': 'web_study',  
        'USER': 'root',  
        'PASSWORD':  
        'HOST' :  
        'PORT' : '3306',  
    }  
}
```

이러면 안 됨

Django VS Flask



- **models.py**

```
1 from django.db import models
2
3
4 class User(models.Model):
5
6     id = models.IntegerField(primary_key=True)
7     name = models.TextField(max_length=50)
8     view_count = models.IntegerField()
9
10    class Meta:
11        db_table = 'User'
```

- **urls.py**

```
from django.contrib import admin
from django.urls import path
from .views import add_view_cnt

urlpatterns = [
    path('admin/', admin.site.urls),
    path('add/', add_view_cnt, name='add_view_cnt'),
]
```

- **views.py**

```
django > web_study > views.py > add_view_cnt
1 from .models import User
2 from django.http import JsonResponse
3
4 def add_view_cnt(request):
5
6     if request.method == 'post':
7
8         user = User.objects.get(id=1)
9         user.view_count = user.view_count + 1
10        user.save()
11
12        user_data = {
13            'id' : user.id,
14            'name' : user.name,
15            'view_count' : user.view_count
16        }
17
18        return JsonResponse(user_data, content_type='ap
19
```

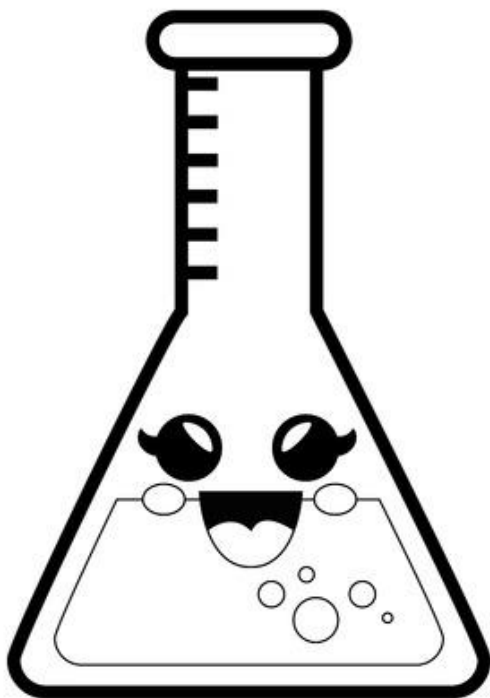
Django VS Flask



cnt += 1

- 쓸 데 없이 많은 파일과 복잡한 구조
- 배터리 포함으로 인한 오버헤드

Django VS Flask



app.py

flask > app.py > ...

```
1 from flask import Flask
2 from sqlalchemy import create_engine
3 from sqlalchemy.ext.automap import automap_base
4 from sqlalchemy.orm import sessionmaker
```

```
5
6 db_url = f'mysql+mysqldb://root:[REDACTED]@3306/web_study'
7 engine = create_engine(url=db_url)
```

```
8
9 Base = automap_base()
10 Base.prepare(engine)
11 User = Base.classes.User
12 session = sessionmaker(autoflush=True, bind=engine)
```

```
13
14 app = Flask(__name__)
```

```
15
16 @app.route('/add', methods=['post'])
```

```
17 def add_view_cnt():
18
19     with session() as db:
20         user = db.query(User).get(1)
21         user.view_count += 1
22         db.commit()
23         result = {
24             'id' : user.id,
25             'name' : user.name,
26             'view_count' : user.view_count
27         }
28
29     return result
30
```

모델 정의

DB연결

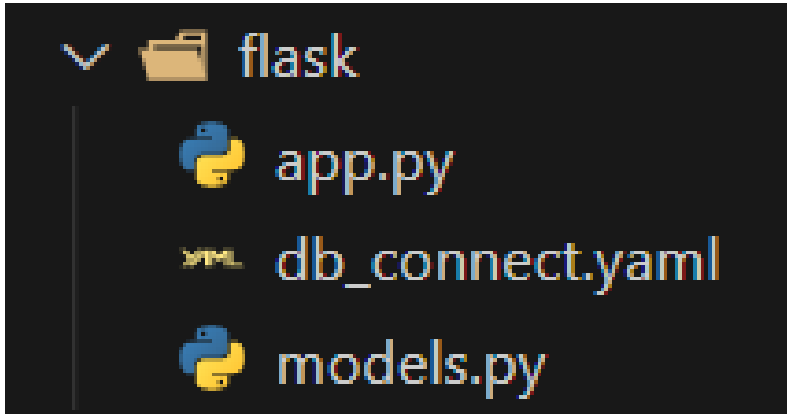
엔드포인트 설정

View 로직

Django VS Flask



Django VS Flask



- 쉬운 시작과 자유로운 패턴은 양날의 검
- DB 연결 정보는 당연히 따로 보관
- 최소한 뷰와 모델은 분리하자

Django VS Flask



▼ RECOMMENDSERVICE

▼ recommend

▼ ai_model

</> 10emotions_model_state_dict...

</> 10emotions_model_state_dict...

▼ views

feedback.py

song_recommend.py

__init__.py

ai_module.py

data_module.py

models.py

.gitignore

config.yaml

Dockerfile

requirements.txt

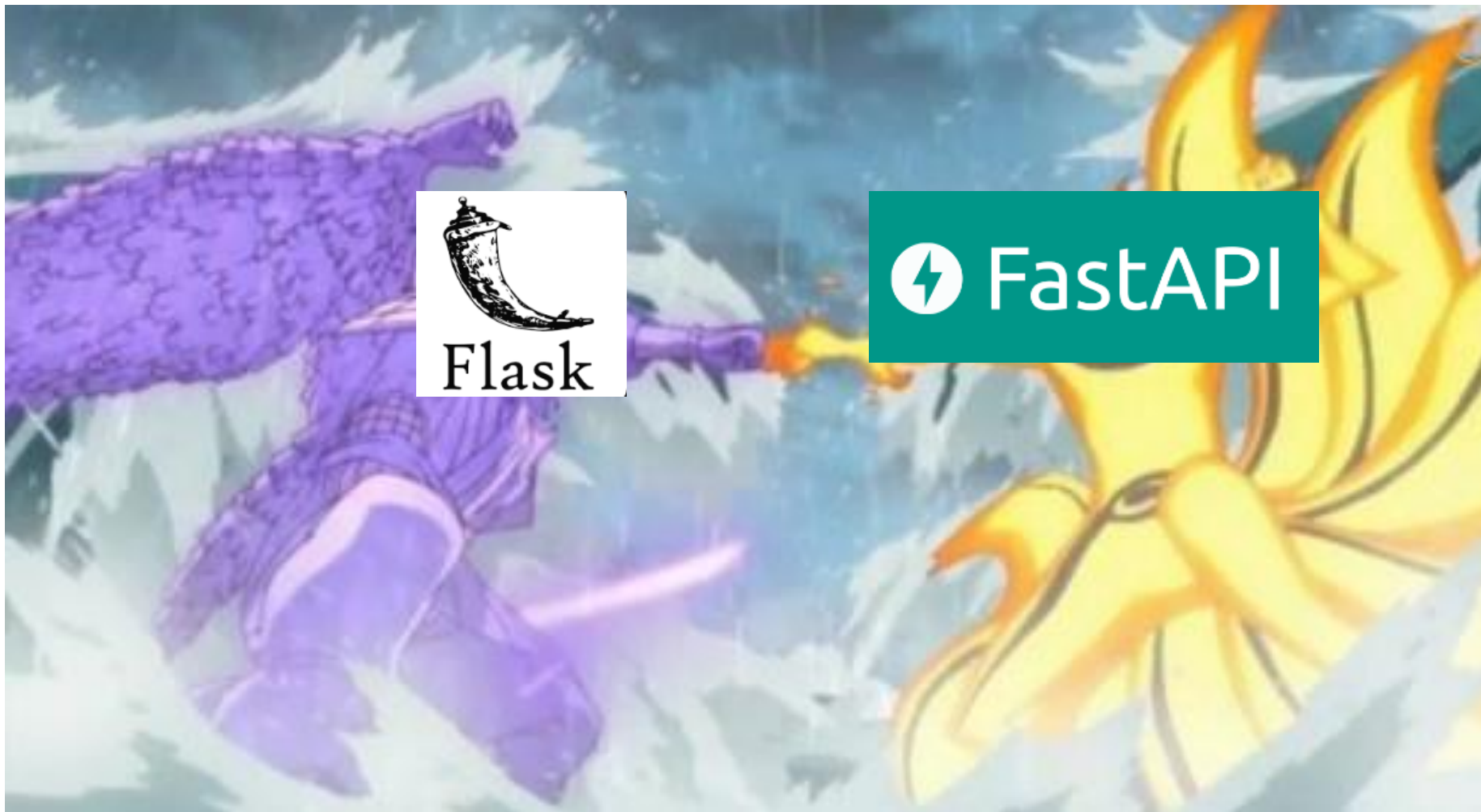
- 한국어 텍스트 감정 분류 모델을 통한 노래 추천
추천 결과 피드백
- Model, View를 분리
- 서버가 갖고 있을 정보들을 _module.py에서 로드
- 이는 예시일 뿐, 문제만 없도록 자유롭게 설계

Django VS Flask



Django	Flask
강력한 기본 기능	미니멀한 구조로 필요한 것만 선택 가능
대규모 어플리케이션에 적합	작은 프로젝트 및 프로토타이핑 적합

FastAPI





**DB의 User 테이블의 id가 1인 row를 들고 와서
view_count column에 1을 추가하고
결과를 반환하는 POST요청 API**

FastAPI



```
app.py X
flask > app.py > ...
1 from flask import Flask
2 from sqlalchemy import create_engine
3 from sqlalchemy.ext.automap import automap_base
4 from sqlalchemy.orm import sessionmaker
5
6 db_url = f'mysql+mysqldb://root:[REDACTED]'
7 engine = create_engine(url=db_url)
8
9 Base = automap_base()
10 Base.prepare(engine)
11 User = Base.classes.User
12 session = sessionmaker(autoflush=True, bind=engine)
13
14 app = Flask(__name__)
15
16 @app.route('/add', methods=['post'])
17 def add_view_cnt():
18
19     with session() as db:
20         user = db.query(User).get(1)
21         user.view_count += 1
22         db.commit()
23         result = {
24             'id' : user.id,
25             'name' : user.name,
26             'view_count' : user.view_count
27         }
28
29     return result
30
```

```
app.py X
fastapi > app.py > ...
1 from fastapi import FastAPI, Request
2 from sqlalchemy import create_engine
3 from sqlalchemy.ext.automap import automap_base
4 from sqlalchemy.orm import sessionmaker
5
6 db_url = f'mysql+mysqldb://root:[REDACTED]'
7 engine = create_engine(url=db_url)
8
9 Base = automap_base()
10 Base.prepare(engine)
11 User = Base.classes.User
12 session = sessionmaker(autoflush=True, bind=engine)
13
14 app = FastAPI()
15
16 @app.post('/add')
17 def add_view_cnt(request:Request):
18
19     with session() as db:
20         user = db.query(User).get(1)
21         user.view_count += 1
22         db.commit()
23         result = {
24             'id' : user.id,
25             'name' : user.name,
26             'view_count' : user.view_count
27         }
28     return result
29
```



타입 힌팅

```
@app.post('/add')  
def add_view_cnt(request: Request):  
    |
```



Python 자체에서는 타입의 가이드라인. 강제성 X

```
test.py > ...  
1  def add(a:str, b:str) -> str:  
2      return a+b  
3  
4  print(add(1,2))
```

문제 출력 디버그 콘솔 터미널

```
SSAFY@DESKTOP-1JVI7QI MINGW64 ~/Desktop/webframework  
$ python test.py  
3
```



강제성을 가짐! 경로 매개변수, 함수 매개변수 모두!

```
@app.get('/add/{a:int}/{b:int}')
def int_int(a:int, b:int):
    result = a+b
    return {'result' : result}

@app.get('/add/{a:str}/{b:int}')
def str_int(a:int, b:int):
    return {'여기선': '에러를 뱉습니다!'}

@app.get('/add/{a:str}/{b:str}')
def str_str(a, b):
    return {'이렇게' : '''다른 엔드포인트로
인식을 하게 됩니다'''}
```

```
127.0.0.1:8000/add/1/2
{
  "이렇게": "다른 엔드포인트로 인식하게 됩니다"
}
```

```
127.0.0.1:8000/add/a/1
{
  "detail": [
    {
      "type": "string_type",
      "loc": [
        "path",
        "b"
      ],
      "msg": "Input should be a valid string",
      "input": 1,
      "url": "https://errors.pydantic.dev/2.3/v/string_type"
    }
  ]
}
```

```
127.0.0.1:8000/add/a/b
{
  "이렇게": "다른 엔드포인트로 인식하게 됩니다"
}
```



스웨거 자동 생성 (/docs or /redoc)

```
app.py > mul
1  from fastapi import FastAPI
2
3  app = FastAPI()
4
5  @app.get('/add/{a:int}/{b:int}')
6  def add(a,b):
7      return {'result' : a+b}
8
9  @app.get('/minus/{a:int}/{b:int}')
10 def minus(a:int, b:int):
11     return {'result' : a-b}
12
13 @app.get('/mul/{a}/{b}')
14 def mul(a:int, b:int):
15     return {'result' : a*b}
```

FastAPI

0.1.0

OAS 3.1

</openapi.json>

default

GET

</add/{a}/{b}> Add

GET

</minus/{a}/{b}> Minus

GET

</mul/{a}/{b}> Mul

Schemas

HTTPValidationError > Expand all object



생년월일 : 2018 / 12 / 5 (아직 유치원생)

Django : 2005년, 고3

Flask : 2010년, 중1

닉값하는 프레임워크

Restful API 구축에 최적화 (풀스택으로 쓰는 것도 가능하긴 함)

비동기 처리, 자동 API문서 생성, 타입 힌팅 및 검사

데이터 직렬화, WebSocket, OAuth2, JWT, CORS 등등...



GPT야! 어떤 웹 프레임워크를 선택하는 게 좋을까?



안녕하세요! 웹 프레임워크를 선택할 때는 프로젝트의 목표, 요구사항, 개발자의 경험 등 다양한 요소를 고려해야 합니다. 어떤 웹 프레임워크가 가장 좋은지 판단하는 것은 프로젝트의 성격에 따라 다를 수 있습니다. 그래도 몇 가지 일반적인 팁을 제공해드릴게요:

상황을 고려하여 적절히 선택하자!

9기 여러분의 “폭풍 성장을 응원합니다!!”

코치도 응원합니다!!

