

07 PJT

API Server 제작

챕터의 포인트

- 목표
- [실습] 날씨 데이터를 활용한 REST API Server 구축

목표

프로젝트 목표

- REST API 서버를 직접 구현합니다.
- 우리는 그 동안 OpenWeatherMap API 와 같이

누군가 만들어준 REST API Server를 사용하는 클라이언트였습니다.

- 이제는 직접 서버를 구축해 볼 차례입니다.

| 무슨 서버를 구축할까?

- 클라이언트에게 날씨 정보를 제공해주는 서버를 구축합니다.
- 사실 우리는 날씨 정보가 없습니다.

(비밀) 우리는 날씨 정보 원본 데이터를 OpenWeatherMap API 를 통해 가져올 것입니다.

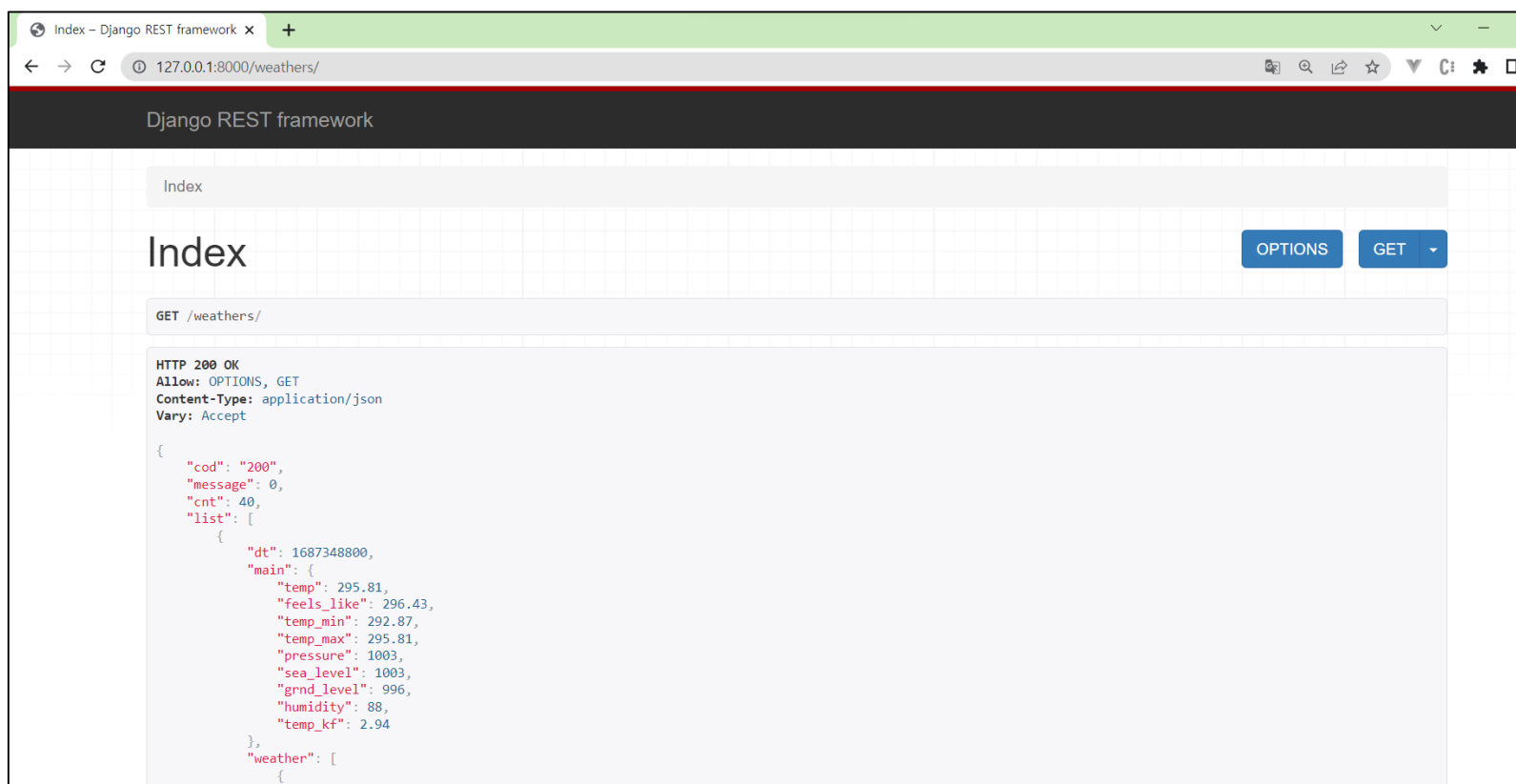
(클라이언트는 이 사실을 모를 것입니다.)



| 완성된 작품 예시

금융상품비교

영화추천서비스



[참고] 백엔드 개발 vs 프론트엔드 개발

- 백엔드 개발

- REST API 서버 개발

- 프론트엔드 개발

- REST API를 사용하여, 결과를 받아 화면 구성



이번 프로젝트 목표

Django로 백엔드를 개발하고,
차후 Vue.js를 학습하여 프론트엔드를 개발하여
하나의 완성된 웹 Application을 개발 할 예정입니다.

| 정리

- 함께 개발하는 것
 - 날씨 정보를 제공하는 REST API 서버 개발
 - OpenWeatherMap API를 활용하여 데이터를 가져옵니다.
 - DB를 구축하여 날씨 정보를 DB에 저장 후 활용합니다.

금융상품비교

영화추천서비스

날씨 데이터를 활용한 REST API Server 구축

준비사항 (API KEY 발급)

| Quiz.

- 왜 API KEY 를 발급받아야 할까요?

금융상품비교

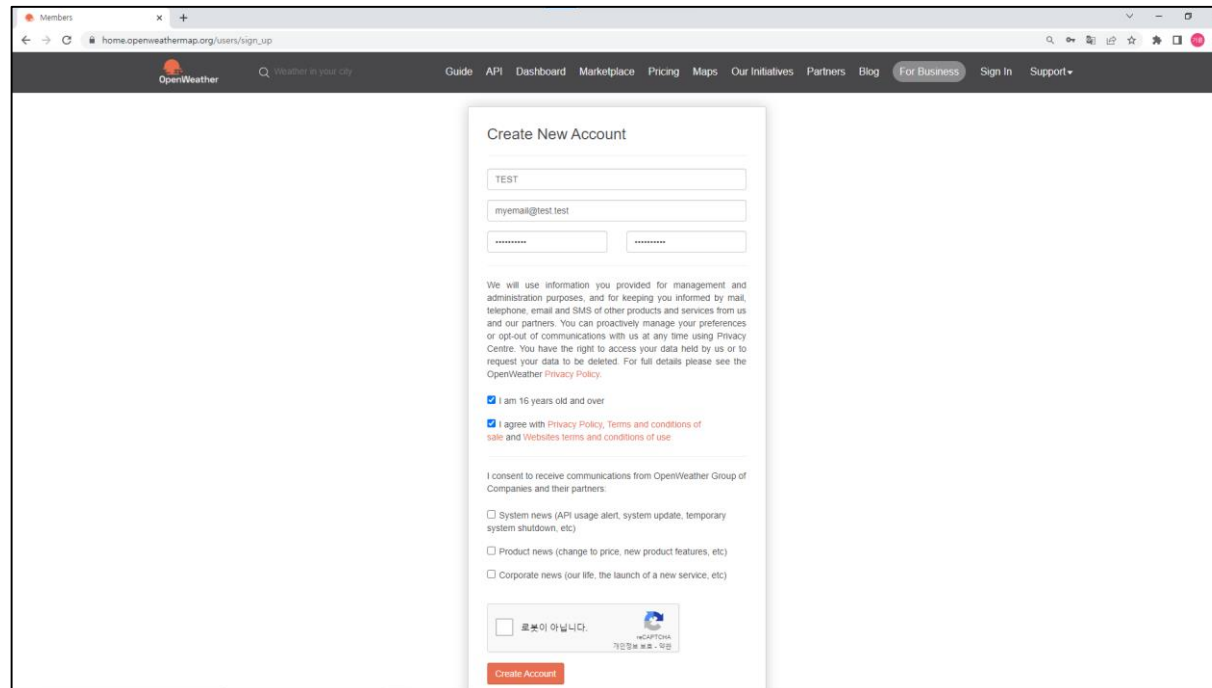
영화추천서비스

| OpenWeatherMap API - KEY 발급(1/3)

금융상품비교

영화추천서비스

- 사이트 접속 및 회원가입 진행



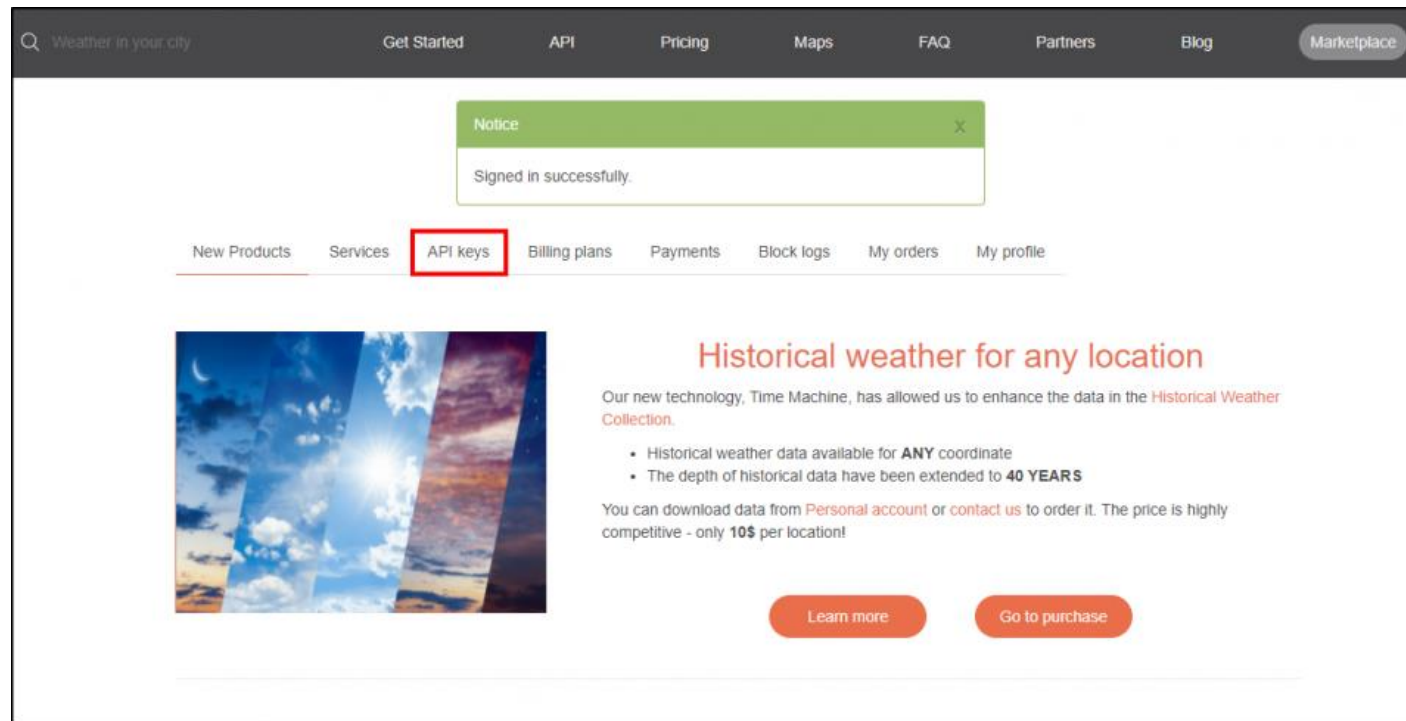
The screenshot shows the 'Create New Account' page on the OpenWeatherMap website. The page is titled 'Create New Account' and contains a registration form. The form includes fields for a username (pre-filled with 'TEST'), an email address (pre-filled with 'myemail@test.test'), and a password (pre-filled with '*****'). Below the form, there is a paragraph of text explaining the use of user information for management and administration purposes, and for keeping the user informed by mail, telephone, email and SMS of other products and services from us and our partners. It also mentions that users can proactively manage their preferences or opt-out of communications with us at any time using the Privacy Centre. Below this text, there are two checkboxes: 'I am 16 years old and over' (checked) and 'I agree with Privacy Policy, Terms and conditions of sale and Websites terms and conditions of use' (checked). Below these checkboxes, there is a statement 'I consent to receive communications from OpenWeather Group of Companies and their partners:' followed by three checkboxes: 'System news (API usage alert, system update, temporary system shutdown, etc)' (unchecked), 'Product news (change to price, new product features, etc)' (unchecked), and 'Corporate news (our life, the launch of a new service, etc)' (unchecked). At the bottom of the form, there is a checkbox labeled '로봇이 아닙니다.' (I am not a robot) with a reCAPTCHA logo. A red 'Create Account' button is located at the bottom right of the form.

OpenWeatherMap API - KEY 발급(2/3)

금융상품비교

영화추천서비스

- API Keys 탭으로 이동

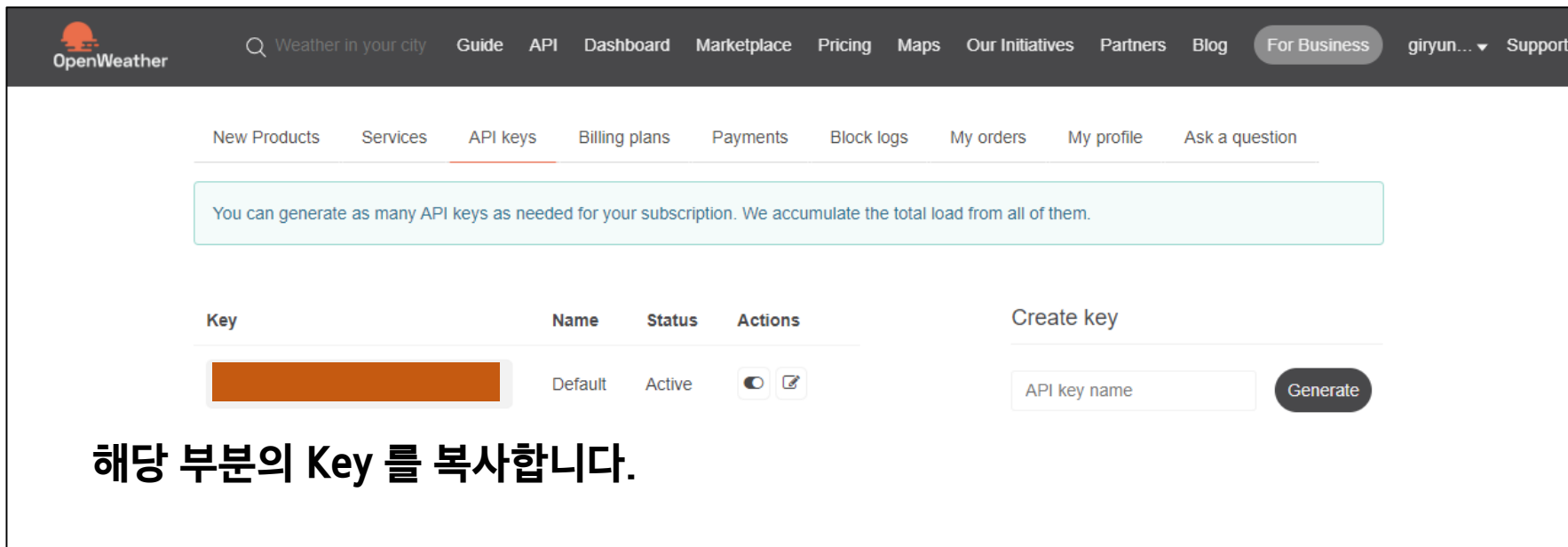


OpenWeatherMap API - KEY 발급(3/3)

금융상품비교

영화추천서비스

- API Key 복사
- API Key 를 복사하여 코드에서 활용합니다.



OpenWeatherMap API keys management page. The page shows a table with one API key, 'Default', which is active. A 'Generate' button is visible next to the key. A text box above the table states: 'You can generate as many API keys as needed for your subscription. We accumulate the total load from all of them.'

Key	Name	Status	Actions
[Redacted Key]	Default	Active	[Toggle] [Copy]

Create key

API key name

해당 부분의 Key 를 복사합니다.

구현하기

[참고사항] 프로젝트 구조

- 외부 API 를 활용하여 데이터를 다운로드 받아 DB 에 저장합니다.
- 왜 DB에 저장하나요 ?
 - DB 에 저장하여 여러 번 외부로부터 데이터를 다운로드 받지 않도록 구성할 수 있습니다.
 - 네트워크 연결이 불안정하거나 API 서비스가 일시적으로 중단된 경우에도 애플리케이션이 작동할 수 있습니다.



| 구현 목표

- 구현하고자 하는 기능 목록은 다음과 같습니다.
 - A. 서울의 5일 치 예보 데이터 확인
 - B. 예보 데이터 중 원하는 데이터만 DB에 저장
 - C. 저장된 전체 데이터 조회
 - D. 특정 조건의 데이터 확인하기: 섭씨 30도가 넘는 시간대만 조회
- API docs: <https://openweathermap.org/forecast5>

금융상품비교

영화추천서비스

5 Day / 3 Hour Forecast

API doc

Subscribe

- 5 day forecast for any location on the globe
- 5 day forecast with a 3-hour step
- JSON and XML formats
- Included in both free and paid subscriptions

A. 서울의 5일 치 예보 데이터 확인

- API 에 요청을 보내고 데이터를 확인합니다.
- DB에 저장하기 전 데이터를 확인하기 위한 과정입니다.

Built-in API request by city name

You can search weather forecast for 5 days with data every 3 hours by city name. All weather data can be obtained in JSON and XML formats.

API call

```
api.openweathermap.org/data/2.5/forecast?q={city name}&appid={API key}
```



A. 서울의 5일 치 예보 데이터 확인

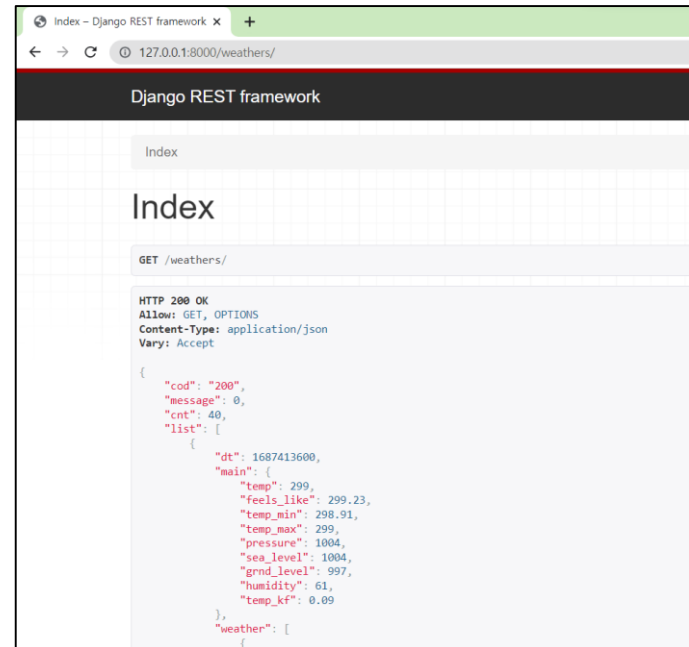
- views.py 작성 예시

```
# API 요청 후 데이터 확인
@api_view(['GET'])
def index(request):
    api_key = settings.API_KEY
    city = "Seoul,KR"

    url = f'http://api.openweathermap.org/data/2.5/forecast?q={city}&appid={api_key}'
    response = requests.get(url).json()

    return Response(response)
```

- 결과 예시



B. 예보 데이터 중 원하는 데이터만 DB에 저장

- A번에서 받은 데이터 중 “시간, 온도, 체감온도” 3가지만 사용합니다.
 - dt_txt: 시간
 - temp: 온도
 - feels_like: 체감 온도

```
{
  "list": [
    {
      "dt": 1687413600,
      "main": {
        "temp": 299,
        "feels_like": 299.23,
        "temp_min": 298.91,
        "temp_max": 299,
        "pressure": 1004,
        "sea_level": 1004,
        "grnd_level": 997,
        "humidity": 61,
        "temp_kf": 0.09
      },
      "weather": [
        {
          "id": 500,
          "main": "Rain",
          "description": "light rain",
          "icon": "10d"
        }
      ],
      "clouds": {
        "all": 68
      },
      "wind": {
        "speed": 4.6,
        "deg": 275,
        "gust": 6.11
      },
      "visibility": 10000,
      "pop": 0.53,
      "rain": {
        "3h": 0.79
      },
      "sys": {
        "pod": "d"
      }
    },
    {
      "dt_txt": "2023-06-22 06:00:00"
    }
  ]
}
```

B. 예보 데이터 중 원하는 데이터만 DB에 저장

- 3가지 필드를 DB에 저장하기 위해 다음과 같이 models.py 를 작성합니다.

```
from django.db import models

# Create your models here.
class Weather(models.Model):
    dt_txt = models.DateTimeField() # 데이터 예측 시간
    temp = models.FloatField() # 온도(기본값: 켈빈)
    feels_like = models.FloatField() # 체감온도(기본값: 켈빈)
```

- 마이그레이션 파일을 생성하고 DB에 반영합니다.

```
$ python manage.py makemigrations
```

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

B. 예보 데이터 중 원하는 데이터만 DB에 저장

- 받아온 데이터 중 3가지 데이터만 추출하여 저장해야 합니다.
- key 값이 “list” 인 데이터를 반복하며 확인합니다.
- 시간은 UTC 로 저장되어 있어,
코드와 같이 한국 시간으로 변경하여 줍니다.
- 추출한 데이터를 Serializer 에 넣어줍니다.
- 데이터 유효성 검증 후 저장합니다.

```
for li in response.get('list'):
    # dt(unix_time) 형태를 datetime 으로 변환해야 한다.
    # 한국 시간대를 나타내는 timezone 객체 생성
    dt_str = li.get('dt_txt')
    dt = datetime.strptime(dt_str, "%Y-%m-%d %H:%M:%S")

    # 기존 시간대를 UTC로 가정하고 UTC 오프셋 적용
    utc_offset = timedelta(hours=9) # 한국 시간은 UTC+9
    korea_dt = dt + utc_offset

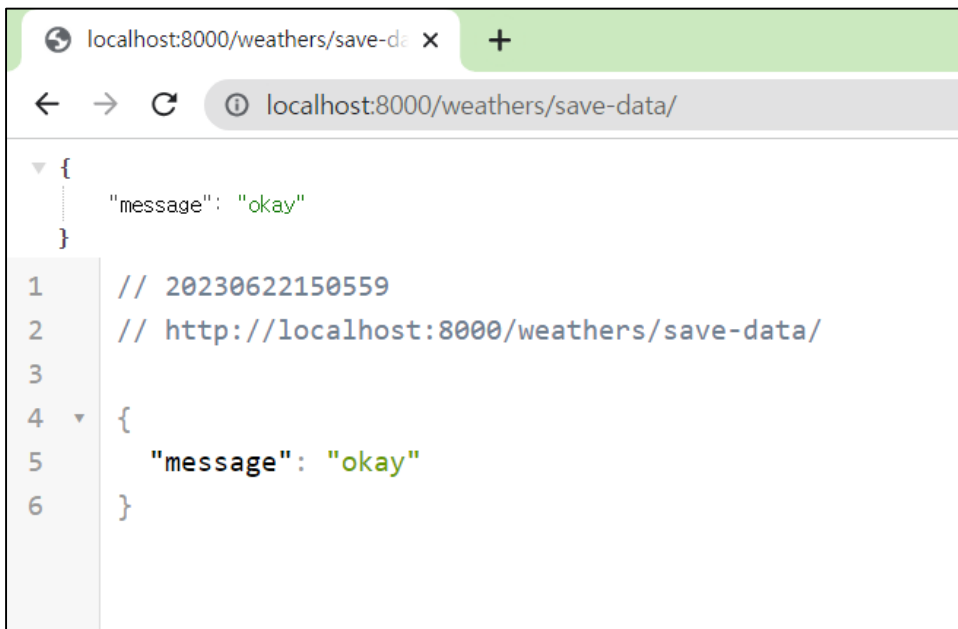
    korea_dt_str = korea_dt.strftime('%Y-%m-%d %H:%M:%S')

    # 데이터 저장을 위하여 필요한 데이터만 dictionary 형태로 만든다.
    save_data = {
        'dt_txt': korea_dt_str,
        'temp': li.get('main').get('temp'),
        'feels_like': li.get('main').get('feels_like'),
    }
    serializer = WeatherSerializer(data=save_data)
    if serializer.is_valid(raise_exception=True):
        serializer.save()
```

B. 예보 데이터 중 원하는 데이터만 DB에 저장

- 저장된 데이터는 vscode 의 sqlite extension 을 통해 확인합니다.

- 저장 후 화면

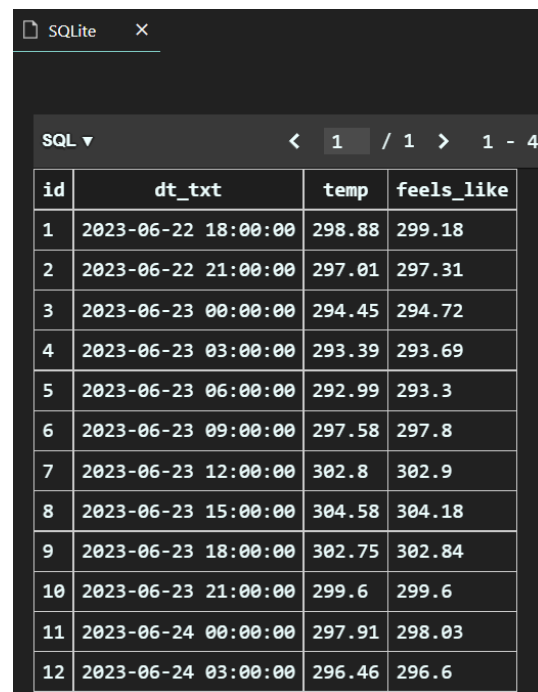


A screenshot of a web browser window. The address bar shows 'localhost:8000/weathers/save-data/'. The main content area displays a JSON response:

```
{  "message": "okay"}
```

 The browser's developer tools are open, showing the response in the console.

- 저장 결과 확인



A screenshot of the SQLite extension in VS Code. It shows a table with 12 rows of weather data. The columns are 'id', 'dt_txt', 'temp', and 'feels_like'.

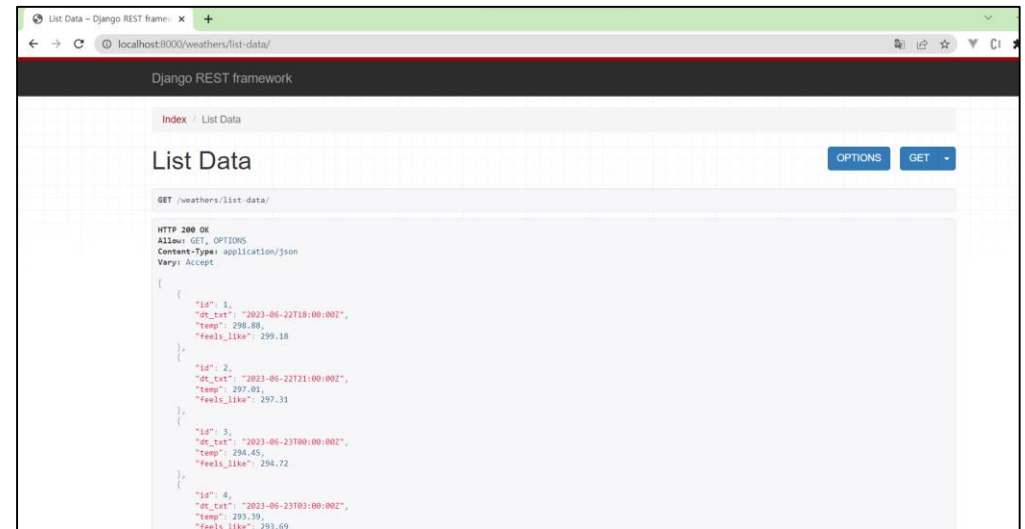
id	dt_txt	temp	feels_like
1	2023-06-22 18:00:00	298.88	299.18
2	2023-06-22 21:00:00	297.01	297.31
3	2023-06-23 00:00:00	294.45	294.72
4	2023-06-23 03:00:00	293.39	293.69
5	2023-06-23 06:00:00	292.99	293.3
6	2023-06-23 09:00:00	297.58	297.8
7	2023-06-23 12:00:00	302.8	302.9
8	2023-06-23 15:00:00	304.58	304.18
9	2023-06-23 18:00:00	302.75	302.84
10	2023-06-23 21:00:00	299.6	299.6
11	2023-06-24 00:00:00	297.91	298.03
12	2023-06-24 03:00:00	296.46	296.6

C. 저장된 전체 데이터 조회

- ORM 을 활용해 전체 데이터를 확인합니다.
- views.py 예시

```
# 전체 데이터 조회
@api_view(['GET'])
def list_data(request):
    weathers = Weather.objects.all()
    serializers = WeatherSerializer(weathers, many=True)
    return Response(serializers.data)
```

- 결과 화면 예시



D. 특정 조건의 데이터 확인하기: 섭씨 30도가 넘는 시간대만 조회

- 저장한 데이터 중 조건에 맞는 데이터만 반환해 줍니다.
- 전체 데이터를 반복하며 섭씨 30도가 넘는 데이터만 새로운 리스트로 만들어 줍니다.
- 새로운 리스트를 Serializer 를 통해 사용자에게 반환합니다.

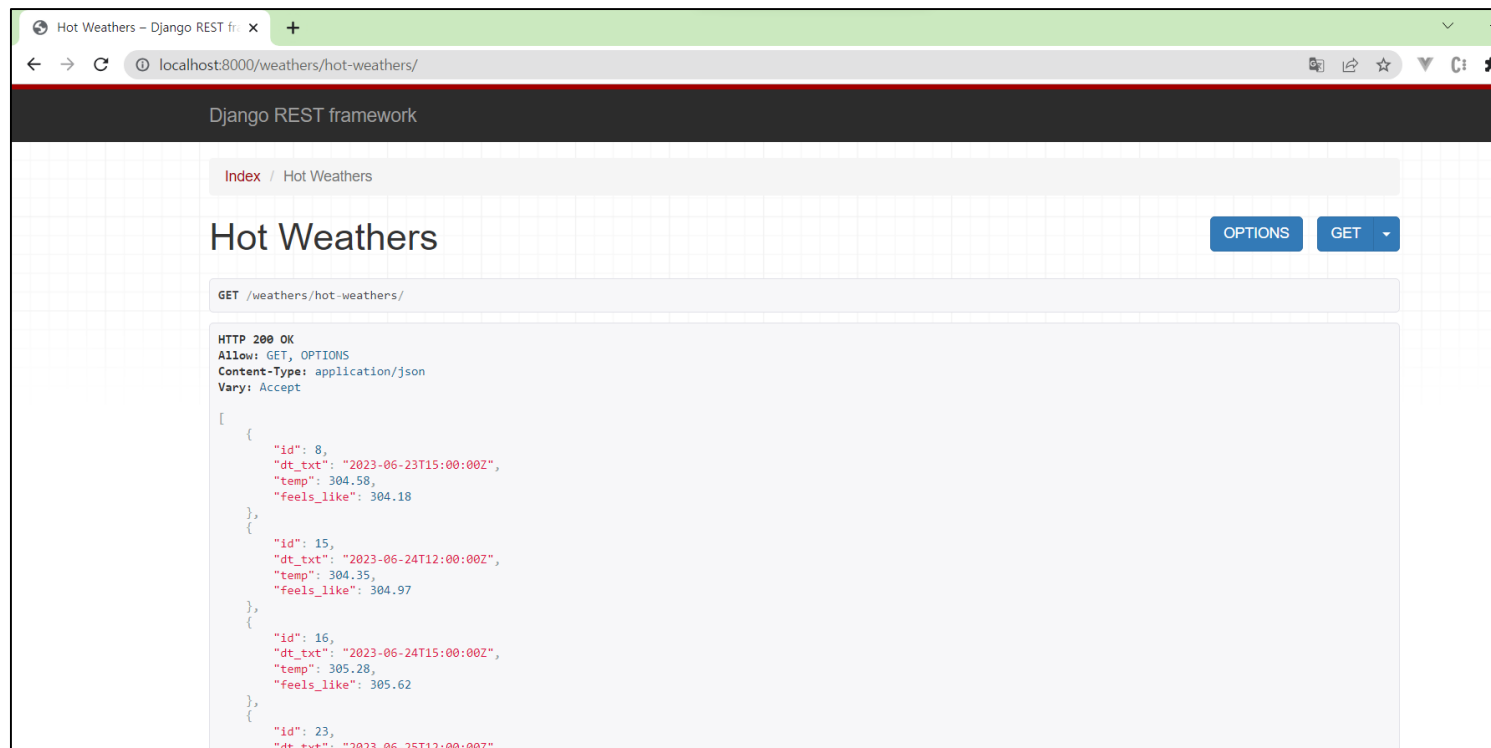
```
# 특정 조건의 데이터만 확인하기: 섭씨 30도가 넘는 시간대만 조회
@api_view(['GET'])
def hot_weathers(request):
    weathers = Weather.objects.all()
    hot_weathers = []
    for weather in weathers:
        # 섭씨 = 켈빈 - 273.15
        celsius = round(weather.temp - 273.15, 2)
        if celsius > 30:
            hot_weathers.append(weather)
    serializers = WeatherSerializer(hot_weathers, many=True)
    return Response(serializers.data)
```

D. 특정 조건의 데이터 확인하기: 섭씨 30도가 넘는 시간대만 조회

금융상품비교

영화추천서비스

- 결과 화면 예시



도전 과제

금융 상품 비교 앱 PJT 07

| 관통 Ver1 - PJT07 도전 과제

- 프로젝트명: 금융 상품 데이터를 활용한 REST API Server 구축
- 목표
 - 정기예금 데이터를 활용한 REST API Server 구축하기
- 특징
 - 금융감독원 API 를 활용한 데이터 수집

영화 추천 서비스 PJT 07

| 관통 Ver2 - PJT07 도전 과제

- 프로젝트명: DB 설계를 활용한 REST API 설계
- 목표
 - 영화 데이터를 활용한 REST API Server 구현
- 특징
 - 제공된 json 데이터를 활용하여 REST API Server 구현
 - TMDB API 를 활용하여 수집한 데이터를 제공합니다.