18장 웹 API

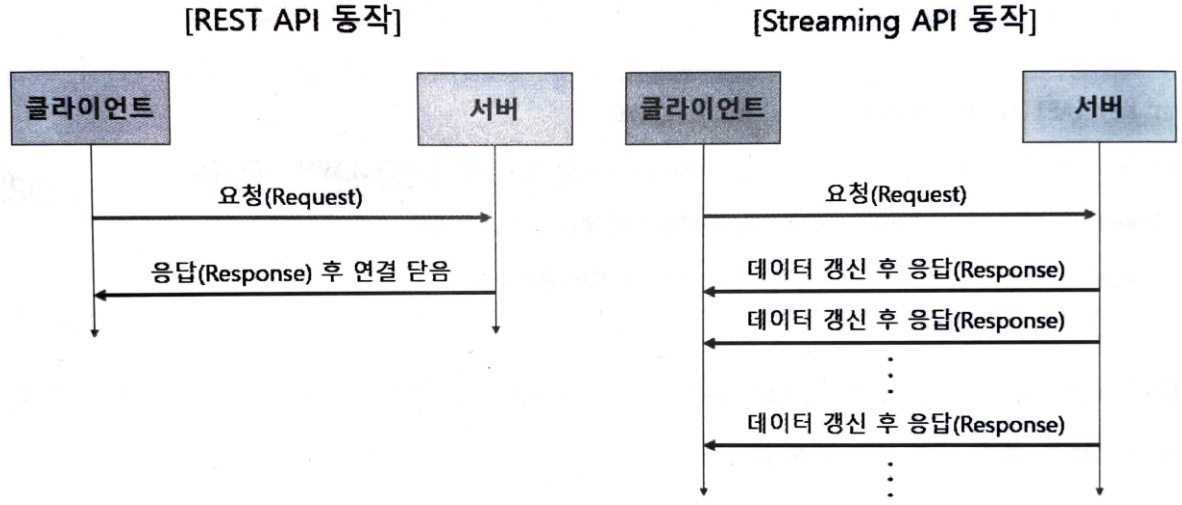
18.1 웹 API의 이해

■ 위키백과에서는 API와 웹API를 다음과 같이 설명하고 있다.
- API(Application Programming Interface): 응용 프로그램에서 사용할 수 있도록 운영체제나 프로그래밍 언어가 제공하는 기능을 제어할 수 있게 만든 인터페이스를 뜻한다. 주로 파일 제어, 창 제어, 화상 처리, 문자 제어 등을 위한 인터페이스를 제공한다.
- 웹 API: 웹 애플리케이션 개발에서 다른 서비스에 요청을 보내고 응답을 받기 위해 정의된 명세를 일컫는다.

18.1.1 웹 API의 데이터 획득 과정

■ 웹 API에는 REST API와 Streaming API가 있다.
- REST API: 이미 존재하고 있는 데이터를 공유하는데 이용되며 데이터를 요청하고 응답한 후에는 연결이 끊어진다.
- Streaming API: 향후 발생할 이벤트에 대해 등록해 놓고 그 이벤트가 발생하면 데이터를 갱신(update)한 후에 응답한다. 응답한 이후에도 강제로 연결을 끊기 전까지는 연결을 계속 유지한다.

■ REST API와 Streaming API의 데이터 요청 및 응답 과정



18.1.2 웹 API의 인증 방식

■ 인증이 필요한 웹 API의 경우, 초기에는 아이디와 비밀번호를 통해 인증하거나 웹 API별로 제각기 다른 인증 방식을 사용했으나 보안과 호환성의 문제로 OAuth가 탄생했고 최근에는 OAuth 인증 방식을 대부분 채택하고 있다.

■ OAuth는 외부에서 해당 서비스에 접속하는 모바일, 데이크톱, 웹 애플리케이션의 보안 인증을 허용하는 개방형 인증 규약이다.

■ OAuth 인증 방식에서는 아이디와 비밀번호 기반의 인증 방법 대신, API 키(Key)와 접속 토큰(Access Token), 그리고 이들의 비밀번호를 이용해 애플리케이션별로 인증을 수행하고 서비

스를 이용할 수 있는 권한을 얻는다.

18.1.3 응답 데이터의 형식 및 처리

■ 웹 API로 요청해서 응답받은 데이터의 형식은 주로 JSON과 XML이다.

(1) JSON 형식의 데이터 처리

■ JSON(JavaScript Object Notaton)은 자바스크립트 언어와 함께 사용될 목적으로 만들었지만 대부분의 프로그래밍 언어에서 이용할 수 있다.

■ JSON에서 하나의 데이터 집합을 객체(Object)라고 하는데 기본적인 객체의 구성은 '이름(name): 값(value)'으로 이뤄진 쌍의 집합이다.

■ json 라이브러리 활용해 파이썬의 딕셔너리 데이터를 JSON 형태로 변환하고 JSON 형태 데이터를 파이썬의 딕셔너리 타입으로 변환해 보자.
- 21~22: json.dumps()를 이용해 딕셔너리 데이터를 JSON 형태의 데이터(str)로 변환한다.
- 31~32: json.dumps()에 옵션을 추가해서 변환한 후에 출력된 결과를 보면, JSON 형식의 데이터가 들여쓰기되어 보기 편하게 출력되고 한글도 잘 출력된다.
- 50~51: json.loads()를 이용해 JSON 형식의 데이터를 파이썬의 딕셔너리 데이터로 변환한다.

[ch18_api/ex01_json.py]

```
01  import json
02
03  python_dict = {
04      "이름": "홍길동",
05      "나이": 25,
06      "거주지": "서울",
07      "신체정보": {
08          "키": 175.4,
09          "몸무게": 71.2
10      },
11      "취미": [
12          "등산",
13          "자전거타기",
14          "독서"
15      ]
16  }
17
18  print(type(python_dict))
19  # <class 'dict'>
20
21  json_data = json.dumps(python_dict)
22  print(type(json_data))
23  # <class 'str'>
24  print(json_data)
25  '''
26  {"\uc774\ub984":  "\ud64d\uae38\ub3d9",  "\ub098\uc774":  25,  "\uac70\uc8fc\uc9c0":  "\uc11c\uc6b8",
27  "\uc2e0\uccb4\uc815\ubcf4":   {"\ud0a4":   175.4,   "\ubab8\ubb34\uac8c":   71.2},   "\ucde8\ubbf8":
28  ["\ub4f1\uc0b0", "\uc790\uc804\uac70\ud0c0\uae30", "\ub3c5\uc11c"]}
29  '''
30
31  json_data = json.dumps(python_dict, indent=3, sort_keys=True, ensure_ascii=False)
32  print(json_data)
```

```
'''
{
   "거주지": "서울",
   "나이": 25,
   "신체정보": {
      "몸무게": 71.2,
      "키": 175.4
   },
   "이름": "홍길동",
   "취미": [
      "등산",
      "자전거타기",
      "독서"
   ]
}
'''

json_dict = json.loads(json_data)
print(type(json_dict))
# <class 'dict'>

print(json_dict['신체정보']['몸무게'])
# 71.2

print(json_dict['취미'])
# ['등산', '자전거타기', '독서']

print(json_dict['취미'][0])
# 등산
```

(2) XML 형식의 데이터 처리

■ 다음의 코드는 XML 형식의 데이터를 파이썬의 딕셔너리 타입의 데이터로 변환해 원하는 데이터를 추출하는 과정을 설명하고 있다.

- 19~26: 'xml_attribs=True'를 사용했지만 이 옵션을 쓰지 않아도 기본값이다. 출력 결과에서 '키'와 '몸무게'의 속성은 '@속성이름'으로, 속성을 갖는 태그의 문자열은 '#text'로 표시됨을 알 수 있다.
- 64~70: 'xml_attribs=False'로 설정하면 XML 형식의 데이터에서 지정했던 속성이 모두 무시되면서 딕셔너리 형식으로 변환된다.

[ch18_api/ex02_xml.py]

```
import xmltodict
xml_data = """<?xml version="1.0" encoding="UTF-8" ?>
<사용자정보>
    <이름>홍길동</이름>
    <나이>25</나이>
    <거주지>서울</거주지>
    <신체정보>
        <키 unit="cm">175.4</키>
        <몸무게 unit="kg">71.2</몸무게>
    </신체정보>
    <취미>등산</취미>
    <취미>자전거타기</취미>
    <취미>독서</취미>
</사용자정보>
"""
print(xml_data)
```

```
17
18
19  dict_data = xmltodict.parse(xml_data, xml_attribs=True)
20  print(dict_data)
21  '''
22  OrderedDict([('사용자정보', OrderedDict([('이름', '홍길동'), ('나이', '25'), ('거주지',
23  '서울'), ('신체정보', OrderedDict([('키', OrderedDict([('@unit', 'cm'), ('#text', '175.4')])), ('몸무게
24  ', OrderedDict([('@unit', 'kg'), ('#text', '71.2')]))])), ('취미', ['등산
25  ', '자전거타기', '독서'])]))])
26  '''
27
28  print(dict_data['사용자정보']['이름'])
29  # 홍길동
30
31  print(dict_data['사용자정보']['신체정보'])
32  '''
33  OrderedDict([('키',     OrderedDict([('@unit',     'cm'),     ('#text',     '175.4')])),     ('몸무게',
34  OrderedDict([('@unit', 'kg'), ('#text', '71.2')]))])
35  '''
36
37  print(dict_data['사용자정보']['신체정보']['키']['@unit'])
38  # cm
39
40  print(dict_data['사용자정보']['신체정보']['키']['#text'])
41  # 175.4
42
43
44  dict_data = xmltodict.parse(xml_data)
45
46  user_name = dict_data['사용자정보']['이름']
47  body_data = dict_data['사용자정보']['신체정보']
48
49  height = body_data['키']['#text']
50  height_unit = body_data['키']['@unit']
51
52  weight = body_data['몸무게']['#text']
53  weight_unit = body_data['몸무게']['@unit']
54
55  print("[사용자 {0}의 신체정보]".format(user_name))
56  print("*키: {0}{1}".format(height, height_unit))
57  print("*몸무게: {0}{1}".format(weight, weight_unit))
58  '''
59  [사용자 홍길동의 신체정보]
60  *키: 175.4cm
61  *몸무게: 71.2kg
62  '''
63
64  dict_data2 = xmltodict.parse(xml_data, xml_attribs=False)
65  print(dict_data2)
66  '''
67  OrderedDict([('사용자정보', OrderedDict([('이름', '홍길동'), ('나이', '25'), ('거주지',
68  '서울'), ('신체정보', OrderedDict([('키', '175.4'), ('몸무게', '71.2')])), ('취미', ['등
69  산', '자전거타기', '독서'])]))])
70  '''
```

18.1.4 웹 사이트 주소에 부가 정보 추가하기

(1) 웹 사이트 주소에 경로 추가하기

■ 다음 코드는 기본 웹 사이트 주소는 고정하고 하위 경로만 변경해 URL을 생성하고 requests

라이브러리의 'requests.get()'를 이용해 데이터를 요청하고 응답받는 예이다.

[ch18_api/ex03_url.py]

```
01  base_url = "https://api.github.com/"
02  sub_dir = "events"
03  url = base_url + sub_dir
04  print(url)      # https://api.github.com/events
05
06
07  import requests
08
09  base_url = "https://api.github.com/"
10  sub_dirs = ["events", "user", "emails" ]
11
12  for sub_dir in sub_dirs:
13      url_dir = base_url + sub_dir
14      r = requests.get(url_dir)
15      print(r.url)
16  '''
17  https://api.github.com/events
18  https://api.github.com/user
19  https://api.github.com/emails
20  '''
```

(2) 웹 사이트 주소에 매개변수 추가하기

■ 다음 코드는 날씨 데이터를 제공하는 웹 API에 여러 매개변수(API 키, 위도, 경도, 단위 키와 값)를 전달하기 위해 직접 URL을 생성한 예이다.

- 29: 직접 생성한 URL을 요청 주소(url)와 요청 매개변수(params)로 분리한 후 'requests.get(url, params=req_parameter)'을 이용해 URL을 생성한다.

[ch18_api/ex04_param.py]

```
01  import requests
02
03  LAT = '37.57' # 위도
04  LON = '126.98' # 경도
05  API_KEY = 'b235c57pc357fb68acr1e81' # API 키(임의의 API 키)
06  UNIT = 'metric' # 단위
07
08  site_url = "http://api.openweathermap.org/data/2.5/weather"
09  parameter = "?lat=%s&lon=%s&appid=%s&units=%s"%(LAT, LON, API_KEY, UNIT)
10  url_para = site_url + parameter
11  r = requests.get(url_para)
12
13  print(r.url)
14  '''
15  http://api.openweathermap.org/data/2.5/weather?lat=37.57&lon=126.98&appid=b235c57pc357fb68acr1e81&units
16  =metric
17  '''
18
19
20  import requests
21
22  LAT = '37.57' # 위도
23  LON = '126.98' # 경도
24  API_KEY = 'b235c57pc357fb68acr1e81' # API 키(임의의 API 키)
25  UNIT = 'metric' # 단위
26
```

```
27  req_url = "http://api.openweathermap.org/data/2.5/weather"
28  req_parameter = {"lat":LAT, "lon":LON , "appid": API_KEY, "units":UNIT}
29  r = requests.get(req_url,params=req_parameter)
30  print(r.url)
31  '''
32  http://api.openweathermap.org/data/2.5/weather?lat=37.57&lon=126.98&appid=b235c57pc357fb68acr1e81&units
33  =metric
34  '''
```

(3) 웹 사이트 주소의 인코딩과 디코딩

[ch18_api/ex05_encoding.py]

```
01  import requests
02
03  API_KEY                                                                                                    =
04  "et5piq3pfpqLEWPpCbvtSQ%2Bertertg%2Bx3evdvbaRBvhWEerg3efac2r3f3RfhDTERTw%2B9rkvoewRV%2Fovmrk3dq%3D%3D"
05
06  API_KEY_decode = requests.utils.unquote(API_KEY)
07
08  print("Encoded url:", API_KEY)
09  print("Decoded url:", API_KEY_decode)
10  '''
11  Encoded                                                                                                  url:
12  et5piq3pfpqLEWPpCbvtSQ%2Bertertg%2Bx3evdvbaRBvhWEerg3efac2r3f3RfhDTERTw%2B9rkvoewRV%2Fovmrk3dq%3D%3D
13  Decoded url: et5piq3pfpqLEWPpCbvtSQ+ertertg+x3evdvbaRBvhWEerg3efac2r3f3RfhDTERTw+9rkvoewRV/ovmrk3dq==
14  '''
```

18.2 API 키를 사용하지 않고 데이터 가져오기

18.2.1 국제 우주 정거장의 정보 가져오기

- Open Notify(http://open-notify.org/)는 국제 우주 정거장의 현재 위치(위도, 경도), 지정된 위치를 지나가는 시간 및 국제 우주 정거장에 있는 우주인의 수와 이름을 알려주는 웹 API를 제공한다.

[ch18_api/ex06_open-notify.py]

```
01  import requests
02  import json
03
04  url = "http://api.open-notify.org/iss-now.json"
05
06  r = requests.get(url)
07  print(r.text)
08  '''
09  {"message": "success", "iss_position": {"longitude": "-155.5470", "latitude": "13.6561"}, "timestamp":
10  1637328033}
11  '''
12
13
14  import requests
15  import time
16
17  url = "http://api.open-notify.org/iss-now.json"
```

```
18
19  def ISS_Position(iss_position_api_url):
20      json_to_dict = requests.get(iss_position_api_url).json()
21      return json_to_dict["iss_position"]
22
23  for k in range(5):
24      print(ISS_Position(url))
25      time.sleep(10)  # 10초 동안 코드 실행을 일시적으로 중지한다.
26  '''
27  {'longitude': '-152.1241', 'latitude': '9.0961'}
28  {'longitude': '-151.7364', 'latitude': '8.5671'}
29  {'longitude': '-151.3498', 'latitude': '8.0377'}
30  '''
```

~~18.2.2 국가 정보 가져오기~~

~~18.3 트위터에 메시지 작성하고 가져오기~~

18.4 정부의 공공 데이터 가져오기

18.4.1 회원 가입 및 서비스 신청

■ 공공 데이터 포털의 웹 API를 이용하려면 공공 데이터 포털(https://www.data.go.kr)에 접속하여 일반회원으로 가입한다.

18.4.2 주소 및 우편번호 가져오기

■ 공공 데이터 포털에서 제공하는 웹 API의 경우 각 서비스마다 API를 이용하기 위한 참고 문서가 있다. (새주소5자리우편번호조회서비스명세서.docx)

■ 호출 URL
- http://openapi.epost.go.kr/postal/retrieveNewAdressAreaCdService/retrieveNewAdressAreaCdService/getNewAddressListAreaCd?ServiceKey=인증키&searchSe=road&srchwrd=세종로 17

[ch18_api/ex08_address.py]

```
01  import xmltodict
02  import requests
03
04  # 자신의 인증키를 복사해서 입력합니다.
05  API_KEY                                                                                                    =
06  'Nthm35khfZR6DRDqGytWeSGTeOWEmKBf293ItGFcZHOP%2BDHWzPGpb4bAnsLTfCSSfhwbCyhaaMFm%2FR7uuhIyRw%3D%3D'
07  API_KEY_decode = requests.utils.unquote(API_KEY)
08  print(API_KEY_decode)
09
10
11  req_url                                                                                                    =
12  "http://openapi.epost.go.kr/postal/retrieveNewAdressAreaCdService/retrieveNewAdressAreaCdService/getNew
13  AddressListAreaCd"
14
15  search_Se = "road"
16  srch_wrd = "반포대로 201"
17
```

```
req_parameter = {"ServiceKey": API_KEY_decode,
                 "searchSe": search_Se, "srchwrd": srch_wrd}

r = requests.get(req_url, params=req_parameter)
xml_data = r.text
print(xml_data)


dict_data = xmltodict.parse(xml_data)
print(dict_data)


adress_list = dict_data['NewAddressListResponse']['newAddressListAreaCd']

print("[입력한 도로명 주소]", srch_wrd)
print("[응답 데이터에서 추출한 결과]")
print("- 우편번호:", adress_list['zipNo'])
print("- 도로명 주소:", adress_list['lnmAdres'])
print("- 지번 주소:", adress_list['rnAdres'])
```

~~18.4.3 날씨 정보 가져오기~~
~~18.4.4 대기 오염 정보 가져오기~~