기초 개발 지식 공부하기
Python 라이브러리를 활용한 API 호출(서버와의 통신) 방법

API 호출 | 클라이언트에서 API 호출을 통해 서버와 통신하기

강사 나동빈

기초 개발 지식 공부하기

Python 라이브러리를 활용한 API 호출(서버와의 통신) 방법

REST (Representational State Transfer) 등장 배경

- HTTP는 다양한 HTTP 메서드(GET, POST, PUT, DELETE 등)를 지원한다.
- 실제로는 서버가 HTTP 메서드를 기존 설명에 맞게 사용하지 않더라도, 프로그램 개발은 가능하다.
- 하지만 각 서비스가 서로 다른 방식으로 개발하면, 개발자 사이의 소통에 문제가 발생할 수 있다.
- 따라서 기준이 되는 아키텍처로 REST를 채택할 수 있다.

REST 이해하기

- REST는 Representational State Transfer의 약자이다.
- 말 그대로 특정한 자원(resource)에 대하여, 자원의 상태에 대한 정보를 주고받는 개발 방식이다.
- REST의 구성 요소는 다음과 같다.

자원(resource)	URI를 이용
행위(verb)	HTTP 메서드를 이용
표현(representation)	페이로드(payload)를 이용

REST 예제 살펴보기

- REST 방식을 채택한 서버로 요청(request)을 보내는 예시는 다음과 같다.
- 클라이언트가 회원가입을 하고 싶은 상태다.
- 이때, 아이디는 "gildong", 비밀번호는 "1234"로 설정하고 싶다면?

자원	회원(user)
행위	회원 등록
표현	아이디: "gildong", 비밀번호: "1234"

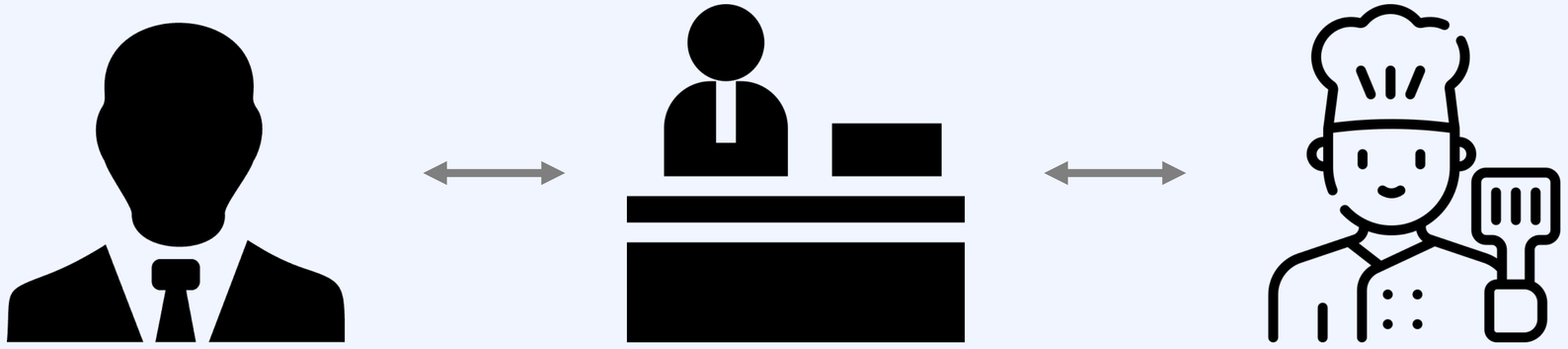
REST 예제 살펴보기

- 클라이언트가 회원가입을 하고 싶은 상태다.
- 이때, 아이디는 "gildong", 비밀번호는 "1234"로 설정하고 싶다면?
- 동일한 내용을 HTTP 패킷으로 표현하면 다음과 같다.

```
URI: https://www.example.com/users
HTTP Method: POST
Payload: {"id" : "gildong", "password" : "1234"}
```

REST API

- API (Application Programming Interface): 프로그램이 상호작용하기 위한 인터페이스
- REST API: REST 아키텍처를 따르는 API
- REST API 호출: REST 방식을 따르고 있는 서버에 특정한 요청(request)을 전송하는 행위



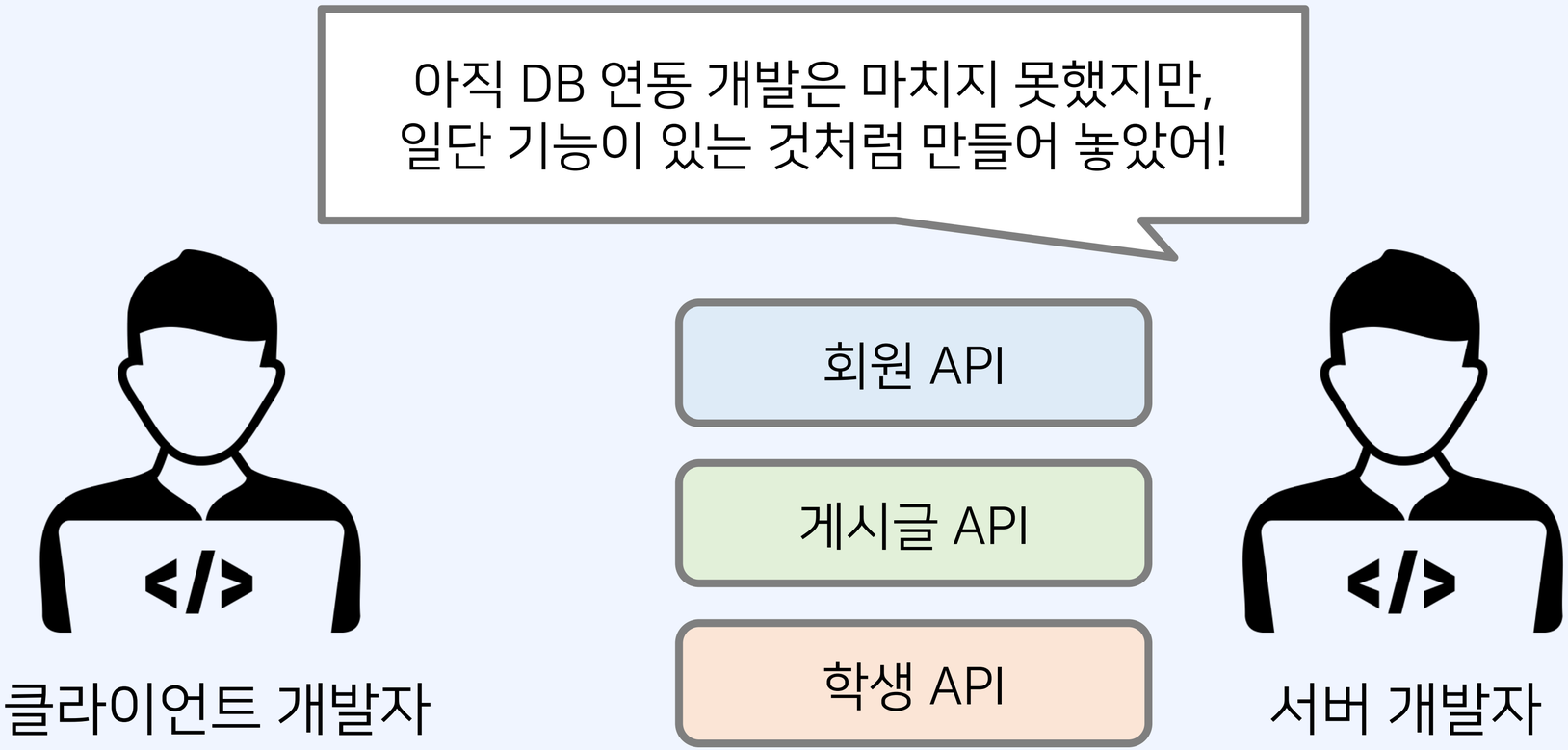
API 사용자 | API (점원) | API 제공자 (주방장)

REST API 연습하기

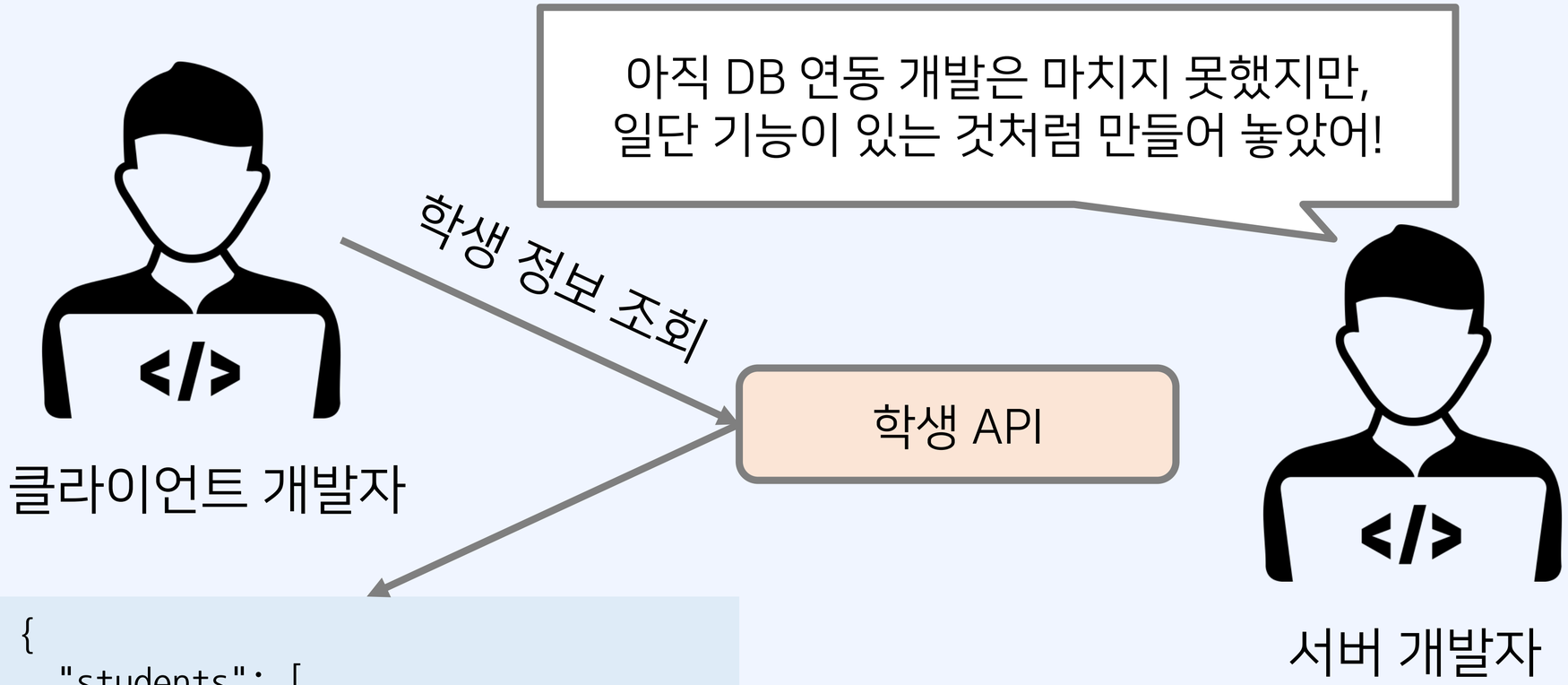
- **목킹(mocking)**: 어떠한 기능이 있는 것처럼 흉내내어 구현한 것을 의미한다.
- 클라이언트 개발을 위해 간단히 서버 기능을 테스트할 때 사용한다.
- 처음부터 모든 서버 기능을 개발하고, 클라이언트 개발을 시작하면 개발 일정에 지연이 생길 수 있다.

REST API 연습하기

- **목킹(mocking)**: 어떠한 기능이 있는 것처럼 흉내내어 구현한 것을 의미한다.



REST API 연습하기




```
{
  "students": [
    {"name": "홍길동", "age": 20},
    {"name": "김삿갓", "age": 25},
    {"name": "나동빈", "age": 30}
  ]
}
```

REST API 연습하기

- REST API 목킹 서비스 예시: https://jsonplaceholder.typicode.com/
- 사용자(user) 정보 API 확인해 보기

1) 전체 사용자 목록: https://jsonplaceholder.typicode.com/users
2) 특정 사용자: https://jsonplaceholder.typicode.com/users/1

REST API 호출 실습 - 사용자 정보 조회: user_api.py


```
import requests

# REST API 경로에 접속하여 응답(Response) 데이터 받아오기
target = "https://jsonplaceholder.typicode.com/users"
response = requests.get(url=target)

# 응답(Response) 데이터가 JSON 형식이므로 바로 파이썬 객체로 변환
data = response.json()

# 모든 사용자 정보를 확인하여 이름 정보만 삽입
name_list = []
for user in data:
    name_list.append(user['name'])

print(name_list)

# 모든 사용자(user) 정보를 확인하며 사용자 별 휴대폰 번호 정보 저장
phone_list = []
for user in data:
  phone_list.append((user['name'], user['phone']))

print(phone_list)
```

Flask란?

- 파이썬 기반의 웹 애플리케이션 개발을 위한 **프레임워크**(framework)다.
- 프레임워크의 사용 난이도가 낮아서 입문용으로 좋다.
- 가벼운 기능 개발 목적으로 간단히 사용해 볼 수 있다.

Flask로 기본적인 웹 사이트 만들기: basic_server.py

- **Flask 웹 서버 프로그램**은 기본적으로 5000 포트를 사용한다.
- route() 데코레이터를 통해 (클라이언트가 특정한 URL을 방문할 때) 작성된 함수가 트리거된다.
- 아래 예시에서는 localhost:5000/home, localhost:5000/board에 접속할 수 있다.

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
@app.route('/home')
def home():
    return 'Hello, World!'

@app.route('/board')
def board():
    return 'This is a board page!'

if __name__ == '__main__':
    app.run()
```

Flask로 간단한 회원 API 만들기: user_server.py

- 간단히 회원(user) 정보를 처리하는 API를 만들 수 있다.
- 현재 예시에서는 REST API를 따르지 않는다.
- 실제 REST API에서는 HTTP 메서드로 회원 조회/등록/수정/삭제 기능을 구분한다.

쿠키(Cookie)와 세션(Session)

- **쿠키**: 사용자가 특정한 웹 사이트에 방문할 때, 사용자 컴퓨터에 저장하는 기록 파일이다.
- 서버의 자원을 전혀 사용하지 않는다.
- **사용 예시**: "아이디와 비밀번호를 저장하시겠습니까?"

쿠키(Cookie)와 세션(Session)

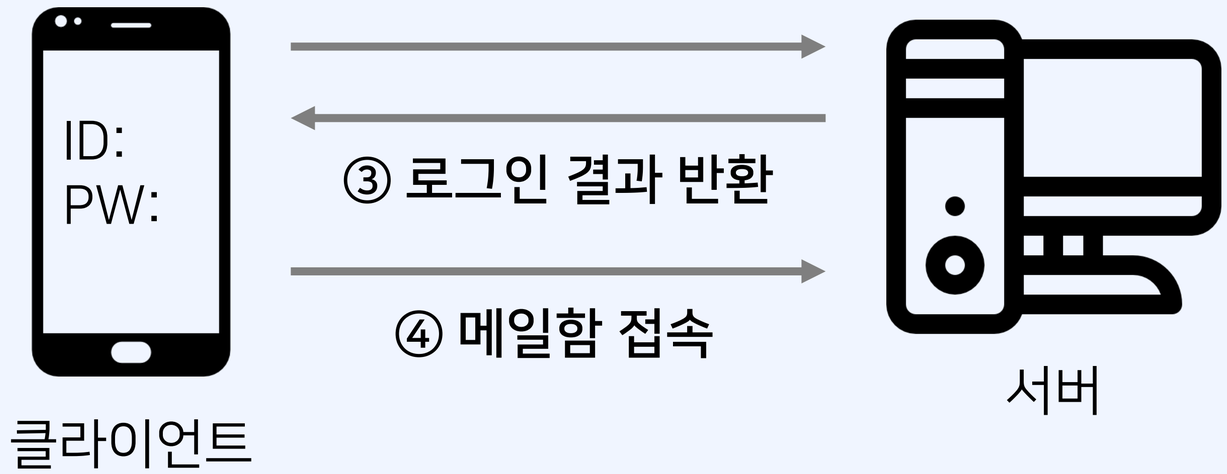
- **세션**: 한 명의 사용자(브라우저)의 상태를 유지하는 기술이다.
- 서버가 클라이언트에게 고유한 Session ID를 부여하면, 클라이언트는 접속할 때마다 Session ID와 함께 요청한다.
- **사용 예시**: 웹 사이트에 한 번 로그인 하면, 다른 페이지로 이동해도 계속 접속 상태가 유지된다.
- 만약 Session ID를 다른 클라이언트에게 탈취당하면, 다른 사람이 자신의 행세를 할 수 있다.

세션(Session) 개요

- 서버에서 가지고 있는 객체로, 특정 사용자의 로그인 정보를 유지하기 위해 사용할 수 있다.
- 예를 들어 우리가 웹 사이트에 로그인한 뒤에, 서버에서는 **세션 ID에 따른 회원 ID 정보를 기록**한다.
- 클라이언트는 해당 세션을 계속 유지한다. 예를 들어 메일함에 접속할 때도 세션 ID를 서버에 전송한다.
- 다시 말해 세션은 자신이 누구인지를 서버에 알려주는 역할을 수행한다.

세션(Session) 개요

- 서버에서 가지고 있는 객체로, 특정 사용자의 로그인 정보를 유지하기 위해 사용할 수 있다.



② 세션 정보 기록

Session ID	정보
“JKLAMFJDIS”	ID: “minjeong”, …
“KAMZXIDUSA”	ID: “gildong”, …
…	…

Flask에서 세션 사용하기: session_server.py

- Flask에서는 세션(session)을 사용하기 위해 비밀 키(secret_key)를 설정해야 한다.
- **세션 저장**: session[**{변수 이름}**] = {값}
- **세션 삭제**: session.pop(**{변수 이름}**, None)
- 장바구니 기능을 세션을 이용해 구현한 예시를 확인해 보자.