**초기 데이터를 구축하는 3가지 방법**

1 파일형태 (CSV) 임포트

2 open api 사용 //자체 디비에 저장할 때 저작권 문제가 발생하는지 고민해야됨

3 크롤링해서 우리 디비에 넣기. //1번 2번 둘다 지원 안될 때.

**데이터를 가져오는 파트**

디비와 연관 없는 것은 프론트의 상태관리 툴(axios)에서 가져옴.

디비와 연관된 것은 백엔드의 spring에서 가져옴.

**OPEN API 사용방법. //제공하는 측에 따라 규칙이나 형식이 달라질 수 있음.**

보통 제공하는 사이트에서 api를 사용할 때 필요한 ‘키’를 발급한다.

Api 호출은 url로 요구하는 형식에 맞춰서 요청하면된다.

OpenWeather이 요구하는 api 호출 형식이다.  
http://api.openweathermap.org/data/2.5/weather?q={city}&APPID={API\_KEY}&lang=kr

다른 옵션들은 사이트 명시되어있다. 쿼리스트링으로 속성들을 지정하는 경우가 일반적이다.

보통 OpenApi의 요청 결과로 JSON형식이 응답으로 많이 온다.

백엔드에서 응답을 받을 경우(DB와 연관된 것이라는 거겠쬬?)  
응답온 JSON을 java 객체 형태(POJO)인 DTO에 담아서 사용하면 된다.

프론트에서 응답을 받을 경우엔(DB와 연관없는 것이라는 거겠쬬?) (아마 axios로 요구하려나)  
응답온 JSON을 js 객체에 담아서 사용하면 된다.

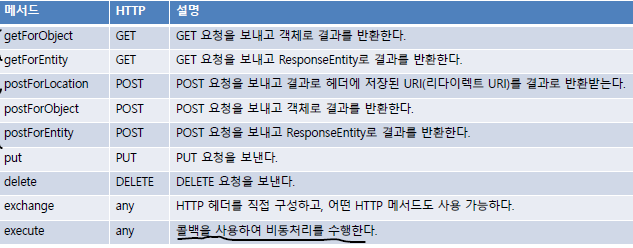
**RestTemplate**

백엔드(Spring)에서 외부의 api를 통해(요청을 보내) 데이터를 얻기 위해 사용.  
객체로 rest방식 api를 호출할 수 있는 spring내장 클래스

서버 측에서 내가 만든 rest기반 api를 ‘제공’하는 역할 🡺 RestController.   
예시) @RequestMapping(“/urlExample”)  
내 애플리케이션이 api를 제공한다면 RestController 필요.

내 쪽에서 외부에서 제공하는 api를 ‘호출’(‘요청’)하는 역할 🡺 RestTemplate.  
예시) restTemplate.getForObjects( … )  
내 애플리케이션이 외부 api를 호출한다면 RestTemplate을 이용.

구성 및 준비 :   
implementation 'org.apache.httpcomponents:httpcore:4.4.15'  
implementation 'org.apache.httpcomponents:httpclient:4.5.13'

메서드들:  


Get요청 :   
??DTO변수=restTemplate.getForObject( url , ???DTO.class ); //JSON🡺java객체

Post요청 :   
응답 바디만 필요할 때  
LoginRequest request = new LoginRequest("hong", "1234");  
LoginResponse response = restTemplate.postForObject(url, request, LoginResponse.class);

응답 헤더, 상태코드, 바디 등 전체가 필요할 때   
LoginRequest request = new LoginRequest("hong", "1234");  
ResponseEntity<LoginResponse> responseEntity =  
restTemplate.postForEntity(url, request, LoginResponse.class);

**URL을 쉽게 구성해주는 UriComponentsBuilder 활용.**  
String URL = UriComponetsBuilder.fromHttpUrl( “기본 url” )  
 .queryParam(“속성명”, “속성값”)  
 .queryParam(“속성명2”, “속성값2”)  
 …  
 .toUriString( );

**RestTemplate에 대한 추가 정보**

Spring3부터 지원하고 Spring5부터 지원 안함. Why? 동기 방식으로 작동해서  
비동기/논블로킹이 필요한 시대에는 맞지 않아서.

WebClient객체가 이를 대신하는 객체인데 이는 RestTemplate + 비동기 + 논블로킹 버전이다.

WebClient객체는 spring webflux의 일부이지만 그 부분만 의존성을 추가해서 활성화하면 WebClient객체를 Spring MVC에서 사용할 수 있다.

하지만 기본적으로 WebClient는 비동기/논블로킹 으로 동작하지만   
Spring MVC에서는 동기적으로 동작하게 해야한다. (.block메서드 사용해서)