초기 데이터를 구축하는 방법

1 파일 형태로(CSV)
2 open api사용

자체 디비에 저장할것인가를 고민해봐야해.
저작권문제가 밟힘.

3open api를 제공안하는 데이터는 크롤링 프로그램 짜서
긁어서 우리DB에 넣는.

날씨정보를 얻는 open api를 활용해보자

공공데이터포탈. data.go.kr
엑셀 및 api 제공함.

OpenApi - OpenWeather

프론트 axios에서 할거냐
백엔드 spring에서 하거냐

디비 연관된 것은 백엔드에서
아닌것은 프론트에서
가져옴.

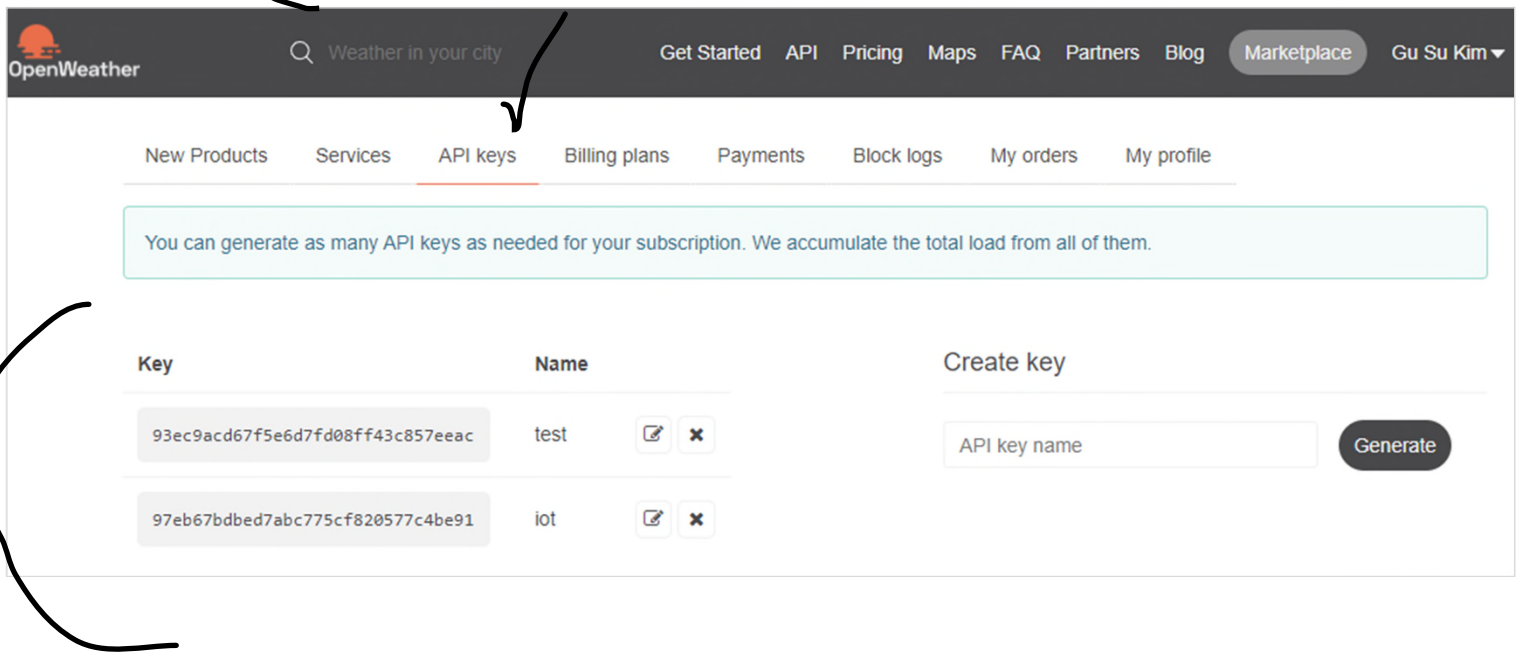
사실 해당 예시는 프론트가 더 적합하긴함

OpenApi - OpenWeather

- **openweather**
  - https://openweathermap.org/
  - 회원가입 -> 로그인
  - API키 발급
    - 회원 가입시 자동 발급

키??

남용되는것을 방지하기 위해 access key 바급



발급받은 access key

OpenApi - OpenWeather

- **api 호출** 제공하는 기본 url이 있어.
  - http://api.openweathermap.org/data/2.5/weather?q={city}&APPID={API_KEY}&lang=kr

    쿼리스트링을 구성해야함

    http://api.openweathermap.org/data/2.5/weather?q=seoul&APPID=xxxxxxxxx&lang=kr

  - 온도는 캘빈 온도값
    - 섭씨 변환
      - 캘빈 - 272
  - 섭씨 요청하기

    http://api.openweathermap.org/data/2.5/weather?q=seoul&units=metric&APPID=xxxxxxxxx&lang=kr
  - 위도,경도로 요청하기

    http://api.openweathermap.org/data/2.5/weather?lat=yyy&lon=xxx&units=metric&APPID=xxxxxxxxx&lang=kr

OpenApi - OpenWeather

결과 json

앞 url요청 결과

날씨아이콘에 대한 id


```
{
 "coord": { "lon": 126.9778, "lat": 37.5683 },
 "weather": [
  { "id": 800, "main": "Clear", "description": "맑음", "icon": "01d" }
 ],
 "base": "stations",
 "main": {
  "temp": 272.82,
  "feels_like": 268.77,
  "temp_min": 272.15,
  "temp_max": 273.15,
  "pressure": 1028,
  "humidity": 34
 },
 "visibility": 10000,
 "wind": { "speed": 1.03, "deg": 120 },
 "clouds": { "all": 0 },
 "dt": 1612420804,
```


온도가 켈빈 온도임;

java에서 JSON형태를
어떻게 처리할까?

map으로 처리.
pojo(DTO)에 매핑.

2번째가 장점이 많아.
관리하기도 쉽고

JSON형태가 플랫하지 않은데
=>툴을 써서 pojo에 매핑해보자

10페이지에 툴 있음

OpenApi - OpenWeather

결과 json


```
  "sys": {
    "type": 1,
    "id": 8105,
    "country": "KR",
    "sunrise": 1612391603,
    "sunset": 1612429114
  },
  "timezone": 32400,
  "id": 1835848,
  "name": "Seoul",
  "cod": 200
}
```

OpenApi - OpenWeather

날씨 아이콘 URL

- http://openweathermap.org/img/w/{icon}.png

```
{
 "coord": { "lon": 126.9778, "lat": 37.5683 },
 "weather": [
  { "id": 800, "main": "Clear", "description": "맑음", "icon": "01d" }
 ],
 "base": "stations",
 ...
}
```

JSON에서 추출해서 url구성후 요청

- http://openweathermap.org/img/w/01d.png

저렇게 배정해주면 날씨에 대한 아이콘 얻을수있음

rest controller와 rest template은 역할이 다르다.
restcontroller는 서버 측, 즉 api를 제공하는 역할을 한다!! 예시) @GetMapping("/api/hello")
rest template는 클라 측, 즉 다른 서버의 api를 '호출'하는 역할을 한다!! restTemplate.getForObjects(...)

내 애플리케이션이 api를 제공한다면 @RestController필요.
내 애플리케이션이 외부 api를 호출한다면 RestTemplate필요.

KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

RestTemplate

axios와 대응하는

스프링에 포함된 기능

[KB] IT's Your Life

KB 국민은행

객체로 rest 방식 api를 호출할 수 있는 spring 내장 클래스

RestTemplate

REST 서비스의 호출 방법

- RestTemplate
  - Spring 3부터 지원
  - REST API 호출이후 응답을 받을 때까지 기다리는 동기 방식
- AsyncRestTemplate
  - Spring 4에 추가
  - 비동기 RestTemplate
- WebClient
  - Spring 5에 추가
  - 논블럭, 리엑티브 웹 클라이언트로 동기, 비동기 방식 지원

참고로 spring 5부터는 rest template는 deprecated됐다
또한 비동기 rest template도 deprecated됐다.

참고 webclient는 resttemplate의 비동기+논블로킹버전.
외부api를 호출할때 사용하는 클라이지만
동작 방식과 성능에서 resttemplate과 큰차이가 있다.
비동기 http클라이언트. resttemplate의 최신대체제.

하지만 spring mvc가 아닌 spring webflux 기반임.

✅ 언제 WebFlux를 써야 할까?

조건	WebFlux 적합 여부
실시간 데이터 스트리밍 (예: 채팅, 주식 등)	✅ 매우 적합
API Gateway / Proxy 서비스	✅ 적합
대량의 비동기 외부 호출	✅ 적합
단순 CRUD 위주의 앱	❌ MVC가 더 간단하고 효율적일 수 있음
기존 MVC 프로젝트	❌ 굳이 전환할 필요 없음 (성능 병목 없을 시)

항목	Spring MVC	Spring WebFlux
동작 방식	블로킹 (Blocking)	논블로킹 (Non-blocking)
서버 종류	Tomcat, Jetty (서블릿 기반)	Netty, Undertow (리액티브 기반)
API 응답 타입	일반 `String`, `ResponseEntity` 등	`Mono`, `Flux` 등 리액티브 타입
쓰레드 모델	요청당 하나의 쓰레드(Thread-per-request)	이벤트 루프 기반 (작은 쓰레드 수로 처리)
성능 (고트래픽)	제한적 (쓰레드 자원 소비 큼)	우수 (높은 동시성, 작은 리소스로 처리)
학습 난이도	낮음 (전통적인 방식)	높음 (리액티브 개념 필요)
주 사용 사례	CRUD 웹 앱, 내부 서비스	실시간, 스트리밍, 대규모 트래픽 API

RestTemplate

의존성

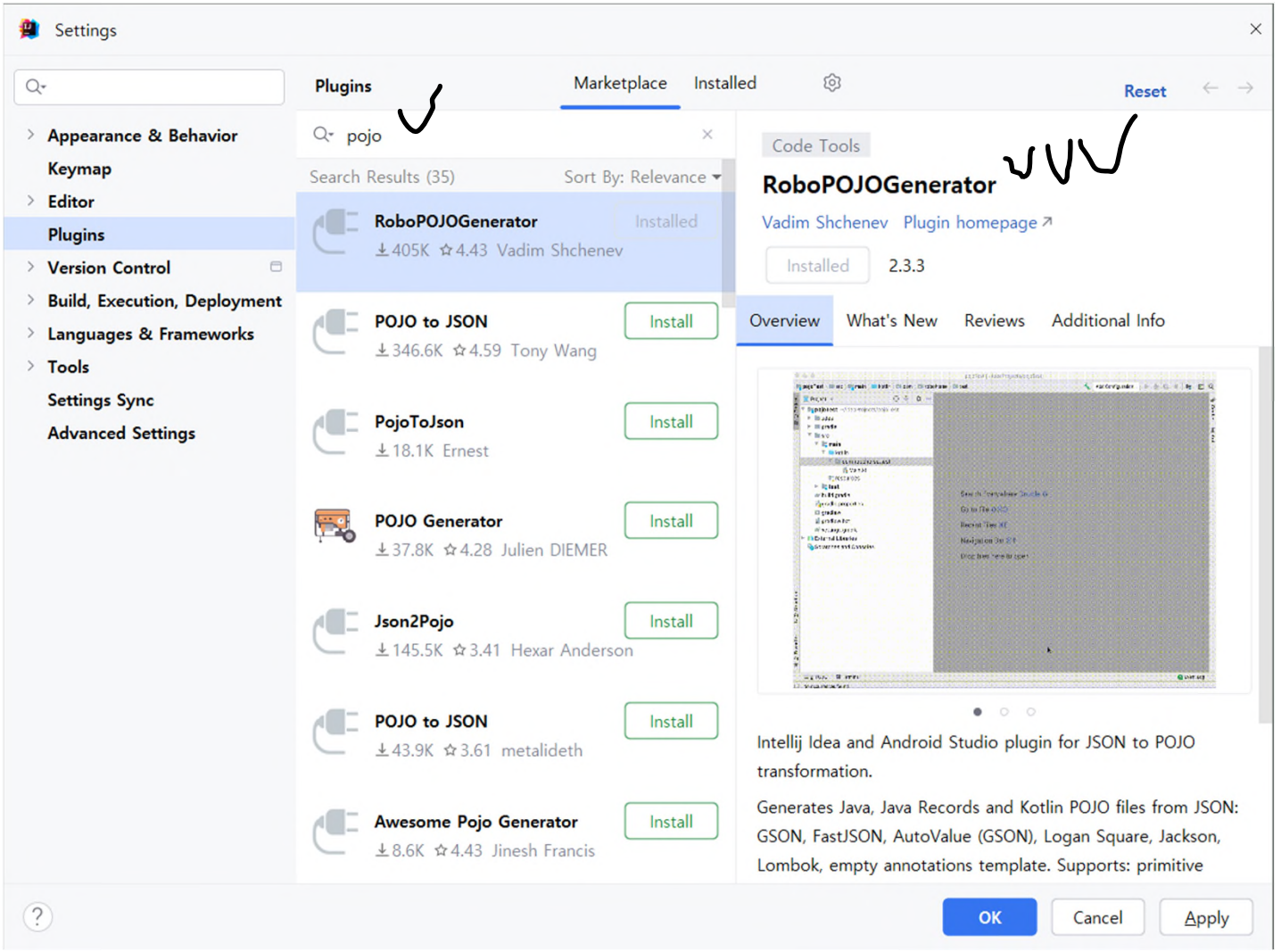
```
implementation 'org.apache.httpcomponents:httpcore:4.4.15'
implementation 'org.apache.httpcomponents:httpclient:4.5.13'
```

RestTemplate

- **Json to Java Object**
  - DTO Generator 플러그인
    - File > Settings... > Plugins
    - pojo 검색

JSON에서
키 값이 클래스명이 됨.

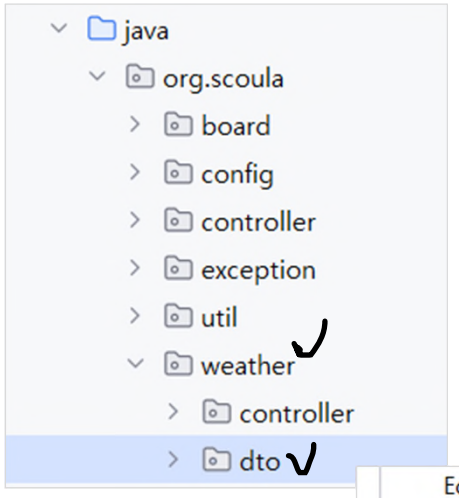
DTO로 매핑해줌.

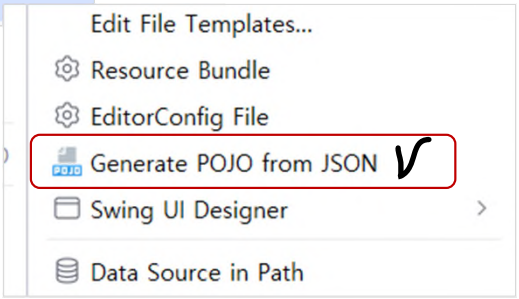


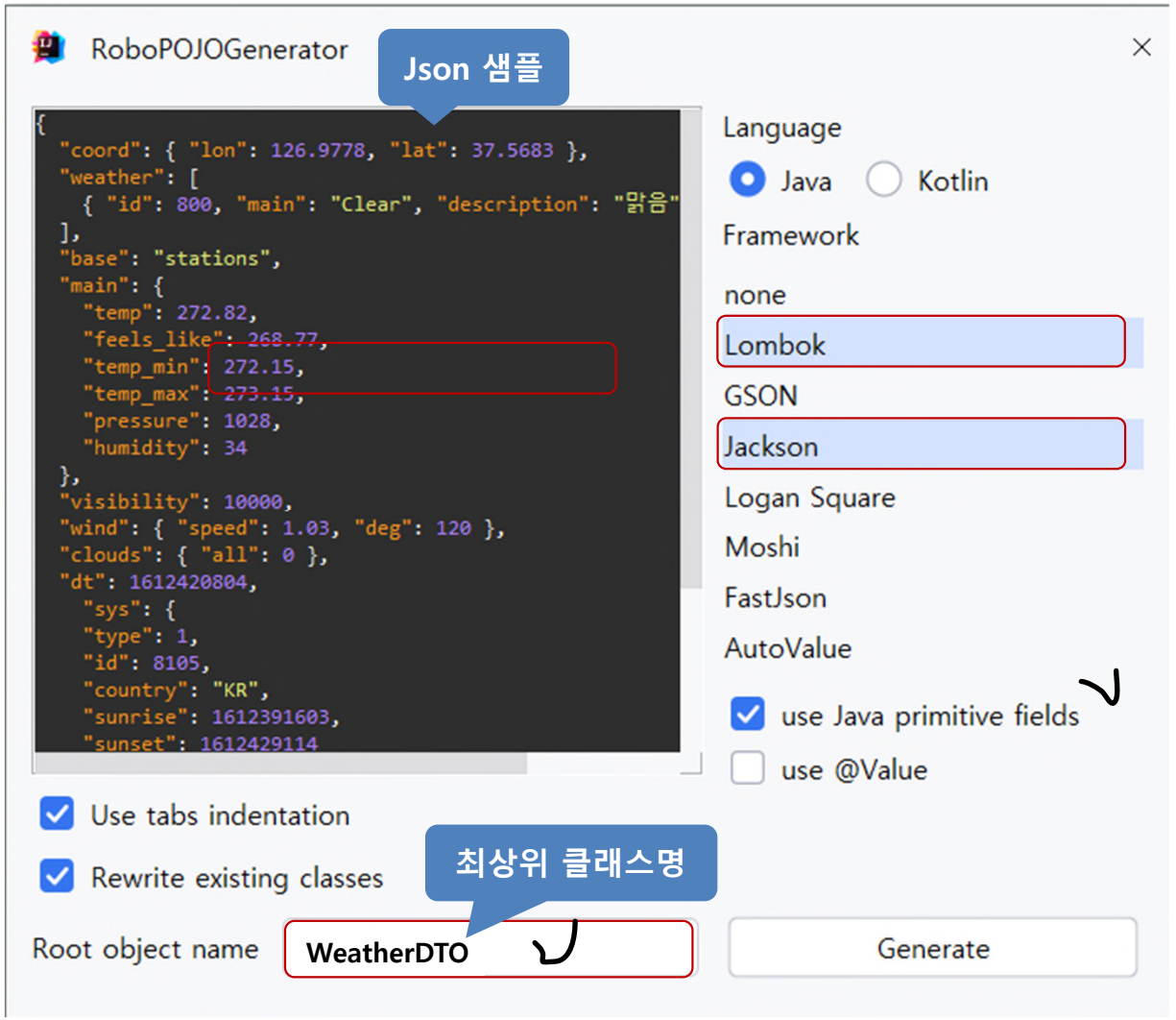
RestTemplate

변환하기

- 대상 패키지 선택 >>
  New > Generate POJO from JSON

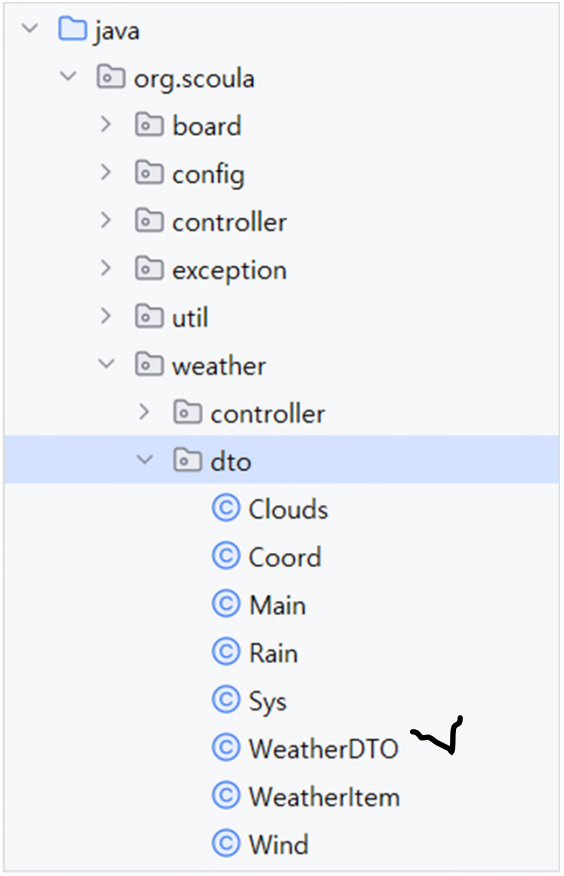






해당 플러그인 안되면 jsontopojo 구글검색해서 웹사이트에서 진행하는 방법도 있다
jsonschema2pojo 라는 사이트가 대표적
사이트에서 json jacson2어노테이션스타일 지정하고 ㄱㄱ

변환 결과




```
@Data
public class WeatherDTO{
        private int visibility;
        private int timezone;
        private Main main;
        private Clouds clouds;
        private Sys sys;
        private int dt;
        private Coord coord;
        private List<WeatherItem> weather;
        private String name;
        private int cod;
        private int id;
        private String base;
        private Wind wind;
}
```

RestTemplate

axios처럼 모든 요청이 메소드로 구분함.

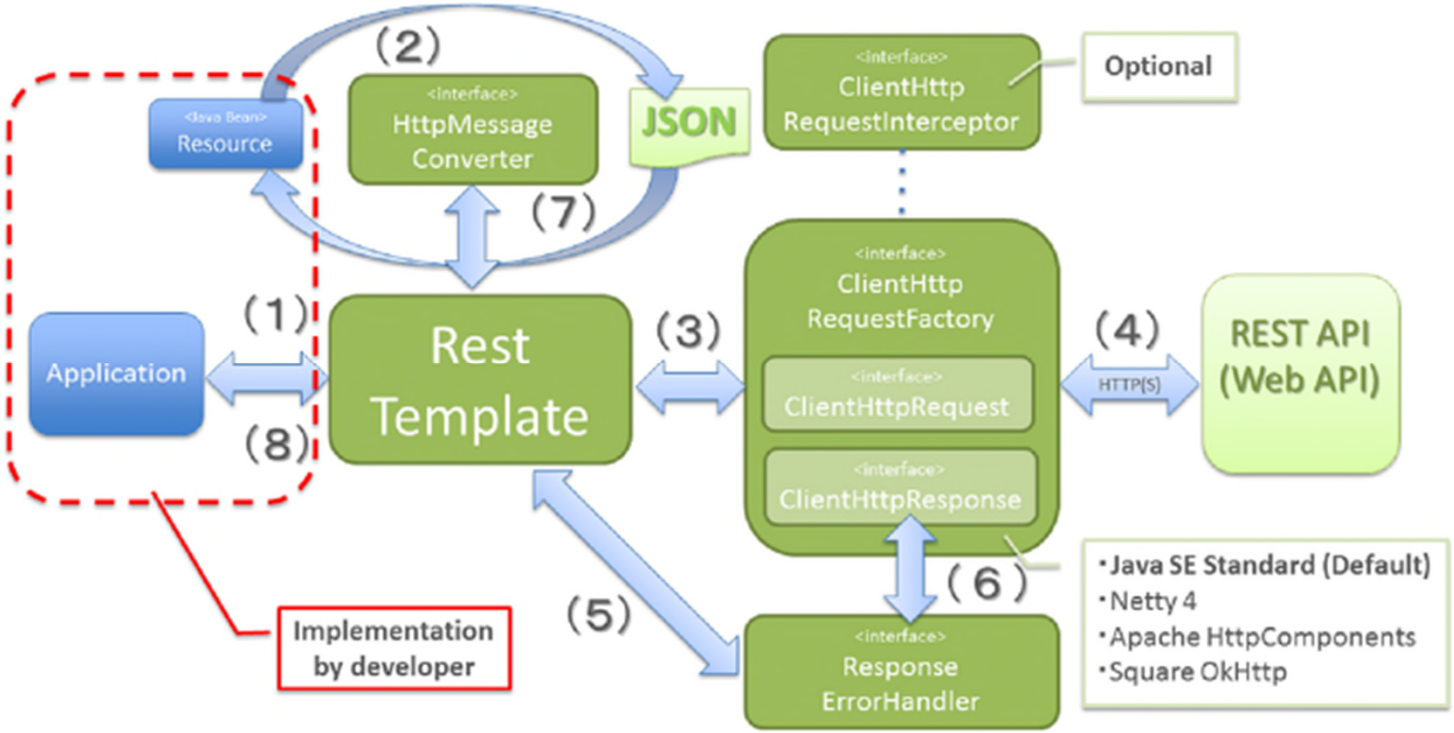
RestTemplate 메서드

메서드	HTTP	설명
getForObject	GET	GET 요청을 보내고 객체로 결과를 반환한다.
getForEntity	GET	GET 요청을 보내고 ResponseEntity로 결과를 반환한다.
postForLocation	POST	POST 요청을 보내고 결과로 헤더에 저장된 URI(리다이렉트 URI)를 결과로 반환받는다.
postForObject	POST	POST 요청을 보내고 객체로 결과를 반환한다.
postForEntity	POST	POST 요청을 보내고 ResponseEntity로 결과를 반환한다.
put	PUT	PUT 요청을 보낸다.
delete	DELETE	DELETE 요청을 보낸다.
exchange	any	HTTP 헤더를 직접 구성하고, 어떤 HTTP 메서드도 사용 가능하다.
execute	any	콜백을 사용하여 비동처리를 수행한다.

RestTemplate

RestTemplate의 동작 원리

- HttpClient
  - HTTP를 사용하여 통신하는 범용 라이브러리
- RestTemplate
  - HttpClient를 추상화(HttpEntity의 json, xml 등)해서 제공



RestTemplate

RestTemplate 사용 방법

GET 요청

```
RestTemplate restTemplate = new RestTemplate();
String url = UriComponentsBuilder.fromHttpUrl("기본 URL")
        .queryParam("속성명", "속성값")
        ...
        .toUriString();

WeatherDTO weather = restTemplate.getForObject(url, WeatherDTO.class);
```

요청을 빌드해야됨.

요청 url과 해당 타입으로 리턴하라

get요청을 오브젝트로
받아라!

알아서 JSON형식을
지정 타입으로 캐스팅까지해서 리턴

RestTemplate 사용 방법

- POST 요청
  - T postForObject(URI url, Object request, Class<T> responseType)
  - ResponseEntity<T> postForEntity (URI url, HttpEntity request, Class<T> responseType)

resources :: application.properties

```
#driver=com.mysql.cj.jdbc.Driver
#url=jdbc:mysql://127.0.0.1:3306/scoula_db
jdbc.driver=net.sf.log4jdbc.sql.jdbcapi.DriverSpy
jdbc.url=jdbc:log4jdbc:mysql://localhost:3306/scoula_db
jdbc.username=scoula
jdbc.password=1234

weather.url=http://api.openweathermap.org/data/2.5/weather
# 발급받은 본인의 API KEY 지정
weather.api_key=93ec9acd67f5e6d7fd08ff43c857eeac
weather.icon_url=http://openweathermap.org/img/w/%s.png
```

앱에서 사용할 open api 정보들

ServletConfig.java

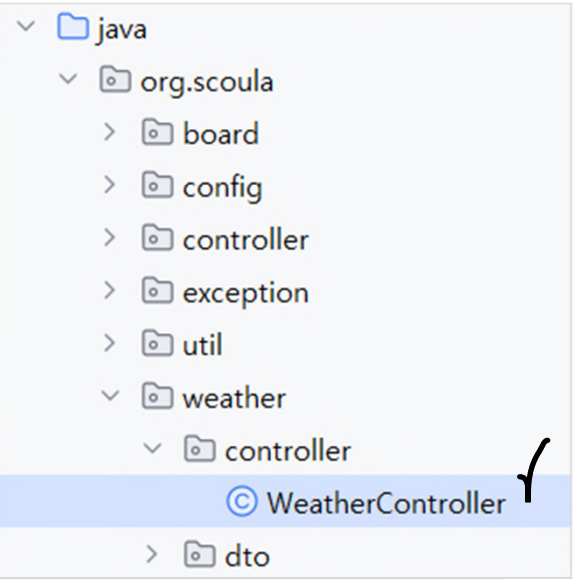
```
@EnableWebMvc
@ComponentScan(basePackages = {
        "org.scoula.controller",
        "org.scoula.exception",
        "org.scoula.board.controller",
        "org.scoula.weather.controller"
})
public class ServletConfig  implements WebMvcConfigurer {
…
```

WeatherController.java

```
@Controller
@Log4j2
@RequestMapping("/weather")
@PropertySource({"classpath:/application.properties"})
public class WeatherController {
    @Value("${weather.url}")
    private String URL;

    @Value("${weather.icon_url}")
    private String ICON_URL;

    @Value("${weather.api_key}")
    private String API_KEY;
```



/weather/seoul는 우리가 조회하기 위한

이거는 api에게 보내는url

RestTemplate

WeatherController.java

```
@GetMapping({"", "/{city}"})
public String weather(Model model, @PathVariable(value="city", required = false) String city) {
    city = city == null ? "seoul" : city;

    RestTemplate restTemplate = new RestTemplate();
    String url = UriComponentsBuilder.fromHttpUrl(URL)
            .queryParam("q", city)
            .queryParam("units", "metric")
            .queryParam("APPID", API_KEY)
            .queryParam("lang", "kr")
            .toUriString();
    WeatherDTO weather = restTemplate.getForObject(url, WeatherDTO.class);
    String iconUrl = ICON_URL.formatted(weather.getWeather().get(0).getIcon());
    log.info("오늘의 날씨: " + weather);

    model.addAttribute("city", city);
    model.addAttribute("weather", weather);
    model.addAttribute("iconUrl", iconUrl);

    return "weather/today";
  }
}
```

어노테이션으로 디폴트값을 지정해줄 수 있음

rest Template으로 url 구성하고

http://api.openweathermap.org/data/2.5/weather?q=seoul&units=metric&APPID=xxxxxxxxx&lang=kr

get요청하고 결과를 object로 반화되게

IO작업이니 예외처리해줘야함. 런타임 예외여서 처리코드가 없나보다! 라고 유추가능

weather.icon_url=http://openweathermap.org/img/w/%s.png

weather/today.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
  <title>Title</title>
</head>
<body>
  <div>
    <h2>${city}</h2>
    오늘의 날씨: ${weather.weather[0].description} <img src="${iconUrl}"/>
  </div>
  <div>
    온도: ${weather.main.temp}° / 습도: ${weather.main.humidity}%
  </div>
</body>
</html>
```