KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

OpenApi - OpenWeather

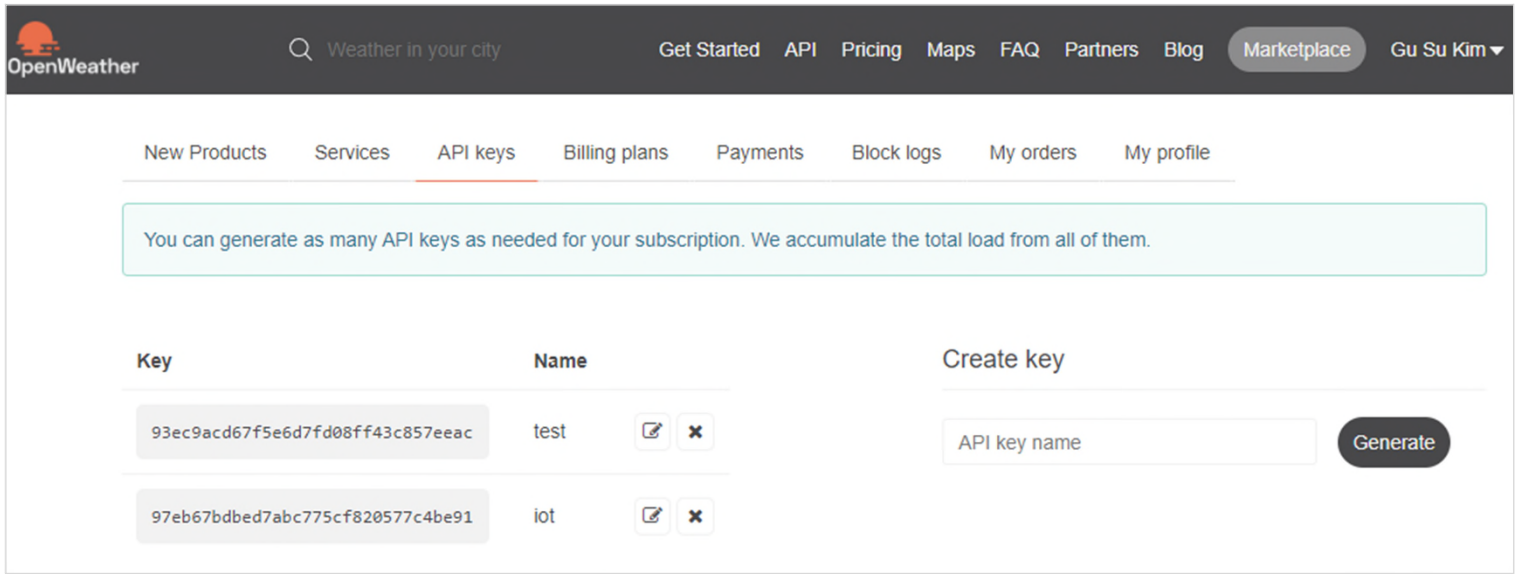
[KB] IT's Your Life

KB 국민은행

OpenApi - OpenWeather

openweather

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



OpenApi - OpenWeather

api 호출

- http://api.openweathermap.org/data/2.5/weather?q={city}&APPID={API_KEY}&lang=kr

  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


```
{
 "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,
```

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",
 ...
}
```

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

KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

RestTemplate

[KB] IT's Your Life

KB 국민은행

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

RestTemplate

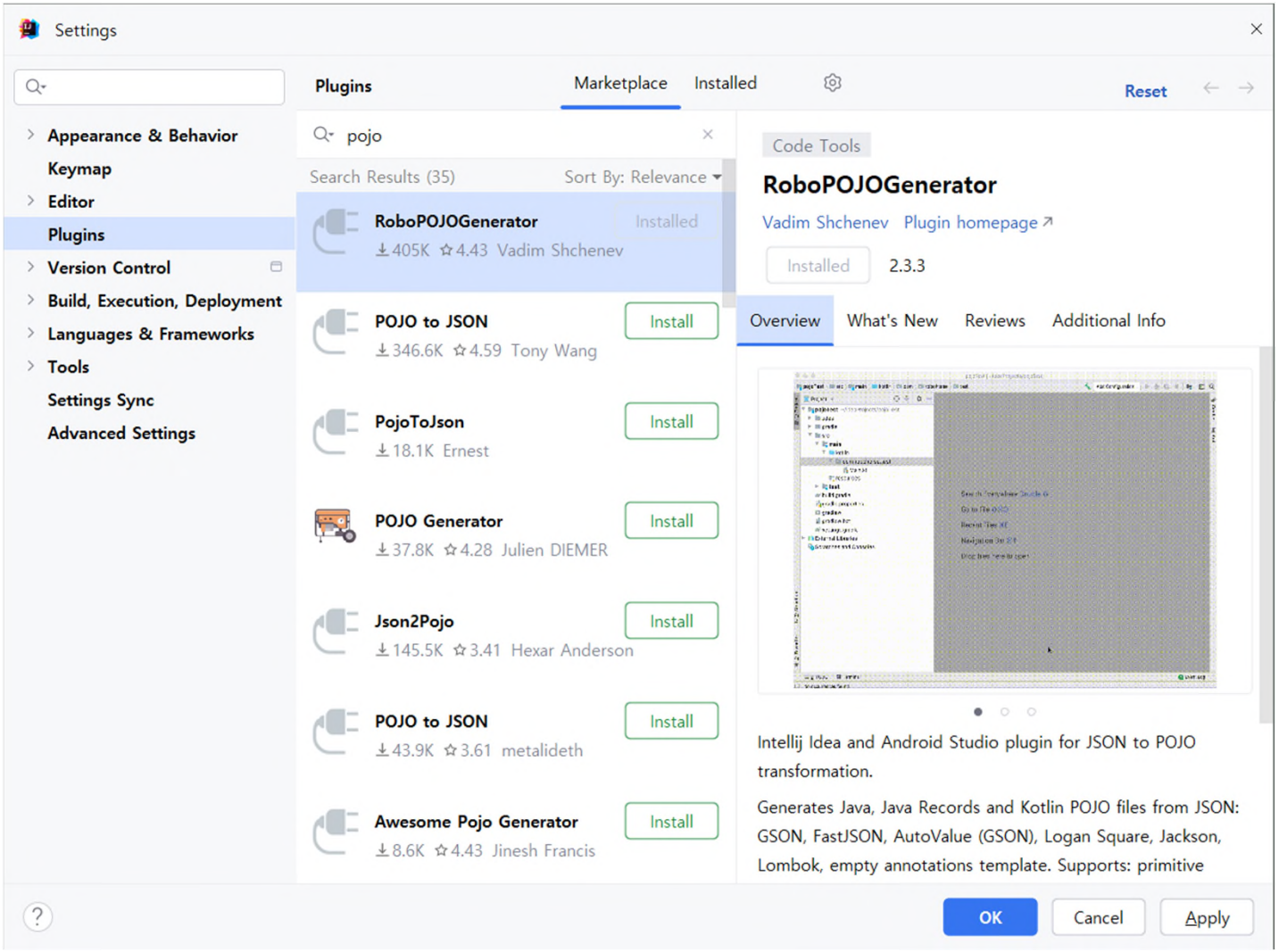
의존성

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

RestTemplate

Json to Java Object

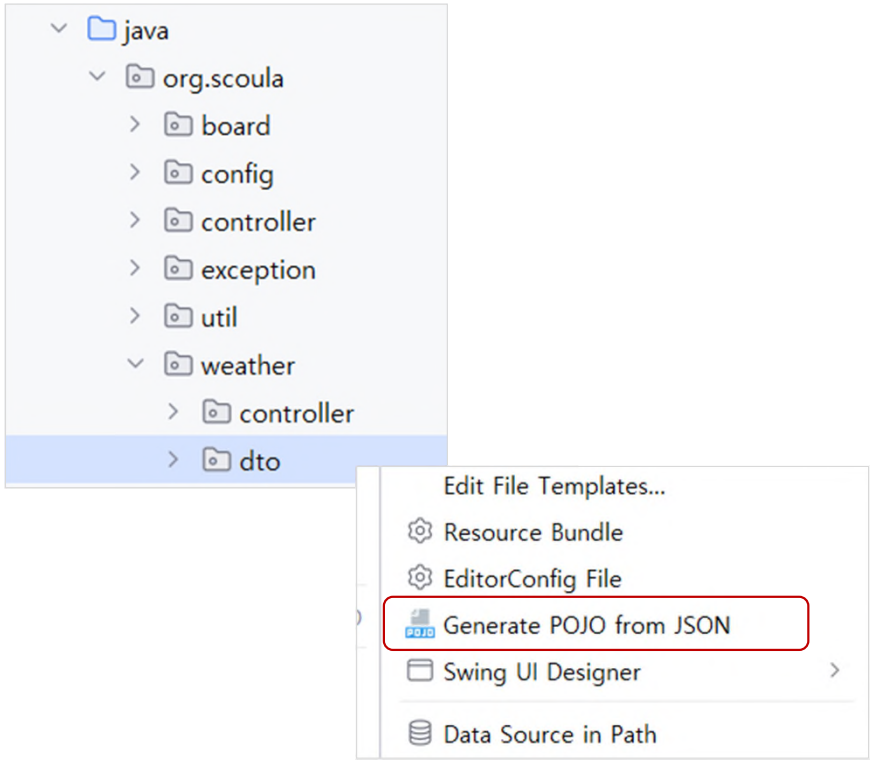
- DTO Generator 플러그인
  - File > Settings... > Plugins
  - pojo 검색

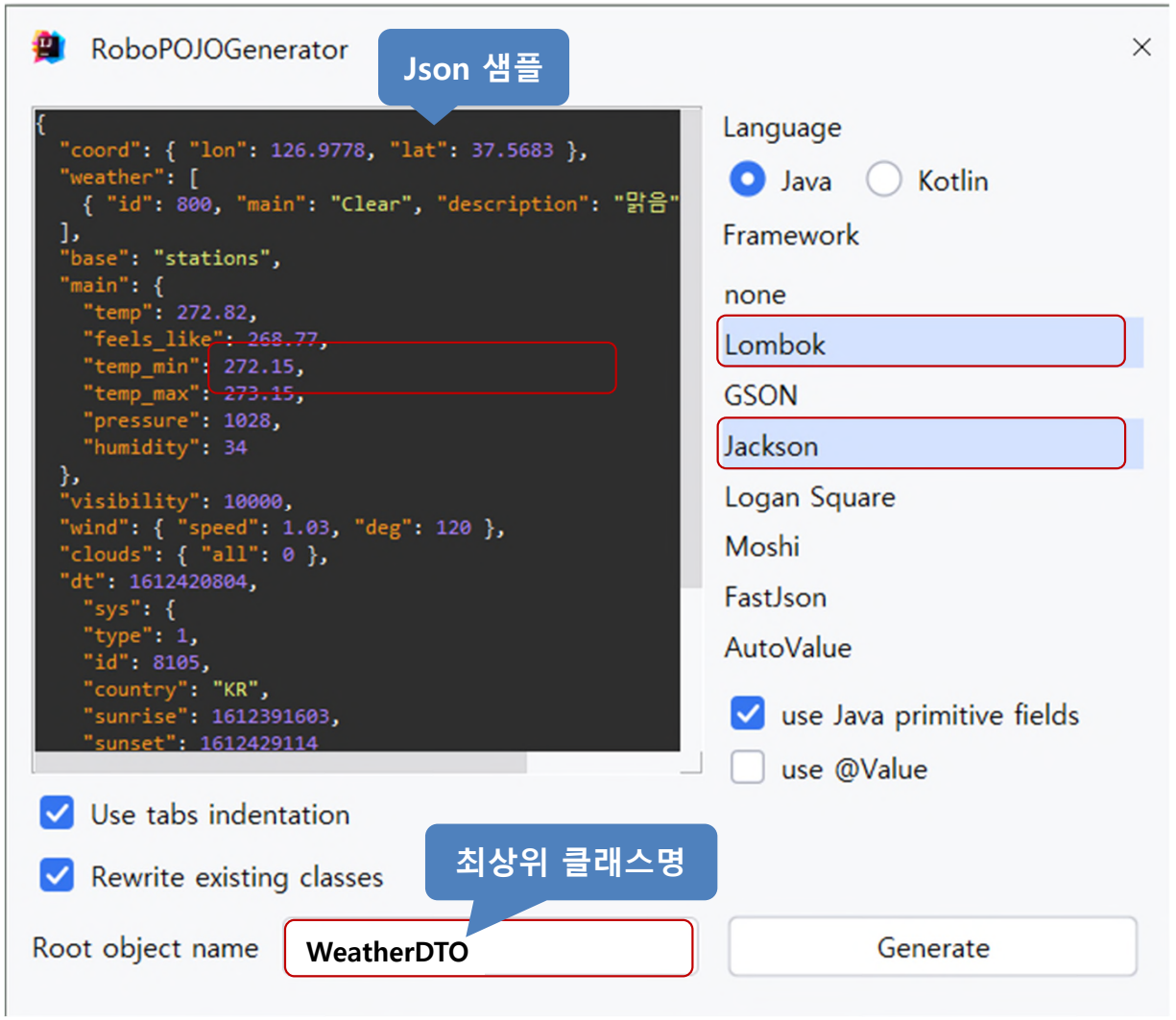


RestTemplate

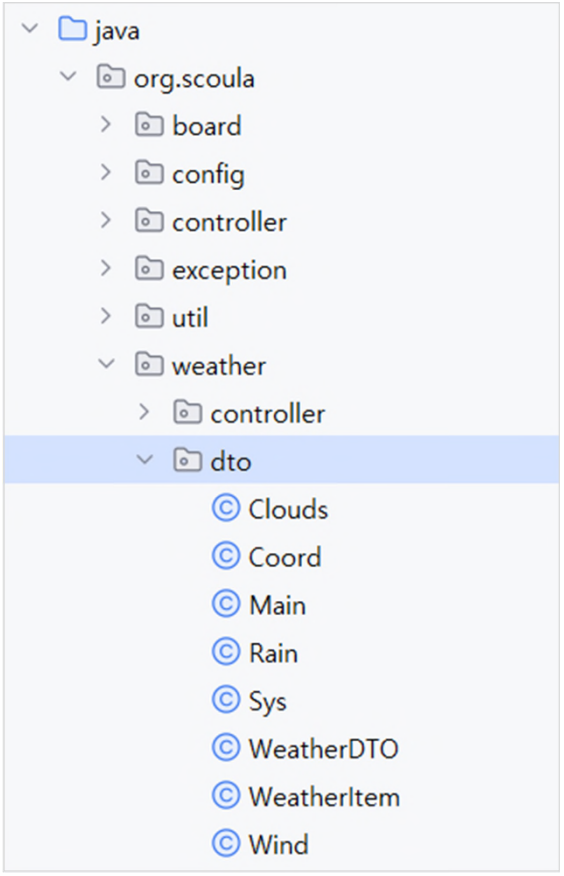
변환하기

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





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;
}
```

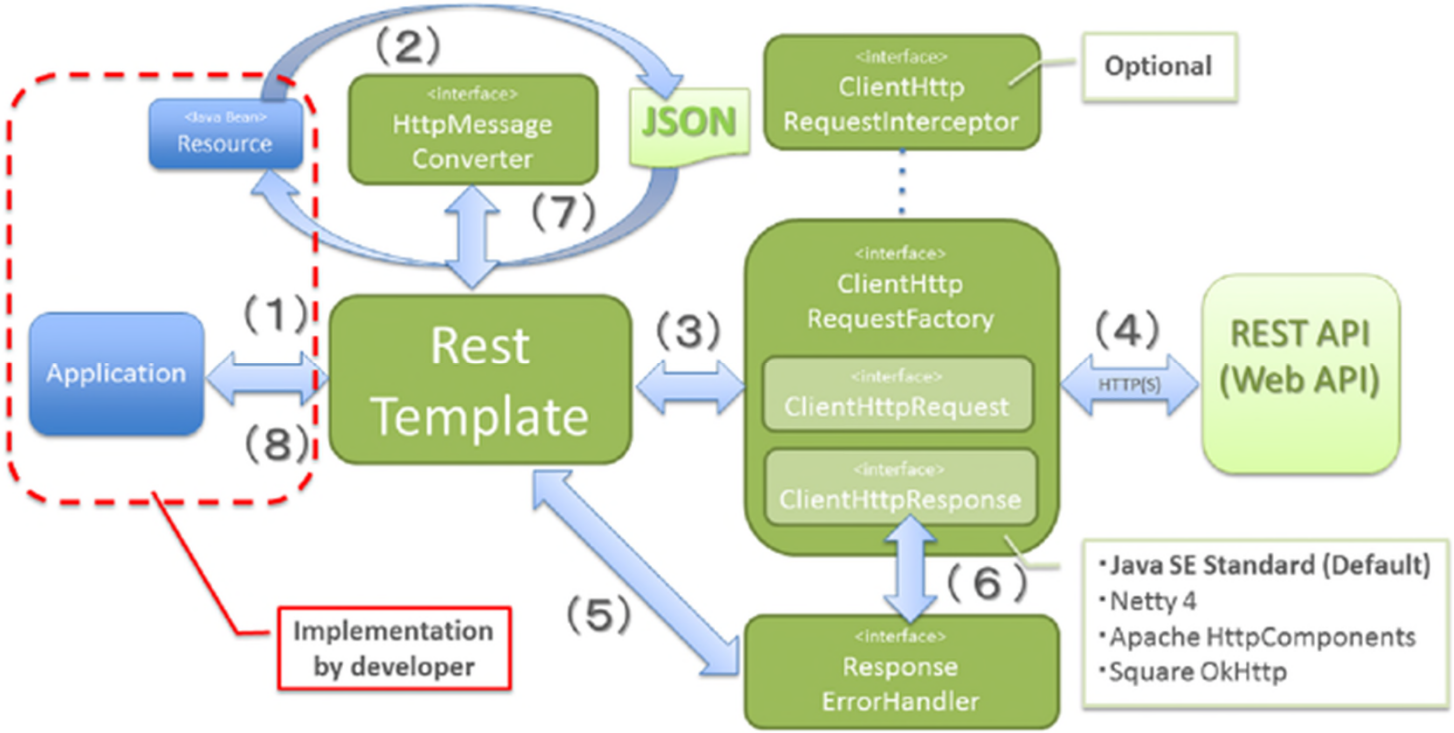
RestTemplate 메서드

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 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);
```

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
```

ServletConfig.java

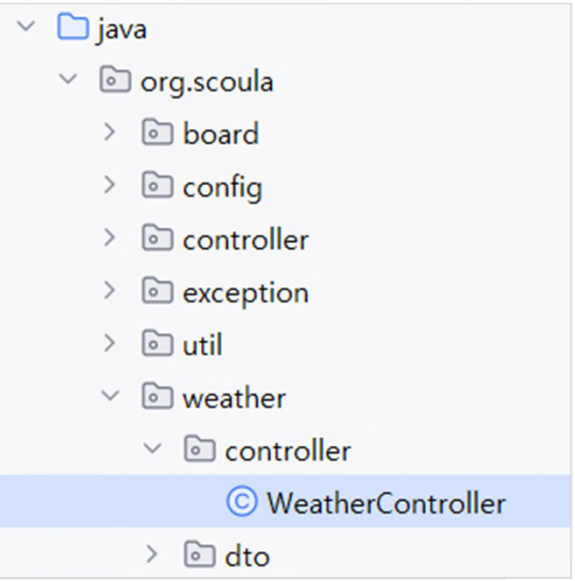
```
@EnableWebMvc
@ComponentScan(basePackages = {
      "org.scoula.controller",
      "org.scoula.exception",
      "org.scoula.board.controller",
      "org.scoula.weather.controller"
})
public class ServletConfig  implements WebMvcConfigurer {
…
```

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;
```



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";
  }
}
```

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

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>
```