

2024년 상반기 K-디지털 트레이닝

RestTemplate

[KB] IT's Your Life

✓ 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
- 논블록, 리액티브 웹 클라이언트로 동기, 비동기 방식 지원

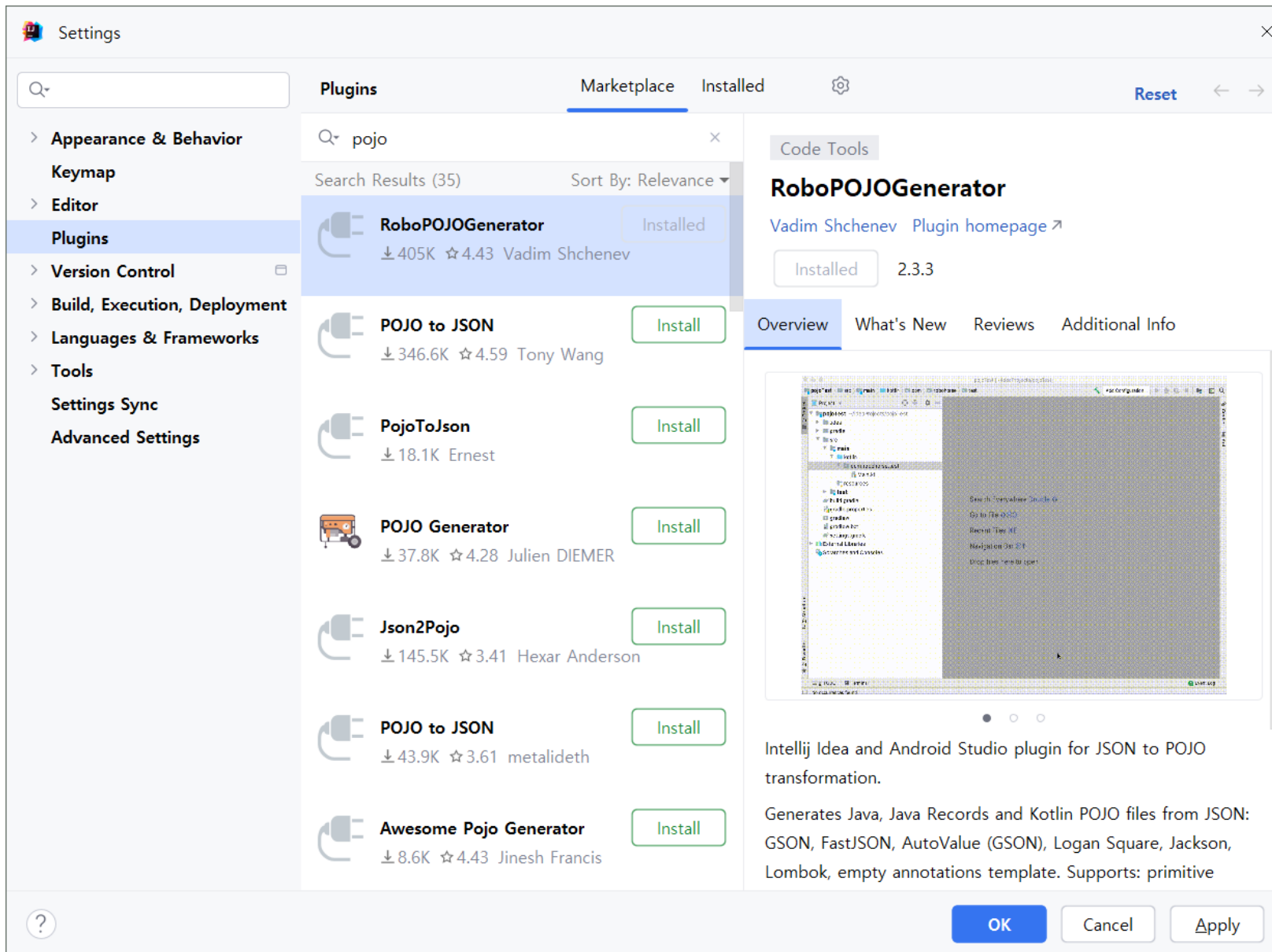
✓ 의존성

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

✓ Json to Java Object

○ DTO Generator 플러그인

- File > Settings... > Plugins
- pojo 검색



Settings

Plugins Marketplace Installed

Search Results (35) Sort By: Relevance

RoboPOJOGenerator Installed
405K ☆4.43 Vadim Shchenev

POJO to JSON Install
346.6K ☆4.59 Tony Wang

PojoToJson Install
18.1K Ernest

POJO Generator Install
37.8K ☆4.28 Julien DIEMER

Json2Pojo Install
145.5K ☆3.41 Hexar Anderson

POJO to JSON Install
43.9K ☆3.61 metalideth

Awsome Pojo Generator Install
8.6K ☆4.43 Jinesh Francis

RoboPOJOGenerator
Vadim Shchenev Plugin homepage
Installed 2.3.3

Overview What's New Reviews Additional Info

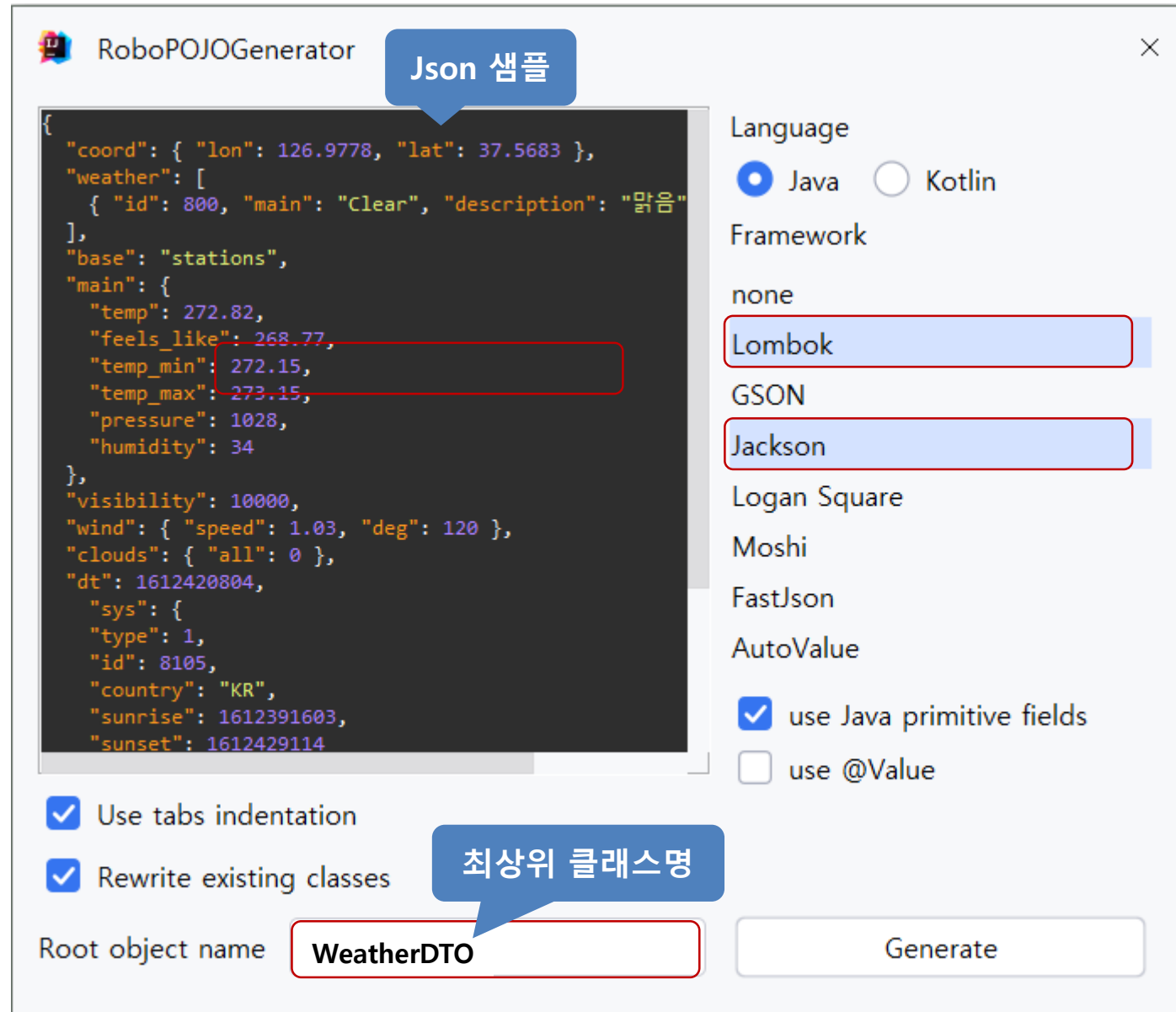
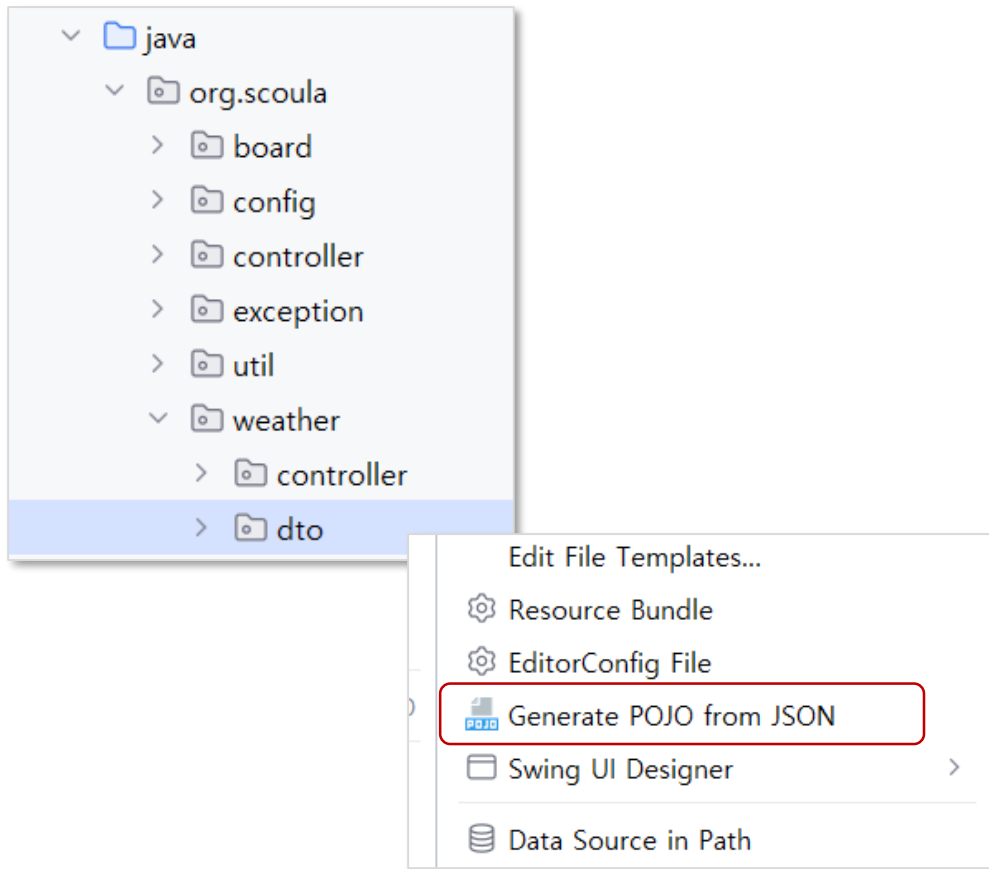
IntelliJ Idea and Android Studio plugin for JSON to POJO transformation.

Generates Java, Java Records and Kotlin POJO files from JSON: GSON, FastJSON, AutoValue (GSON), Logan Square, Jackson, Lombok, empty annotations template. Supports: primitive

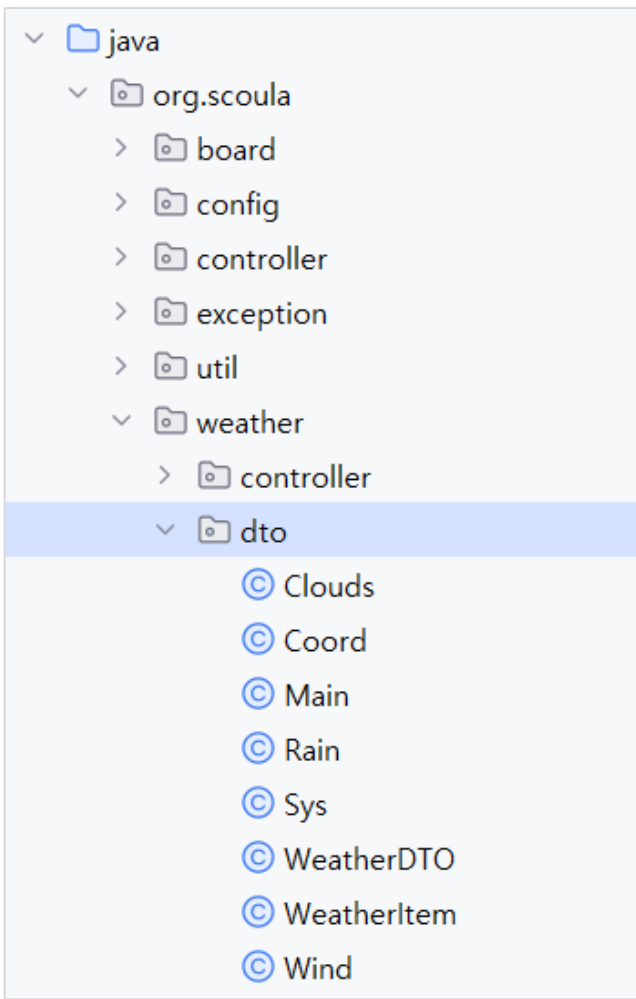
OK Cancel Apply

✓ 변환하기

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



✓ 변환 결과



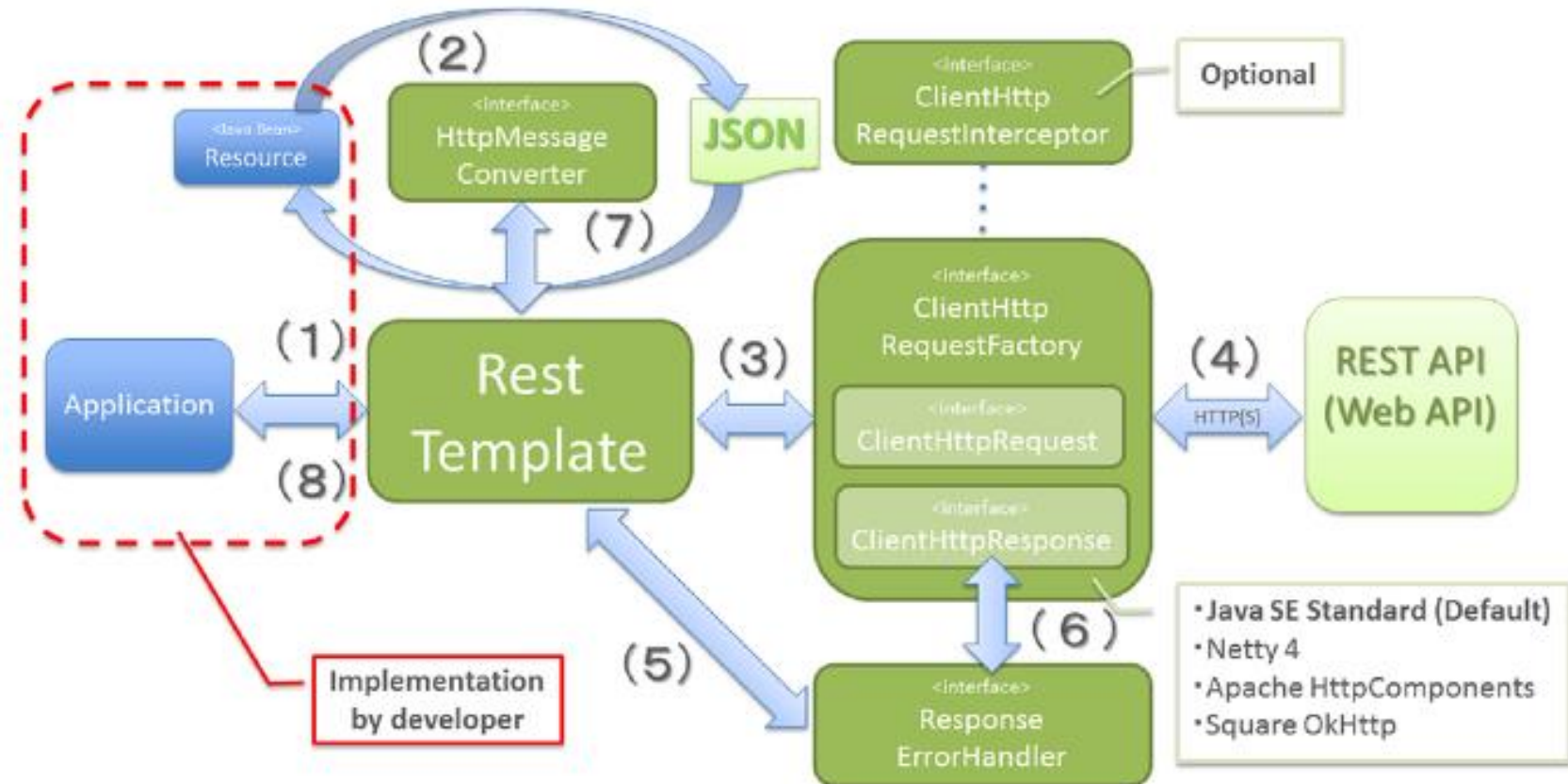
```
@Data
public class WeatherDTO{
    private int visibility;
    private int timezone;
    private Main main;
    private Clouds clouds;
    private Sys sys;
    private int dt;
    private Coord coord;
    private List<WeatherItem> weather;
    private String name;
    private int cod;
    private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

✓ 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")  
    .queryParams("속성명", "속성값")  
    ...  
    .toUriString();
```

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

✓ RestTemplate 사용 방법

○ POST 요청

- `T postForObject(URL url, Object request, Class<T> responseType)`
- `ResponseEntity<T> postForEntity (URL 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
```

resources :: application.properties

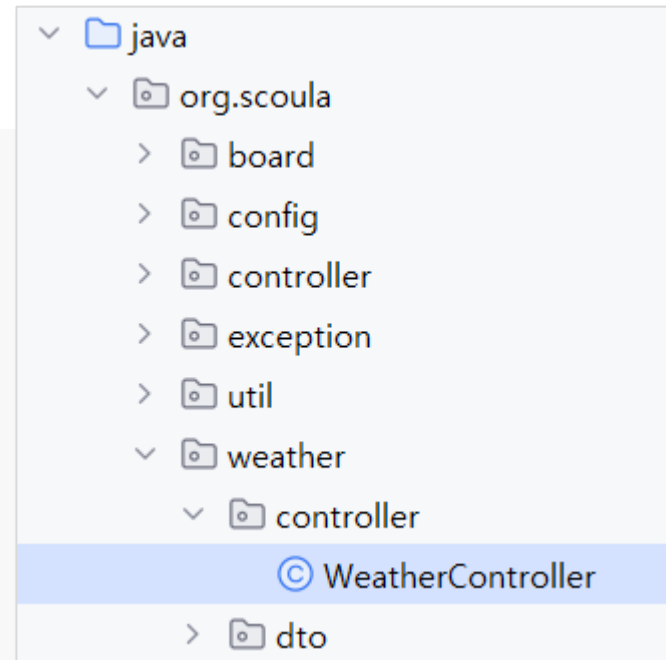
```
@EnableWebMvc
@ComponentScan(basePackages = {
    "org.scoula.controller",
    "org.scoula.exception",
    "org.scoula.board.controller",
    "org.scoula.weather.controller"
})
public class ServletConfig implements WebMvcConfigurer {
    ...
}
```

WeatherController.java

```
@Controller
@Slf4j
@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)  
        .queryParams("q", city)  
        .queryParams("units", "metric")  
        .queryParams("APPID", API_KEY)  
        .queryParams("lang", "kr")  
        .toUriString();
```

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

```
    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";
```

```
}
```

```
}
```

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} 
  </div>
  <div>
    온도: ${weather.main.temp}° / 습도: ${weather.main.humidity}%
  </div>
</body>
</html>
```

오늘의 날씨: 구름조금
온도: 29.76° / 습도: 58%

