

날씨 정보 API 사용

기상청에서 제공하는 날씨 정보를 공공데이터 포털을 통해서 API 활용하였다

해당 방법은 오늘 ~ 10일 까지의 날씨 정보를 제공하며 총 16개의 지역에 대해서만 사용하였다.

(서울, 부산, 대구, 인천, 광주, 대전, 울산, 경기, 강원, 충북, 충남, 전북, 전남, 경북, 경남, 제주)



해당 글에 사용된 모든 데이터와 자료 출처는 공공데이터 포털에서 제공하는 기상청 조회서비스 오픈 API 활용가이드들에서 인용하였습니다

→ [단기예보 조회서비스 링크](#)

[기상청41_단기예보 조회서비스 오픈API활용가이드_\(240715\).zip](#)

→ [중기예보 조회서비스 링크](#)

[기상청28_중기예보 조회서비스 오픈API활용가이드_240715.zip](#)

→ 실제 사용 예시

날씨 정보 확인하기

서울 ▼	지역 선택				
서울	날짜	최저기온 °C	최고기온 °C	SKY+PTY(날씨)	POP(강수확률)
부산					
대구	11-18	-2	7	맑음	0
인천					
광주	11-19	0	10	구름많음	20
대전					
울산	11-20	2	11	구름많음	20
경기					
강원	11-21	6	14	맑음	20
충북					
충남	11-22	2	9	맑음	10
전북					
전남	11-23	1	10	맑음	10
경북					
경남	11-24	3	12	구름많음	20
제주					
	2024-11-25	4	14	흐림	40
	2024-11-26	7	11	흐림	40
	2024-11-27	3	8	흐림	40
	2024-11-28	0	7	맑음	20

0. Introduction

우선, 기상청에서 제공하는 날씨 정보는 크게 2가지였다. 1. 단기예보 (오늘 ~ 2일 후) 와 2. 중기예보 (3일 ~ 10일)로 구성되어 있었다. 제공되는 정보들에서 자료 코드 명들은 다음과 같다.

	단기	중기
최저기온	TMN	taMin
최고기온	TMX	taMax
하늘상태	SKY	wf
강수형태	PTY	-
강수확률	POP	rnSt

단기예보는 시간 별로 모든 기상 정보를 다 제공하였으며 중기예보는 날짜 별로 기상 정보 별로 (기온 / 기상) 다르게 제공하였다. 이 때, 공통적인 최저기온, 최고기온, 강수확률은 데이터를 그대로 내보냈지만 **하늘상태와 강수형태**의 경우 단기예보와 중기예보가 제공하는 데이터가 달랐기 때문에 다음과 같이 가공하였다.

단기예보 조회서비스 오픈 API 활용가이드에 따르면 하늘 상태 코드와 강수 형태 코드에 대한 설명은 다음과 같다

특정 요소의 코드값 및 범주

- 하늘상태(SKY) 코드 : 맑음(1), 구름많음(3), 흐림(4)
- 강수형태(PHY) 코드 : (초단기) 없음(0), 비(1), 비/눈(2), 눈(3), 빗방울(5), 빗방울눈날림(6), 눈날림(7) ←
(단기) 없음(0), 비(1), 비/눈(2), 눈(3), 소나기(4) ←

이로 인해 단기예보의 경우 SKY와 PHY 코드 두 가지를 합성하여서 기상 정보를 만들었다.

→ 단기예보 처리 코드 중

```
String wthrSKY_PHY = "";
switch (wthrSKY) {
case "1":
    wthrSKY_PHY += "맑음";
    break;
case "3":
    wthrSKY_PHY += "구름많음";
    break;
case "4":
    wthrSKY_PHY += "흐림";
    break;
}
switch (wthrPHY) {
case "1":
    wthrSKY_PHY += " (비)";
    break;
case "2":
    wthrSKY_PHY += " (비/눈)";
    break;
case "3":
    wthrSKY_PHY += " (눈)";
    break;
case "4":
    wthrSKY_PHY += " (소나기)";
    break;
}
```

중기예보는 하나의 형태로 기상 정보가 제공되어 그것을 그대로 활용하였다.

1. 단기예보

→ 정보 제공 시간

○ 단기예보

- Base_time : 0200, 0500, 0800, 1100, 1400, 1700, 2000, 2300 (1일 8회)
- API 제공 시간(~이후) : 02:10, 05:10, 08:10, 11:10, 14:10, 17:10, 20:10, 23:10

○ 최고/최저기온의 발표시간별 저장되는 예보자료 시간

발표시각 (KST)	최저기온				최고기온			
	오늘	내일	모레	글피	오늘	내일	모레	글피
2	○	○	○		○	○	○	
5		○	○		○	○	○	
8		○	○		○	○	○	
11		○	○		○	○	○	
14		○	○			○	○	
17		○	○	○		○	○	○
20		○	○	○		○	○	○
23		○	○	○		○	○	○

해당 글에서는 오늘 ~ 2일 후까지의 정보가 필요하기 때문에 발표시각 Base_time : 0200을 사용하였다.

⇒ 단기예보조회 사용

상세기능

목록 조회

실황정보를 조회하기 위해 발표일자, 발표시각, 예보지점 X 좌표, 예보지점 Y 좌표의 조회 조건으로 자료구분코드, 실험값, 발표일자, 발표시각, 예보지점 X 좌표, 예보지점 Y 좌표의 정보를 조회하는 기능

활용승인 절차 개발단계 : 자동승인 / 운영단계 : 자동승인
 신청가능 트랙 개발계정 : 10,000 / 운영계정 : 활용사례 등록시 신청하면 트랙픽 증가 가능
 요청주소 http://apis.data.go.kr/1360000/VilageFcstInfoService_2.0/getUltraSrtNcst
 서비스URL http://apis.data.go.kr/1360000/VilageFcstInfoService_2.0

[활용신청](#)

요청변수(Request Parameter)

항목명(국문)	항목명(영문)	항목크기	항목구분	샘플데이터	항목설명
서비스키	ServiceKey	4	필수	-	공공데이터포털에서 받은 인증키
페이지 번호	pageNo	4	필수	1	페이지번호
한 페이지 결과 수	numOfRows	4	필수	1000	한 페이지 결과 수
응답자료형식	dataType	4	옵션	XML	요청자료형식(XML/JSON) Default: XML
발표일자	base_date	8	필수	20210628	'21년 6월 28일 발표
발표시각	base_time	4	필수	0600	06시 발표(정시단위)
예보지점 X 좌표	nx	2	필수	55	예보지점의 X 좌표값
예보지점 Y 좌표	ny	2	필수	127	예보지점의 Y 좌표값

⇒ 요청 변수

요청변수(Request Parameter)

항목명(국문)	항목명(영문)	항목크기	항목구분	샘플데이터	항목설명
서비스키	ServiceKey	4	필수	-	공공데이터포털에서 받은 인증키
페이지 번호	pageNo	4	필수	1	페이지번호
한 페이지 결과 수	numOfRows	4	필수	1000	한 페이지 결과 수
응답자료형식	dataType	4	옵션	XML	요청자료형식(XML/JSON) Default: XML
발표일자	base_date	8	필수	20210628	'21년 6월 28일 발표
발표시각	base_time	4	필수	0600	06시 발표(정시단위)
예보지점 X 좌표	nx	2	필수	55	예보지점의 X 좌표값
예보지점 Y 좌표	ny	2	필수	127	예보지점의 Y 좌표값

→ API 요청 코드

```
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.io.BufferedReader;
import java.io.IOException;

public class ApiExplorer {
```

```

public static void main(String[] args) throws IOException
    StringBuilder urlBuilder = new StringBuilder("http://.
    urlBuilder.append("? " + URLEncoder.encode("serviceKey
    urlBuilder.append("& " + URLEncoder.encode("pageNo", "U
    urlBuilder.append("& " + URLEncoder.encode("numOfRows"
    urlBuilder.append("& " + URLEncoder.encode("dataType",
    urlBuilder.append("& " + URLEncoder.encode("base_date"
    urlBuilder.append("& " + URLEncoder.encode("base_time"
    urlBuilder.append("& " + URLEncoder.encode("nx", "UTF-8
    urlBuilder.append("& " + URLEncoder.encode("ny", "UTF-8
    URL url = new URL(urlBuilder.toString());
    HttpURLConnection conn = (HttpURLConnection) url.open
    conn.setRequestMethod("GET");
    conn.setRequestProperty("Content-type", "application/
    System.out.println("Response code: " + conn.getRespon
    BufferedReader rd;
    if(conn.getResponseCode() >= 200 && conn.getResponseC
        rd = new BufferedReader(new InputStreamReader(con
    } else {
        rd = new BufferedReader(new InputStreamReader(con
    }
    StringBuilder sb = new StringBuilder();
    String line;
    while ((line = rd.readLine()) != null) {
        sb.append(line);
    }
    rd.close();
    conn.disconnect();
    System.out.println(sb.toString());
}
}

```

⇒ 아래는 해당 API를 사용해서 뽑은 예시 결과 파일이다.

short_term_weather.xml

2. 중기 예보

- 중기육상예보조회

목록

중기육상예보조회

조회

예보구역코드, 발표시각의 조회 조건으로 예보일로부터 3일에서 10일까지 육상날씨정보를 조회하는 기능

활용승인 절차 개발단계 : 자동승인 / 운영단계 : 자동승인

신청가능 트랙픽 개발계정 : 10,000 / 운영계정 : 활용사례 등록시 신청하면 트랙픽 증가 가능

요청주소 <http://apis.data.go.kr/1360000/MidFcstInfoService/getMidLandFcst>

서비스URL <http://apis.data.go.kr/1360000/MidFcstInfoService>

활용신청

요청변수(Request Parameter)

항목명(국문)	항목명(영문)	항목크기	항목구분	샘플데이터	항목설명
서비스키	ServiceKey	4	필수	-	공공데이터포털에서 받은 인증키
페이지 번호	pageNo	4	필수	1	페이지번호
한 페이지 결과 수	numOfRows	4	필수	10	한 페이지 결과 수
응답자료형식	dataType	4	옵션	XML	요청자료형식(XML/JSON)Default: XML
예보구역코드	regId	8	필수	11800000	1180000 서울, 인천, 경기도 11D10000 등 (활용가이드 하단 참고 자료 참조)
발표시각	tmFc	12	필수	202107300600	-월 2회(06:00,18:00)회 생성되며 발표시각을 입력 YYYYMMDD0600(1800)-최근 24시간 자료만 제공

출력결과(Response Element)

항목명(국문)	항목명(영문)	항목크기	항목구분	샘플데이터	항목설명
결과코드	resultCode	2	필수	00	결과코드
결과메시지	resultMsg	50	필수	OK	결과메시지

```
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.io.BufferedReader;
import java.io.IOException;

public class ApiExplorer {
    public static void main(String[] args) throws IOException {
        StringBuilder urlBuilder = new StringBuilder("http://");
        urlBuilder.append("?") + URLEncoder.encode("serviceKey", "UTF-8");
        urlBuilder.append("&") + URLEncoder.encode("pageNo", "UTF-8");
        urlBuilder.append("&") + URLEncoder.encode("numOfRows", "UTF-8");
        urlBuilder.append("&") + URLEncoder.encode("dataType", "UTF-8");
        urlBuilder.append("&") + URLEncoder.encode("regId", "UTF-8");
        urlBuilder.append("&") + URLEncoder.encode("tmFc", "UTF-8");
        URL url = new URL(urlBuilder.toString());
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("GET");
        conn.setRequestProperty("Content-type", "application/json");
```

```

        System.out.println("Response code: " + conn.getResponseCode());
        BufferedReader rd;
        if(conn.getResponseCode() >= 200 && conn.getResponseCode() < 300) {
            rd = new BufferedReader(new InputStreamReader(conn.getInputStream()));
        } else {
            rd = new BufferedReader(new InputStreamReader(conn.getErrorStream()));
        }
        StringBuilder sb = new StringBuilder();
        String line;
        while ((line = rd.readLine()) != null) {
            sb.append(line);
        }
        rd.close();
        conn.disconnect();
        System.out.println(sb.toString());
    }
}

```

[long_term_info.xml](#)

⇒ 강수확률, 날씨에 대한 정보를 얻을 수 있다.

- 중기기온조회

상세기능

목록 중기기온조회 조회

예보구역코드, 발표시각의 조회 조건으로 예보일로부터 3일에서 10일까지 최저/최고기온정보를 조회하는 기능

활용승인 절차 개발단계 : 자동승인 / 운영단계 : 자동승인
 신청가능 트래픽 개발계정 : 10,000 / 운영계정 : 활용사례 등록시 신청하면 트래픽 평가 가능
 요청주소 <http://apis.data.go.kr/1360000/MidFcstInfoService/getMidTa>
 서비스URL <http://apis.data.go.kr/1360000/MidFcstInfoService>

[활용신청](#)

요청변수(Request Parameter)

항목명(국문)	항목명(영문)	항목크기	항목구분	샘플데이터	항목설명
서비스키	ServiceKey	4	필수	-	공공데이터포털에서 받은 인증키
페이지 번호	pageNo	4	필수	1	페이지번호
한 페이지 결과 수	numOfRows	4	필수	10	한 페이지 결과 수
응답자료형식	dataType	4	옵션	XML	요청자료형식(XML/JSON)Default: XML
예보구역코드	regId	8	필수	11B10101	11B10101 서울, 11B20201 인천 등 (별칭역설자료 참고)
발표시각	tmFc	12	필수	201309030600	-월 2회(06:00,18:00)회 생성되며 발표시각을 입력- YYYYMMDD0600(1800) 최근 24시간 자료만 제공

응답결과(Response Element)

항목명(국문)	항목명(영문)	항목크기	항목구분	샘플데이터	항목설명
---------	---------	------	------	-------	------

```
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.io.BufferedReader;
import java.io.IOException;

public class ApiExplorer {
    public static void main(String[] args) throws IOException {
        StringBuilder urlBuilder = new StringBuilder("http://");
        urlBuilder.append("?") + URLEncoder.encode("serviceKey", "UTF-8");
        urlBuilder.append("&") + URLEncoder.encode("pageNo", "UTF-8");
        urlBuilder.append("&") + URLEncoder.encode("numOfRows", "UTF-8");
        urlBuilder.append("&") + URLEncoder.encode("dataType", "UTF-8");
        urlBuilder.append("&") + URLEncoder.encode("regId", "UTF-8");
        urlBuilder.append("&") + URLEncoder.encode("tmFc", "UTF-8");
        URL url = new URL(urlBuilder.toString());
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("GET");
        conn.setRequestProperty("Content-type", "application/json");
        System.out.println("Response code: " + conn.getResponseCode());
        BufferedReader rd;
        if(conn.getResponseCode() >= 200 && conn.getResponseCode() < 300) {
            rd = new BufferedReader(new InputStreamReader(conn.getInputStream(), "UTF-8"));
        } else {
            rd = new BufferedReader(new InputStreamReader(conn.getErrorStream(), "UTF-8"));
        }
        String line;
        while((line = rd.readLine()) != null) {
            System.out.println(line);
        }
    }
}
```

```

        rd = new BufferedReader(new InputStreamReader(con
    } else {
        rd = new BufferedReader(new InputStreamReader(con
    }
    StringBuilder sb = new StringBuilder();
    String line;
    while ((line = rd.readLine()) != null) {
        sb.append(line);
    }
    rd.close();
    conn.disconnect();
    System.out.println(sb.toString());
    }
}

```

long_term_ta.xml

⇒ 기온 정보들을 얻을 수 있다.

3. 위치 데이터 사전 처리

두 가지 예보에서 얻는 데이터들은 위와 같다. 이 때, 지역 별로 정보를 얻기 위해서는 단기예보의 경우 예보 지점의 x, y 좌표가 필요하며 중기예보의 경우 예보 구역 코드가 필요하다.

총 16개의 구역 서울, 부산, 대구, 인천, 광주, 대전, 울산, 경기, 강원, 충북, 충남, 전북, 전남, 경북, 경남, 제주 에 대해서 진행하였다.

해당 내용들을 API활용 가이드에서 제공한 정보들을 취합해서 다음과 같이 정리하였다.

- 단기 예보 조회용 x, y 좌표

날씨용 코드 241017 ☆ 📁 🌐

파일 수정 보기 삽입 서식 데이터 도구 확장 프로그램 도움말

🔍 메뉴 ↶ ↷ 🏠 📏 100% ▾ | W % .0. .00 123 | 기본값... ▾ | - 10 + | B I ↶ A |

J22 ▾ | 📏

	A	B	C	D	E	F	G	H
1	region	regId	regName	nx	ny	대표 도시	대표 도시	시
2	1	11B10101	서울	60	127			서울특별시
3	2	11H20201	부산	98	76			부산광역시
4	3	11H10701	대구	89	90			대구광역시
5	4	11B20201	인천	55	124			인천광역시
6	5	11F20501	광주	58	74			광주광역시
7	6	11C20401	대전	67	100			대전광역시
8	7	11H20101	울산	102	84			울산광역시
9	8	11B20601	경기	60	120	수원	수원	경기도
10	9	11D20501	강원	73	134	강릉	강릉	강원특별자치도
11	10	11C10101	충북	69	107	충주	충주	충청북도
12	11	11C20501	충남	68	100	부여	부여	충청남도
13	12	11F10201	전북	63	89	전주	전주	전북특별자치도
14	13	21F20802	전남	51	67	전주	전주	전라남도
15	14	11H10501	경북	87	106	안동	안동	경상북도
16	15	11H20701	경남	91	77	진주	진주	경상남도
17	16	11G00201	제주	52	38	제주	제주	제주특별자치도

(이 때, 광역시들을 제외하고는 각 지역의 최대한 중심 대표 도시들의 좌표를 활용하였다.)

- 중기 육상 예보 조회용 구역 코드

예보구역코드 [❏]	구역 [❏]
11B00000 [❏]	서울, 인천, 경기도 [❏]
11D10000 [❏]	강원도영서 [❏]
11D20000 [❏]	강원도영동 [❏]
11C20000 [❏]	대전, 세종, 충청남도 [❏]
11C10000 [❏]	충청북도 [❏]
11F20000 [❏]	광주, 전라남도 [❏]
11F10000 [❏]	전북자치도 [❏]
11H10000 [❏]	대구, 경상북도 [❏]
11H20000 [❏]	부산, 울산, 경상남도 [❏]
11G00000 [❏]	제주도 [❏]



중기 예보 조회의 경우 **기온 조회** 와 **육상 예보 조회** (기상 정보) 에 사용되는 **예보 구역 코드가 다르다!**

⇒ 중기 기온 조회의 경우 단기 예보 조회와 동일한 예보 구역 코드를 사용한다.

⇒ 중기 육상 예보 조회의 경우 다른 예보 구역 코드를 사용한다.

위 내용들을 모두 취합해서 MySQL 을 활용해서 DB를 생성하였다.

	region	reg_id_short	reg_name	nx	ny	reg_id_long
▶	1	11B10101	서울	60	127	11B00000
	2	11H20201	부산	98	76	11H20000
	3	11H10701	대구	89	90	11H10000
	4	11B20201	인천	55	124	11B00000
	5	11F20501	광주	58	74	11F20000
	6	11C20401	대전	67	100	11C20000
	7	11H20101	울산	102	84	11H20000
	8	11B20601	경기	60	120	11B00000
	9	11D20501	강원	73	134	11D10000
	10	11C10101	충북	69	107	11C10000
	11	11C20501	충남	68	100	11C20000
	12	11F10201	전북	63	89	11F10000
	13	21F20802	전남	51	67	11F20000
	14	11H10501	경북	87	106	11H10000
	15	11H20701	경남	91	77	11H20000
	16	11G00201	제주	52	38	11G00000
*	NULL	NULL	NULL	NULL	NULL	NULL

→ DB 테이블 생성 코드

```
CREATE TABLE `pjweather` (
  `wthrIdx` int NOT NULL AUTO_INCREMENT,
  `wthrDate` varchar(45) NOT NULL,
  `region` varchar(45) NOT NULL,
  `wthrTMin` varchar(45) NOT NULL,
  `wthrTMax` varchar(45) NOT NULL,
  `wthrSKY_PTY` varchar(45) NOT NULL,
  `wthrPOP` varchar(45) NOT NULL,
  `wthrPM10` varchar(45) DEFAULT NULL,
  `wthrEtc01` varchar(45) DEFAULT NULL,
  `wthrEtc02` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`wthrIdx`)
```



```
public class RegionVO {
    private String region, reg_id_short, reg_name, nx, ny, re

    // 아래는 getter & setter
```

→ xml parsing 하고 정보 저장하는 코드

- weathercontroller.java

```
package com.ict.mytravellist.WTHR.controller;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.util.Date;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

import com.ict.mytravellist.WTHR.service.WeatherService;
import com.ict.mytravellist.WTHR.vo.RegionVO;
import com.ict.mytravellist.vo.WeatherVO;

@Controller
public class WeatherController {

    @Autowired
    private WeatherService weatherService;
```

```

@GetMapping("/load_weather")
public void getWthrDatas(HttpServletRequest request) {
    try {
        // DB 초기화
        weatherService.deleteWthrInfo();

        for (int i = 1; i < 17; i++) {
            getWthrDataRegion(i);
            System.out.println(i + "번째 성공");
        }

        request.setAttribute("result", "1");
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void getWthrDataRegion(int regionNum) {
    int i = 0;
    String region = String.valueOf(regionNum);

    String shorts = weatherShort(region);
    int tMinIdx = 0;
    int skyIdx = 0;
    int dateIdx = 0;
    int ptyIdx = 0;
    int popIdx = 0;
    int tMaxIdx = 0;

    while (i < 3) {
        WeatherVO pvo = new WeatherVO();
        tMinIdx = shorts.indexOf("TMN", tMinIdx + 1);
        String wthrTMin = shorts.substring(tMinIdx + 79,
            shorts.substring(tMinIdx + 28, tMinIdx
                + shorts.substring(tMinIdx + 30, shorts.i

```

```

dateIdx = shorts.indexOf("<fcstTime>1200</fcstTim
skyIdx = shorts.indexOf("SKY", dateIdx + 1);
String wthrSKY = shorts.substring(skyIdx + 79, sh
ptyIdx = shorts.indexOf("PTY", dateIdx + 1);
String wthrPTY = shorts.substring(ptyIdx + 79, sh

String wthrSKY_PTY = "";
switch (wthrSKY) {
case "1":
    wthrSKY_PTY += "맑음";
    break;
case "3":
    wthrSKY_PTY += "구름맑음";
    break;
case "4":
    wthrSKY_PTY += "흐림";
    break;
}
switch (wthrPTY) {
case "1":
    wthrSKY_PTY += " (비)";
    break;
case "2":
    wthrSKY_PTY += " (비/눈)";
    break;
case "3":
    wthrSKY_PTY += " (눈)";
    break;
case "4":
    wthrSKY_PTY += " (소나기)";
    break;
}
popIdx = shorts.indexOf("POP", dateIdx + 1);
String wthrPOP = shorts.substring(popIdx + 79, sh

dateIdx = shorts.indexOf("<fcstTime>1300</fcstTim

```



```

        tMaxIdx = shorts.indexOf("TMX", tMaxIdx + 1);
        String wthrTMax = shorts.substring(tMaxIdx + 79,

        pvo.setWthrDate(wthrDate);
        pvo.setWthrTMin(wthrTMin);
        pvo.setWthrTMax(wthrTMax);
        pvo.setWthrSKY_PTY(wthrSKY_PTY);
        pvo.setWthrPOP(wthrPOP);
        pvo.setRegion(region);

        weatherService.insertWthrInfo(pvo);
        i++;
    }

    String longs = weatherLong(region);
    LocalDate now = LocalDate.now();
    while (i < 11) {

        WeatherVO pvo = new WeatherVO();
        String wthrDate = now.plusDays(i).toString();

        tMinIdx = longs.indexOf(String.valueOf("<taMin" +
        tMaxIdx = longs.indexOf(String.valueOf("<taMax" +

        String wthrTMin = null;
        String wthrTMax = null;

        if (i == 10) {
            wthrTMin = longs.substring(tMinIdx + 9, longs
            wthrTMax = longs.substring(tMaxIdx + 9, longs
        } else {
            wthrTMin = longs.substring(tMinIdx + 8, longs
            wthrTMax = longs.substring(tMaxIdx + 8, longs
        }

        String wthrPOP = null;
        String wthrSKY_PTY = null;
        int skyptyIdx = 0;

```

```

        if (i < 8) {
            popIdx = longs.indexOf(String.valueOf("<rnSt"
            skyptyIdx = longs.indexOf(String.valueOf("<wf

            wthrPOP = longs.substring(popIdx + 9, longs.i
            wthrSKY_PTY = longs.substring(skyptyIdx + 7,
        } else if (i < 10) {
            popIdx = longs.indexOf(String.valueOf("<rnSt"
            skyptyIdx = longs.indexOf(String.valueOf("<wf

            wthrPOP = longs.substring(popIdx + 7, longs.i
            wthrSKY_PTY = longs.substring(skyptyIdx + 5,
        } else {
            popIdx = longs.indexOf(String.valueOf("<rnSt"
            skyptyIdx = longs.indexOf(String.valueOf("<wf

            wthrPOP = longs.substring(popIdx + 8, longs.i
            wthrSKY_PTY = longs.substring(skyptyIdx + 6,
        }

        pvo.setWthrDate(wthrDate);
        pvo.setWthrTMin(wthrTMin);
        pvo.setWthrTMax(wthrTMax);
        pvo.setWthrSKY_PTY(wthrSKY_PTY);
        pvo.setWthrPOP(wthrPOP);
        pvo.setRegion(region);

        weatherService.insertWthrInfo(pvo);
        i++;
    }
}

public String weatherShort(String region) {
    RegionVO wvo = weatherService.getRegInfo(region);

    String nx = wvo.getNx();
    String ny = wvo.getNy();

```

```

// 오늘 날짜
SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMdd");
Date now = new Date();
String today = sdf.format(now);

BufferedReader rd = null;
URLConnection conn = null;
StringBuilder sb = null;

// 단기 예보
try {
    StringBuilder urlBuilder = new StringBuilder(
        "http://apis.data.go.kr/1360000/VilageFcns"
        urlBuilder.append("? " + URLEncoder.encode("servic
            + "= 서비스키");
        urlBuilder.append(
            "& " + URLEncoder.encode("pageNo", "UTF-8"
            urlBuilder.append("& " + URLEncoder.encode("numOfR
                + URLEncoder.encode("1000", "UTF-8")); /*
            urlBuilder.append("& " + URLEncoder.encode("dataTy
                + URLEncoder.encode("XML", "UTF-8")); /*
            urlBuilder.append("& " + URLEncoder.encode("base_d
            urlBuilder.append("& " + URLEncoder.encode("base_t
                + URLEncoder.encode("0200", "UTF-8")); /*
            urlBuilder.append(
                "& " + URLEncoder.encode("nx", "UTF-8") +
            urlBuilder.append(
                "& " + URLEncoder.encode("ny", "UTF-8") +
            URL url = new URL(urlBuilder.toString());
            conn = (URLConnection) url.openConnection();
            conn.setRequestMethod("GET");
            System.out.println("Response code: " + conn.getRe

            if (conn.getResponseCode() >= 200 && conn.getResp
                rd = new BufferedReader(new InputStreamReader
            } else {
                rd = new BufferedReader(new InputStreamReader

```

```

        }
        sb = new StringBuilder();

        String line;
        while ((line = rd.readLine()) != null) {
            sb.append(line);
        }

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            rd.close();
            conn.disconnect();
        } catch (Exception e2) {
            e2.printStackTrace();
        }
    }
    return sb.toString();
}

public String weatherLong(String region) {
    RegionVO wvo = weatherService.getRegInfo(region);
    String regId = wvo.getReg_id_short();

    SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMdd");
    Date now = new Date();
    String today = sdf.format(now);

    BufferedReader rd = null;
    HttpURLConnection conn = null;
    StringBuilder sb = null;

    try {
        StringBuilder urlBuilder = new StringBuilder(
            "http://apis.data.go.kr/13600000/MidFcstIn
            urlBuilder.append("? " + URLEncoder.encode("servic
            + "=서비스키");
    }

```

```

urlBuilder.append(
    "&" + URLEncoder.encode("pageNo", "UTF-8")
urlBuilder.append("&" + URLEncoder.encode("numOfR
    + URLEncoder.encode("10", "UTF-8")); /* 현재
urlBuilder.append("&" + URLEncoder.encode("dataTy
    + URLEncoder.encode("XML", "UTF-8")); /*
urlBuilder.append("&" + URLEncoder.encode("regId"
urlBuilder.append(
    "&" + URLEncoder.encode("tmFc", "UTF-8")
URL url = new URL(urlBuilder.toString());
conn = (HttpURLConnection) url.openConnection();
conn.setRequestMethod("GET");
System.out.println("Response code2: " + conn.getR

if (conn.getResponseCode() >= 200 && conn.getResp
    rd = new BufferedReader(new InputStreamReader
} else {
    rd = new BufferedReader(new InputStreamReader
}

sb = new StringBuilder();
String line;

while ((line = rd.readLine()) != null) {
    if (line.equals("<?xml version=\"1.0\" encodi
        sb.append(line);
        continue;
    }
    int start = line.indexOf("<item>");
    int end = line.lastIndexOf("</item>");
    sb.append(line.substring(start, end));
}

String result = weatherLong2(region, today);
sb.append(result);

return sb.toString();

```

```

    } catch (Exception e) {
        e.printStackTrace();
        return null;
    } finally {
        try {
            rd.close();
            conn.disconnect();
        } catch (Exception e2) {
            e2.printStackTrace();
        }
    }
}

public String weatherLong2(String region, String today) {
    BufferedReader rd = null;
    HttpURLConnection conn = null;
    StringBuilder sb = null;

    RegionVO wvo = weatherService.getRegInfo(region);
    String regIdLong = wvo.getReg_id_long();

    try {
        StringBuilder urlBuilder = new StringBuilder(
            "http://apis.data.go.kr/13600000/MidFcstIn
            urlBuilder.append("? " + URLEncoder.encode("servic
            + "=서비스키");
        urlBuilder.append(
            "& " + URLEncoder.encode("pageNo", "UTF-8"
            urlBuilder.append("& " + URLEncoder.encode("numOfR
            + URLEncoder.encode("10", "UTF-8")); /* 한
            urlBuilder.append("& " + URLEncoder.encode("dataTy
            + URLEncoder.encode("XML", "UTF-8")); /*
            urlBuilder.append("& " + URLEncoder.encode("regId"
            + URLEncoder.encode(regIdLong, "UTF-8"));
            urlBuilder.append(
                "& " + URLEncoder.encode("tmFc", "UTF-8")
            /* -일 2회(06:00,18:00)회 생성되며 발표시각을 입력 YYYY
            urlBuilder.toString());

```

```

        conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("GET");
        System.out.println("Response code3: " + conn.getResponseCode());

        if (conn.getResponseCode() >= 200 && conn.getResponseCode() < 300) {
            rd = new BufferedReader(new InputStreamReader(conn.getInputStream()));
        } else {
            rd = new BufferedReader(new InputStreamReader(conn.getErrorStream()));
        }

        sb = new StringBuilder();
        String line;

        while ((line = rd.readLine()) != null) {
            if (line.equals("<?xml version='1.0' encoding='utf-8'>")) {
                continue;
            }
            int start = line.indexOf("<item>") + 6;
            int end = line.lastIndexOf("</item>") + 7;
            sb.append(line.substring(start, end));
        }
        return sb.toString();
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    } finally {
        try {
            rd.close();
            conn.disconnect();
        } catch (Exception e2) {
            e2.printStackTrace();
        }
    }
}
}
}

```

→ weatherservice & dao

```

@Service
public class WeatherServiceImpl implements WeatherService{

    @Autowired
    private WeatherDAO weatherDAO;

    @Override
    public RegionVO getRegInfo(String region) {
        return weatherDAO.getRegInfo(region);
    }

    @Override
    public int insertWthrInfo(WeatherVO pvo) {
        return weatherDAO.insertWthrInfo(pvo);
    }

    @Override
    public int deleteWthrInfo() {
        return weatherDAO.deleteWthrInfo();
    }

    @Override
    public List<WeatherVO> getWthrInfo(String region) {
        return weatherDAO.getWthrInfo(region);
    }

    // 여기 부터는 DAO
    @Repository
    public class WeatherDAOImpl implements WeatherDAO {

        @Autowired
        private SqlSessionTemplate sqlSessionTemplate;

        @Override
        public RegionVO getRegInfo(String region) {
            return sqlSessionTemplate.selectOne("reginfo.getreginfo")
        }
    }
}

```



```

@Override
public int insertWthrInfo(WeatherVO pvo) {
    return sqlSessionTemplate.insert("reginfo.insertwthrinfo", pvo);
}

@Override
public int deleteWthrInfo() {
    return sqlSessionTemplate.delete("reginfo.delete");
}

@Override
public List<WeatherVO> getWthrInfo(String region) {
    return sqlSessionTemplate.selectList("reginfo.getwthrinfo", region);
}

```

→ mapper.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
    PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
    "https://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="reginfo">
    <select id="getreginfo" parameterType="String" resultType="WeatherVO">
        select * from regioninfo where region = #{region}
    </select>

    <delete id="delete">
        delete from pjweather
    </delete>

    <select id="insertwthrinfo" parameterType="wthrvo" resultType="wthrvo">
        insert into pjweather (wthrDate, wthrTMin, wthrTMax, wthrHumidity)
        values (#{wthrDate}, #{wthrTMin}, #{wthrTMax}, #{wthrHumidity})
    </select>

    <select id="getwthrinfo" parameterType="String" resultType="WeatherVO">
        select * from pjweather where region = #{region}
    </select>
</mapper>

```

→ 확인용 jsp

```
<div id="graph">
    <h2>날씨 정보 확인하기</h2><br>
    <select name="region" id="region">
        <option value="1" selected>서울</option>
        <option value="2">부산</option>
        <option value="3">대구</option>
        <option value="4">인천</option>
        <option value="5">광주</option>
        <option value="6">대전</option>
        <option value="7">울산</option>
        <option value="8">경기</option>
        <option value="9">강원</option>
        <option value="10">충북</option>
        <option value="11">충남</option>
        <option value="12">전북</option>
        <option value="13">전남</option>
        <option value="14">경북</option>
        <option value="15">경남</option>
        <option value="16">제주</option>
    </select>
    <button onclick="load()">지역 선택</button>

    <div>
        <table id="list">
            <thead>
                <tr><th>날짜</th><th>최저기온 °C</th>
            </thead>
            <tbody id="tbody">
            </tbody>
        </table>
    </div>

    <script type="text/javascript">
function load(){
    $("#tbody").empty();
    $.ajax({
```

```

        url : "/getwthrinfo",
        method : "post",
        data : "region="+$("#region").val(),
        dataType : "json",
        success : function(data){
            let tbody = "";
            $.each(data, function(index, obj)
                tbody += "<tr>";
                tbody += "<td>" + obj.wthrData;
                tbody += "<td>" + obj.wthrTMI;
                tbody += "<td>" + obj.wthrTMA;
                tbody += "<td>" + obj.wthrSKY;
                tbody += "<td>" + obj.wthrPOP;
                tbody += "</tr>"
            });
            $("#tbody").append(tbody);
        },
        error : function(){
            alert("날씨 정보 최신화 필요!")
        }
    })
}
</script>
</div>
</div>

```

→ restcontroller

```

@RestController
public class WeatherAjaxController {

    @Autowired
    private WeatherService weatherService;

    @RequestMapping(value="/getwthrinfo", produces = "application/json")
    @ResponseBody
    public String getAjaxList2(String region) {
        List<WeatherVO> list = weatherService.getWthrInfo(region);
    }
}

```

```

        if(list != null) {
            Gson gson = new Gson();
            String jsonString = gson.toJson(list);
            return jsonString;
        }
        return "fail";
    }
}

```

5. 결과

날씨 정보 확인하기

서울 ▼	지역 선택				
서울	짜	최저기온 °C	최고기온 °C	SKY+PTY(날씨)	POP(강수확률)
부산					
대구	11-18	-2	7	맑음	0
인천					
광주	11-19	0	10	구름많음	20
대전					
울산	11-20	2	11	구름많음	20
경기	11-21	6	14	맑음	20
강원					
충북	11-22	2	9	맑음	10
충남					
전북	11-23	1	10	맑음	10
전남	11-24	3	12	구름많음	20
경북					
경남	11-25	4	14	흐림	40
제주					
	2024-11-26	7	11	흐림	40
	2024-11-27	3	8	흐림	40
	2024-11-28	0	7	맑음	20



마치며...

xml 파일을 직접 index를 찾아서 파싱하는 방법으로 구현했지만 JSON을 활용한다면 더 편리하게 했을 것 같다는 생각이 많이 들었다. 기회가 된다면 해당 코드를 JSON version으로 구현해보고 싶다.