날씨 정보 API 사용

1. 단기예보

→ 정보 제공 시간

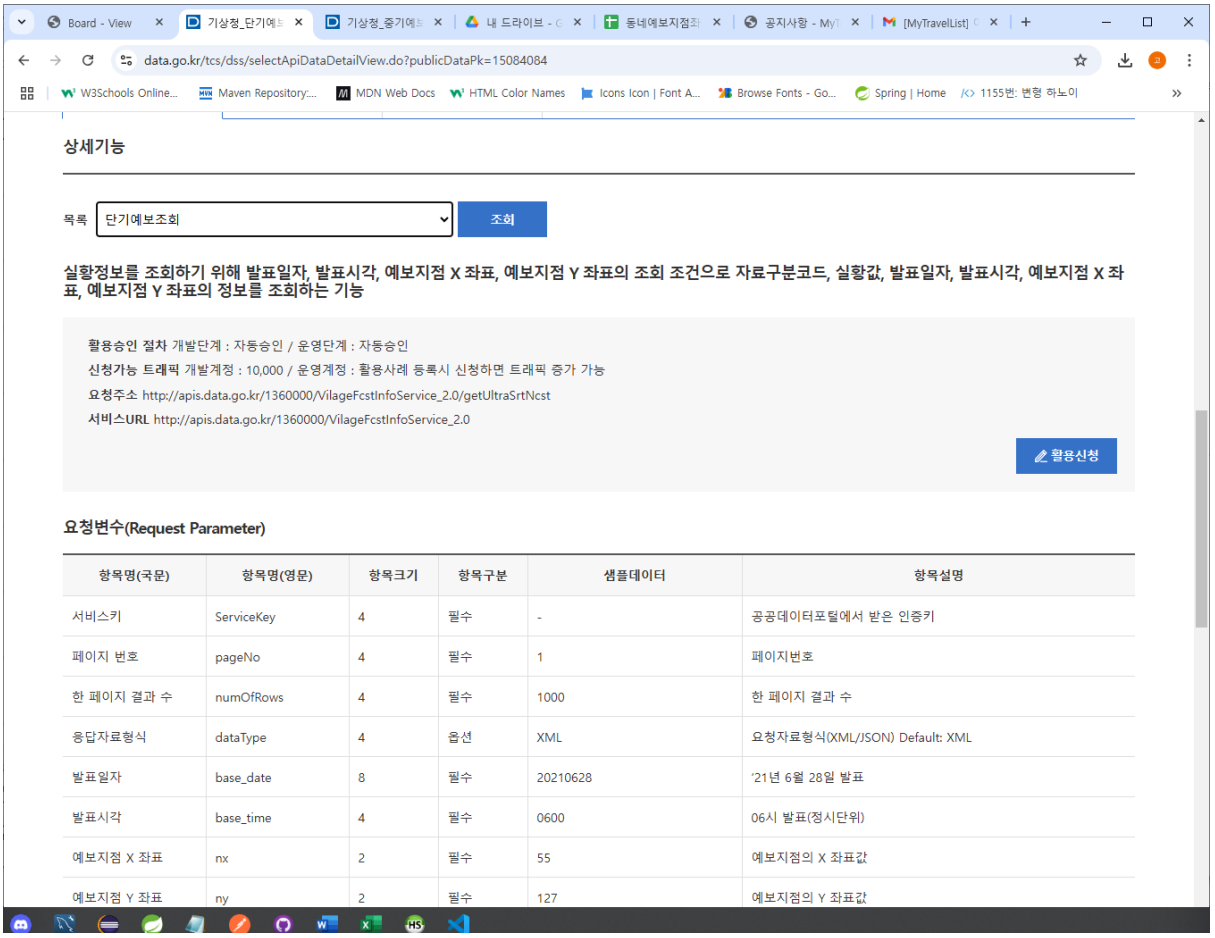
❍단기예보

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

⇒ 단기예보조회 사용



⇒ 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;
```

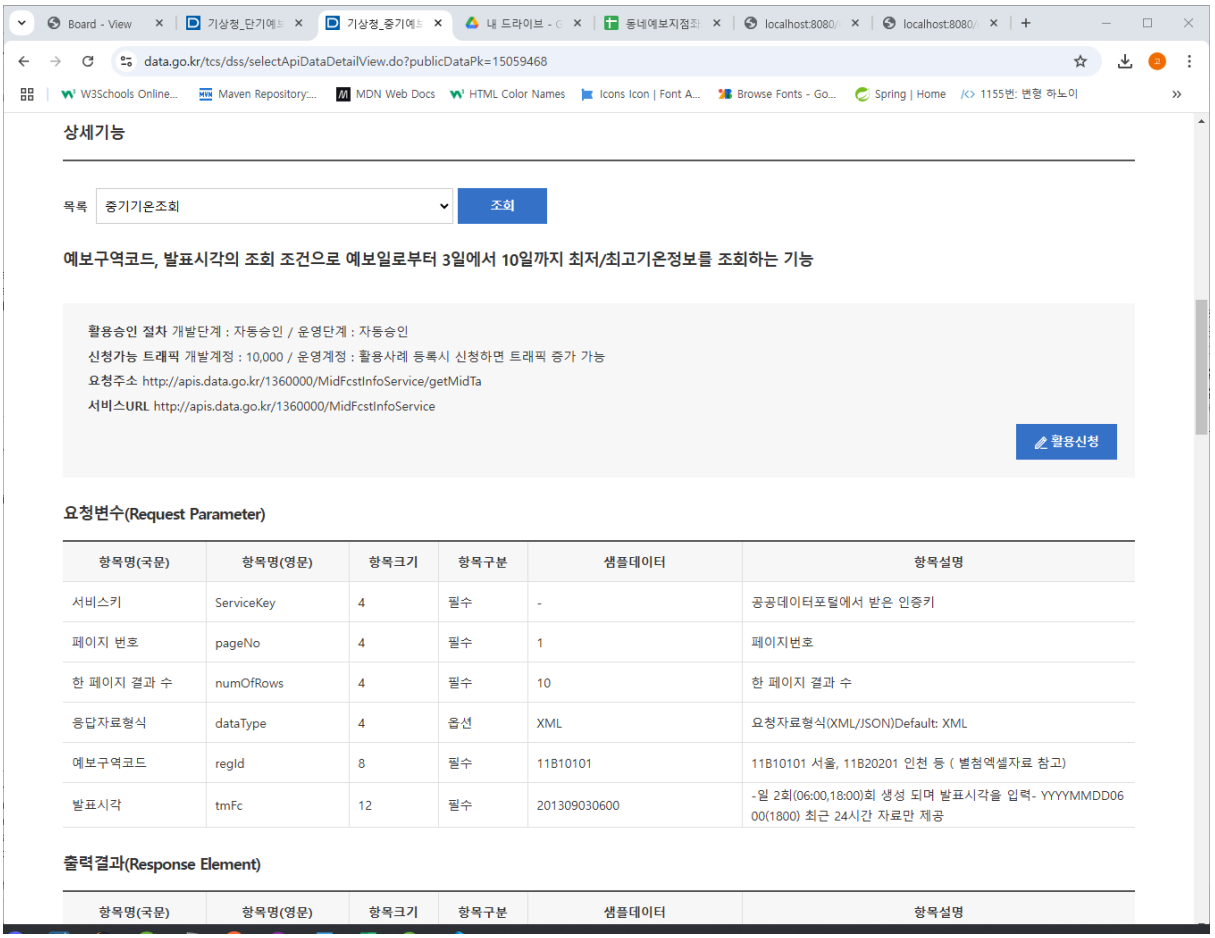
```
import java.net.URL;
import java.net.URLEncoder;
import java.io.BufferedReader;
import java.io.IOException;

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());
    }
}
```

short_term_weather.xml

→ 여기에는 최고기온, 최저기온, 강수확률, 날씨정보가 다 있음

2. 중기 예보

- 중기기온조회



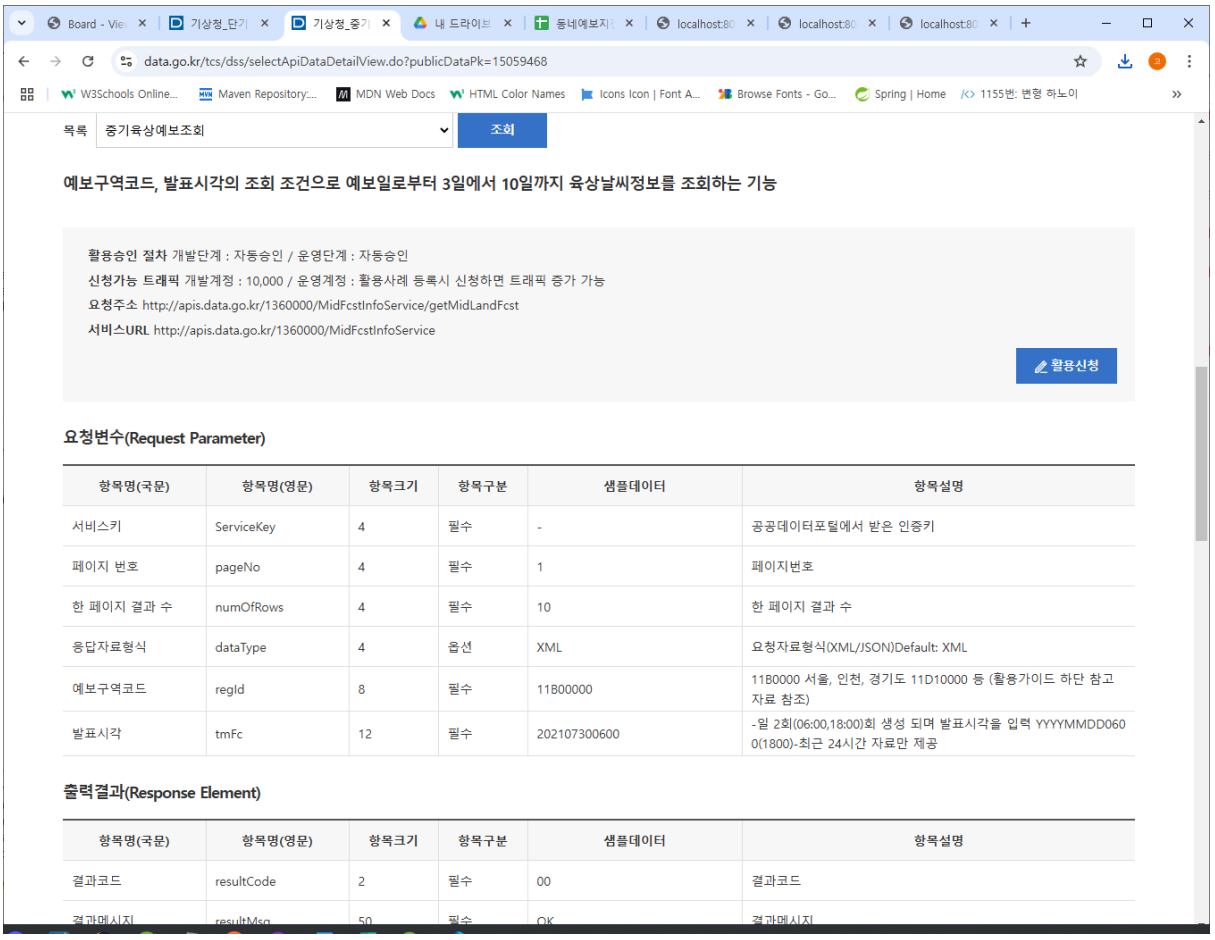

```
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.io.BufferedReader;
import java.io.IOException;
```

```
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("regId","UT
        urlBuilder.append("&" + URLEncoder.encode("tmFc","UTF
        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());
    }
}
```

long_term_ta.xml

→ 기온 정보만 있음

- 중기육상예보조회




```
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("regId","UT
```

```
        urlBuilder.append("&" + URLEncoder.encode("tmFc","UTF
        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());
    }
}
```

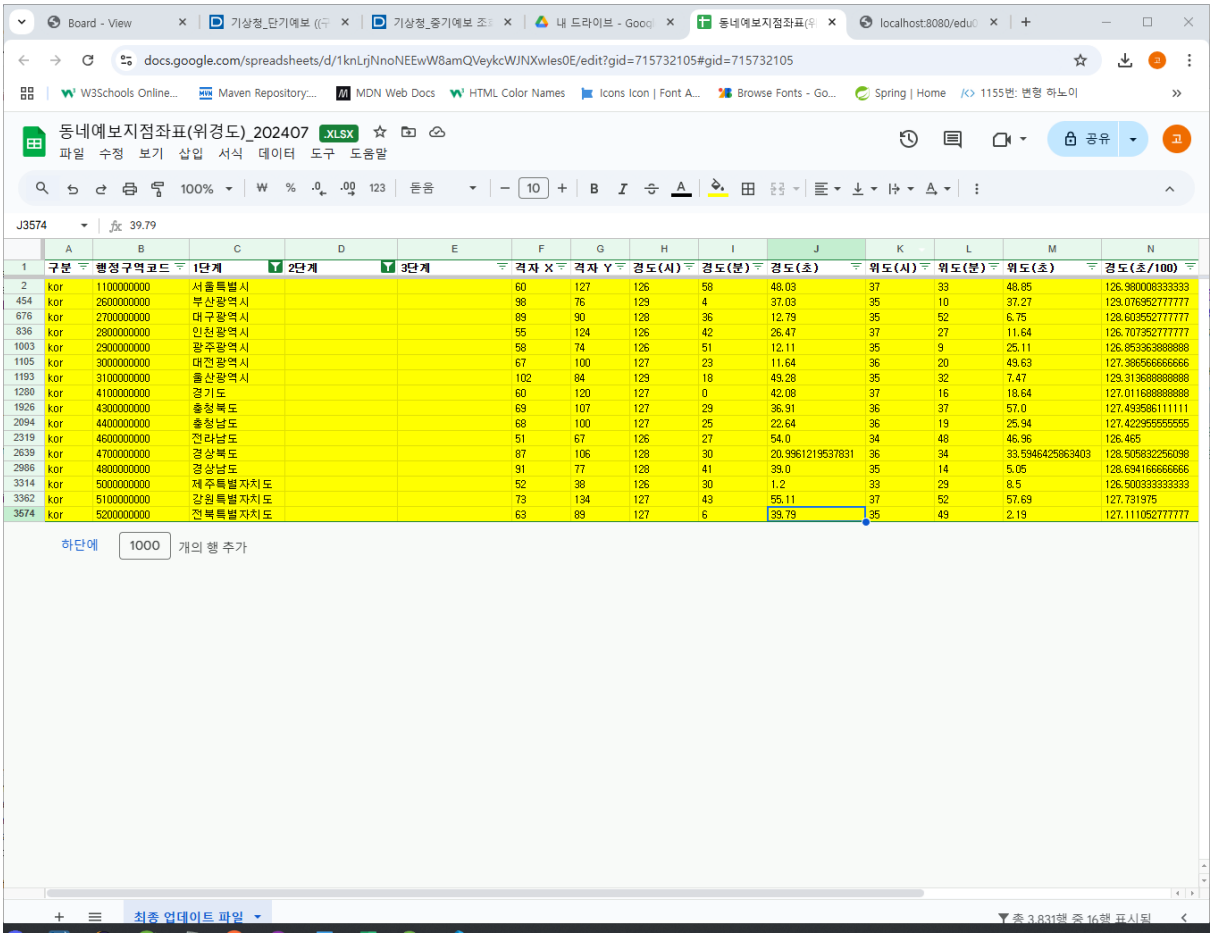
long_term_info.xml

→ 강수확률, 날씨에 대한 정보

→ DB 테이블 생성

```
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`)
);
CREATE TABLE `regioninfo` (
  `region` int NOT NULL,
  `reg_id_short` varchar(45) NOT NULL,
  `reg_name` varchar(45) NOT NULL,
  `nx` int NOT NULL,
  `ny` int NOT NULL,
  `reg_id_long` varchar(45) NOT NULL,
  PRIMARY KEY (`region`)
);
INSERT INTO regioninfo (`region`,`reg_id_short`,`reg_name`,`n
INSERT INTO regioninfo (`region`,`reg_id_short`,`reg_name`,`n
INSERT INTO regioninfo (`region`,`reg_id_short`,`reg_name`,`n
INSERT INTO regioninfo (`region`,`reg_id_short`,`reg_name`,`n
INSERT INTO regioninfo (`region`,`reg_id_short`,`reg_name`,`n
INSERT INTO regioninfo (`region`,`reg_id_short`,`reg_name`,`n
INSERT INTO regioninfo (`region`,`reg_id_short`,`reg_name`,`n
INSERT INTO regioninfo (`region`,`reg_id_short`,`reg_name`,`n
INSERT INTO regioninfo (`region`,`reg_id_short`,`reg_name`,`n
INSERT INTO regioninfo (`region`,`reg_id_short`,`reg_name`,`n
INSERT INTO regioninfo (`region`,`reg_id_short`,`reg_name`,`n
INSERT INTO regioninfo (`region`,`reg_id_short`,`reg_name`,`n
INSERT INTO regioninfo (`region`,`reg_id_short`,`reg_name`,`n
INSERT INTO regioninfo (`region`,`reg_id_short`,`reg_name`,`n
INSERT INTO regioninfo (`region`,`reg_id_short`,`reg_name`,`n
INSERT INTO regioninfo (`region`,`reg_id_short`,`reg_name`,`n
```



동네예보지점좌표(위경도)_202407 .XLSX

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	구분	행정구역코드	1단계	2단계	3단계	격자 X	격자 Y	경도(시)	경도(분)	경도(초)	위도(시)	위도(분)	위도(초)	경도(초/100)
2	kor	1100000000	서울특별시			60	127	126	58	48.03	37	33	48.85	126.980008333333
454	kor	2600000000	부산광역시			98	76	129	4	37.03	35	10	37.27	129.076952777777
676	kor	2700000000	대구광역시			89	90	128	36	12.79	35	52	6.75	128.603552777777
836	kor	2800000000	인천광역시			55	124	126	42	26.47	37	27	11.64	126.707352777777
1003	kor	2900000000	광주광역시			58	74	126	51	12.11	35	9	25.11	126.853363888888
1105	kor	3000000000	대전광역시			67	100	127	23	11.64	36	20	49.63	127.386566666666
1193	kor	3100000000	울산광역시			102	84	129	18	49.28	35	32	7.47	129.313688888888
1280	kor	4100000000	경기도			60	120	127	0	42.08	37	16	18.64	127.011688888888
1926	kor	4300000000	충청북도			69	107	127	29	36.91	36	37	57.0	127.493586111111
2094	kor	4400000000	충청남도			68	100	127	25	22.64	36	19	25.94	127.422955555555
2319	kor	4600000000	전라남도			51	67	126	27	54.0	34	48	46.96	126.465
2639	kor	4700000000	경상북도			87	106	128	30	20.9961219537831	36	34	33.5946425863403	128.505832256098
2986	kor	4800000000	경상남도			91	77	128	41	39.0	35	14	5.05	128.684166666666
3314	kor	5000000000	제주특별자치도			52	38	126	30	1.2	33	29	8.5	126.500333333333
3362	kor	5100000000	강원특별자치도			73	134	127	43	55.11	37	52	57.69	127.731975
3574	kor	5200000000	전북특별자치도			63	89	127	6	39.79	35	49	2.19	127.111052777777

→ WeatherVO.java

```
public class WeatherVO {
    private String wthrDate, wthrTMin, wthrTMax, wthrSKY_PTY,

// 아래는 getter & setter
```

→ RegionVO

```
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,
            String wthrDate = shorts.substring(tMinIdx + 24,
                    + 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 += "구름많음";
                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;
        HttpURLConnection conn = null;
```

```
        StringBuilder sb = null;

        // 단기 예보
        try {
            StringBuilder urlBuilder = new StringBuilder(
                    "http://apis.data.go.kr/1360000/VilageFcs
            urlBuilder.append("?" + URLEncoder.encode("servic
                    + "=Phsud7RN2nkw6wPmg2Fa7q%2BQZbDH%2Bnpp3

            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 = (HttpURLConnection) 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/1360000/MidFcstIn
            urlBuilder.append("?" + URLEncoder.encode("servic
                    + "=Phsud7RN2nkw6wPmg2Fa7q%2BQZbDH%2Bnpp3



            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"
            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/1360000/MidFcstIn
            urlBuilder.append("?" + URLEncoder.encode("servic
                    + "=Phsud7RN2nkw6wPmg2Fa7q%2BQZbDH%2Bnpp3



            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 생성 되며 발표시각을 입력 YYYY
```

```
                    urlBuilder.toString());
            conn = (HttpURLConnection) url.openConnection();
            conn.setRequestMethod("GET");
            System.out.println("Response code3: " + 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
                    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.getregin
```

```
    }

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

    @Override
    public int deleteWthrInfo() {
        return sqlSessionTemplate.delete("reginfo.delete");
    }
        @Override
    public List<WeatherVO> getWthrInfo(String region) {
        return sqlSessionTemplate.selectList("reginfo.getwhtr
    }
```

→ 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
        select * from regioninfo where region = #{region}
    </select>

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

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

    <select id="getwhtrinfo" parameterType="String" resultTyp
        select * from pjweather where region = #{region}
```

```
    </select>
</mapper>
```

→ 확인용 jsp

```
<div>
        <table id="list">
            <thead>
                <tr><th>날짜</th><th>최저기온 ℃</th><th>최고기온
            </thead>
            <tbody id="tbody">

            </tbody>
        </table>
    </div>

    <script type="text/javascript">
    function load(){
        $("#tbody").empty();
        $.ajax({
            url : "/test01",
            method : "post",
            data : "region="+$("#region").val(),
            dataType : "json",
            success : function(data){
                let tbody = "";
                $.each(data, function(index, obj){
                    tbody += "<tr>";
                    tbody += "<td>" + obj.wthrDate +"</td>"
                    tbody += "<td>" + obj.wthrTMin +"</td>"
                    tbody += "<td>" + obj.wthrTMax +"</td>"
                    tbody += "<td>" + obj.wthrSKY_PTY +"</td>
                    tbody += "<td>" + obj.wthrPOP +"</td>"
                    tbody += "<td>" +"아직" +"</td>"
                    tbody += "</tr>"
                });
                $("#tbody").append(tbody);
            },
```

```
            error : function(){
                alert("가져오기 실패에요")
            }
        })
    }


    </script>
```

→ restcontroller

```
@RestController
public class WeatherAjaxController {

    @Autowired
    private WeatherService weatherService;

    @RequestMapping(value="/getwthrinfo", produces = "applica
    @ResponseBody
    public String getAjaxList2(String region) {
        List<WeatherVO> list = weatherService.getWthrInfo(reg
        if(list != null) {
            Gson gson = new Gson();
            String jsonString = gson.toJson(list);
            return jsonString;
        }
        return "fail";
    }



}
```