기술문서

Android Studio 에서 OpenAPI(기상청_단기예보 ((구)_동네예보) 조회 서비스) 의 사용.

Android Studio Library

- viewBinding: 효율적인 컴포넌트 탐색.
- retrofit2: API 통신 라이브러리.

  Build.gradle: dependencies 추가

  1. implementation 'com.squareup.retrofit2:retrofit:2.8.1'
  2. implementation 'com.squareup.retrofit2:converter-gson:2.8.1'

  AndroidManifest.xml: permission 추가

  1. <uses-permission android:name="android.permission.INTERNET" />
  2. <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

Android Studio File

- Class
  1. MainActivity.kt
  2. WeatherAdapter
  3. WeatherInfo
- Interface
  1. WeatherInterface
- Layout
  1. activity_main.xml
  2. list_item_weather.xml

OpenAPI(기상청_단기예보 ((구)_동네예보) 조회서비스)

- REST API
- JSON+XML 데이터포맷.
- API 인증키 발급.(DATA.GO.KR 개인 계정)

일반 인증키 (Encoding)	75U3aIZ9	ıdXtra%2FxxN	3ZetrwsLuRzXxC(	AsuSTKg6EA%3D%3D
일반 인증키 (Decoding)	75U3aIZ9(	V2irSgtd9yTiJ2hdXtra/	ZetrwsLuRzXxCdUnxEU(	==

- 30분 마다 갱신 및 발표되는 초 단기 날씨 예측 사용.

Android Studio File Class

MainActivity.kt

- 데이터 요청 및 데이터 수신 관련 작업 수행.
- RecyclerView 와 연결 작업 수행.
- onCreate()

  viewBinding 변수 선언 및 정의.

  RecyclerView Manager 설정

- setWeather()

  OpenAPI 를 사용 하기 위한 변수 정의.(행렬, 해당 페이지 번호, 데이터 타입, 날짜, 시간, 위도, 경도)

  서버와 통신 및 데이터 수신.(강수 확률, 강수 형태, 1시간당 기온, 습도, 풍속 데이터를 수신)

```
// 날씨 정보 가져오기
// (한 페이지 결과 수 = 60, 페이지 번호 = 1, 응답 자료 형식-"JSON", 발표 날짜, 발표 시각, 예보지점 좌표)
val call = ApiObject.retrofitService.GetWeather( num_of_rows: 60,  page_no: 1,  data_type: "JSON", base_date, base_time, nx, ny)
```

  데이터 저장 및 RecyclerView 와 연결.

- Fun getBaseTime()

  날씨 데이터는 00분 30분 기준으로 기상청에서 발표되기 때문에 현재 시간(분)에 관련하여 데이터를 수신할 때 간단한 논리를 필요로 함.

WeatherAdapter

- RecyclerView에 속하는 개별 ItemView를 설정.
- 수신 받은 데이터 형식 변환.

 # 특정 요소의 코드값 및 범주

- 하늘상태(SKY) 코드 : 맑음(1), 구름많음(3), 흐림(4)
- 강수형태(PTY) 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)

- onCreateViewHolder()

  ViewHolder 생성 및 반환.

- onBindViewHolder()

  ViewHolder 와 Item 연결.

- inner class ViewHolder()

  RecyclerView 에 연결 될 Item 설정.

  Item 설정은 수신 받은 데이터 를 토대로 작성됨.(강수 확률, 강수 형태, 1시간당 기온, 습도, 풍속 데이터를 토대로 작성)

- TranslateRainType()

  수신 받은 데이터 형식 변환.(데이터가 이용자가 보기 편하도록 보정.)

```
// RainType
fun TranslateRainType(rainType : String) : String {
    when(rainType) {
        "0" -> return "맑음"
        "1" -> return "비"
        "2" -> return "비/눈"
        "3" -> return "눈"
        else -> return "강수 형태 오류"
    }
}
```

WeatherInterface

- retrofit2 Library 를 사용하여 서버에 데이터를 요청하는 인터페이스.

```
interface WeatherInterface {
    // getUltraSrtFcst : 초단기 예보 조회 + 인증키
    @GET("getUltraSrtFcst?serviceKey=                    /W2irSgtd9yTiJ2hdXtra%2FxxNbNrKasu303J                              %3D%3D")

    fun GetWeather(@Query("numOfRows") num_of_rows : Int,   // 한 페이지 경과 수
                   @Query("pageNo") page_no : Int,          // 페이지 번호
                   @Query("dataType") data_type : String,   // 응답 자료 형식
                   @Query("base_date") base_date : String,  // 발표 일자
                   @Query("base_time") base_time : String,  // 발표 시각
                   @Query("nx") nx : String,                // 예보지점 X 좌표
                   @Query("ny") ny : String)                // 예보지점 Y 좌표
            : Call<WEATHER>
}
```

- 서버에 데이터를 요청하는 방식은 http 웹 통신의 일종.
- 허가된 사용자만 데이터 요청 가능.(API 보안 KEY 존재)
- http 웹 통신은 header, body 등 다양한 요소로 구성됨. 따라서 다양한 data Class 를 구현.

```
data class WEATHER (val response : RESPONSE)
data class RESPONSE(val header : HEADER, val body : BODY)
data class HEADER(val resultCode : Int, val resultMsg : String)
data class BODY(val dataType : String, val items : ITEMS, val totalCount : Int)
data class ITEMS(val item : List<ITEM>)
// category : 자료 구분 코드, fcstDate : 예측 날짜, fcstTime : 예측 시간, fcstValue : 예보 값
data class ITEM(val category : String, val fcstDate : String, val fcstTime : String, val fcstValue : String)
```

WeatherInfo

- 수신 받을 데이터의 형식. 일종의 Data Class