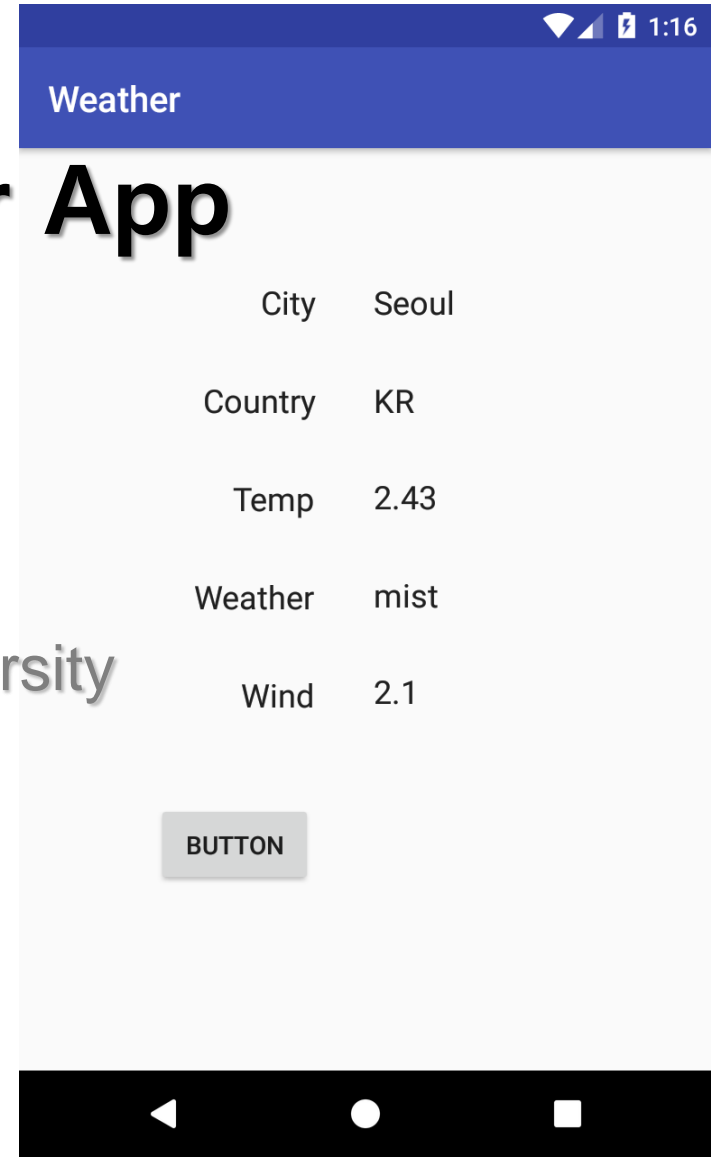


# Making Mobile Apps

## Lec 22. Weather App



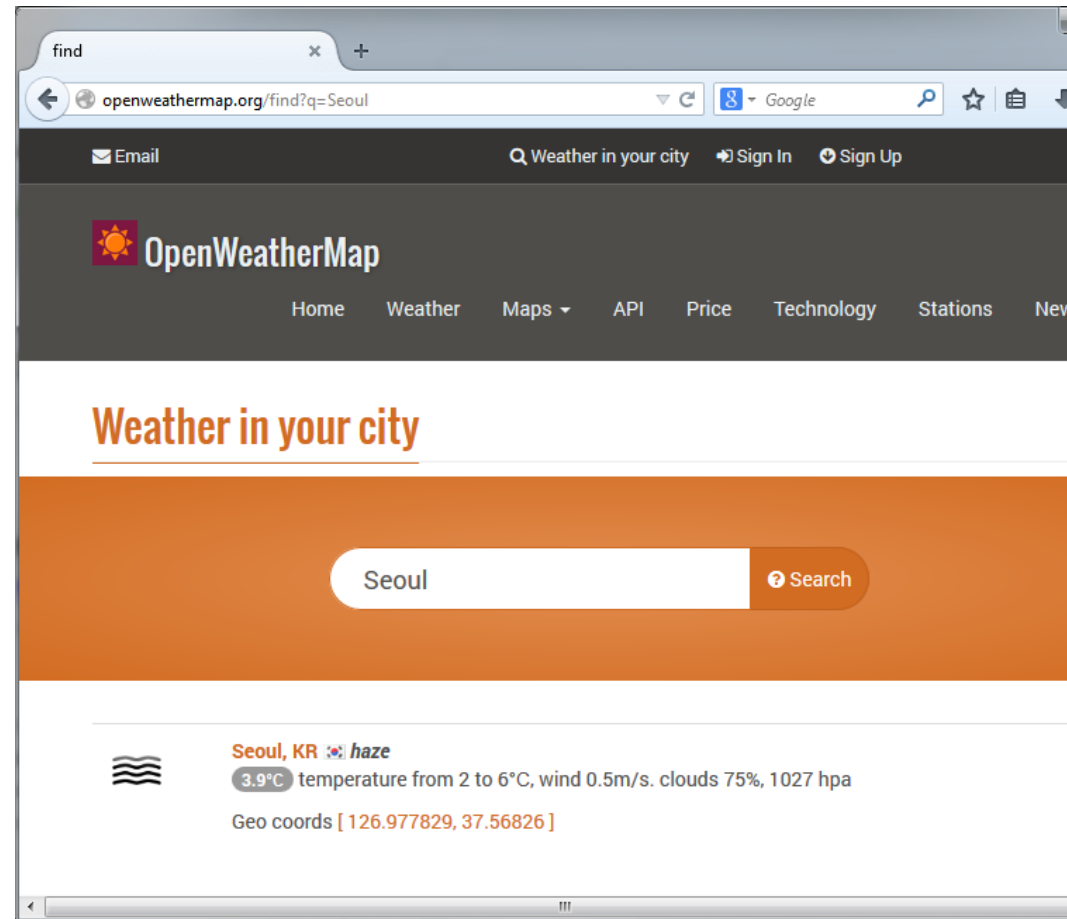
Ewha Womans University



# Simple Weather App

2

- Get weather data from a cloud service



# Free service for low volume (2015)

3

Weather API Price-list

www.openweathermap.org/price


OpenWeatherMap

Home Weather Maps API Price Technology Stations News

	Free	Developer	Professional	Enterprise
<b>How many requests does your application make?</b>				
Calls per minute (no more than)	3,000	30,000	60,000	200,000
Calls per day (no more than)	4,000,000	40,000,000	70,000,000	200,000,000
<b>What type of weather data do you need*?</b>				
Current conditions	✓	✓	✓	✓
Weather forecasts	✓	✓	✓	✓
Weather maps	✓	✓	✓	✓
Sunrise, sunset	✓	✓	✓	✓

# Free service for low volume (now)

4



The screenshot shows a web browser window with the title 'Weather API Price-list'. The address bar shows 'openweathermap.org/price'. The page features a large orange heading 'Current weather and forecasts collection'. Below this is a table comparing different API pricing tiers: Free, Startup, Developer, Professional, and Enterprise. The 'Free' tier is highlighted as the current service for low volume.

	Free	Startup	Developer	Professional	Enterprise
<b>Price</b> Price is fixed, no other hidden costs.	<b>Free</b>	40 USD / month	180 USD / month	470 USD / month	2,000 USD / month
<b>Subscribe</b>	<a href="#">Get API key and Start</a>	<a href="#">Subscribe</a>	<a href="#">Subscribe</a>	<a href="#">Subscribe</a>	<a href="#">Subscribe</a>
Calls per minute (no more than)	60	600	3,000	30,000	200,000
Current weather API	✓	✓	✓	✓	✓
5 days/3 hour forecast API	✓	✓	✓	✓	✓
16 days/daily forecast API	-	-	✓	✓	✓

## How to work with API key

To get access to weather API you need an API key whatever account you chose from Free to Enterprise.

### How to get API key (APPID)

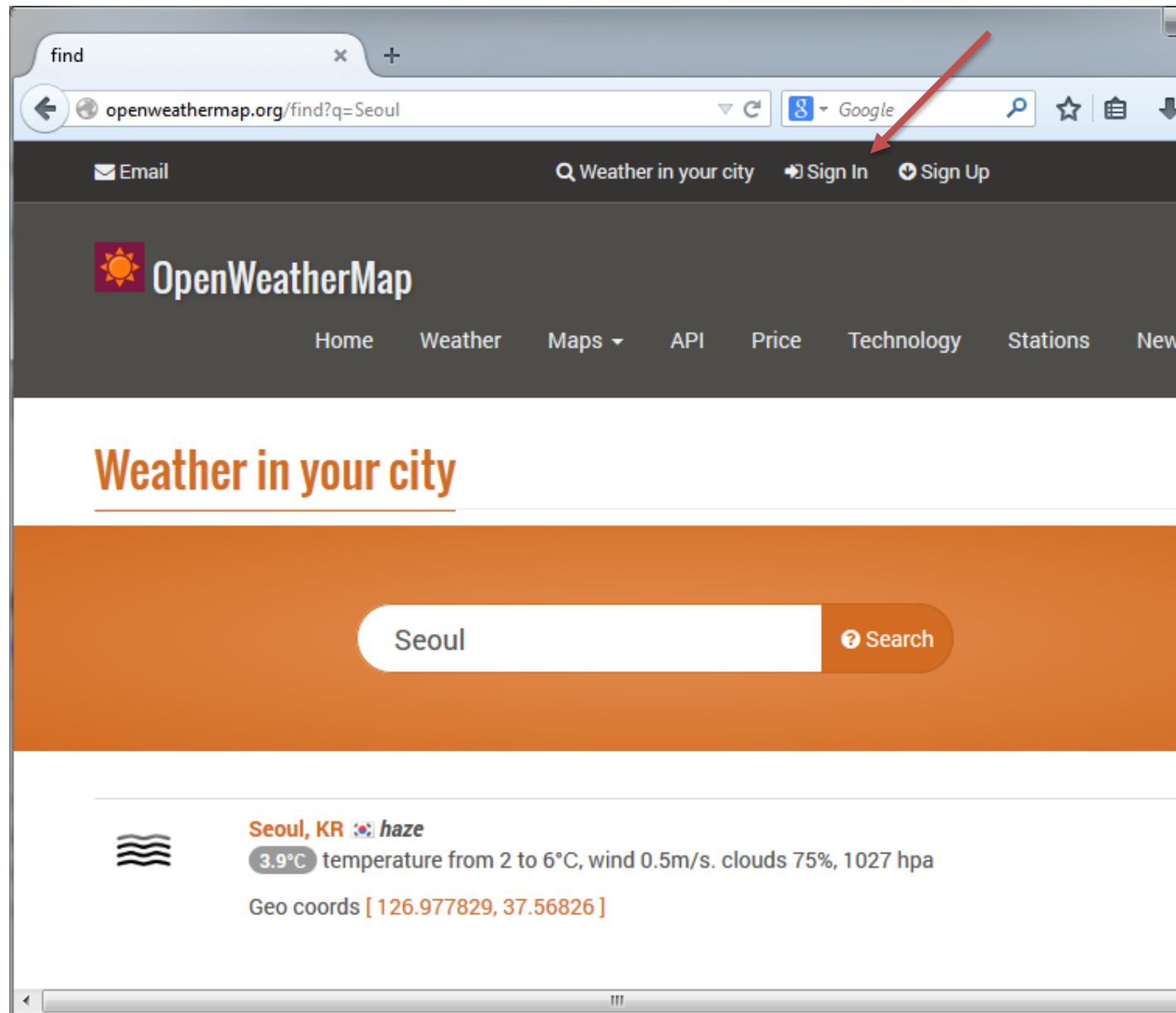
- 1 Register on the [Sign up](#) page
- 2 Get unique API key on your personal page

### How to use API key

- Add the following parameter to the GET request: APPID=APIKEY  
Example: [api.openweathermap.org/data/2.5/forecast/city?id=524901&APPID=1111111111](https://api.openweathermap.org/data/2.5/forecast/city?id=524901&APPID=1111111111)
- OR add the following line to http header of request to the server: x-api-key.APIKEY

# Get API key

6



The screenshot shows a web browser window with the URL `openweathermap.org/find?q=Seoul`. A red arrow points to the "Sign In" link in the top navigation bar. The page features the OpenWeatherMap logo and a search bar with "Seoul" entered. Below the search bar, the weather for Seoul, KR is displayed as "haze" with a temperature of 3.9°C. The page also includes a navigation menu with links to Home, Weather, Maps, API, Price, Technology, Stations, and News.

find x +

openweathermap.org/find?q=Seoul

Email Weather in your city Sign In Sign Up

OpenWeatherMap

Home Weather Maps API Price Technology Stations New

Weather in your city

Seoul Search

Seoul, KR haze


3.9°C temperature from 2 to 6°C, wind 0.5m/s. clouds 75%, 1027 hpa

Geo coords [ 126.977829, 37.56826 ]

# Get your API key

7

home.openweathermap.org Search

 OpenWeatherMap Home Weather Maps API Price Stations

**Username**

**Email**

**Full name**

---

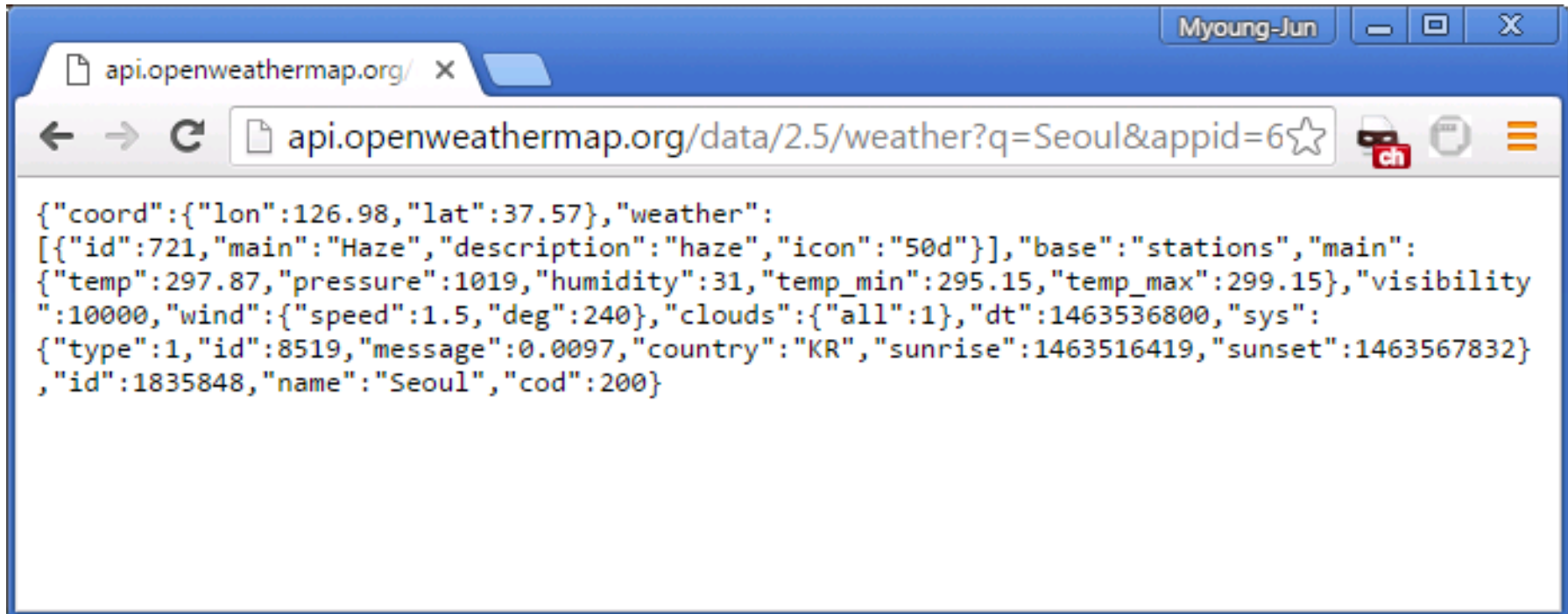
**API key**

# How it works

8

- [http request](#)
  - <http://api.openweathermap.org/data/2.5/weather?q=Seoul&appid=6af8dad4784c634d3674f60110f2a015>
- [JSON response](#)

API key



The screenshot shows a web browser window with the URL `api.openweathermap.org/data/2.5/weather?q=Seoul&appid=6af8dad4784c634d3674f60110f2a015`. The response is a JSON object containing weather data for Seoul, including coordinates, weather conditions (Haze), temperature, pressure, humidity, wind speed, and visibility.

```
{
  "coord": {
    "lon": 126.98,
    "lat": 37.57
  },
  "weather": [
    {
      "id": 721,
      "main": "Haze",
      "description": "haze",
      "icon": "50d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 297.87,
    "pressure": 1019,
    "humidity": 31,
    "temp_min": 295.15,
    "temp_max": 299.15
  },
  "visibility": 10000,
  "wind": {
    "speed": 1.5,
    "deg": 240
  },
  "clouds": {
    "all": 1
  },
  "dt": 1463536800,
  "sys": {
    "type": 1,
    "id": 8519,
    "message": 0.0097,
    "country": "KR",
    "sunrise": 1463516419,
    "sunset": 1463567832
  },
  "id": 1835848,
  "name": "Seoul",
  "cod": 200
}
```



# ***JSON (JavaScript Object Notation)***

```
{
  "coord":{
    "lon":127.02,
    "lat":37.52
  },
  "sys":{
    "message":0.2388,
    "country":"South Korea",
    "sunrise":1400703436,
    "sunset":1400755196
  },
  "weather":[
    {
      "id":800,
      "main":"Clear",
      "description":"Sky is Clear",
      "icon":"01d"
    }
  ],
  "base":"cmc stations",
```

```
    "main":{
      "temp":292.684,
      "temp_min":292.684,
      "temp_max":292.684,
      "pressure":1001.37,
      "sea_level":1024.8,
      "grnd_level":1001.37,
      "humidity":68
    },
    "wind":{
      "speed":1.66,
      "deg":228.001
    },
    "clouds":{
      "all":0
    },
    "dt":1400720147,
    "id":1897000,
    "name":"Seoul",
    "cod":200
  }
}
```

# JSON (JavaScript Object Notation)

10

object → {

```
"firstName": "John",  
"lastName": "Smith",  
"age": 25,
```

obj.firstName → John

obj.getString("firstName")

object →

```
"address": {  
  "streetAddress": "21 2nd Street",  
  "city": "New York",  
  "state": "NY",  
  "postalCode": 10021
```

obj.address.city → New York

obj.getJSONObject("address")  
 .getString("city")

array →

```
"phoneNumbers": [  
  {  
    "type": "home",  
    "number": "212 555-1234"  
  },  
  {  
    "type": "fax",  
    "number": "646 555-4567"  
  }  
]
```

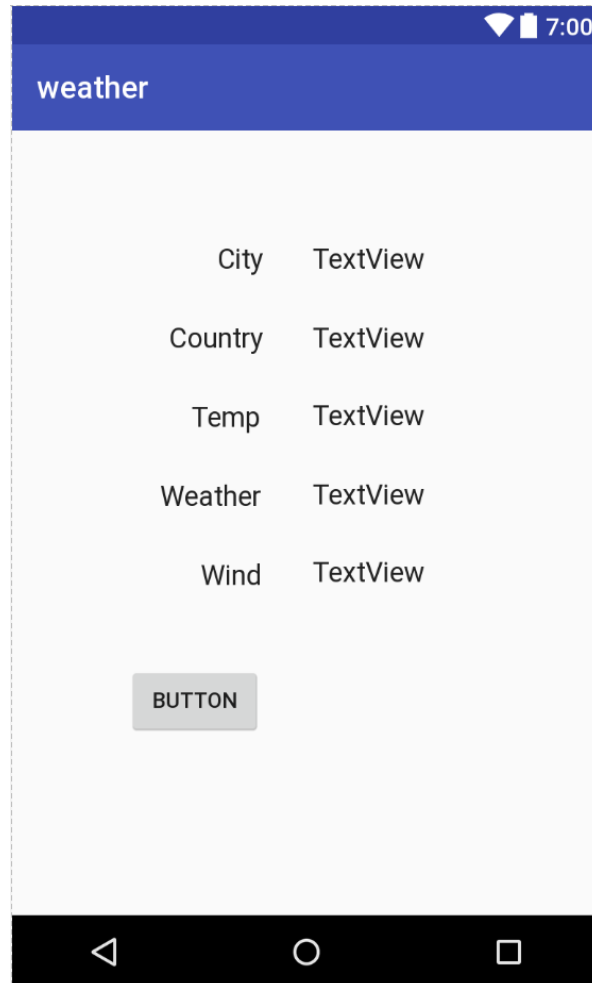
obj.phoneNumbers[0] →

{ "type": "home",  
 "number": "212 555-1234" }

obj.phoneNumbers[1].type → fax

```
}
```

obj.getJSONArray("phoneNumbers")  
 .getJSONObject(1).getString("type")



## Component Tree



### ▼ ConstraintLayout

Ab **textView** - "Wind"

Ab **textView5** - "Weather"

Ab **textView4** - "Temp"

Ab **textView3** - "Country"

Ab **textView2** - "City"

Ab **text\_wind** (TextView) - "Text"

Ab **text\_weather** (TextView) - "T"

Ab **text\_temp** (TextView) - "Text"

Ab **text\_country** (TextView) - "T"

Ab **text\_city** (TextView) - "TextVi"

OK **button** - "Button"

# We need Anko library

12



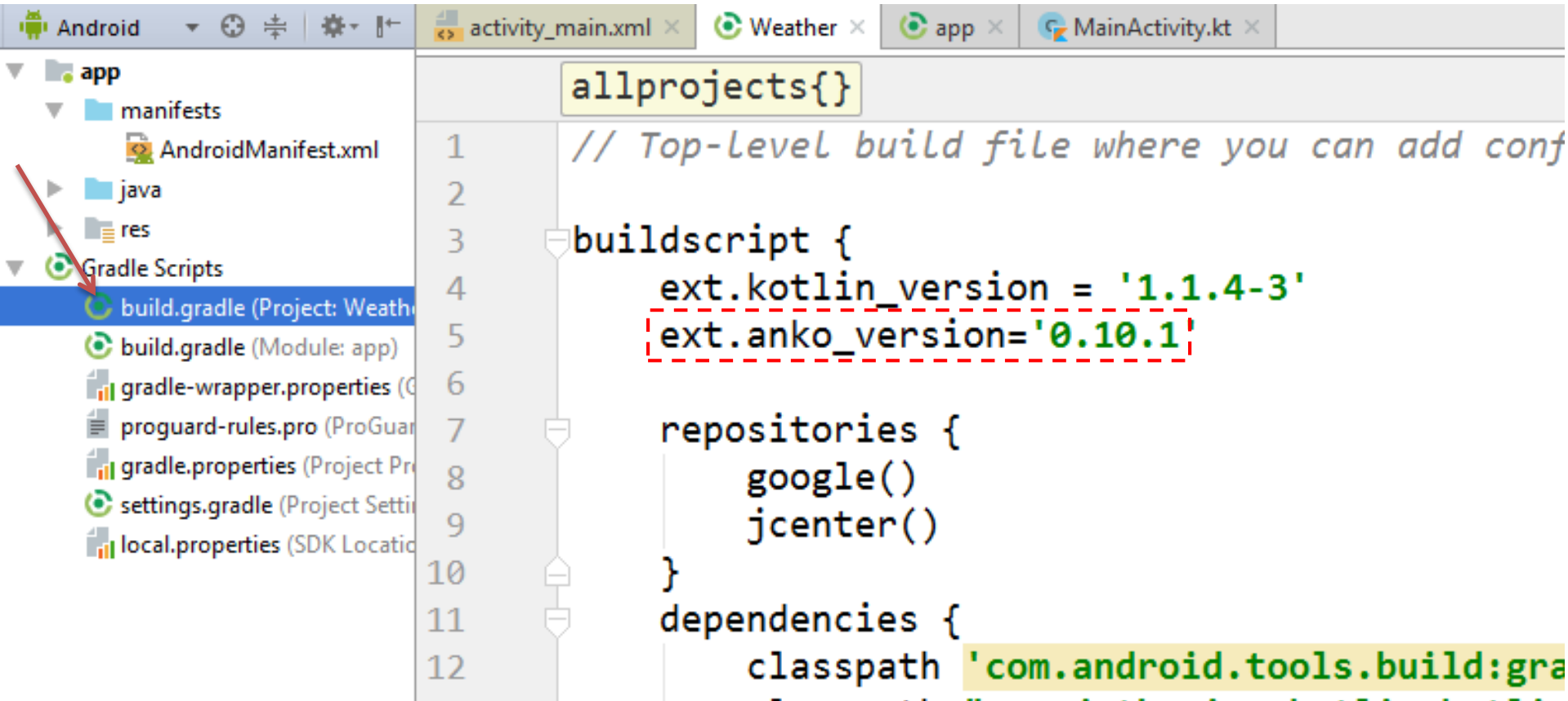
The screenshot shows the Android Studio IDE with the following components:

- Left Sidebar (Project View):** Shows the project structure. The 'app' module is expanded, and 'build.gradle (Module: app)' is selected. A red arrow points to this file.
- Top Tabs:** activity\_main.xml, Weather, app, MainActivity.kt.
- Main Editor:** Displays the content of build.gradle (Module: app). The code is as follows:

```
20 minifyEnabled false
21 proguardFiles getDefaultProguardFile('proguard-android-optimize.txt')
22 }
23 }
24 }
25
26 dependencies {
27     implementation fileTree(dir: 'libs', include: ['*.jar'])
28     implementation 'com.android.support:appcompat-v7:26.1.0'
29     implementation 'com.android.support.constraint:constraint-layout:1.0.2'
30     testImplementation 'junit:junit:4.12'
31     androidTestImplementation ('com.android.support.test.espresso:espresso-core:2.2.2')
32         exclude group: 'com.android.support', module: 'support-annotations'
33 }
34 implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
35 compile "org.jetbrains.anko:anko:$anko_version"
36 }
37
```

# We need Anko library

13



The screenshot shows an IDE interface with the following components:

- Left Sidebar (Project Explorer):** Displays the project structure. The 'app' folder is expanded, showing 'manifests' (containing 'AndroidManifest.xml'), 'java', 'res', and 'Gradle Scripts'. Under 'Gradle Scripts', 'build.gradle (Project: Weather)' is selected and highlighted with a red arrow.
- Top Tab Bar:** Shows open files: 'activity\_main.xml', 'Weather', 'app', and 'MainActivity.kt'.
- Main Editor:** Displays the 'build.gradle' file for the 'Weather' project. The file content is as follows:

```
allprojects{  
    // Top-level build file where you can add conf  
  
    buildscript {  
        ext.kotlin_version = '1.1.4-3'  
        ext.anko_version='0.10.1'  
    }  
  
    repositories {  
        google()  
        jcenter()  
    }  
  
    dependencies {  
        classpath 'com.android.tools.build:gra'
```

```
class MainActivity : AppCompatActivity() {
```

```
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
    }
```

```
    val url = "http://api.openweathermap.org/data/2.5/weather?" +  
        "q=Seoul&units=metric&appid=6af8dad4784c634d3674f60110f2a015"
```

```
    fun get_weather(view : View) {  
        doAsync {  
            val result = URL(url).readText()  
            uiThread {  
                Log.d("json", result)  
                LongToast("Request performed")  
  
                var json = JSONObject(result)  
  
                val city = json.getString("name")  
                val country = json.getJSONObject("sys").getString("country")  
                val temp = json.getJSONObject("main").getString("temp")  
                val weather = json.getJSONArray("weather").getJSONObject(0)  
                    .getString("description")  
                val wind = json.getJSONObject("wind").getString("speed")  
  
                text_city.text = city  
                text_country.text = country  
                text_temp.text = temp  
                text_weather.text = weather  
                text_wind.text = wind  
            }  
        }  
    }  
}
```

```
val url = "http://api.openweathermap.org/data/2.5/weather?" +  
          "q=Seoul&units=metric&appid=6af8dad4784c634d3674f60110f2a015"
```

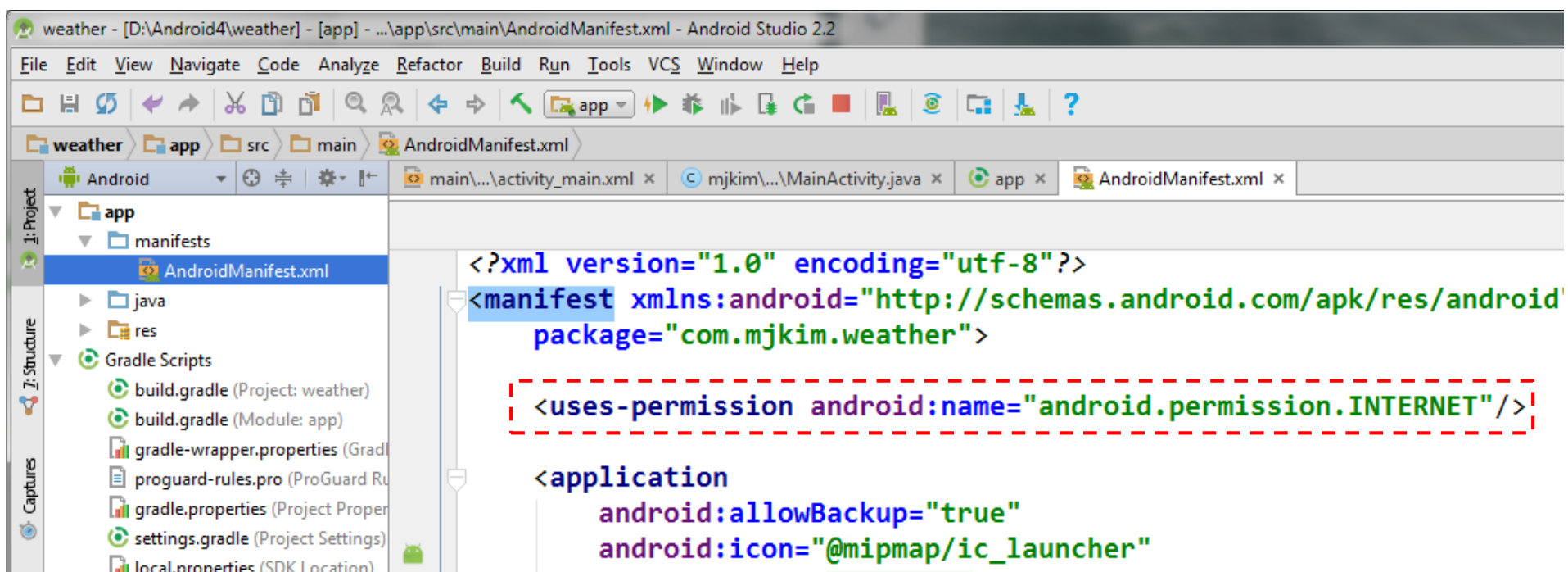
```
doAsync {  
    val result = URL(url).readText()  
    uiThread {  
        Log.d("json", result)  
        LongToast("Request performed")  
  
        var json = JSONObject(result)  
  
        val city = json.getString("name")  
        val country = json.getJSONObject("sys").getString("country")  
        val temp = json.getJSONObject("main").getString("temp")  
        val weather = json.getJSONArray("weather").getJSONObject(0)  
            .getString("description")  
        val wind = json.getJSONObject("wind").getString("speed")  
  
        text_city.text = city  
        text_country.text = country  
        text_temp.text = temp  
        text_weather.text = weather  
        text_wind.text = wind  
    }  
}
```



# Internet Permission

17

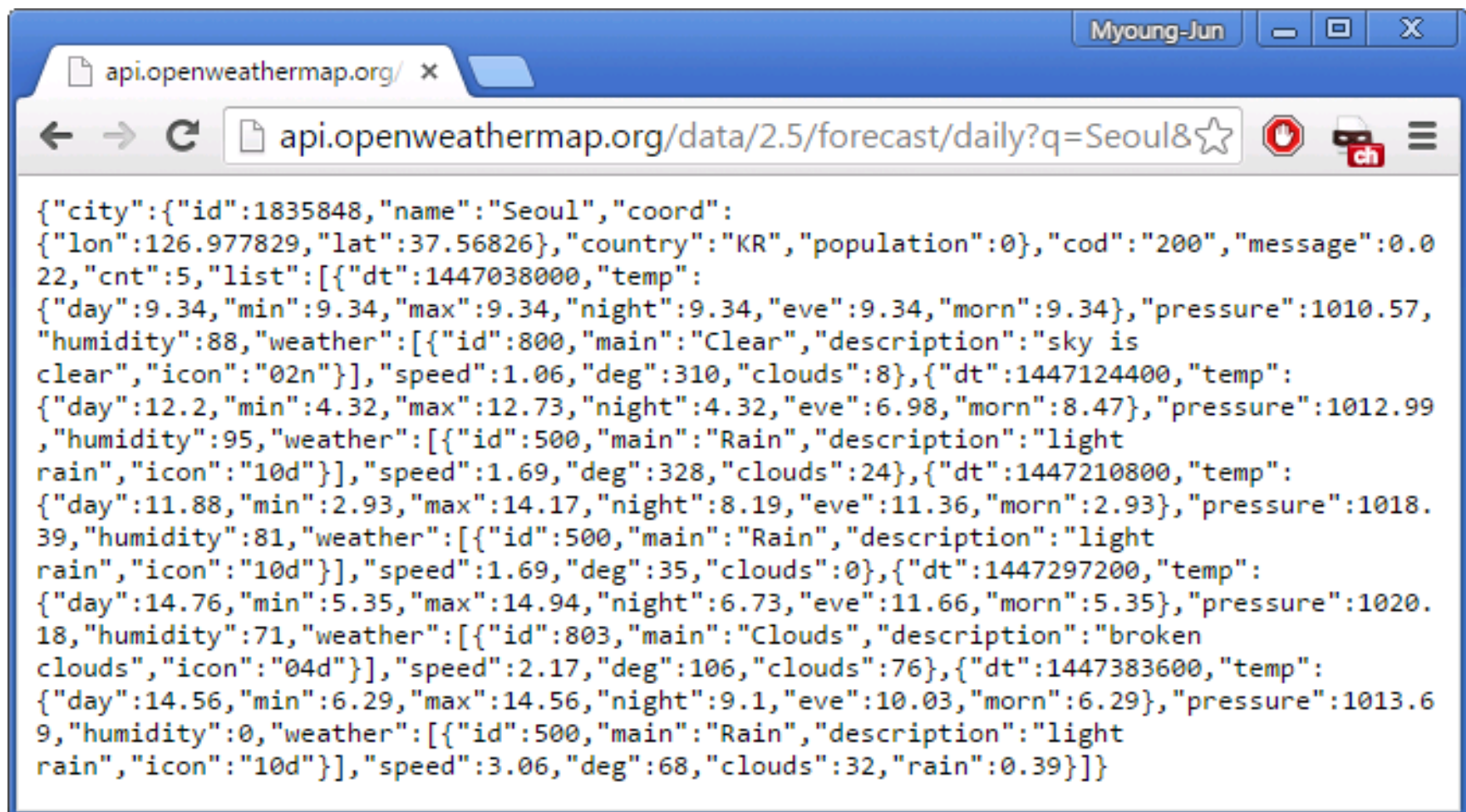
- Accessing Internet Needs a Permission



# Exercise

18

- Make a 5-day forecast
- <http://api.openweathermap.org/data/2.5/forecast/daily?q=Seoul&units=metric&cnt=5&appid=6af8dad4784c634d3674f60110f2a015>



The screenshot shows a web browser window with the address bar displaying the URL: `api.openweathermap.org/data/2.5/forecast/daily?q=Seoul&units=metric&cnt=5&appid=6af8dad4784c634d3674f60110f2a015`. The browser's address bar also shows a star icon for bookmarks and a red octagonal stop icon. The main content area of the browser displays the JSON response from the API, which includes city information (Seoul, KR) and a 5-day forecast with temperature, pressure, humidity, and weather conditions (Clear, Rain, Clouds).

```
{
  "city": {
    "id": 1835848,
    "name": "Seoul",
    "coord": {
      "lon": 126.977829,
      "lat": 37.56826
    },
    "country": "KR",
    "population": 0
  },
  "cod": "200",
  "message": 0.022,
  "cnt": 5,
  "list": [
    {
      "dt": 1447038000,
      "temp": {
        "day": 9.34,
        "min": 9.34,
        "max": 9.34,
        "night": 9.34,
        "eve": 9.34,
        "morn": 9.34
      },
      "pressure": 1010.57,
      "humidity": 88,
      "weather": [
        {
          "id": 800,
          "main": "Clear",
          "description": "sky is clear",
          "icon": "02n"
        }
      ],
      "speed": 1.06,
      "deg": 310,
      "clouds": 8
    },
    {
      "dt": 1447124400,
      "temp": {
        "day": 12.2,
        "min": 4.32,
        "max": 12.73,
        "night": 4.32,
        "eve": 6.98,
        "morn": 8.47
      },
      "pressure": 1012.99,
      "humidity": 95,
      "weather": [
        {
          "id": 500,
          "main": "Rain",
          "description": "light rain",
          "icon": "10d"
        }
      ],
      "speed": 1.69,
      "deg": 328,
      "clouds": 24
    },
    {
      "dt": 1447210800,
      "temp": {
        "day": 11.88,
        "min": 2.93,
        "max": 14.17,
        "night": 8.19,
        "eve": 11.36,
        "morn": 2.93
      },
      "pressure": 1018.39,
      "humidity": 81,
      "weather": [
        {
          "id": 500,
          "main": "Rain",
          "description": "light rain",
          "icon": "10d"
        }
      ],
      "speed": 1.69,
      "deg": 35,
      "clouds": 0
    },
    {
      "dt": 1447297200,
      "temp": {
        "day": 14.76,
        "min": 5.35,
        "max": 14.94,
        "night": 6.73,
        "eve": 11.66,
        "morn": 5.35
      },
      "pressure": 1020.18,
      "humidity": 71,
      "weather": [
        {
          "id": 803,
          "main": "Clouds",
          "description": "broken clouds",
          "icon": "04d"
        }
      ],
      "speed": 2.17,
      "deg": 106,
      "clouds": 76
    },
    {
      "dt": 1447383600,
      "temp": {
        "day": 14.56,
        "min": 6.29,
        "max": 14.56,
        "night": 9.1,
        "eve": 10.03,
        "morn": 6.29
      },
      "pressure": 1013.69,
      "humidity": 0,
      "weather": [
        {
          "id": 500,
          "main": "Rain",
          "description": "light rain",
          "icon": "10d"
        }
      ],
      "speed": 3.06,
      "deg": 68,
      "clouds": 32,
      "rain": 0.39
    }
  ]
}
```