IAT359 Mobile Computing

Fall 2022

Instructor: Hanieh Shakeri

TA: Parnian Taghipour

week 8 check-in

- Quiz 3 today **(due tomorrow at 2:20PM)**
- Lab 8 (participation) – **due Friday at midnight**
- Milestone 2 **due Nov 22**
- About 1 month left for research study participation (bonus marks)
- Next week (Week 9) is our last instructional workshop – after that, we will move to weekly team check-ins during workshop time.

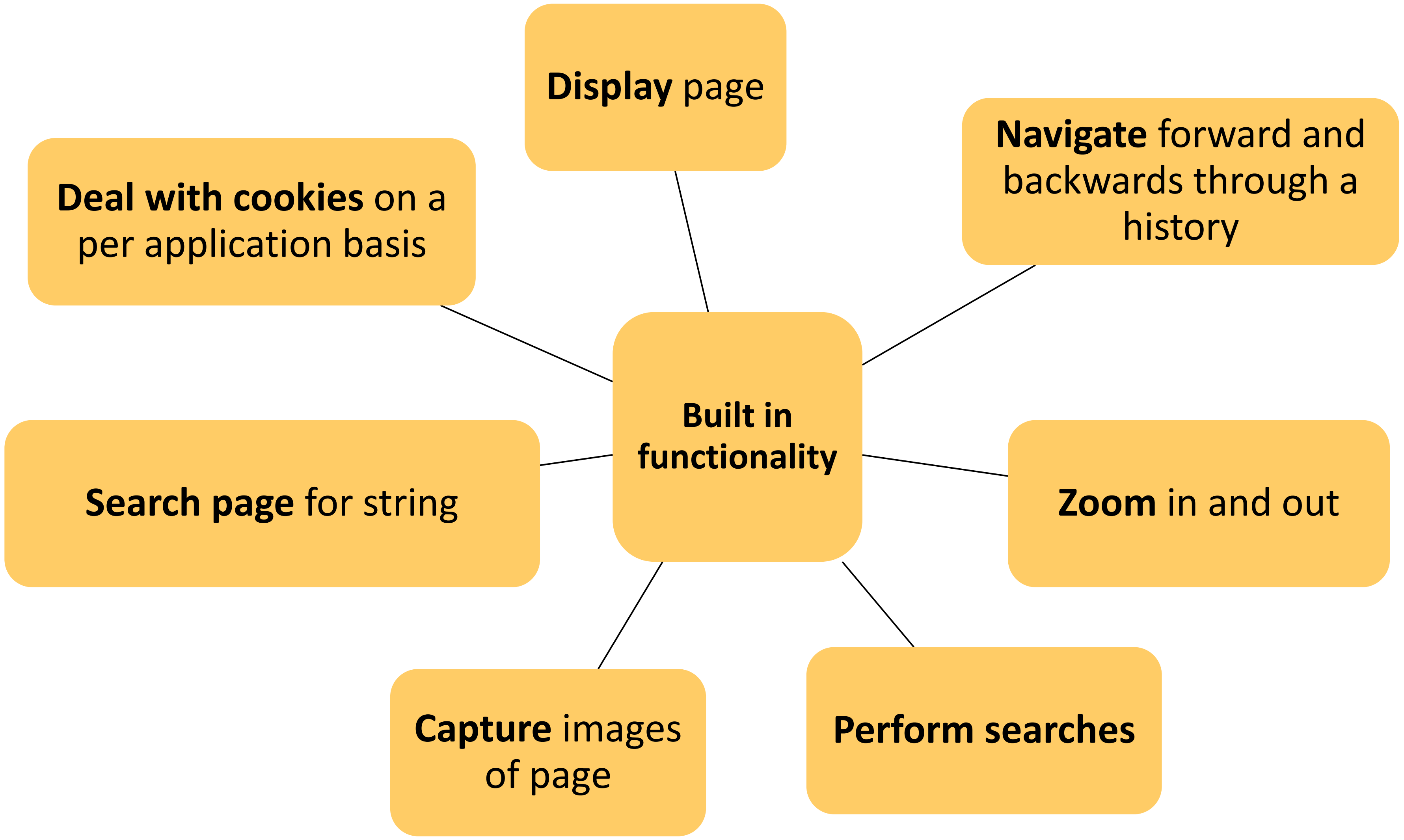
lecture 8

- Using WebView
- Monitoring the network connection
- Connecting to a web service
- Parsing and consuming data from the network

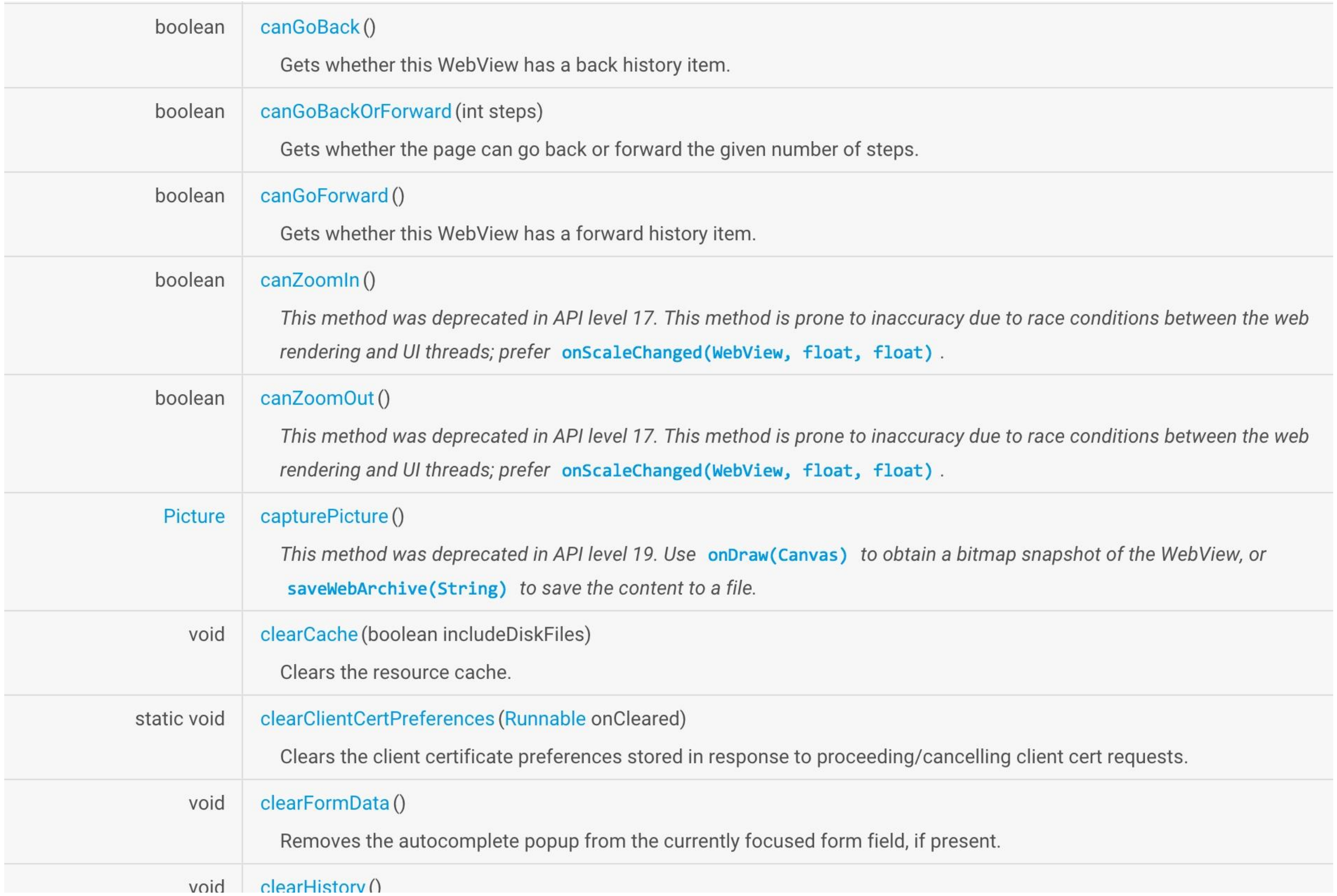
WebView

- A view that displays web pages
  - Basis for creating your own web browser
  - OR: just display some online content inside of your Activity
- Originally used the WebKit rendering engine. In API level 19 Chromium was introduced for improved JavaScript performance, which is now used along with the Blink rendering engine.

WebView


Display page
Navigate forward and backwards through a history
Deal with cookies on a per application basis
Built in functionality
Search page for string
Zoom in and out
Capture images of page
Perform searches

WebView



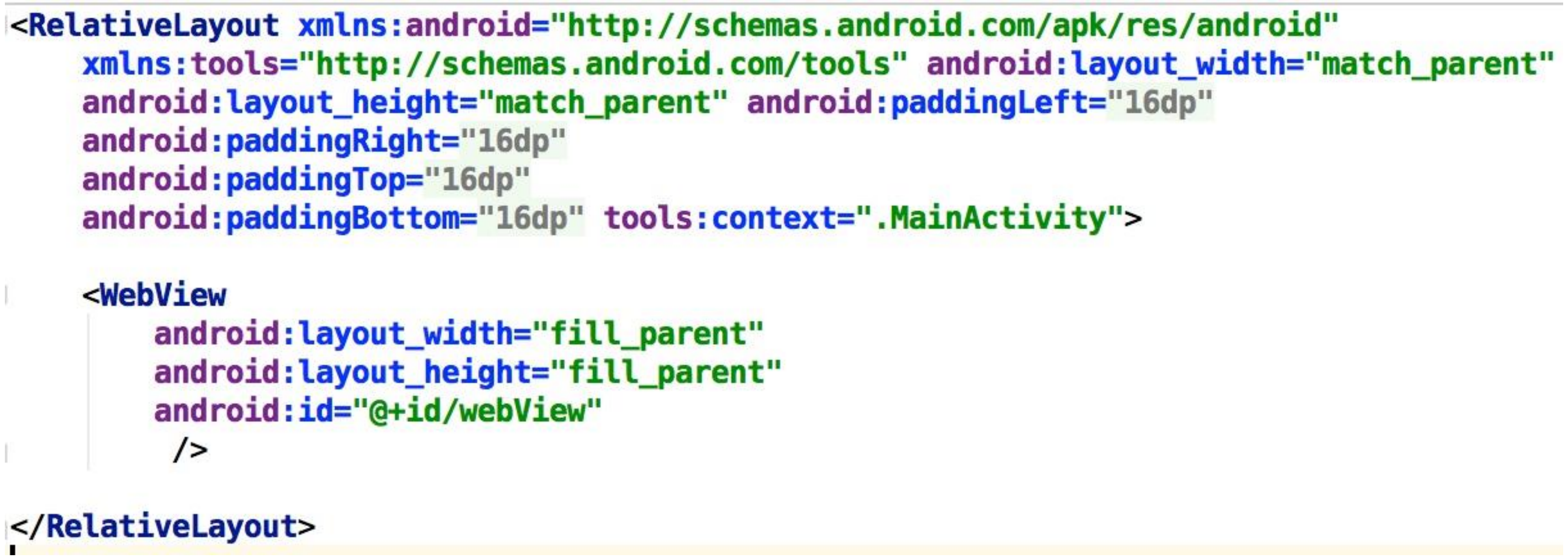
boolean	canGoBack () Gets whether this WebView has a back history item.
boolean	canGoBackOrForward (int steps) Gets whether the page can go back or forward the given number of steps.
boolean	canGoForward () Gets whether this WebView has a forward history item.
boolean	canZoomIn () *This method was deprecated in API level 17. This method is prone to inaccuracy due to race conditions between the web rendering and UI threads; prefer* `onScaleChanged(WebView, float, float)` .
boolean	canZoomOut () *This method was deprecated in API level 17. This method is prone to inaccuracy due to race conditions between the web rendering and UI threads; prefer* `onScaleChanged(WebView, float, float)` .
Picture	capturePicture () *This method was deprecated in API level 19. Use* `onDraw(Canvas)` *to obtain a bitmap snapshot of the WebView, or* `saveWebArchive(String)` *to save the content to a file.*
void	clearCache (boolean includeDiskFiles) Clears the resource cache.
static void	clearClientCertPreferences (Runnable onCleared) Clears the client certificate preferences stored in response to proceeding/cancelling client cert requests.
void	clearFormData () Removes the autocomplete popup from the currently focused form field, if present.
void	clearHistory ()

scenarios for using WebView

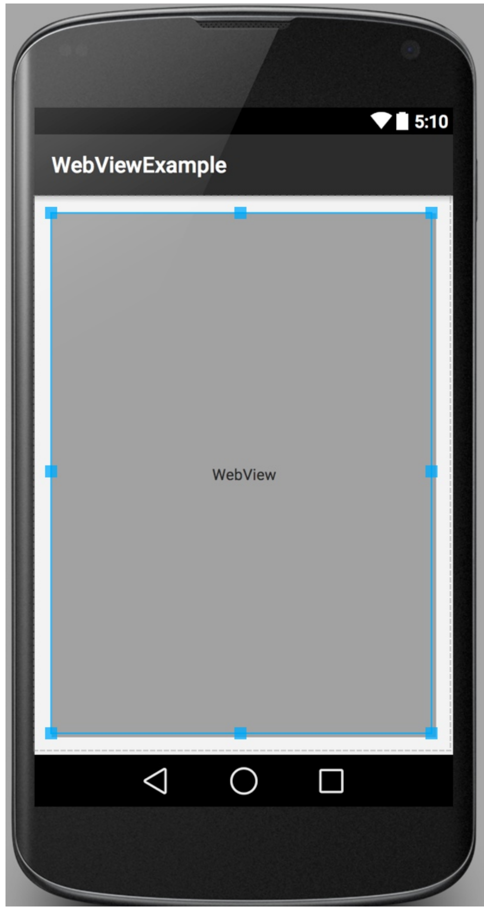
- Instead of built in browser:
  - Provide info the app might need to update such as end user agreement or user guide (instead of doing app update)
  - Display documents hosted online
- Application provides data that ALWAYS requires internet connection to retrieve data
  - As opposed to performing network request and parsing data to display in Android layout

WebView sample code

- Simple app to view and navigate web pages – WebViewExample app
- Main activity layout:

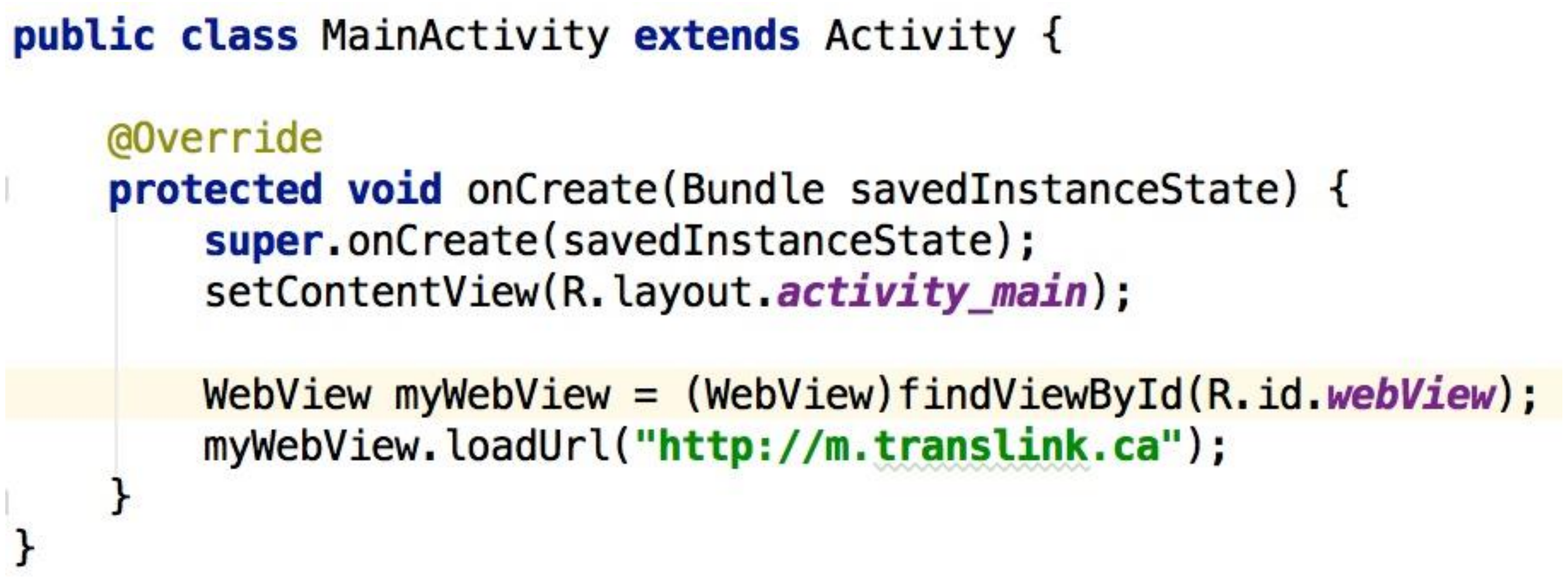


```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp" tools:context=".MainActivity">

    <WebView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:id="@+id/webView"
         />

</RelativeLayout>
```



WebView Activity

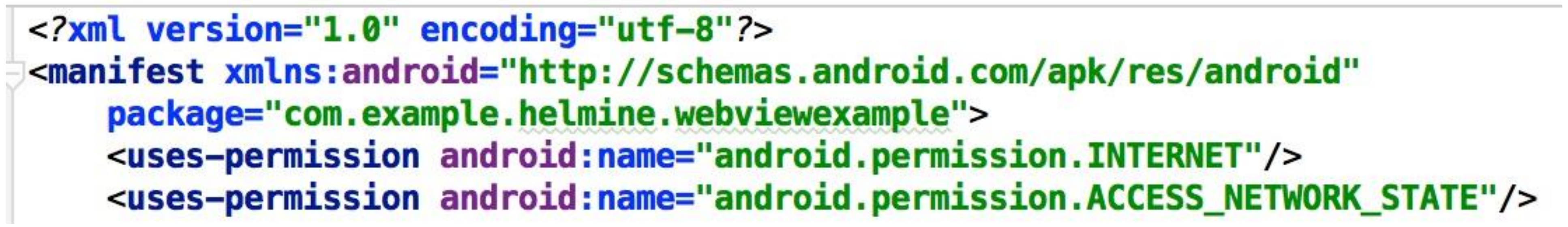
- Override onCreate
- Go to entered URL



```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        WebView myWebView = (WebView)findViewById(R.id.webView);
        myWebView.loadUrl("http://m.translink.ca");
    }
}
```

WebView example

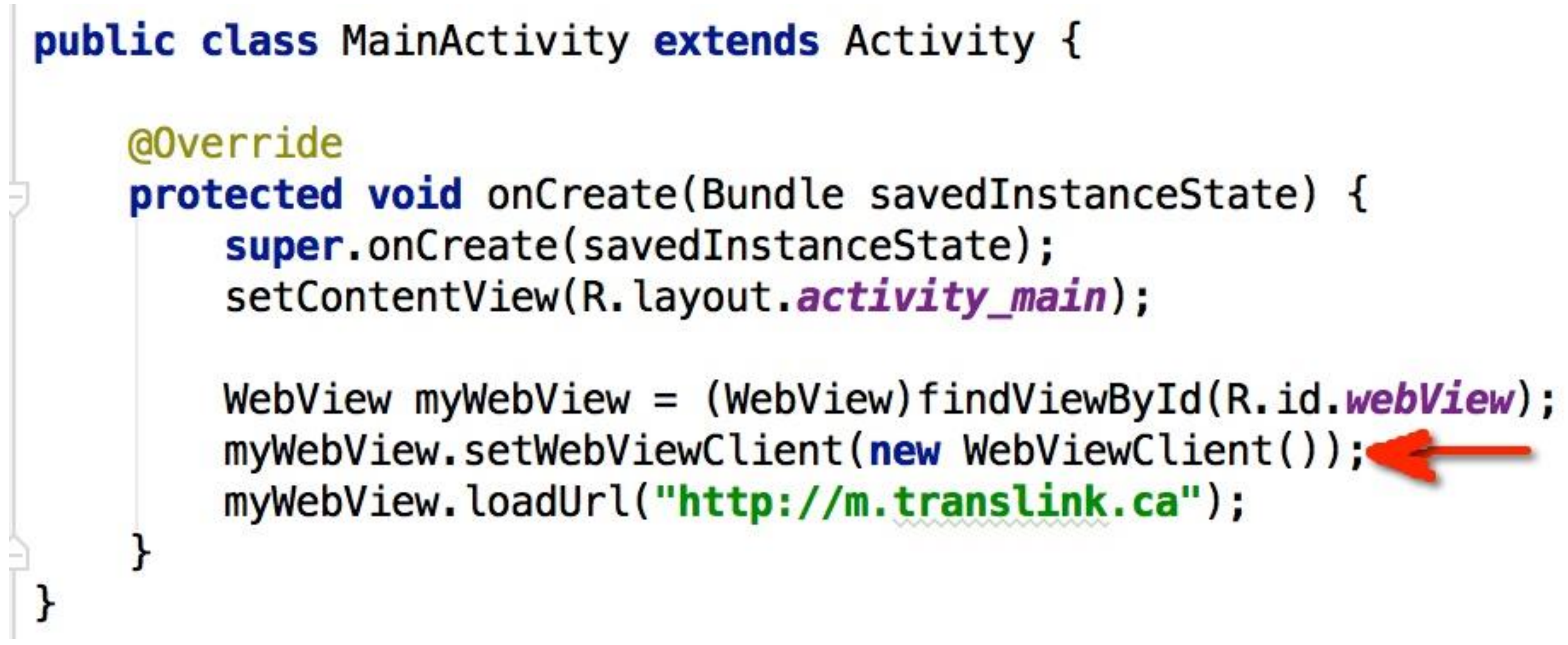
- Must add permission for app to use Internet
- Manifest file



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.helmine.webviewexample">
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
```

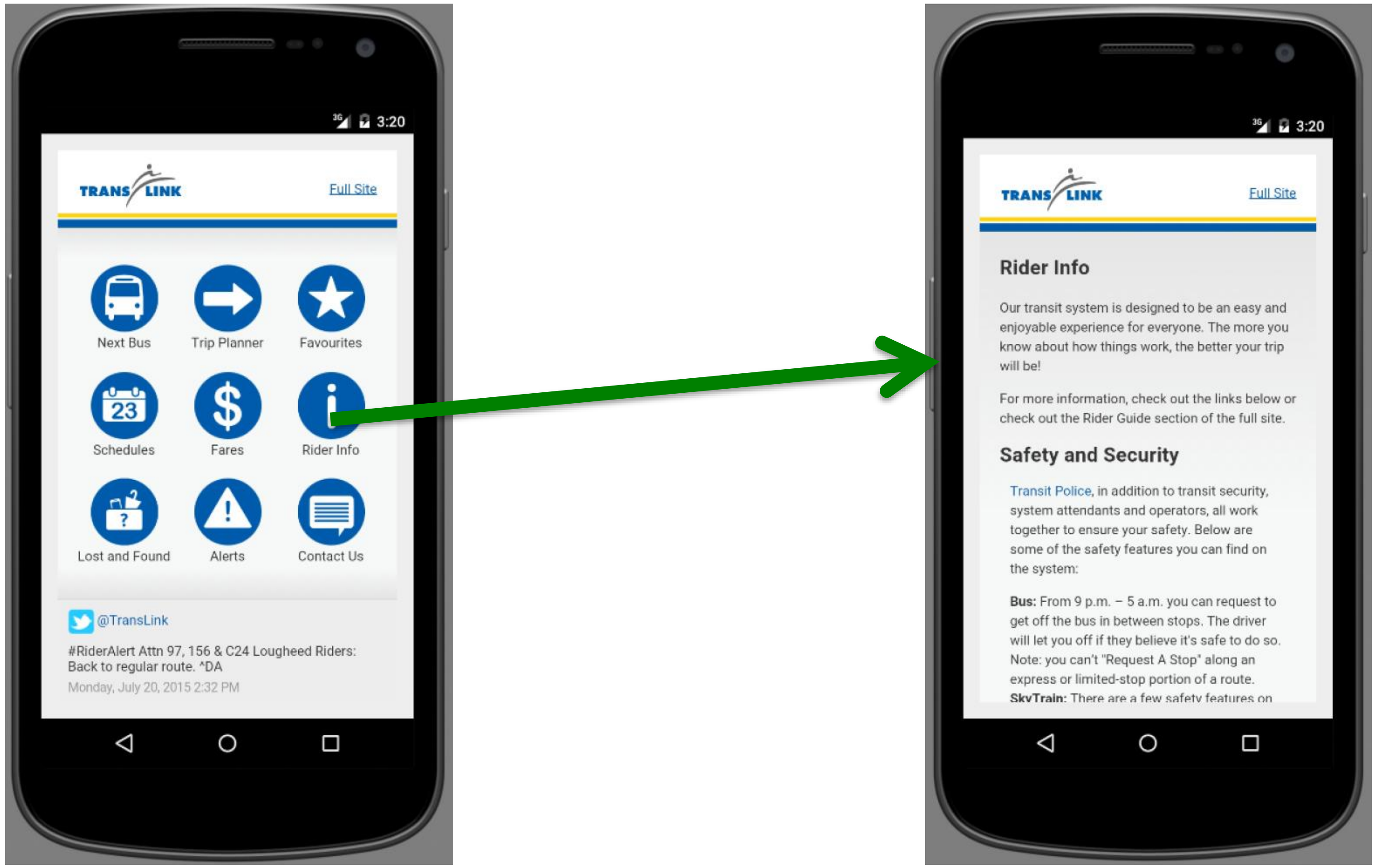
handling URL requests

- To enable activity to handle its own URL requests:
  - simply provide a WebViewClient for your WebView, using setWebViewClient().



```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        WebView myWebView = (WebView)findViewById(R.id.webView);
        myWebView.setWebViewClient(new WebViewClient());
        myWebView.loadUrl("http://m.translink.ca");
    }
}
```

result



#RiderAlert Attn 97, 156 & C24 Lougheed Riders: Back to regular route. ^DA

Monday, July 20, 2015 2:32 PM

Rider Info

Our transit system is designed to be an easy and enjoyable experience for everyone. The more you know about how things work, the better your trip will be!

For more information, check out the links below or check out the Rider Guide section of the full site.

Safety and Security

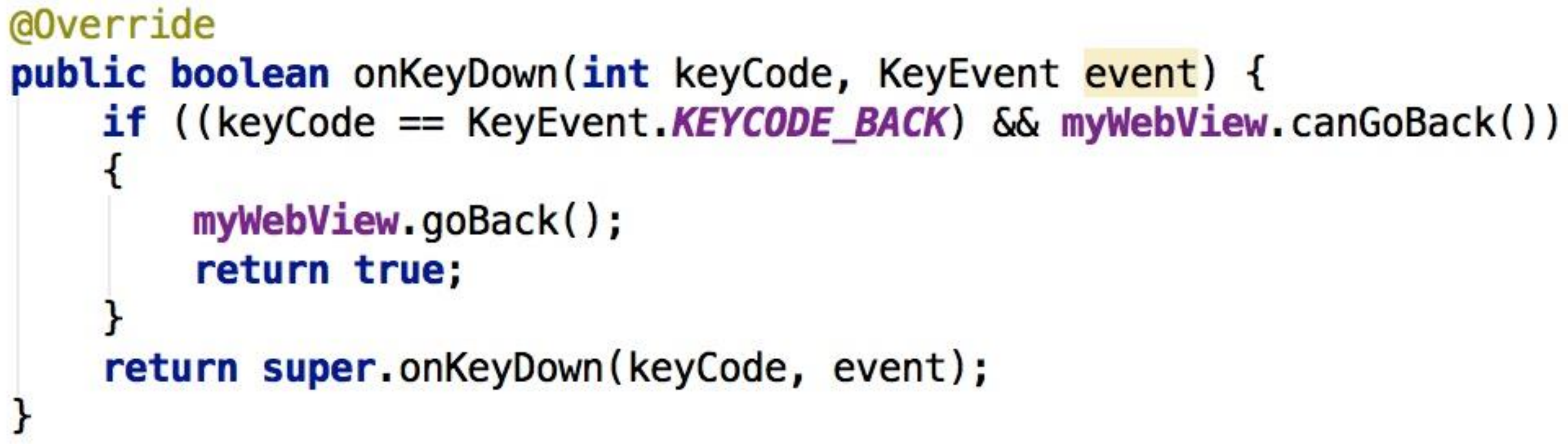
Transit Police, in addition to transit security, system attendants and operators, all work together to ensure your safety. Below are some of the safety features you can find on the system:

Bus: From 9 p.m. – 5 a.m. you can request to get off the bus in between stops. The driver will let you off if they believe it's safe to do so. Note: you can't "Request A Stop" along an express or limited-stop portion of a route.

SkyTrain: There are a few safety features on

navigating back

- Making previous changes disables the back button
- Must override **onKeyDown()** method



```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if ((keyCode == KeyEvent.KEYCODE_BACK) && myWebView.canGoBack())
    {
        myWebView.goBack();
        return true;
    }
    return super.onKeyDown(keyCode, event);
}
```

web services

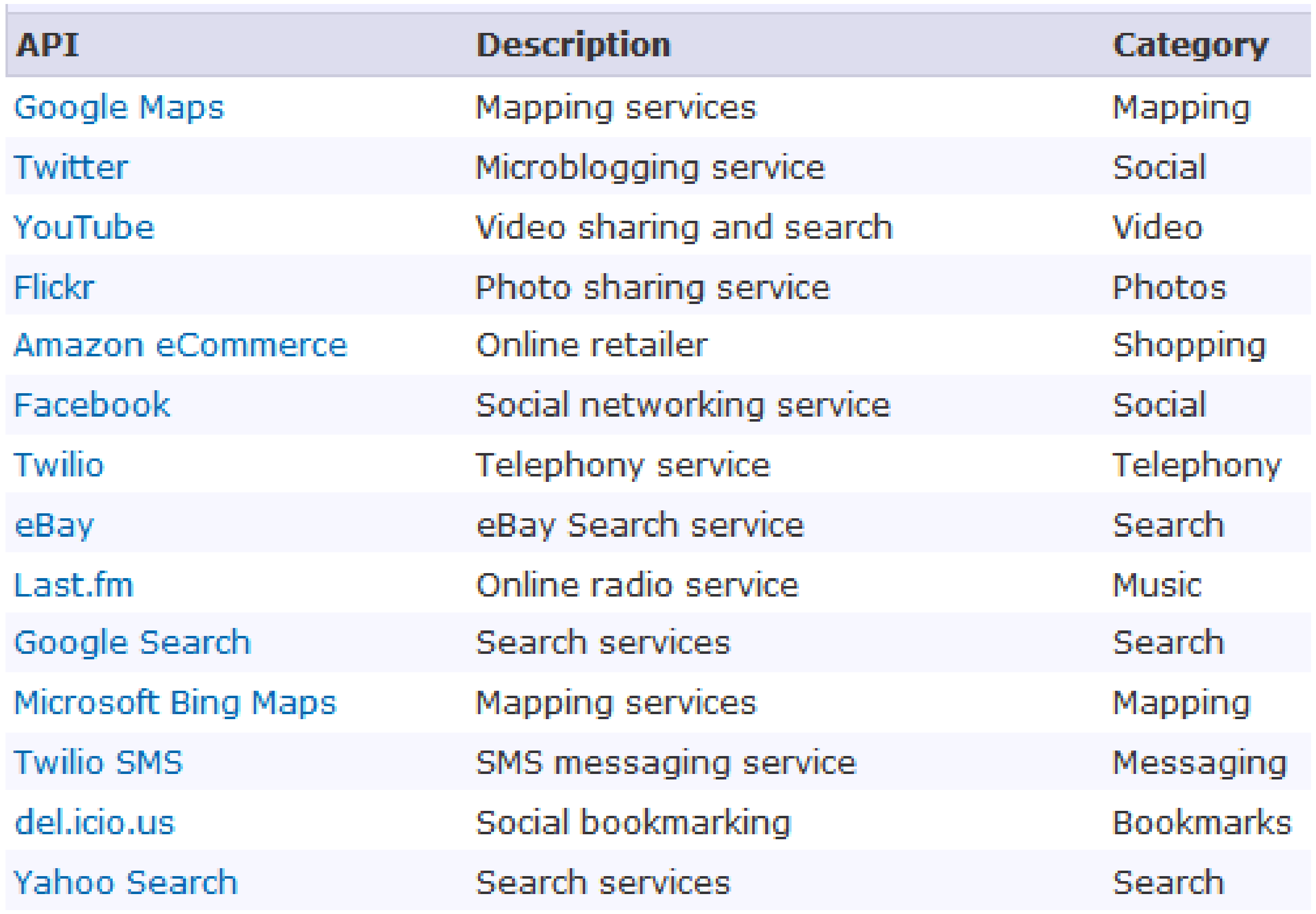
"Web services are a means of exposing an API over a technology-neutral network endpoint.

They are a means to call a remote method or operation that's not tied to a specific platform or vendor and get a result."

- *Android in Action* 3rd edition

web services sources

- https://www.programmableweb.com/news/which-are-developers-favorite-apis/research/2019/10/24



API	Description	Category
Google Maps	Mapping services	Mapping
Twitter	Microblogging service	Social
YouTube	Video sharing and search	Video
Flickr	Photo sharing service	Photos
Amazon eCommerce	Online retailer	Shopping
Facebook	Social networking service	Social
Twilio	Telephony service	Telephony
eBay	eBay Search service	Search
Last.fm	Online radio service	Music
Google Search	Search services	Search
Microsoft Bing Maps	Mapping services	Mapping
Twilio SMS	SMS messaging service	Messaging
del.icio.us	Social bookmarking	Bookmarks
Yahoo Search	Search services	Search

sample app

- Use a web service – http://www.geonames.org/export/ws-overview.html – to get current weather conditions

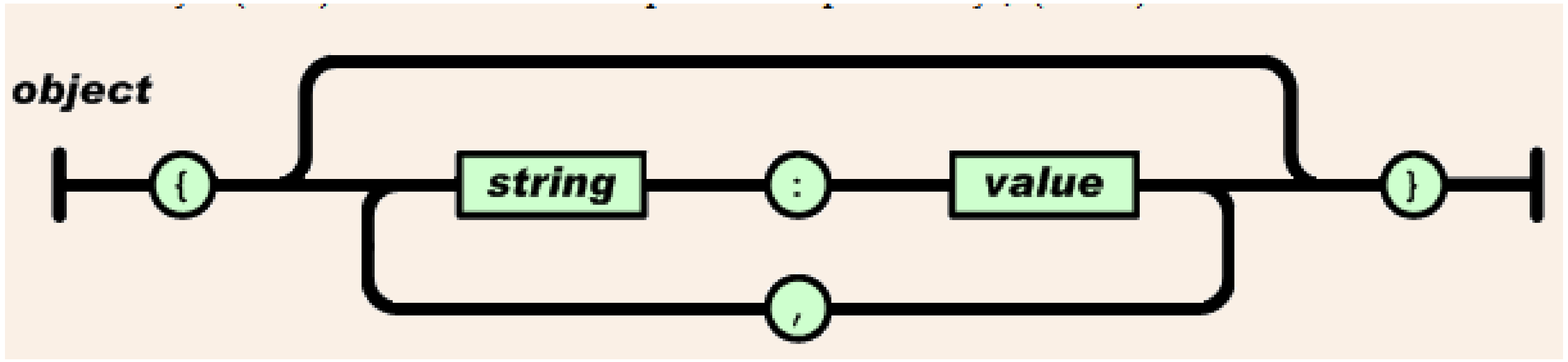
- Data from a web service can be returned as XML or JSON strings

JSON

- **J**ava**S**cript **O**bject **N**otation
- Is a way to represent JavaScript objects as Strings
- Alternative to XML for passing data between servers and clients

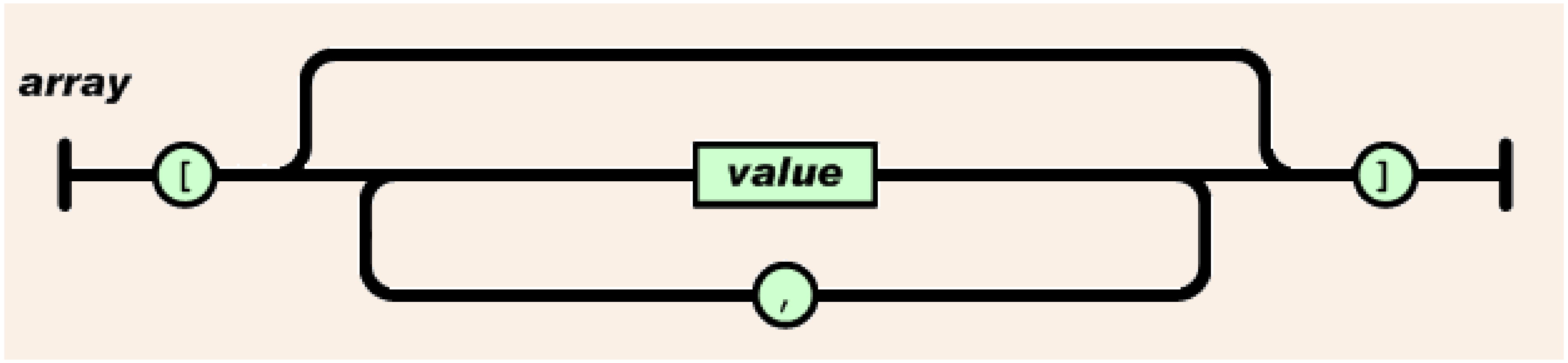
JSON Format

- Built on two structures
  - Collection of name-value pairs: a.k.a. objects, records, structs, etc.
  - An ordered list of values: a.k.a. an array
- Objects

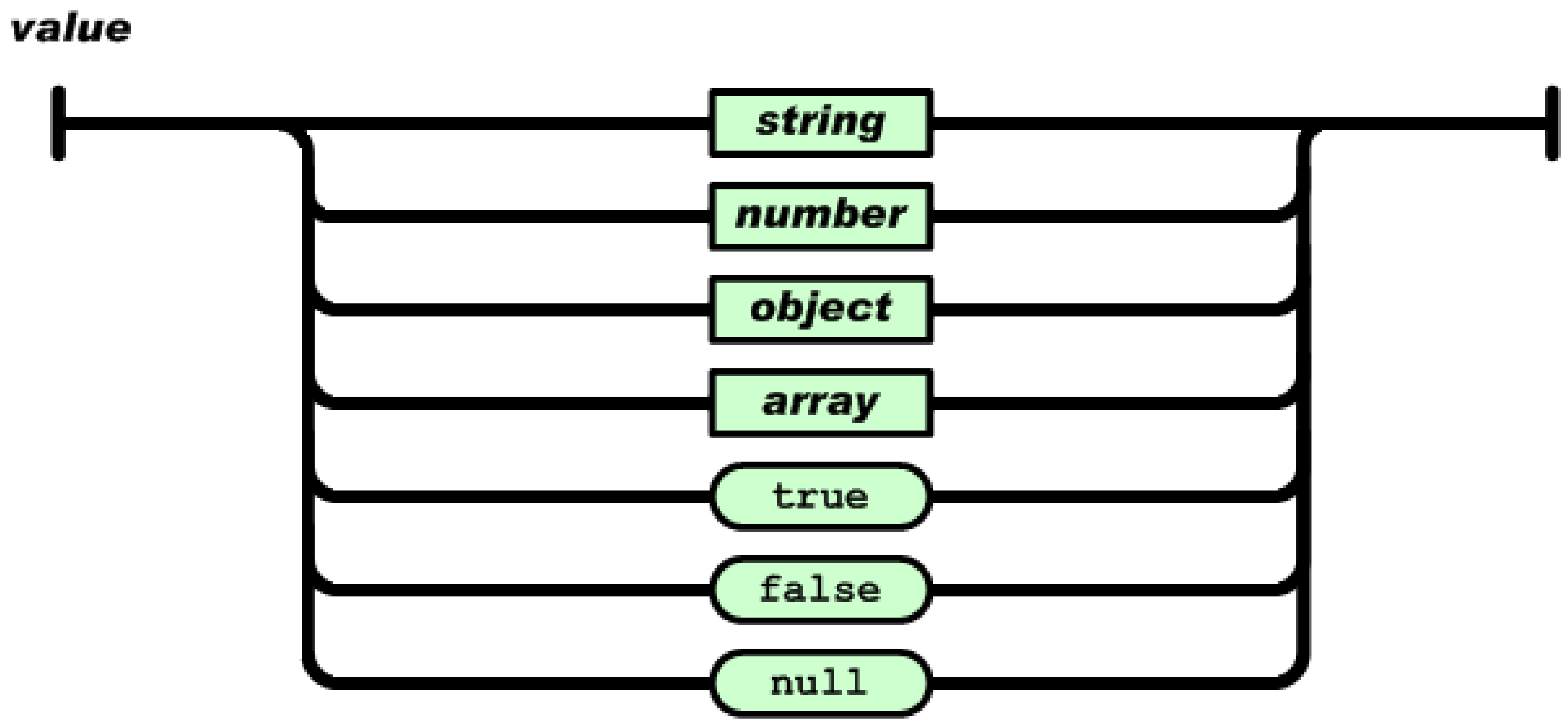


JSON Format

- Arrays
- Values
  - String, number, object, array, true, false, null



JSON values



- http://www.json.org/

JSON examples

- **Value (String)**
  - "Round Rock"
- **Array**
  - ["Round Rock", "Dallas", "Houston"]
- **Object**
  - {"height":70,"weight":165}

steps

1. Check the network connection

2. Perform network operation on a separate thread – setup separate thread

3. Connect and download data

4. Convert data from network into a target data type

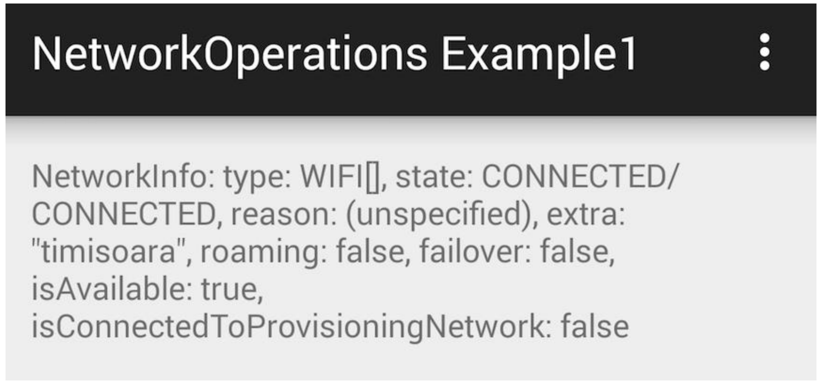
1. check the network connection

Is network connection available?

- **getActiveNetworkInfo()**
- **isConnected()**
- Device may be out of range of network
- User disabled wifi or mobile data access

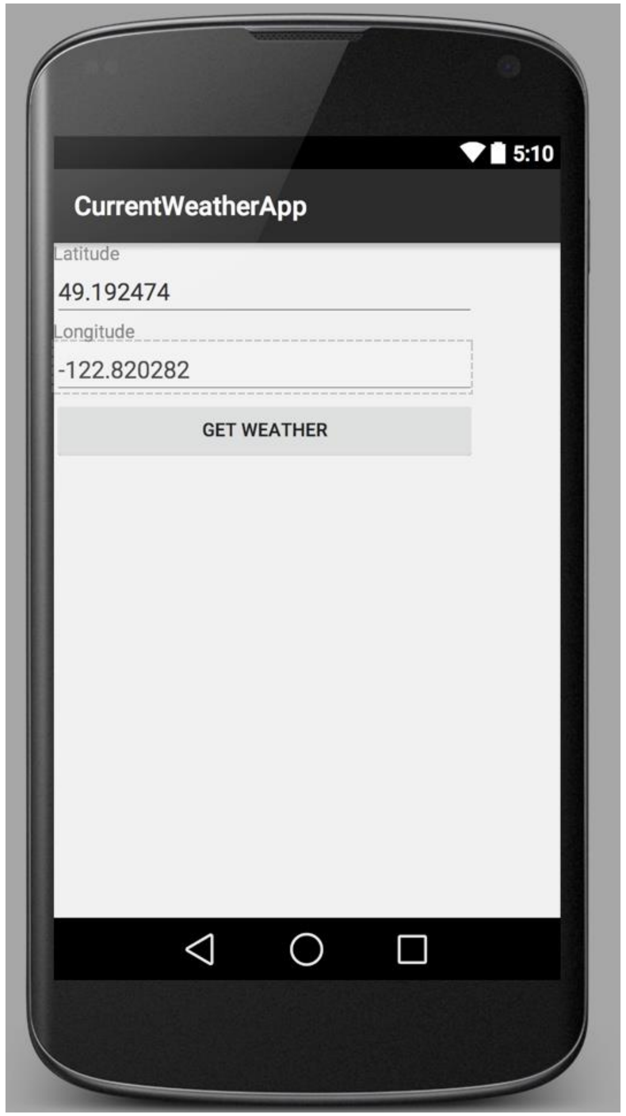
code

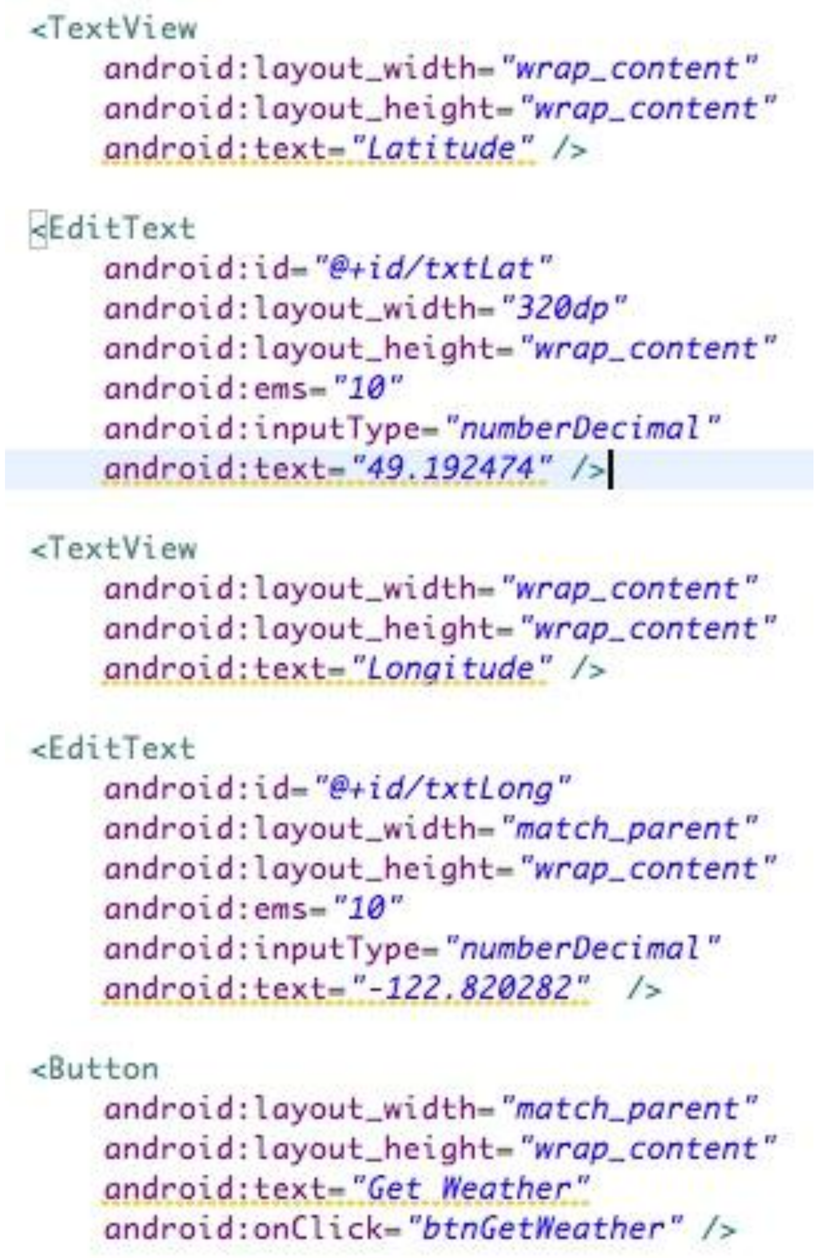
```
public void checkConnection(){
    ConnectivityManager connectMgr =
            (ConnectivityManager)getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo networkInfo = connectMgr.getActiveNetworkInfo();
    if(networkInfo != null && networkInfo.isConnected()){
        //fetch data

        //display network information in a TextView
        TextView txtView1 = (TextView)findViewById(R.id.TextView1);
        String networkInformation = networkInfo.toString();
        txtView1.setText(networkInformation);
        Toast.makeText(this, "connection ok", Toast.LENGTH_LONG).show();
    }
    else {
        //display error
        Toast.makeText(this, "no network connection", Toast.LENGTH_LONG).show();
    }
}
```


NetworkOperations Example1

NetworkInfo: type: WIFI[], state: CONNECTED/CONNECTED, reason: (unspecified), extra: "timisoara", roaming: false, failover: false, isAvailable: true, isConnectedToProvisioningNetwork: false

app GUI





```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Latitude" />

<EditText
    android:id="@+id/txtLat"
    android:layout_width="320dp"
    android:layout_height="wrap_content"
    android:ems="10"
    android:inputType="numberDecimal"
    android:text="49.192474" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Longitude" />

<EditText
    android:id="@+id/txtLong"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:inputType="numberDecimal"
    android:text="-122.820282"  />

<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Get Weather"
    android:onClick="btnGetWeather" />
```

2. perform network operation on a separate thread – setup separate thread

- Why?
  - Unpredictable delays -> poor user experience
- Solution:
  - Separate thread using AsyncTask

onPreExecute() runs on UI thread before background processing begins

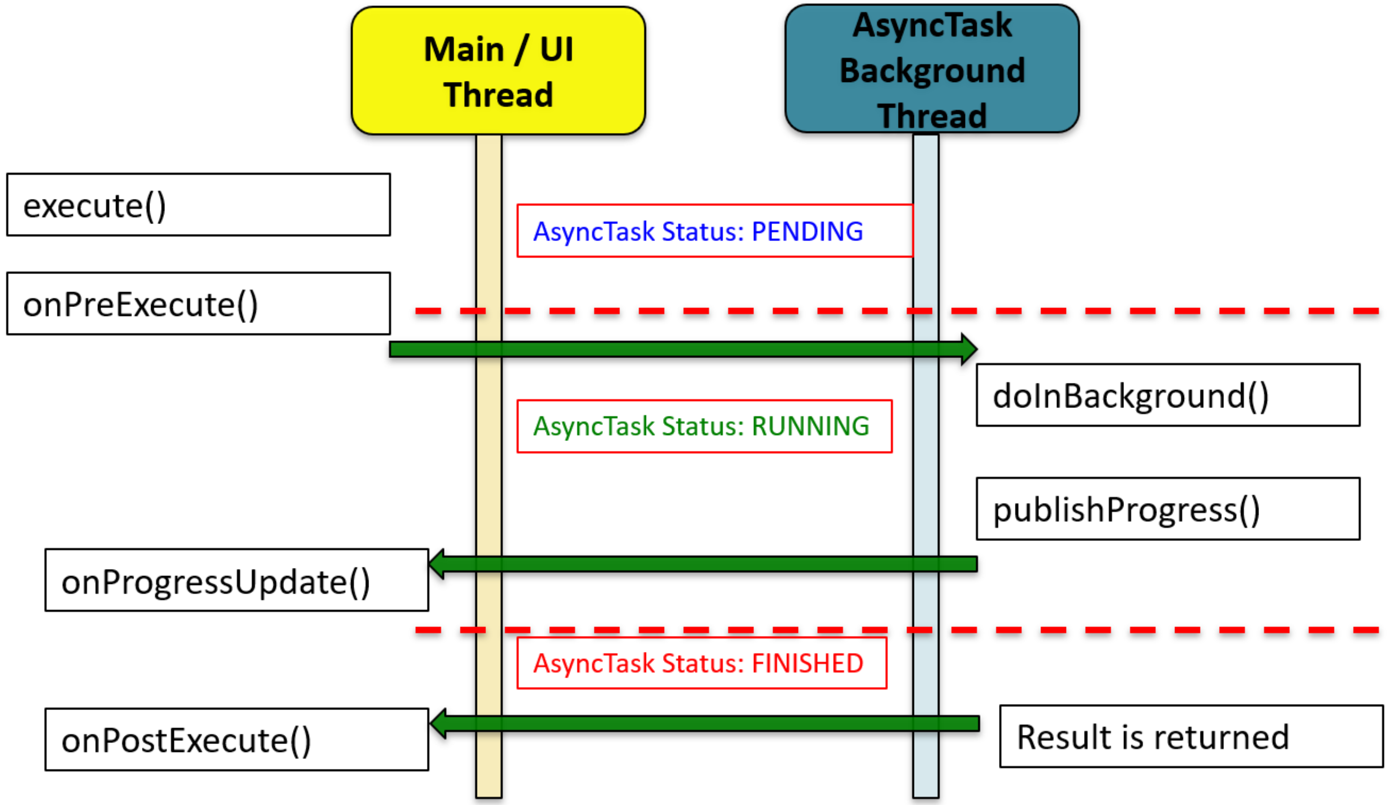
doInBackground(Param... params) runs on a background thread and won't block UI thread

publishProgress(Progress... values) method invoked by doInBackground triggers call to **onProgressUpdate()** method on UI thread

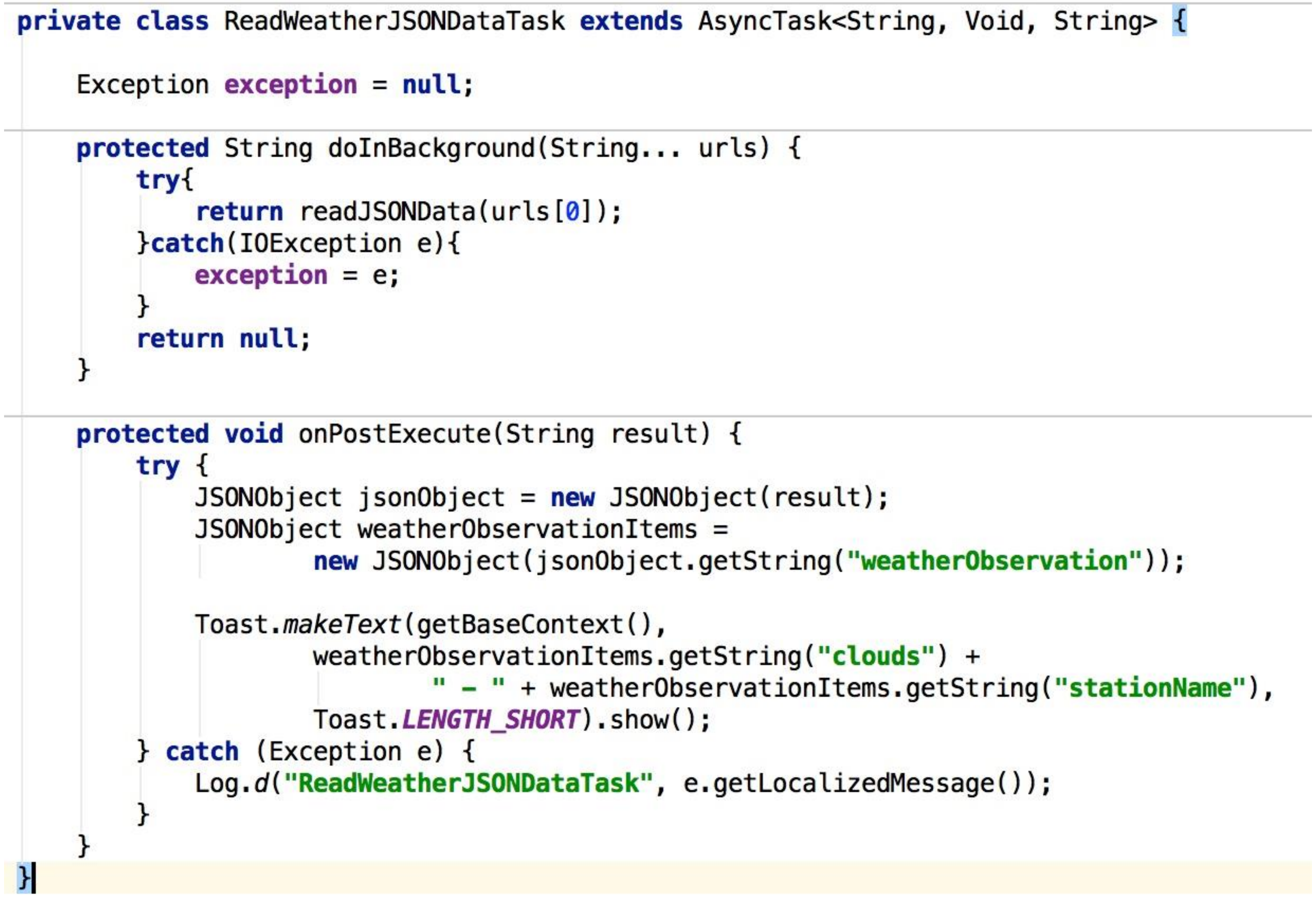
onPostExecute(Result result) runs on UI thread once doInBackground is done

AsyncTask - revisited

steps for AsyncTask



setting up AsyncTask



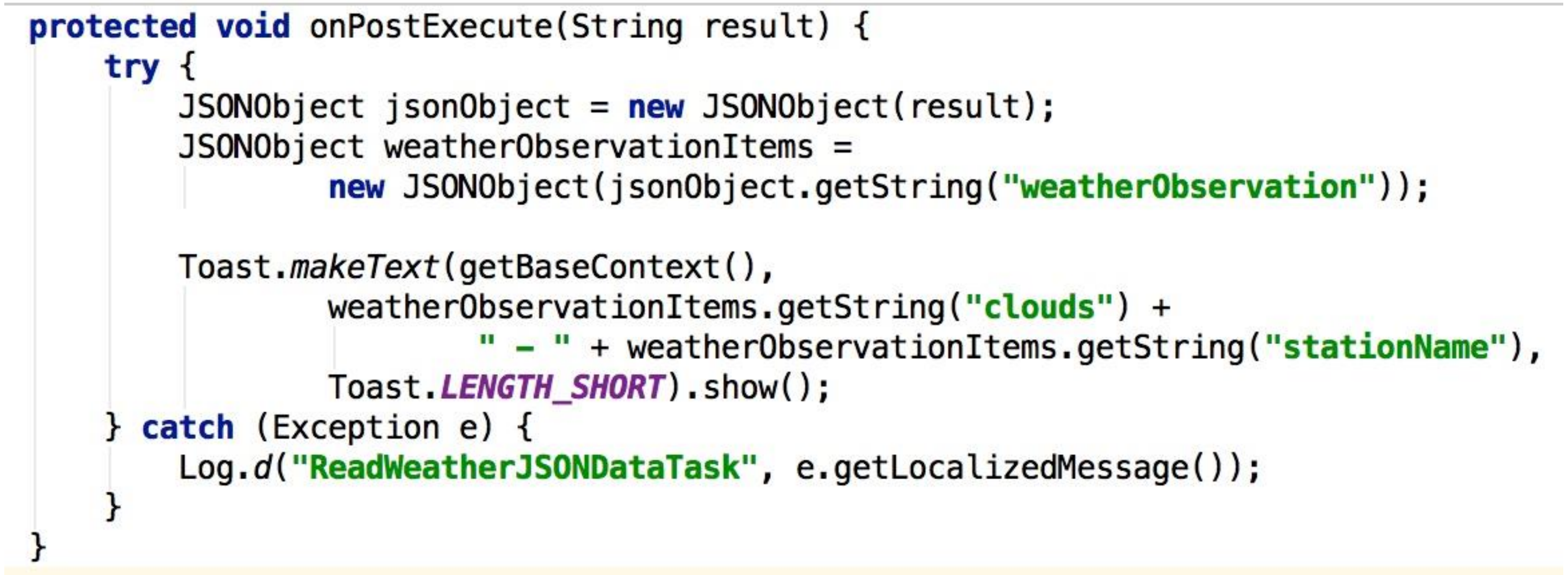
```
private class ReadWeatherJSONDataTask extends AsyncTask<String, Void, String> {

    Exception exception = null;

    protected String doInBackground(String... urls) {
        try{
            return readJSONData(urls[0]);
        }catch(IOException e){
            exception = e;
        }
        return null;
    }

    protected void onPostExecute(String result) {
        try {
            JSONObject jsonObject = new JSONObject(result);
            JSONObject weatherObservationItems =
                    new JSONObject(jsonObject.getString("weatherObservation"));

            Toast.makeText(getBaseContext(),
                    weatherObservationItems.getString("clouds") +
                            " - " + weatherObservationItems.getString("stationName"),
                    Toast.LENGTH_SHORT).show();
        } catch (Exception e) {
            Log.d("ReadWeatherJSONDataTask", e.getLocalizedMessage());
        }
    }
}
```

AsyncTask – doInBackground()

- Connect to the web service using a URL
- The readJSONData() method returns a JSON string with the result from the web service
- The JSON string with the result is provided to the onPostExecute() method

AsyncTask – onPostExecute() method

- Takes the JSON result string and parses it.
  - First, the JSON result string is passed as the argument to the constructor of the JSONObject class. This creates a new JSONObject object (jsonObject) with key/value mappings from the JSON string.
  - Then get the value of the weatherObservation key by using the getString() method of jsonObject.
  - The values are then passed as the constructor of the JSONObject class, creating another JSONObject – weatherObservationItems.
  - Finally, you extract the value of the clouds and stationName keys by calling the getString() method of weatherObservationItems.



```
protected void onPostExecute(String result) {
    try {
        JSONObject jsonObject = new JSONObject(result);
        JSONObject weatherObservationItems =
                new JSONObject(jsonObject.getString("weatherObservation"));

        Toast.makeText(getBaseContext(),
                weatherObservationItems.getString("clouds") +
                        " - " + weatherObservationItems.getString("stationName"),
                Toast.LENGTH_SHORT).show();
    } catch (Exception e) {
        Log.d("ReadWeatherJSONDataTask", e.getLocalizedMessage());
    }
}
```

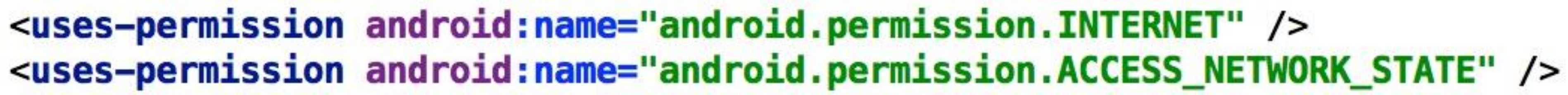
3. connect and download data

- How: use HTTP to send and receive data

HttpURLConnection - **recommended**

Apache HttpClient

- To perform network access in an Android app:



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

Apache HttpClient

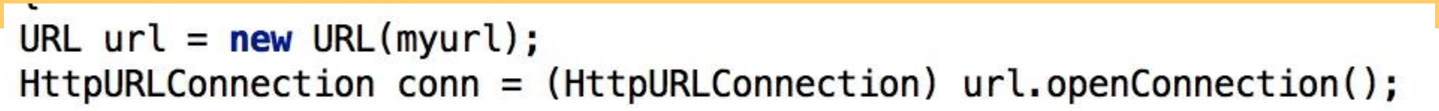
- Minimal code for simple HTTP processing
- Stable, very few bugs
- However: development of Apache HttpClient has been frozen since 2011
  - It is NOT the recommended approach

HttpURLConnection

- Member of the package java.net
- URL class also used
- To process the requests and responses: other classes from the package java.io
- Lightweight client
- Perfectly suited for use with web services

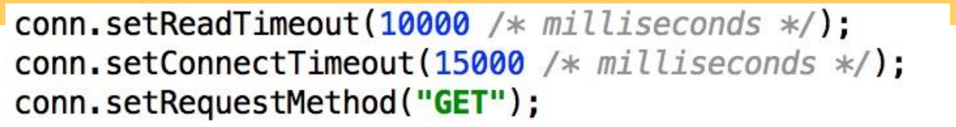
HttpURLConnection - usage

1. Obtain a new HttpURLConnection by calling URL.openConnection() and casting the result to HttpURLConnection



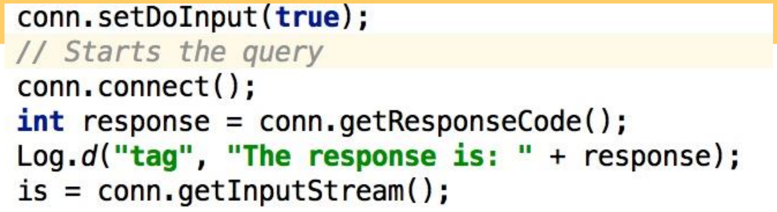
```
URL url = new URL(myurl);
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
```

2. Prepare the request.



```
conn.setReadTimeout(10000 /* milliseconds */);
conn.setConnectTimeout(15000 /* milliseconds */);
conn.setRequestMethod("GET");
```

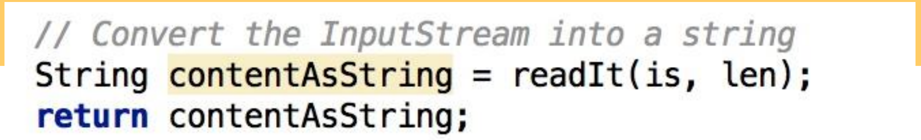
3. Transmit data by writing to the stream returned by getInputStream().



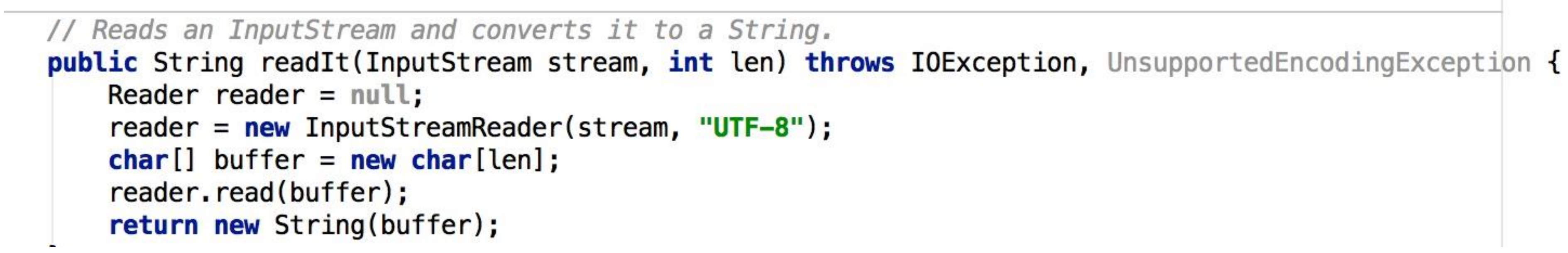
```
conn.setDoInput(true);
// Starts the query
conn.connect();
int response = conn.getResponseCode();
Log.d("tag", "The response is: " + response);
is = conn.getInputStream();
```

HttpURLConnection – usage (cont'd)

4. Read the response. The response body may be read from the stream returned by getInputStream(). If the response has no body, that method returns an empty stream.

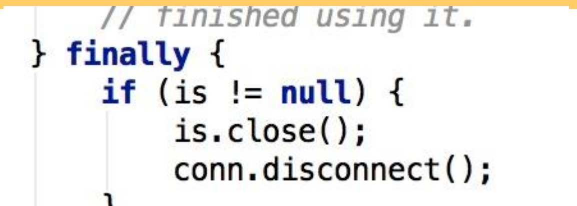


```
// Convert the InputStream into a string
String contentAsString = readIt(is, len);
return contentAsString;
```



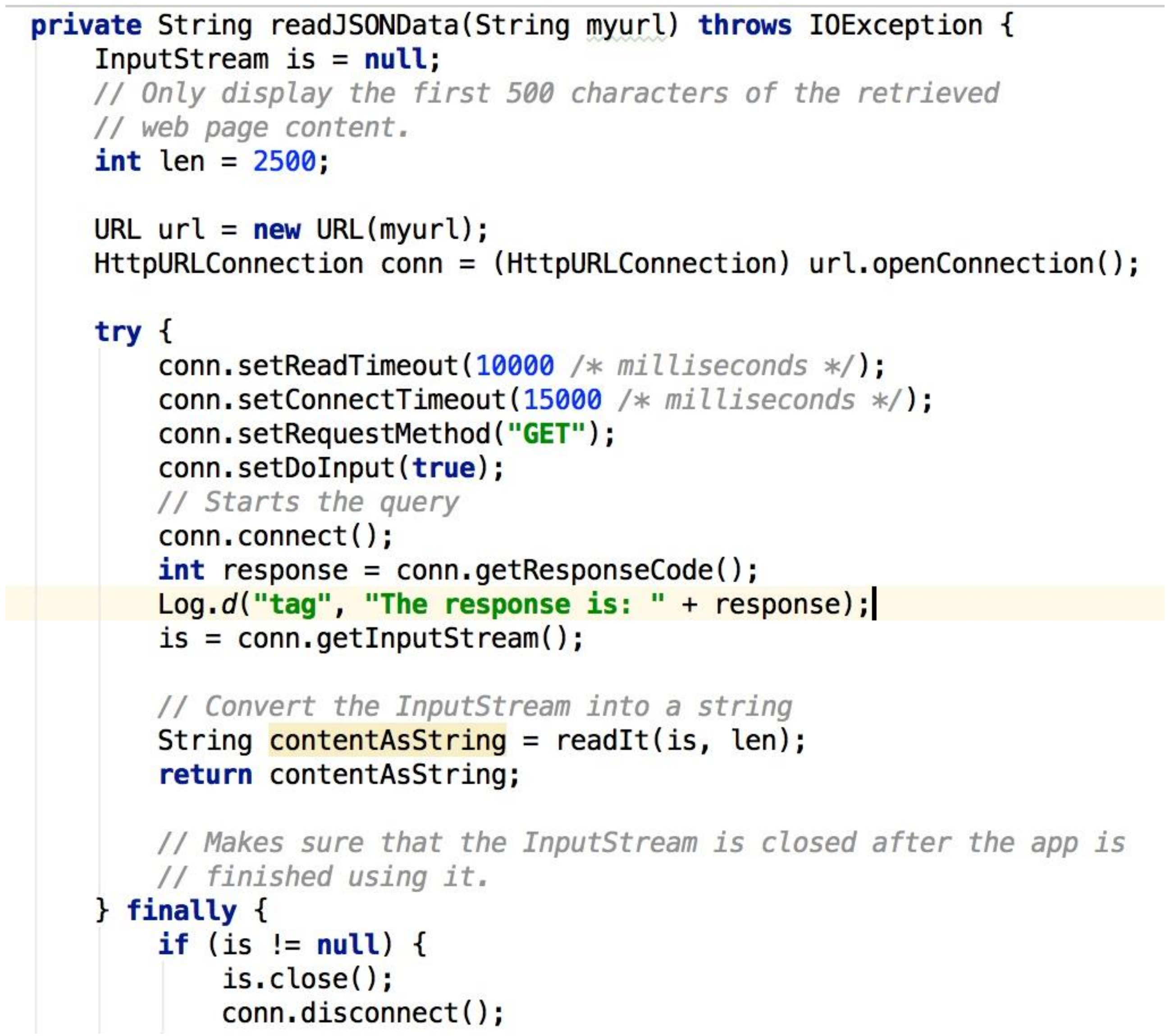
```
// Reads an InputStream and converts it to a String.
public String readIt(InputStream stream, int len) throws IOException, UnsupportedEncodingException {
    Reader reader = null;
    reader = new InputStreamReader(stream, "UTF-8");
    char[] buffer = new char[len];
    reader.read(buffer);
    return new String(buffer);
```

5. Disconnect. Once the response body has been read,
the HttpURLConnection should be closed by calling disconnect(). Disconnecting releases the resources held by a connection so they may be closed or reused.



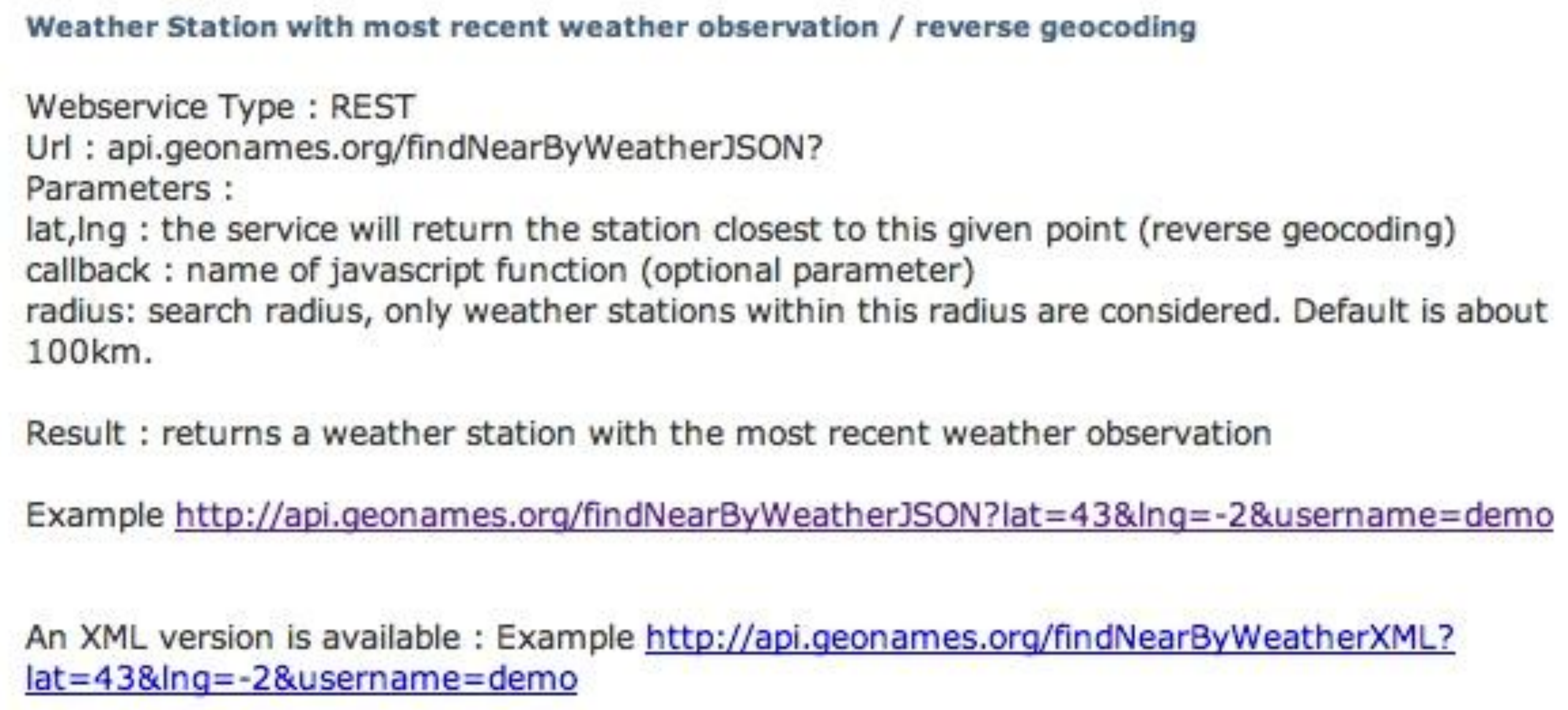
```
    // finished using it.
} finally {
    if (is != null) {
        is.close();
        conn.disconnect();
```

get data from WebService – use HttpURLConnection



```
private String readJSONData(String myurl) throws IOException {
    InputStream is = null;
    // Only display the first 500 characters of the retrieved
    // web page content.
    int len = 2500;

    URL url = new URL(myurl);
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();

    try {
        conn.setReadTimeout(10000 /* milliseconds */);
        conn.setConnectTimeout(15000 /* milliseconds */);
        conn.setRequestMethod("GET");
        conn.setDoInput(true);
        // Starts the query
        conn.connect();
        int response = conn.getResponseCode();
        Log.d("tag", "The response is: " + response);
        is = conn.getInputStream();

        // Convert the InputStream into a string
        String contentAsString = readIt(is, len);
        return contentAsString;

        // Makes sure that the InputStream is closed after the app is
        // finished using it.
    } finally {
        if (is != null) {
            is.close();
            conn.disconnect();
```

web service for weather



Weather Station with most recent weather observation / reverse geocoding

Webservice Type : REST
Url : api.geonames.org/findNearByWeatherJSON?
Parameters :
lat,lng : the service will return the station closest to this given point (reverse geocoding)
callback : name of javascript function (optional parameter)
radius: search radius, only weather stations within this radius are considered. Default is about 100km.

Result : returns a weather station with the most recent weather observation

Example http://api.geonames.org/findNearByWeatherJSON?lat=43&lng=-2&username=demo

An XML version is available : Example http://api.geonames.org/findNearByWeatherXML?lat=43&lng=-2&username=demo

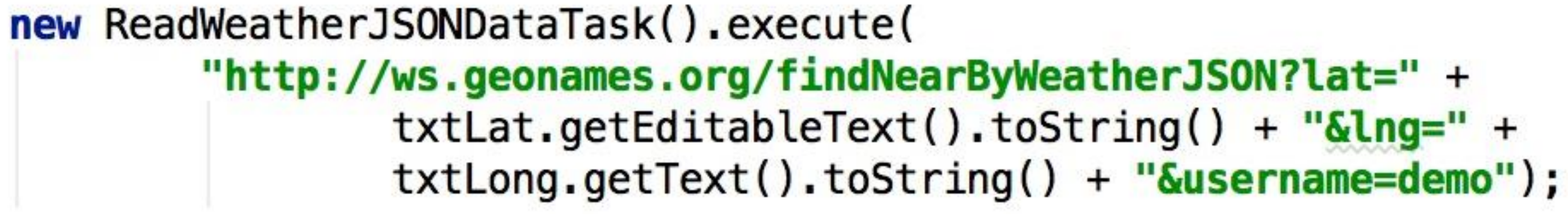
weather data from server

- Connect to the GeoNames web services to collect weather information

- To get the weather information of a particular location, we will use the following URL (enter the actual latitude and longitude):
  - http://api.geonames.org/findNearByWeatherJSON?lat=43&lng=-2&username=demo

use of weather data from server

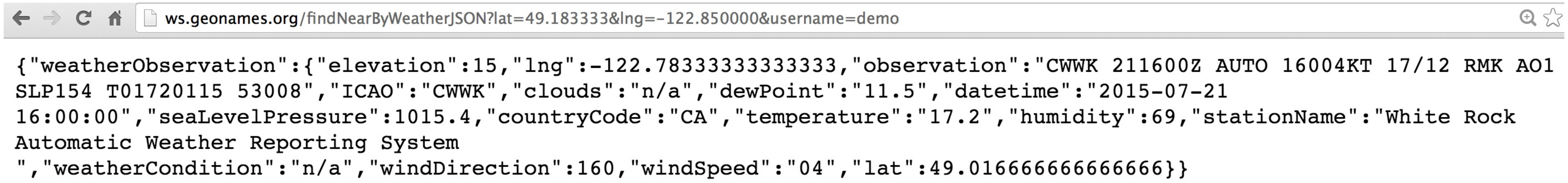
- Requires registration and a key value
  - Username: create an account and get your own username and password
  - Username= demo: limited number of requests per hour
- Key is used in requests
- Construct the request URL:



```
new ReadWeatherJSONDataTask().execute(
        "http://ws.geonames.org/findNearByWeatherJSON?lat=" +
                txtLat.getEditableText().toString() + "&lng=" +
                txtLong.getText().toString() + "&username=demo");
```

use of weather data from server

- Root key: weatherObservation.
- Its value is a collection of key/value pairs, such as clouds, weatherCondition, etc.



ws.geonames.org/findNearByWeatherJSON?lat=49.183333&lng=-122.850000&username=demo

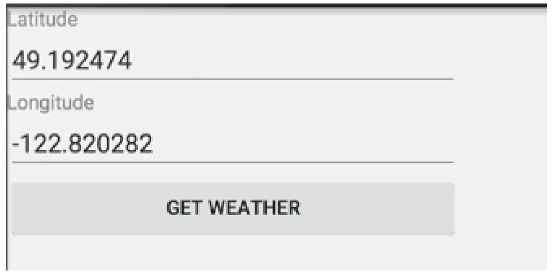
{"weatherObservation":{"elevation":15,"lng":-122.78333333333333,"observation":"CWWK 211600Z AUTO 16004KT 17/12 RMK AO1 SLP154 T01720115 53008","ICAO":"CWWK","clouds":"n/a","dewPoint":"11.5","datetime":"2015-07-21 16:00:00","seaLevelPressure":1015.4,"countryCode":"CA","temperature":"17.2","humidity":69,"stationName":"White Rock Automatic Weather Reporting System ","weatherCondition":"n/a","windDirection":160,"windSpeed":"04","lat":49.016666666666666}}

‘get weather’ button in the UI

Add the buttonGetWeather() method to MainActivity.java:

```java
public void buttonGetWeather(View view) {
    EditText txtLat = (EditText) findViewById(R.id.txtLat);
    EditText txtLong = (EditText) findViewById(R.id.txtLong);

    new ReadWeatherJSONDataTask().execute(
            "http://ws.geonames.org/findNearByWeatherJSON?lat=" +
                    txtLat.getEditableText().toString() + "&lng=" +
                    txtLong.getText().toString() + "&username=demo");
}
```

```xml
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Get Weather"
    android:onClick="buttonGetWeather" />
```



Latitude

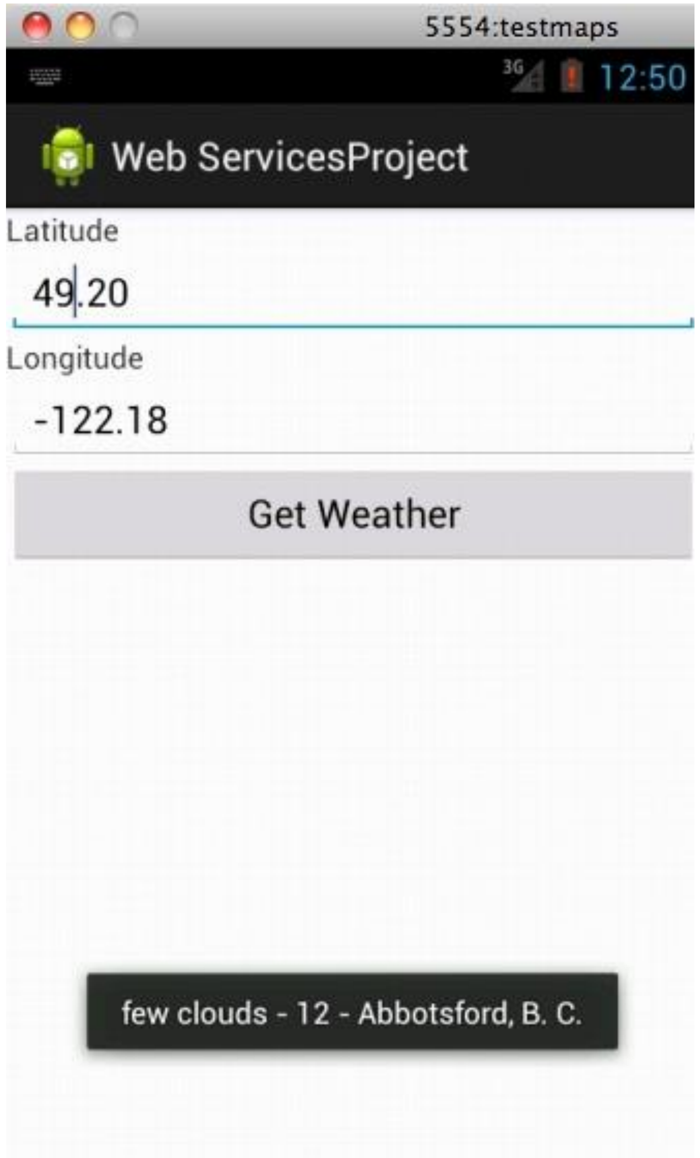
49.192474

Longitude

-122.820282

GET WEATHER

outcome



few clouds - 12 - Abbotsford, B. C.

resources

- WebView:
  http://developer.android.com/guide/webapps/webview.html

- Connecting to the network: http://developer.android.com/training/basics/network-ops/connecting.html
- Managing network usage: http://developer.android.com/training/basics/network-ops/managing.html

- Android's HTTP Clients: http://android-developers.blogspot.ca/2011/09/androids-http-clients.html
- Multithreading for performance: http://android-developers.blogspot.ca/2010/07/multithreading-for-performance.html