Connecting to the internet

- Steps
- Check Connectivity
- HttpURLConnection
- Connect, send and download
- Processing the data

Connecting to the Internet

Steps

- Check connectivity
- Connect to an URL
- Send data to an URL
- Retrieve data from the URL
- Process the data

Connecting to the Internet

This needs the
android.permission.ACCESS_NETWORK_STATE

Check the connectivity

Use the **ConnectivityManager** System Service

```
private static final String DEBUG_TAG = "NetworkStatusExample";
...
ConnectivityManager connMgr = (ConnectivityManager)
        getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo networkInfo = connMgr.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
boolean isWifiConn = networkInfo.isConnected();
networkInfo = connMgr.getNetworkInfo(ConnectivityManager.TYPE_MOBILE);
boolean isMobileConn = networkInfo.isConnected();
Log.d(DEBUG_TAG, "Wifi connected: " + isWifiConn);
Log.d(DEBUG_TAG, "Mobile connected: " + isMobileConn);
```

```
public boolean isOnline() {
    ConnectivityManager connMgr = (ConnectivityManager)
            getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
    return (networkInfo != null && networkInfo.isConnected());
}
```

IMPULSANDO EL CONOCIMIENTO TIC CUALIFICADO

Connecting to the Internet

This needs the
android.permission.INTERNET

HttpURLConnection

The Android platform includes the **HttpURLConnection** client, which supports HTTPS, streaming uploads and downloads, configurable timeouts, IPv6, and connection pooling.

Connect to an URL

```
URL url = new URL(myurl);
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setReadTimeout(10000 /* milliseconds */);
conn.setConnectTimeout(15000 /* milliseconds */);
conn.setRequestMethod("GET");
conn.setDoInput(true);
// Starts the query
conn.connect();
int response = conn.getResponseCode();
```

Not on the UI thread!

IMPULSANDO EL CONOCIMIENTO TIC CUALIFICADO

Connecting to the Internet

Sending Data

```
    ...
      conn.setRequestMethod("POST");
      connection.setDoOutput(true);
      connection.connect();

      OutputStreamWriter wr = new
OutputStreamWriter(connection.getOutputStream());
      wr.write(string);
      wr.flush();

    //process the response
```

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

    try {
        URL url = new URL(myurl);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        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(DEBUG_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();
        }
    }
}
```

Connecting to the Internet

Process the data

Depending on the data format, multiple paths:

1) XML
2) JSON
3) Text
4) Binary
5) ...

Connecting to the Internet

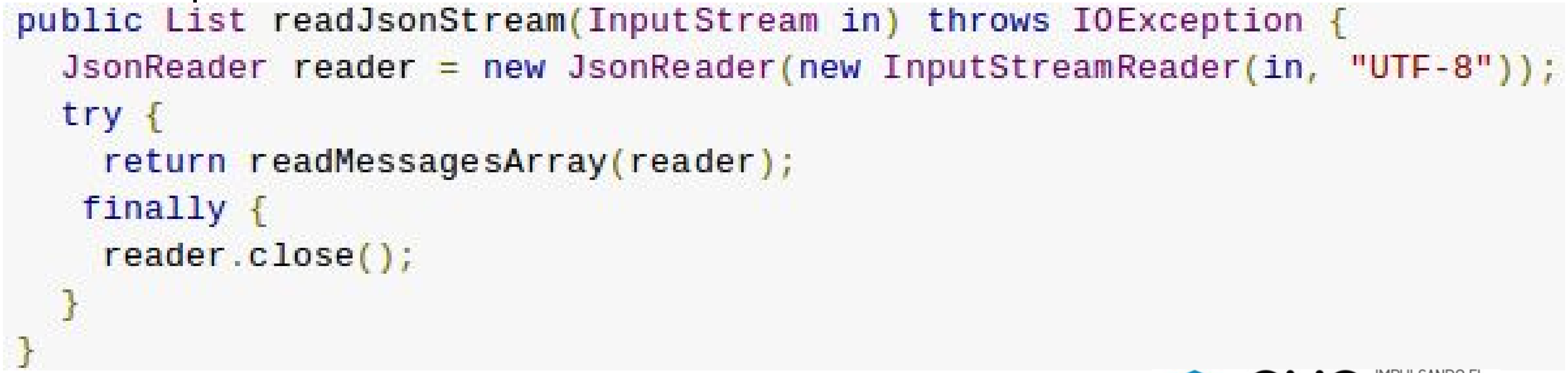
Parsing an XML

Android recommends **XmlPullParser**, which is an efficient and maintainable way to parse XML on Android. Historically Android has had two implementations of this interface:

- **KXmlParser** via XmlPullParserFactory.newPullParser().
- **ExpatPullParser**, via Xml.newPullParser().

Parsing a JSON (I)

Android has the class **JSONReader** which parses the contents of an InputStream



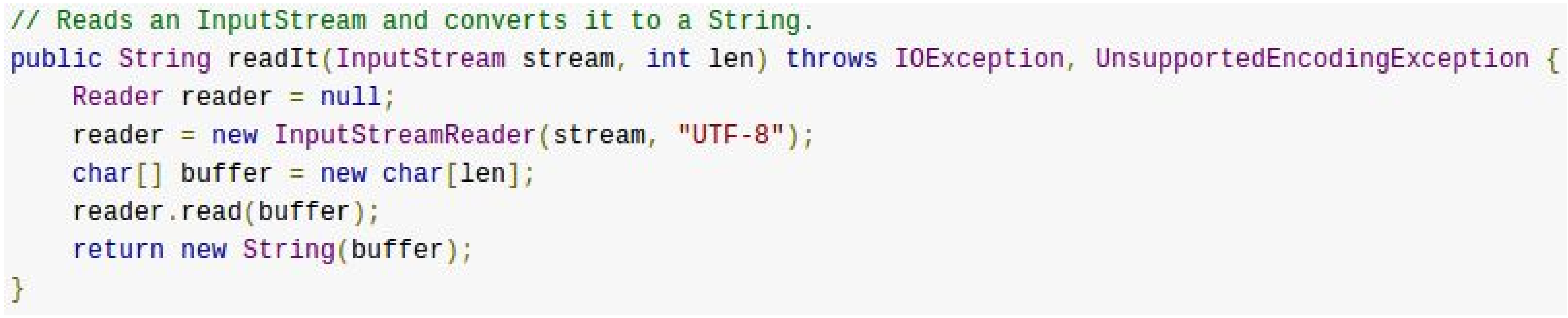
```
public List readJsonStream(InputStream in) throws IOException {
  JsonReader reader = new JsonReader(new InputStreamReader(in, "UTF-8"));
  try {
    return readMessagesArray(reader);
   finally {
    reader.close();
  }
}
```

Connecting to the Internet

Parsing a JSON (II)

Use an external Library, such as **GSON**

Converting to a String:



```java
// 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);
}
```