

# CMSC436: Programming Handheld Systems

# Networking

# Today's Topics

Networking

Android networking classes

Processing HTTP responses

# Networking

Early handheld devices gave us mobility, but had limited connectivity

Today's devices have greater mobility and connectivity

Today, many applications use data and services via the Internet

# Networking

Android includes multiple networking support classes, e.g.,

java.net – (Socket, URL, URLConnection)

# Example Application

Sends a request to a networked server for earthquake data

Receives the earthquake data

Displays the requested data

# Sending HTTP Requests

Socket

HttpURLConnection

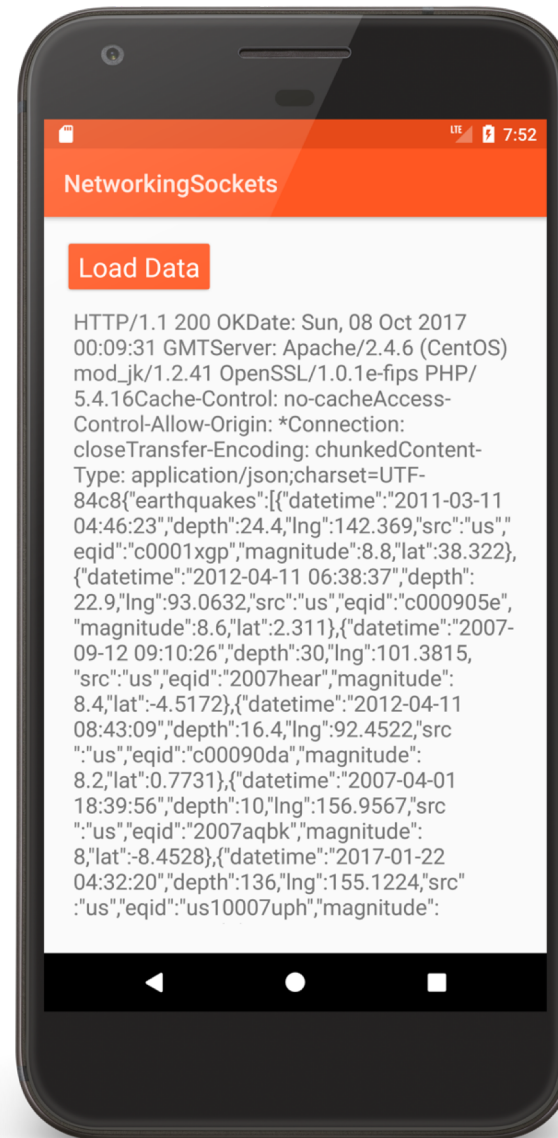
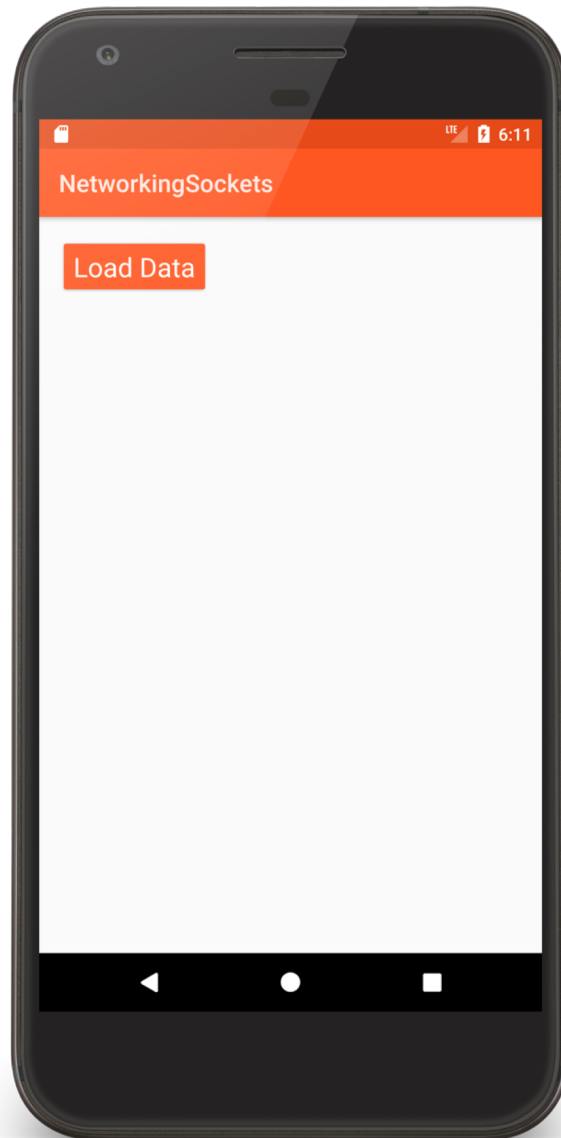
# Networking Permissions

Applications need permission to open network sockets

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



# Networking Sockets



```
static class HttpGetTask extends AsyncTask<Void, Void, String> {
    private static final String HOST = "api.geonames.org";
    // Get your own user name at http://www.geonames.org/login
    private static final String USER_NAME = "aporter";
    private static final String HTTP_GET_COMMAND =
        "GET /earthquakesJSON?north=44.1&south=-9.9&east=22.4&west=55.2&username="
        + USER_NAME + " HTTP/1.1" + "\n" + "Host: " + HOST + "\n"
        + "Connection: close" + "\n\n";
    ...
    protected String doInBackground(Void... params) {
        Socket socket = null;
        String data = "";
    }
    ...
}
```

```
try {
    socket = new Socket(HOST, 80);
    PrintWriter pw = new PrintWriter(
        new OutputStreamWriter(socket.getOutputStream()), true);
    pw.println(HTTP_GET_COMMAND);
    data = readStream(socket.getInputStream());
} catch (IOException exception) {
    exception.printStackTrace();
} finally {
    ...
}
return data;
}
```

```
private String readStream(InputStream in) {  
    BufferedReader reader = null;  
    StringBuilder data = new StringBuilder();  
    try {  
        reader = new BufferedReader(new InputStreamReader(in));  
        String line;  
        while ((line = reader.readLine()) != null) {  
            data.append(line);  
        }  
    } catch (IOException e) {  
        Log.e(TAG, "IOException");  
    } finally {  
        ...  
    }  
    return data.toString();  
}
```

...

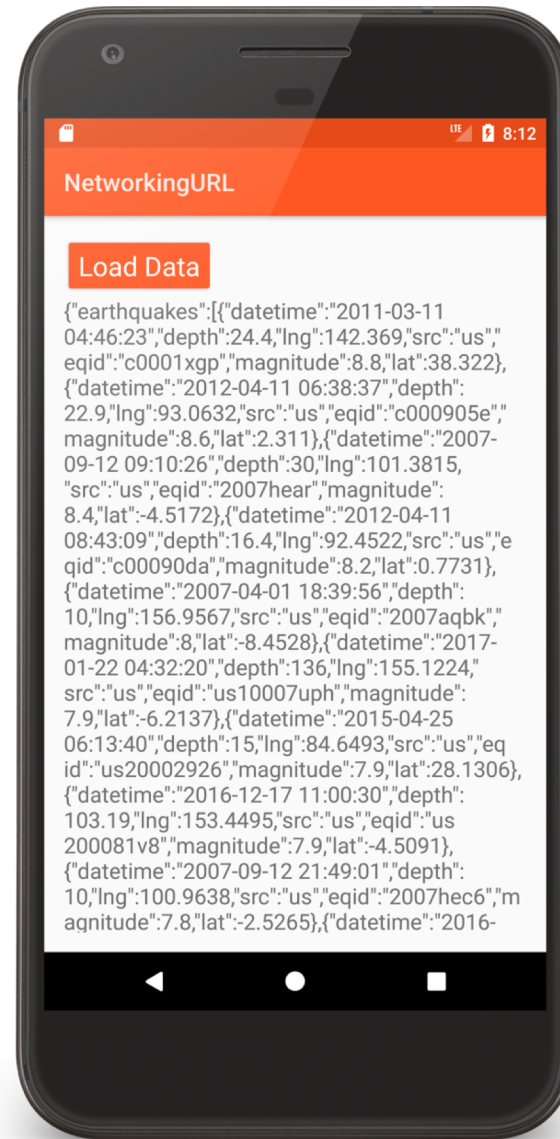
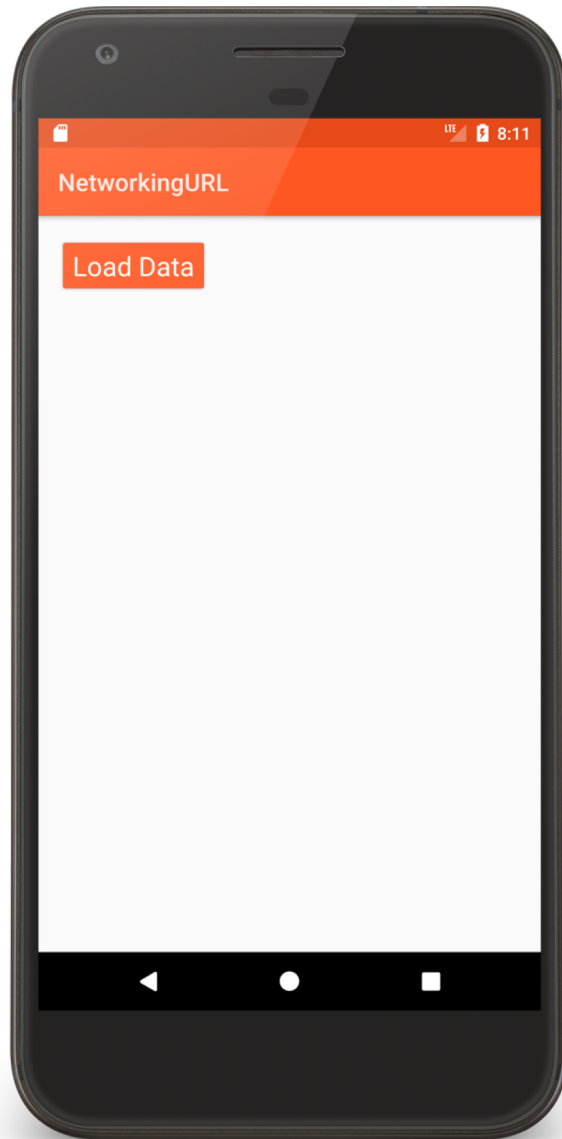
# HttpURLConnection

Higher-level than Sockets

# Usage Pattern

1. Get an `HttpURLConnection`
2. Prepare your request
3. Optionally, upload a request body
4. Read the response
5. Disconnect.

# Networking URL



```
static class HttpGetTask extends AsyncTask<Void, Void, String> {
    private static final String USER_NAME = "aporter";
    private static final String HOST = "api.geonames.org";
    private static final String URL = "http://" + HOST +
        "/earthquakesJSON?north=44.1&south=-9.9&east=-22.4&west=55.2&username="
        + USER_NAME;
    ...
    protected String doInBackground(Void... params) {
        String data = null;
        HttpURLConnection httpURLConnection = null;

        try {
            // 1. Get connection. 2. Prepare request (URI)
            httpURLConnection = (HttpURLConnection) new URL(URL)
                .openConnection();
        }
        ...
    }
}
```



```
// 3. This app does not use a request body  
// 4. Read the response  
InputStream in = new BufferedInputStream(httpURLConnection.getInputStream());  
data = readStream(in);  
} catch (MalformedURLException exception) {  
    Log.e(TAG, "MalformedURLException");  
} catch (IOException exception) {  
    Log.e(TAG, "IOException");  
} finally {  
    if (null != httpURLConnection) {  
        // 5. Disconnect  
        httpURLConnection.disconnect();  
    }  
}  
return data;  
}
```

# Processing Http Responses

Will focus on two popular formats:

JSON

XML

# Javascript Object Notation (JSON)

A lightweight data interchange format

Data packaged in two types of structures:

- Maps of key/value pairs

- Ordered lists of values

See: <http://www.json.org/>

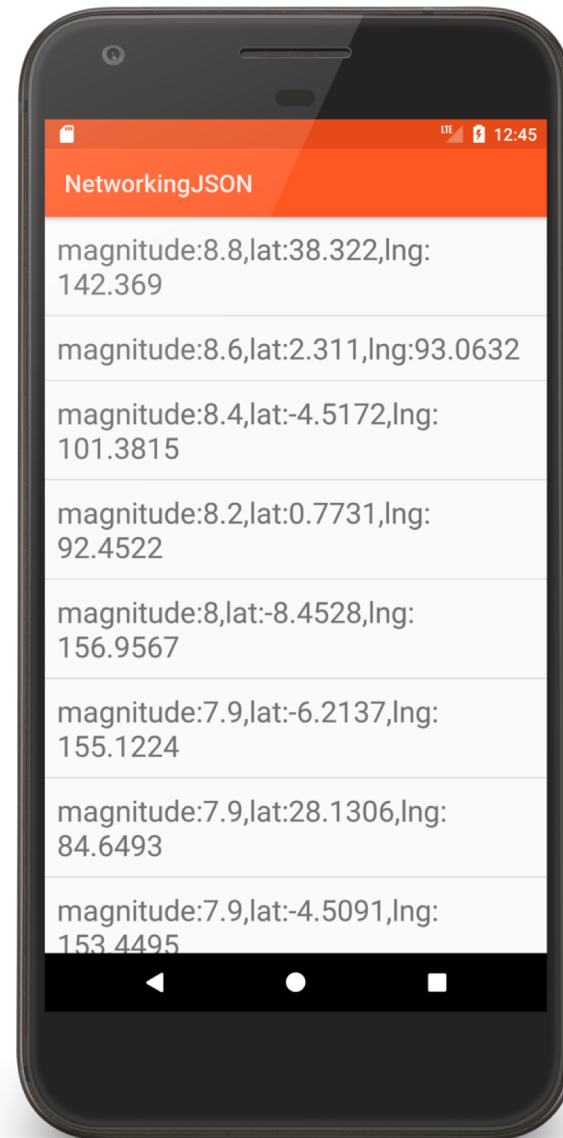
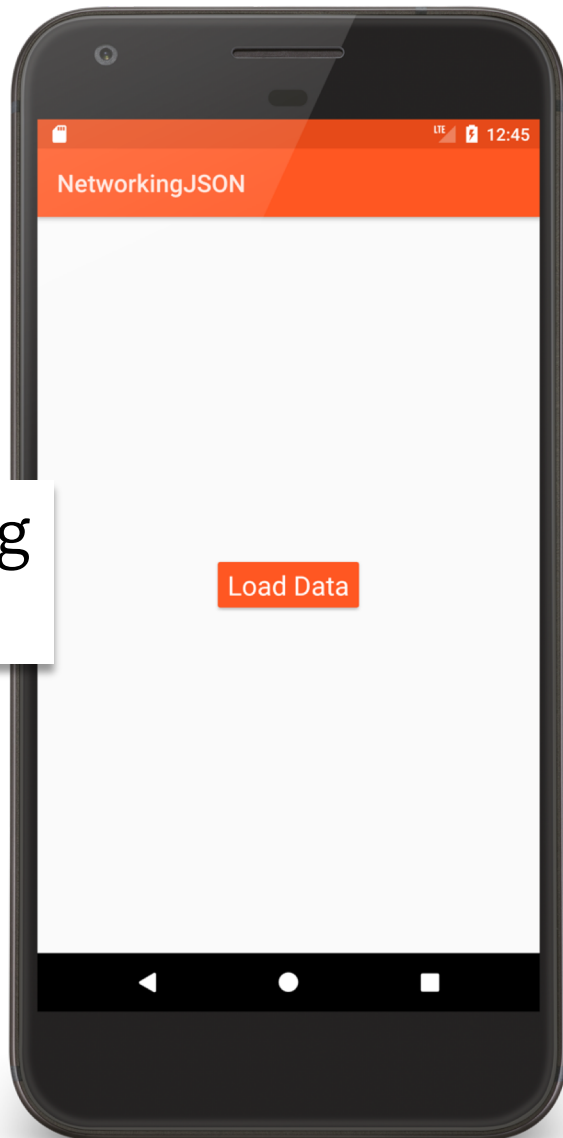
# Earthquake Data Request (JSON)

<http://api.geonames.org/earthquakesJSON?north=44.1&south=-9.9&east=-22.4&west=55.2&username=demo>

# JSON Response

```
{"earthquakes": [  
  {"eqid":"c0001xgp","magnitude":8.8,"lng":142.369, "src":"us", "datetime":"2011-  
03-11 04:46:23","depth":24.4,"lat":38.322}  
  {"eqid":"2007hear","magnitude":8.4,"lng":101.3815,"src":"us","datetime":"2007-  
09-12 09:10:26","depth": 30,"lat":-4.5172},  
  ...  
  {"eqid":"2010xkbv","magnitude":7.5,"lng":91.9379,"src":"us","datetime":"2010-  
06-12 17:26:50","depth":35,"lat":7.7477}  
]  
}
```

# Networking JSON



```
protected List<String> doInBackground(Void... params) {
```

```
...
```

```
    // 1. Get connection.
```

```
    //2. Prepare request (URI)
```

```
    // 3. This app does not use a request body
```

```
    // 4. Read the response
```

```
    // 5. Disconnect
```

```
    // 6. Parse the JSON-formatted response
```

```
    return parseJsonString(data);
```

```
}
```

```
private List<String> parseJsonString(String data) {
```

```
String LONGITUDE_TAG = "lng";
```

```
String LATITUDE_TAG = "lat";
```

```
String MAGNITUDE_TAG = "magnitude";
```

```
String EARTHQUAKE_TAG = "earthquakes";
```

```
List<String> result = new ArrayList<>();
```

```
try {
```

```
    // Get top-level JSON Object - a Map
```

```
    JSONObject responseObject = (JSONObject) new JSONTokener(data).nextValue();
```

```
    // Extract value of "earthquakes" key -- a List
```

```
    JSONArray earthquakes = responseObject.getJSONArray(EARTHQUAKE_TAG);
```

```
    ...
```



```
...
// Iterate over earthquakes list
for (int idx = 0; idx < earthquakes.length(); idx++) {
    // Get single earthquake mData - a Map
    JSONObject earthquake = (JSONObject) earthquakes.get(idx);
    // Summarize earthquake mData as a string and add it to result
    result.add(MAGNITUDE_TAG + ":" + earthquake.get(MAGNITUDE_TAG) + ","
        + LATITUDE_TAG + ":" + earthquake.getString(LATITUDE_TAG) + ","
        + LONGITUDE_TAG + ":" + earthquake.get(LONGITUDE_TAG));
}
} catch (JSONException e) {
    e.printStackTrace();
}
return result;
}
```

# eXtensible Markup Language (XML)

XML documents can contain markup & content

Markup encodes a description of the document's storage layout and logical structure

Content is everything else

See <http://www.w3.org/TR/xml>

# Earthquake Data (XML)

[http://api.geonames.org/earthquakes?north=44.1  
&south=-9.9&east=-22.4&  
west=55.2&username=demo](http://api.geonames.org/earthquakes?north=44.1&south=-9.9&east=-22.4&west=55.2&username=demo)

# XML Response

```
<geonames>  
  <earthquake>  
    <src>us</src>  
    <eqid>c0001xgp</eqid>  
    <datetime>2011-03-11 04:46:23</datetime>  
    <lat>38.322</lat>  
    <lng>142.369</lng>  
    <magnitude>8.8</magnitude>  
    <depth>24.4</depth>  
  </earthquake>  
  ...  
</geonames>
```

# Parsing XML

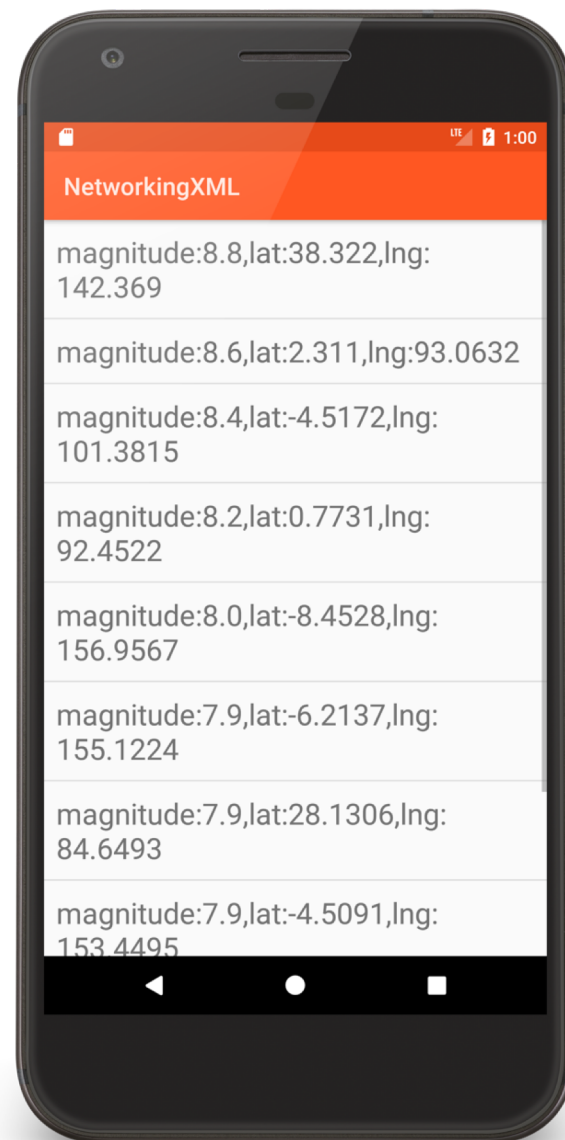
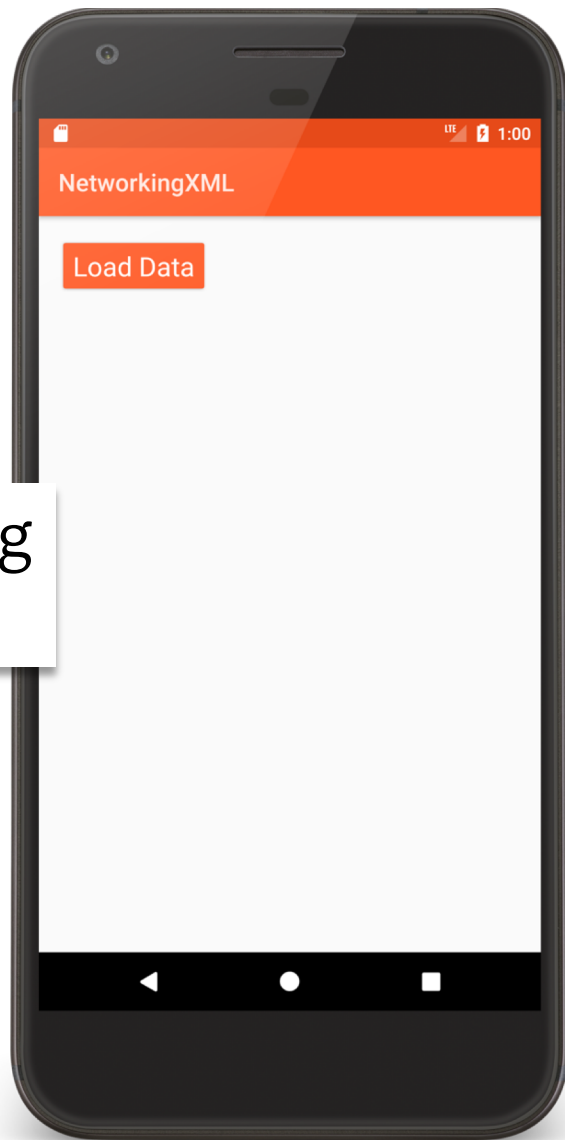
Several types of parsers available

DOM – Converts document into a tree of nodes

SAX – streaming with application callbacks

Pull – Application iterates over XML entries

# Networking XML



```
private List<String> parseXmlString(String data) {  
    try {  
        // Create the Pull Parser  
        XmlPullParserFactory factory = XmlPullParserFactory.newInstance();  
        XmlPullParser xpp = factory.newPullParser();  
        xpp.setInput(new StringReader(data));  
        // Get the first Parser event and start iterating over the XML document  
        int eventType = xpp.getEventType();  
        while (eventType != XmlPullParser.END_DOCUMENT) {  
            switch (eventType) {  
                case XmlPullParser.START_TAG: startTag(xpp.getName()); break;  
                case XmlPullParser.END_TAG: endTag(xpp.getName()); break;  
                case XmlPullParser.TEXT: text(xpp.getText()); break;  
            }  
            eventType = xpp.next();  
        }  
        return mResults;  
    }  
}
```

```
private void startTag(String localName) {  
    switch (localName) {  
        case LATITUDE_TAG:  
            mIsParsingLat = true; break;  
        case LONGITUDE_TAG:  
            mIsParsingLng = true; break;  
        case MAGNITUDE_TAG:  
            mIsParsingMag = true; break;  
    }  
}
```

```
private void text(String text) {  
    if (mIsParsingLat) { mLat = text.trim(); }  
    else if (mIsParsingLng) { mLng = text.trim(); }  
    else if (mIsParsingMag) { mMag = text.trim(); }  
}
```



```
private void endTag(String localName) {  
    switch (localName) {  
        case LATITUDE_TAG:  
            mIsParsingLat = false; break;  
        case LONGITUDE_TAG:  
            mIsParsingLng = false; break;  
        case MAGNITUDE_TAG:  
            mIsParsingMag = false; break;  
        case EARTHQUAKE_TAG:  
            mResults.add(MAGNITUDE_TAG + ":" + mMag + "," + LATITUDE_TAG + ":"  
                + mLat + "," + LONGITUDE_TAG + ":" + mLng);  
            mLat = null;  
            mLng = null;  
            mMag = null;  
            break;  
    }  
}  
...
```

# Next Time

Firestore

# Example Applications

NetworkingSockets

NetworkingURL

NetworkingJSON

NetworkingXML