# 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" />
```

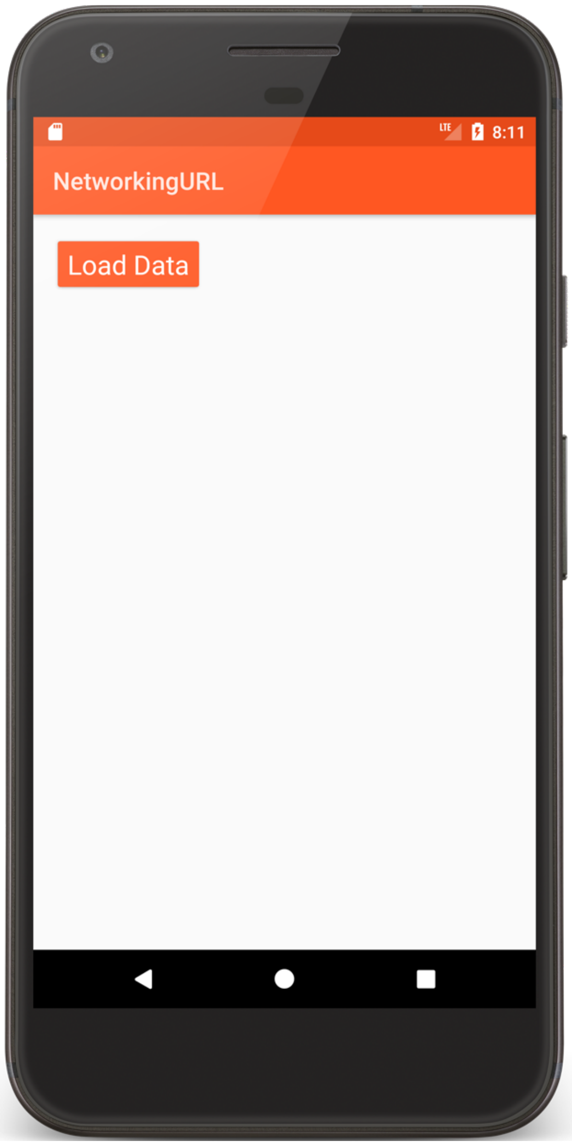
## HttpURLConnection

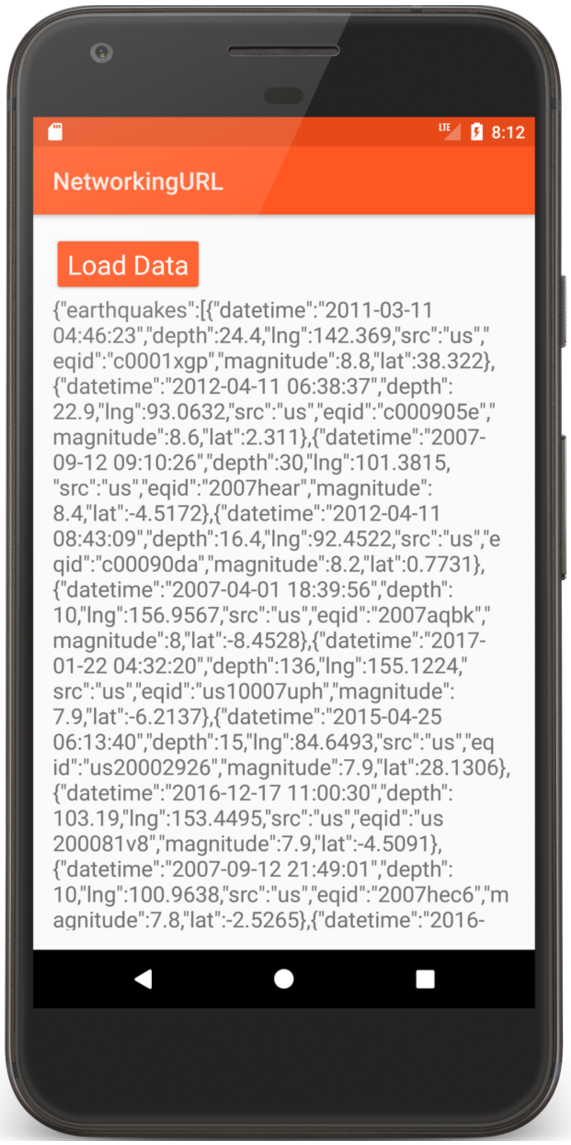
Higher-level than Sockets

## Usage Pattern

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

## Networking URL





```
{"earthquakes":[{"datetime":"2011-03-11
04:46:23","depth":24.4,"lng":142.369,"src":"us","
eqid":"c0001xgp","magnitude":8.8,"lat":38.322},
{"datetime":"2012-04-11 06:38:37","depth":
22.9,"lng":93.0632,"src":"us","eqid":"c000905e","
magnitude":8.6,"lat":2.311},{"datetime":"2007-
09-12 09:10:26","depth":30,"lng":101.3815,
"src":"us","eqid":"2007hear","magnitude":
8.4,"lat":-4.5172},{"datetime":"2012-04-11
08:43:09","depth":16.4,"lng":92.4522,"src":"us","e
qid":"c00090da","magnitude":8.2,"lat":0.7731},
{"datetime":"2007-04-01 18:39:56","depth":
10,"lng":156.9567,"src":"us","eqid":"2007aqbk","
magnitude":8,"lat":-8.4528},{"datetime":"2017-
01-22 04:32:20","depth":136,"lng":155.1224,"
src":"us","eqid":"us10007uph","magnitude":
7.9,"lat":-6.2137},{"datetime":"2015-04-25
06:13:40","depth":15,"lng":84.6493,"src":"us","eq
id":"us20002926","magnitude":7.9,"lat":28.1306},
{"datetime":"2016-12-17 11:00:30","depth":
103.19,"lng":153.4495,"src":"us","eqid":"us
200081v8","magnitude":7.9,"lat":-4.5091},
{"datetime":"2007-09-12 21:49:01","depth":
10,"lng":100.9638,"src":"us","eqid":"2007hec6","m
agnitude":7.8,"lat":-2.5265},{"datetime":"2016-
```

# RetainedFragment.kt

```
internal class HttpGetTask(retainedFragment: RetainedFragment) :
AsyncTask<Void, Void, String>() {
  companion object {
  …
    private const val URL = ("http://" + HOST +
    "/earthquakesJSON?north=44.1&south=-9.9&east=-22.4&west=55.2&username="
    + USER_NAME)
    }
  …
    override fun doInBackground(vararg params: Void): String? {
        var data: String? = null
        var httpUrlConnection: HttpURLConnection? = null
        try {
            // 1. Get connection. 2. Prepare request (URI)
            httpUrlConnection = URL(URL).openConnection()
                                                  as HttpURLConnection
```

## RetainedFragment.kt

```
        // 3. This app does not use a request body
        // 4. Read the response
        val inputStream = BufferedInputStream(
            httpUrlConnection.inputStream
        )
        data = readStream(inputStream)
    } catch (exception: MalformedURLException) {
        Log.e(TAG, "MalformedURLException")
    } catch (exception: IOException) {
        Log.e(TAG, exception.toString())
    } finally {
        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/

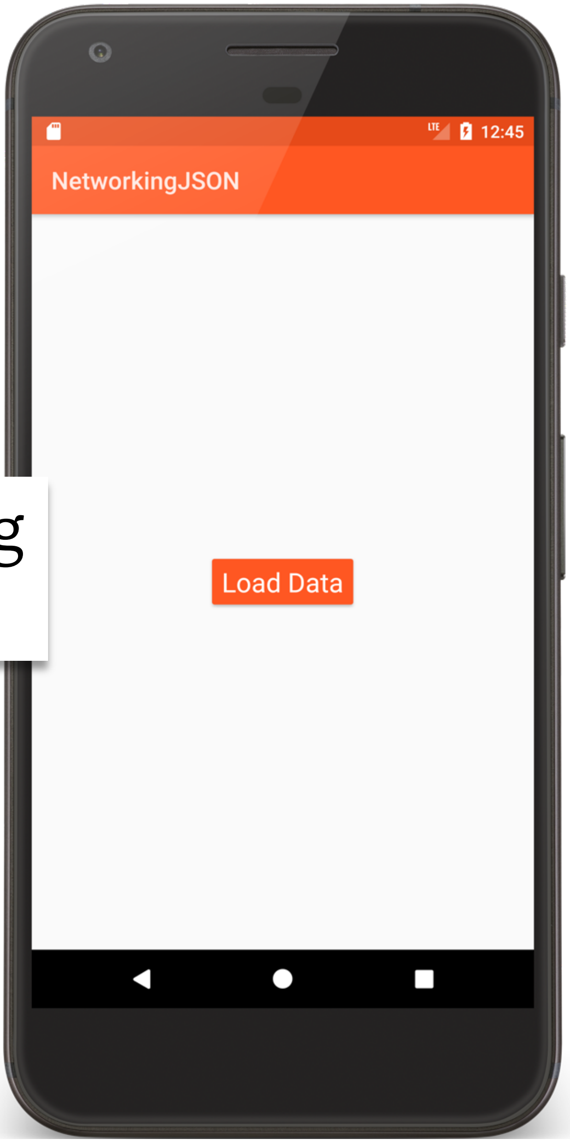
## Earthquake Data Request (JSON)

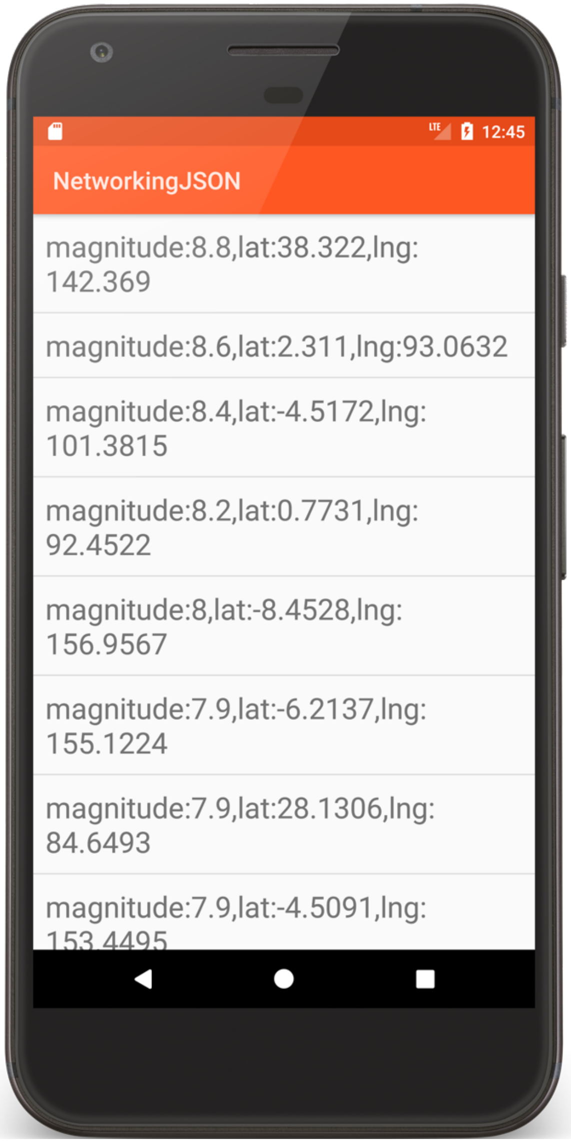
http://api.geonames.org/earthquakesJSON?north=44.1&south=-9.9&east=-22.4&west=55.2&username=demo

# 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





```
magnitude:8.8,lat:38.322,lng:
142.369
magnitude:8.6,lat:2.311,lng:93.0632
magnitude:8.4,lat:-4.5172,lng:
101.3815
magnitude:8.2,lat:0.7731,lng:
92.4522
magnitude:8,lat:-8.4528,lng:
156.9567
magnitude:7.9,lat:-6.2137,lng:
155.1224
magnitude:7.9,lat:28.1306,lng:
84.6493
magnitude:7.9,lat:-4.5091,lng:
153.4495
```

# RetainedFragment.kt

```
override fun doInBackground(vararg params: Void): List<String> {
  …
  // 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);
  }
```

## RetainedFragment.kt

```kotlin
private fun parseJsonString(data: String?): List<String> {
    val result = ArrayList<String>()
    try {
        // Get top-level JSON Object - a Map
        val responseObject = JSONTokener(data).nextValue() as JSONObject

        // Extract value of "earthquakes" key -- a List
        val earthquakes = responseObject.getJSONArray(EARTHQUAKE_TAG)

        // Iterate over earthquakes list
        for (idx in 0 until earthquakes.length()) {

            // Get single earthquake mData - a Map
            val earthquake = earthquakes.get(idx) as JSONObject
```

# RetainedFragment.kt

```
            // 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 (e: JSONException) {
        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

## 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

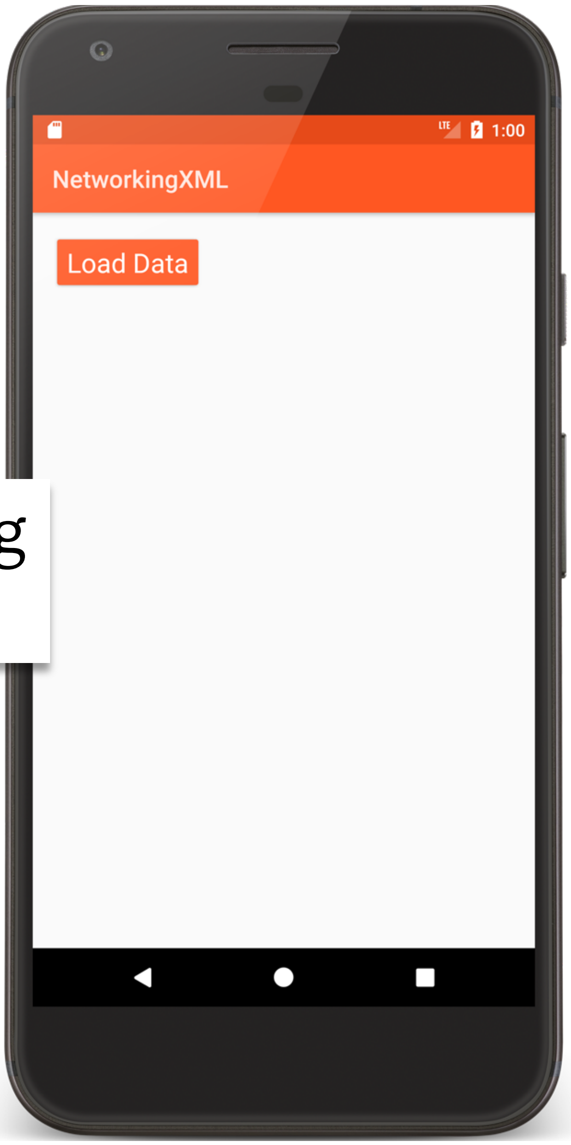
Several types of parsers available

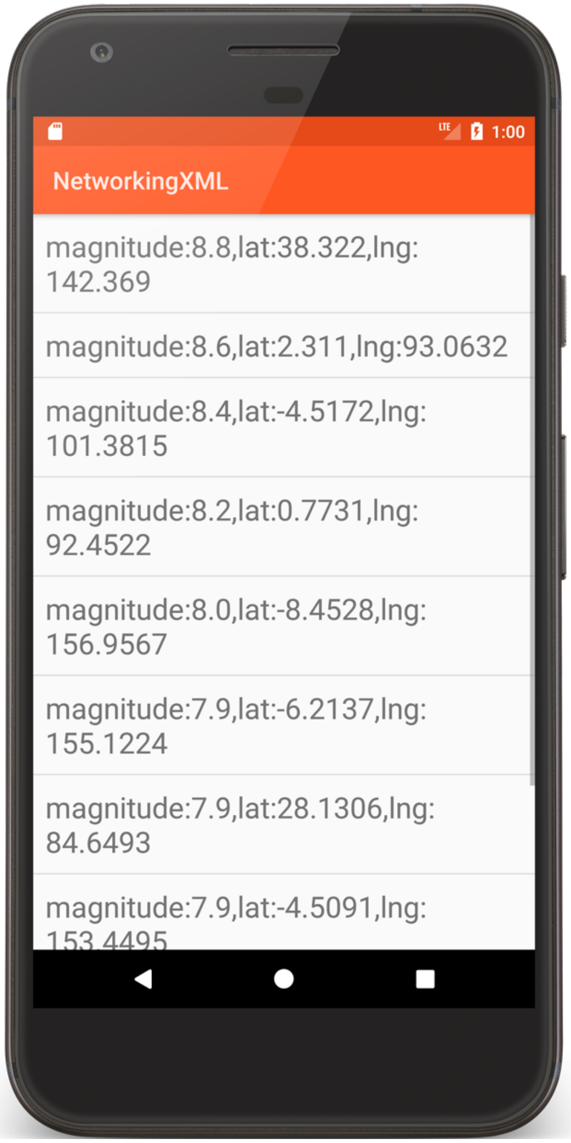
DOM – Converts document into a tree of nodes

SAX – streaming with application callbacks

Pull – Application iterates over XML entries

# Networking XML





```
magnitude:8.8,lat:38.322,lng:
142.369
magnitude:8.6,lat:2.311,lng:93.0632
magnitude:8.4,lat:-4.5172,lng:
101.3815
magnitude:8.2,lat:0.7731,lng:
92.4522
magnitude:8.0,lat:-8.4528,lng:
156.9567
magnitude:7.9,lat:-6.2137,lng:
155.1224
magnitude:7.9,lat:28.1306,lng:
84.6493
magnitude:7.9,lat:-4.5091,lng:
153.4495
```

## RetainedFragment.kt

```kotlin
private fun parseXmlString(data: String?): List<String>? {
    try {
      // Create the Pull Parser
      val factory = XmlPullParserFactory.newInstance()
      val xpp = factory.newPullParser()
      xpp.setInput(StringReader(data!!))

      // Get the first Parser event and start iterating over the XML document
      var eventType = xpp.eventType
      while (eventType != XmlPullParser.END_DOCUMENT) {
          when (eventType) {
            XmlPullParser.START_TAG -> startTag(xpp.name)
            XmlPullParser.END_TAG -> endTag(xpp.name)
            XmlPullParser.TEXT -> text(xpp.text)
          }
          eventType = xpp.next()
      }
```

## RetainedFragment.kt

```
 return mResults
    } catch (e: XmlPullParserException) {
        e.printStackTrace()
    } catch (e: IOException) {
        e.printStackTrace()
    }
    return null
}
```

## RetainedFragment.kt

```kotlin
private fun startTag(localName: String) {
    when (localName) {
        LATITUDE_TAG -> mIsParsingLat = true
        LONGITUDE_TAG -> mIsParsingLng = true
        MAGNITUDE_TAG -> mIsParsingMag = true
    }
}

private fun text(text: String) {
    when {
        mIsParsingLat -> mLat = text.trim { it <= ' ' }
        mIsParsingLng -> mLng = text.trim { it <= ' ' }
        mIsParsingMag -> mMag = text.trim { it <= ' ' }
    }
}
```

# RetainedFragment.kt

```kotlin
private fun endTag(localName: String) {
    when (localName) {
        LATITUDE_TAG -> mIsParsingLat = false
        LONGITUDE_TAG -> mIsParsingLng = false
        MAGNITUDE_TAG -> mIsParsingMag = false
        EARTHQUAKE_TAG -> {
            mResults.add(MAGNITUDE_TAG + ":" + mMag + "," + LATITUDE_TAG + ":"
                       + mLat + "," + LONGITUDE_TAG + ":" + mLng)
            mLat = null
            mLng = null
            mMag = null
        }
    }
}
```

## Next Time

Graphics and Animation

## Example Applications

NetworkingURL

NetworkingJSON

NetworkingXML