Networking

Programming the Android Platform

Networking

- Many applications provide & use data & services via the Internet
- Android includes multiple networking support classes, e.g.,
  - java.net – (Socket, URL)
  - org.apache - (HttpRequest, HttpResponse)
  - android.net – (URI, AndroidHttpClient,AudioStream)

Networking Permissions

- Typically need permission to access the Internet
  - <uses-permission android:name="android.permission.INTERNET" />

Using Sockets

```
public class NetworkingSocketsActivity extends Activity {
 TextView mTextView = null;
 public void onCreate(Bundle savedInstanceState) {
   ...
   // assuming webserver on 192.168.1.10, listening on port 80
   new HttpGet().execute("192.168.1.10");
 }
 private void onFinishGetRequest(String result) {
   mTextView.setText(result);
}
```

Using Sockets (cont.)

```
private class HttpGet extends AsyncTask<String, Void, String> {
 protected String doInBackground(String... params) {
  Socket socket = null;
  StringBuffer data = new StringBuffer();
  try {
   socket = new Socket(params[0], 80);
   PrintWriter pw = new PrintWriter(
        new OutputStreamWriter(socket.getOutputStream()),true);
   pw.println("GET /json_example.txt");
   ...
```

Using Sockets (cont.)

```
        ...
        BufferedReader br = new BufferedReader(
            new InputStreamReader( socket.getInputStream()));
        String rawData;
        while ((rawData = br.readLine()) != null) {
          data.append(rawData);
        }
      } catch ...
      // close socket  ...
      return data.toString();
    }
    protected void onPostExecute(String result) {
      onFinishGetRequest(result);
    }
...
```

Using URLConnection

```
public class NetworkingURLActivity extends Activity {
 TextView mTextView = null;
 public void onCreate(Bundle savedInstanceState) {
   new HttpGetTask().execute( "http://api.geonames.org/...");
 }
 ...
```

Using URLConnection (cont.)

```
…
 private class HttpGetTask extends AsyncTask<String, Void, String> {
    protected String doInBackground(String… params) {
     StringBuffer data = new StringBuffer();
     BufferedReader br = null;
     try {
       HttpURLConnection  conn =
            (HttpURLConnection) new URL(params[0]).openConnection();
      // read & process response
     }
     // close outputStream
     return data.toString();
}
…
```

Using HTTPClient

```
...
 private class HttpGetTask extends AsyncTask<String, Void, String> {
  protected String doInBackground(String... params) {
   AndroidHttpClient client = AndroidHttpClient.newInstance("");
   HttpGet request = new HttpGet(params[0]);
   ResponseHandler<String> responseHandler =
                                          new BasicResponseHandler();
   try {
    return client.execute(request, responseHandler);
   } catch ...
   return null;
 }
...
```

Parsing Http Responses

- Several popular formats including
  - JSON
  - XML

JSON

- Javascript Object Notation
  - http://www.json.org/
- Intended to be a lightweight data interchange format
- Data packaged in two types of structures:
  - Maps of key/value pairs
  - Ordered lists of values

Earthquake Data (JSON Output)

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

- Produces

{"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}
]
}

Parsing JSON

```
class HttpGetTask extends AsyncTask<String, Void, List<String>> {
 protected List<String> doInBackground(String... params) {
  AndroidHttpClient client = AndroidHttpClient.newInstance("");
  HttpGet request = new HttpGet(params[0]);
  JSONResponseHandler responseHandler =
                                          new JSONResponseHandler();
  try {
   return client.execute(request, responseHandler);
  } catch …
  return null;
 }
…
```

Parsing JSON (cont.)

```
class JSONResponseHandler implements ResponseHandler<List<String>> {
 public List<String> handleResponse(HttpResponse response) throws
                                                ClientProtocolException, IOException {
 List<String> result = new ArrayList<String>();
 String JSONResponse =
                        new BasicResponseHandler().handleResponse(response);
 try {
  JSONObject object =
            (JSONObject) new JSONTokener(JSONResponse).nextValue();
  JSONArray earthquakes = object.getJSONArray("earthquakes");
  for (int i = 0; i < earthquakes.length(); i++) {
   JSONObject tmp = (JSONObject) earthquakes.get(i);
   result.add("mag:" + tmp.get("magnitude") +
                          " lat:" + tmp.getString("lat") + " lng:" + tmp.get("lng"));
   }
 } catch (JSONException e) { … }
return result;
…
```

XML

- eXtensible Markup Language
  - See http://www.w3.org/TR/xml/
- XML documents
  - made up of storage units called entities
  - Entities can contain markup, which encodes a description of the document's storage layout and logical structure

Parsing XML

- Several types of parsers available
  - SAX – streaming with application callbacks
  - Pull – Application iterates over XML entries
  - DOM – Coverts document into a tree of nodes

Earthquake Data (XML)

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

```
<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 (Sax-Based)

```
...
private class XMLResponseHandler implements
                                        ResponseHandler<List<String>> {
  public List<String> handleResponse(HttpResponse response)
                                  throws ClientProtocolException, IOException {
    try {
      SAXParserFactory spf = SAXParserFactory.newInstance();
      SAXParser sp = spf.newSAXParser();
      XMLReader xr = sp.getXMLReader();
      XMLContentHandler handler = new XMLContentHandler();
      xr.setContentHandler(handler);
      xr.parse(new InputSource(response.getEntity().getContent()));
      return handler.getData();
    } catch ...
    return null;
  }
}...
```

Parsing XML (Sax-Based)

```
private class XMLContentHandler extends DefaultHandler {
  String lat = null, lng = null, mag=null;
  boolean parsingLat = false, parsingLng = false, parsingMag=false;
  List<String> results = new ArrayList<String>();

  public void startElement(String uri, String localName,
            String qName,Attributes attributes) throws SAXException {
    if (localName.equals("lat")) {
      parsingLat = true;
    } else if (localName.equals("lng")) {
      parsingLng = true;
    } else if (localName.equals("magnitude")) {
      parsingMag = true;
    }
}
```

Parsing XML (Sax-Based)

```
public void characters(char[] ch, int start, int length)
                                                    throws SAXException {
  if (parsingLat) {
    lat = new String(ch, start, length).trim();
  } else if …
}

public void endElement(String uri, String localName, String qName) {
  if (localName.equals("lat")) {
    parsingLat = false;
  } else if …
  } else if (localName.equals("earthquake")) {
    results.add("lat:" + lat + " lng: " + lng + " mag:" + mag);
    lat = null; lng = null; mag = null;
  }
}

public List<String> getData() { return results; }
…
```

Lab Assignment

Source Code Examples

- NetworkingAndroidHttpClient
- NetworkingAndroidHttpClientJSON
- NetworkingAndroidHttpClientXML
- NetworkingSockets
- NetworkingURL