Lecture Networks: Remote data access

DT228/3

Dr. Susan McKeever

So far, all processing has been local to the phone

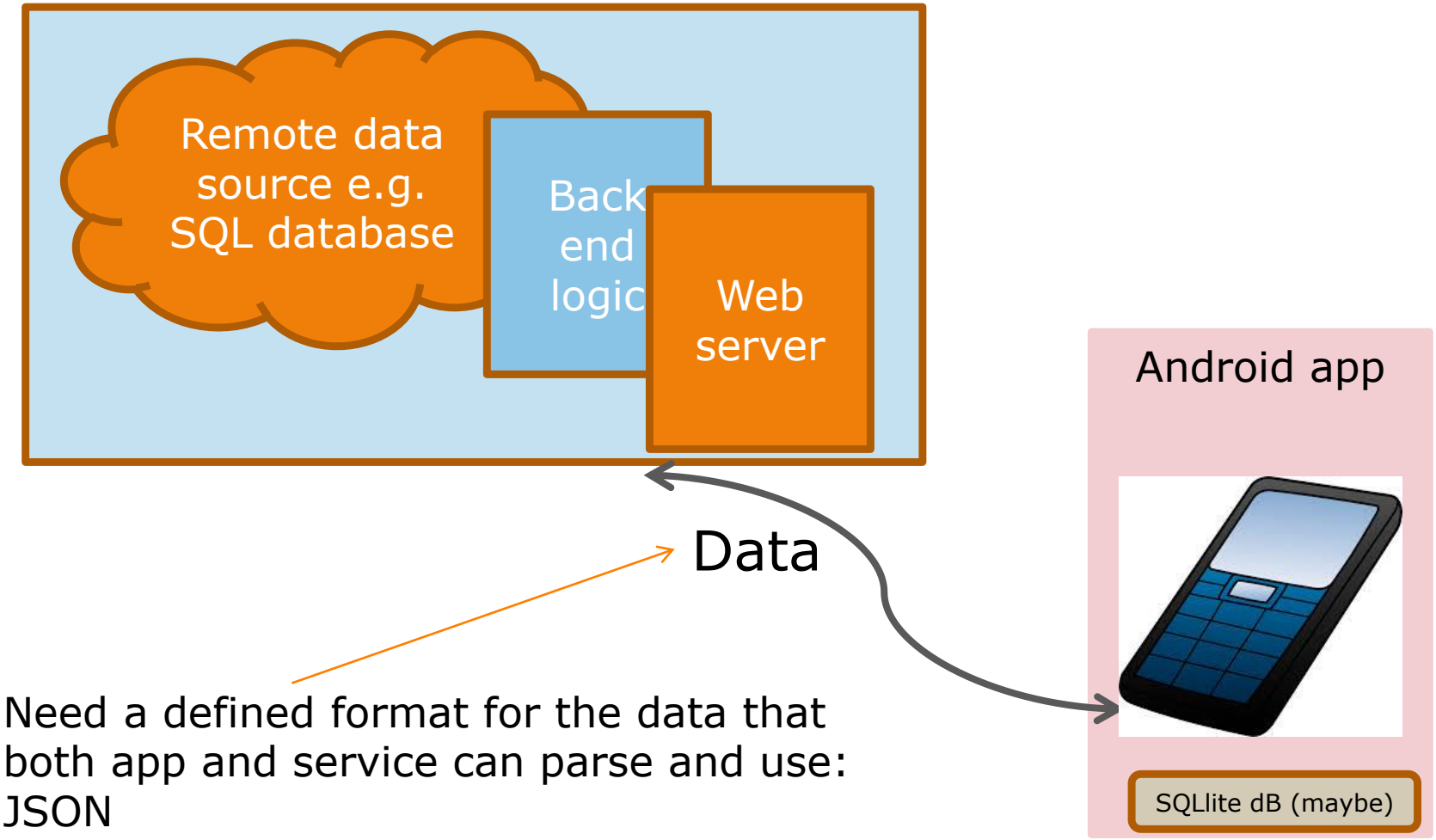
SQLlite local database.

But may be requirement to use “remote” resources (i.e. on the cloud).

e.g. news apps, social media apps etc

Many apps need remote data storage

Architecture components: App with remote data



Need a defined format for the data that both app and service can parse and use: JSON

Concerns

Privacy concerns

Device performance degradation

Unwanted network data charges

Opportunity for app to fail if network access not guaranteed

Testing: On emulator, computer network connection used – likely to be faster than real phone

Access to remote data resources:

1. Need a Network connection

2. Need to use a Network Protocol..
   - Usually HTTP (remember GET and POST?)

3. Need to understand format of data being sent back from (or to) server?
   - Output of a DB query? (String/XMl/JSON)
   - HTML page? Etc
   - This is where the most variation occurs when coding networked apps

Manifest file permissions

(1) Add the permission "android.permission.INTERNET".

`<uses-permissions>` tag

(2) Good practice in your code to check if network connection available - so this too:

Add the permission "android.permission.ACCESS_NETWORK_STATE".

With `<uses-permissions>` tag

Example: Retrieve data (i.e. webpage) from a particular URL Various steps

- Check if there is a valid network connectivity (good practice)

- Then..Make a HTTP connection

  - Getting a network connection is slow!
  - Use AsyncTask*
  - See code sample..

* In fact, As of Android 3.0 (Honeycomb) the system is configured to crash with a NetworkOnMainThreadException exception, if network is accessed in the main user interface thread

Checking if the phone has a valid network connection

Where else was getSystemService() used?

```
ConnectivityManager connMgr =      (ConnectivityManager)
     getSystemService(Context.CONNECTIVITY_SERVICE);

NetworkInfo networkInfo =          connMgr.getActiveNetworkInfo();

if (networkInfo != null && networkInfo.isConnected())
{
         // go ahead with whatever network task is to be done//
}
else
{
   // Display an error

}+
```

An example..

1. How many classes are in here?
2. What superclasses are involved here?
3. Where is the button listener used?
4. What check is being done in the onClick method?
5. What is “NetworkInfo”?
6. what does the “doInBackground” method do?
7. where/how is the “doInBackground” method called from?
8. What is onPostExecute doing? How is it called?
9. What work, if any, is done on parsing out the response information?
10. What is “conn” referring to in the code?
11. What is an inputStream as used here?
12. What is the `readIt` method doing?
13. What is the textview being set to ?

Various different classes for making HTTP connections

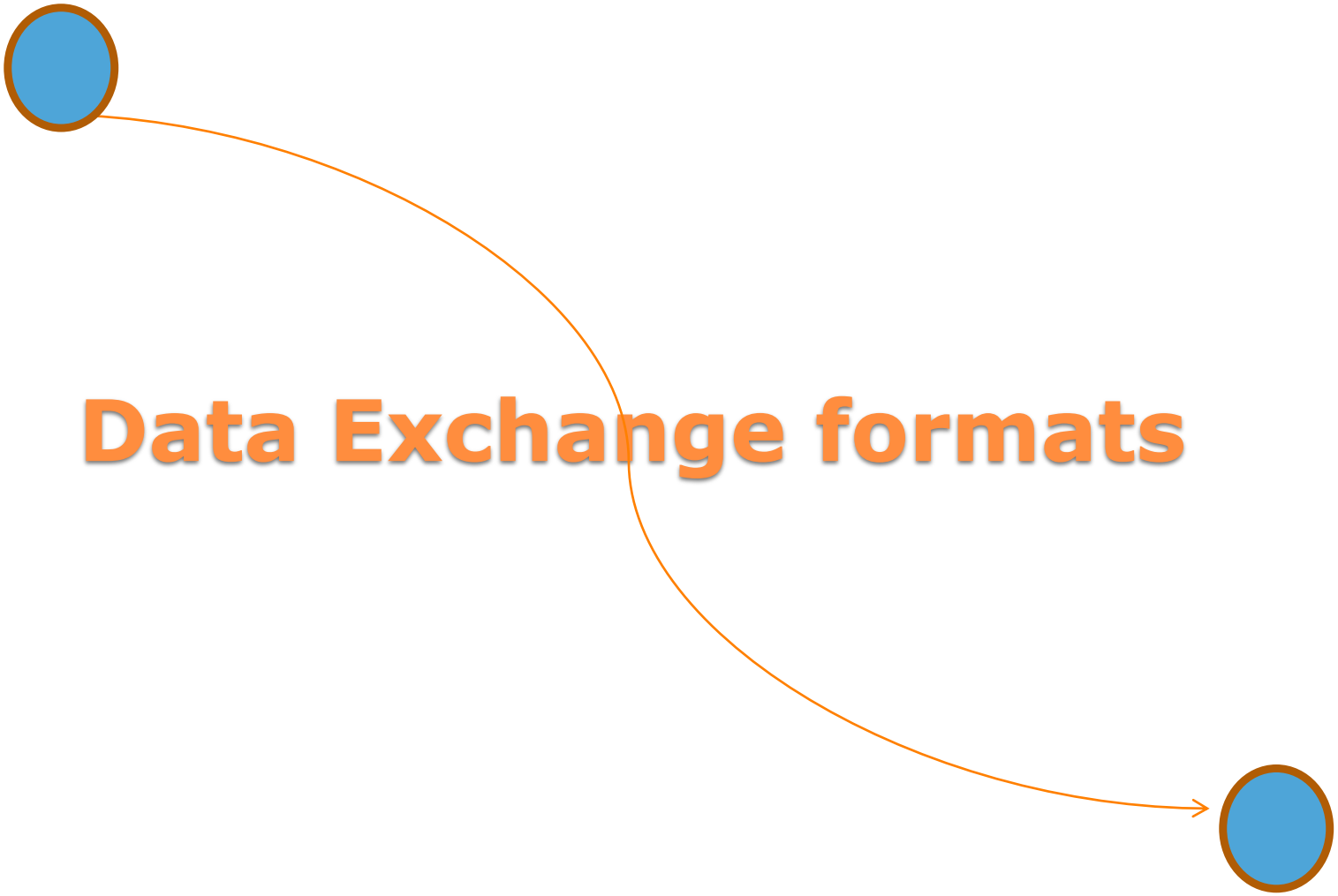
HttpURLConnection (as shown in code sample..)

The following two do the same thing - but are more abstracted:

HttpClient

DefaultHttpClient (deprecated in API 23)

Data Exchange formats



What is JSON?

- JSON stands for **J**ava**S**cript **O**bject **N**otation
- JSON is lightweight text-data interchange format
- JSON is language independent *
- JSON is "self-describing" and easy to understand

JSON uses JavaScript syntax for describing data objects, but JSON is still language and platform independent.

JSON parsers and JSON libraries exists for many different programming languages.

Sound familiar?

Sample of JSON

```
{
        "employees": [
        { "firstName":"John" , "lastName":"Doe" },
        { "firstName":"Anna" , "lastName":"Smith" },
        { "firstName":"Peter" , "lastName":"Jones" }
        ]
}
```

“Maps” to a relational dB table – how?

.

More..

```
{
  “Task  List”: [
       { “Task":“Shop"  , “Description":”Groceries“,
                                Complete: “No”},
       { “Task":“Review"  , “Description":”Homework“,
                                Complete: “Yes”},
       { “Task":“Tidy up"  , “Description":”House work“,
                                Complete: “No”},
    ]
}
```

JSON versus XML for sending/receiving data remotely

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
</popup> </menu>
```

XML

JSON versus XML for sending/receiving data remotely

```
{"menu": {
      "id": "file",
       "value": "File",
         "popup": {
         "menuitem": [
         {"value": "New", "onclick": "CreateNewDoc()"},
         {"value": "Open", "onclick": "OpenDoc()"},
         {"value": "Close", "onclick": "CloseDoc()"}
         ]
      }
   }
}
```

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
</popup> </menu>
```

JSON versus XML for sending/receiving data remotely

JSON

•+Less verbose, so takes up less space
•+ less bandwidth
•+ Simpler grammar
•+Self describing – Not extensible as such – it doesn't need to be
!Tags/attributes are irrelevant
•+ Data stored in arrays - easier for OO languages to retrieve
•+ Good for data exchange
•Support for built into programming languages e.g. java script/ java

-Now main stream
- Limited to data exchange

XML

+ Easier to read by human (arguably)
+ Longer term standard
+ Self describing
+ good for document exchange
+ Parsers available – DOM, SAX etc.

-Easy to read (human)
-More verbose than JSON
-Was mainstream
-Data stored in trees with tags/attributes: potentially more difficult to parse than JSON

JSON versus XML for sending/receiving data remotely

If you are not familiar with JSON format, look at these links:

https://www.w3schools.com/js/js_json_syntax.asp

and if you want published info on difference to XML, read this:

https://www.w3schools.com/js/js_json_xml.asp

JSON

JSON objects are constructed in a *key*:*value* pair format. The object's elements are separated by commas, and each object is contained within curly braces

An array of values (e.g. objects) is contained within square brackets **[ ]**

```
{
        "employees": [
        { "firstName":"John" , "lastName":"Doe" },
        { "firstName":"Anna" , "lastName":"Smith" },
        { "firstName":"Peter" , "lastName":"Jones" }
        ]
}
```

How many arrays here? How many objects?

Parsing Json (and beware of arrays vs objects

JSONArray
[- represents it's a json **Array** node

```
{
        "contacts": [
            {
                "name": "Terry Obrien"
                "mobile" : "5548875"
                 "gender" :"male"
            }
            {
                 "name": "joe"
                 "mobile":"0990#2"
                 "gender":"female"
            }
}
```

JSONObject
{ - represents it's a json object node

Key : value pairs

JSONObject versus JSONarray

[....] is an JSON array of values e.g. **[** "android" , "httpclient" , "internet" **]**

{....} is an JSON object of key:value pairs e.g. **{** tags:[...] , categories:[...] , url:"…" , title:"…" **}**

JSONObject can contain JSONArray

JSONArray can contain JSONObject

Easier if you need to know the structure you're processing

Parsing Json

JSON data – coming from a remote server

Might want to :

- save to app dB
- Display it in a list or textview
- Count it

ETC.

So might want to parse it

A variety of support classes available e.g. JSONarray. JSONObject, JSONTokeniser and more..

Parsing Json example (and beware of arrays vs objects.

```
{
   "employees": [
       { "firstName":"John" , "lastName":"Doe" },
       { "firstName":"Anna" , "lastName":"Smith" },
       { "firstName":"Peter" , "lastName":"Jones" }
       ]
}
```

To parse this JSON data:

Open array “Employees”

Then.. from 1 to last array entry: get all the objects (key/value pairs).. and store them

Parsing Json (and beware of arrays vs objects.

```
{
   "employees": [
        { "firstName":"John" , "lastName":"Doe" },
        { "firstName":"Anna" , "lastName":"Smith" },
        { "firstName":"Peter" , "lastName":"Jones" }
        ]
}
```

To parse this JSON data:

Open array "Employees"

Then.. from 1 to last array entry: get all the objects (key/value pairs).. and store them

```
JSONarray employees = new JSONArray(jsonstringname);
for (int i = 0; i < employees.length(); i++)
{
  JSONObject  jsonData= employees.getJSONObject(i);
  String firstName = jsonData.getString("firstName");
  String lastName = jsonData.getString("lastName");
}
```

Writing JSON

So far... looked at receiving JSON from remote source and parsing it.

But may want to send app data to remote source.

... need to create or write JSON

Scenarios..?

Writing JSON

Writing JSON is very simple. Just create the JSONObject or JSONArray and use the toString() method.

```
public void writeJSON()
{
JSONObject object = new JSONObject();
try {
   object.put("name", "Jack Murphy");
   object.put("score", new Integer(200));
   object.put("current", new Double(152.32));
   object.put("nickname", “Beano"); }
catch (JSONException e)
      { e.printStackTrace(); }
      System.out.println(object); }
```

HTTP messages from server contain a status code -

```
 int okayCode = 200;
…..
 HttpResponse response = client.execute(httpGet);
 StatusLine statusLine = response.getStatusLine();
 int statusCode = statusLine.getStatusCode();
 if (statusCode == okayCode) {
      HttpEntity entity = response.getEntity();
       etc….
```

Note: Other HTTP libraries

Looked at networking fundamentals in the core Android API

There is also:

Volley is an HTTP library that makes networking for Android apps:

- easier and
- faster.

Volley is available through the open AOSP repository (not suitable for large downloads)

Summary

Most apps require network access of some form

Client server protocol is usually HTTP

Data exchange format is usually JSON

JSON versus XML – overlaps/ differences

JSON – parsing it… writing it.