http://www.android.com/

Interacting with servers

Consuming Webservices

XML vs. JSON

Using libraries

Communication with server

- Mobile apps frequently have a server connected component
  - The server access can provide many services as
    - Community collaboration, data backup, data feeds (in both ways), intensive processing etc.
- Android have good support for networking (java.net.*, javax.net.* and android.net.*)
  - SSL support for mobile, URL sanitizers etc.
- Querying the network status of the different interfaces with the ConnectivityManager and Networkinfo
  - On Wi-Fi or mobile network, roaming, network available, etc.

```java
// The ConnectivityManager can:
// Monitor network connections (Wi-Fi, GPRS, UMTS, etc.)
// Send broadcast intents when network connectivity changes
// Attempt to "fail over" to another network when connectivity to a network is lost
// Provide an API that allows applications to query the coarse-grained or fine-grained state of the available networks
public boolean isNetworkAvailable() {
	ConnectivityManager cm = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
	NetworkInfo networkInfo = cm.getActiveNetworkInfo();
	// if no network is available networkInfo will be null otherwise check if we are connected
	if (networkInfo != null && networkInfo.isConnected()) {
		return true;
	}
	return false;
}
```

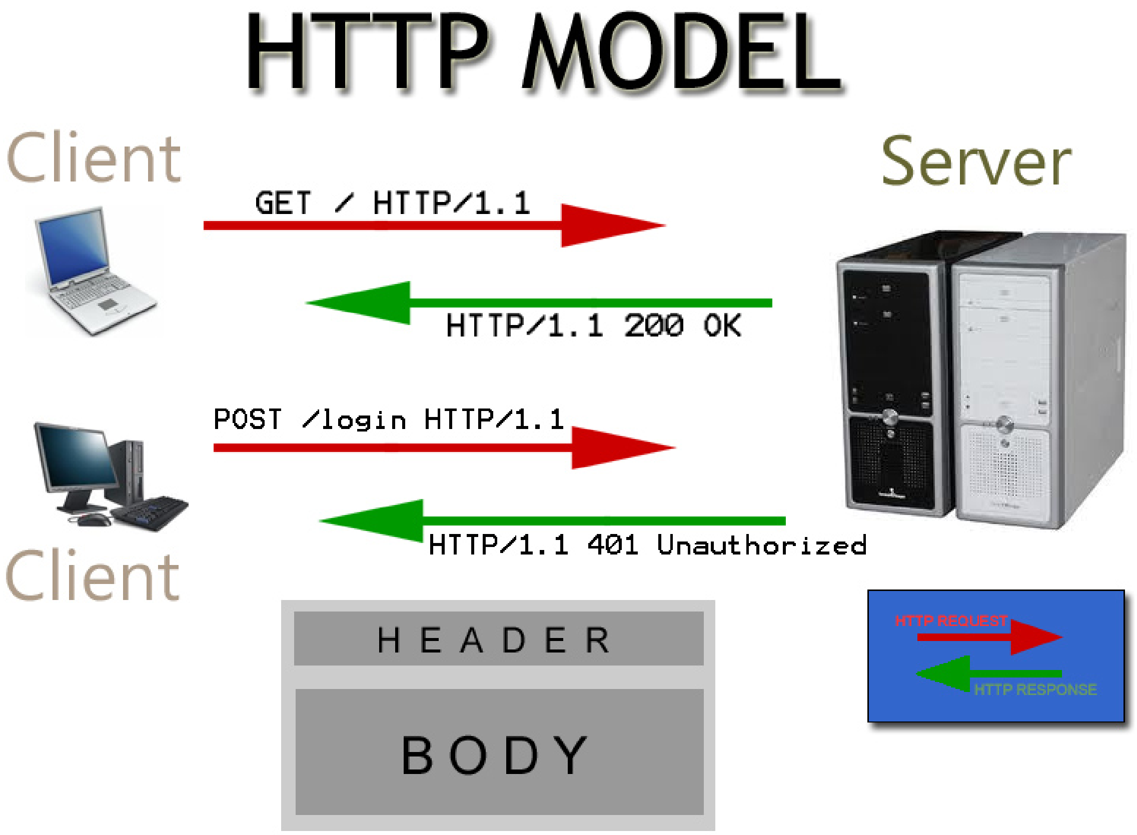
See the utils.Connectivity class in the jsonokhttp.7z app!

Server communication options

- Socket based communication (java.net.socket)
  - Direct control low-level
  - Any protocol can be implemented
- HTTP protocol
  - java.net.HttpURLConnection or javax.net.ssl.HttpsURLConnection
  - Note! org.apache.http.* is deprecated from API 23 and higher
- Web services
  - Widely adopted solution for InterApplication Communication (IAC)
  - Built on XML or JSON and HTTP communication
  - Android provides support for clients to Web services
  - Parsing and creating data-interchange via org.json.*, android.util.Json***, org.xml.*, android.util.xml and javax.xml.*

The HTTP protocol

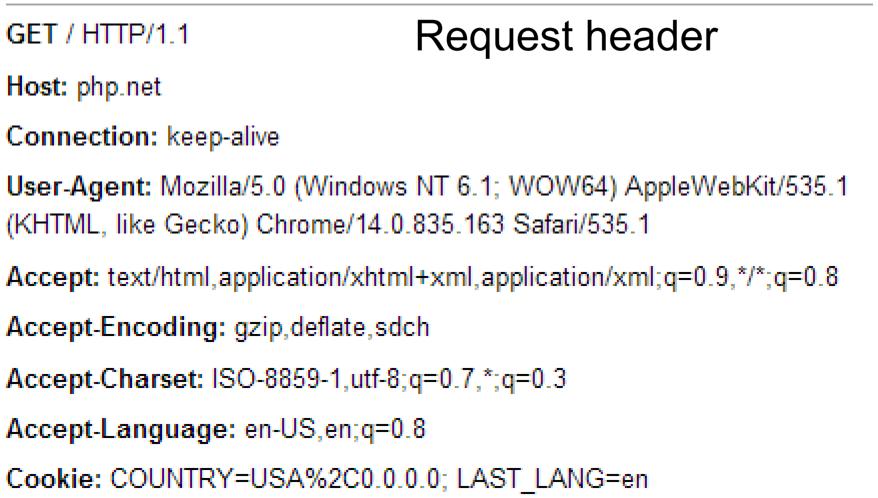
- TCP/IP based request/response protocol
- HTTP requests (known as methods)
  - GET (retrive/select) or POST (create/insert)
- HTTP response
  - In all cases a resonse code
  - will be returned
- HTTP message
  - **Request/response line** - the http method/status
  - **Header variables** - request metadata
  - **Message body** - content of message



HTTP GET request and response

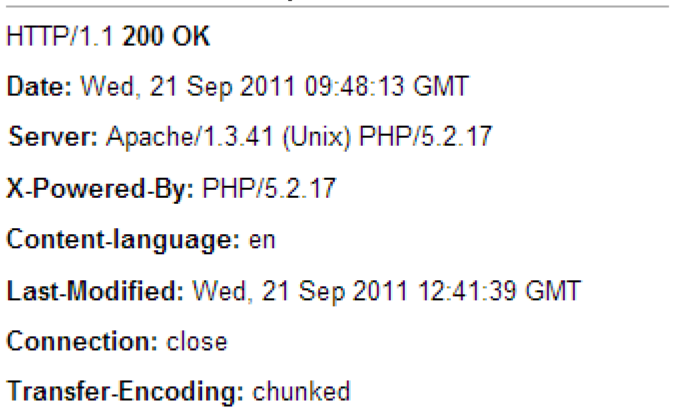
- The very first line of the request header is made up of three parts. The first is the verb (in this case GET) and the second is the URI/path (in this case / or the root path) and the third indicates which version of the HTTP protocol the machine understands (in this case HTTP 1.1)
- The second line of the request header indicates the HOST (in this case php.net). The next few lines tell us a little about the connection the client is trying to make, the User-Agent information supplied by the client and some acceptable attributes for the information they expect to receive back from the server. It may also include additional optional headers like a query string, cookies, etc...
- http://wiki.hashphp.org/HttpPrimer

Request header



```
GET / HTTP/1.1
Host: php.net
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.1
(KHTML, like Gecko) Chrome/14.0.835.163 Safari/535.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate,sdch
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
Accept-Language: en-US,en;q=0.8
Cookie: COUNTRY=USA%2C0.0.0.0; LAST_LANG=en
```

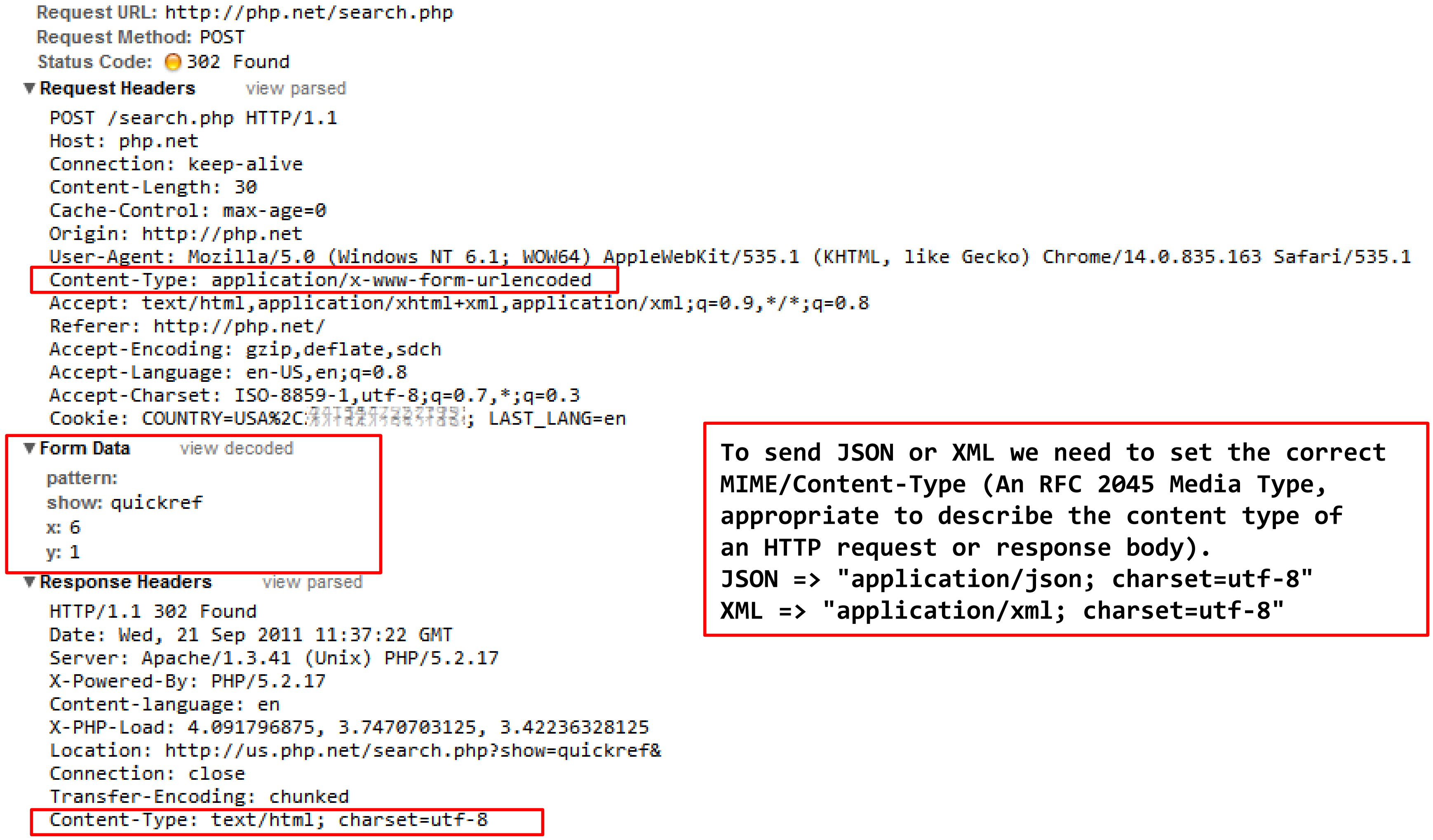
Response header



```
HTTP/1.1 200 OK
Date: Wed, 21 Sep 2011 09:48:13 GMT
Server: Apache/1.3.41 (Unix) PHP/5.2.17
X-Powered-By: PHP/5.2.17
Content-language: en
Last-Modified: Wed, 21 Sep 2011 12:41:39 GMT
Connection: close
Transfer-Encoding: chunked
```

HTTP POST request and response

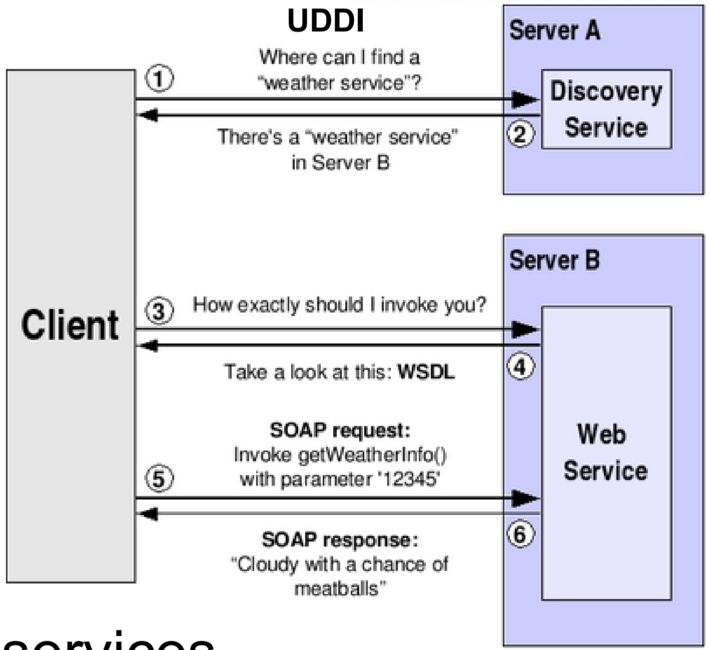
- In this request we a have section called Form Data (or POST body) and it contains information we want to give the server this time



```
Request URL: http://php.net/search.php
Request Method: POST
Status Code: 302 Found
▼Request Headers      view parsed
  POST /search.php HTTP/1.1
  Host: php.net
  Connection: keep-alive
  Content-Length: 30
  Cache-Control: max-age=0
  Origin: http://php.net
  User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.1 (KHTML, like Gecko) Chrome/14.0.835.163 Safari/535.1
  Content-Type: application/x-www-form-urlencoded
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
  Referer: http://php.net/
  Accept-Encoding: gzip,deflate,sdch
  Accept-Language: en-US,en;q=0.8
  Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
  Cookie: COUNTRY=USA%2C[redacted]; LAST_LANG=en
▼Form Data      view decoded
  pattern:
  show: quickref
  x: 6
  y: 1
▼Response Headers      view parsed
  HTTP/1.1 302 Found
  Date: Wed, 21 Sep 2011 11:37:22 GMT
  Server: Apache/1.3.41 (Unix) PHP/5.2.17
  X-Powered-By: PHP/5.2.17
  Content-language: en
  X-PHP-Load: 4.091796875, 3.7470703125, 3.42236328125
  Location: http://us.php.net/search.php?show=quickref&
  Connection: close
  Transfer-Encoding: chunked
  Content-Type: text/html; charset=utf-8
```

```
To send JSON or XML we need to set the correct
MIME/Content-Type (An RFC 2045 Media Type,
appropriate to describe the content type of
an HTTP request or response body).
JSON => "application/json; charset=utf-8"
XML => "application/xml; charset=utf-8"
```

Web services


UDDI
Server A
Discovery Service
Server B
Web Service
Client
① Where can I find a "weather service"?
② There's a "weather service" in Server B
③ How exactly should I invoke you?
④ Take a look at this: WSDL
⑤ SOAP request: Invoke getWeatherInfo() with parameter '12345'
⑥ SOAP response: "Cloudy with a chance of meatballs"

- Web services connect distributed applications
  - Interoperable, cross-platform, cost-effective, …
- A web service can be regarded as a web application but without a view
  - The client is the view
- Two styles of web services
- Simple Object Access Protocol (SOAP) services
  - Highly structured XML messages communicated over HTTP
  - Clearly defined service interface published in WSDL
  - Ideally suited to integrating enterprise applications
- Represental State Transfer (REST) services
  - Lightweight informal services
  - Data typically in XML or JSON format
  - Easy to consume from Web browsers and mobile clients

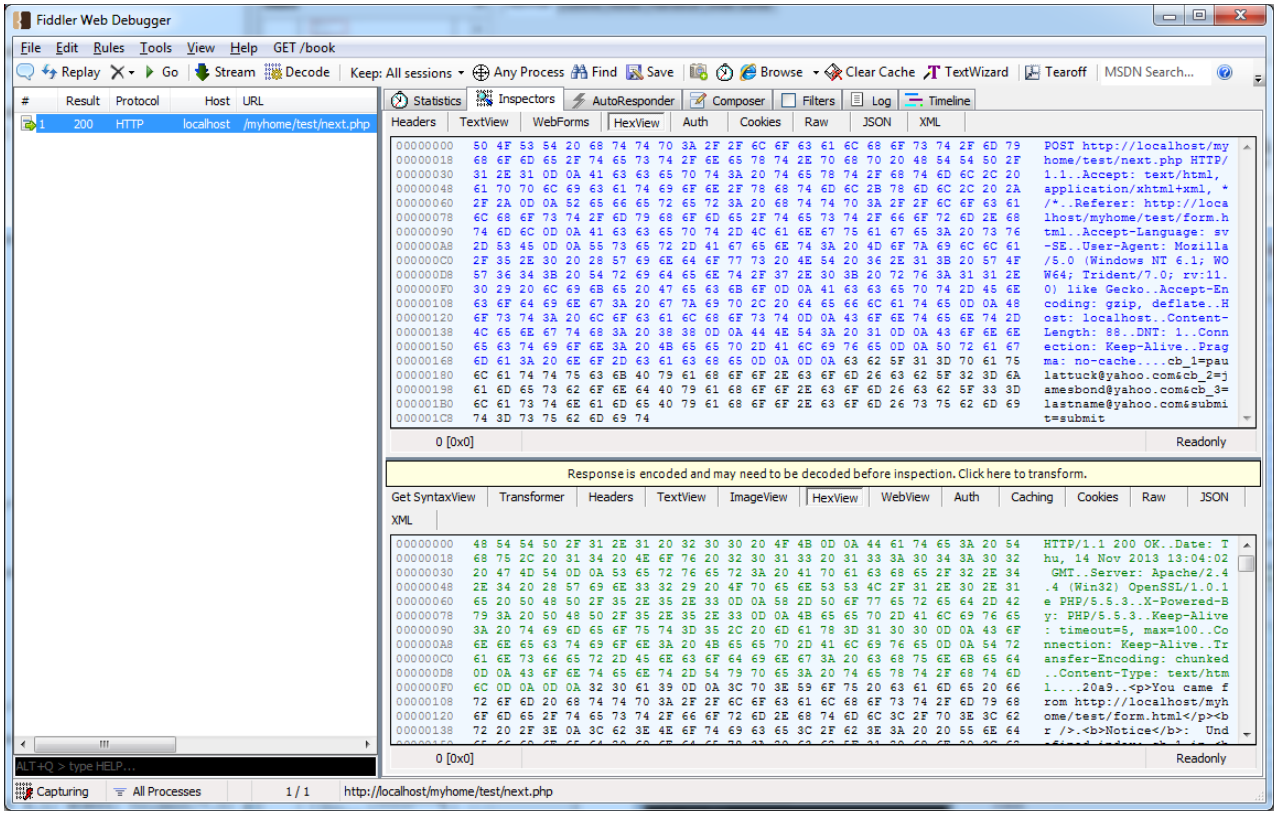
Web service client

- Use a HTTP GET/POST capable client
- Create a valid request from application data
- Send the request and parse the results
  - Message data must be converted to/from Java data types
- Android have built-in support for both XML via
  - Standard Java XML processing as
    - SAX (Simple API for XML)
    - DOM (Document Object Model) and
    - XML Pull parsers
- and JSON via org.json classes
  - JSON (JavaScript Object Notation) is the mosts common for REST-based services
  - A lightweight data-interchange format which is completely language independent
  - Easy for humans to read and write and easy for machines to parse and generate

Debug with tools like Fiddler

- Fiddler as proxy via device/emulator system settings
  - http://docs.telerik.com/fiddler/configure-fiddler/tasks/ConfigureForAndroid
  - Or use proxy settings in app for the connection

```java
// http://stackoverflow.com/questions/1432961/how-do-i-make-httpurlconnection-use-a-proxy
URL url = new URL(urlString);
Proxy proxy = new Proxy(Proxy.Type.HTTP, new InetSocketAddress("130.243.36.55", 8888));
URLConnection conn = url.openConnection(proxy);
```


Fiddler Web Debugger
File Edit Rules Tools View Help GET /book
Replay Go Stream Decode Keep: All sessions Any Process Find Save Browse Clear Cache TextWizard Tearoff MSDN Search...
Statistics Inspectors AutoResponder Composer Filters Log Timeline
Headers TextView WebForms HexView Auth Cookies Raw JSON XML
Get SyntaxView Transformer Headers TextView ImageView HexView WebView Auth Caching Cookies Raw JSON XML
ALT+Q > type HELP...

#	Result	Protocol	Host	URL
1	200	HTTP	localhost	/myhome/test/next.php

```
00000000  50 4F 53 54 20 68 74 74 70 3A 2F 2F 6C 6F 63 61 6C 68 6F 73 74 2F 6D 79  POST http://localhost/my
00000018  68 6F 6D 65 2F 74 65 73 74 2F 6E 65 78 74 2E 70 68 70 20 48 54 54 50 2F  home/test/next.php HTTP/
00000030  31 2E 31 0D 0A 41 63 63 65 70 74 3A 20 74 65 78 74 2F 68 74 6D 6C 2C 20  1.1..Accept: text/html, 
00000048  61 70 70 6C 69 63 61 74 69 6F 6E 2F 78 68 74 6D 6C 2B 78 6D 6C 2C 20 2A  application/xhtml+xml, *
00000060  2F 2A 0D 0A 52 65 66 65 72 65 72 3A 20 68 74 74 70 3A 2F 2F 6C 6F 63 61  /*..Referer: http://loca
00000078  6C 68 6F 73 74 2F 6D 79 68 6F 6D 65 2F 74 65 73 74 2F 66 6F 72 6D 2E 68  lhost/myhome/test/form.h
00000090  74 6D 6C 0D 0A 41 63 63 65 70 74 2D 4C 61 6E 67 75 61 67 65 3A 20 73 76  tml..Accept-Language: sv
000000A8  2D 53 45 0D 0A 55 73 65 72 2D 41 67 65 6E 74 3A 20 4D 6F 7A 69 6C 6C 61  -SE..User-Agent: Mozilla
000000C0  2F 35 2E 30 20 28 57 69 6E 64 6F 77 73 20 4E 54 20 36 2E 31 3B 20 57 4F  /5.0 (Windows NT 6.1; WO
000000D8  57 36 34 3B 20 54 72 69 64 65 6E 74 2F 37 2E 30 3B 20 72 76 3A 31 31 2E  W64; Trident/7.0; rv:11.
000000F0  30 29 20 6C 69 6B 65 20 47 65 63 6B 6F 0D 0A 41 63 63 65 70 74 2D 45 6E  0) like Gecko..Accept-En
00000108  63 6F 64 69 6E 67 3A 20 67 7A 69 70 2C 20 64 65 66 6C 61 74 65 0D 0A 48  coding: gzip, deflate..H
00000120  6F 73 74 3A 20 6C 6F 63 61 6C 68 6F 73 74 0D 0A 43 6F 6E 74 65 6E 74 2D  ost: localhost..Content-
00000138  4C 65 6E 67 74 68 3A 20 38 38 0D 0A 44 4E 54 3A 20 31 0D 0A 43 6F 6E 6E  Length: 88..DNT: 1..Conn
00000150  65 63 74 69 6F 6E 3A 20 4B 65 65 70 2D 41 6C 69 76 65 0D 0A 50 72 61 67  ection: Keep-Alive..Prag
00000168  6D 61 3A 20 6E 6F 2D 63 61 63 68 65 0D 0A 0D 0A 63 62 5F 31 3D 70 61 75  ma: no-cache....cb_1=pau
00000180  6C 61 74 74 75 63 6B 40 79 61 68 6F 6F 2E 63 6F 6D 26 63 62 5F 32 3D 6A  lattuck@yahoo.com&cb_2=j
00000198  61 6D 65 73 62 6F 6E 64 40 79 61 68 6F 6F 2E 63 6F 6D 26 63 62 5F 33 3D  amesbond@yahoo.com&cb_3=
000001B0  6C 61 73 74 6E 61 6D 65 40 79 61 68 6F 6F 2E 63 6F 6D 26 73 75 62 6D 69  lastname@yahoo.com&submi
000001C8  74 3D 73 75 62 6D 69 74                                                  t=submit
```

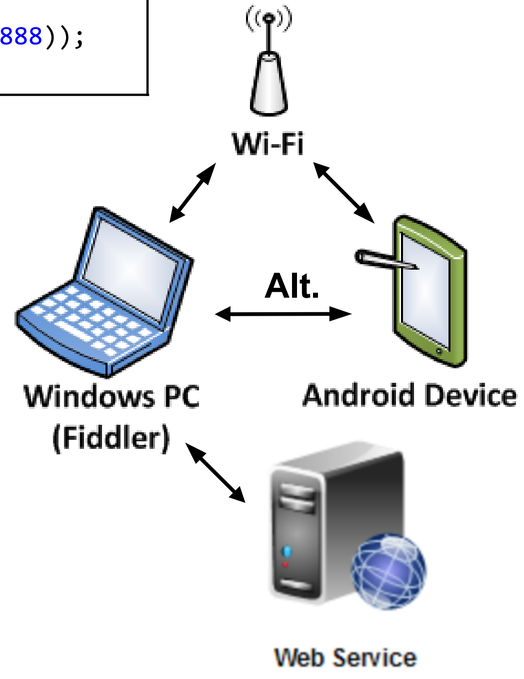
0 [0x0] Readonly

Response is encoded and may need to be decoded before inspection. Click here to transform.

```
00000000  48 54 54 50 2F 31 2E 31 20 32 30 30 20 4F 4B 0D 0A 44 61 74 65 3A 20 54  HTTP/1.1 200 OK..Date: T
00000018  68 75 2C 20 31 34 20 4E 6F 76 20 32 30 31 33 20 31 33 3A 30 34 3A 30 32  hu, 14 Nov 2013 13:04:02
00000030  20 47 4D 54 0D 0A 53 65 72 76 65 72 3A 20 41 70 61 63 68 65 2F 32 2E 34   GMT..Server: Apache/2.4
00000048  2E 34 20 28 57 69 6E 33 32 29 20 4F 70 65 6E 53 53 4C 2F 31 2E 30 2E 31  .4 (Win32) OpenSSL/1.0.1
00000060  65 20 50 48 50 2F 35 2E 35 2E 33 0D 0A 58 2D 50 6F 77 65 72 65 64 2D 42  e PHP/5.5.3..X-Powered-B
00000078  79 3A 20 50 48 50 2F 35 2E 35 2E 33 0D 0A 4B 65 65 70 2D 41 6C 69 76 65  y: PHP/5.5.3..Keep-Alive
00000090  3A 20 74 69 6D 65 6F 75 74 3D 35 2C 20 6D 61 78 3D 31 30 30 0D 0A 43 6F  : timeout=5, max=100..Co
000000A8  6E 6E 65 63 74 69 6F 6E 3A 20 4B 65 65 70 2D 41 6C 69 76 65 0D 0A 54 72  nnection: Keep-Alive..Tr
000000C0  61 6E 73 66 65 72 2D 45 6E 63 6F 64 69 6E 67 3A 20 63 68 75 6E 6B 65 64  ansfer-Encoding: chunked
000000D8  0D 0A 43 6F 6E 74 65 6E 74 2D 54 79 70 65 3A 20 74 65 78 74 2F 68 74 6D  ..Content-Type: text/htm
000000F0  6C 0D 0A 0D 0A 32 30 61 39 0D 0A 3C 70 3E 59 6F 75 20 63 61 6D 65 20 66  l....20a9..<p>You came f
00000108  72 6F 6D 20 68 74 74 70 3A 2F 2F 6C 6F 63 61 6C 68 6F 73 74 2F 6D 79 68  rom http://localhost/myh
00000120  6F 6D 65 2F 74 65 73 74 2F 66 6F 72 6D 2E 68 74 6D 6C 3C 2F 70 3E 3C 62  ome/test/form.html</p><b
00000138  72 20 2F 3E 0A 3C 62 3E 4E 6F 74 69 63 65 3C 2F 62 3E 3A 20 20 55 6E 64  r />.<b>Notice</b>:  Und
```

0 [0x0] Readonly

Capturing | All Processes | 1 / 1 | http://localhost/myhome/test/next.php



See the utils.MyProxy class used with OkHttp in the jsonokhttp.7z app!

Networking with Android Volley OkHttp

- Behavior changes in Android for API levels
  - https://developer.android.com/about/ > Behaviour Changes
- Since the Apache HttpClient removal from API-23 it is recommended to use the HttpURLConnection instead
  - https://developer.android.com/reference/java/net/HttpURLConnection.html
  - But it is not so convenient and easy to use in many use cases...
- If you are doing frequent networking with small traffic against a web service in your app it is more beneficial to use the Volley or OKHttp libraries instead
  - Volley may be a little hard to set up, but when it is done it is very easy to do networking
- If we don't need to handle requests in a queue, prioritize requests, or schedule requests, we could use OkHttp
  - OkHttp is a HTTP and HTTP/2 (former SPDY protocol) client, which processes, tokenizes, simplifies, and compresses HTTP requests

Networking with

- OkHttp is a robust and efficient HTTP Client
  - OkHttp perseveres when the network is troublesome: it will silently recover from common connection problems
  - https://github.com/codepath/android_guides/wiki/Using-OkHttp
  - https://github.com/square/okhttp and http://square.github.io/okhttp/
- Make **synchronous** lightweight network calls on a separate thread, background service or with AsyncTask
- Make **asynchronous** calls by creating a Call object, using the enqueue() method, and passing an anonymous Callback object that implements both onFailure() and onResponse(). You will need to use runOnUiThread() or some post() method to get the result back on the main thread
- Include this statement in your Android Studio build.gradle file

```
dependencies {
 compile 'com.squareup.okhttp3:okhttp:3.9.0'
}
```

```
// Syncronous GET example, download JSON string and return as JSONObject
public static JSONObject getDataFromWeb() {
    OkHttpClient client = new OkHttpClient();
    Request request = new Request.Builder()
            .url(URL_TO_THE_WEB_SERVICE)
            .get()
            .addHeader("cache-control", "no-cache")
            .build();
    try {
        // synchronous network call
        response = client.newCall(request).execute();
        return new JSONObject(response.body().string());
    }
    catch (@NonNull IOException | JSONException e) {
        Log.e(TAG, "" + e.getLocalizedMessage());
    }
    return null;
}
```

For example usage of OkHttp, Retrofit, Picasso, GSON and org.json.* see the jsonokhttp.7z app!

Asyncronous OkHttp GET with proxy

```
public static void getDataFromWeb(Context ctx) {
    final Context context = ctx;
    client = new OkHttpClient.Builder()
            .connectTimeout(60, TimeUnit.SECONDS)
            .writeTimeout(60, TimeUnit.SECONDS)
            .readTimeout(60, TimeUnit.SECONDS)
            .proxy(new Proxy(Proxy.Type.HTTP, new InetSocketAddress(proxyHost, proxyPort)))
            .proxyAuthenticator(proxyAuthenticator)
            .build();
    Request request = new Request.Builder()
            .url(URL_TO_THE_WEB_SERVICE)
            .get()
            .addHeader("cache-control", "no-cache")
            .build();
    client.newCall(request).enqueue(new Callback()
    {
        @Override
        public void onFailure(Call call, IOException e) {
            e.printStackTrace();
            Log.e(TAG, "onFailure: " + e.toString());
        }
        @Override
        public void onResponse(Call call, final Response response) throws IOException
        {
            if (!response.isSuccessful()) {
                Log.e(TAG, "Unexpected code: " + response);
                throw new IOException("Unexpected code " + response);
            }
            // Read data on the OkHttp worker thread
            final String responseData = response.body().string();
            Log.d(TAG, "responseData: " + responseData);
            try {
                ((MainActivity) context).parseJson(new JSONObject(responseData));
            }
            catch (@NonNull JSONException e) {
                Log.e(TAG, "" + e.getLocalizedMessage());
            }
            // Run view-related code back on the main thread
            ((MainActivity)context).runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    ((MainActivity) context).updateListView();
                }
            });
        }
    });
}
```

OkHttp 3.x POST

```java
public static final MediaType MEDIA_JSON = MediaType.parse("application/json; charset=utf-8");
/**
http://stackoverflow.com/questions/23456488/how-to-use-okhttp-to-make-a-post-request
http://stackoverflow.com/questions/34179922/okhttp-post-body-as-json
*/
public static String postData(String jsonPostDataString,
            String httpUrl)
{
    OkHttpClient client = new OkHttpClient();
    /*
    RequestBody formBody = new FormBody.Builder()
            .add("message", postData)
            .build();
    */
    RequestBody jsonBody = RequestBody.create(MEDIA_JSON,
            jsonPostDataString);
    Request request = new Request.Builder()
            .url(httpUrl)
            .post(jsonBody)
            .build();
    try {
        // Syncronous POST example - upload the JSON string
        Log.d(TAG, "client.newCall(request).execute()");
        Response response =
                client.newCall(request).execute();
        // Do something with the response...
        return response.toString();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return null;
}
```

```java
// Builds a test JSONObject for POST request (jsonPostDataString)
public static String jsonToString() {
    try {
        JSONObject jObject = new JSONObject();
        jObject.put("DeviceId", "352957061403327");
        jObject.put("AuthCode", "hjo@du.se");
        jObject.put("FrictionValue", 0.5);
        jObject.put("TimeStamp", "2015-12-24 08:14");
        jObject.put("FileName", "20151224081425-a0ce4bbb-FRC.zip");
        jObject.put("Bearing", 45.5);
        jObject.put("GpsAccuracy", 4.5);
        jObject.put("StartSpeed", 55.5);
        jObject.put("EndSpeed", 25.5);
        jObject.put("Altitude", 155.5);
        // send thumbnail image
        String encodedImage = getImageAsThumbnail(context);
        jObject.put("ThumbnailImage", encodedImage);
        // location start
        JSONObject jObjectLatStart = new JSONObject();
        jObjectLatStart.put("LatStart", 81.3444);
        JSONObject jObjectLngStart = new JSONObject();
        jObjectLngStart.put("LngStart", 82.2323);
        // location end
        JSONObject jObjectLatEnd = new JSONObject();
        jObjectLatEnd.put("LatEnd", 81.3454);
        JSONObject jObjectLngEnd = new JSONObject();
        jObjectLngEnd.put("LngEnd", 82.2333);
        JSONArray coordinates = new JSONArray();
        coordinates.put(jObjectLatStart);
        coordinates.put(jObjectLngStart);
        coordinates.put(jObjectLatEnd);
        coordinates.put(jObjectLngEnd);
        jObject.put("Coordinates", coordinates);
        Log.d(TAG, "toJson: " + jObject.toString());
        return jObject.toString();
    } catch (JSONException e) {
        System.err.println(e.toString() + " : " + e.getMessage());
        e.printStackTrace();
    }
    return null;
}
```

Networking with Android Volley

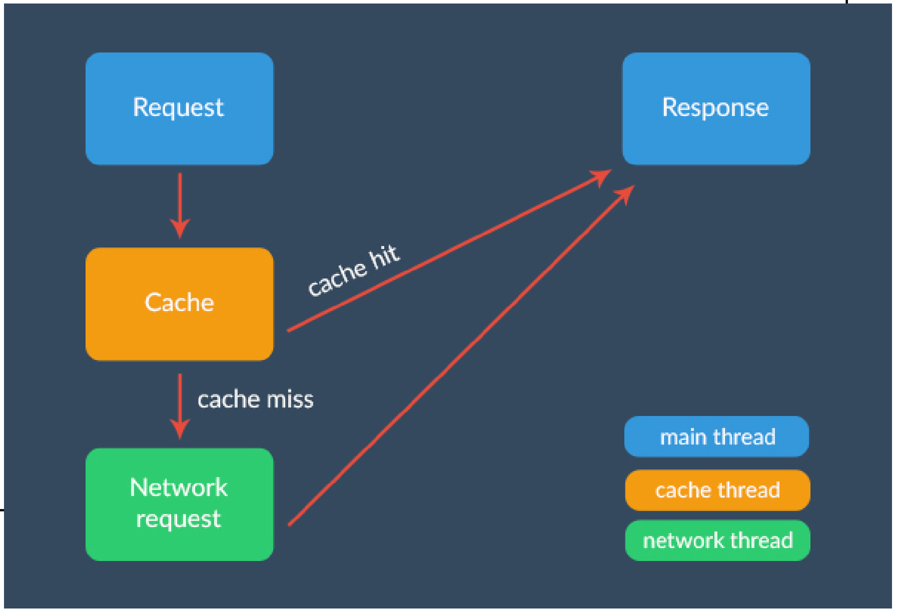
- According to the official definition and list of features from https://developer.android.com/training/volley/index.html, “Volley is an HTTP library that makes networking for Android apps easier and most importantly, faster“...
- Volley offers the following benefits
  - Automatic scheduling of network requests
  - Multiple concurrent network connections
  - A transparent disk and memory response caching with a standard HTTP cache coherence
  - Support for request prioritization
  - Cancellation of request API; this means that you can cancel a single request, or set blocks or scopes of requests to cancel
  - Ease of customization; for example, for retry and backoff
  - Strong ordering, which makes it easy to correctly populate your UI with data fetched asynchronously from the network
  - Debugging and tracing tools

Networking with Android Volley

- Example, but alot of other things must be set up as well!

```
final TextView mTextView = (TextView) findViewById(R.id.text);
...
// Instantiate the RequestQueue
RequestQueue queue = Volley.newRequestQueue(this);
String url ="http://www.google.com";
// Request a string response from the provided URL
StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                // Display the first 500 characters of the response string
                mTextView.setText("Response is: "
                     + response.substring(0, 500));
            }
        }, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        mTextView.setText("That didn't work!");
    }
});
// Add the request to the RequestQueue
queue.add(stringRequest);
```

Simplified life of a Volley request



Networking with Android Volley

- Include this statement in your Android Studio build.gradle file to use Volley

```
dependencies {
   compile 'com.android.volley:volley:1.0.0'
}
```

- Introduction tutorials to set up Volley correct
  - https://github.com/codepath/android_guides/wiki/Networking-with-the-Volley-Library
  - http://code.tutsplus.com/tutorials/an-introduction-to-volley--cms-23800
  - http://androidsrc.net/android-volley-library-tutorial/
- Example apps and source
  - An example application using Volley: http://code.tutsplus.com/tutorials/creating-a-weather-application-for-mars-using-volley--cms-23812
  - The source code is found here: https://github.com/tutsplus/Android-VolleyWeatherOnMars
  - **I have put some Volley projects in my example web folder as well: http://users.du.se/~hjo/cs/common/androidexamples/ ... volley*.7zip**

Networking with Retrofit v2+ 1

- A type-safe HTTP client which depends on OkHttp
  - https://square.github.io/retrofit/
- Turns your HTTP API into a Java interface and makes up/down-loading JSON and XML simple
- Uses ”converters” to serialize and deserialize data into Java Objects
  - GSON, Jackson, Moshi, Simple XML library, ...
- Supports synchronously and asynchronously calls
- Calls can easly be canceled plus many other functions

```
dependencies {
    // http://search.maven.org/#search%7Cga%7C1%7Ccom.google.code.gson
    compile 'com.google.code.gson:gson:2.7'
    // https://guides.codepath.com/android/Consuming-APIs-with-Retrofit
    compile 'com.squareup.retrofit2:retrofit:2.3.0'
    compile 'com.squareup.retrofit2:converter-gson:2.3.0'
    provided 'org.glassfish:javax.annotation:10.0-b28'
}
```

For example usage of OkHttp, Retrofit, Picasso, GSON and org.json.* see the jsonokhttp.7z app!

Networking with Retrofit v2+ 2

- Create your java model classes manually or automatically via jsonschema2pojo (Plain Old Java Objects): http://www.jsonschema2pojo.org/
- Create the Retrofit instance

```
// Trailing slash is needed
public static final String BASE_URL = "http://api.myservice.com/";
// To send out network requests to an API, we use the Retrofit builder class and specify the base URL for the service.
Retrofit retrofit = new Retrofit.Builder()
        .baseUrl(BASE_URL)
        .addConverterFactory(GsonConverterFactory.create())
        .build();
// construct a service leveraging the MyApiEndpointInterface interface with the defined endpoints
MyApiEndpointInterface apiService = retrofit.create(MyApiEndpointInterface.class);
```

- Define the endpoints (web service API/methods) in the interface

```
/* Annotation and Description
@Path  - variable substitution for the API endpoint (i.e. username will be swapped for {username} in the URL endpoint).
@Query - specifies the query key name with the value of the annotated parameter.
@Body  - payload for the POST call (serialized from a Java object to a JSON string)
@Header - specifies the header with the value of the annotated parameter */
public interface MyApiEndpointInterface {
    @Headers({"Cache-Control: max-age=640000", "User-Agent: My-App-Name"})
    // Request method and URL specified in the annotation
    @GET("users/{username}")
    Call<User> getUser(@Path("username") String username);
    @GET("group/{id}/users")
    Call<List<User>> groupList(@Path("id") int groupId, @Query("sort") String sort);
    @POST("users/new")
    Call<User> createUser(@Body User user);
}
```

Networking with Retrofit v2+ 3

- Accessing the API asynchronously and synchronously request
  - Retrofit will download and parse the API data on a background thread, and then deliver the results back to the UI thread via the onResponse or onFailure method. **User user is the parsed ready to use object!**

```
String username = "hans"; // API call will be: http://api.myservice.com/users/hans
Call<User> call = apiService.getUser(username);
call.enqueue(new Callback<User>() {
    @Override
    public void onResponse(Call<User> call, Response<User> response) {
        int statusCode = response.code();
        User user = response.body();
    }
    @Override
    public void onFailure(Call<User> call, Throwable t) {
        // Log error here since request failed http://api.myservice.com/users/hans
    }
});
```

```
String username = "hans";  // A syncronous call
Call<User> call = apiService.getUser(username);
User user = response.body();
```

- Introduction tutorials to set up and use Retrofit v2+
  - https://guides.codepath.com/android/Consuming-APIs-with-Retrofit
  - https://inthecheesefactory.com/blog/retrofit-2.0/en

Picasso

- Image downloading library with automatic RAM memory and disk caching
- One line of code is ofthen enough to load a remote image into a ImageView
  - Supports resizing, cropping, placeholders, autofit, rotate, …, etc.

```
Picasso.with(context).load(item.getProfilePic() /* Uri, File, Res, String, ... */)
     .placeholder(R.mipmap.ic_launcher)
     .error(R.mipmap.ic_launcher)
     .into(imageView);
```

- The global default Picasso instance
  - This instance is automatically initialized with defaults that are suitable to most implementations
  - Three download threads for disk and network access
- Use a custom Picasso instance for non-standard cases

```
// create a Picasso.Builder object
Picasso.Builder picassoBuilder = new Picasso.Builder(context);
// make your adjustments to Picasso here
// PicassoBuilder creates the Picasso object to do the actual requests
Picasso picasso = picassoBuilder.build();
picasso.load("file:///android_asset/myPic.png").into(imageView);
```

Be lazy and productive!

- During the Android application development there is many useful tools which could be found handy
- Android layout finder which eleminate all the findViewById() boilerplate code
  - https://www.buzzingandroid.com/tools/android-layout-finder/
- JSON validator and viewer
  - http://jsonlint.com/ and http://jsonviewer.stack.hu/
- JSON to POJO Creator
  - The JsonSchema2Pojo tool converts JSON to Java classes
  - http://www.jsonschema2pojo.org/
- Android DPI calculator, Gradle Please dependency finder, Postman – Rest client, Android Asset Studio, …
- And many more essential tools listed here (16+ tools)
  - http://www.technotalkative.com/lazy-android-part-7-useful-tools/

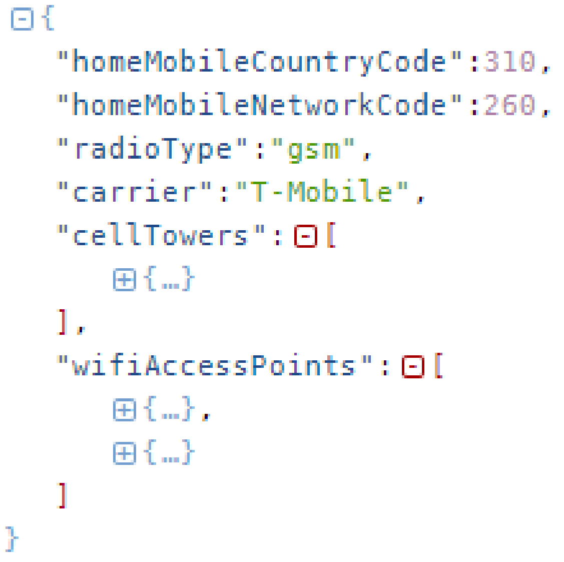
JSON format

- JSON is built on two structures
  - A collection of name/value pairs as in an object
    - If using an automatic de/serializer keep the exact names in the class
  - An ordered list of values as in an array
- Objects
  - Begins with and ends with (**{ }**)
- Object members
  - Consist of strings and values separated by colon (**:**)
  - Members are separated by commas
- Arrays
  - Begin and end with brackets (**[ ]**) and contain values
  - Values are separated by commas

```
{
 "homeMobileCountryCode": 310,
 "homeMobileNetworkCode": 260,
 "radioType": "gsm",
 "carrier": "T-Mobile",
 "cellTowers": [
  {
   "cellId": 39627456,
   "locationAreaCode": 40495,
   "mobileCountryCode": 310,
   "mobileNetworkCode": 260,
   "age": 0,
   "signalStrength": -95
  }
 ],
 "wifiAccessPoints": [
  {
   "macAddress": "01:23:45:67:89:AB",
   "signalStrength": 8,
   "age": 0,
   "signalToNoiseRatio": -65,
   "channel": 8
  },
  {
   "macAddress": "01:23:45:67:89:AC",
   "signalStrength": 4,
   "age": 0
  }
 ]
}
```

JSON processing

- Values
  - Can be a string, a number, an object, byte data as images etc. needs to be Base64 encoded/decoded, an array, true/false or null
- Strings
  - Surrounded by double quotes
  - Contain Unicode characters or common backslash escapes
- JSON data is represented by JSONObject
  - A property map – name value pairs
- JSON array data by Arrays
  - An array of JSONObject
- JSON data can be parsed by JSONTokener
- JSON data can be created by calling the toString() method on JSONObject



```
⊟{
    "homeMobileCountryCode":310,
    "homeMobileNetworkCode":260,
    "radioType":"gsm",
    "carrier":"T-Mobile",
    "cellTowers":⊟[
        ⊞{…}
    ],
    "wifiAccessPoints":⊟[
        ⊞{…},
        ⊞{…}
    ]
}
```

Create a JSON message

- The JSON API is guaranteed to create syntactically correct JSON strings – check against a JSON validator
- To convert data (app variables) to JSON
  - Create a JSONObject
  - Put requirerd data into the internal map
  - Call toString() which performs serialization

```java
public JSONObject expenseToJson(Expense exp)
{
      JSONObject obj = new JSONObject();
      try{
            // populate the map
            // a JSONObject, JSONArray, String, Boolean, Integer, Long, Double, NULL, or null.
            obj.put("description", exp.description);
            obj.put("amount", exp.amount);
            obj.put("expenseDate", exp.expenseDate);
      }
      catch(JSONException e){
            Log.d(TAG, "JSONException", e);
      }
      return obj;
}
Expense expensObj = new Expense();
// convert expense object to JSONObject then call toString()
String json = expenseToJson(expensObj).toString();
```

```java
class Expense{
      String description;
      float amount;
      String expenseDate;
}
```

Create JSON from array data

- To convert array data to JSON
  - Convert each object in the array you got to JSONObject
  - Put the object into a JSON array
  - Serialize to JSON by calling toString on the JSONArray
  - Check result against JSON valdator

```
List<Expense> expenses = new ArrayList<Expense>(3);
expenses.add(expensObj);

JSONArray jArray = new JSONArray();

for(Expense exp : expenses){
     // Appends value to the end of this array. In this case a JSONObject
// a JSONObject, JSONArray, String, Boolean, Integer, Long, Double, NULL, or null.
     jArray.put(expenseToJson(exp));
}

// Serialize to JSON
String jsonArray = jArray.toString();
```

```
class Expense{
      String description;
      float amount;
      String expenseDate;
}
```

Create JSON from byte data

- How to convert an image to string data for JSON

```java
/**
 * Decode the result at: https://www.base64decode.org/
 * https://stackoverflow.com/questions/4830711/how-to-convert-a-image-into-base64-string
 * https://stackoverflow.com/questions/2577221/android-how-to-create-runtime-thumbnail
 * https://stackoverflow.com/questions/30598357/who-is-adding-n-in-base64-encoded-image-when-i-write-it-in-a-file-java
 */
private static String getImageAsThumbnailTest(Context context)
{
    final int THUMB_SIZE = 48;
    final int THUMB_QUALITY = 80;
    // String fileName = Environment.getExternalStorageDirectory() + "/understanding.jpg";
    // Bitmap ThumbImage = ThumbnailUtils.extractThumbnail(BitmapFactory.decodeFile(fileName),
      // THUMB_SIZE, THUMB_SIZE /*, Bitmap.Config.ARGB_8888*/);
    Bitmap ThumbImage = ThumbnailUtils.extractThumbnail(BitmapFactory.decodeResource(context.getResources(),
      R.raw.understanding), THUMB_SIZE, THUMB_SIZE /*, Bitmap.Config.ARGB_8888 */);
    if (ThumbImage != null) {
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        ThumbImage.compress(Bitmap.CompressFormat.JPEG, THUMB_QUALITY, baos);
        byte[] bytes = baos.toByteArray();
        String imageStr = Base64.encodeToString(bytes, Base64.NO_WRAP);
        Log.d(TAG, "imageStr.length(): " + imageStr.length());
        Log.d(TAG, "THUMB_SIZE=48px: " + imageStr);
        return imageStr;
    }
    else {
        Log.d(TAG, "ThumbImage is NULL");
        return "";
    }
}
```

Send message, get response and process JSON 1

- To send data use the POST method
  - Specify the MIME type of the content
    - "application/json; charset=utf-8" for JSON data
  - Add the JSON data to the request as an entity
- Process JSON
  - Construct a JSONTokener around the response data
  - Get the JSONObjects or JSONArrays by calling nextValue()
    - Returns null if no more data
  - Iterate through JSONArrays if necessary
  - Extract data from JSONObject
    - Call the XXX getXXX(String name) methods
    - XXX is JSONObject, JSONArray, String, Boolean, Long, Double, Int, etc.
  - Parsing code normally makes assumptions about data structure
    - Catch and log exceptions in case data is wrong

Send message, get response and process JSON 2 (Apache HTTP client)

```
public void postDataJSON(String uri, String jsonArray)
{
        // Create a new HttpClient and Post Header
        HttpClient client = new DefaultHttpClient();
        HttpPost post = new HttpPost(uri);
        Log.d(TAG, uri.toString());
        try {
                // Add your data
                // set MIME type
                post.addHeader("Content-Type", "application/json; charset=utf-8");
                // add the JSON data as an entity
                post.setEntity(new StringEntity(jsonArray));
                // Handler that encapsulates the process of generating a response object from a HttpResponse.
                ResponseHandler<String> responseHandler = new BasicResponseHandler();
                // Execute HTTP Post Request
                String jsonResponse = client.execute(post, responseHandler);
                Log.d(TAG, jsonResponse.toString());
                // parse the response
                parseJSON(jsonResponse);
        }
        catch (ClientProtocolException e) {
                Log.d(TAG, "ClientProtocolException", e);
        }
        catch (IOException e) {
                Log.d(TAG, "IOException", e);
        }
        catch (Exception e) {
                Log.d(TAG, "Exception", e);
        }
        finally{
                client.getConnectionManager().shutdown();
        }
}
```

If you do not use any modern network library you must use either java.net.HttpURLConnection or javax.net.ssl.HttpsURLConnection

Send message, get response and process JSON 3

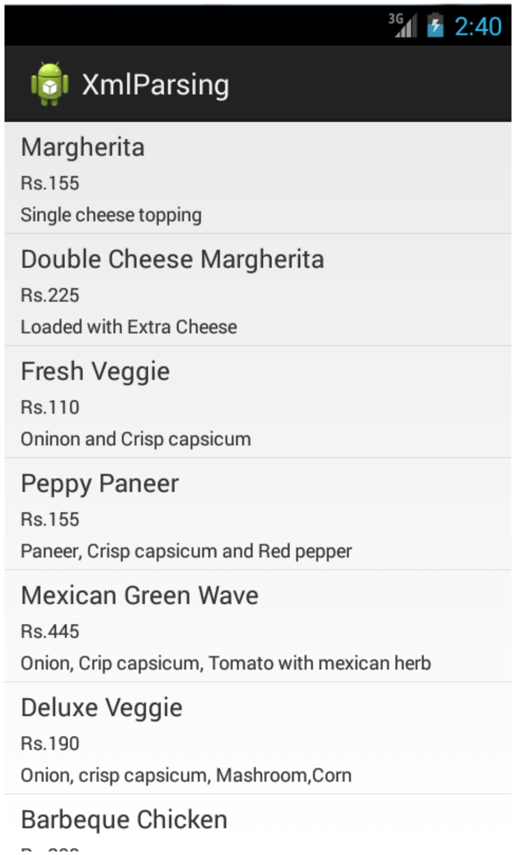
```java
public void parseJSON(String jsonString)
{
	// response string could look like this
	jsonString = "{\"expense\":[" +
		"{\"description\":\"Severn bridge toll\",\"amount\":\"15.0\",\"expenseDate\":\"1305876297746\"}," +
		"{\"description\":\"Slap up lunch\",\"amount\":\"2000.0\",\"expenseDate\":\"1396250633843\"}]}";

	JSONTokener tokenizer = new JSONTokener(jsonString);

	try{
		// Returns the next value from the input.
		JSONObject wrapper = (JSONObject) tokenizer.nextValue();
		// Returns the value mapped by name if it exists and is a JSONArray.
		JSONArray expenses = wrapper.getJSONArray("expense");
		for(int i=0; i < expenses.length(); i++)
		{
		    // Returns the value mapped by name if it exists, coercing it if necessary.
		    String value =
		        expenses.getJSONObject(i).getString("description");
		    Log.d(TAG, "description: " + value);
		}
	 }
	catch(JSONException e){
		Log.d(TAG, "JSONException", e);
	}
}
```

```
{"expense":[{"description":"Severn bridge
toll","amount":"15.0","expenseDate":"1305876297746"},
{"description":"Slap up
lunch","amount":"2000.0","expenseDate":"1396250633843"}]}
```

```
⊟{
   "expense":⊟[
      ⊟{
         "description":"Severn bridge toll",
         "amount":"15.0",
         "expenseDate":"1305876297746"
      },
      ⊟{
         "description":"Slap up lunch",
         "amount":"2000.0",
         "expenseDate":"1396250633843"
      }
   ]
}
```

JsonParsing and XmlParsing example

- Full sample on JSON and XML parsing and network
  - Source in examples
    - http://www.androidhive.info/2011/11/android-xml-parsing-tutorial/
    - http://www.androidhive.info/2012/01/android-json-parsing-tutorial/



2:40

XmlParsing

Margherita
Rs.155
Single cheese topping

Double Cheese Margherita
Rs.225
Loaded with Extra Cheese

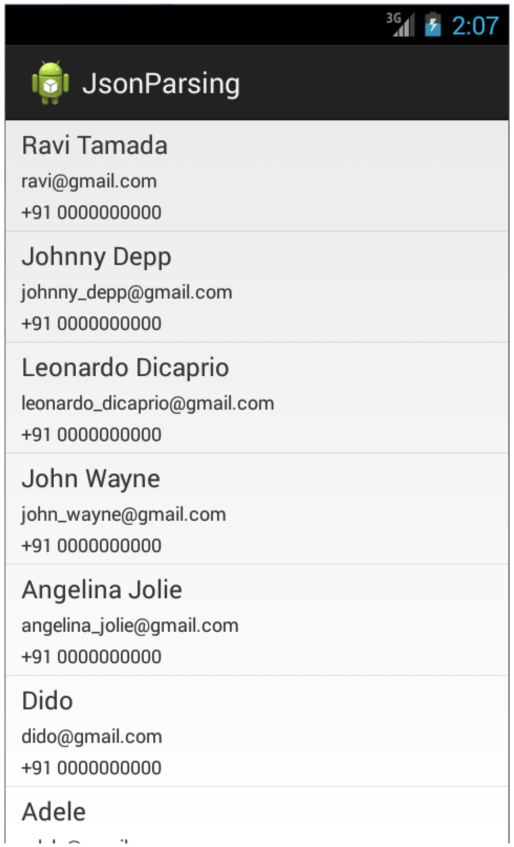
Fresh Veggie
Rs.110
Oninon and Crisp capsicum

Peppy Paneer
Rs.155
Paneer, Crisp capsicum and Red pepper

Mexican Green Wave
Rs.445
Onion, Crip capsicum, Tomato with mexican herb

Deluxe Veggie
Rs.190
Onion, crisp capsicum, Mashroom,Corn

Barbeque Chicken



2:07

JsonParsing

Ravi Tamada
ravi@gmail.com
+91 0000000000

Johnny Depp
johnny_depp@gmail.com
+91 0000000000

Leonardo Dicaprio
leonardo_dicaprio@gmail.com
+91 0000000000

John Wayne
john_wayne@gmail.com
+91 0000000000

Angelina Jolie
angelina_jolie@gmail.com
+91 0000000000

Dido
dido@gmail.com
+91 0000000000

Adele

XML vs. JSON 1

- Anyone who has done programming in recent years has encountered XML (eXtensible Markup Language) - which is self-describing
- XML must follow certain rules and guidelines
  - A XML schema is a document that describes the structure of a particular XML file
  - CDATA tags within an XML document can contain any kind of data (binary data)
- Example packaging data into an XML structure and placing it within the response element
  - An API call might request a customer record with a last name equaling Mott
- XML is usually the default format for ”heavier” web services

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<request>
      <query>
            <lastname>Mott</lastname>
            <maxhits>100</maxhits>
      </query>
</request>

---------------------------------------------------------

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response>
      <returncode>200</returncode>
      <query>
            <lastname>Mott</lastname>
            <hits>1</hits>
      </query>
      <data>
            <![CDATA[
            <contact>
                  <firstname>Troy</firstname>
                  <lastname>Mott</lastname>
                  <age>not telling</age>
            </contact>
            ]]>
      </data>
</response>
```

XML vs. JSON 2

- Many Internet API providers offers JSON (JavaScript Object Notation) as a data-format option for light web services
  - Usually used in AJAX (Asynchronous JavaScript and XML) web-programming technology which enables web pages to update dynamically by refreshing data in selected pockets on the page
  - Because less data is transferred – and much less data is parsed – an AJAX-enabled application can provide a much better user experience than a traditional web application - no web page reloads
- A sample response from an application responding to a query as
  - http://yourserver/app/searchcontact?Mott
  - The response is a string representation of a JavaScript object - a JSON string – translate methods: JSON.parse(string), JSON.stringify(object)
    - [{"firstname":"Troy","lastname":"Mott","age":"don't ask!"}]
- In summary, JSON is
  - A data-interchange format to encode JavaScript objects as strings
  - Limited to text/numeric values - binary values are not permitted!
  - More economical than XML in terms of data size, at the expense of readability – data is accessible thru object property dot notation

XML and JSON Twitter RSS feeds

```
<?xml version="1.0" encoding="UTF-8"?>
<statuses type="array">
          <status>
                    <created_at>Thu Apr 29 05:25:29 +0000 2010</created_at>
                    <id>13052369631</id>
                    <text>Wrapping up new article on JSON for Android
                              programmers...</text>
                    <source>
                              <a href="http://www.linkedin.com/" rel="nofollow">
                                        LinkedIn</a>
                    </source>
                    <truncated>false</truncated>
                    <in_reply_to_status_id />
                    <in_reply_to_user_id />
                    <favorited>false</favorited>
                    <in_reply_to_screen_name />
                    <user>
                              <id>15221439</id>
                              <name>fableson</name>
                              <screen_name>fableson</screen_name>
                              <location>Byram Township, NJ</location>
                              <description />
                              <profile_image_url>http://a3.twimg.com/profile_images/260492935
                                        /bookcover_normal.jpg</profile_image_url>
                              <url>http://msiservices.com</url>
                              <protected>false</protected>
                              <followers_count>52</followers_count>
                              <profile_background_color>
                                        9ae4e8
                                        <profile_text_color>000000</profile_text_color>
                                        <profile_link_color>0000ff</profile_link_color>
                                        <profile_sidebar_fill_color>e0ff92
</profile_sidebar_fill_color>
                                        <profile_sidebar_border_color>87bc44
</profile_sidebar_border_color>
                                        <friends_count>10</friends_count>
                                        <created_at>Tue Jun 24 17:04:11 +0000 2008</created_at>
                                        <favourites_count>0</favourites_count>
                                        <utc_offset>-18000</utc_offset>
                                        <time_zone>Eastern Time (US & Canada)</time_zone>
                                        <profile_background_image_url>http://s.twimg.com/a/1272044617/
                                                  images/themes/theme1/bg.png</profile_background_image_url>
                                        <profile_background_tile>false</profile_background_tile>
                                        <notifications>false</notifications>
                                        <geo_enabled>false</geo_enabled>
                                        <verified>false</verified>
                                        <following>false</following>
                                        <statuses_count>91</statuses_count>
                                        <lang>en</lang>
                                        <contributors_enabled>false</contributors_enabled>
                    </user>
                    <geo />
                    <coordinates />
                    <place />
                    <contributors />
          </status>
</statuses>
```

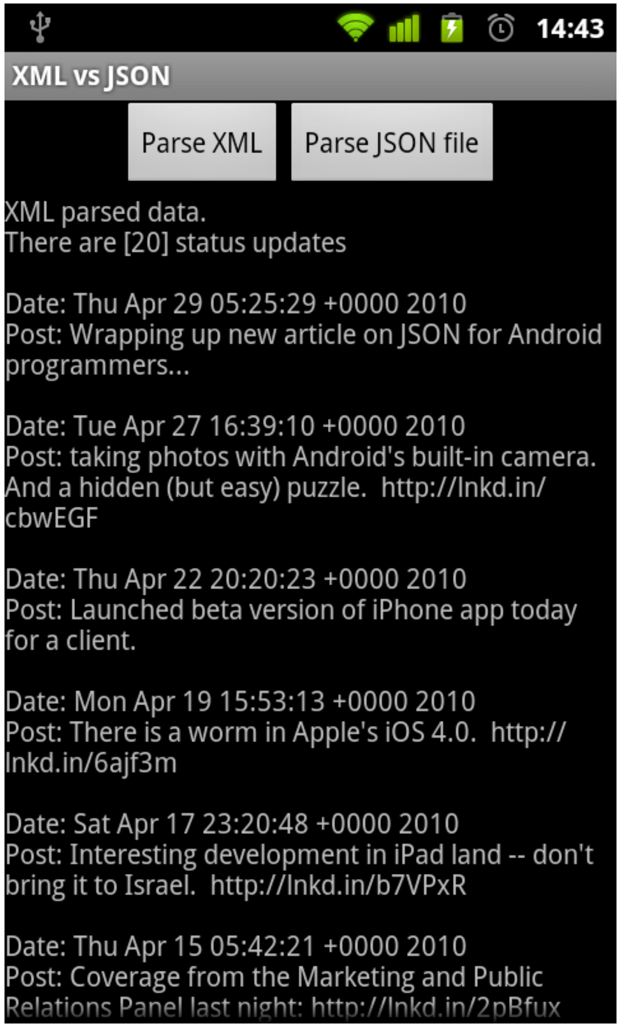
```
[
{"in_reply_to_status_id":null,
"favorited":false,
"created_at":"Thu Apr 29 05:25:29 +0000 2010",
"in_reply_to_screen_name":null,
"geo":null,
"source":"<a href=\"http://www.linkedin.com/\" rel=\"nofollow\
">LinkedIn</a>",
"contributors":null,
"place":null,
"truncated":false,
"coordinates":null,
"user":
{
"friends_count":10,
"description":"",
"lang":"en",
"statuses_count":91,
"time_zone":"Eastern Time (US & Canada)",
"profile_link_color":"0000ff",
"favourites_count":0,
"created_at":"Tue Jun 24 17:04:11 +0000 2008",
"contributors_enabled":false,
"profile_sidebar_fill_color":"e0ff92",
"following":null,
"geo_enabled":false,
"profile_background_image_url":"http://s.twimg.com/a/1272044617/i
/theme1/bg.png",
"profile_image_url":"http://a3.twimg.com/profile_images/260492935
/bookcover_normal.jpg",
"notifications":null,
"profile_sidebar_border_color":"87bc44",
"url":"http://msiservices.com",
"verified":false,
"profile_background_tile":false,
"screen_name":"fableson",
"protected":false,
"location":"Byram Township, NJ",
"profile_background_color":"9ae4e8",
"name":"fableson",
"followers_count":52,
"id":15221439,
"utc_offset":-18000,
"profile_text_color":"000000"
},
"in_reply_to_user_id":null,
"id":13052369631,
"text":"Wrapping up new article on JSON for Android programmers..."}
]
```

XML SAX parser 1

- http://www.ibm.com/developerworks/xml/library/x-andbene1/

See the XML vs JSON example

```
void examineXMLFile()
{
	try {
		// get the XML file from internal resource
		InputSource is = new InputSource(getResources()
			.openRawResource(R.raw.xmltwitter));
		// create the factory
		SAXParserFactory factory =
			SAXParserFactory.newInstance();
		// create a parser
		SAXParser parser = factory.newSAXParser();
		// create the reader (scanner)
		XMLReader xmlreader = parser.getXMLReader();
		// instantiate our handler
		TwitterFeedHandler tfh = new TwitterFeedHandler();
		// assign our handler
		xmlreader.setContentHandler(tfh);
		// perform the synchronous parse
		xmlreader.parse(is);
		// should be done... let's display our results
		tvData.setText(tfh.getResults());
	}
	catch (Exception e) {
		tvData.setText(e.getMessage());
	}
}
```



XML parsed data.
There are [20] status updates

Date: Thu Apr 29 05:25:29 +0000 2010
Post: Wrapping up new article on JSON for Android programmers...

Date: Tue Apr 27 16:39:10 +0000 2010
Post: taking photos with Android's built-in camera. And a hidden (but easy) puzzle. http://lnkd.in/cbwEGF

Date: Thu Apr 22 20:20:23 +0000 2010
Post: Launched beta version of iPhone app today for a client.

Date: Mon Apr 19 15:53:13 +0000 2010
Post: There is a worm in Apple's iOS 4.0. http://lnkd.in/6ajf3m

Date: Sat Apr 17 23:20:48 +0000 2010
Post: Interesting development in iPad land -- don't bring it to Israel. http://lnkd.in/b7VPxR

Date: Thu Apr 15 05:42:21 +0000 2010
Post: Coverage from the Marketing and Public Relations Panel last night: http://lnkd.in/2pBfux

XML SAX parser 2

TwitterFeedHandler

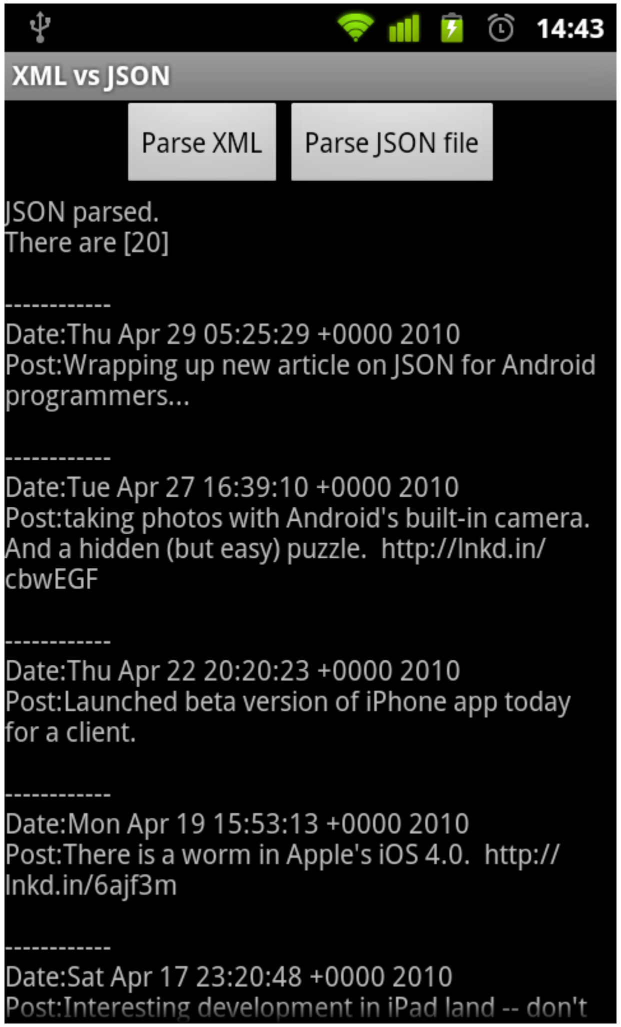
```java
public class TwitterFeedHandler extends DefaultHandler {
     StringBuilder sb = new StringBuilder("");
     String ret = "";
     boolean bStore = false;
     int howMany = 0;

     TwitterFeedHandler() {
     }

     String getResults(){
         return "XML parsed data.\nThere are [" + howMany +
                  "] status updates\n\n" + ret;
     }

     @Override
     public void startDocument() throws SAXException {
         // Receive notification of the beginning of the document
     }

     @Override
     public void endDocument() throws SAXException {
         // Receive notification of the end of the document
     }

     // Receive notification of the start of an element
     @Override
     public void startElement(String namespaceURI, String localName,
         String qName, Attributes atts) throws SAXException {
          try {
                 if (localName.equals("status")) {
                          sb.delete(0, sb.length());
                          bStore = true;
                 }
                 // turn off handling of the user element
                 if (localName.equals("user")) {
                          bStore = false;
                  }

                 if (localName.equals("text")) {
                          sb.delete(0, sb.length());
                 }

                if (localName.equals("created_at")) {
                          sb.delete(0, sb.length());
                 }
```

```java
             } catch (Exception e) {
                Log.d("error in startElement", e.getStackTrace().toString());
             }
        }

        // Receive notification of the end of an element.
        @Override
        public void endElement(String namespaceURI, String localName,
                     String qName) throws SAXException {
           if (bStore) {
                     // store Date
                     if (localName.equals("created_at")) {
                             ret += "Date: " + sb.toString() + "\n";
                             sb.delete(0, sb.length());
                         return;
                     }
                     // turn on handling after the user element
                     if (localName.equals("user")) {
                             bStore = true;
                     }
                     // store the Post (text)
                     if (localName.equals("text")) {
                             ret += "Post: " + sb.toString() + "\n\n";
                             sb.delete(0, sb.length());
                             return;
                     }
           }
           // count the number of status elements
             if (localName.equals("status")) {
                    howMany++;
                    bStore = false;
             }
        }

        // Receive notification of character data inside an element
        @Override
        public void characters(char ch[], int start, int length) {
           if (bStore) {
               String theString = new String(ch, start, length);
               this.sb.append(theString);
               Log.d("xmlvsjson", "characters() " + theString);
           }
        }
}
```

JSON Array/Object parser

- http://www.ibm.com/developerworks/xml/library/x-andbene1/

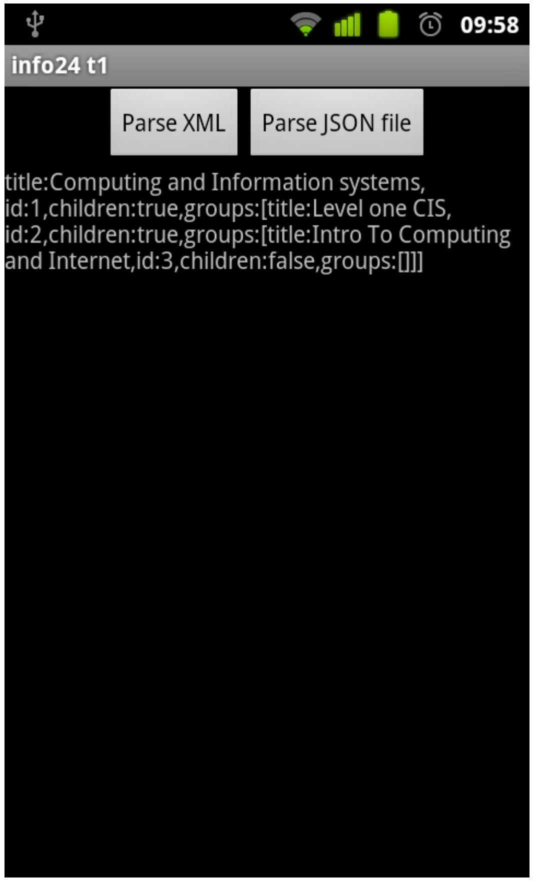
```java
void examineJSONFile()
{
      try{
            String x = "";
            // get the JSON file from internal resource
            InputStream is = this.getResources()
                  .openRawResource(R.raw.jsontwitter);
            byte [] buffer = new byte[is.available()];
            while (is.read(buffer) != -1)
                  ;
            String jsontext = new String(buffer);
            JSONArray entries = new JSONArray(jsontext);

            x = "JSON parsed.\nThere are [" + entries.length() + "]\n\n";

            for (int i=0; i<entries.length(); i++)
            {
                  JSONObject post = entries.getJSONObject(i);
                  x += "------------\n";
                  x += "Date:" + post.getString("created_at") + "\n";
                  x += "Post:" + post.getString("text") + "\n\n";
            }
            tvData.setText(x);
      }
      catch (Exception e){
            tvData.setText("Error w/file: " + e.getMessage());
      }
}
```



```
JSON parsed.
There are [20]

------------
Date:Thu Apr 29 05:25:29 +0000 2010
Post:Wrapping up new article on JSON for Android
programmers...

------------
Date:Tue Apr 27 16:39:10 +0000 2010
Post:taking photos with Android's built-in camera.
And a hidden (but easy) puzzle.  http://lnkd.in/
cbwEGF

------------
Date:Thu Apr 22 20:20:23 +0000 2010
Post:Launched beta version of iPhone app today
for a client.

------------
Date:Mon Apr 19 15:53:13 +0000 2010
Post:There is a worm in Apple's iOS 4.0.  http://
lnkd.in/6ajf3m

------------
Date:Sat Apr 17 23:20:48 +0000 2010
Post:Interesting development in iPad land -- don't
```

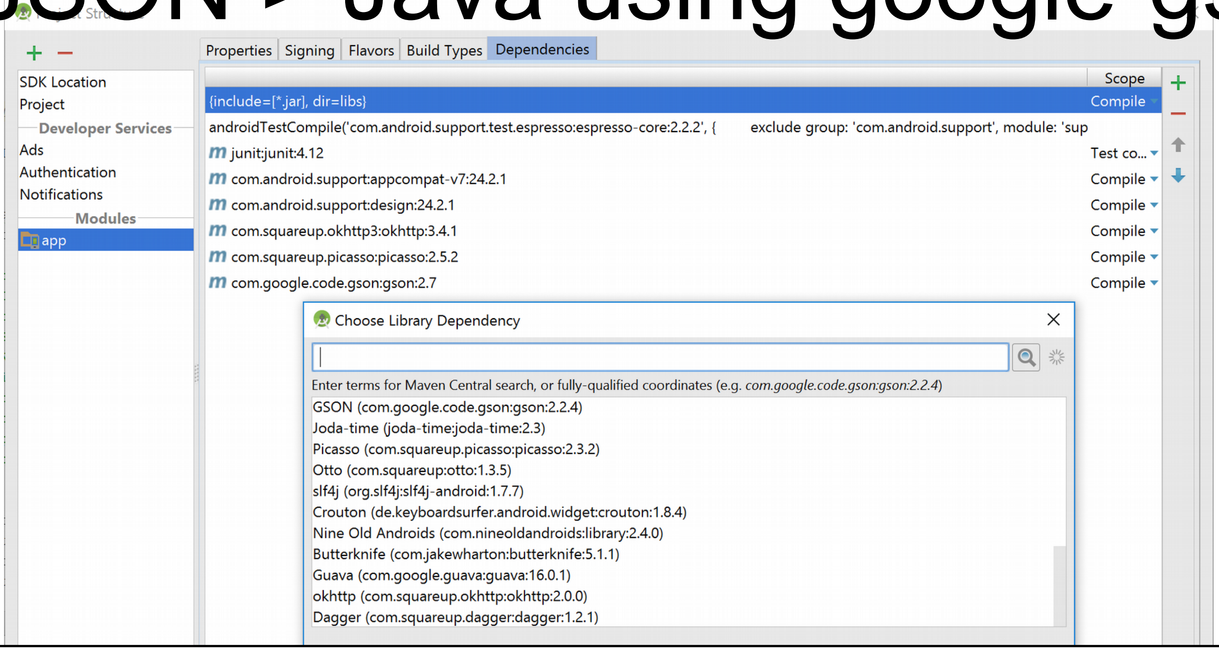
JSON > Java using google-gson 1

```java
// http://stackoverflow.com/questions/1688099/converting-json-to-java/1688182
public void gsontest(){
      try {
            InputStream is = getResources().openRawResource(R.raw.gsonjson);
            byte [] buffer = new byte[is.available()];
            while (is.read(buffer) != -1)
                  ;
            String json = new String(buffer);
/*              String json =
                    "{"
                        + "'title': 'Computing and Information systems',"
                        + "'id' : 1,"
                        + "'children' : 'true',"
                        + "'groups' : [{"
                            + "'title' : 'Level one CIS',"
                            + "'id' : 2,"
                            + "'children' : 'true',"
                            + "'groups' : [{"
                                + "'title' : 'Intro To Computing and Internet',"
                                + "'id' : 3,"
                                + "'children': 'false',"
                                + "'groups':[]"
                            + "}]"
                        + "}]"
                    + "}";
*/
          Log.d(TAG + " gsontest", json);
          // Now do the magic - http://code.google.com/p/google-gson/
          GsonData gdata = new Gson().fromJson(json, GsonData.class);
          tvData.setText(gdata.toString());
          // translate back to json
          String jsonStr = new Gson().toJson(gdata);
      }
      catch (Exception e) {
            tvData.setText(e.getMessage() + e.getStackTrace().toString());
      }
}
```



```
title:Computing and Information systems,
id:1,children:true,groups:[title:Level one CIS,
id:2,children:true,groups:[title:Intro To Computing
and Internet,id:3,children:false,groups:[]]]
```

JSON > Java using google-gson 2



Properties | Signing | Flavors | Build Types | Dependencies

SDK Location
Project
Developer Services
Ads
Authentication
Notifications
Modules
app

	Scope
{include=[*.jar], dir=libs}	Compile
androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', { exclude group: 'com.android.support', module: 'sup	
junit:junit:4.12	Test co...
com.android.support:appcompat-v7:24.2.1	Compile
com.android.support:design:24.2.1	Compile
com.squareup.okhttp3:okhttp:3.4.1	Compile
com.squareup.picasso:picasso:2.5.2	Compile
com.google.code.gson:gson:2.7	Compile

Choose Library Dependency

Enter terms for Maven Central search, or fully-qualified coordinates (e.g. *com.google.code.gson:gson:2.2.4*)

GSON (com.google.code.gson:gson:2.2.4)
Joda-time (joda-time:joda-time:2.3)
Picasso (com.squareup.picasso:picasso:2.3.2)
Otto (com.squareup:otto:1.3.5)
slf4j (org.slf4j:slf4j-android:1.7.7)
Crouton (de.keyboardsurfer.android.widget:crouton:1.8.4)
Nine Old Androids (com.nineoldandroids:library:2.4.0)
Butterknife (com.jakewharton:butterknife:5.1.1)
Guava (com.google.guava:guava:16.0.1)
okhttp (com.squareup.okhttp:okhttp:2.0.0)
Dagger (com.squareup.dagger:dagger:1.2.1)

Benchmarks

https://github.com/eishay/jvm-serializers/wiki/

```
public class GsonData {
      private String title;
      private Long id;
      private Boolean children;
      private List<GsonData> groups;

      public String getTitle() { return title; }
      public Long getId() { return id; }
      public Boolean getChildren() { return children; }
      public List<GsonData> getGroups() { return groups; }

      public void setTitle(String title) { this.title = title; }
      public void setId(Long id) { this.id = id; }
      public void setChildren(Boolean children) { this.children = children; }
      public void setGroups(List<GsonData> groups) { this.groups = groups; }

      public String toString() {
            return String.format("title:%s,id:%d,children:%s,groups:%s", title, id, children, groups);
      }
}
```

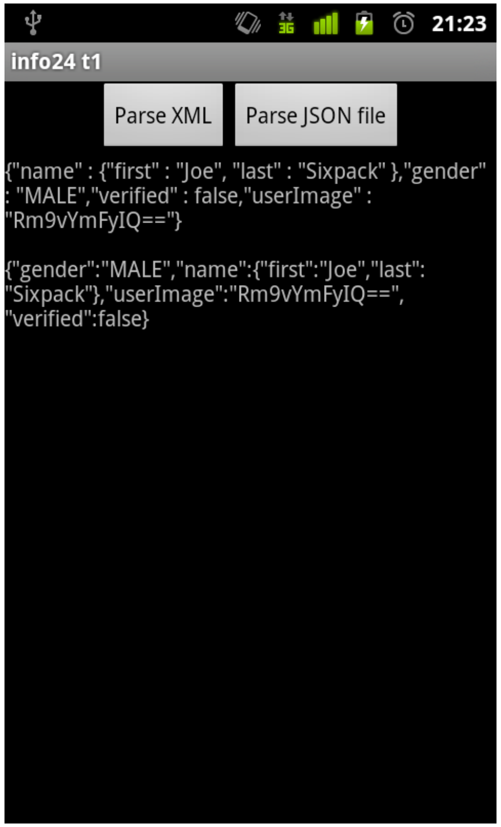
Note!

Variable names must match exactly between JSON and the Java class

JSON org.codehaus.jackson

- Jackson offers 3 alternative methods for processing JSON
  - Streaming API (aka "Incremental parsing/generation") reads and writes JSON content as discrete events
  - Tree Model provides a mutable in-memory tree representation of a JSON document
  - Data Binding converts JSON to and from POJOs (Plain Old Java Objects) based either on property accessor conventions or annotations

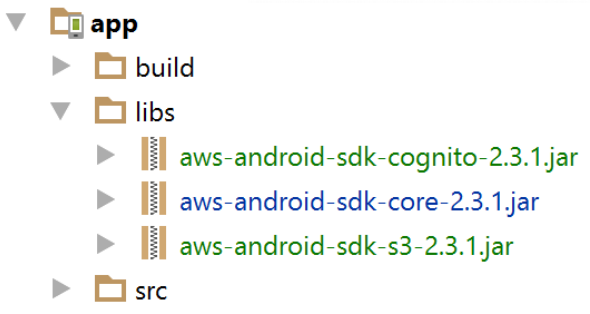
```java
// http://wiki.fasterxml.com/JacksonInFiveMinutes
// http://wiki.fasterxml.com/JacksonDownload
public void jacksontest(){
   try {
        InputStream is = getResources().openRawResource(R.raw.jacksonjson);
        byte [] buffer = new byte[is.available()];
        while (is.read(buffer) != -1)
            ;

       String jsonin = new String(buffer);
       Log.d(TAG+" jacksontest", jsonin);

       // It takes two lines of Java to turn it into a User instance
       ObjectMapper mapper = new ObjectMapper(); // can reuse, share globally
       User user = mapper.readValue(jsonin, User.class);

       // Marshalling back to JSON is similarly straightforward
       String jsonout = mapper.writeValueAsString(user);
       tvData.setText(jsonin + "\n\n" + jsonout);
   }
   catch (Exception e) {
      tvData.setText(e.getMessage() + e.getStackTrace().toString());
   }
}
```



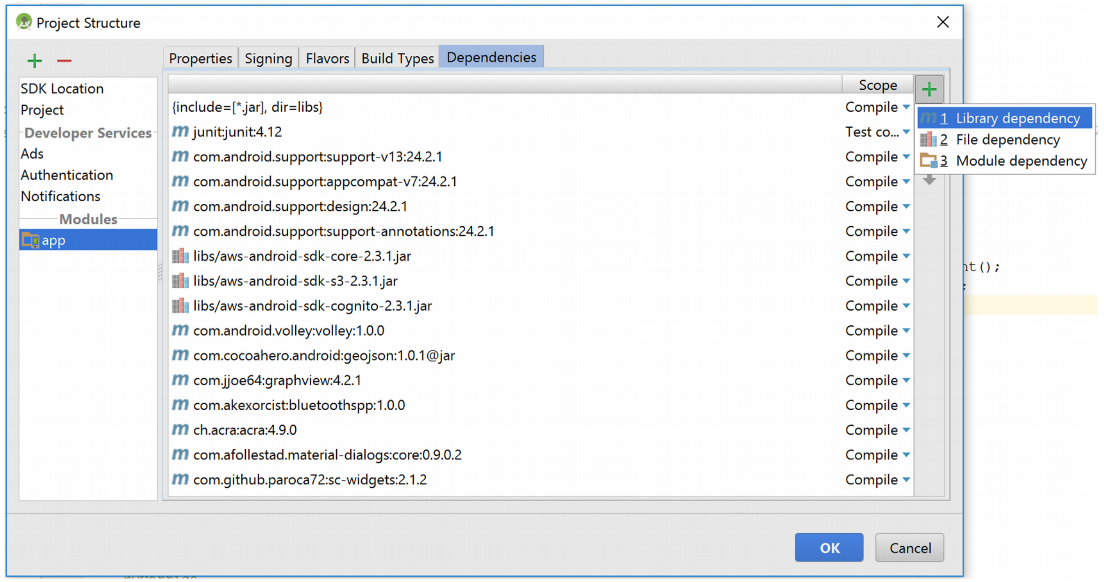
```
{"name" : {"first" : "Joe", "last" : "Sixpack" },"gender"
: "MALE","verified" : false,"userImage" :
"Rm9vYmFyIQ=="}

{"gender":"MALE","name":{"first":"Joe","last":
"Sixpack"},"userImage":"Rm9vYmFyIQ==",
"verified":false}
```

Using libraries

- Libraries is either distrubuted as built-in or external Jar files
- Or as source (Module dependency)
- Android Libraries
  - GUI
  - Game engines
  - Charts
  - Social (Facebook, Twitter, etc.)
  - RPC (JSON/XML)
  - Contacts
  - NFC
  - Networking protocols
  - …



```
▼ app
  ▶ build
  ▼ libs
    ▶ aws-android-sdk-cognito-2.3.1.jar
    ▶ aws-android-sdk-core-2.3.1.jar
    ▶ aws-android-sdk-s3-2.3.1.jar
  ▶ src
```



Project Structure

SDK Location
Project
Developer Services
Ads
Authentication
Notifications
Modules
app

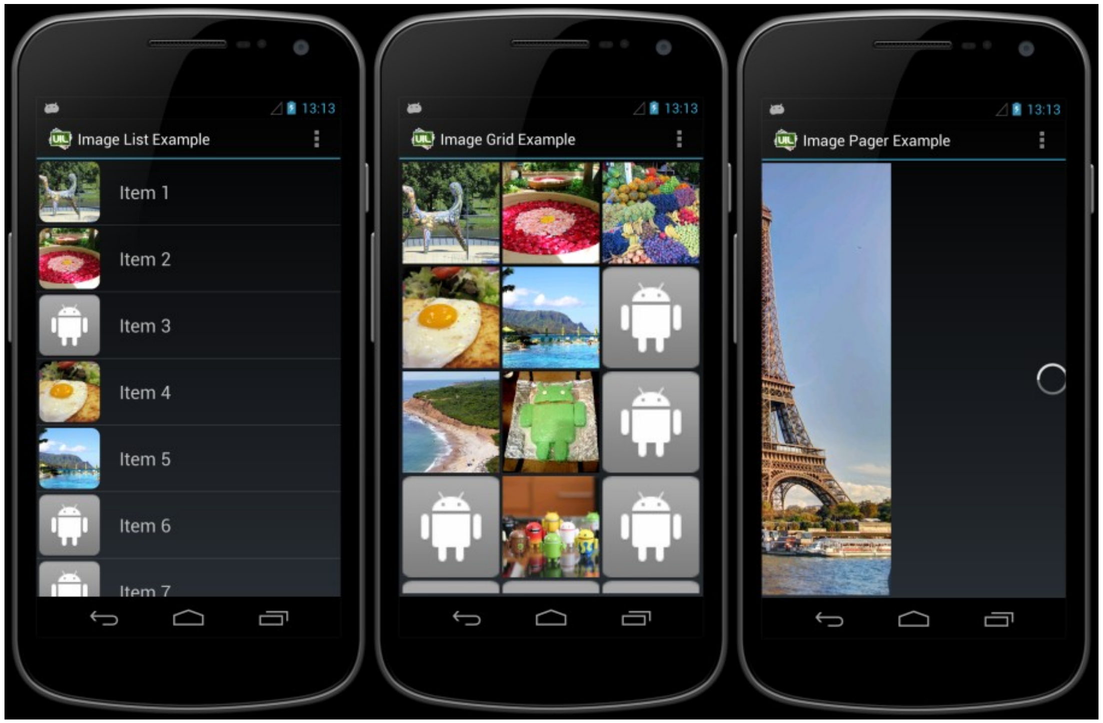
Properties | Signing | Flavors | Build Types | Dependencies

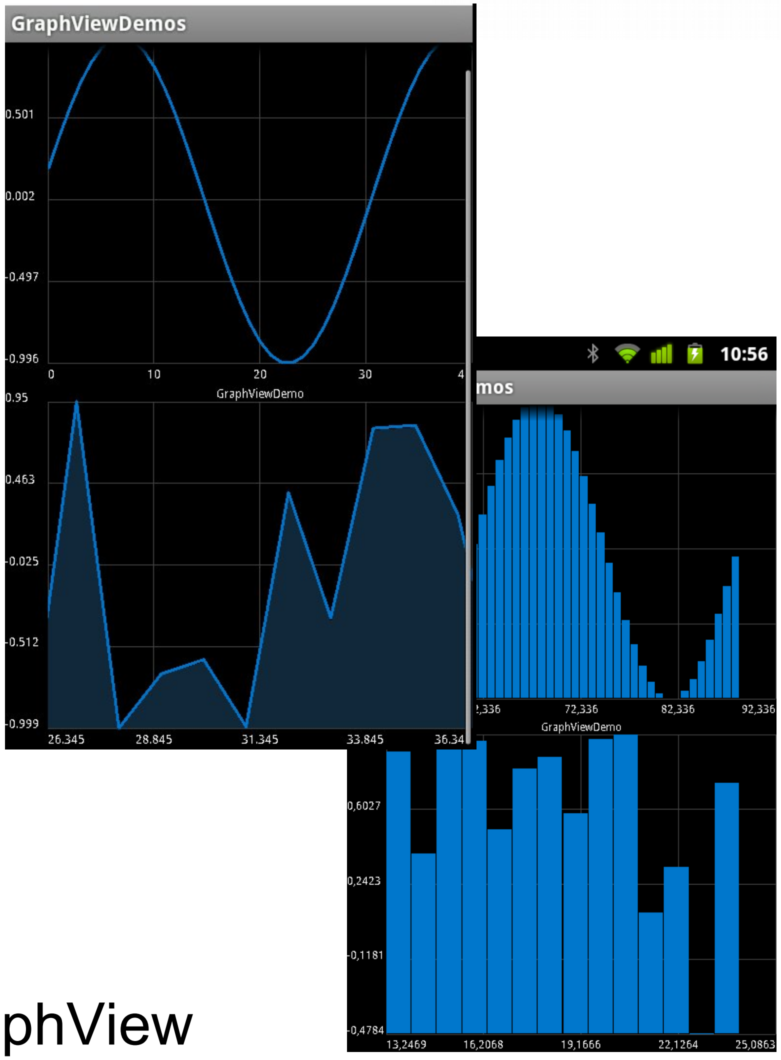
	Scope
{include=[*.jar], dir=libs}	Compile
junit:junit:4.12	Test co...
com.android.support:support-v13:24.2.1	Compile
com.android.support:appcompat-v7:24.2.1	Compile
com.android.support:design:24.2.1	Compile
com.android.support:support-annotations:24.2.1	Compile
libs/aws-android-sdk-core-2.3.1.jar	Compile
libs/aws-android-sdk-s3-2.3.1.jar	Compile
libs/aws-android-sdk-cognito-2.3.1.jar	Compile
com.android.volley:volley:1.0.0	Compile
com.cocoahero.android:geojson:1.0.1@jar	Compile
com.jjoe64:graphview:4.2.1	Compile
com.akexorcist:bluetoothspp:1.0.0	Compile
ch.acra:acra:4.9.0	Compile
com.afollestad.material-dialogs:core:0.9.0.2	Compile
com.github.paroca72:sc-widgets:2.1.2	Compile

https://android-arsenal.com/

ANDROID ARSENAL

Libraries

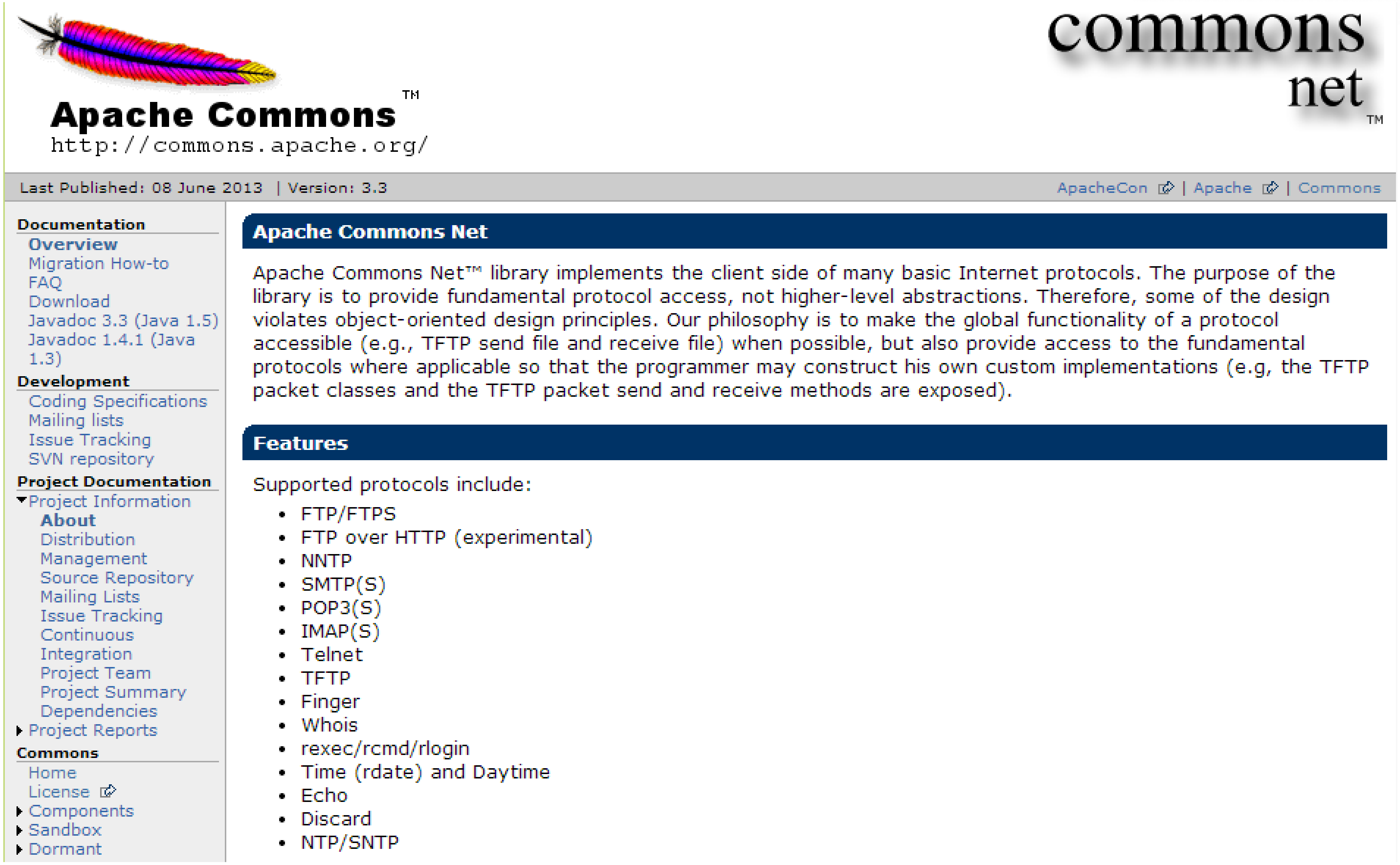




Android-Universal-Image-Loader
https://github.com/nostra13/Android-Universal-Image-Loader

GraphView
http://android-graphview.org/

Internet protocols simple



Apache Commons™
http://commons.apache.org/

commons net™

Last Published: 08 June 2013 | Version: 3.3 ApacheCon | Apache | Commons

Documentation
- **Overview**
- Migration How-to
- FAQ
- Download
- Javadoc 3.3 (Java 1.5)
- Javadoc 1.4.1 (Java 1.3)

Development
- Coding Specifications
- Mailing lists
- Issue Tracking
- SVN repository

Project Documentation
- Project Information
  - **About**
  - Distribution
  - Management
  - Source Repository
  - Mailing Lists
  - Issue Tracking
  - Continuous Integration
  - Project Team
  - Project Summary
  - Dependencies
- Project Reports

Commons
- Home
- License
- Components
- Sandbox
- Dormant

Apache Commons Net

Apache Commons Net™ library implements the client side of many basic Internet protocols. The purpose of the library is to provide fundamental protocol access, not higher-level abstractions. Therefore, some of the design violates object-oriented design principles. Our philosophy is to make the global functionality of a protocol accessible (e.g., TFTP send file and receive file) when possible, but also provide access to the fundamental protocols where applicable so that the programmer may construct his own custom implementations (e.g, the TFTP packet classes and the TFTP packet send and receive methods are exposed).

Features

Supported protocols include:

- FTP/FTPS
- FTP over HTTP (experimental)
- NNTP
- SMTP(S)
- POP3(S)
- IMAP(S)
- Telnet
- TFTP
- Finger
- Whois
- rexec/rcmd/rlogin
- Time (rdate) and Daytime
- Echo
- Discard
- NTP/SNTP

Building Apache HTTP clients

- **Deprecated!**
- The core Apache HTTP components provides a set of classes and interfaces for the HTTP protocol in org.apache.http.*
  - **HttpMessage** – the request or response message (HttpPost or HttpResponse)
  - **Header** – wrappers for the HTTP message headers
  - **HttpEntity** – the body of a message when supplied
  - **HttpConnection** – the underlying network connection
- Use of the API is simplified by using the org.apache.http.impl.* classes
  - The **DefaultHttpClient** which extends **AbstractHttpClient** provides a implementation of all the needed interfaces
- If you want to continue use Apache HttpClient you need to have this statement in your Android Studio build.gradle file

```
android {
    useLibrary 'org.apache.http.legacy'
}
```

If you do not use any modern network library you must use either java.net.HttpURLConnection or javax.net.ssl.HttpsURLConnection

Simple Apache HTTP GET

```
// basic Apache HTTP GET
// http://www.vogella.com/articles/ApacheHttpClient/article.html
public void httpGet(String uri)
{
    HttpClient client = new DefaultHttpClient();
    HttpGet request = new HttpGet(uri);
    try{
        HttpResponse response = client.execute(request);
    // Get the response, getEntity() obtains the message entity of this response, if any
        BufferedReader rd =
            new BufferedReader(new InputStreamReader(response.getEntity().getContent()));
        String line = "";
        while ((line = rd.readLine()) != null) {
          textView.append(line + "\n");
          Log.d(TAG, line);
        }
    }
    catch(IOException e) {
        Log.d(TAG, "IOException", e);
    }
    catch(Exception e) {
        Log.d(TAG, "Exception", e);
    }
    finally{
        client.getConnectionManager().shutdown();
    }
}
```

Build and make a Apache HTTP request

- Build a HTTP request
- Create a handler class to process the response (optional)
  - ResponseHandler<T> class is used to process the server response
  - Convert error status codes to ClientProtocolException
  - Convert the response data into an object of type T
- Create a DefaultHttpClient and execute the request
  - Get a server response
  - A ClientProtocolException is thrown if error
- Shut down the connection
- If not using a ResponseHandler you must handle the response

```
HttpResponse response = client.execute(post);
BufferedReader rd = new BufferedReader(new InputStreamReader(response.getEntity().getContent()));
String line = "";
while ((line = rd.readLine()) != null) {
    Log.d(TAG, line);
    if (line.startsWith("Error")) {
        String key = line.substring(5);
        Toast.makeText(this, key, Toast.LENGTH_LONG).show();
        // Do something with the key
    }
}
```

Apache POST with a ResponseHandler

```
// basic Apache form data urlencoded HTTP POST with a ResponseHandler
public void postData(String uri) {
      // Create a new HttpClient and Post Header
      HttpClient client = new DefaultHttpClient();
      HttpPost post = new HttpPost(uri); // "https://www.google.com/accounts/ClientLogin"
      try { // Add your data
            List<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>(5);
            nameValuePairs.add(new BasicNameValuePair("Email", "youremail"));
            nameValuePairs.add(new BasicNameValuePair("Passwd", "yourpassword"));
            nameValuePairs.add(new BasicNameValuePair("accountType", "GOOGLE"));
            nameValuePairs.add(new BasicNameValuePair("source","Google-cURL-Example"));
            nameValuePairs.add(new BasicNameValuePair("service", "ac2dm"));
            post.setEntity(new UrlEncodedFormEntity(nameValuePairs));
            // Handler that encapsulates the process of generating a response object from a HttpResponse.
            ResponseHandler<String> responseHandler = new BasicResponseHandler();
            // Execute HTTP Post Request
            String responseBody = client.execute(post, responseHandler);
            Log.d(TAG, responseBody.toString());
            // TODO parse the responseBody
      }
      catch (ClientProtocolException e) {
            Log.d(TAG, "ClientProtocolException", e);
      }
      catch (IOException e) {
            Log.d(TAG, "IOException", e);
      }
      catch (Exception e) {
            Log.d(TAG, "Exception", e);
      }
      finally{
            client.getConnectionManager().shutdown();
      }
}
```