

Networking in Android

Tran Giang Son, tran-giang.son@usth.edu.vn

ICT Department, USTH



Network

- What's a network? Why network?



Network

- What's a network? Why network?
- OSI Model? TCP/IP?

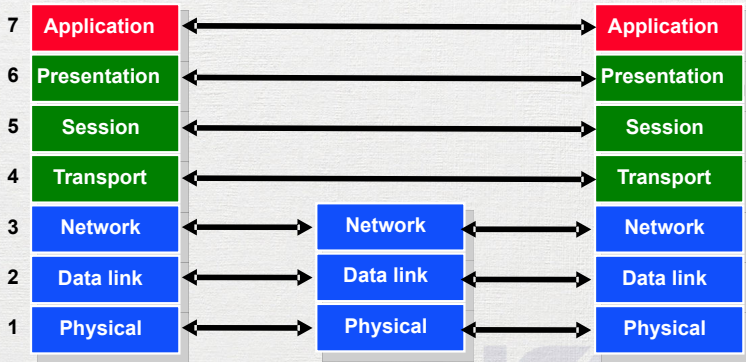


Network

- What's a network? Why network?
- OSI Model? TCP/IP?
- How to make a request in JavaScript?



Network Layering



Networking in Android

- Socket: TCP/UDP
- Protocol: HTTP / FTP / SIP / SMTP / IMAP / ...
- In this course
 - HTTP Client with JSON representation
 - Why?



Contents

- Permissions
- Embedded package/class
- External library
- Data representation



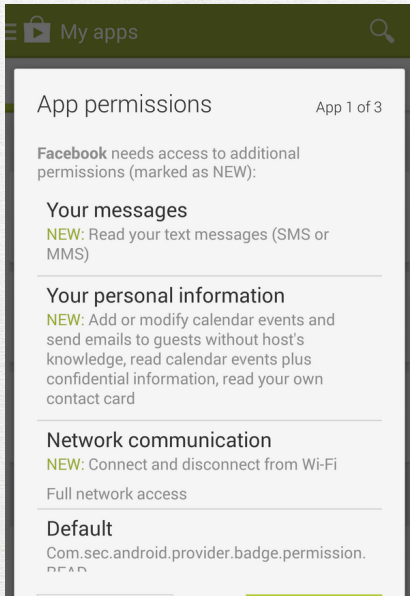
Permissions



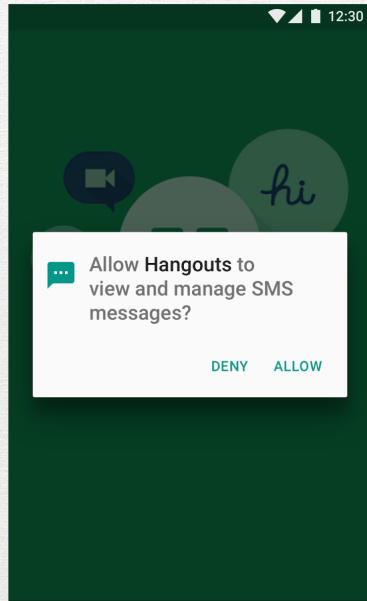
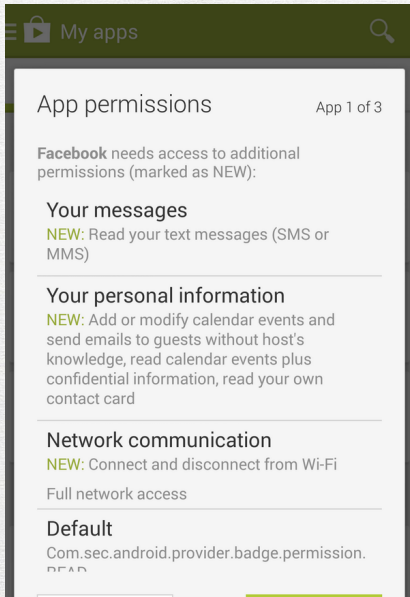
What?



What?



What?



What?

- Android has privilege-separation
- Sandboxing
 - System user ID
 - System group ID
- Specific actions require permissions



Why?

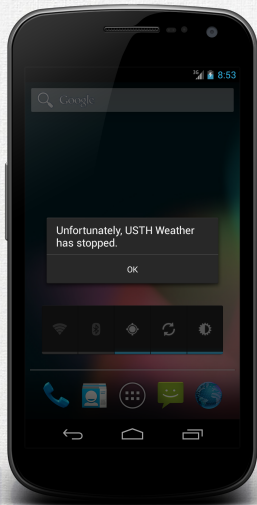
- Privacy is an important aspect
- Permission is a way to implement/improve security and privacy
- Each “sensitive” action requires a separated permission
 - Read external storage
 - Write external storage
 - Read contact list
 - ...



Why?

What happens if you don't have permission?

```
[138] NetworkDispatcher.run: Unhandled exception  
java.lang.SecurityException: Permission denied  
(missing INTERNET permission?)  
    at java.net.InetAddress.lookupHostByName(InetAddress.java:464)  
    at java.net.InetAddress.getAllByNameImpl(InetAddress.java:252)  
    at java.net.InetAddress.getAllByName(InetAddress.java:215)
```



How?

Marshmallow+ has two main levels of permissions:

- Normal: no effect on user privacy, requires user confirmation
- “Dangerous”: affect user privacy or device operations, requires confirmation



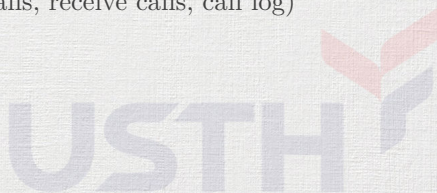
How?

- Normal level
 - Internet access
 - Read network state
 - Set timezone, set wallpaper...



How?

- Normal level
 - Internet access
 - Read network state
 - Set timezone, set wallpaper...
- Dangerous level
 - Read/write external storage
 - Access contact list
 - Access phone (make phone calls, receive calls, call log)
 - Send / receive SMS
 - Calendar, events
 - Microphone
 - Camera



How?

- Define what permissions are needed in the manifest
- For internet access

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



How?

- Define what permissions are needed in the manifest
- For internet access

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

- Normal permission, so no need special treatment
 - Request permissions at runtime
 - or reducing `targetSdkVersion`



Embedded package



Embedded package

- java.net
 - java.net.URL
 - java.net.HttpURLConnection
- java.io.InputStream



Embedded package

1. Create URL from string
2. Make a request to server
3. Receive response
4. Process response



Embedded package

1. Create URL from string

```
URL url = new URL("http://ict.usth.edu.vn/wp-content/" +  
    "uploads/usth/usthlogo.png");
```

2. Make a request to server
3. Receive response
4. Process response



Embedded package

1. Create URL from string
- 2. Make a request to server**

```
URLConnection connection =  
    (URLConnection) url.openConnection();  
connection.setRequestMethod("GET");  
connection.setDoInput(true);  
// allow reading response code and response dataconnection.  
connection.connect();
```

3. Receive response
4. Process response



Embedded package

1. Create URL from string
2. Make a request to server
3. **Receive response**

```
int response = connection.getResponseCode();  
Log.i("USTHWeather", "The response is: " + response);  
InputStream is = connection.getInputStream();
```

4. Process response



Embedded package

1. Create URL from string
2. Make a request to server
3. Receive response
4. **Process response**
 - Different response type requires different data treatment
 - Image: transform to bitmap
 - JSON/XML : parsing (later...)



Embedded package: image response

- Decode data to bitmap

```
Bitmap bitmap = BitmapFactory.decodeStream(is);
```



Embedded package: image response

- Decode data to bitmap

```
Bitmap bitmap = BitmapFactory.decodeStream(is);
```

- Show it

```
ImageView logo = (ImageView) findViewById(R.id.logo);  
logo.setImageBitmap(bitmap);
```



Embedded package: image response

- Decode data to bitmap

```
Bitmap bitmap = BitmapFactory.decodeStream(is);
```

- Show it

```
ImageView logo = (ImageView) findViewById(R.id.logo);  
logo.setImageBitmap(bitmap);
```

- Don't forget to disconnect

```
connection.disconnect();
```



Embedded package: recap

```
// initialize URL
URL url = new URL("http://ict.usth.edu.vn/wp-content/" +
    "uploads/usth/usthlogo.png");

// Make a request to server
URLConnection connection =
    (URLConnection) url.openConnection();
connection.setRequestMethod("GET");
connection.setDoInput(true);
// allow reading response code and response dataconnection.
connection.connect();

// Receive response
int response = connection.getResponseCode();
Log.i("USTHWeather", "The response is: " + response);
InputStream is = connection.getInputStream();

// Process image response
Bitmap bitmap = BitmapFactory.decodeStream(is);
ImageView logo = (ImageView) findViewById(R.id.logo);
logo.setImageBitmap(bitmap);

connection.disconnect();
```

Practical Work 15

- «Upgrade» your previous AsyncTask
- Perform a **real** network request to USTH's server
 - Download USTH logo
 - Show it on an ImageView of ForecastFragment



Embedded package

- Limitation?
 - «A lot» of code
 - No queue
 - No cache (mem-based and disk-based)



Embedded package

- How can we improve?
- «A lot» of code
- No queue
- No cache



Embedded package

- How can we improve?
 - «A lot» of code
 - No queue
 - No cache
-



Embedded package

- How can we improve?
 - «A lot» of code
 - No queue
 - No cache
-
- Code reuse



Embedded package

- How can we improve?
 - «A lot» of code
 - No queue
 - No cache
-
- Code reuse
 - Make a queue



Embedded package

- How can we improve?
 - «A lot» of code
 - No queue
 - No cache
-
- Code reuse
 - Make a queue
 - Make a cache manager



Embedded package

That's **too much**



Embedded package

That's **too much**

Re-invent the wheel



Embedded package

That's **too much**

Re-invent the wheel

There should be someone who has **already** done those



Embedded package



External Library



External Library

Volley

View [introduction](#) on YouTube



Volley: What?

- An Android HTTP Client library
- Made within AOSP [Android OpenSource Project]
- The wheel
 - Doesn't need to be reinvented ☺
 - Reuse



Volley: Why?

- Simple to use
- Powerful
- Extendable
- Cache



Volley: Why?

- Simple to use
- Powerful
- Extendable
- Cache
- Maintained by Google 😊



Volley: How?

- Add INTERNET permission, if you haven't done so
- Clone volley repository

```
implementation 'com.android.volley:volley:1.1.1'
```



Volley: How?

1. Create request queue (one per app)
2. Create request with listeners
3. Add request to queue



Volley: How?

1. Create request queue (one per app)

```
RequestQueue queue = Volley.newRequestQueue(context);
```

- One per app. How?
2. Create request with listeners
 3. Add request to queue



Volley: How?

1. Create request queue (one per app)
2. **Create request with listeners**
 - Two main request types
 - ImageRequest
 - StringRequest

```
ImageRequest imageRequest = new ImageRequest(  
    "http://ict.usth.edu.vn/wp-content/uploads/usth/usthlogo.png",  
    listener, 0, 0, ImageView.ScaleType.CENTER,  
    Bitmap.Config.ARGB_8888, null);
```

- Listener: see next slide

3. Add request to queue



Volley: How?

1. Create request queue (one per app)
2. **Create request with listeners**
 - Listener
 - Is a callback
 - 2 types: error listener and response listener

```
Response.Listener<Bitmap> listener =  
    new Response.Listener<Bitmap>() {  
        @Override  
        public void onResponse(Bitmap response) {  
            ImageView iv = (ImageView) findViewById(R.id.logo);  
            iv.setImageBitmap(response);  
        }  
    };  
};
```

3. Add request to queue

Volley: How?

1. Create request queue (one per app)
2. Create request with listeners
3. **Add request to queue**

```
queue.add(imageRequest);
```

Profit.



Volley: Recap

```
// once, should be performed once per app instance
RequestQueue queue = Volley.newRequestQueue(context);

// a listener (kinda similar to onPostExecute())
Response.Listener<Bitmap> listener =
    new Response.Listener<Bitmap>() {
        @Override
        public void onResponse(Bitmap response) {
            ImageView iv = (ImageView) findViewById(R.id.logo);
            iv.setImageBitmap(response);
        }
    };

// a simple request to the required image
ImageRequest imageRequest = new ImageRequest(
    "http://ict.usth.edu.vn/wp-content/uploads/usth/usthlogo.png",
    listener, 0, 0, ImageView.ScaleType.CENTER,
    Bitmap.Config.ARGB_8888, null);

// go!
queue.add(imageRequest);
```

Practical Work 16

- «Upgrade» your previous AsyncTask HttpURLConnection to Volley
- Perform a **real** network request to USTH's server
 - Download USTH logo
 - Show it on an ImageView in the ForecastFragment



Data Representation



Data Representation

- Previously, on MAD
 - Image URL + HttpURLConnection + AsyncTask → Bitmap
 - Image URL + ImageRequest + RequestQueue → Bitmap
- How about other kinds of data?



Data Representation: Remind

- AJAX in Web Dev?



Data Representation: Remind

- AJAX in Web Dev?
- Do we really use the «X» in AJAX?



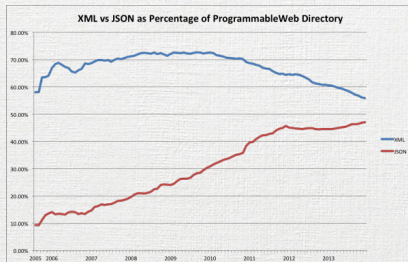
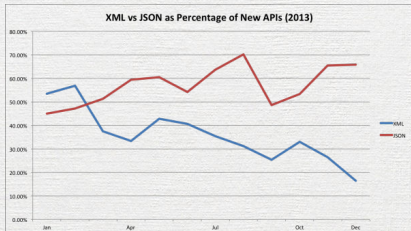
Data Representation: Remind

- AJAX in Web Dev?
- Do we really use the «X» in AJAX?
- If not, what then?



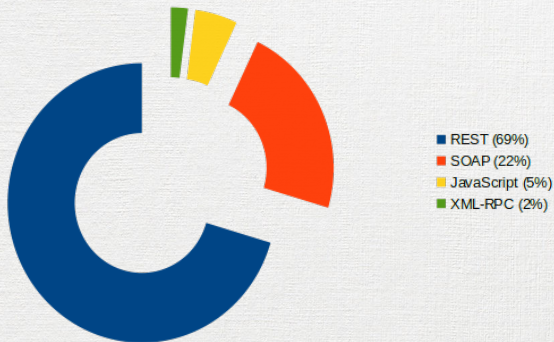
Data Representation: Trends

- XML vs JSON¹



¹Source: [ProgrammableWeb](http://ProgrammableWeb.com)

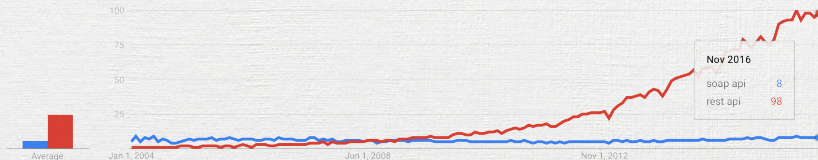
Data Representation



Backend APIs, March 2014²

²[Source: ProgrammableWeb](#)

Data Representation



Google Search trends, Nov 2016³

³Source: Google Trends

Data Representation

- JSON
 - Can represent structured data
 - Simple to use
 - Less verbose
 - Getting more attraction



Data Representation

```
{
  "query": {
    "count": 1,
    "created": "2016-11-28T08:09:31Z",
    "lang": "en-us",
    "results": {
      "channel": {
        "units": {
          "distance": "mi",
          "pressure": "in",
          "speed": "mph",
          "temperature": "F"
        },
      },
    },
  },
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<query xmlns:yahoo="http://www.yahooapis.com/v1/base.r
  yahoo:count="1" yahoo:created="2016-11-28T08:11:19
  <results>
    <channel>
      <language>en-us</language>
      <lastBuildDate>Mon, 28 Nov 2016 03:11 PM I
      <ttl>60</ttl>
      <yweather:location
        xmlns:yweather="http://xml.weather.yah
        city="H&agrave; N&#7897;i" country="Vi
      <yweather:wind
        xmlns:yweather="http://xml.weather.yah
        chill="75" direction="25" speed="7"/>
      ...
    </channel>
  </results>
</query>
```


Data Representation

- Yahoo Weather service



Data Representation

- Yahoo Weather service
- JSON: 2883 bytes



Data Representation

- Yahoo Weather service
- JSON: 2883 bytes
- XML: 4412 bytes



JSON: Examples

```
{  
  "title": "Yahoo! Weather",  
  "width": "142",  
  "height": "18",  
  "link": "http://weather.yahoo.com",  
  "url": "http://l.yimg.com/a/i/brand/purplelogo//uh/us/r  
}
```



JSON: How to get data

- Getting JSON data from server: just strings...

```
StringRequest request = new StringRequest(url,
    new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            Log.i("USTHWeather", "Json response " + response);
        }
    },
    new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
        }
    }
});
```

JSON: Parsing

- Built-in JSONObject / JSONArray
- Google's GSON
- JsonPath
- and many more... (check github!)



JSON: Parsing using built-in JSONObject / JSONArray

- JSON data

```
{  
    "title": "Yahoo! Weather",  
    "width": "142",  
    "height": "18",  
    ...  
}
```

- Java code:

```
JSONObject obj = new JSONObject(response);  
String titleValue = obj.getString("title");  
JSONObject subObject = obj.getJSONObject("location");
```

Practical Work 17

- In your WeatherFragment:
 - Ask OpenWeather for real weather
 - Decode its Json data
 - Display on your Fragment
 - Reference:
 - openweathermap.org

