B53 – Android – Faire un appel externe

## Utilité

Permet d’appeler une page Web ou un API (Application Programing Interface) et en recevoir le contenu.

Un API permet d’obtenir des données en contactant une application sur un autre serveur. Habituellement, les données retournées sont en format JSON.

## Étapes

### Ajouter la permission « Internet » pour l’application Android

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

### Ajouter la dépendance à Volley

Volley est une librairie permettant de faire efficacement des appels externes. Dans le fichier build.gradle, ajouter dans les dépendances :

dependencies {

…

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

}

### Faire un appel externe

RequestQueue queue = Volley.*newRequestQueue*(this);
String url ="https://www.google.ca";

StringRequest stringRequest = new StringRequest(Request.Method.*GET*, url,
 new Response.Listener<String>() {
 @Override
 public void onResponse(String response) {
 Log.i("App", response);
 }
 },

 new Response.ErrorListener() {
 @Override
 public void onErrorResponse(VolleyError error) {
 Log.e("App", "Erreur...");
 }
 }

);
 queue.add(stringRequest);

## JSON (JavaScript Object Notation)

JSON est une représentation des données qui est la norme pour échanger des données entre les systèmes.

Pour représenter une liste :

["Fred", "Antoine", "Marc-André", "Autre"]

Pour représenter un objet :

{

 nom : "Fred",

 role : ["Prof", "DBA"],

 lieu : "Montreal,

 no : 233,

 actif : true,

 cours : ["DBA", "Web", "Linux"]

}

### Exemple d’implémentation

Prenons l’exemple de l’objet ci-dessus qui serait dans une variable de type String nommée « response ».

try {
 JSONObject data = new JSONObject(response);
 String name = data.getString("nom");
 Log.i(name); // Fred

 // Pour lire un tableau

 JSONArray tab = jObject.getJSONArray("cours");

 StringBuilder coursesTxt = new StringBuilder();

 for (int i=0; i < tab.length(); i++) {
 String course = tab.getString(i);
 coursesTxt.append(course);
 coursesTxt.append("\n");
 }

 Log.i(coursesTxt.toString()); //DBA\NWEB\NLinux\n

}
catch (Exception e) {

}

## Divers : Avoir le positionnement (latitude/longitude) et la ville

Pour utiliser le GPS du téléphone, il faut d’abord ajouter la permission au programme :

Dans le manifest.xml

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"/>

Ensuite, dans l’application, il faut également demander la permission :

int permissionCheck = ContextCompat.*checkSelfPermission*(this, Manifest.permission.*ACCESS\_FINE\_LOCATION*);
if(permissionCheck != PackageManager.*PERMISSION\_GRANTED*) {
 ActivityCompat.*requestPermissions*(this, new

 String[]{Manifest.permission.*ACCESS\_FINE\_LOCATION*}, 1);
}

Il ne reste qu’à écouter et attendre les mises-à-jour de position

public class MainActivity extends AppCompatActivity **implements LocationListener {**

@Override
protected void onCreate(Bundle savedInstanceState) {

 ...

 try {

 LocationManager locationManager = (LocationManager)

 getSystemService(Context.LOCATION\_SERVICE);
 locationManager.requestLocationUpdates(locationManager.*GPS\_PROVIDER*,

 0, 0, this);
 } catch (SecurityException se) {
 Log.*e*("MainActivity", "err", se);
 }

}

@Override
public void onLocationChanged(Location location) {
 Log.*d*("MainActivity", "Latitude:" + location.getLatitude() + ", Longitude:" +

 location.getLongitude());

 Geocoder geocoder = new Geocoder(this, Locale.getDefault());

 List<Address> addresses = geocoder.getFromLocation(location.getLatitude(),

 location.getLongitude(), 1);

 String address = addresses.get(0).getAddressLine(0);

 Log.d("MainActivity", address);
}

@Override
public void onProviderDisabled(String provider) {
 Log.*d*("Position","disable");
}

@Override
public void onProviderEnabled(String provider) {
 Log.*d*("Position","enable");
}

@Override
public void onStatusChanged(String provider, int status, Bundle extras) {
 Log.*d*("Position","status");
}