



ANDROID

Mini-projet :

Application de localisation

des stations de Velo'v

Semestre 4

lionel.buathier@univ-lyon1.fr

Objectif du Projet

- On souhaite afficher sous forme de liste et sur une carte, les différentes stations de vélos en libre service (velo'v) de la ville de Lyon (et éventuellement d'autres villes).
- On utilisera l'API proposée par JCDecaux qui retourne un fichier JSON contenant la localisation et l'état des différentes stations de vélos dans la ville choisie (ici Contract=Lyon) :

<https://developer.jcdecaux.com/>

⇒ Commencer par créer un compte afin d'avoir une clé d'API



1. Connexion au serveur distant

- ▶ Mettre en place l'AsyncTask nécessaire pour la connexion à l'API JCDecaux (avec votre ApiKey) et charger le fichier Json correspondant à la ville de Lyon (contract=Lyon).
- ▶ La connexion est sécurisée, il faudra utiliser la classe `HttpURLConnection`.
- ▶ Attention, le flux étant parfois très volumineux, on ne peut pas le lire avec un seul 'readLine'. Dans ce cas, il est plus judicieux d'utiliser un `StringBuilder` plutôt que de concaténer une chaîne.
En java, chaque concaténation équivaut à créer une nouvelle chaîne recevant une copie de l'ancienne et de la chaîne à ajouter.

<https://stackoverflow.com/questions/4666748/how-to-read-bufferedReader-faster>



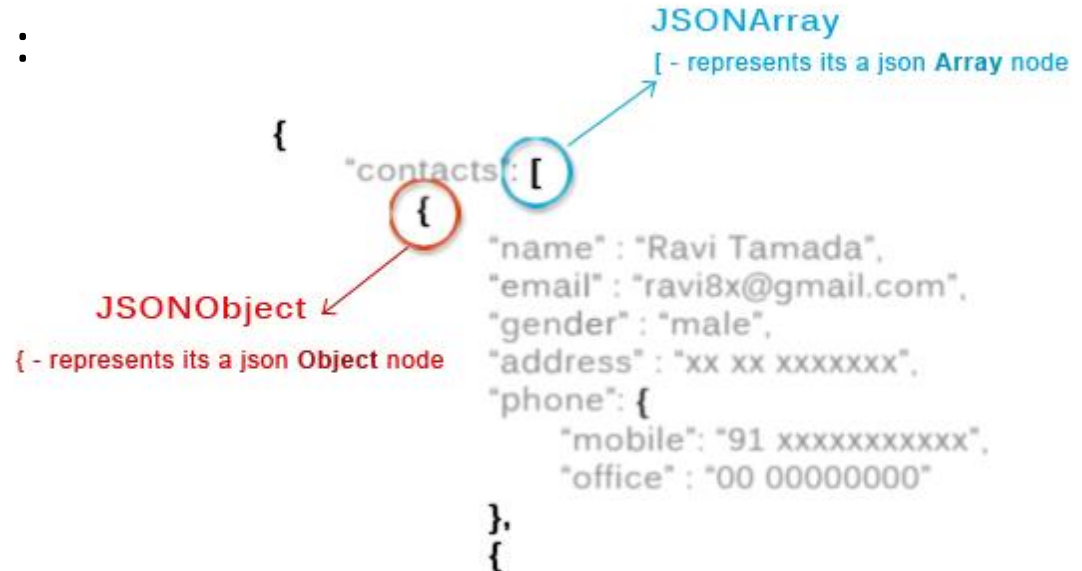
2. Parsage Json

- ▶ On peut parser le flux Json très simplement avec les classes :
 - JSONObject : lorsque le nœud commence par une accolade {
 - JSONArray : lorsque le nœud commence par un crochet [

JSON Structure

AndroidHive

- ▶ Exemple de fichier Json :



2. Parsage Json (suite)

- ▶ Dans un premier temps, vous afficherez les informations extraites dans le LogCat (avec la classe Log - cf. exemple ci-après).
- ▶ Ensuite, pour simplifier, vous pouvez commencer par mettre les données dans une chaîne et afficher son contenu dans une WebView avec la fonction loadData() ou dans un TextView.

▶ Ressources :

http://www.tutorialspoint.com/android/android_json_parser.htm

http://androidexample.com/JSON_Parsing_-

Android_Example/index.php?view=article_description&aid=71&aaid=95#

<http://stackoverflow.com/questions/9605913/how-to-parse-json-in-android>

<http://www.androidhive.info/2012/01/android-json-parsing-tutorial/>

Méthode récursive :

<http://developer.android.com/reference/android/util/JsonReader.html>



Exemple de Parsage

```
try {  
    url = new URL(host);  
    HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();  
    if (urlConnection.getResponseCode() == HttpURLConnection.HTTP_OK) {  
        // chargement du flux  
        InputStreamReader isr = new  
            InputStreamReader(urlConnection.getInputStream());  
        BufferedReader input = new BufferedReader(isr);  
        String jsonStr = input.readLine();  
        // ajouter un String Builder si plus d'une ligne  
  
        // parsing du flux  
        JSONObject jsonObject = new JSONObject(jsonStr);  
        Log.d("type", jsonObject.getString("type"));  
  
        input.close();  
    }  
    urlConnection.disconnect();  
  
} catch (...) {}
```



3. Ajout d'une classe de données

- ▶ Maintenant que vous savez "parser" le fichier, vous pouvez :
 - stocker les données dans une classe spécifique
 - ⇒ sans oublier de mettre en place les assesseurs et mutateurs (getters et setters)
 - Afficher ces données sous forme de chaîne de caractère dans une ListView, au moyen d'un Adapter de base (cf. TP5)



3bis. Rafraichir une ListView dans une AsyncTask

Main Activity

```
... void onCreate(...)
List list = new ArrayList<Prevision>();
MyAdapter adapter =
    new MyAdapter(this, list);

maListView.setAdapter(adapter);

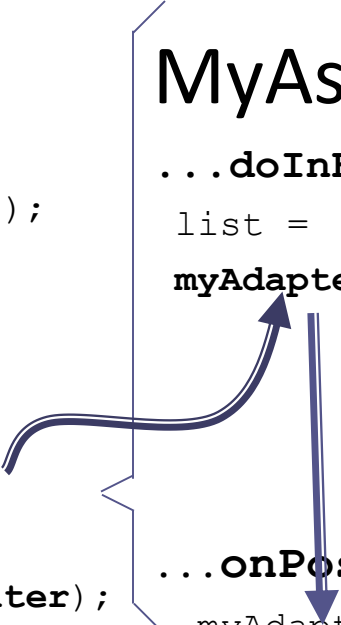
...void myOnClick(View v){
    new MyAsyncTask().execute(list, adapter);
}
```

MyAsyncTask

```
...doInBackground(Object... params)
    list = (ArrayList<Prevision>)params[0];
    myAdapter = (MyAdapter)params[1];

    // Lecture et parsing du flux
    // mise à jour de list

...onPostExecute(String s){
    myAdapter.notifyDataSetChanged();
}
```



4. Amélioration de l'affichage

- ▶ L'idée est de créer **votre propre Adapter** afin d'améliorer l'aspect graphique de l'affichage (icônes, polices, couleurs ...), tout en conservant un affichage simplifié de la liste des stations.
- ▶ Ensuite, lorsqu'on clique sur un item, proposer une nouvelle vue avec un affichage plus complet (effet de zoom) avec des détails sur la station choisie, par exemple, le nombre de vélos disponibles, d'emplacements libres, etc.
- ▶ Vous pouvez éventuellement remplacer la ListView par une RecyclerView.



5. Affichage dans Google Maps

- ▶ Une ville étant sélectionnée (par défaut ou grâce aux préférences), afficher sur une carte Google Maps, les stations et les informations essentielles s'y rapportant.
- ▶ Les détails d'une station seront affichés en cliquant sur son pictogramme
- ▶ Une fonction recherche de station pourra être proposée
- ▶ L'affichage sera, de préférence, adapté au niveau de zoom (utilisation des clusters)



5. Maps - quelques liens utiles

- ▶ Introduction à l'API Google Maps Android

<https://developers.google.com/maps/documentation/android-api/intro>

- ▶ Voir notamment les rubriques

- Controls and Gestures
- Events
- Marker
- Camera and View
- Marker Clustering

- ▶ Ajouter la dépendance vers la librairie Google Maps Android API utility (bien mettre à jour les google-services au niveau du sdk manager) :

<http://googlemaps.github.io/android-maps-utils/>



5. Maps - calcul d'itinéraire

- ▶ Une fonction calcul d'itinéraire pourra être proposée :
 - soit à partir du lieu de l'utilisateur, vers une station donnée,
 - soit d'une station vers une autre
- ▶ Un guidage par le système de navigation de Google pourra également être proposé. C'est intégré à la MAP par défaut, mais il faut le configurer en mode vélo.



6. Test de l'état de la connexion wifi ou 3G

- ▶ Ajouter un test de l'état de la connexion wifi ou 3G qui évite que l'application "plante" lors d'une requête http.
- ▶ S'il n'y a pas de connexion, on le signalera à l'utilisateur au moyen d'un Toast ou une AlertDialog.
- ▶ On pourra éventuellement recharger les dernières informations enregistrées dans un fichier local depuis la méthode onStop(), lorsque l'application est mise en arrière-plan ou arrêtée.
- ▶ On peut utiliser un WriteObject /ReadObject sur un flux ObjectOutputStream (ObjectInputStream) pour enregistrer une List ou ses dérivés (ArrayList, etc.) puisqu'elle implémente l'interface Serializable . Il faut bien sûr que les attributs et objets la constituant soient également « Serializable ».



7. Gestion des favoris et des préférences

- ▶ Ajouter la possibilité de mettre des stations en favori par un clic long sur une station. Ceux-ci seront mémorisés sur le téléphone grâce aux Shared-Preferences :

<https://developer.android.com/training/data-storage/shared-preferences.html#java>

- ▶ On pourra également prévoir un menu de gestion des préférences (PreferenceActivity) sur lequel on démarre au premier lancement de l'application ou éventuellement Navigation Drawer (un menu glissant - plus compliqué) :
 - pour choisir la ville ou d'autres paramètres de l'application

Ressources :

<http://www.ace-art.fr/wordpress/2010/07/20/tutoriel-android-partie-5-les-ecrans-de-preferences/>

<https://developer.android.com/training/implementing-navigation/nav-drawer.html>



8. Ecran d'accueil (splash screen)

- ▶ On peut simplement afficher une image, avec ou sans animation, puis lancer un Intent après un délai prédéfini.

<http://www.androidhive.info/2013/07/how-to-implement-android-splash-screen-2/>

- ▶ Mais on peut également en profiter pour charger les données pendant ce temps. Il faut alors mettre en place une méthode de call back pour se synchroniser avec la tâche asynchrone :

<https://stackoverflow.com/questions/9273989/how-do-i-retrieve-the-data-from-asyncTasks-doinbackground/14129332#14129332>

- ▶ **Remarque** : ne pas utiliser la méthode `get()` sur l'`AsyncTask`, car celle-ci attend la fin de la méthode `doInBackground` et bloque l'`UI Thread`...



Evaluation -Rendus

- ▶ L'évaluation se fera sous forme de mini présentation (10 minutes) de votre application et ensuite par l'analyse du code.
- ▶ Les critères pris en compte seront : le bon fonctionnement, la qualité du code, l'aspect ergonomique et graphique
- ▶ Vous déposerez sous Claroline un zip portant votre nom et contenant votre projet complets

