IUT Lyon 1
l'excellence technologique

Institut Universitaire de Technologie
Département Informatique
Site de Bourg-en-Bresse

ANDROID

# Mini projet
# Application d'affichage des prix des carburants

Semestre 4

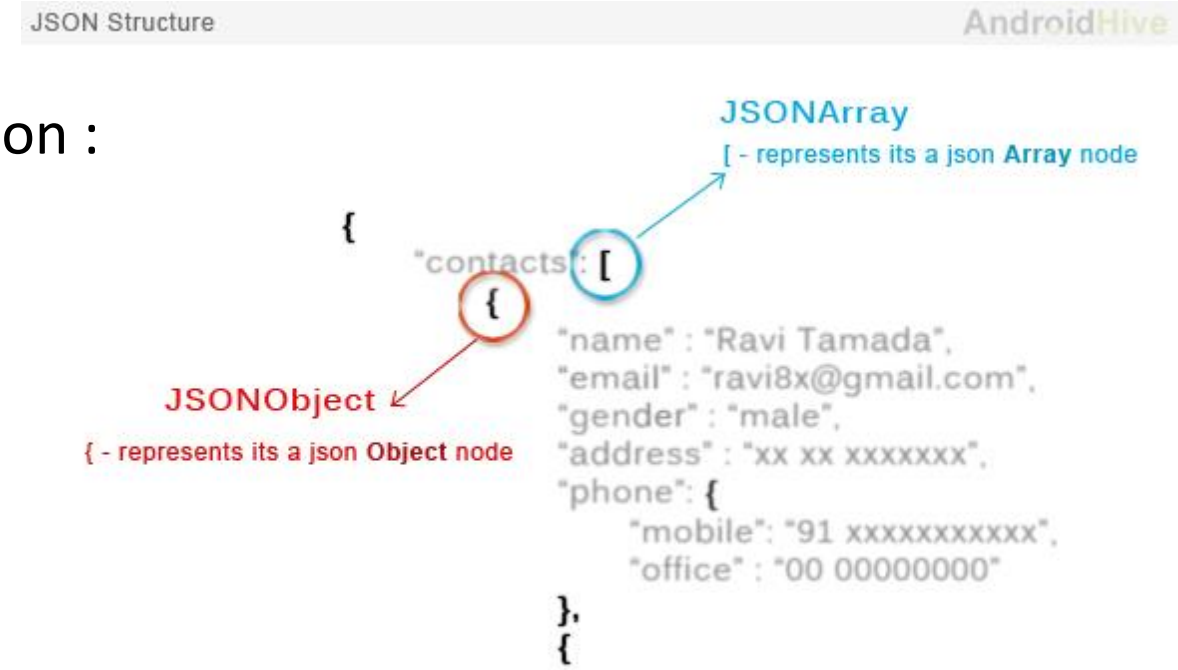
lionel.buathier@univ-lyon1.fr

# Introduction

- On souhaite réaliser une application android d'affichage des prix des carburants, à la manière du site https://www.prix-carburants.gouv.fr/
- On va procéder par étapes :
  - 1. Récupérer le flux JSON provenant du même site et mis à disposition sur le lien : http://perso.univ-lyon1.fr/lionel.buathier/carburants/PrixCarburants_instantane.json
    ou XML provenant du même site et mis à disposition sur le lien : http://perso.univ-lyon1.fr/lionel.buathier/carburants/PrixCarburants_instantane.xml
  - 2. Parser le flux et afficher les données « brutes » extraites dans une ListView
  - 3. Utiliser une classe spécifique pour les données, les trier avant de les afficher
  - 4. Améliorer l'affichage (adpater personnalisé)
  - 5. Afficher les stations avec Google Maps
  - 6. Vérifier l'état de la connexion (wifi/3G)
  - 7. Afficher un  écran d'accueil (splash screen) lors du lancement
  - 8. Ajouter un menu de gestion des préférences / favoris pour la ville à consulter

# 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 **[**




- 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 chaine 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_discription&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();

        // parsage du flux
        JSONObject jsonObject = new JSONObject(jsonStr);
        Log.d("count", jsonObject.getInt("cnt"));
        Log.d("previsions", jsonObject.getJSONArray("list").toString());

        input.close();
    }
    urlConnection.disconnect();

} catch (…) {}
```

# 2. Parsage XML avec XMLPullParser

- On peut parser le flux xml avec un des parseurs disponible pour Android (XmlPullParser, Sax, DOM, etc.)

http://developer.android.com/reference/org/xmlpull/v1/XmlPullParser.html

http://developer.android.com/training/basics/network-ops/xml.html

- Dans un premier temps, vous afficherez les informations extraites dans le LogCat (avec la classe Log).
- Ensuite, pour simplifier, vous pouvez commencer par mettre les données dans une chaine (ou ArrayList) et afficher son contenu sous forme de chaine dans une WebView avec la fonction loadData() ou dans un TextView.

# Exemple de Parsage avec XMLPullParser

```
try {
    url = new URL(" http://www.lemonde.fr/rss/une.xml");
    HttpURLConnection  urlConnection = (HttpURLConnection) url.openConnection();
    if (urlConnection.getResponseCode() == HttpURLConnection.HTTP_OK) {
        XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
        //factory.setNamespaceAware(true);
        XmlPullParser xpp = factory.newPullParser();
        xpp.setInput(urlConnection.getInputStream(), "UTF-8");
        while (xpp.getEventType() != XmlPullParser.END_DOCUMENT) {
            if ((xpp.getEventType() == XmlPullParser.START_TAG) {
                if (xpp.getName().equals("title")) {
                    Log.d("GetRSS", xpp.nextText());
                }
            }
            xpp.next();
        }
    }
    urlConnection.disconnect();

} catch (…) {}
```

# 2. XPP : Parsage des balises avec attributs

Avec **XmlPullParser**, si la donnée à extraire :

- est insérée entre 2 balises ouvrante et fermante, on peut l'obtenir par la méthode :
  - ⇨ nextText()

- est encapsulée dans l'entête de la balise, c'est un attribut. On peut obtenir :
  - ⇨ un attributs par son nom ⇨ *getAttributeValue(null,"nom")*
  - ⇨ un attributs par sa position ⇨ *getAttributeValue(int )*
  - ⇨ le nombre d'attibuts ⇨ *getAttributeCount()*

# 2(suite). Parsage XML avec SAX

- SAX s'utilise très classiquement, comme en *Java SE*.
- cf. fichier sous spiral : *Parser un fichier xml avec SAX*

- http://code.tutsplus.com/tutorials/android-sdk-build-a-simple-sax-parser--mobile-9041
- http://android-er.blogspot.fr/2010/04/simple-rss-reader-using-androids.html
- http://stackoverflow.com/questions/4827344/how-to-parse-xml-using-the-sax-parser
-

# 3. Exploitation des données

- Le flux XML fourni est très volumineux et n'est pas classés par ville.
- Pour traiter ces informations correctement, il faut les extraire et les **stocker dans une classe spécifique**.
  - ⇨ Transformer les information de la date en type Date (définir un *DateFormat* et convertir la chaine en *Date* avec ce format)
- On pourra ainsi trier les ressources,
  - ⇨ Votre classe devra implémenter la classe Comparable
  - ⇨ Surcharger la méthode *CompareTo* pour définir le tri les données
  - ⇨ Trier ensuite votre ArrayList avec *Collections.sort()*
- Pour Afficher votre ArrayList dans une ListView sans avoir à redéfinir votre propre adapter, il faut redéfinir toString().

# 3bis. Rafraichir une ListView dans une AsyncTask

- Vos données sont maintenant stockées dans une ArrayList.
- Vous pouvez donc les afficher au moyen d'une ListView
- **Rq**: depuis la méthode *onPostExecute* de l'AsyncTask, penser à signaler à *l'adapter* de *l'UIThread* qu'une donnée a été mise à jour par la méthode *notifyDataSetChanged()*

## 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,adpater);
}
```

## MyAsyncTask

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

        // Lecture et parsage 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 (icones, 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/

# 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 (ObjectIntputStream ) 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. 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 tache 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…

# 8. 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

# 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