Tache asynchrone et service web

Tache Asynchrone

Android est un système multitâche, ce qui signifie que plusieurs threads peuvent s'exécuter en parallèle. Lorsque notre application s'exécute, il y a un thread par défaut qu'on appelle *Thread principal* ou *Thread UI* (*User Interface*).

Android ne permet pas l'exécution directe de tâches coûteuses comme l'accès à des données fournies par un service Web grâce à des requêtes http. Afin de ne pas bloquer l'interface graphique quand on exécute des opérations qui prennent un peu de temps, on utilise ce qu'on appelle des *tâches asynchrones*, qui exécutent le traitement souhaité dans un thread qui n'est pas celui de l'interface.

Ceci est grandement facilité par l'utilisation de la classe `AsyncTask`, que l'on devra dériver pour créer une classe adaptée. Par exemple :

```
class GetData extends AsyncTask<String, String, String> {
    @Override
    protected String doInBackground(String... uri) {
        String responseString = "";
        URL url=null;
        StringBuilder response  = new StringBuilder();
        try {
            url = new URL(uri[0]);
            HttpURLConnection httpconn = (HttpURLConnection) url.openConnection();
            if (httpconn.getResponseCode() == HttpURLConnection.HTTP_OK)
            {
                BufferedReader input = new BufferedReader(new InputStreamReader(httpconn.getInputStream()),8192);
                String strLine = null;
                while ((strLine = input.readLine()) != null)
                {
                    response.append(strLine);
                }
                input.close();
            }
        } catch (java.io.IOException e) {
            e.printStackTrace();
        }
        return response.toString();

        }
        @Override
        protected void onPostExecute(String result) {
            super.onPostExecute(result);
            //Do anything with response..//
        }

    }
```

A quoi servent les trois paramètres `<String,String,String>` de la classe asynchrone ?

L'utilisation se fera simplement de la façon suivante :

```
String s = "http://www.service.fr/Données.php?&Code=" + code;
GetData gt = new GetData();
String jsonData = gt.execute(s).get();
```

Remarque : La méthode `get` utilisée ici présente l'inconvénient d'être potentiellement bloquante ce qui amène la nécessité d'utiliser la méthode `onPostExecute` pour des requêtes susceptibles d'avoir une réponse volumineuse ou longue à obtenir.

Notre objectif est maintenant de récupérer ces données directement depuis notre application, en faisant une requête HTTP.

Il faudra penser en premier de donne les permissions à notre application :

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

Compléments :
http://www.android-dev.fr/mise_en_place_d_une_tache_asynchrone

JSON (*Javascript Object Notation*).

La plupart des services web renvoient des données au format *XML* ou *JSON*. Cependant, *JSON* est souvent utilisé pour envoyer des données sur les mobiles parce qu'il est plus simple et moins verbeux que *XML*. Les données prennent donc moins de place, ce qui permet d'économiser de la bande passante (qui peut être rare sur certains mobiles).

Exemple de format JSON :

```
{
 "fruits": [
   { "kiwis": 3,
     "mangues": 4,
     "pommes": null
   },
   { "panier": true }
 ],
 "legumes": {
     "patates": "amandine",
     "poireaux": false
   },
   "viandes": ["poisson","poulet","boeuf"]
}
```

- {...} : les accolades définissent un objet.

"patates":"amandine" : Les guillemets (double-quotes) et les double-points définissent un couple clé/valeur (on parle de membre).

- [...] : Les crochets définissent un tableau.

Les virgules permettent de séparer les membres d'un tableau ou, comme ici, d'un objet. A noter : pas de virgule pour le dernier membre d'un objet, sinon, il ne sera pas valide et vous aurez des erreurs lors de l'analyse du fichier.

Travail demandé

Etape 1 :

Nous allons maintenant procéder à la récupération de données. Rendez-vous avec un navigateur à l'adresse suivante :

http://www.labri.fr/perso/acasteig/teaching/android/geo.php?city=Talence

La réponse fournie par le service web est bien au format *JSON*.

Pour vous familiariser avec les taches asynchrones et le JSON, créez une activité (avec un bouton et un Textview) et votre classe asynchrone Getdata.

L'appui sur le bouton appellera la classe asynchrone qui retournera la « météo » de la ville passée en paramètre.

Analysez (parsez) le résultat à l'aide de la classe *JSONObject*

Affichez quelques éléments dans *LogCat* pour vérifier que l'analyse fonctionne.

Partie 2 :

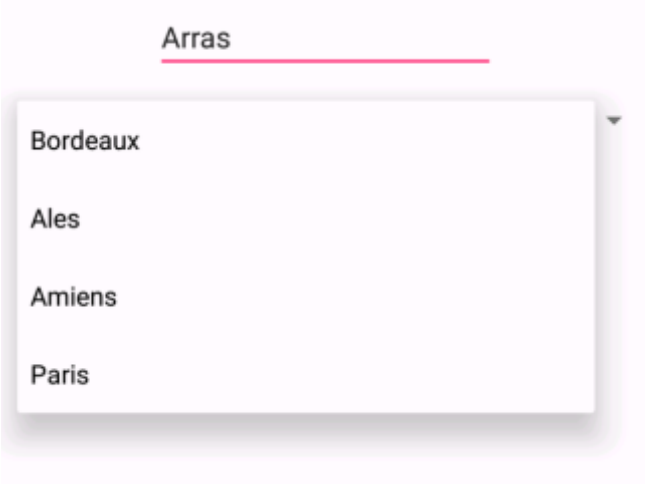
Inscrivez-vous sur le site worldweatheronline.

Nous allons utiliser leur API pour obtenir des données réelles et plus complètes (mais également plus difficile à parser).

Vous pourrez utiliser le site https://jsonformatter.curiousconcept.com/ pour visualiser les données JSON récupérées.

Modifier votre classe Getdata pour récupérer ces données météo (N'oubliez pas votre clé d'API et le format, donc lisez rapidement la doc de l'API ☺).

Maintenant, dans l'activité principale, créer un EditTtext pour saisir la ville et un spinner pour pouvoir réutiliser les villes déjà saisies.



Pour cela, on utilisera les shared preferences pour sauvegarder les villes et peupler le spinner de sélection.

Ainsi, lors de l'initialisation de l'application le spinner est déjà peuplé des villes précédentes et on ajoute systématiquement au spinner les nouvelles recherches.

Maintenant lors du parsing, vous devez au moins récupérer la température ambiante "temp_C", la description "weatherDesc" et l'adresse de l'icône "weatherIconUrl"

Pour récupérer, l'image associée à cette URL, vous devrez sans doute utiliser une autre tâche asynchrone.

Il vous reste à mettre en forme votre application ☺