IUT Lyon 1
l'excellence technologique

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

ANDROID

TP 5 : Connexion à un serveur distant - AsyncTask Affichage d'un flux RSS dans une WebView

Semestre 4
lionel.buathier@univ-lyon1.fr

1. Connexion à un serveur distant

- Le temps de réponse d'une connexion distante est toujours incertain
- Sous Android, l'interface utilisateur ne peut pas être bloquée plus de **5s**
- Par protection, Android, ne permet plus d'ouvrir une socket directement dans l'interface utilisateur, sauf dans une WebView (avec la méthode *loadUrl*)

- Il faut donc mettre en place un mécanisme asynchrone pour la connexion :
  - soit une **AsyncTask** (solution recommandée),
  - (soit un Thread, mais pas recommandé)

2. Utilisation de la classe AsyncTask

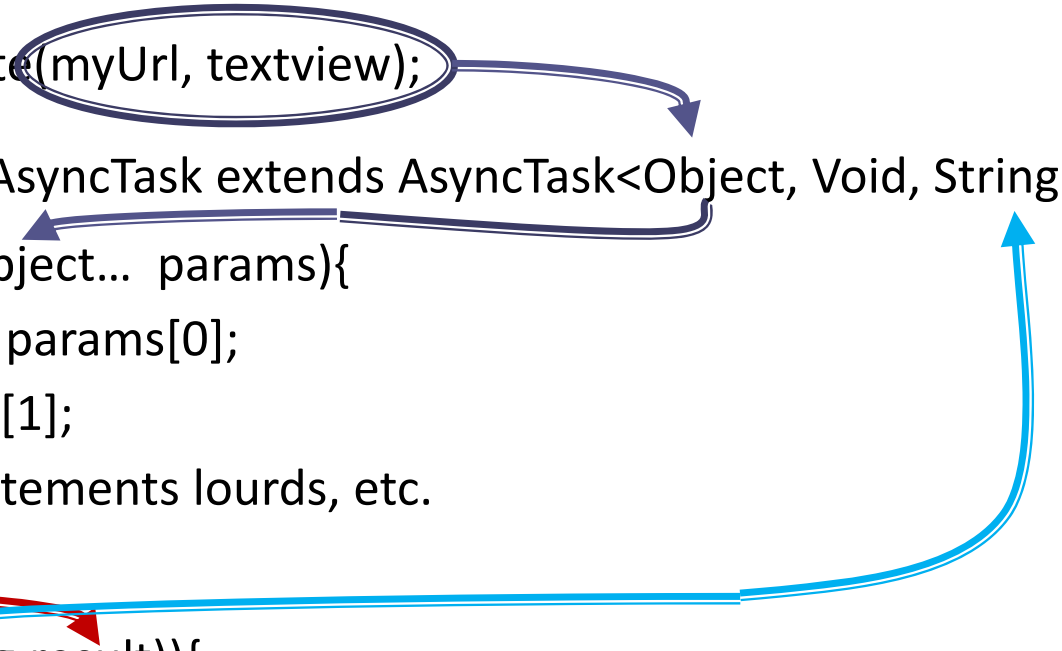
- La classe **AsyncTask<U,V,W>** basée sur 3 types génériques :
  - ⇨ U: le type du paramètre envoyé à l'exécution (reçu par l'asyncTask)
  - ⇨ V: le type de l'objet permettant de notifier de la progression de l'exécution
  - ⇨ W: le type du résultat de l'exécution retourné à onPostExecute

- AsyncTask définie 4 méthodes, dont 2 nous intéressent :
  - ⇨ **void onPreExecute() :** permet de créer une barre de progression
  - ⇨ **W doInBackground(U...):** travail dans le thread qui s'exécute en tâche de fond
  - ⇨ **void onProgressUpdate(V...)** qui permet d'actualiser une ProgressBar
  - ⇨ **void onPostExecute(W):** permet de mettre à jour l'UI, après traitement
- Ressource : http://developer.android.com/reference/android/os/AsyncTask.html

2. Utilisation de la classe AsyncTask

Exécution et passage de paramètres

- Dans l'activity :
  - new MyAsyncTask().execute(myUrl, textview);
- Dans la classe (**externe**) MyAsyncTask extends AsyncTask<Object, Void, String> :
  - String **doInBackGround**(Object… params){
    String chaine = (String) params[0];
    tv = (Textview) params[1];
    // connexions http, traitements lourds, etc.
    Return unechaine; }
  - void **onPostExecute**(String result)){
    tv.setText(result); }



3. Connexion à un serveur distant

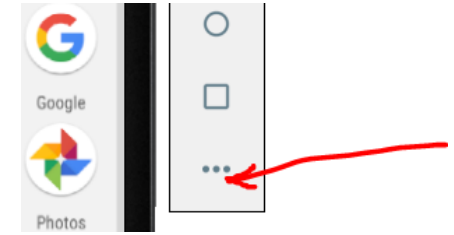
- La connexion se fait obligatoirement dans la méthode *doInBackGround* d'une AsyncTask :

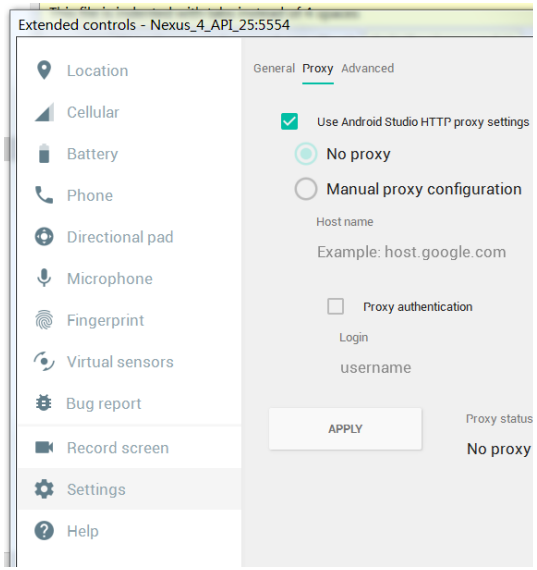
```
URL url = new URL(URL_NAME);
HttpURLConnection  urlConnection = (HttpURLConnection) url.openConnection();
    if (urlConnection.getResponseCode() == HttpURLConnection.HTTP_OK){
            BufferedReader in = new BufferedReader(
                        new InputStreamReader(urlConnection.getInputStream() )  );
          Log.d ("Async" , in.readLine());  // ou boucle tant qu'il y a des lignes
          in.close();   // et on ferme le flux
    }
     urlConnection.disconnect();
```

- La mise à jour de l'affichage se fait nécessairement dans la méthode *onPostExecute*

4. Proxy des AVD des postes de l'iut

- Sur les postes de l'iut, après avoir défini les proxys http et https pour Android Studio, il faut également paramétrer le proxy pour accéder à internet depuis les AVD.
- On peut le faire depuis les paramètres de l'AVD





Mais on peut aussi le faire en java, à l'ouverture de la connexion http, en passant une instance de Proxy à l'ouverture de la connexion :

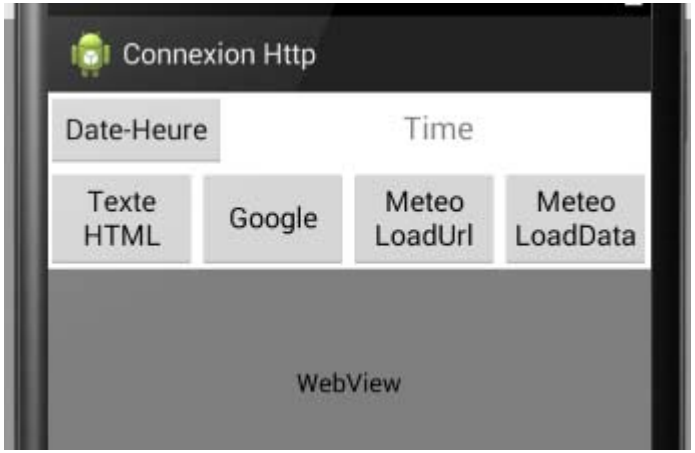
```
Proxy proxy = new Proxy(Proxy.Type.HTTP,
      new InetSocketAddress("proxy.univ-lyon1.fr", 3128));
// new InetSocketAddress("proxy.iutbourg.univ-lyon1.fr", 3128));

urlConnection = (HttpURLConnection) url.openConnection(proxy); // si proxy nécessaire
//urlConnection = (HttpURLConnection) url.openConnection(); // si pas nécessaire
```

5. Exercice : connexions http diverses

- Nous allons tester différentes manières de traiter des informations provenant de serveurs distants.

- Créer une activité avec :
  - 5 boutons
  - un TextView
  - une WebView



- Donner au préalable la permission à l'application de se connecter à l'internet dans le manifest.xml

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

5.1 Connexion à un serveur fournissant la date et l'heure courante avec 'AsyncTask'

- **Bouton Date-Heure :**

  On veut afficher dans un TextView la date et l'heure retournées par le serveur :
  http://perso.univ-lyon1.fr/lionel.buathier/time/

  Lorsqu'on appuie sur un bouton 'Date Time', exécuter une AsyncTask qui :
  - dans doInBackgroung() :
    - Ouvre une connexion http de type HttpUrlConnection,
    - Ouvre un flux en entrée et récupérer la chaine envoyée par le serveur
    - Ferme le flux et la connexion

    dans onPostExecute() :
    - affiche la chaine (date/heure) retournée par le serveur dans un TextView
- Attention à bien afficher sur le TextView dans la méthode 'onPostExecute ' afin de laisser Android gérer l'affichage dans l'UI Thread
- N'oubliez pas de donner la permission pour la connexion à l'internet (manifest.xml)

http://developer.android.com/reference/java/net/HttpURLConnection.html

5.2 Chargement direct dans une WebView

- **Bouton Text HTML :**
  - On peut également afficher directement une chaine HTML avec LoadData (si chaine stockée en local, donc chargement rapide) :

```
String strHtml = "<html><body><strong> <em>Ceci est un texte au format
   HTML </em> </strong></br>qui s'affiche très simplement</body></html>";
myWebview.loadData(strHtml , "text/html; charset=utf-8", “UTF-8”);
```

- **Bouton Google :**
  - La méthode loadUrl de la WebView permet d’afficher le contenu d’une page web directement dans l’activité :

```
myWebview.loadUrl("http://www.google.fr/");
```

Lorsqu’on charge le site de google, un navigateur est automatiquement lancé.
On peut désactiver cette option par la ligne de commande suivante :

```
myWebview.setWebViewClient(new WebViewClient());
```

5.3 Connexion au flux RSS d'actualité

- **Bouton Actu LoadUrl :**

Afficher le flux « brut », avec loadUrl(), retourné par le flux RSS d'un site de météo ou d'actualité (Le Monde ou L'équipe) :

https://www.lemonde.fr/rss/une.xml

http://www.lequipe.fr/rss/actu_rss.xml

5.4 Connexion au flux RSS d'actualité
Affichage avec loadData

- **Bouton Actu LoadData :**
- On veut maintenant lire le flux RSS du site meteorologic et l'afficher au format Html avec la méthode loadData().
  - **Une AsyncTask est donc nécessaire.**
  - Le texte affiché correspond au contenu de la balise CDATA.

- Vous simplifierez l'affichage en ne conservant que les lignes intéressantes.

Quelques liens utiles

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