



**Institut Universitaire de Technologie**

**Département Informatique**

Site de Bourg-en-Bresse



android

# **TP 6 : Connexion à un serveur distant - AsyncTask Affichage d'un flux RSS dans une WebView**

Semestre 4

**[lionel.buathier@univ-lyon1.fr](mailto: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 10s
- ▶ 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)



## 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 définit 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("URL\_RSS", textView);

- ▶ Dans Classe (**externe**) MyAsyncTask extends AsyncTask<Object, Void, String> :

- String doInBackground(Object... params){  
    String rss = (String) params[0];  
    tv = (TextView) params[1];  
    // connexions http, traitements lourds, etc.  
    Return chaine2; }  
◦ onPostExecute(String result)){  
    tv.setText(result); }



### 3. Connexion à un serveur distant

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

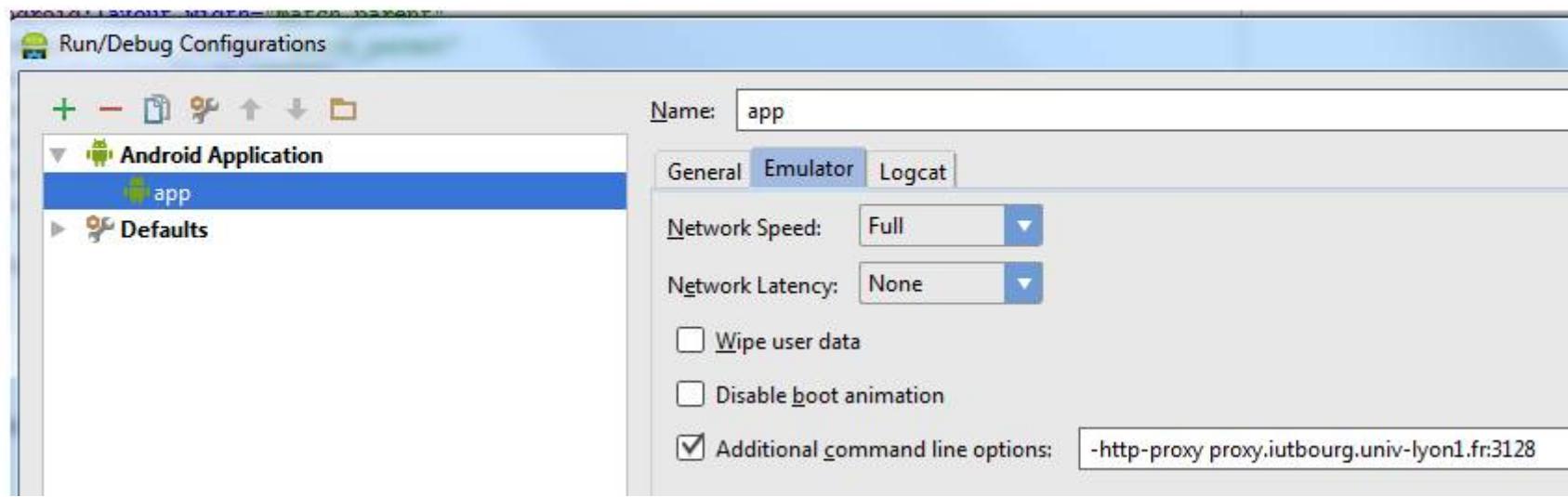
```
URL url = new URL(URL_NAME);
HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
if (urlConnection.getResponseCode() == HttpURLConnection.HTTP_OK){
    in = new BufferedReader(
        new InputStreamReader(urlConnection.getInputStream() ) );
    //ensuite, on récupère le contenu du flux avec in.readLine(), tant qu'il y a des
    //données dans le flux d'entrée
    in.close(); // et on ferme le flux
}
```

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

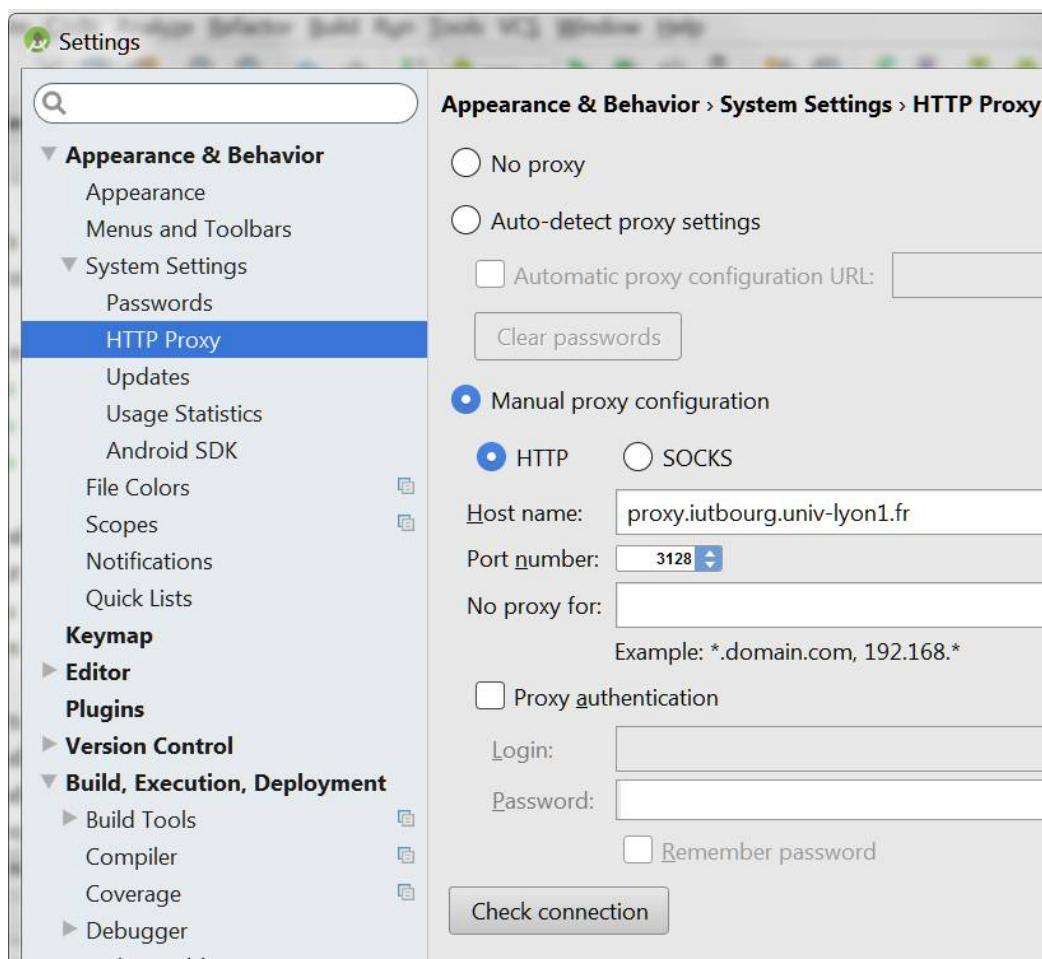


## 4. Proxy des postes de l'iut

- ▶ NB : sur les postes de l'iut, **avant de lancer l'AVD**, paramétrer le proxy de l'AVD , dans le menu Run/config d'Android Studio avec la commande :
  - ▶ -http-proxy proxy.iutbourg.univ-lyon1.fr:3128 -no-audio (pour Lyon)
  - ▶ -http-proxy proxy.univ-lyon1.fr:3128 -no-audio (pour Bourg)



# Proxy android studio

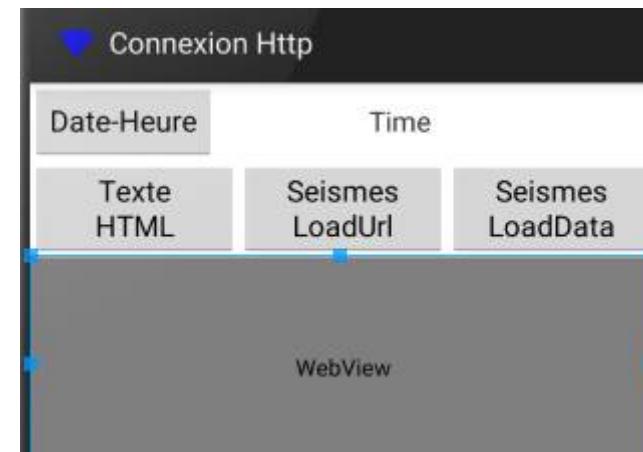


## 5. Exercice : connexions http diverses

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

- ▶ Créer une activité avec :

- 4 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 maintenant afficher dans un TextView la date et l'heure retournées par le serveur suivant : <http://www.timeapi.org/utc/now>

⇒ Lorsqu'on appuie sur un bouton 'Date Time', exécuter une AsyncTask qui :

- Dans doInBackground() :

- Ouvre une connexion http de type HttpURLConnection,
  - Ouvre un flux en entrée et récupérer la chaîne envoyée par le serveur
  - Ferme le flux et la connexion (et donc le flux)

- Dans onPostExecute() :

- affiche la chaîne (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 afficher directement une chaîne HTML avec LoadData (si chaîne stockée en local, donc chargement rapide) :

```
String strHtml = "<html><body><b> Ceci est un texte au format HTML</b><br>qui s'affiche très simplement</body></html>";
```

```
webview.loadData(strHtml , "text/html; charset=utf-8" , "UTF-8");
```



## 5.3 Affichage direct du flux RSS USGS de relevés sismiques

### ▶ Bouton Séismes LoadUrl :

La méthode `loadUrl` de la `WebView` permet d'afficher le contenu d'une page web directement dans l'activité (donc sans `AsyncTask`).

Afficher le flux « brut » retourné par le flux RSS du site USGS de relevés d'activités sismiques, avec la méthode `loadUrl` :

[http://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/4.5\\_day.geojson](http://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/4.5_day.geojson)



## 5.4 Connexion au flux RSS USGS de relevés sismiques et affichage avec loadData

### ▶ Bouton Séisme LoadData :

- ▶ On veut maintenant lire le flux RSS du site USGS de relevés d'activités sismiques, et l'afficher au format avec la méthode loadData().
  - Une AsyncTask est nécessaire.
- ▶ Attention, le flux étant parfois très volumineux, on ne peut pas le lire avec un seul 'readLine'.



## Quelques liens utiles

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

