ESIGELEC irseem

Android et Innovation

Adnane Cabani

D1 272

cabani@esigelec.fr

Les services web REST

Disponibilité du réseau

- Utiliser la classe **ConnectivityManager**
  - Superviser l'état des connexions (Wi-Fi, GPRS, UMTS, etc)
  - Diffuser des intentions quand la connectivité change
  - Tenter de basculer vers un autre réseau lorsque la connexion à un réseau est perdue
  - Fournir une API qui permet aux applications d'interroger l'état à gros grain ou à grain fin des réseaux disponibles

Exemple

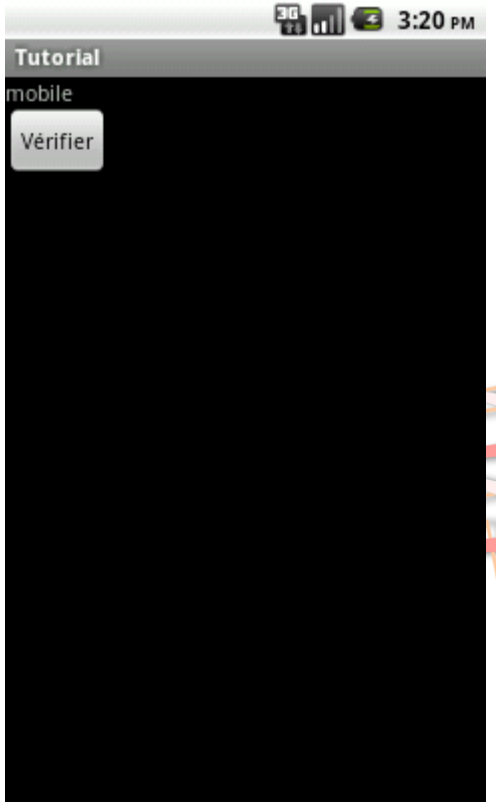
```java
package com.cabani.exemple;

import android.app.Activity;

public class MonActivite extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button btn = (Button) findViewById(R.id.button1);

        btn.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                // Appel du manager de connexion
                ConnectivityManager connectivityManager =
                        (ConnectivityManager) getSystemService(CONNECTIVITY_SERVICE);
                NetworkInfo networkInfo = connectivityManager.getActiveNetworkInfo();
                TextView tv = (TextView) findViewById(R.id.textView1);
                // Retourne le type de connexion mobile ou WiFi
                if (networkInfo != null)
                    tv.setText(networkInfo.getTypeName());
                else
                    tv.setText("Pas de réseau !");
            }
        });
    }
}
```



AndroidManifest.xml

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

Requête HTTP de type GET

- Ne jamais effectuer un traitement long dans le UI Thread

➔Utiliser un thread séparé (la classe **AsyncTask**)

La classe AsyncTask

- Classe abstraite et générique

```
public class MaClasse extends AsyncTask<Params, Progress, Result>
```

Trois paramètres :

- Params : les objets sur lesquels on va faire une opération
- Progress : type de l'objet qui transmettra l'état d'avancement du traitement
- Result : type de l'objet qui encapsulera le résultat

La classe AsyncTask

- Implémenter au moins la méthode **doInBackground(Params…)** où le traitement long est effectué

- Une tâche asynchrone passe par 4 étapes :
  - **onPreExecute()** : initialiser le traitement
  - **doInBackground(Params…)** : effectuer le traitement
  - **onProgressUpdate(Progress…)** : exécuté par le UI Thread
  - **onPostExecute(Result)** : invoqué par le UI Thread une fois le doInBackground terminé. Sert pour la màj de l'interface

Tâche asynchrone

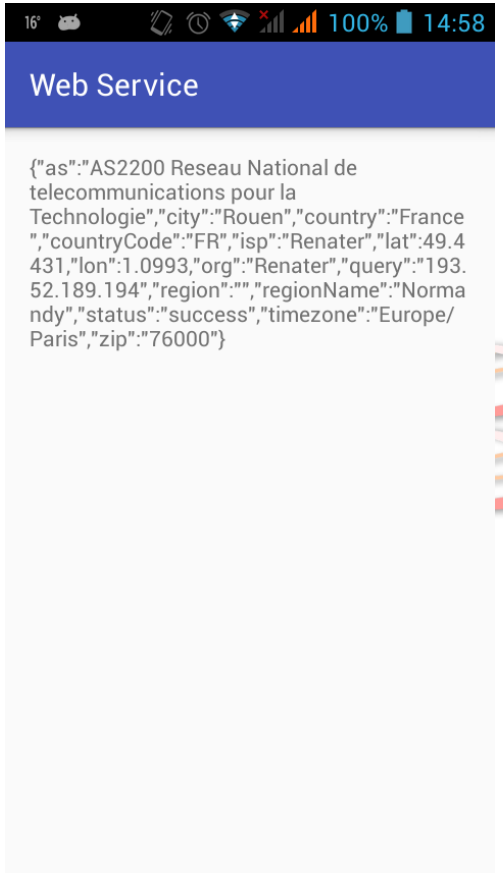
Exemple

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        TextView tv = (TextView) findViewById(R.id.textView);
        tv.setText("onCreate");
        new RequeteHttp().execute(tv);
    }

}
```



```java
public class RequeteHttp extends AsyncTask<TextView, Void, String> {

    TextView tv;

    @Override
    protected String doInBackground(TextView... params) {
        this.tv = params[0];
        return executerRequete();
    }

    @Override
    protected void onPostExecute(String result) {
        tv.setText(result);
    }

    public String executerRequete() {
        HttpURLConnection urlConnection = null;
        String webcontent = null;
        try {
            URL url = new URL("http://ip-api.com/json/193.52.189.194");
            urlConnection = (HttpURLConnection) url.openConnection();
            InputStream in = new BufferedInputStream(urlConnection.getInputStream());
            webcontent = generateString(in);
            Log.d("TP", webcontent);
        } catch (IOException e) {
            e.printStackTrace();
        }
        finally {
            if (urlConnection != null) {
                urlConnection.disconnect();
            }
        }
        return webcontent;
    }

    private String generateString(InputStream stream) {
        InputStreamReader reader = new InputStreamReader(stream);
        BufferedReader buffer = new BufferedReader(reader);
        StringBuilder sb = new StringBuilder();
        try {
            String cur;
            while ((cur = buffer.readLine()) != null) {
                sb.append(cur + System.getProperty("line.separator"));
            }
            stream.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return sb.toString();
    }
}
```