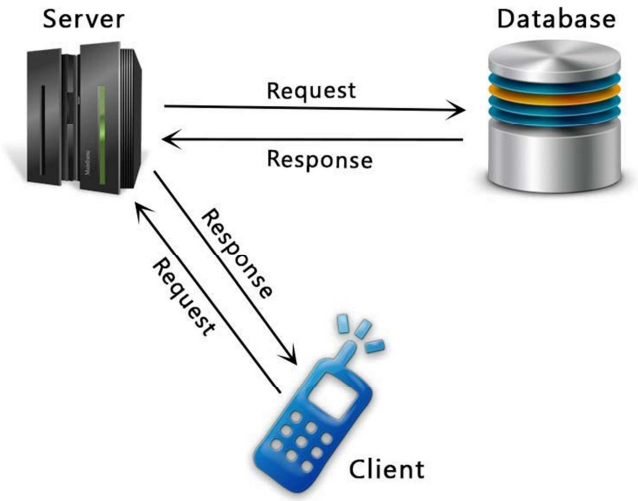
Android
Interroger un Web Service Json



Contenu

Contexte

Nous allons voir ici comment récupérer des données contenues dans une base de données intranet par le biais d'un service web accessible d'Internet. Les informations transiteront sous forme d'objets JSON, facilement manipulable par le langage Php et les terminaux mobiles Android.

La base MYSQL

La base de données interne contient une table « truc ». Cette table est constituée de 3 champs :

- Id : un identifiant clé primaire de la table (int)
- Lib : un libellé quelconque (varchar(45))
- Val : une valeur quelconque (int)

La ressource PHP (web service)

Elle est à l'adresse suivante :
http://sio.jbdelasalle.com:10380/~mtest/ws.php

Elle crée une instance d'un objet PDO afin d'interroger la base Mysql si besoin. Selon la valeur de la variable (en GET) « requete », elle va interroger la bd et retourner le résultat encode en JSON.

http://sio.jbdelasalle.com:10380/~mtest/ws.php?requete=getLesTrucs

Renvoie :

[{"id":"1","0":"1","lib":"Un premier truc","1":"Un premier truc","val":"10","2":"10"},{"id":"2","0":"2","lib":"Un deuxi\u00e8me truc","1":"Un deuxi\u00e8me truc","val":"20","2":"20"}]

C'est l'encodage JSON du contenu de la table « truc » : (Select * from truc)

La classe métier associée

Du côté de notre client Android, il semble plus pratique de créer l'équivalent de notre table truc.

Soit la classe Truc :

```
public class Truc {
    private int id ;
    private String lib ;
    private int val ;
    public static Truc trucFactory(JSONObject jsono){...}
    public Truc(int id, String lib, int val) {...}
    public int getId() { return id; }
    public String getLib() { return lib; }
    public int getVal() { return val; }
    @Override
    public String toString() { return  lib +" ("+ val+")"; }
}
```

Cette classe contient un constructeur standard, les getters, est une surcharge de toString nécessaire à l'affichage dans un listView par exemple.

Elle contient également une méthode trucFactory statique, qui prend en paramètre un objet JSONObject.

Interroger le web service

Requête http

Une reqûete http met un certain temps à s'exécuter. L'application de doit pas être « bloqué » le temps que la requête ne récupère le résultat demandé. Pour cela, la requête http doit s'exécuter dans un Thread différent du Thread principal de l'application. La classe qui va gérer l'exécution de la requête doit donc implémenter l'interface Thread, afin d'être exécutée en parallèle.

Communication entre les 2 objets

Comment la classe Threadé va prévenir le programme principal que la requête est terminée ?

Il faut un objet connu par le Thread principal ET le Thread de la requête. C'est excatement le travail d'un objet Handler. A la construction de l'objet Threadé, on lui passe la référence de l'objet appelant afin que l'objet Threadé puisse prévenir l'Objet du Thread principal quand le travail sera terminé.

```
public class WSConnexion implements Runnable {
    private Handler handler;
    private HttpClient httpClient;
    private String reponse;

    public WSConnexion(Handler handler) {
        this.handler = handler;
        this.httpClient = new DefaultHttpClient();
        Thread t;
        t = new Thread(this);
        t.start();
    }
```

Que fait le run ?

```
@Override
public void run() {
    try {
        HttpResponse httpResponse = httpClient.execute(
                new HttpGet(
                        new URI("http://sio.jbdelasalle.com:10380/" +
                                "~mtest/ws.php?requete=getLesTrucs")));
        reponse = this.lireReponse(httpResponse);
        handler.sendEmptyMessage(0);
    } catch (ClientProtocolException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } catch (URISyntaxException e) {
        e.printStackTrace();
    }
}
```

C'est lui qui « exécute » réellement la requête (ici GET). Il récupère le résultat de la requête dans un objet HttpResponse qui contient la chaîne de caractère attendue. Une fois récupéré, il peut interpréter son contenu (un/des objets JSON) grâce à la méthode « lireReponse » et prévenir le Thread principal par le biais de l'Handler que le travail est fini et qu'il peut venir récupérer le résultat quand il le souhaite.

Que contient la réponse http ?

Elle contient le flux de données envoyées par le serveur web. Afin de pouvoir manipuler plus facilement ce flux, il suffit de lire les octets 1 à 1 afin de reconstituer un String grâce au StringBuilder.

```
private String lireReponse(HttpResponse httpResponse)
        throws IllegalStateException, IOException {
    InputStream is = httpResponse.getEntity().getContent();
    Reader in = new BufferedReader(new InputStreamReader(is, "UTF-8"));
    StringBuilder sb = new StringBuilder();
    char[] buff = new char[10000];
    int i = 0;
    while (i >= 0) {
        sb.append(buff, 0, i);
        i = in.read(buff);
    }
    return sb.toString();
}
```

L'objet Threadé contient donc maintenant le String, réponse de la requête… et a prévenu le Thread principal en lui envoyant un message contenant la valeur 0.

L'Activity principale

Par le biais de son Handler qu'elle est elle-même créé avant de transmettre sa référence à l'objet chargé d'effectuer la requête, elle reçoit un message lui confirmant l'arrivée de la réponse http.

Afin de faire patienter l'utilisateur, il est habituel d'afficher des Progress Bar ou Dialog.

Le constructeur

Il doit préparer la ProgressDialog, créer l'Handler, créer et démarrer la Thread de requête en lui donnant la référence à l'Handler.

```
public class MainActivity extends ActionBarActivity {
    private ListView listView;
    private ProgressDialog progressDialog;
    private WSConnexion wsc;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        wsc = new WSConnexion(creerHandler());
        listView = (ListView) findViewById(R.id.listView);
        progressDialog = ProgressDialog.show(this
                , "On gère des trucs", "Récupération des trucs");
    }
```

La création de l'Handler

Il va tout d'abord stopper la ProgressDialog afin de rendre la main à l'utilisateur, puis travailler sur la chaîne de caractère récupérée de la requête.

Que va faire mon objet quand il va recevoir le message ?

```
private Handler creerHandler() {
    return new Handler() {
        public void handleMessage(Message msg) {
            switch (msg.what) {
                case 0:
                    progressDialog.dismiss();
                    terminerWSC();
                    break;
            }
        }
    };
}
```

Transformation Des trucs

La réponse de la requête peut-être maintenant traitée. Le constructeur de JSONArray va nous permettre de transformer la chaîne de caractères (réponse de la requête) en tableau de JSONObjet.

Puis, grâce à la méthode statique trucFactory de la classe Truc, nous allons pouvoir générer nos objets Truc afin de les stocker dans un ArrayList. L'ArrayList n'a plus qu'à être chargé dans un ListView par exemple.

```
protected void terminerWSC() {
    try {
        ArrayList<Truc> lesTrucs = new ArrayList<>();
        JSONArray jsonArray = new JSONArray(wsc.getReponse());
        for (int i = 0; i < jsonArray.length(); i++) {
            lesTrucs.add(Truc.trucFactory(jsonArray.getJSONObject(i)));
        }
        listView.setAdapter(
                new ArrayAdapter<Truc>(this
                        , android.R.layout.simple_list_item_1
                        , lesTrucs));
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

Construire un Truc à partir d'un JSONObject

Le travail est similaire à la récupération de données provenant d'une base de données.

Nous récupérons, champ par champ le résultat d'une ligne, afin de les transmettre au constructeur et de les renvoyer au programme appelant.

L'intérêt principal de rendre cette méthode statique est qu'il n'y a pas besoin d'instancier un objet Truc au préalable.

```
public static Truc trucFactory(JSONObject jsono){
    Truc truc = null;
    try {
        truc = new Truc(jsono.getInt("id"),jsono.getString("lib"),jsono.getInt("val"));
    } catch (JSONException e) {
        e.printStackTrace();
    }
    return truc ;
}
```

Résultats