Accés à une BD distante avec android en utilisant une Application Web

Plan

Références

[1] http://blog.xebia.fr/2013/07/08/android-appels-reseau-avec-volley/

[2] https://developer.android.com/training/volley/index.html

[3] https://www.androidtutorialpoint.com/networking/android-volley-tutorial/

I. Introduction

Android ne peut pas communiquer directement avec une base de données distante. Une application tierce doit s'insérer entre l'application android et le SGBD, cette application reçoit les requêtes de l'application android pour les envoyer au SGBD, par la suite il récupère le résultat et envoi la réponse à l'application android. Cette application tiers peut être une application web ou un service web ou autre.

Dans ce cours, l'application tierce est une application web. Pour appeler cette application, il existe trois solutions d'application mobile :

- utiliser une application web,
- utiliser une application hybride,
- utiliser une application native.

II. Application Web

Dans cette solution aucune ligne de code n'est écrite. L'utilisateur lance le navigateur du Smartphone puis tape l'adresse de l'application Web. Cette solution nécessite une connexion internet et elle fonctionne sur tous les Smartphones.

III. Application hybride

Cette solution consiste à ajouter un WebView à l'activité et appeler l'adresse de l'application Web.
Exemple :

```
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.webkit.WebView;
public class MainActivity extends Activity {
  private WebView webV1;
  String url="http://192.100.30.12:80/bib/index.html";
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    init();
  }
  private void init() {
    webV1=(WebView)findViewById(R.id.webV1);
    webV1.getSettings().setLoadsImagesAutomatically(true);
    webV1.getSettings().setJavaScriptEnabled(true);
    webV1.loadUrl(url);
  }
}
```

IV. Application native

Cette solution consiste à écrire une application native qui envoie des requêtes http à l'application web et reçoit les réponses dans un format xml, json ou autre …

Dans les cas à traiter, On suppose que les pages de l'application Web utilisent la méthode "POST" et que le format des réponses est "JSON".

IV.1. Android – Appels réseau avec Volley

IV.1.1. Encore une nouvelle API réseau ?

Il existe plusieurs moyens à notre disposition sur Android pour faire des appels réseaux [1]:

- On peut citer par exemple HttpURLConnection qui souffre de certains bugs sur les versions d'Android antérieures à Gingerbread (Android 3.0), dont un nous obligeant à désactiver le KeepAlive.
- Il y a également Apache HTTP Client qui fonctionne mieux pour les appareils pre-Gingerbread mais qui comporte également quelques bugs et n'est pas vraiment supporté par l'équipe Android (l'API étant trop grande et difficile à améliorer sans casser la compatibilité).

Que ce soit avec HttpURLConnection ou HTTPClient, il est nécessaire de faire les appels réseaux dans un thread différent du thread principal qui est dédié à l'interface utilisateur.

Pour cela, on utilise généralement sur Android ce qu'on appelle des AsyncTask, qui permettent d'exécuter du code en tâche de fond et de récupérer le résultat sur le thread principal pour nous permettre de mettre à jour l'interface utilisateur.

Ces AsyncTask ont également plusieurs problèmes, le plus important étant qu'ils peuvent être à l'origine de fuites de mémoire.

De plus, contrairement à la croyance populaire, les AsyncTask s'exécutent par défaut en série et non en parallèle, ce qui peut poser des problèmes de performance.

C'est pour pallier tous ces problèmes que Volley a vu le jour :

- Volley fonctionne avec un mécanisme de thread pool configurable
- Il supporte la priorisation des requêtes (par exemple, on peut vouloir définir une priorité haute pour le téléchargement de données importantes comme les informations du client, et une priorité plus faible pour le téléchargement d'images)

- Les résultats des requêtes sont automatiquement mis en cache disque et mémoire
- On peut très facilement annuler des requêtes en cours, afin d'éviter qu'elles ne soient exécutées pour rien
- Volley supporte nativement le JSON, les images, et le texte brut. Il est possible de rajouter du support à d'autres types de données facilement.
- Enfin, on a la possibilité d'implémenter ses propres mécanismes de custom retry / backoff

Attention toutefois, Volley n'est pas forcément la solution à tous vos problèmes réseaux !

Même s'il a été pensé pour être très performant sur toutes les opérations de type RPC, il est déconseillé d'utiliser Volley pour du téléchargement de données très importantes, comme les opérations de streaming ou de téléchargement de médias audio/vidéo [1].

IV.1.2. Création de la Request

Nous allons maintenant créer notre ListActivity qui appellera les serveurs REST de GitHub à l'aide de Volley pour récupérer la liste des membres de l'organisation Xebia-France

Lorsque l'on souhaite faire une requête HTTP avec Volley, il suffit d'instancier objet héritant de Request, en lui spécifiant les comportements à exécuter en cas de réussite (dans notre cas, afficher la liste des utilisateurs récupérée) et en cas d'échec (afficher un message d'erreur).

Il faudra ensuite envoyer cette Request à la RequestQueue que l'on a initialisé précédemment pour qu'elle soit exécutée.

Volley fournit par défaut les classes suivantes héritant de Request :

- StringRequest qui retourne une String à partir d'une URL
- ImageRequest, qui retourne un Bitmap
- JsonObjectRequest, qui retourne un JSONObject
- JsonArrayRequest, qui retourne un JSONArray

Il est évidemment possible de créer ses propres classes héritant de Request, afin que Volley fasse un traitement plus adapté à nos besoins.

Ici, nous souhaitons recevoir un objet de type JSONArray, nous allons donc lancer une JsonArrayRequest.

IV.2. Permission Internet

Dans AndroidManifest.xml, ajouter :

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

IV.3. Appel de la bibliothèque dans le gradule

Dans build.gradule de App, ajouter dans dependecies :

```
implementation 'com.android.volley:volley:1.1.1'
```

IV.4. Le format JSON

JSON (JavaScript Object Notation) est un format léger d'enregistrement et d'échange de données. JSON est indépendant du langage de programmation, il est auto descriptif et simple à comprendre.

Les règles syntaxiques de JSON sont :

- Les données sont des paires nom/valeur séparés par " :"
- Les données sont séparées par des virgules ","
- Les accolades délimites les objets "{}"
- Les crochets délimites les tableaux "[]"

Une valeur JSON peut être :

- Un nombre (entier ou réel)
- Une chaîne de caractère (entre "")
- Un booléen (true ou false)
- Un tableau (entre [])
- Un objet (entre {})
- null

Exemple :

```
{"liste":
   [
       {"nom":"Mohamed", "prenom":"Ben Saleh",   "age" :54},
       {"nom":"Saleh",   "prenom":"Ben Salem",   "age" :63},
       {"nom":"Salem",   "prenom":"Ben Mohamed", "age" :65}
   ]
}
```

Cet exemple contient un objet JSON qui contient un tableau "liste". "liste" contient trois objets, le premier a pour nom "Mohamed", pour prenom "Ben Saleh", pour age "54"…

IV.5. JSONObject

Cette classe modélise un objet JSON. Parmi les méthodes de JSONObject :

- *JSONObject(String jsonObjectString)*
- *String getString(String name), int getInt(String name)…*
- *JSONObject getJSONObject(String name)*
- *JSONArray getJSONArray(String name)*

IV.6. JSONArray

Cette classe modélise un tableau JSON. Parmi les méthodes de JSONArray :

- *int length() : retourne la taille du tableau*
- *JSONObject getJSONObject(int index)*
- *JSONArray getJSONArray(int index)*

IV.7. Exemple1

ListeLivres.php ne prend aucun paramètre, elle retourne un objet JSON qui contient tous les livres comme suit :

```
{
  "liste":
     [
       {"id":"1","titre":"t1","nbPge":"110",},
       …
       {"id":"9","titre":"t9","nbPge":"190",}
    ]
}
```

```
  private void remplir(){
    // Instantiate the RequestQueue.
    RequestQueue queue = Volley.newRequestQueue(this);
    String url ="http://192.168.1.6/JSON_PHP_Bib/bib/liste/ListeLivres.php";

// Request a string response from the provided URL.
    StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
        new Response.Listener<String>() {
          @Override
          public void onResponse(String response) {
            mTextView.setText("Les titres des livres sont : \n");
            try {
              JSONObject o=new JSONObject(response);
              JSONArray a = o.getJSONArray("liste");
              for (int i = 0; i < a.length(); i++) {
                //getting the json object of the particular index inside the array
                JSONObject l = a.getJSONObject(i);

                String titre=l.getString("titre");

                mTextView.append(titre+" ");
              }
```

```
            } catch (JSONException e) {
              mTextView.setText("Problème d'analyse JSON:"+e.getMessage());
            }

          }

        }, new Response.ErrorListener() {
      @Override
      public void onErrorResponse(VolleyError error) {
        mTextView.setText("Problème d'appel HTTP:"+error.getMessage());
      }

    });
// Add the request to the RequestQueue.
    queue.add(stringRequest);
  }
```

IV.8. Exemple2

La page Ajout.php permet d'ajouter un livre avec le titre et le nbPage passés en paramètre. La réponse est un fichier JSON {" Reponse" : "SUCCES"} en cas de succès et {" Reponse" : "ECHEC"} en cas d'échec.

```
  private void ajouter(){
    // Instantiate the RequestQueue.
    RequestQueue queue = Volley.newRequestQueue(this);
    String url ="http://192.168.1.6/JSON_PHP_Bib/bib/ajout/AJOUT.php";

// Request a string response from the provided URL.
    StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
        new Response.Listener<String>() {
          @Override
          public void onResponse(String response) {
            mTextView.setText("Etat : \n");
            try {
              JSONObject o=new JSONObject(response);
              String etat = o.getString("ETAT");
              mTextView.append(etat);

            } catch (JSONException e) {
              mTextView.setText("Problème d'analyse JSON:"+e.getMessage());
            }

          }

        }, new Response.ErrorListener() {
      @Override
      public void onErrorResponse(VolleyError error) {
        mTextView.setText("Problème d'appel HTTP:"+error.getMessage());
      }

    }){//adding parameters to the request
      @Override
      protected Map<String, String> getParams() throws AuthFailureError {
        Map<String, String> params = new HashMap<>();
        params.put("titre", "JSON");
        params.put("nbPage", "123");
        return params;
      }};
// Add the request to the RequestQueue.
    queue.add(stringRequest);
  }
```