Connexions réseau

Jean-Marc Lecarpentier
Université de Caen

Au programme

- Exécution de requêtes HTTP
- Utilisation de thread
- État du réseau : wifi, GSM, rien ?
- Outils Android
- Bibliothèque Volley

Problématique

- UI : thread principale
- Requêtes HTTP : impossible de prévoir le temps de réponse ⇒ risque de ANR

- Lancer les requêtes en arrière-plan
- Utilisation de callbacks
- Outils Android & Java ou bibliothèque Volley

Accès réseau

- Permissions à ajouter dans le Manifest

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

État de la connexion

- Objet Manager de connexion

```
ConnectivityManager manager = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE)
```

- Regarder quel type de connexion est présent

```
NetworkInfo net = manager.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
boolean isWifiConn = net.isConnected();
net = manager.getNetworkInfo(ConnectivityManager.TYPE_MOBILE);
boolean isMobileConn = net.isConnected();
```

- Vérifier simplement l'état de la connexion :

```
NetworkInfo net = manager.getActiveNetworkInfo();

if (net != null && net.isConnected()) {
    // on peut lancer des requêtes HTTP
} else {
    // dire à l'utilisateur que le réseau n'est pas disponible
}
```

- Toujours utiliser isConnected(), avoir la présence du réseau ne suffit pas

- Préférences utilisateur : exécuter seulement si connecté au wifi par exemple

  Voir en détails : http://developer.android.com/training/basics/network-ops/managing.html

AsyncTask

- Utilitaire pour exécuter une tâche en dehors de la thread UI
- Peut être utilisé par tout composant
- AsyncTask<Params, Progress, Result>
  mettre Void si non utilisé (par ex. si Progress non affiché)
- doInBackground(Params… *param*) pour programmer la tâche en arrière plan (worker thread) ⇒ **méthode obligatoire**
- onPostExecute(Result *result*) une fois terminé
- Possibilité de “Progress Bar” avec onProgressUpdate()
- execute() pour lancer AsyncTask à partir d’un composant

AsyncTask et HTTP

```
if (net != null && net()) {
        new FetchUrl().execute(“http://www.google.com”);
}
```

```
private class FetchUrl extends AsyncTask<String, Void, String> {
    @Override
    protected String doInBackground(String... urls) {
        // paramètre vient de l’appel à execute() : ici url[0]
        try {
          // effectuer la connexion, obtenir le contenu en InputStream
          // puis le convertir en String
            return resultString;
        } catch (IOException e) {
            return "Unable to retrieve web page. URL may be invalid.";
        }
    }
    // onPostExecute est appelé lorsque le download est fini.
    @Override
    protected void onPostExecute(String result) {
        // faire ce qu’on veut avec le résultat
   }
}
```

http://developer.android.com/reference/android/os/AsyncTask.html

Effectuer la requête

- Choisir un client HTTP : classe HttpURLConnection (pour applications API > 9 Gingerbread)

- Effectuer la connexion

- Récupérer le contenu en objet InputStream

- Convertir le résultat en String ou autre type, par ex. en Bitmap pour une image

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

DownloadManager

- Classe pour gérer les requêtes longues (Download)
- Gestion en tâche de fond, thread séparée
- Gestion des erreurs, enregistrement du fichier

http://developer.android.com/reference/android/app/DownloadManager.html

Bibliothèque Volley

- Simplifie les requêtes HTTP
- Travaille en arrière-plan
- Possibilité de file d'attentes (*Request Queues*)
- Non prévue pour les longs téléchargements (utiliser DownloadManager)

Installer Volley

- Importer Volley dans AndroidStudio

  - git clone https://android.googlesource.com/platform/frameworks/volley
  - AndroidStudio menu File > Import Module… et choisir le dossier volley
  - Fichier settings.gradle :

    include **':app'**, **':volley'**

- Déclarer Volley comme dépendance

  - Fichier build.gradle de l'app, mettre dans la partie dependencies { … }

    compile project(**':volley'**)

- Ou bien ajouter le fichier .jar dans le dossier libs/ de votre projet

  - Fichier build.gradle de l'app doit avoir dans la partie dependencies { … }

    compile fileTree(dir: **'libs'**, include: [**'*.jar'**])

- C'est prêt !

Effectuer une requête

- Créer une file : raccourci newRequestQueue

```
RequestQueue queue = Volley.newRequestQueue(getApplicationContext()); // ou this
String url =“http://www.google.com”;
```

- Créer un objet de requête avec ses callbacks :

```
StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
        new Response.Listener() {
    @Override
    public void onResponse(Object response) {
        // response.toString() donne le code HTML reçu
    }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        // code à exécuter si la requête échoue
    }
});
```

- Ajouter la requête à la file

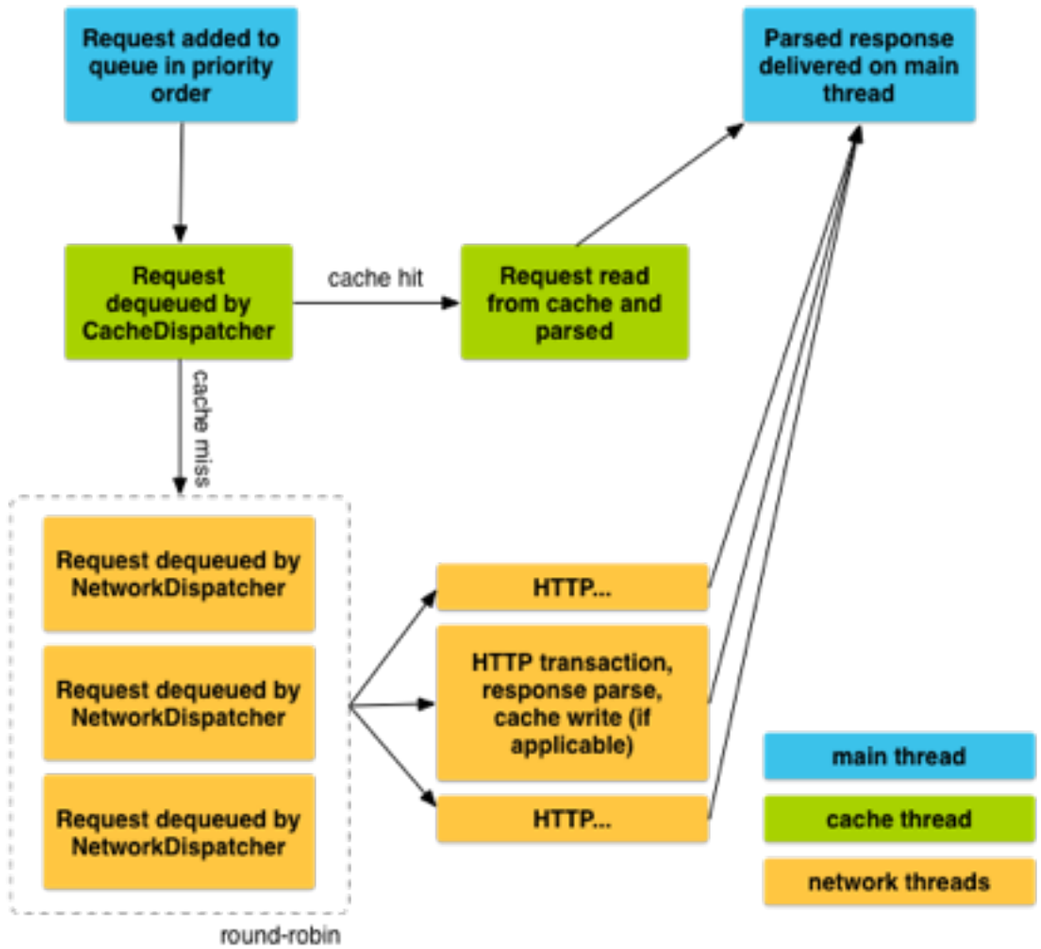
```
queue.add(stringRequest);
```

Gestion des *threads*

- Possibilité de lancer une requête depuis n'importe quel *thread*

- Mais réponse **toujours** postée au *thread* UI (*main thread*)
  ⇒ ne pas utiliser Volley pour

  des opérations HTTP dont la réponse doit être traitée en arrière-plan



Annuler une requête

- Possibilité d'annuler une requête ou toutes
- Penser à le faire dans le onStop() de l'activité
  ⇒ évite les erreurs lorsque la requête est finie
- Possibilité de "tagger" les requêtes pour les annuler plus facilement

  http://developer.android.com/training/volley/simple.html#cancel

Files de requêtes

- Avoir une et une seule file de requêtes pour l'app
- Utiliser un Singleton RequestQueue

http://developer.android.com/training/volley/requestqueue.html

Requêtes de base

- Dépend du type de réponse attendue : String, Image ou JSON
- StringRequest : cf. exemple précédent
- ImageRequest : obtenir une image via son URL
- ImageLoader pour la gestion du chargement de plusieurs images
- NetworkImageView pour un affichage plus simple
- Gestion du cache

http://developer.android.com/training/volley/request.html#request-image

Requêtes JSON

- JSON : format d'échange de données avec Web Services en mode REST
- JsonArrayRequest pour récupérer une liste
- JsonObjectRequest pour obtenir un objet

http://developer.android.com/training/volley/request.html#request-json

Pour aller plus loin

- Downloads et gestion de la batterie
  http://developer.android.com/training/efficient-downloads/index.html

- Synchronisation de données entre serveur et appareil Android
  http://developer.android.com/training/sync-adapters/index.html

- Volley : créer son propre type de requête
  http://developer.android.com/training/volley/request-custom.html

En bref

- Requêtes ⇒ en dehors du *thread UI*

- Options possibles : AsyncTask, DownloadManager, Volley, etc

- Choix de la méthode en fonction des cas d'utilisation et des besoins de l'application