UNITÉ D ’ENSEIGNEMENT (UE) : DÉVELOPPEMENT MOBILE

Ch7 :

Connexions réseau Internet

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" />
```

.Objet Manager de connexion

```
ConnectivityManager connMgr =
(ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
```

État de la connexion

- Regarder quel type de connexion est présent

```
NetworkInfo networkInfo =
connMgr.getNetworkInfo(ConnectivityManager.TYPE_WIFI);

boolean isWifiConn = networkInfo.isConnected();

networkInfo = connMgr.getNetworkInfo(ConnectivityManager.TYPE_MOBILE);

boolean isMobileConn = networkInfo.isConnected();
```

- Vérifier simplement l'état de la connexion

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

if (net != null && net.isConnected()) {

    // on peut lancer des requêtes HTTP

 } else {

    // dire à l'utilisateur que le réseau n'est pas disponible

 }
```

État de la connexion

- **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 et HTTP

```
private class FetchUrl extends AsyncTask<String, Void, String>{
  @Override
  protected String doInBackground(String urls) { // paramètre vient de l'appel à execute()
    try
    {  // effectuer la connexion, obtenir le contenu en InputStream puis le convertir en String
      return resultString;     }
    catch (IOException e )     {       return "URL introuvable";     }
  }
  // onPostExecute est appelé lorsque le download est fini.
  @Override
  protected void onPostExecute ( String result)
  {     // faire ce qu'on veut avec le résultat   }
}
if ( net != null  &&  net.isConnected())
{  new  FetchUrl().execute("http://www.google.com");}
```

Effectuer la requête

- **Choisir un client HTTP : classe HttpURLConnection**
- **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)**

Effectuer une requête

.Créer une file : raccourci newRequestQueue

```
RequestQueue queue = Volley.newRequestQueue(getApplicationContext());

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

Conclusion

- **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**