université de BORDEAUX

Développement Android

Tâches asynchrones, requêtes HTTP, parsing JSON

TD3

Dans ce TD nous continuons à développer notre application. En particulier, nous récupérons des données sur internet pour mettre à jour les informations affichées.

Lorsqu'une question est précédée d'un "▷ ", vous devez consigner votre réponse dans un fichier texte. Ce fichier sera à remettre à la fin de la séance.

1 Tâches asynchrones (piste bleue)

Android est un système multi-tâche, ce qui signifit que plusieurs thread peuvent s'exécuter en parallèle. [1] Lorsque notre application s'exécute, il y a un thread par défaut qu'on appelle *Thread principal* ou *Thread UI* (User Interface). Afin de ne pas bloquer l'interface graphique quand on exécute des opérations qui prennent un peu de temps, on utilise ce qu'on appelle des tâches asynchrones, qui exécutent le traitement souhaité dans un *autre* thread.

Créez une nouvelle classe avec le code suivant. Pour l'instant, ne cherchez pas à comprendre la partie entre chevrons (`<String,Void,String>`).

```
public class MyTask extends AsyncTask<String,Void,String>{
    @Override
    protected String doInBackground(String... strings) {
        System.out.println("Hello from another Thread");
        return null;
    }
}
```

→ En vous aidant de la documentation de la classe `AsyncTask`, trouvez comment instancier et exécuter cette tâche depuis votre activité. Plus précisément, on cherchera à l'exécuter depuis la méthode `onCreate` de la seconde activité.
▷ Quelle instruction avez-vous ajouté pour créer et lancer cette tâche ?

→ Ajoutez un paramètre à l'appel et affichez-le depuis la tâche asynchrone.
▷ Quel est le type de la variable `strings` dans `doInBackground` ? Est-ce une chaîne de caractère ? Un tableau de chaînes de caractère ? Plus généralement, que signifient les trois points `...` derrière `String` ?

2 Interrogation d'un service web (piste rouge, mais guidée)

Laissons de côté notre tâche asynchrone pour le moment. Nous allons maintenant nous concentrer sur la récupération de données sur le réseau pour notre application de géolocalisation. Rendez-vous à l'adresse suivante :

1. Ce n'est pas vraiment en parallèle, mais du point de vue du programmeur, c'est pareil.

`http://www.labri.fr/perso/acasteig/teaching/android/geo.php?city=Talence`

La réponse fournie par le service web est au format JSON (Javascript Object Notation). La plupart des services web renvoient des données au format XML ou JSON. Cependant, JSON est souvent utilisé pour envoyer des données sur les mobiles parce qu'il est plus simple et moins verbeux que XML. Les données prennent donc moins de place, ce qui permet d'économiser de la bande passante (qui peut être rare sur certains mobiles).

→ Renseignez-vous à propos du format JSON et comprenez son fonctionnement.

Notre objectif est maintenant de récupérer ces données directement depuis notre application, en faisant une requête HTTP.

→ Implémentez cette requête dans le onCreate de votre seconde activité à l'aide de `HttpURLConnection`. Vous trouverez ci-dessous une méthode utile pour convertir le `InputStream` obtenu en chaîne de caractère. SELON votre version du SDK et votre plateforme, vous devrez utiliser l'une ou l'autre de ces versions :

```
protected String readStream(InputStream in) throws IOException {
    BufferedReader br = new BufferedReader(new InputStreamReader(in));
    StringBuilder sb = new StringBuilder();
    String line;
    while ((line = br.readLine()) != null)
        sb.append(line + "\n");
    br.close();
    return sb.toString();
}

protected String readStream(InputStream in) throws Exception {
    byte[] buffer = new byte [512]; // attention, limité à 512 caractères
    for (int i=0; i<512 ;i++)
        buffer[i] = '|';
    in.read(buffer);
    String s = new String(buffer, "UTF-8");
    return s.substring(0, s.indexOf('|'));
}

protected String readStream(InputStream in) throws Exception {
    ByteArrayBuffer baf = new ByteArrayBuffer(50);
    byte[] buffer = new byte[512];
    int count;
    while ((count = in.read(buffer)) != -1)
        baf.append(buffer, 0, count);
    return new String(baf.toByteArray(), "UTF-8");
}
```

→ Une fois ces traitements codés, tentez de lancer votre application. Une erreur (exception) devrait se produire et la trace d'exécution apparaître dans LogCat.
▷ Quelle est cette exception ? (Trouvez la ligne pertinente dans LogCat)
▷ Quelle solution proposez-vous ?

→ Lancez à nouveau votre application. Une nouvelle exception se produit.
▷ Quelle est cette exception ?
▷ Quelle solution proposez-vous ?

→ Lancez une dernière fois votre application.
À ce stade, vous devriez voir apparaître les données au format JSON dans LogCat.

3 Parsing JSON (piste bleue)

→ Créez une méthode vide `updateCityInfo(String result)` **dans votre seconde activité** et appelez cette méthode **depuis doInBackground** en lui passant le résultat JSON obtenu.
▷ Quel problème se pose ?
▷ Quelle solution proposez-vous ? (Vous pouvez chercher de l'inspiration avec le raccourci Alt+Insert dans `MyTask`.)

→ Une fois le problème résolu, parsez le résultat dans cette méthode à l'aide de la classe `JSONObject` (vous avez le choix entre plusieurs constructeurs, choisissez bien).

→ Affichez quelques éléments dans LogCat pour vérifier que le parsing fonctionne.

4 Mise à jour des composants graphiques (piste rouge)

On y est presque ! Tentez maintenant de modifier le texte d'un de vos composants graphiques depuis l'intérieur de `updateCityInfo()`. Tentez d'exécuter. Une nouvelle exception se produit.
▷ Quelle est cette exception ?

→ Retournez voir la documentation en ligne de la classe `AsyncTask` et prenez le temps de lire la description des trois méthodes : `onPreExecute()`, `doInBackground()` et `onPostExecute()`.

▷ Prenez vraiment le temps de comprendre...

→ Ajoutez une méthode `onPostExecute()` dans `MyTask` à l'aide du raccourci Ctrl+o.

▷ Affichez le numéro du Thread courant lors de l'exécution de `doInBackground()` et de `onPostExecute()`. Quels sont les threads correspondants ? Sont-ils les mêmes ? Comprenez.

▷ Si vous avez compris, vous savez maintenant à quel endroit il faut appeler la méthode `updateCityInfo()`. À quel endroit ? Vous savez aussi comment passer un paramètre de `doInBackground` à `onPostExecute`. Comment ?
Vous pouvez maintenant mettre vos composants à jour avec les données récupérées.

5 Wikidata (piste noire)

Cette section ne concerne que les étudiants qui ont terminé le TD avant la fin de la séance.

Le challenge est de récupérer de vraies données en utilisant l'API de Wikidata à la place des données factices utilisées jusqu'à présent.

À rendre par courriel :

Une archive zip nommée `td3-nom.zip` si vous êtes seul ou `td3-nom1-nom2.zip` si vous êtes deux, où `nom1` est le nom de famille alphabétiquement le plus petit de votre binôme, et `nom2` le suivant. L'archive doit contenir :

1. Votre fichier de réponse `.txt`
2. Votre fichier de layout `.xml`
3. Les fichiers source de votre activité `.java`