INFORMATIQUE
TECHNIQUE

Expert

Développez une application Android

Programmation en Java sous Android Studio

Téléchargement
www.editions-eni.fr

Sylvain HÉBUTERNE

eni Editions

Table des matières

Les éléments à télécharger sont disponibles à l'adresse suivante :
http://www.editions-eni.fr
Saisissez la référence de l'ouvrage **EIASAND** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Chapitre 3
Préparation du projet LocDVD

Chapitre 4
Consultation et saisie de données

Chapitre 5
Persistance des données

Chapitre 6
Contrôles avancés

Chapitre 7
Les fragments

Chapitre 8
Navigation et boîtes de dialogue

Chapitre 9
Tâches asynchrones et services

Chapitre 10
Réseau et Internet

Chapitre 11
Exploiter le téléphone

Chapitre 12
Sortir de l'application

Chapitre 13
Design avancé

Chapitre 14
Images, son et vidéo

Chapitre 15
Publier une application

Chapitre 9
Tâches asynchrones et services

1. Exécuter des actions en tâche de fond

Pour toute application, le confort et l'expérience utilisateur sont des points essentiels : une application doit, sur smartphone et tablette, réagir immédiatement à chaque sollicitation de l'utilisateur. Pour garantir une réponse optimale, la plateforme Android introduit une règle : toute application qui ne réagit pas à une demande utilisateur dans un délai de 10 secondes est réputée ne pas répondre. Dans une telle situation, une erreur ANR, pour *Application Not Responding* (« l'application ne répond pas »), est levée, et l'application est susceptible d'être arrêtée par le système.

Pour les opérations qui peuvent potentiellement prendre du temps (dont la réponse n'est pas immédiate), il est donc fortement recommandé d'effectuer des traitements asynchrones : l'opération est exécutée en arrière-plan, et l'utilisateur peut être prévenu de la progression de l'opération.

Bien qu'il soit possible de mettre en place un tel mécanisme en utilisant les classiques classes de gestion des threads de Java, la plateforme fournit une classe abstraite, `android.os.AsyncTask`, qui prend en charge la majeure partie de la mise en place d'une solution asynchrone, et allège d'autant le travail du développeur.

```
AsyncTask<Params, Progress, Result>
```

Les types génériques `Params`, `Progress`, `Result` représentent, comme détaillé ci-dessous, les types de données passés en paramètre ou retournés par les méthodes exposées par la classe.

Pour simplifier la conception d'une opération d'arrière-plan, `AsyncTask` sépare le traitement en plusieurs phases, chacune étant représentée par une méthode.

- `void onPreExecute()` : cette méthode est exécutée au lancement de la tâche asynchrone. Elle s'exécute sur le thread principal, ce qui permet de manipuler les composants de l'interface utilisateur. Il ne faut pas invoquer cette méthode directement, mais appeler la méthode `execute`, qui lance le traitement.
- `Result doInBackground(Params... params)` : cette méthode est abstraite, elle doit obligatoirement être surchargée, et s'exécute dans un thread d'arrière-plan, lorsque la méthode `onPreExecute` est terminée. C'est dans cette méthode que le traitement doit être effectué, aucune opération sur les composants de l'interface n'étant par ailleurs possible. `doInBackground` prend en paramètre un ensemble de données de type générique `Params` et doit renvoyer en retour un objet de type générique `Result`.
- `void onPostExecute(Result result)` : cette méthode est invoquée après la méthode `doInBackground`. Elle prend en paramètre l'objet de type générique `Result` renvoyé par `doInBackground`. Cette méthode étant exécutée par le thread principal, la manipulation des composants de l'interface utilisateur est possible.
- `void onProgressUpdate(Progress… values)` : cette méthode s'exécute sur le thread principal, lorsque la méthode `publishProgress` est invoquée par la méthode `doInBackground`. Elle est typiquement prévue pour gérer l'affichage d'une boîte de dialogue de progression.

Dans le projet LocDVD, l'insertion des DVD exemples est une tâche qui peut potentiellement prendre un certain temps : il n'y a ici que quelques DVD, mais on pourrait imaginer fournir un fichier plus étoffé, qui prendrait du temps à être interprété. Il est donc plus que recommandé de transformer ce traitement en traitement asynchrone.

- Éditez le fichier `MainActivity.java` qui contient la méthode effectuant l'insertion des DVD exemples.
- Dans la classe `MainActivity`, définissez une classe `AsyncReadEmbeddedData` qui étend la classe `AsyncTask<String, Integer, Boolean>`.

```
class AsyncReadEmbeddedData extends AsyncTask<String, Integer, Boolean> {
}
```

- Il faut surcharger les méthodes `onPreExecute`, `doInBackground`, `onProgressUpdate` et `onPostExecute` :

```
class AsyncReadEmbeddedData extends AsyncTask<String, Integer, Boolean> {

    @Override
    protected void onPreExecute()  {
    }


    @Override
    protected Boolean doInBackground(String... params) {
        return false;
    }

    @Override
    protected void onProgressUpdate(Integer... values) {
    }

    @Override
    protected void onPostExecute(Boolean result)  {
    }

};
```

- La méthode `doInBackground` reprend le code de la méthode `readEmbeddedData` qui effectue la lecture et l'insertion des DVD exemples. Le nom du fichier, au lieu d'être inscrit directement dans le corps de la méthode, est passé en paramètre de `doInBackground`.

Le corps de `doInBackground` est, pour l'instant, le suivant :

```
@Override
protected Boolean doInBackground(String... params) {
    String dataFile = params[0];
    InputStreamReader reader = null;
    InputStream file=null;
```

```
    BufferedReader bufferedReader=null;
    try {
        file = getAssets().open(dataFile);
        reader = new InputStreamReader(file);
        bufferedReader = new BufferedReader(reader);
        String line= null;
        while((line=bufferedReader.readLine())!=null) {
            String [] data = line.split("\\|");
            if(data!=null && data.length==4) {
                DVD dvd = new DVD();
                dvd.titre = data[0];
                dvd.annee = Integer.decode(data[1]);
                dvd.acteurs = data[2].split(",");
                dvd.resume = data[3];
                dvd.insert(MainActivity.this);
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }  finally {
        if(bufferedReader!=null) {
            try {
                bufferedReader.close();
                reader.close();
                SharedPreferences sharedPreferences =

getSharedPreferences("com.exemple.locDVD.prefs",

Context.MODE_PRIVATE);
                SharedPreferences.Editor editor =
sharedPreferences.edit();
                editor.putBoolean("embeddedDataInserted", true);
                editor.commit();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    return false;
}
```

- Pour informer l'utilisateur de l'avancement de l'initialisation, il faut publier, à chaque insertion, le nombre de DVD insérés dans la base. Pour cela, il faut implémenter un compteur, et invoquer, à chaque itération de la boucle `while`, la méthode `publishProgress` :

```
try {

    int counter = 0;

    file = getAssets().open(dataFile);
    reader = new InputStreamReader(file);
    bufferedReader = new BufferedReader(reader);
    String line= null;
    while((line=bufferedReader.readLine())!=null) {
        String [] data = line.split("\\|");
        if(data!=null && data.length==4) {
            DVD dvd = new DVD();
            dvd.titre = data[0];
            dvd.annee = Integer.decode(data[1]);
            dvd.acteurs = data[2].split(",");
            dvd.resume = data[3];
            dvd.insert(MainActivity.this);

            publishProgress(++counter);

        }
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

- `doInBackground` doit renvoyer vrai si l'insertion s'est correctement déroulée, et faux dans le cas contraire. Dans le cadre de l'application, il est en outre intéressant de rajouter une temporisation entre l'insertion de deux DVD, pour visualiser plus facilement le traitement en arrière-plan. Une version complète de la méthode est proposée ci-dessous :

```
@Override
protected Boolean doInBackground(String... params) {
    boolean result = false;
    String dataFile = params[0];
    InputStreamReader reader = null;
    InputStream file=null;
    BufferedReader bufferedReader=null;
```

```
    try {
        int counter = 0;
        file = getAssets().open(dataFile);
        reader = new InputStreamReader(file);
        bufferedReader = new BufferedReader(reader);
        String line= null;
        while((line=bufferedReader.readLine())!=null) {
            String [] data = line.split("\\|");
            if(data!=null && data.length==4) {
                DVD dvd = new DVD();
                dvd.titre = data[0];
                dvd.annee = Integer.decode(data[1]);
                dvd.acteurs = data[2].split(",");
                dvd.resume = data[3];
                dvd.insert(MainActivity.this);
                publishProgress(++counter);
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if(bufferedReader!=null) {
            try {
                 bufferedReader.close();
                 reader.close();
                 SharedPreferences sharedPreferences =
getSharedPreferences("com.exemple.locDVD.prefs",
Context.MODE_PRIVATE);
                 SharedPreferences.Editor editor =
sharedPreferences.edit();
                 editor.putBoolean("embeddedDataInserted", true);
                 editor.commit();
                 result = true;
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
   }
   return result;
}
```