Application MyLifeCycleSQL (PARTIE 2)

Comme mentionné sur le site « Android developer », toute tâche susceptible de bloquer l'interface utilisateur doit être réalisée dans un processus indépendant. Ainsi, il est recommandé que l'ouverture d'une base de données soit traitée dans une tâche asynchrone en utilisant la classe AsyncTask.

Lire à ce sujet :

Keeping Your App Responsive , en ce qui concerne les généralités sur le traitement des tâches longues au niveau de votre application, et

AsyncTask, en ce qui concerne l'utilisation de la classe AsyncTask (cf. la figure en fin de document pour un résumé des différents paramètres et méthodes à surcharger pour le fonctionnement de cette classe).

Vous définirez à l'intérieur de la classe principale de votre activité, la classe **AsyncOpenDB** qui se chargera de la connexion à votre base de données ainsi que de l'affichage d'une barre de progression indiquant l'état de chargement (cf. Fig. 3).

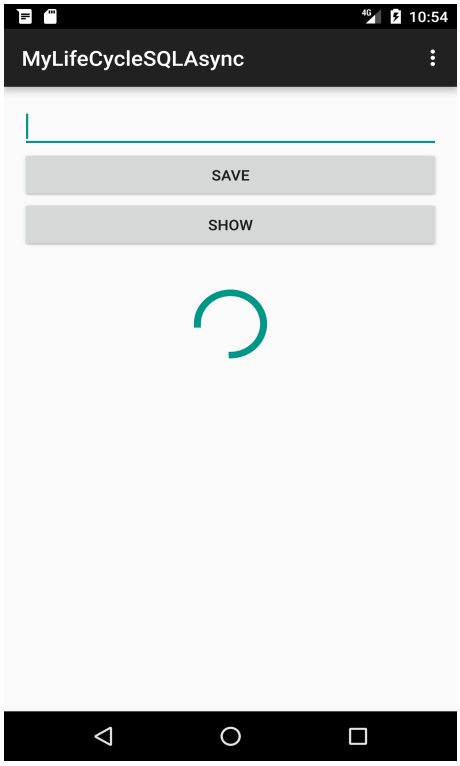


Fig. 3

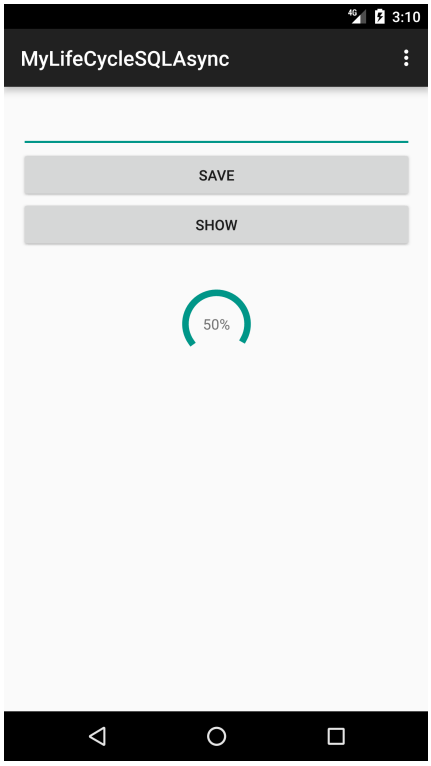


Fig. 4

La classe **AsyncOpenDB** prendra en charge 3 types de paramètres : 1 de type MyDBHelper et 2 de types Integer. Cette classe devra surcharger les méthodes suivantes :

- `doInBackground`: se charge d'ouvrir votre base de données, d'appeler la méthode

`publishProgress` afin d'envoyer un entier correspondant au pourcentage à afficher pour la barre de progression. Ce pourcentage variera par pas de 10 en partant de 0. Pour simuler un temps d'attente, vous effectuerez une pause de 1 seconde entre chaque appel à `publishProgress` en utilisant la méthode statique `Thread.sleep(long time)`. La valeur de retour de la méthode `doInBackground` sera la valeur du numéro associé à la dernière insertion dans la table **matable** ou 0 si la table est vide.

- `OnPreExecute` : se charge de gêrer l'affichage de la barre de progression. Vous utiliserez pour cela la classe **ProgressBar** (cf. ci après).
- `OnProgressUpdate` : se charge de mettre à jour l'affichage de la barre de progression avec la valeur fournie par `publishProgress`.
- `OnPostExecute` : se charge d'initialiser la valeur du champ **numero** pour la prochaine insertion en fonction de la valeur renvoyée par `doInBackground`.

Barre de progression (étape 1)

Dans un premier temps, nous allons nous intéresser à une barre de progression qui n'affiche pas le pourcentage de la progression (cf. fig. 3). Ainsi il n'y aura aucune action à prévoir pour l'instant dans la méthode **OnProgressUpdate**.

La barre de progression que vous allez utiliser n'aura pas été déclarée au préalable dans le fichier xml décrivant votre layout. Elle sera entièrement créée et gêrée par votre activité. Ceci se fera au niveau de la méthode **OnPreExecute**. Une fois l'objet de type **ProgressBar** correspondant déclaré, il va donc falloir indiquer comment l'intégrer dans votre interface. Consultez les différentes méthodes disponibles associées à cet objet sur le site de développement d'Android : La classe ProgressBar

Vous utiliserez le constructeur

```
ProgressBar(Context context, AttributeSet attrs, int defStyleAttr)
```

avec :

- pour *context* : `votre_classe_principale.this`,
- pour *attrs* : `null`,
- pour *defStyleAttr* : `android.R.attr.progressBarStyleLarge`

ceci afin d'obtenir la barre de progression de la figure 4. Vous spécifierez la zone de progression de votre barre (de 0 à 100) en utilisant les méthodes **setProgress** et **setMax**. Enfin, il faudra utiliser la méthode **setVisibility** pour rendre visible votre barre au sein de votre application. Cette dernière n'apparaîtra qu'une fois que vous l'aurez rajoutée à votre Layout principal. Pour cela, rajoutez dans votre fichier XML de description de votre application, un identifiant au layout principal. Puis dans votre application Java, déclarez un objet de type **LinearLayout** et liez-le au layout de votre fichier XML. Il ne vous reste plus qu'à utiliser la méthode **addView** de l'objet **LinearLayout** pour ajouter votre barre de progression à votre layout principal.

Barre de progression (étape 2)

Afin d'afficher le pourcentage de progression, il va falloir déclarer une TextView pour afficher le texte correspondant. Cet texte devra s'afficher au centre du cercle représentant votre barre de progression (cf. fig. 4). Pour effectuer ceci, vous allez (via votre application Java) :

- rajouter un **RelativeLayout** dans votre Layout principal qui sera de même largeur que le

Layout principal,

- créer une **TextView** qui sera centrée dans ce **RelativeLayout**,
- placer votre **ProgressBar** dans ce **RelativeLayout** en la centrant elle aussi.

L'ajout d'un **RelativeLayout** passe tout d'abord par la création d'un objet de type **RelativeLayout** :

```
RelativeLayout relative = new RelativeLayout(MonAppli.this);
```

Les caractéristiques de ce layout se définissent en utilisant l'objet statique **RelativeLayout.LayoutParams**. Vous spécifierez au constructeur que l'élément à créer possèdera les propriétés suivantes :

- largeur : s'adaptera à celle du parent (constante MATCH_PARENT de l'objet **RelativeLayout.LayoutParams**),
- hauteur : s'adaptera à son contenu (constante WRAP_CONTENT de l'objet **RelativeLayout.LayoutParams**).

Ces paramètres ainsi définies, il faut les appliquer à votre objet **RelativeLayout** en utilisant la méthode **setLayoutParams**.

Il vous faudra faire de même pour votre **ProgressBar** i.e. :

- définir un **RelativeLayout.LayoutParams** de largeur et hauteur WRAP_CONTENT,
- utiliser les constantes prédéfines de la classe **RelativeLayout** pour spécifier à l'aide de la méthode **addRule** que votre barre de progression doit être centrée dans son élément parent (*addRule(RelativeLayout. ?*)).
- appliquer cet ensemble de paramètres avec **setLayoutParams**.
- ajouter votre **ProgressBar** au **RelativeLayout** précédemment créé (méthode **addView**).

Vous devez suivre exactement les mêmes étapes pour créer la **TextView** qui contiendra le texte affichant le pourcentage de progression.

Il ne vous reste plus qu'à utiliser la méthode **addView** pour rajouter à votre Layout principal, le **RelativeLayout** contenant votre barre de progression et votre zone de texte. N'oubliez pas d'utiliser la méthode **publishProgress** et de compléter la méthode **OnProgressUpdate** afin de gêrer la mise à jour de l'affichage du texte correspondant au pourcentage de progression.

PARTIE 3

Dans ce qui précède, vous avez gêré l'ouverture de votre base de données via un processus détaché. De façon plus générale, il est recommandé que toute action concernant une base de données soit effectuée par un processus détaché. Ainsi l'affichage du contenu de la base doit lui aussi être traité de cette façon.

Définissez à l'intérieur de la classe principale de votre activité, la classe **AsyncReadDB** dont les paramètres seront de type MyDB*Helper, Void et Cursor*. Le deuxième paramètre est de type *Void* car pour la lecture nous ne gêrerons pas l'affichage du pourcentage de progression, il sera donc inutile de redéfinir la méthode **OnProgressUpdate**. Seule la barre circulaire s'affichera. La méthode **doInBackground** renverra un curseur sur le résultat de la requête d'extraction des données de la base et ce curseur sera traité dans la méthode **onPostExecute** pour afficher les résultats dans votre application.

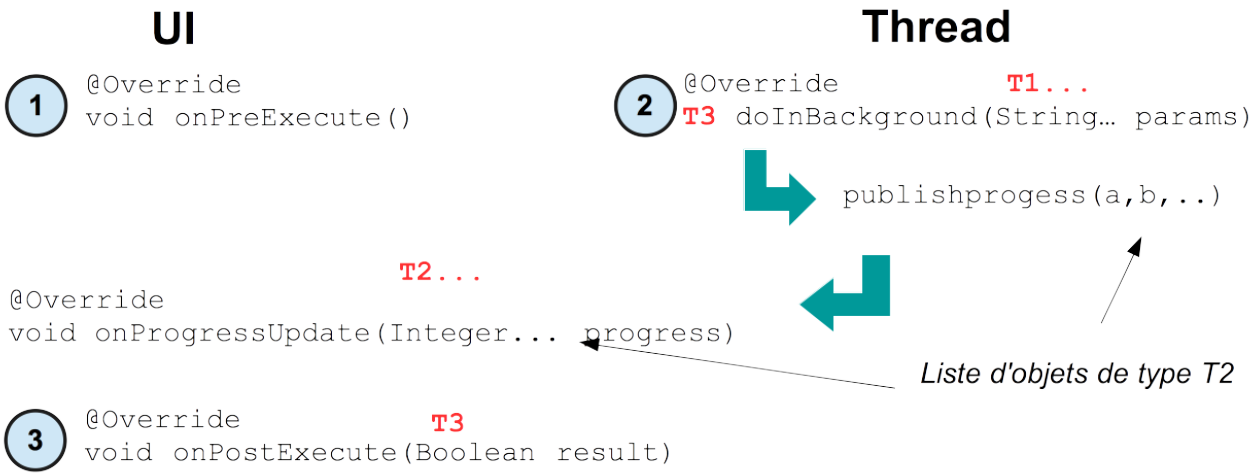
Procédez de même pour la sauvegarde des données dans la base. Attention dans ce cas, la méthode

execute() devra prendre en compte deux paramètres : l'objet SQLiteDataBase et la valeur à insérer dans la base. Il est possible de passer une liste de paramètres à la méthode **execute()** mais il faudra alors spécifier que le premier paramètre de votre classe **AsyncWriteDB** est de type **object** et gêrer à l'intérieur de votre méthode doInBackground les différents paramètres reçus selon leur type.

```
class MyAsyncTask extends AsyncTask<T1, T2, T3>{...}
```

T1 T2 T3

AsyncTask<*String*, *Integer*, *Boolean*>




```
new MyAsyncTask().execute(Objet de type T1);
```