Développement mobile - TP3

September 24, 2018

Victor COLEAU
victor.coleau@etu.univ-tours.fr
Thomas COUCHOUD
thomas.couchoud@etu.univ-tours.fr

Chapter 1

Questions

Le but de ce TP est de mettre en place un service. L'interface graphique se connectera ensuite à ce dernier pour récupérer les données produites et les afficher dans la vue.

Un service est une sorte de tâche de fond à laquelle différents intents peuvent se connecter pour récupérer des informations.

Afin de mettre cela en place il faut définir une classe qui assurera les fonctionnalités du service voulu. Pour faire ceci, elle devra étendre la classe android Service. Cette dernière implémente les fonctions de création/destruction, run et bind/unbind. Lors du run nous créons nos données (l'heure) puis notifions les différents listeners (la vue). Cette dernière s'enregistrera comme listener afin d'être notifiée lors de l'arrivée de nouvelles données et de les afficher.

Des problèmes ont été rencontrés lors de la création du service. En effet, il faut utiliser le contexte de l'activité principale et pas un autre. De plus, lors de l'appel de unbind, cela ne déclenche pas automatiquement la méthode onServiceDisconnected, il faut donc l'appeler nous-même.

1.1 Explication de code

1.1.1 MainActivity

La première étape fut de modifier la méthode onCreate afin d'y récupérer les 4 boutons ainsi que le champ de texte présents dans l'interface graphique.

Ensuite, nous avons déclaré un nouveau listener. Un listener est un pattern design permettant de notifier d'autres classes que certains événements se sont produit dans la classe déclarante. Dans notre cas, l'événement correspond à la mise à jour de l'heure lue. Cependant, il est à noter qu'il est nécessaire de faire cette modification au sein du thread UI (thread principal). Etant donné que l'événement provient d'un processus différent, il est donc impératif de passer par la méthode runOnUiThread.

Ensuite, nous déclarons le ServiceConnection qui gère le binding et le unbinding au service. La méthode onServiceConnected permet de récupérer l'instance du service et d'y ajouter le listener précédement déclaré. La méthode onServiceDisconnected supprime le listener. Il est à noter que cette dernière méthode n'est appelée que dans le cas de déconnections inattendues.

Par la suite, nous déclarons le listener de chaque bouton :

- Start : on démarre le service avec le contexte de la vue de l'application. L'Intent passé en paramètre a pour contexte notre MainActivity et pour cible la classe du service (MyService).
- Stop : on vérifie si le service est bindé, et si tel est le cas, on l'unbind et on appelle la méthode onServiceDisconnected à la main. On met fin à l'exécution du service.
- Connect : on appelle bindService avec les paramètres attendus. Cela permet de recevoir les informations du service dans notre activité.
- Disconnect : on vérifie si le service est bindé, et si tel est le cas, on l'unbind et on appelle la méthode onServiceDisconnected à la main.

1.1.2 IBackgroundService

Cette interface permet de produire les événements. Elle ne propose que deux méthodes : ajouter et supprimer un listener.

1.1.3 IBackgroundServiceListener

Cette interface consomme les événements (déclarée sur le tas par MainActivity). Elle ne propose qu'une méthode dataChanged appelée par IBackgroundService lorsqu'une nouvelle donnée est disponible.

1.1.4 MyService

Cette classe contient l'implémentation de Service et IBackgroundService.

Lors de la création de ce service par Android via onCreate, on créé un nouveau Timer ainsi qu'un BackgroundServiceBinder.

Une fois créé, ce service peut être démarré depuis MainActivity grâce à la méthode startService. Cela appelle la méthode onStartCommand. Cette dernière contient la logique du service. Nous y lançons une tache à interval régulier à l'aide de notre objet Timer. Cette tache récupère la date actuelle, la formate et appelle fireDataChanged.

Une fois lancé, il est possible de se connecter au service. Pour cela, la méthode onBind renverra l'instance créée de BackgroundServiceBinder. De plus, les méthodes onRebind, onUnbind sont laissées selon leur implémentation par défaut.

Une fois déconnecté, le service peut être arrêté depuis MainActivity grâce à la méthode stopService. Cela appelle la méthode onDestroy qui vide la liste des listener et arrête le Timer.

De plus, le service contient les méthodes implémentées de IBackgroundService permettant la gestion de la liste des listener.

Enfin, la méthode fireDataChanged appelle les différents listener pour les notifier de la présence d'un nouveau temps à afficher.

1.1.5 BackgroundServiceBinder

Cet objet créé par le service est passé en paramètre de la méthode onServiceConnected et permet de récupérer l'instance de notre service afin d'y effectuer les actions nécessaires.