Cédric Cognard
Paul Van-Elsue

Système mobiles

Compte rendu de TP n°3 :

Utiliser les services

Introduction:

Le but de ce TP est d'aborder la notion de services, ainsi que de voir et d'utiliser les Services Android. Nous allons élaborer un service qui récupérera toutes les secondes l'heure courante au format : heure-minute-seconde et nous l'utiliserons pour l'afficher sur l'interface utilisateur.

Les Services

Les services peuvent être classés en 2 catégories, il y a d'abord les services locaux qui s'exécutent dans l'application elle-même (et sont accessibles par elle uniquement) puis les services distants (Remote) qui s'exécutent dans un autre thread.

Pour définir un service, il faut surcharger plusieurs méthodes:

- onCreate est exécuté lors de la création du service
- onStart ou onStartCommand (pour les SDK plus récents) quand la tâche de fond démarre
- onBind lorsqu'un client se connecte au service
- onUnbind lorsqu'il se déconnecte
- onDestroy lorsqu'on veut arrêter le service et libérer ses ressources.

Pour créer un nouveau service, il faut étendre la classe Service par héritage, puis réimplémenter les méthodes citées ci-dessus

La variable **START_NOT_STICKY** indique que le service ne doit tourner en tâche de fond que quand il reçoit des données d'une activity de l'application et doit doit s'arrêter une fois qu'il a fini ce qu'il doit faire. **START_STICKY** indique qu'il faut explicitement indiquer au service de se démarrer ou se stopper.

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    Log.d(getClass().getName(),  msg: "onStartCommand");

    timer.scheduleAtFixedRate(() → {
            DateFormat dateFormat = new SimpleDateFormat( pattern: "HH:mm:ss");
            Date date = new Date();
            Log.d(getClass().getName(), dateFormat.format(date));
            fireDataChanged(dateFormat.format(date));
    },  delay: 0,  period: 1000);

    return START_NOT_STICKY;
}
```

Pour pouvoir utiliser notre service, il faut l'ajouter dans le manifeste de notre application en utilisant la balise *<application/>*.

Pour utiliser les boutons de notre IHM il nous faut des listeners qui vont démarrer ou arrêter les services:

```
View.OnClickListener listener_start = new
Button.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(MainActivity.this,
BackgroundService.class);
        startService(intent);
    }
};

View.OnClickListener listener_stop = new
Button.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(MainActivity.this,
BackgroundService.class);
        stopService(intent);
    }
};
```

Pour abstraire la complexité de l'interface de service, nous allons mettre en place divers classes et interfaces. Pour commencer, l'interface **IBackgroundServiceListener** qui va permettre de notifier tout élément qui s'est "abonné" au service qui implémentera cette interface. Puis **IBackgroundService** qui permettra d'ajouter et retirer des listeners.

Dans notre classe **BackgroundService**, on implémente cette dernière puis on ajoute dans les attributs une *ArrayList* qui contiendra nos listeners et les méthodes correspondante des deux interfaces.

Enfin, on ajoute le classe **BackgroundServiceBinder** qui va nous permettre de nous connecter au service avec lequel on le construit.

Dans notre activité principale, il faut déclarer le listener à notre service et surcharger la méthode dataChanged en indiquant quoi faire lorsqu'on reçoit une notification de notre listener.

```
listener = new IBackgroundServiceListener() {
   public void dataChanged(final Object data) {
       MainActivity.this.runOnUiThread(new Runnable() {
           public void run() {
               TextView welcome_text =
findViewById(R.id.welcom_text);
               welcome_text.setText(data.toString());
           }
       });
   }
};
```

Enfin, il faut définir un objet connection qui va indiquer quoi faire lorsqu'on se connecte et se déconnecte du service. Le flag **BIND_AUTO_CREATE** permet de démarrer le service s'il ne l'est pas déjà lorsqu'on s'y connecte.

Au final nous avons cette application qui permet de se connecter ou de se déconnecter de notre service qui affiche l'heure courante.