TP 3 Android
TI et IQL
S5 - 2021

Broadcast Receiver

Objectif :

L'objectif de cette partie est d'implémenter deux Broadcast Receivers (BR) utilisant la méthode dynamique, statique et en captant un événement système et un événement personnalisé.

Un BR statique utilise le Manifest pour s'enregistrer à des événements. Il reste accessible même si l'activité disparait ou est morte. Certains événements ne peuvent pas s'enregistrer d'une manière statique.

Un BR dynamique s'enregistre aux événements via le code au besoin. Il est lié au cycle de vie de l'activité. Il faut se désinscrire explicitement.

Enoncé :

Créer un nouveau projet avec l'activité par défaut Hello world.

1. Nous allons maintenant créer un Broadcast Receiver **statique** qui répondra à un événement personnalisé :

- Aller dans le volet du projet bouton droit -> nouveau -> other -> Broadcast receiver
- Il suffit alors d'implémenter le comportement du receiver à la réception de l'événement. Autrement dit, implémenter la méthode **onReceive( )**
- Nous choisirons un comportement basique : Log.i(**"BR_TAG"**,**" Event Received "**);
- Maintenant il faudra enregistrer le broadcast receiver dans le Manifest

  Identifier sa balise dans le manifest et ajoutez-y un intent-filter personnalisé comme suit :

```
<intent-filter>
    <action android:name="FAKE_EVENT_INFO" />
</intent-filter>
```

- Enfin il faut générer l'événement « FAKE_EVENT_INFO », pour cela ajoutez un bouton dans votre « main activity » qui permettra d'envoyer cet événement à chaque clique.

  L'événement est envoyé dans un Intent via la méthode sendbroadcast(Intent i)

  Exécutez le code et surveiller le log. Le message « Event Received » doit s'afficher.

2. Maintenant nous allons définir un autre Broadcast Receiver qui interceptera un événement système et qui sera implémenté d'une manière dynamique.

   Il s'agira de capter l'événement Batterie Faible et afficher un message dans le TextView de l'interface de l'application.

   Cette fois-ci nous allons définir une classe interne dans notre « Main_activity » (il est aussi possible de le faire dans une classe séparée).

   a. Déclarer, dans le même fichier de la « Main_activity », une classe MyBroadcastBatteryLow qui héritera de la classe BroadcastReceiver
   b. Implémenter la méthode onReceive( ) : Ecrire dans le TextView « Evenement Batterie faible reçu » (par exemple)
   c. Il faut maintenant instancier et enregistrer votre broadcast receiver dans la main activity :
      - Instancier une variable de type MyBroadcastBatteryLow
      - Créer un Intent filter : IntentFilter **filter** = **new** IntentFilter();
      - Ajoutez l'action Intent.ACTION_BATTERY_LOW à cet intent filter
      - Dans onResume(), enregistrer votre broadcast receiver : **this**.registerReceiver()
      - Dans onPause() : désinscrire votre broadcast receiver

   Exécuter votre code sur l'émulateur. Il est plus facile de simuler le niveau de batterie sur un émulateur sinon utilisez un autre événement tel que : Intent.ACTION_POWER_CONNECTED avec un chargeur. Sur l'émulateur baisser le niveau de batterie graduellement et doucement jusqu'à la génération de l'événement batterie faible par le système.

3. Nous allons maintenant créer un 3ème broadcast receiver plus évolué dont le but est d'intercepter les appels entrants et d'afficher par exemple le numéro de tel appelant.

   Ce Broadcast receiver **doit** être enregistré de manière dynamique.

   - Comme en 2 : créer une classe interne qui hérite de BroadcastReceiver
   - L'action à ajouter au intent-filter est **android.intent.action.PHONE_STATE**
   - Dans le onReceive( ) :

```
String callState =
intent.getExtras().getString(TelephonyManager.EXTRA_STATE);

String number = intent.getExtras().getString(xxx); //TO DO: xxx
// utilisez TelephonyManager pour obtenir le numéro appelant en tant qu'extra

if(callState.equals(TelephonyManager.EXTRA_STATE_RINGING)){
   //appel entrant
     Date callStartTime = new Date();

     //TO DO : afficher num d'appel et date/heure d'appel
}
```

- Enregistrer votre broadcast receiver et désinscrivez-le là où il faut.
- Pour accéder à l'état d'un appel, il faut demander la permission : **android.permission.READ_PHONE_STATE**
- Finalement comme il s'agit d'un appel, il faut en plus demander une permission au moment d'exécution. Ajoutez le code là où il faut :

```
if (ActivityCompat.checkSelfPermission(getApplicationContext(),
        Manifest.permission.READ_PHONE_STATE)
        != PackageManager.PERMISSION_GRANTED) {

    // Should we show an explanation?
    if (ActivityCompat.shouldShowRequestPermissionRationale(this,
            Manifest.permission.READ_PHONE_STATE)) {
    } else {
        ActivityCompat.requestPermissions(this,
                new
String[]{Manifest.permission.READ_PHONE_STATE},
                MY_PERMISSIONS_REQUEST_READ_PHONE_STATE);
    }
}

@Override
public void onRequestPermissionsResult(int requestCode,
                                       String permissions[], int[]
grantResults) {
    switch (requestCode) {
        case MY_PERMISSIONS_REQUEST_READ_PHONE_STATE: {
            // If request is cancelled, the result arrays are empty.
            if (grantResults.length > 0
                    && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {

            } else {
            }
            return;
        }    }
}
```