SCIENCES SORBONNE UNIVERSITÉ

Broadcast Receiver

F. Kordon - Sorbonne Université - CC2018

Renault@lrde.epita.fr

Definition

- **Component that responds to events**

- **Most of the events are triggered by the system**
  - Low battery, screen locked, picture taken...

- **... BUT some may be triggered by applications**
  - To inform that something is now available

- **BroadcastReceivers:**
  - Don't have a GUI
  - But can launch some notifications

Building a BroadcastReceiver

Modify the AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.admin.mybroadcastapplication" >

 <uses-permission
      android:name="android.permission.READ_SMS" />

<uses-permission
      android:name="android.permission.RECEIVE_SMS" />

<receiver android:name=".MyReceiver">

   <intent-filter>
     <action android:name="android.permission.READ_SMS"/>
     <action android:name="android.permission.RECEIVE_SMS"/>
   </intent-filter>

</receiver>
```

Registering & Understanding Lifecycle

Register the BroadcastReceiver

- Specify programmatically
  - the receiver
  - the IntentFilter

```
registerReceiver(new MyReceiver(), new IntentFilter
                 ("android.provider.Telephony.SMS_RECEIVED"));
```

Lifecycle

- Simplest lifecycle possible
- Only one action onReceive
  - In this method, computing must be short and without asynchronous processing

Implementation of a Broadcast Receiver


```
public class MyReceiver extends BroadcastReceiver {
   @Override
   public void onReceive(Context context, Intent intent) {
      NotificationCompat.Builder mBuilder =
            new NotificationCompat.Builder(context);
       mBuilder.setContentTitle("Alert, Critical Message!");
       mBuilder.setContentText("This is Notification Detail!");

       // Create
       NotificationManager notificationmanager =
                (NotificationManager) context
                .getSystemService(Context.NOTIFICATION_SERVICE);

       // Build Notification with Notification Manager
       notificationmanager.notify(0, mBuilder.build());
   }
}
```

No direct access to the UI

Broadcasting its own Events

Defining a new event

```
<receiver android:name=".MyReceiver">
  <intent-filter>
  <action android:name="com.example.mybdcastapp.CUSTOM_INTENT"/>
</intent-filter>
```

Register to an event

```
registerReceiver(new MyReceiver(), new
        IntentFilter(« com.example.mybdcastapp.CUSTOM_INTENT"));
```

Broadcasting

Normal Broadcast

(sendBroadcast)

- Asynchronous messages
- No ordering between receivers
- Efficient

```
Intent intent = new Intent();
intent.setAction("com.example.mybdcastapp.CUSTOM_INTENT");
sendBroadcast(intent);
```

Ordered Broadcast

(sendOrderedBroadcast)

- One receiver at a time
- Priority can be fixed through android:priority
- Receiver can stop diffusion

```
Intent intent = new Intent();
intent.setAction("com.example.mybdcastapp.CUSTOM_INTENT");
sendOrderedBroadcast(intent);
```

A word on Security

- **BroadcastReceiver use Context that**
  - allows access to application-specific ressources & classes
  - etc.

- **Ensure that you only work on Intents or string that are in your namespace**

- **Some applications do not respect IntentFilter**
  - This can be fight using android:exported=false

- **All applications can target a broadcast receiver**
  - Fix that by using Android permissions

Summary

- **BroadcastReceiver offer a way to trace all the system events**

- **Two kind of broadcasting**
  - ordered
  - asynchronous

- **LocalBroadcastManager**
  - Do not use shared memory (inter-processes)
  - High security: we can share sensitive information
  - Other application cannot target these managers