

Broadcast receivers, content providers, services, async tasks

Mateusz Nowotyński Marcin Moskal Kamil Osuch

12.12.2017

Do czego to służy?

BroadcastReceiver pozwala nam na odbieranie powiadomień (Systemu bądź innej aplikacji) wewnątrz naszej aplikacji. Takim powiadomieniem może być na przykład informacja o nowej wiadomości SMS bądź rozładowanej baterii.

Żeby zbudować nasz własny BroadcastReceiver musimy wykonać dwie czynności:

- 1 Stworzyć podklasę klasy BroadcastReceiver
- 2 Wspecyfikować receiver w manifeście aplikacji lub bezpośrednio w kodzie

Przykład klasy Broadcast Receiver'a

```
public class MyBroadcastReceiver extends BroadcastReceiver {  
    private static final String TAG = "com.example.woy.toolbar.MyBroadcastReceiver";  
  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        StringBuilder sb = new StringBuilder();  
        sb.append("Action: " + intent.getAction() + "\n");  
        sb.append("URI: " + intent.toUri(Intent.URI_INTENT_SCHEME).toString() + "\n");  
        String log = sb.toString();  
        Log.d(TAG, log);  
        Toast.makeText(context, log, Toast.LENGTH_LONG).show();  
    }  
}
```

Dodanie do manifestu

```
<receiver android:name=".MyBroadcastReceiver" android:exported="true">  
  <intent-filter>  
    <action android:name="android.intent.action.BOOT_COMPLETED"/>  
    <action android:name="android.intent.action.INPUT_METHOD_CHANGED" />  
  </intent-filter>  
</receiver>
```

Uwaga!

Powyższe rozwiązanie nie zadziała w wersji API wyższej niż 25. Nie można wtedy użyć manifestu do zadeklarowania receiver'a dla broadcast'ów nieskierowanych bezpośrednio dla Twojej aplikacji (z wyjątkiem kilku wyszczególnionych).

Rejestracja w kodzie

```
public class MainActivity extends Activity {  
  
    private IntentFilter filter =  
        new IntentFilter( action: "android.provider.Telephony.SMS_RECEIVED");  
  
    private BroadcastReceiver broadcast = new MyBroadcastReceiver();  
  
    @Override  
    public void onResume() {  
        super.onResume();  
        registerReceiver(broadcast, filter);  
    }  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
  
    @Override  
    public void onPause() {  
        unregisterReceiver(broadcast);  
        super.onPause();  
    }  
}
```

Manifest

W przypadku dodania deklaracji receiver'a do manifestu, jest on rejestrowany w momencie zainstalowania aplikacji. Receiver staje się wtedy oddzielnym punktem wejścia aplikacji. Oznacza to, że system może wystartować aplikację i przekazać wysyłany broadcast do receiver'a.

Rejestracja w kodzie

Receiver'y zarejestrowane w kodzie odbierają broadcast'y tak długo jak kontekst w którym zostały zarejestrowane jest aktywny.

Android oferuje trzy sposoby wysyłania broadcast'ów:

- **sendOrderedBroadcast** - wysyła broadcast do jednego receiver'a na raz. Gdy receiver zakończy swoje wykonanie, może propagować rezultat do innego receiver'a, lub też całkowicie przerwać broadcast. Kolejność broadcast'ów może być określana za pomocą atrybutu **android-priority**

Android oferuje trzy sposoby wysyłania broadcastów:

- **sendOrderedBroadcast** - wysyła broadcast do jednego receiver'a na raz. Gdy receiver zakończy swoje wykonanie, może propagować rezultat do innego receiver'a, lub też całkowicie przerwać broadcast. Kolejność broadcastów może być określana za pomocą atrybutu **android-priority**
- **sendBroadcast** - tzw. normalny broadcast, wysyła broadcast do wszystkich receiver'ów w nieokreślonej kolejności. Jest szybszy, ale nie można kontrolować przepływu broadcast'u między receiver'ami, czy go zatrzymać.

Android oferuje trzy sposoby wysyłania broadcast'ów:

- **sendOrderedBroadcast** - wysyła broadcast do jednego receiver'a na raz. Gdy receiver zakończy swoje wykonanie, może propagować rezultat do innego receiver'a, lub też całkowicie przerwać broadcast. Kolejność broadcast'ów może być określana za pomocą atrybutu **android-priority**
- **sendBroadcast** - tzw. normalny broadcast, wysyła broadcast do wszystkich receiver'ów w nieokreślonej kolejności. Jest szybszy, ale nie można kontrolować przepływu broadcast'u między receiver'ami, czy go zatrzymać.
- **LocalBroadcastManager.sendBroadcast** - wysyła broadcast do receiver'ów, które są w tej samej aplikacji, co strona wysyłająca.

Przykład wysyłania broadcast'ów

```
Intent intent = new Intent();  
intent.setAction("com.example.broadcast.MY_NOTIFICATION");  
intent.putExtra( name: "data", value: "Example data");  
sendBroadcast(intent);
```

Uwaga!

Pomimo tego, że Intent jest używany zarówno do wysyłania broadcast'ów, jak i startowania activity (przy pomocy **startActivity**), akcje te nie są ze sobą powiązane. Broadcast receiver'y nie odbiorą akcji używanej do wystartowania activity.

Wysyłanie

sendBroadcast(intent, permission) - wysłany w ten sposób broadcast może zostać odebrany tylko przez receiver posiadający uprawnienie **permission**

Odbieranie

registerBroadcaster(receiver, intent-filter, permission, handler) - Receiver odbierze broadcast'y tylko od wysyłających, którzy posiadają uprawnienie **permission**