

Android - Broadcast receivers

A broadcast is a message that any app can receive. The system delivers various broadcasts for system events, such as when the system boots up or the device starts charging or when a phone call or sms are incoming. But also we can deliver a broadcast to ourselves or to other apps by passing an `Intent` to `sendBroadcast()`

Note: Although Broadcast Receivers can be used to send local messages within the application (as shown next) it is generally used to send message across applications, if you only need to send a local message within the app, consider using the `LocalBroadcastManager`

The `BroadcastReceiver` class is an abstract class designed to receive broadcasts that sent either by the system upon registering to them with intent filters or by the context `sendBroadcast(intent)` function.

When extending the `BroadcastReceiver` class you must implement the `onReceive` function. Upon receiving the broadcast you have registered to, the system automatically creates an instance of your class and calls that function passing it the triggering intent as a parameter together with the application's context. The `onReceive` function can do any kind of short term actions (the broadcast receiver will only be alive for 5 seconds).

It is important to understand that the broadcast receiver work on the main thread.

In order to receive a broadcast you must first register to it, this can be done either in the manifest file or dynamically in your java file, if you register dynamically you should remember to unregister it when you no longer need it (failing to do so will slow down the system).

Steps to create broadcastreceiver

In general you must declare the action you want inside an intent filter and register your receiver to it, then you can either send the action yourself or more likely you will get the event from the

system (in case the action is incoming sms or phone call or all the other system events), then an instance of your BroadcastReceiver will be created and onReceive will be called.

The steps to receive broadcast are as follows:

1. Create a new class that extends "BroadcastReceiver"
2. Override the "onReceive" function.
3. Inside the "onReceive" you can check the triggering action using the `intent.getAction()` and perform the necessary operations.
4. register your receiver:
 - A. In the manifest file: Go to the manifest and add a "<receiver>" tag with the name of the class (<receiver "android:name= ".ReceiverName">). inside the "<receiver>" tag add an "<intent-filter>" with the actions that will activate this receiver.
 - B. Dynamically register the receiver through the context **registerReceiver** function. The function accept two arguments: the first is an instance of the receiver while the second is the intent filter containing the action that will trigger the receiver. In case you register the receiver in your onCreate or onResume methods don't forget to unregister it in the onDestroy and onPause function to avoid overloading the system with unnecessary calls.

Note: some system events can't be registered in the android manifest file and must be registered dynamically through the **registerReceiver** function, for example the screen on/off event, battery changed, power connection and more frequent event (in order for the system to save calls since all are very frequent actions).

Local Broadcast Manager

Use this class if you want to send messages within your app. You can send private data, we can be sure that no one but us will send the message and it is more efficient than the `sendBroadcast()` function which transmitted to all of the system.

The concept is the same, where you want to receive the message get the `LocalBroadcastManager` instance and register your receiver to specific intent filter with your own custom action like this :

```
LocalBroadcastManager.getInstance(this).registerReceiver(mReceiver,filter);
```

And don't forget to unregister in the `onDestroy()` :

```
LocalBroadcastManager.getInstance(this).unregisterReceiver(mReceiver);
```

In the place where you want to send the broadcast create an intent give him the same action you defined in the intent-filter, add the extras you want and send you broadcast with

```
LocalBroadcastManager.getInstance(this).sendBroadcast(intent);
```