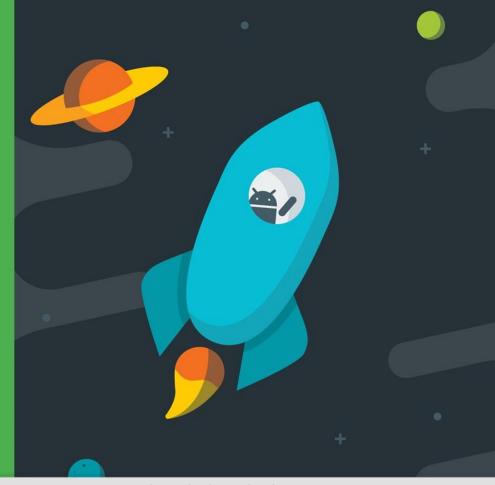
Android Developer Fundamentals V2

Background Tasks

Lesson 7



7.3 Broadcasts

Contents

- Broadcasts
- Send a custom broadcasts
- Broadcast receivers
- Implementing broadcast receivers
- Restricting the broadcasts
- Best practices

Broadcasts

Broadcasts



Broadcasts

Broadcasts

Broadcasts are messages sent by Android system and other Android apps, when an event of interest occurs.

Broadcasts are wrapped in an Intent object. This Intent object's contains the event details such as, android.intent.action.HEADSET_PLUG, sent when a wired headset is plugged or unplugged.

Types of broadcasts

Types of broadcast:

- System broadcast.
- Custom broadcast.

Android Developer Fundamentals V2

System broadcasts

System broadcast are the messages sent by the Android system, when a system event occurs, that might affect your app.

Few examples:

- An Intent with action, <u>ACTION BOOT COMPLETED</u> is broadcasted when the device boots.
- An Intent with action, <u>ACTION POWER CONNECTED</u> is broadcasted when the device is connected to the external power.

Android Developer Fundamentals V2

Custom broadcasts

Custom broadcasts are broadcasts that your app sends out, similar to the Android system.

For example, when you want to let other app(s) know that some data has been downloaded by your app, and its available for their use.

Send a custom broadcasts

Android Developer Fundamentals V2

Send a custom broadcast

Android provides three ways for sending a broadcast:

- Ordered broadcast.
- Normal broadcast.
- Local broadcast.

Ordered Broadcast

- Ordered broadcast is delivered to one receiver at a time
- To send a ordered broadcast, use the sendOrderedBroadcast() method
- Receivers can propagate result to the next receiver or even abort the broadcast.
- Control the broadcast order with android:priority attribute in the manifest file
- Receivers with same priority run in arbitrary order.



License.

Normal Broadcast

- Delivered to all the registered receivers at the same time, in an undefined order.
- Most efficient way to send a broadcast.
- Receivers can't propagate the results among themselves, and they can't abort the broadcast.
- The sendBroadcast () method is used to send a normal broadcast.

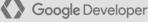
Local Broadcast

- Sends broadcasts to receivers within your app.
- No security issues since no interprocess communication.
- To send a local broadcast:
 - To get an instance of LocalBroadcastManager.
 - Call sendBroadcast () on the instance.

LocalBroadcastManager.getInstance(this)

.sendBroadcast(customBroadcastIntent);

Broadcast Receivers



Android Developer Fundamentals V2

What is a broadcast receiver?

- Broadcast receivers are app components.
- They register for various system broadcast and or custom broadcast.
- They are notified (via an Intent):
 - By the system, when an system event occurs that your app is registered for.
 - By another app, including your own if your app is registered for that custom event.

Register your broadcast receiver

Android Developer Fundamentals V2

Broadcast receivers can be registered in two ways:

- Static receivers
 - Registered in your AndroidManifest.xml, also called as Manifest-declared receivers.
- Dynamic receivers
 - Registered using app or activities' context in your Java files, also called as Context-registered receivers.

Receiving a system broadcast

- Starting from Android 8.0 (API level 26), static receivers can't receive most of the system broadcasts.
- Use a dynamic receiver to register for these broadcasts.
- If you register for the system broadcasts in the manifest, the Android system won't deliver them to your app.
- A few broadcasts, are excepted from this restriction. See the complete list of <u>implicit broadcast exceptions</u>.

Implementing Broadcast Receivers



This work is licensed under a **Creative**

Commons Attribution 4.0 International

License.

To create a broadcast receiver

- Subclass the BroadcastReceiver class and override its on Receive () method.
- Register the broadcast receiver and specify the intentfilters:

Broadcasts

- Statically, in the Manifest.
- Dynamically, with registerReceiver().

What are Intent-filters

Intent-filters specify the types of intents a broadcast receiver can receive. They filter the incoming intents based on the Intent values like action.

To add an intent-filter:

- To your AndroidManifest.xml file, use <intentfilter> tag.
- To your Java file use the IntentFilter object.

Subclass a broadcast receiver

In Android studio, File > New > Other > BroadcastReceiver

```
public class CustomReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        // This method is called when the BroadcastReceiver
        // is receiving an Intent broadcast.
        throw new UnsupportedOperationException("Not yet implemented");
    }
}
```

This work is licensed under a Creative

Commons Attribution 4.0 International

License.

Implement onReceive()

Example implementation of onReceive() method which handles power connected and disconnected.

```
@Override
public void onReceive(Context context, Intent intent) {
   String intentAction = intent.getAction();
   switch (intentAction){
       case Intent.ACTION POWER CONNECTED:
           break;
       case Intent.ACTION POWER DISCONNECTED:
           break;
```

Register statically in Android manifest

- <receiver> element inside <application> tag.
- <intent-filter> registers receiver for specific intents.

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

Register dynamically

• Register your receiver in onCreate() or onResume().

```
// Register the receiver using the activity context.
this.registerReceiver(mReceiver, filter);
```

Unregister in onDestroy() or onPause().

```
// Unregister the receiver
this.unregisterReceiver(mReceiver);
```

Register a Local broadcast receiver

Android Developer Fundamentals V2

Register local receivers dynamically, because static registration in the manifest is not possible for a local broadcasts.

Register a Local broadcast receiver

To register a receiver for local broadcasts:

- Get an instance of Local Broadcast Manager.
- Call registerReceiver().

```
LocalBroadcastManager.getInstance(this).registerReceiver
(mReceiver,
   new IntentFilter(CustomReceiver.ACTION_CUSTOM_BROADCAST));
```

This work is licensed under a Creative

Commons Attribution 4.0 International

License.

Unregister a Local broadcast receiver

To unregister a local broadcast receiver:

- Get an instance of the Local Broadcast Manager.
- Call LocalBroadcastManager.unregisterReceiver().

```
LocalBroadcastManager.getInstance(this)
```

Android Developer Fundamentals V2

```
.unregisterReceiver (mReceiver);
```

Restricting broadcasts



Android Developer Fundamentals V2

Restricting broadcasts

- Restricting your broadcast is strongly recommended.
- An unrestricted broadcast can pose a security threat.

Android Developer Fundamentals V2

 For example: If your apps' broadcast is not restricted and includes sensitive information, an app that contains malware could register and receive your data.

Ways to restrict a broadcast

- If possible, use a <u>LocalBroadcastManager</u>, which keeps the data inside your app, avoiding security leaks.
- Use the <u>setPackage()</u> method and pass in the package name. Your broadcast is restricted to apps that match the specified package name.
- Access permissions can be enforced by sender or receiver.

Enforce permissions by sender

To enforce a permission when sending a broadcast:

- Supply a non-null permission argument to sendBroadcast().
- Only receivers that request this permission using the <uses-permission> tag in their
 AndroidManifest.xml file can receive the broadcast.

Enforce permissions by receiver

To enforce a permission when receiving a broadcast:

- If you register your receiver dynamically, supply a non-null permission to registerReceiver().
- If you register your receiver statically, use the android:permission attribute inside the <receiver> tag in your AndroidManifest.xml.

Best practices



This work is licensed under a **Creative**

License.

Commons Attribution 4.0 International

Best practices

- Make sure namespace for intent is unique and you own it.
- Restrict broadcast receivers.
- Other apps can respond to broadcast your app sends —use permissions to control this.
- Prefer dynamic receivers over static receivers.
- Never perform a long running operation inside your broadcast receiver.



Learn more

- BroadcastReceiver Reference
- Intents and Intent Filters Guide
- LocalBroadcastManager Reference
- Broadcasts overview

What's Next?

- Concept Chapter: 7.3 Broadcasts
- Practical: 7.3 Broadcast Receivers

Android Developer Fundamentals V2

This work is licensed under a Creative

License.

END