7.3 Broadcasts

Contents

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

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.

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, `ACTION_BOOT_COMPLETED` is broadcasted when the device boots.
- An `Intent` with action, `ACTION_POWER_CONNECTED` is broadcasted when the device is connected to the external power.

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

Send a custom broadcast

Android provides three ways for sending a broadcast:

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

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.

Normal Broadcast Example

```
public void sendBroadcast() {
   Intent intent = new Intent();
   intent.setAction("com.example.myproject.ACTION_SHOW_TOAST");
   // Set the optional additional information in extra field.
   intent.putExtra("data","This is a normal broadcast");
   sendBroadcast(intent);
}
```

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.

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

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

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 implicit broadcast exceptions.

Implementing Broadcast Receivers

To create a broadcast receiver

- Subclass the `BroadcastReceiver` class and override its `onReceive()` method.

- Register the broadcast receiver and specify the intent-filters:
  - 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 `<intent-filter>` 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");
    }
}
```

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

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 `LocalBroadcastManager`.
- Call `registerReceiver()`.

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

Unregister a Local broadcast receiver

To unregister a local broadcast receiver:

- Get an instance of the `LocalBroadcastManager`.
- Call `LocalBroadcastManager.unregisterReceiver()`.

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

Restricting broadcasts

Restricting broadcasts

- Restricting your broadcast is strongly recommended.
- An unrestricted broadcast can pose a security threat.
- 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 `LocalBroadcastManager`, which keeps the data inside your app, avoiding security leaks.
- Use the `setPackage()` 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

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