Android BroadcastReceiver Example Tutorial

by

Today we'll discuss and implement Android BroadcastReceiver that is a very important component of Android Framework.

Android BroadcastReceiver

Android BroadcastReceiver is a dormant component of android that listens to system-wide broadcast events or intents.

When any of these events occur it brings the application into action by either creating a status bar notification or performing a task.

Unlike activities, android `BroadcastReceiver` doesn't contain any user interface. Broadcast receiver is generally implemented to delegate the tasks to services depending on the type of intent data that's received.

Following are some of the important system wide generated intents.

1. **android.intent.action.BATTERY_LOW** : Indicates low battery condition on the device.
2. **android.intent.action.BOOT_COMPLETED** : This is broadcast once, after the system has finished booting
3. **android.intent.action.CALL** : To perform a call to someone specified by the data
4. **android.intent.action.DATE_CHANGED** : The date has changed
5. **android.intent.action.REBOOT** : Have the device reboot
6. **android.net.conn.CONNECTIVITY_CHANGE** : The mobile network or wifi connection is changed(or reset)

Broadcast Receiver in Android

To set up a Broadcast Receiver in android application we need to do the following two things.

1. Creating a BroadcastReceiver
2. Registering a BroadcastReceiver

Creating a BroadcastReceiver

Let's quickly implement a custom BroadcastReceiver as shown below.

```
public class MyReceiver extends BroadcastReceiver {
    public MyReceiver() {
    }

    @Override
    public void onReceive(Context context, Intent intent) {

        Toast.makeText(context, "Action: " + intent.getAction(), Toast.LENGTH_
    }
}
```

BroadcastReceiver is an abstract class with the `onReceiver()` method being abstract.

The `onReceiver()` method is first called on the registered Broadcast Receivers when any event occurs.

The intent object is passed with all the additional data. A Context object is also available and is used to start an activity or service using `context.startActivity(myIntent);` or `context.startService(myService);` respectively.

Registering the BroadcastReceiver in android app

A BroadcastReceiver can be registered in two ways.

1. By defining it in the `AndroidManifest.xml` file as shown below.

```
<receiver android:name=".ConnectionReceiver" >
            <intent-filter>
                <action android:name="android.net.conn.CONNECTIVITY_CHAN
            </intent-filter>
</receiver>
```

Using intent filters we tell the system any intent that matches our subelements should get delivered to that specific broadcast receiver.

2. By defining it programmatically

   Following snippet shows a sample example to register broadcast receiver programmatically.

```
IntentFilter filter = new IntentFilter();
intentFilter.addAction(getPackageName() + "android.net.conn.CONNECTIVITY_

MyReceiver myReceiver = new MyReceiver();
registerReceiver(myReceiver, filter);
```

To unregister a broadcast receiver in `onStop()` or `onPause()` of the activity the following snippet can be used.

```
@Override
protected void onPause() {
    unregisterReceiver(myReceiver);
    super.onPause();
}
```

Sending Broadcast intents from the Activity

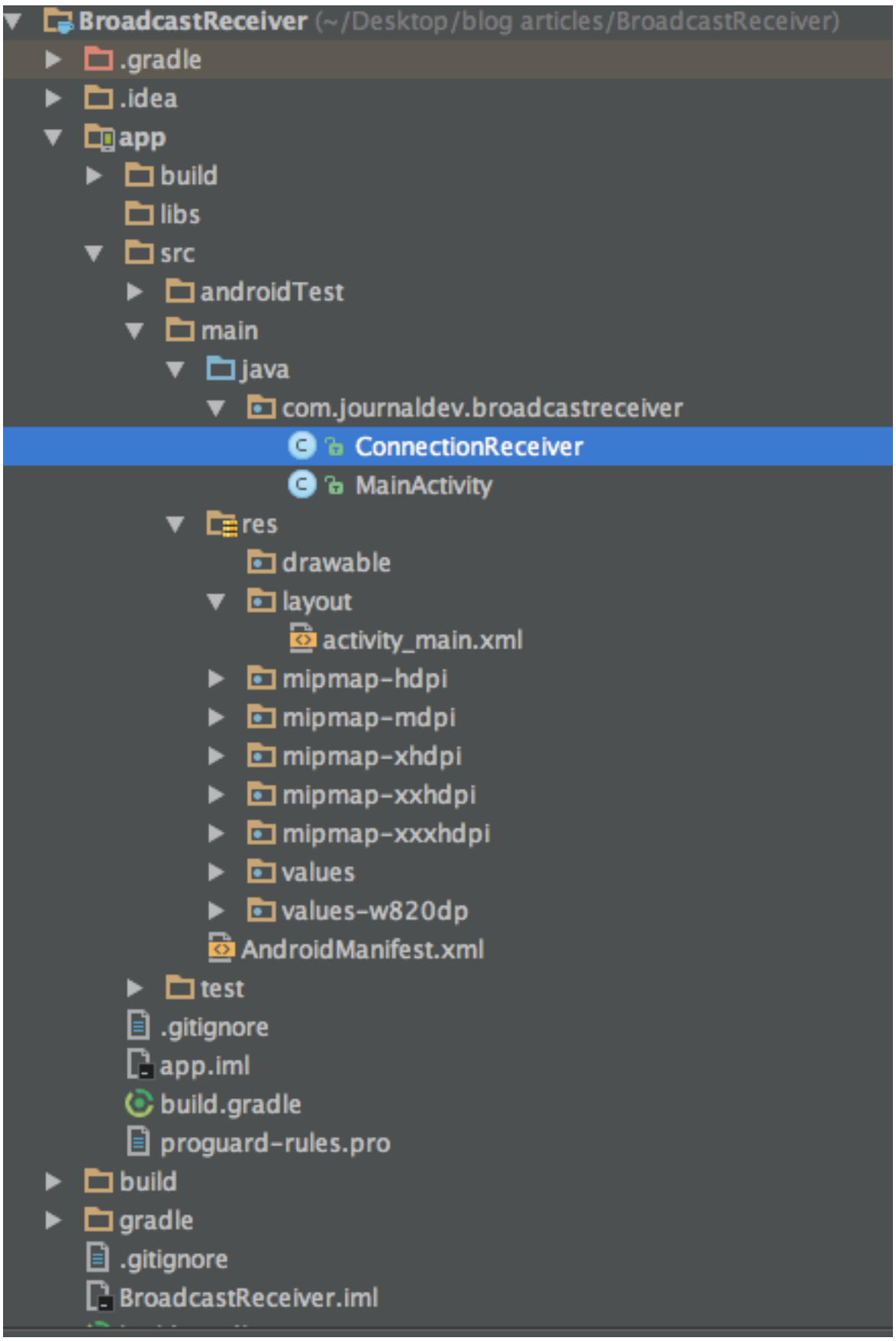
The following snippet is used to send an intent to all the related BroadcastReceivers.

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

Don’t forget to add the above action in the intent filter tag of the manifest or programmatically.

Let’s develop an application that listens to network change events and also to a custom intent and handles the data accordingly.

BroadcastReceiver in Android Project Structure



Android BroadcastReceiver Code

The `activity_main.xml` consists of a button at the centre that sends a broadcast intent.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.journaldev.broadcastreceiver.MainActivity">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button"
        android:text="Send Broadcast"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />
</RelativeLayout>
```

The `MainActivity.java` is given below.

```
package com.journaldev.broadcastreceiver;

import android.content.Intent;
import android.content.IntentFilter;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;

import butterknife.ButterKnife;
import butterknife.InjectView;
import butterknife.OnClick;

public class MainActivity extends AppCompatActivity {
    ConnectionReceiver receiver;
    IntentFilter intentFilter;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    ButterKnife.inject(this);

    receiver = new ConnectionReceiver();
    intentFilter = new IntentFilter("com.journaldev.broadcastreceiver.SOME

}

@Override
protected void onResume() {
    super.onResume();
    registerReceiver(receiver, intentFilter);

}

@Override
protected void onDestroy() {
    super.onDestroy();
        unregisterReceiver(receiver);
    }

    @OnClick(R.id.button)
    void someMethod() {

        Intent intent = new Intent("com.journaldev.broadcastreceiver.SOME_ACTI
        sendBroadcast(intent);
    }
}
```

In the above code we've registered another custom action programmatically.

The ConnectionReceiver is defined in the `AndroidManifest.xml` file as below.

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

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <receiver android:name=".ConnectionReceiver">
        <intent-filter>
            <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
        </intent-filter>
    </receiver>
    </application>
</manifest>
```

The ConnectionReceiver.java class is defined below.

```
public class ConnectionReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {

        Log.d("API123",""+intent.getAction());

        if(intent.getAction().equals("com.journaldev.broadcastreceiver.SOME_AC
            Toast.makeText(context, "SOME_ACTION is received", Toast.LENGTH_LC

        else {
            ConnectivityManager cm =
                    (ConnectivityManager) context.getSystemService(Context.CON

            NetworkInfo activeNetwork = cm.getActiveNetworkInfo();
            boolean isConnected = activeNetwork != null &&
                    activeNetwork.isConnectedOrConnecting();
```

```
            if (isConnected) {
                try {
                    Toast.makeText(context, "Network is connected", Toast.LENG
                } catch (Exception e) {
                    e.printStackTrace();
                }
            } else {
                Toast.makeText(context, "Network is changed or reconnected", T
            }
        }
    }

}
```

In the above code we check the intent action that triggers the `onReceive()` method and based on that display the toast.

Note: To make the broadcast receiver unavailable to external applications, add the attribute `android:exported=false` in the manifest. When we send a broadcast, it is possible for the external applications too to receive them. This can be prevented by specifying this limitation.

The output app in action is given below.

This brings an end android BroadcastReceiver tutorial. You can download the final BroadcastReceivers project from the link below.