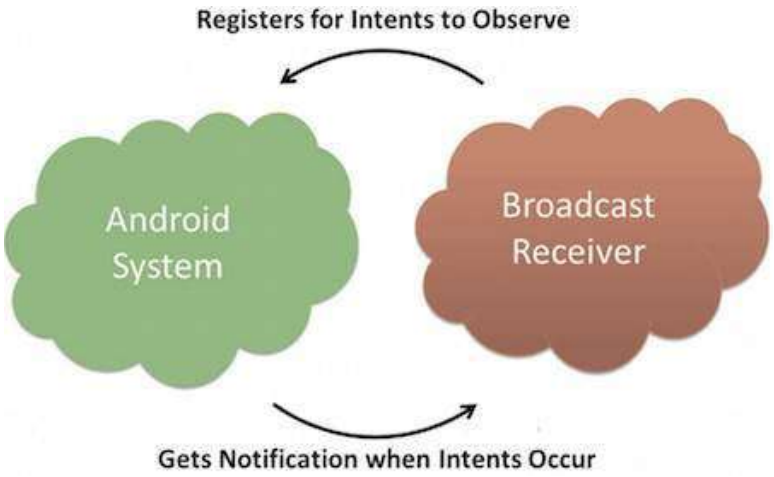
Android BroadcastReceiver

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_SHORT).show();
    }
}
```

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.

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_CHANGE" />
            </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_CHANGE");

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 anintent to all the related BroadcastReceivers.

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

Android BroadcastReceiver Example

To setup a Broadcast Receiver in our android application we need to do following things.

- Create a BroadcastReceiver
- Register a BroadcastReceiver

Create a new android application using android studio and give names as **BroadcastReceiver**.
Now we need to create our own broadcast content file **MyBroadcastReceiver.java** in **\src\main\java\com.l3350.broadcastreceiver** path to define our actual provider and associated methods for that right click on your application folder → Go to **New** → select **Java Class** and give name as **MyBroadcastReceiver.java**.

Once we create a new file **MyBroadcastReceiver.java**, open it and write the code like as shown below

MyBroadcastReceiver.java

```
package com.l3350.broadcastreceiver;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

/**


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

```
      CharSequence iData = intent.getCharSequenceExtra("msg");
      Toast.makeText(context,"I3350 Received Message: "+iData,Toast.LENGTH_LONG).sho
w();
   }
}
```

Now open **activity_main.xml** file from **\src\main\res\layout** path and write the following code.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical" android:layout_width="match_parent"
   android:layout_height="match_parent">
   <EditText
      android:id="@+id/txtMsg"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginLeft="100dp"
      android:layout_marginTop="180dp"
      android:ems="10"/>
   <Button
      android:id="@+id/btnShow"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:onClick="onClickShowBroadcast"
      android:layout_marginLeft="130dp"
      android:text="Show Broadcast"/>
</LinearLayout>
```

Now open **MainActivity.java** file from **\java\com.I3350.broadcastreceiver** path and write following to implement custom broadcast intents.

MainActivity.java

```
package com.I3350.broadcastreceiver;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {

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

```
    public void onClickShowBroadcast(View view){
        EditText st = (EditText)findViewById(R.id.txtMsg);
        Intent intent = new Intent();
        intent.putExtra("msg",(CharSequence)st.getText().toString());
        intent.setAction("com.I3350.CUSTOM_INTENT");
        sendBroadcast(intent);
    }
}
```

Now we need to register our broadcast receiver in android manifest file (**AndroidManifest.xml**) using **<receiver>**attribute like as shown below.

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.I3350.broadcastreceiver">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        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="MyBroadcastReceiver">
            <intent-filter>
                <action android:name="com.I3350.CUSTOM_INTENT">
                </action>
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

Example: AIRPLANE_MODE system broadcast

In this example we will create a receiver which will listen to the AIRPLANE_MODE system broadcast. That means every time the airplane mode is switched on/off android will fire this broadcast and our receiver will be notified.

We will first create a custom receiver class which will extend the BroadcastReceiver abstract class. Our class will have to implement the *onReceive*method.

```
public class AirplaneBroadcast extends BroadcastReceiver {
/*
This method will be invoked when any of the subscribed broadcasts are

fired. We can check which broadcast was fired by using intent.getAction()

This will be usefull if the same receiver is listening to multiple broadcasts

*/
@Override

public void onReceive(Context context, Intent intent) {

//We are checking which broadcast was actually fired

if (intent.getAction() == Intent.ACTION_AIRPLANE_MODE_CHANGED) {

Toast.makeText(context, "Airplace MOdeChanged", Toast.LENGTH_SHORT).show();

}
}
}
```

Once you have created your custom receiver you should now register the it.In this example we will register it in the manifest. Meaning it will always listen for that event. Copy the cope snippet below in your AndroidManifest.xml. You can see how we have specified AIRPLANE_MODE broadcaast in the <intent-filter> tag. This will register our receiver to listen for Airplance mode change events.

```
1. <receiver android:name=".utils.AirplaneBroadcast">
2. <intent-filter>
3. <action android:name="android.intent.action.AIRPLANE_MODE"/>
4. </intent-filter>
5. </receiver>
```