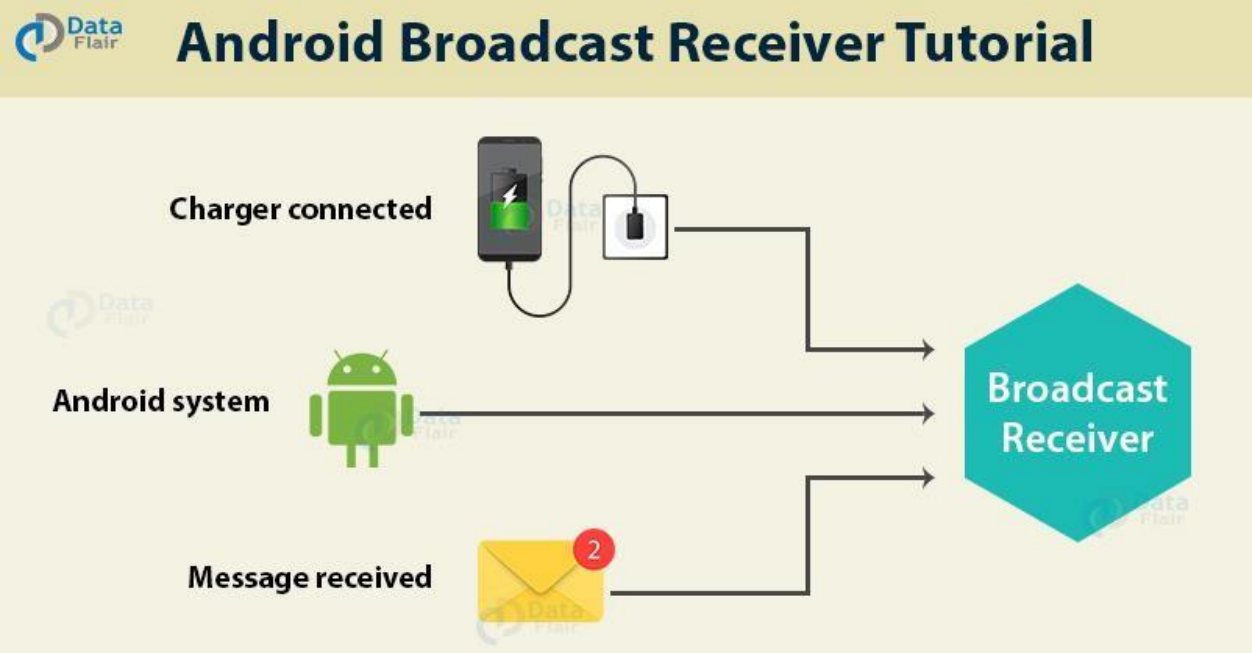
ANDROID BROADCAST RECEIVER

What is Android Broadcast Receiver?

Android Broadcast Receiver is an Android component that is used to broadcast the messages to the system or other applications. The broadcast message is referred to as an Event or Intent. Broadcast

receivers, unlike ***Activities***, have no user interface. It's working is similar to that of a publish-subscribe design pattern. It's used for Asynchronous Inter-Process communication.



Some Android broadcast receiver examples – *low battery notification in the notification bar by the system, notification to other applications when something downloads*, so they can use it when required.

Let us get started with system-generated intents.

System-generated Intents

Let us see some system-generated Intents which are important and are generally

used:

1. **android.intent.action.BATTERY_CHANGED –** This keeps track of the battery's charging state, percentage, and other information about the battery.
2. **android.intent.action.BATTERY_LOW –** It indicates the low battery condition.
3. **android.intent.action.POWER_CONNECTED –** It indicates that the power is connected to the device.
4. **android.intent.action.POWER_DISCONNECTED –** The power is disconnected from the device.
5. **android.intent.action.BOOT_COMPLETED –** This broadcast is shown only once when the device boots for the first time.
6. **android.intent.action.CALL –** This intent is to perform a call to some specific person, according to data.
7. **android.intent.action.DATE_CHANGED –** This means the date of the device has changed.
8. **android.intent.action.REBOOT –** This means that the device has rebooted.
9. **android.intent.action.CONNECTIVITY_CHANGE –** This shows the network connectivity of the device has changed.
10. **android.intent.action.BUG_REPORT –** This reports the bugs if there is any.
11. **android.intent.action.CALL_BUTTON –** The user pressed the call button to make a call, which takes them to an appropriate user interface.

CLASSES OF BROADCAST IN ANDROID

There are broadly two types of broadcast receivers in Android:

1. Ordered Broadcasts

2. Normal Broadcasts

1. Ordered Broadcasts

Ordered Broadcasts are synchronous broadcasts, and are done in proper order. This order is decided by the **android:priority attribute**. The broadcast with the highest priority would execute first and broadcasts with the same priority would not follow any order.

Ordered Broadcasts are done in proper order as they are assigned with some priority. The priority is assigned to the broadcasts using attribute **android:priority**. With the help of priority, the broadcast with the highest priority will execute first. If two broadcasts are found to have the same priority then there no order would be followed. Ordered broadcasts are also called as **Synchronous Broadcasts**.

In this, one broadcast is delivered only to one receiver at a time. When a receiver receives a broadcast it's up to the receiver to pass or abort the broadcast. If a receiver wants, it passes the broadcast to the next receiver else the broadcast doesn't reach the next receiver.

2. Normal Broadcasts

Normal broadcasts are asynchronous and unordered. These receivers run unorderly or all at a time, sometimes. That's why they are efficient, but lack full utilization of the results. These broadcasts are sent using **Context:sendBroadcast**.

In normal broadcast, it's possible that the system sends only one broadcast at a time in order to avoid overhead. It can also abort the broadcast, but those excluding APIs.

Notification that we receive from the applications can also be muted.

Implementation of Broadcast Receivers in Android

In order to build a good application, we need to take the utmost care of the Android broadcast receivers. If our application has perfectly working receivers, it'd be good and interactive.

Now to implement a receiver, we would need to register it, which could be done in the following two ways:

1. Context-registered

This is also known as the dynamically registering method. In this, the registration is done using **Context.registerReceiver()** method.

2. Manifest-registered

This is also known as the statically registering method. In this, the registration is done in the manifest file, using <**register**> tags.

now we'll see how to implement BroadcastReceivers in Android Studio.

First of all, we'll create a new application in Android Studio, we'll name it as**BroadcastReceiver**.

Within the java folder of BroadcastReceiver, create a java file named**MyBroadcastReceiver.java**