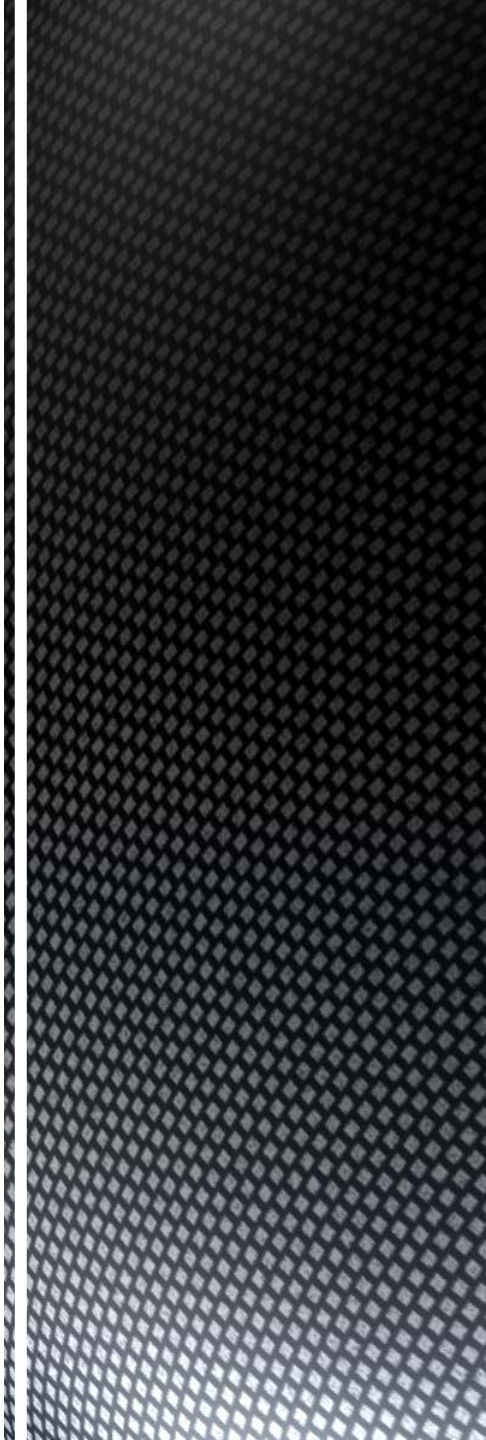
Android

Broadcasting



BroadcastReceiver

Recordatorio:

Un Intent es un mensaje que se puede usar para requerir la realización de una acción por parte de otro componente de nuestra u otra aplicación.

Sus usos funadmentales son:

- Lanzar una Activity
- **Enviar un broadcast**
- Lanzar un Service

Relación de Intents y Broadcast para una versión en concreto:
En el **directorio de instalación del sdk** bajo la platform correspondiente hay un fichero donde se lista cada uno estos ficheros son :
<ususario>\AppData\Local\Android\sdk\platforms\<platform>\
Para Intents: activity_actions.txt
Para Broadcasts: broadcast_actions.txt

Broadcast

Un broadcast es un mensaje que cualquier aplicación puede recibir.

Broadcast receiver

Un broadcast receiver (receiver) es un componente de Android que nos permite registrarnos para recibir eventos de aplicaciones y el propio sistems.

Todos los receivers registrados para un evento son notificados por el sistema cuando sucede el evento..

Enviar un broadcast:

El sistema envía varios broadcasts correspondientes a eventos del sistema, como arranque del sistema (ACTION_BOOT_COMPLETED), comienzo de carga de batería, batería baja, modo avión, etc...

Puedes enviar un broadcast a otras aplicaciones pasando un intent mediante los médotos

- sendBroadcast(),
- sendOrderedBroadcast(),
- sendStickyBroadcast().

Broadcasting in Android

Contenidos

- **The BroadcastReceiver Class**

  Revision de la clase BroadcastReceiver y el Workflow
- **Registration**

  Cómo registrar un BroadcastReceiver en el sistema
- **Broadcast**

  Diferentes formas de realizar broadcast de los eventos
- **Processing**

  Cómo el sistema notifica a un BroadcastReceiver y cómo maneja los broadcast que recibe.

BroadcastReceiver

BroadcastReceiver es la clase base para definir componentes cuyo proposito va a ser esperar a que ocurran ciertos eventos y reaccionar ante ellos.

Un BroadcastReceiver, se registra para recibir un evento especifico en el que está interesado.

Por ejemplo:

Hay un BroadcastReceiver en Android cuyo trabajo es esperar a que haya MMS de salida.

Otro componente, genera el MMS y hace un broadcast de este MMS a través de un Intent con Action: ACTION_SEND message

El BroadcastReceiver, fue registrado para escuchar intents con el Action: ACTION_SEND.

El BroadcastReceiver, recive el intent en su método onReceive

Broadcasting Workflow

1. Los BroadcastReceiver se registran para recibir ciertos Intents
2. Algún componente genera un Intent y hace broadcast al sistema.
3. Android entrega dicho Intent a los BroadcastReceiver que se registraron para recibirlo.
4. El sistema lanza el método onReceive del BroadcastReceiver donde manejará el evento entrante.

Implementando el receiver

Para implementar un receiver debemos extender de la clase BroadcastReceiver.

Crearemos una clase que extienda de BroadcastReceiver
y sobrecargamos el callback onReceive()

El método onReceive(Context,Intent) recibe el Intent que viene con el Broadcast

```
public class MyReceiver extends BroadcastReceiver {

private final String TAG = "Receiver";

@Override
public void onReceive(Context context, Intent intent) {
           Log.i(TAG, “Broadcast Intent received");
           }
}
```

Registering a BroadcastReceiver

Para registrar un BroadcastReceiver el programador tiene dos opciones.

- Registro estático

  Vía AndroidManifest.xml de la aplicación a la que pertenece el BroadcastReceiver que se está registrando.

- Registro dinámico

  En tiempo de ejecución con Context.registerReceiver()

Registro estático

Añadimos etiqueta <receiver> y al menos un <intent-filter>

```
<receiver
  android:enabled=[“true” | “false”] //Para hablitar o no el BroadcastReceiver
  android:exported=[“true” | “false”] //true si recibimos de fuera de la aplicación
  android:icon=“drawable resource”
  android:label=“string resource”
  android:name=“string” //nombre clase que implementa el receiver
  android:permission=“string“ //define permiso que el emisor debe tener
  <intent-filter>
      ... //action, data, categories
  </intent-filter>
</receiver>
```

Ver <receiver>

Nombre de la clase que implementa el receiver

El intent-filter está declarado estáticamente

Ver <intent-filter>

```
<receiver android:name="MyReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED">
        </action>
    </intent-filter>
</receiver>
```

Esta información es leída:

- Cuando el sistema arranca
- Cuando se carga el package (el sistema ya está corriendo)

E1001_SimpleBroadcast

Registro dinámico

Para registrar dinámicamente un BroadcastReceiver necesito un BroadcastManager que me permita el registro dinámico llamando a su método registerReceiver() :

- Usaremos el método registerReceiver() en el onCreate() de la aplicación.
- Se debe des-registrar en el onDestroy() mediante unregisterReceiver().

Si queremos que nuestro BroadcastReceiver responda sólo cuando nuestra actividad está en el “foreground”, podemos registrarla en el onResume() y des-registrarla en el onPause()

El método registerReceiver() está implementado en varios BroadcastManager y usaremos uno de ellos:

- En la clase LocalBroadcastManager si los broadcast que vamos a enviar sólo están dirigidos a la aplicación local y no necesitan ser propagados al resto del sistema.
- La clase Context también lo implementa y propagará el broadcast a todo el sistema por lo que cualquier aplicación es susceptible de recibirlo.

Registro dinámico - Workflow

El Workflow sería:

1. Creamos un objeto IntentFilter especificando en el constructor el tipo de Intent mediante el Action string
2. Creamos el BroadcastReceiver creando un nuevo objeto de nuestra clase receiver.
3. Registramos llamando a una implementación de registerReceiver(), bien la del LocalBroadcastManager o la de Context
4. Enviaremos el broadcast usando el método sendBroadcast(), igualmente el de del LocalBroadcastManager o la de Context
5. Des-registrar el BroadcastReceiver, llamando a unRegisterReceiver()

E1002_SimpleDinBroadcast

Events

1. Android puede enviar los Broadcast Intent de forma Normal u ordenada (in order)
   a) Los Normal Broadcasts se envían a los BroadcastReceiver sin un orden predefinido, pudiendo dos receivers recibir el mismo intent y procesarlos al mismo tiempo.
   b) Ordered broadcasts envían el Intent a multiples receivers uno detras de otro segun la prioridad establecida. Ver <intent-filter>
   c) Los Broadcast pueden ser sticky o non-sticky. (pegados o no-pegados)

Ej. Un intent sticky se queda después de que el broadcast se haya iniciado.
De esta forma, un receiver que se registre después del broadcast todavía recibiría el intent.
Útil cuando queremos saber el estado de algo (nivel de bateria)

Ej. Non-sticky – no se queda tras el broadcast, así que los receivers registrados a posteriori no reciben el intent.
Interesante cuando deseamos saber cuándo se produce un cambio en el algo.