102. BroadCast Receiver

Broadcast Receiver está diseñado para escuchar y reaccionar a eventos o "transmisiones" que suceden dentro de tu aplicación o en el sistema operativo. Estos eventos pueden variar desde notificaciones sobre batería baja, hasta mensajes de otras aplicaciones o incluso eventos del sistema como el arranque del dispositivo.

102.1 Crear un Broadcast Receiver en la aplicación

1. Crear la clase para escuhar derivada de `BroadcastReceiver`
2. Declarar el `BroadcastReceiver`: Puedes hacerlo de dos maneras. La primera es declararlo en el `AndroidManifest.xml` de tu aplicación, lo que es útil para eventos que quieres escuchar todo el tiempo, incluso si tu aplicación no está en ejecución. La segunda manera es registrarlo dinámicamente en tu código (por ejemplo, en una actividad), lo que te da control sobre cuándo quieres empezar a escuchar esos eventos.
3. Implementa `onReceive()`: Este es el método donde defines lo que tu app hará cuando reciba el evento esperado. Aquí es donde la "persona" de nuestro ejemplo se pone de pie y baila o apaga la luz.
4. Filtrar los `Intents`: Para asegurarte de que tu BroadcastReceiver solo reaccione a eventos específicos, utilizas filtros de intents. Estos filtros pueden especificarse en el AndroidManifest para receptores estáticos o en tu código para receptores dinámicos.
5. Registrar y Desregistrar (si es dinámico): Si estás utilizando un receptor dinámico, necesitarás registrar tu BroadcastReceiver cuando quieras que comience a escuchar (por ejemplo, en onStart() de tu actividad) y desregistrarlo cuando ya no quieras que escuche (por ejemplo, en onStop()).

102.2 Ejemplos

Para detectar cuando la batería del dispositivo está baja en Android, puedes utilizar un BroadcastReceiver que escuche el Intent `ACTION_BATTERY_LOW`. Este Intent es transmitido por el sistema cuando la batería del dispositivo cae por debajo de un nivel específico.

Para crear

```
class BatteryLevelReceiver(private val onBatteryLow: () -> Unit) : BroadcastReceiver() {
    override fun onReceive(context: Context?, intent: Intent?) {
        if (intent?.action == Intent.ACTION_BATTERY_LOW) {
            // La batería está baja
            // Realiza la acción correspondiente
            Log.d("BatteryLevelReceiver", "Batería baja detectada.")

            // ademas
            onBatteryLow()
        }
    }
}
```

Para registrarlo

Registro dinámico

```
class MyActivity : AppCompatActivity() {

    private lateinit var batteryLevelReceiver: BatteryLevelReceiver

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        // Instancia el BroadcastReceiver
        batteryLevelReceiver = BatteryLevelReceiver()

        // Registra el BroadcastReceiver
        IntentFilter(Intent.ACTION_BATTERY_LOW).also {
            registerReceiver(batteryLevelReceiver, it)
        }

    }

    override fun onDestroy() {
        super.onDestroy()
        // No olvides desregistrar el BroadcastReceiver
        unregisterReceiver(batteryLevelReceiver)
    }
}
```

Registro estático

```
<receiver android:name=".BatteryLevelReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BATTERY_LOW"/>
    </intent-filter>
</receiver>
```

Una forma alternativa:

En la funcion composable usamos `DisposableEffect` para registrar y desregistrar basado en el ciclo de vida del composable

```
@Composable
fun MyComposableActivity() {
    val context = LocalContext.current

    // Manejar el BroadcastReceiver con DisposableEffect
    DisposableEffect(key1 = context) {
        val batteryLevelReceiver = BatteryLevelReceiver {
            // Aquí puedes manejar lo que sucede cuando la batería está baja
            Log.d("BatteryLevelReceiver", "La batería está baja.")
        }
        val intentFilter = IntentFilter(Intent.ACTION_BATTERY_LOW)
        context.registerReceiver(batteryLevelReceiver, intentFilter)

        onDispose {
            context.unregisterReceiver(batteryLevelReceiver)
        }
    }

    // Tu UI aquí
```

```
    Text(text = "Monitor de nivel de batería")
}
```

102.3 Emitir Broadcast

una aplicación de usuario en Android puede emitir un broadcast. Los broadcasts en Android son un mecanismo de comunicación que permite enviar y recibir mensajes entre aplicaciones o dentro de la misma aplicación, sin necesidad de una conexión directa entre ellas. Estos mensajes se denominan "`intents`".

Existen dos tipos de broadcasts que una aplicación puede emitir:

1. **Broadcasts Explícitos**: Estos son enviados directamente a un componente específico (por ejemplo, un BroadcastReceiver registrado) dentro de la propia aplicación o en otra aplicación. Son más seguros y privados, ya que especificas exactamente a qué componente debe entregarse el mensaje.

```
val intent = Intent(this, MyBroadcastReceiver::class.java)
intent.putExtra("key", "value")
sendBroadcast(intent)
```

2. **Broadcasts Implícitos**: Son broadcasts que no están dirigidos a un componente específico, sino que declaran una acción general. Cualquier aplicación o componente que haya registrado un BroadcastReceiver para esa acción puede recibir y procesar este tipo de broadcast. **Sin embargo, debido a preocupaciones de seguridad y rendimiento, Android ha restringido el uso de broadcasts implícitos en versiones recientes**.

```
val intent = Intent("com.example.MY_ACTION")
intent.putExtra("key", "value")
sendBroadcast(intent)
```

102.4 Enviar emisiones.

En AD

102.5 Restringir emisiones con permisos

En AD

102.6 Recibir emisiones con permisos

En AD

102.7 Consideraciones de seguridad y prácticas recomendadas

En AD

102.8 Excepciones de trasmisión implicita

En AD

Como parte de los Límites de ejecución en segundo plano de **Android 8.0 (API nivel 26)**, las apps que se orientan a la API nivel 26 o superior ya **NO pueden registrar** receptores de emisión para emisiones implícitas en su manifiesto. Sin embargo, varias emisiones actualmente están exentas de estas limitaciones. Las apps pueden seguir registrando objetos de escucha para las siguientes emisiones, sin importar el nivel de API al que se orienten las apps.

Lista de emisiones exentas de excepciones:

ACTION_LOCKED_BOOT_COMPLETED, ACTION_BOOT_COMPLETED

Están exentas porque estas emisiones solo se envían una vez, en el primer inicio, y muchas apps necesitan recibirlas a fin de programar tareas, alarmas, etcétera.

ACTION_USER_INITIALIZE, "android.intent.action.USER_ADDED", "android.intent.action.USER_REMOVED"

Estas emisiones están protegidas con permisos privilegiados, de manera que las apps comunes no pueden recibirlas. "android.intent.action.TIME_SET", ACTION_TIMEZONE_CHANGED, ACTION_NEXT_ALARM_CLOCK_CHANGED Es posible que las apps de reloj necesiten recibir estas emisiones a fin de actualizar las alarmas cuando se cambian la hora, la zona horaria o las alarmas.

ACTION_LOCALE_CHANGED

Solo se envía cuando la configuración regional cambia, lo que no ocurre con frecuencia. Es posible que las apps necesiten actualizar sus datos cuando cambie la configuración regional.

ACTION_USB_ACCESSORY_ATTACHED, ACTION_USB_ACCESSORY_DETACHED, ACTION_USB_DEVICE_ATTACHED, ACTION_USB_DEVICE_DETACHED

Si una app necesita información sobre estos eventos relacionados con USB, actualmente no existe una buena alternativa de registro para recibir la emisión.

ACTION_CONNECTION_STATE_CHANGED, ACTION_CONNECTION_STATE_CHANGED, ACTION_ACL_CONNECTED, ACTION_ACL_DISCONNECTED

No es probable que la experiencia del usuario se vea afectada si las apps reciben emisiones para estos eventos Bluetooth.

ACTION_CARRIER_CONFIG_CHANGED, TelephonyIntents.ACTION_*_SUBSCRIPTION_CHANGED, "TelephonyIntents.SECRET_CODE_ACTION", ACTION_PHONE_STATE_CHANGED, ACTION_PHONE_ACCOUNT_REGISTERED y ACTION_PHONE_ACCOUNT_UNREGISTERED

Es posible que las apps de telefonía del OEM necesiten recibir estas emisiones.

LOGIN_ACCOUNTS_CHANGED_ACTION

Algunas apps necesitan conocer los cambios en las cuentas de acceso a fin de poder configurar operaciones programadas para las cuentas nuevas y modificadas.

ACTION_ACCOUNT_REMOVED

Las apps que tienen visibilidad de una cuenta reciben esta emisión cuando se quita la cuenta. Si este es el único cambio de cuenta en el que la app necesita actuar, se recomienda encarecidamente que la app use esta emisión en lugar de la LOGIN_ACCOUNTS_CHANGED_ACTION obsoleta.

ACTION_PACKAGE_DATA_CLEARED

Solo se envía cuando el usuario borra explícitamente sus datos de la Configuración, por lo que es poco probable que los receptores de emisión afecten significativamente la experiencia del usuario.

ACTION_PACKAGE_FULLY_REMOVED

Es posible que algunas apps necesiten actualizar sus datos almacenados cuando se quite otro paquete; en el caso de estas apps, no hay una buena alternativa de registro para recibir esta emisión.

Nota: Otras emisiones relacionadas con paquetes (como ACTION_PACKAGE_REPLACED) no están exentas de las nuevas restricciones. Estas emisiones son lo suficientemente comunes como para que exista un impacto potencial en el rendimiento si se eximen.

ACTION_NEW_OUTGOING_CALL

Las apps que toman medidas en respuesta a los usuarios que realizan llamadas deben recibir esta emisión.

```
Esta constante fue declarada obsoleta en el nivel de API 29. Las aplicaciones que
redireccionan llamadas salientes deben usar la API CallRedirectionService. Las
aplicaciones que realizan el filtrado de llamadas deben usar la API
CallScreeningService. Las aplicaciones que necesitan ser notificadas sobre el estado
básico de llamada deben usar PhoneStateListener.onCallStateChanged(int, String) para
determinar cuándo se realiza una nueva llamada saliente.
```

ACTION_DEVICE_OWNER_CHANGED

Esta emisión no se envía con mucha frecuencia; algunas apps necesitan recibirla a fin de saber que se modificó el estado de seguridad del dispositivo.

ACTION_EVENT_REMINDER

Lo envía el proveedor de calendario a fin de publicar un recordatorio de evento a la app de calendario. Como el proveedor de calendario no sabe cuál es la app de calendario, esta emisión debe ser implícita.

ACTION_MEDIA_MOUNTED, ACTION_MEDIA_CHECKING, ACTION_MEDIA_UNMOUNTED, ACTION_MEDIA_EJECT, ACTION_MEDIA_UNMOUNTABLE, ACTION_MEDIA_REMOVED, ACTION_MEDIA_BAD_REMOVAL

Estas emisiones se envían como resultado de las interacciones físicas del usuario con el dispositivo (instalación o eliminación de volúmenes de almacenamiento) o como parte de la inicialización (a medida que se montan los volúmenes disponibles), por lo que no son frecuentes y, por lo general, las administra el usuario.

SMS_RECEIVED_ACTION, WAP_PUSH_RECEIVED_ACTION

Las apps que reciben SMS se basan en estas emisiones.

102.9 Apendice

Terminos:

- Broadcast Receiver : Receptores de emisiones

Enlaces:

- Descripción general de las trasmisiones

Versión 0.5 12-12-23
Versión 0.8 8-1-24

¿Fue útil esta página?