Broadcast Receiver (1)

Un receptor de eventos (*Broadcast Receiver*) es un componente de Android que reacciona ante un evento de sistema como sistema operativo cargado, nivel de batería bajo, SMS recibido, etc.

Los eventos son notificados a través de *Intents*, a través de los cuales podremos acceder a información adicional (extras).

Los *Broadcast Receivers* no tienen interfaz de usuario ni deben realizar modificaciones en la interfaz, únicamente podrán mostrar notificaciones.

Además deben realizar tareas ligeras, si se requiere una tarea pesada habrá que iniciar una actividad o un servicio.

51

Broadcast Receiver (2)

- Recibir un evento

Podemos crear *Broadcast Receiver* que respondan ante diferentes eventos del sistema.

La lista de eventos es muy extensa y cada uno tiene una constante asociada.

Algunos ejemplos:

- SMS recibido: android.provider.Telephony.SMS_RECEIVED
- Nivel de batería bajo: android.intent.action.BATTERY_LOW
- Modo avión: android.intent.action.AIRPLANE_MODE
- Botón cámara: android.intent.action.CAMERA_BUTTON
- Reinicio: android.intent.action.REBOOT
- Pantalla apagada: android.intent.action.SCREEN_OFF

52

Broadcast Receiver (3)

- Recibir un evento

Método 1 (API<26): registrar receptor en manifiesto

Este mecanismo requiere dos acciones:

1) Crear una subclase de *BroadcastReceiver* e implementar el método *onReceive()* que recibirá el contexto y un *intent* a través del cual se puede conocer el tipo de notificación que hemos recibido y otra información adicional (extras).

   Si un Broadcast Receiver sólo se va a utilizar dentro de la aplicación (local), en lugar de la clase *BroadcastReceiver*, deberemos utilizar la *LocalBroadcast*.

53

Broadcast Receiver (4)

- Recibir un evento

```
public class Receptor extends BroadcastReceiver {
  @Override
  public void onReceive(Context context, Intent intent) {
    if (intent.getAction()
        .equals("android.intent.action.AIRPLANE_MODE")) {
      String mensa = "";
      int duracion = Toast.LENGTH_SHORT;
      Bundle extras = intent.getExtras();
        if (extras != null) {
          if (extras.getBoolean("state") == true) {
            mensa = "Modo Avión activado";
          } else {
            mensa = "Modo Avión desactivado";
          }
        }
        Toast.makeText(context, mensa, duracion).show();
  }
}
```

54

Broadcast Receiver (5)

- Recibir un evento

2) Registrar la subclase en el archivo de manifiesto como candidata a recibir notificaciones.

   Para ello se especificará mediante la etiqueta *<receiver>* dentro de *<application>*, un filtro de *intent* con el evento que se espera recibir.

```
<receiver android:name=".Receptor" android:exported="false">
    <intent-filter>
        <action android:name="android.intent.action.AIRPLANE_MODE"/>
    </intent-filter>
</receiver>
```

   Si para gestionar el evento se requiere algún permiso tendremos que solicitarlo.

55

Broadcast Receiver (6)

- Recibir un evento
  - *receiver*: permite declarar el *BroadcastReceiver*.
    - *name*: establece la clase que se encarga de gestionar los eventos recibidos.
    - *exported* (*boolean*): permite indicar si otras actividades que no pertenecen a la aplicación pueden utilizar el *Broadcast Receiver*.
  - *intent-filter*: filtro de *intent* en el que se definen los eventos a los que nuestro *Broadcast Receiver* va a responder.
  - *action*: establece cada uno de los eventos que se desea recibir.

56

Broadcast Receiver (7)

- Recibir un evento

Método 2: registrar el receptor de forma dinámica

(único método válido a partir de Android 8, api 26)

En el método anterior, el *Broadcast Receiver* se crea automáticamente al iniciar la aplicación y permanece durante toda la ejecución.

Alternativamente, podemos registrar el *Broadcast Receiver* de forma dinámica y decidir si lo asociamos a la aplicación o a una actividad concreta.

En ambos casos tendremos que crear un filtro de *intent*, añadir los eventos mediante *addAction* y registrar el receptor mediante *registerReceiver.*

57

Broadcast Receiver (8)

- Recibir un evento

Receptor asociado a la aplicación

En algún punto de nuestra aplicación creamos el *Broadcast Receiver*, por ejemplo en el método *onCreate* de nuestro *MainActivity*, y a la hora de registrarlo lo hacemos sobre el contexto de la aplicación.

```
IntentFilter filtro = new IntentFilter();
filtro.addAction("android.intent.action.BATTERY_LOW");
filtro.addAction("android.intent.action.AIRPLANE_MODE");
Receptor receptor = new Receptor();
getApplicationContext().registerReceiver(receptor, filtro);
```

Nota: la clase Receptor es la misma que creamos para el método anterior.

58

Broadcast Receiver (9)

- Recibir un evento

Receptor activo en el ciclo de vida de la actividad

En proceso de creación y registro se realizará en su método *onCreate*, y después anularemos el registro en su método *onDestroy*.

```
public class MainActivity extends AppCompatActivity {
  protected Receptor receptor;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    ...
    IntentFilter filtro = new IntentFilter();
    filtro.addAction("android.intent.action.BATTERY_LOW");
    filtro.addAction("android.intent.action.AIRPLANE_MODE");
    receptor = new Receptor();
    registerReceiver(receptor, filtro);
  }
  @Override
  protected void onDestroy() {
    super.onDestroy();
    unregisterReceiver(receptor);
  }
}
```

59

Broadcast Receiver (10)

- Recibir un evento

Receptor activo mientras la actividad es visible

En proceso de creación y registro se realizará en su método *onResume*, y la anulación del registro en su método *onPause*.

```
public class MainActivity extends AppCompatActivity {
  protected Receptor receptor;
  ...
  @Override
  protected void onResume() {
    super.onResume();
    IntentFilter filtro = new IntentFilter();
    filtro.addAction("android.intent.action.BATTERY_LOW");
    filtro.addAction("android.intent.action.AIRPLANE_MODE");
    receptor = new Receptor();
    registerReceiver(receptor, filtro);
  }
  @Override
  protected void onPause() {
    super.onPause();
    unregisterReceiver(receptor);
  }
}
```

60

Broadcast Receiver (11)

- Enviar un evento

También podemos enviar nuestro propios *Broadcast* desde nuestra aplicación utilizando el método *sendBroadcast*.

```
String MY_BROADCAST="com.example.broadcast.MY_BROADCAST";
Intent broadcast = new Intent(MY_BROADCAST);
sendBroadcast(broadcast);
```

Podríamos añadir datos en el *Broadcast* mediante el método *putExtra* como en cualquier *Intent*.

61

Broadcast Receiver (12)

- Enviar un evento

Para responder a este evento tendríamos que crear una clase y registrar el receptor en el archivo de manifiesto igual que con los Broadcast de sistema.

```
<receiver android:name=".Receptor" android:exported="false">
    <intent-filter>
        <action android:name="com.example.broadcast.MY_BROADCAST"/>
    </intent-filter>
</receiver>
```

62