6 – CLASES PRINCIPALES

CONTENT PROVIDER

Es el mecanismo previsto por Android para compartir información entre distintas apps.

¿Podemos acceder a los Contactos desde nuestra app? Sí, gracias a un Content Provider

CONTENT PROVIDER

Por tanto, puedo:

1 Consumir un Content Provider

2 Insertar, borrar, modificar CP

3 Crear mi propio Content Provider

CONTENT PROVIDER

- Calendar (Fechas)
- Browser (Favoritos, Historial)
- CallLog (Llamadas recientes)
- MediaStore (Multimedia fotos, videos)
- Contacts (Contactos)
- Settings (Ajustes)
- UserDictionary (Palabras propias)

CONTENT PROVIDER

Cada CP tiene una URI que permite su referencia (como en los intents)

<prefijo>://<autoridad>/<datos>/<id>

content://app-package/tabla/id

CONTENT PROVIDER

PERMISOS: Es necesario incluir en el Manifest el permiso oportuno para usar un CP como fuente de datos

<uses-permission android:name="android.permission.READ_USER_DICTIONARY">

CONTENT PROVIDER

Para acceder al CP lo hago a través del ContentResolver, disponible a través del Context

getContentResolver().query ( )

CONTENT PROVIDER

```
Cursor query (Uri uri,
                String[] projection,
                String selection,
                String[] selectionArgs,
               String sortOrder)
```

CONTENT PROVIDER

Uri uri: Id el Content

String[] projection: Qué columnas quiero

String selection: WHERE filtrar fiilas

String[] selectionArgs: ? parametros

String sortOrder: ASC, DESC

AsyncTask

Desde la versión 3, ANDROID No permite conexiones a internet desde el hilo principal (GUI)

Lo normal al intentar conectar conectar es obtener el error

NetworkOnMainThreadException

AsyncTask

Para establecer conexiones a servidores, o dispositivos remotos debo:

1 Añadir permisos de conexión a internet en el Manifest

```
<uses-permission
android:name="android.permission.INTERNET" />
```

AsyncTask

2 Realizar la comunicación en un proceso aparte del Main. En una clase AsyncTask (que herede de esa clase)

AsyncTask Genéricos

Cuando heredo de una clase AsyncTask debo especificar los tipos genéricos

AsyncTask <Params, Progress, Result>

AsyncTask Métodos

```
@Override
protected Result
 doInBackground(Params ... params)
```

AsyncTask Métodos

@Override

protected void onPostExecute

(Result result)

AsyncTask Métodos

Para llamar a una clase AsynTask bastará instanciar un objeto de la clase y llamar al método execute()

ObtenerFecha().execute(p1, p2)

AsyncTask Métodos

Para obtener el resultado tenemos varias alternativas. Una sería usar get()

ObtenerFecha().execute(p1, p2).get()

AsyncTask Métodos

Otra pasar en el constructor el objeto al que llamar y usarlo en postExecute

ObtenerFecha(o).execute(p1, p2)

onPosExecute (Result r) { o.hazalgo (r); }

JSON

Es un formato de representación de información, similar a XML

En programación distribuida, se ha convertido en un estándar por ser agnóstico respecto a un lenguaje concreto

JSON

[{"nombre":"PEPA","edad":88},
{"nombre":"PEPO","edad":45}]

Para acelerar la equivalencia entre texto JSON y clases JAVA, usaremos la librería GSON, desarrollada por Google para Java

JSON

Para pasar de un objeto Java a un texto JSON (Serializar)

```
Gson gson = new Gson ();
String msj_JSON = gson.toJson(o_JAVA);
```

JSON

Para pasar de texto JSON a clase Java (Deserializar)

```
Gson gson = new Gson ();
o_JAVA = gson.fromJson (msj_JSON, o_JAVA.class);
```

JSON

Para pasar de texto JSON a clase compuesta Java (Deserializar lista)

```
ArrayList<Persona> persona =
gson.fromJson (isr,
new TypeToken<ArrayList<Persona>>() {}.getType());
```

Serializar

Serializar es pasar un objeto Java a su representación en bytes para que pueda ser transmitido o almacenado

El proceso inverso es deserializar: convertir de bytes a objeto Java

Serializar

Para poder serializar un objeto, lo único que debo indicar es que su clase implementa la interfaz seriaizable

MyClase implements Serializable

(no implica sobreescrbir ningún método)

Parcelabe

Serializable, es algo costoso. Por ello, los creadores de Android, han hecho la interfaz Parcelable.

Es idéntico a Serizable en idea (escribo objetos para transmitirlos) pero más eficiente.

Parcelabe

Mi clase deberá implementar Parcelabe si quiero escribir objetos de esa clase para poder transferirlos entre activities o guardarlos en el Bundle para recrear la actividad.

Al implementar Parcelabe, sí que debo sobreescribir varios métodos

Service

Un Service es una clase usada para desarrollar tareas repetitivas en “background” (no asociadas a una UI)

Además, un Service puede quedar definido y accesible para el resto de apps.

Service - TIPOS

STARTED.- Los más comunes. Son invocados mediante startService() No devuelven nada a la clase llamante. Al finalizar, el propio servicio se detiene

Service - TIPOS

BOUND.- Se invocan mediante bindService(). Puedo comunicarme en ambos sentidos con el llamante (Inter Process Communication). Múltiples llamantes pueden usar el servicio. Cuando dejan de usarlo, el servicio se destruye

Service

Un Servicio debe declararse en el Manifest (usemos el asistente de Studio)

```
<service
 android:name="MyService"
 android:icon="@drawable/icon"
 android:label="@string/service_name" >
</service>
```

Service

Y heredará de la clase Service

```
public class MyService extends Service {

  @Override //para Start services
  public int onStartCommand(Intent intent, int flags, int startId)
  //para Bound services
  IBinder onBind (Intent intent)
```

Service

```
public class MyService extends Service {


  @Override //para inicilizar, llamado una vez
  public void onCreate ()
   //la última llamada que recibe el servicio. Sobreescribir si se
//necsesita liberar recursos (threads, recivers, etc...)
  public onDestroy ()
```

Service

Puedo lanzar un start service desde una Activity, desde un Reciver o desde otro servicio. Para ello usaré el método Context.startService (intent)

(llamaría con bindService si fuera un bound Service)

Service - HILOS

¿Puedo conectarme a internet desde un Service? → ¿Estoy en un hilo separado del principal?

NO. Por defecto, el hilo Servicio se ejecuta en el mismo hilo que el programa principal

Service - HILOS

Si quiero que mi servicio se ejecute en un hilo independiente puedo:

1 Crear un AsynTask desde el servicio

2 Crear un Thread desde el servicio

3 Crear un IntentService

Service - HILOS

¿Merece la pena crear un Hilo nuevo para la ejecución del servicio?

- Sí, si voy a reaizar una tarea muy pesada (piensa que le quito tiempo de ejecución al hilo prinicipal GUI)
- Si necesito conectarme a internet

Service - HILOS

AMPLIACIÓN

https://developer.android.com/guide/components/processes-and-threads.html

IntentService

Heredo de IntentService para crear un servicio que cada vez que sea invocado, ejecutará secuencialmente (

onHandleIntent (Intent)

IntentService

Un único hilo, atenderá la llamada. Ante sucesivas llamadas, se crea automáticamente una cola de servicio, autogestionada por Android

El propio Servicio, se autodestruye (no me preocupo por finalizarlo)

Valores devueltos

Un Servicio, puede ser eliminado arbitrariamente por el Sistema.

Por ello, onStartCommand debe devolver un valor entero indicando qué hacer

Valores devueltos

onStartCommand debe devolver uno de las siguientes valores constantes

START_NOT_STICKY /No se reinicia/

START_STICKY /Se reinicia sin INTENT/

START_REDELIVER_INTENT /Se reincia con INTENT

ForegroundServices

Para que el Sistema tenga en consideración a un servicio a la par que una actividad, debo lanzar una notificación e invocar dentro de onStartCommand

startForeground(id_not, notification)

ForegroundServices

Este servicio tendrá presencia en la UI (mediante una notificación). Así, Android, en caso de tener que eliminar a un proceso, dejará a este tipo de servicios con la misma prioridad que una actividad

Servicios del Sistema

A través del método getSystemServices

puedo acceder a diferentes servicios que me oferece el sistema

https://developer.android.com/reference/android/content/Context.html#getSystemService(java.lang.String)

ConnectivityManager, LayoutInflater, DowloadManager, PowerService, etc..

IntentFilter

Es la antesala de la ejecución de un componente (servicio, reciver o activity)

Dada la descripción de un IntentFilter y su correspondencia (match) con un evento (broadcast) se ejecutará su componente asociado

BroadcastReceiver

Como resultado de una actividad o servicio, el componente finalizado puede enviar un señal de broadcast.

Esta, puede ser capturada por un IntentFilter y lanzar el objeto BroadcastReciver al que fue asociado

BroadcastReceiver

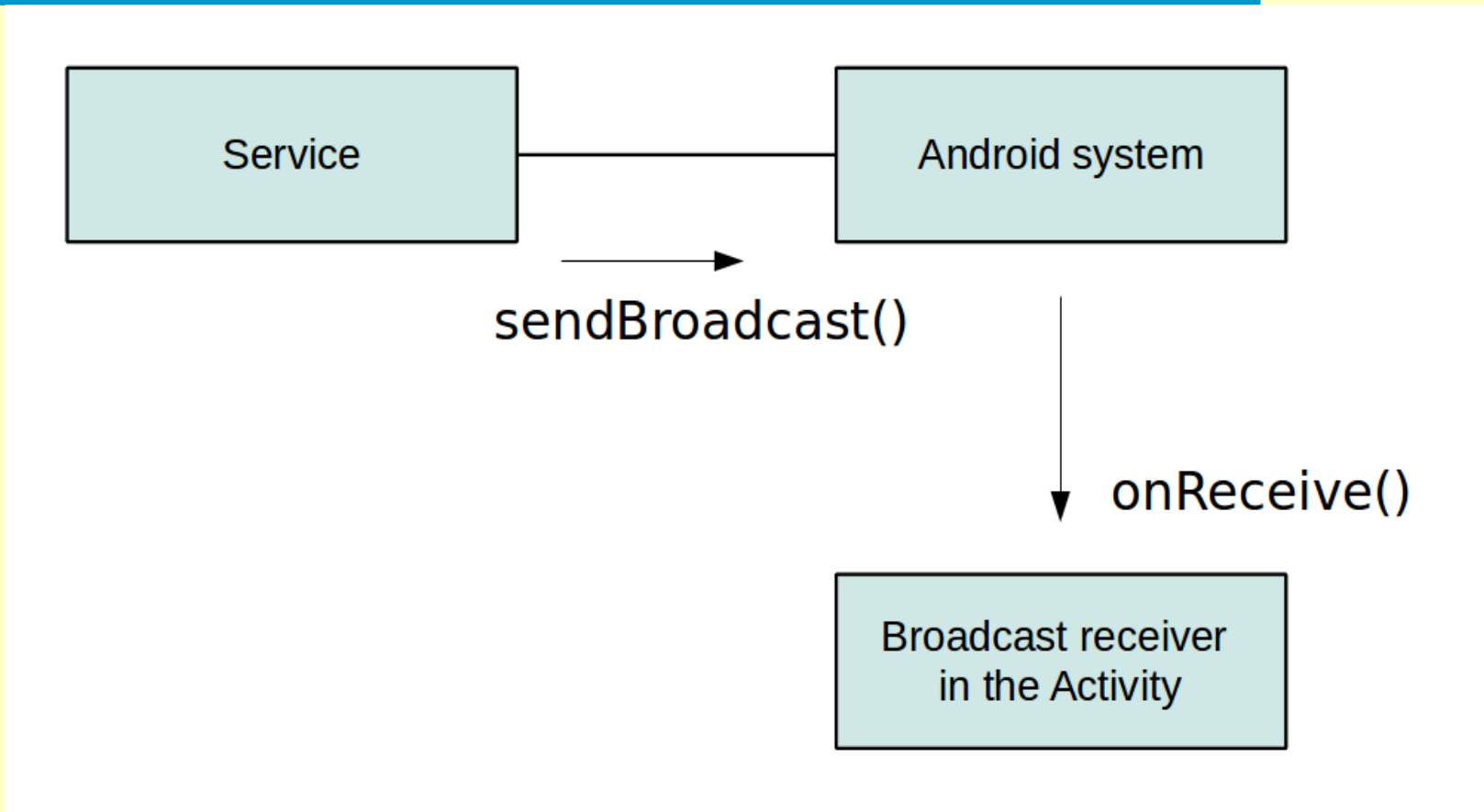
Puedo asociar un intent-filter con su broadcastreciver

- Dinámicamente (Context.registerReciver(reciver, filter)
- Estáticamente (En el Manifest)

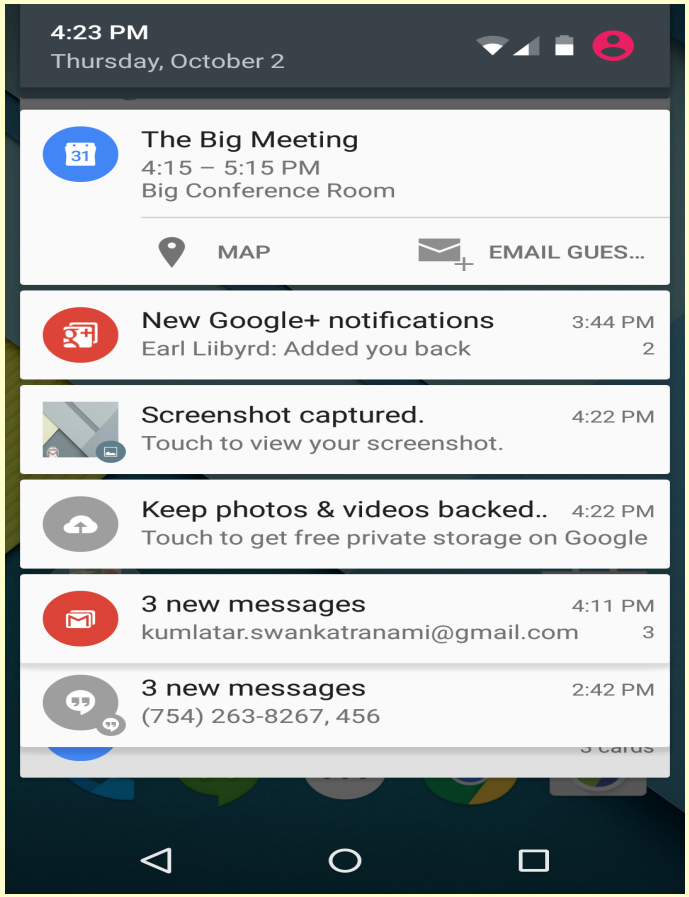
BroadcastReceiver

Para declarar un BroadcastReciver, usaré el asistente para que me cree la entrada en el Manifest y heredaré de BroadcastReciver (@override onRecive)

Reciver + IntenFilter



Notification



Notification

Para crear una notificación debo indicar:

Icono, título y texto

De forma opcional indicaré la Activity a la que dirigirse tras tocar la notificación

Notification

Además, puedo usar la clase Notification.Builder para componer mi notificación

Y emplear el servicio NotificationManager para lanzarla getSystemService (Context.NOTIFICATION_SERVICE);

PendingIntent

Es una especie de TOKEN (código de seguridad) que asocio a un Intent, para garantizarle los permisos necesarios para ejecutarse como si fuera yo (mi app)

EJ: Notificaciones

PendingIntent

En la práctica es útil/necesario cuando uso los servicios de Alarma y Notificación.

Ambos retomarán contacto con mi app, por medio del PendingIntent, lo que garantizará que tiene los permisos de mi app. (Manifest)

PendingIntent

Es como una forma de programar un retorno a mi app. Deberé usar getService(), getBroadCast() y getActivity(), para crear el PendingIntent correspondiente en caso de querer alcanzar un servicio, un receiver o una activity respectivamente

AlarmManager

Permite programar tareas y determinar así cuándo se lanzará un Intent, que a su vez lanzará a ejecución de otro componente (Actividad, Servicio o Reciver)

AlarmManager

AlarmManager es un servicio ofrecido por el SO. Por lo que para acceder a él, tan sólo debo ejecutar la instrucción siguiente

```
context.getSystemService(Context.ALARM_SERVICE);
```

AlarmManager

```
Intent intentAlarm = new Intent(context, AlarmaReciver.class);

AlarmManager alarmManager = (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);

PendingIntent pi = PendingIntent.getBroadcast(context, ID_PROCESO_ALARMA, intentAlarm, PendingIntent.FLAG_UPDATE_CURRENT);
alarmManager.set(AlarmManager.RTC_WAKEUP, tiempo, pi);
```

AlarmManager

Al setear la Alarma indico: Tipo, momento (ms) y proceso (PendingIntent) Tipo es una variable entera definida como constante y su rango puede ser:

ELAPSED_REALTIME

ELAPSED_REALTIME_WAKEUP

RTC

RTC_WAKEUP

AlarmManager

Al setear la Alarma indico: Tipo, momento (ms) y proceso (PendingIntent) Tipo es una variable entera definida como constante y su rango puede ser:

ELAPSED_REALTIME

ELAPSED_REALTIME_WAKEUP

RTC

RTC_WAKEUP

RemoteViews

RemoteViews es un clase que permite representar vistas desde un hilo distinto de la interfaz principal

Es el método ofrecido por Android para construir Notificaciones personalizadas y añadir a ellas un comportamiento dinámico (OnClick sobre sus elementos)

RemoteViews

1 Defino RemoteView (indicando su layout)

2 Asocio a cada elemento su PendingIntent (Sustituye a onClickListener)

3 Asocio el RemoteView a su Notificación (SetContent)

RemoteViews

1 Defino RemoteView (indicando su layout)

2 Asocio a cada elemento su PendingIntent (Sustituye a onClickListener)

3 Asocio el RemoteView a su Notificación (SetContent)

DreamServices

Android tiene una API que permite definir procesos a modo de SalvaPantallas.

Lo curioso de estos procesos es que al ser Servicios, no tienen acceso a la Interfaz gráfica de usuario de forma normal y a su vez, pueden prescindir de Actividad principal

DreamServices

```
<activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN"/>

<category android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
    </activity>
```

DreamServices

Para crear mi Salvapantallas y que aparezca luego como opción en Ajustes → Pantalla; debo crear una app, con un Servicio que herede de DreamService.

Ello me obligará a implementar dos métodos básicos

OnAttachedWindow (preparo)

OnDreamingStarted (se ejecuta)

DreamServices

Además, puedo crear una Actividad de ajustes, cuya única finalidad, será permiter la selección de unas preferencias por parte del usuario, para poder ser usadas en la fase de preparación del servicio.

DreamServices

Para permiter ajustes, indicaré en el Manifest la siguiente entrada

```
<meta-data
android:name="android.service.dream"
android:resource="@xml/my_dream" />
```

Siendo my_dream, un fichero cuya única entrada contiene la descripción de la clase principal

Animation

Para manejar la IU, no puedo acceder al layout del Servicio como hago con las actividades.

Debo definir animaciones (preferiblmente por comoidad mediante un XML) y asociarlas al Layout a aplicar de mi Servicio (elementos)

Animation

Hay 4 tipos de animaciones/clases

Transalate: para jugar con el movimiento

Rotate: para rotar un objeto

Alpha: para alterar el color

Scala: parajugar con el tamaño

Animation

Además, AnimationSet (<set>), me permite definir agrupaciones de animaciones, para que todas apliquen de forma conjunta a un mismo elemento gráfico / layout