Desarrollo Android

Arkaitz Garro

Master en desarrollo de Apps Móviles

Intent

Fundamentos

Intent: introducción

Un `Intent` es utilizado como mecanismo de paso de mensajes, entre aplicaciones o dentro de una misma app

Iniciar un servicio o actividad explícitamente, a través del nombre de su clase

Iniciar una actividad o servicio para realizar una acción con o sin datos

Indicar que un evento a ocurrido (Broadcast)

Las propias aplicaciones del sistema (Telefono, SMS...) responden a Intents concretos, que nosotros también podemos escuchar, y por tanto, reemplazar el comportamiento del sistema

Utilizar Intents, incluso dentro de la propia aplicación, es uno de los fundamentos de diseño de Android

Iniciar actividades

El uso más común de `Intent` es el de lanzar actividades

Para crear y mostrar una nueva actividad

```
startActivity(myIntent);
```

Éste método busca e inicia una actividad que mejor encaje con el `Intent`

El Intent puede especificar la clase de la actividad a iniciar, o una acción concreta, en cuyo caso será Android quien decida que actividad iniciar (`intent resolution`)

Iniciar actividades explicitamente

```
Intent intent = new Intent(MyActivity.this, MyOtherActivity.class);
startActivity(intent);

//startActivityForResult(intent);
```

Intent resolution

```
if (somethingWeird && itDontLookGood) {
    Intent intent =
        new Intent(Intent.ACTION_DIAL, Uri.parse(“tel:555-2368”));

    startActivity(intent);
}
```

Intent resolution

Android resuelve el `Intent` e inicia la actividad correspondiente

Cuando se construye el `Intent`, se indica la acción a realizar y opcionalmente unos datos asociados

¿Qué actividad se ejecuta, si hay varias que pueden hacerlo?

Nosotros solicitamos acciones, pero no especificamos la aplicación

En Android quien decide, pero en este caso, nos muestra (como usuarios) las opciones existentes

¿Y si no hay ninguna...?

Intent resolution

No hay ninguna garantía de que nuestro `Intent` sea procesado por alguna actividad

```
// Check if an Activity exists to perform this action.
PackageManager pm = getPackageManager();
ComponentName cn = intent.resolveActivity(pm);

if (cn == null) {
   Uri marketUri = Uri.parse(“market://search?q=pname:com.myapp.packagename”);
   Intent marketIntent = new Intent(Intent.ACTION_VIEW).setData(marketUri);

   // If the Google Play Store is available, use it to download an application
   // capable of performing the required action. Otherwise log an error.
   if (marketIntent.resolveActivity(pm) != null)
      startActivity(marketIntent);
   else
      Log.d(TAG, “Market client not available.”);
} else {
   startActivity(intent);
}
```

Intent: obtener resultados

Cuando una actividad es iniciada a través de `startActivity`, es un proceso independiente que no devuelve resultados cuando se termina

Cuando es necesario algún tipo de feedback, disponemos del método `startActivityForResult`

Su funcionamiento es exactamente igual a `startActivity`, con la diferencia que debemos indicar un `Request Code`, que será utilizado posteriormente para identificar la respuesta

Intent: obtener resultados

```
private static final int SHOW_SUBACTIVITY = 1;

private void startSubActivity() {
   Intent intent = new Intent(this, MyOtherActivity.class);
   startActivityForResult(intent, SHOW_SUBACTIVITY);
}
```

```
private static final int PICK_CONTACT_SUBACTIVITY = 2;

private void startSubActivityImplicitly() {
   Uri uri = Uri.parse(“content://contacts/people”);
   Intent intent = new Intent(Intent.ACTION_PICK, uri);
   startActivityForResult(intent, PICK_CONTACT_SUBACTIVITY);
}
```

Intent: devolver resultados

Cuando una actividad está lista para finalizar, debemos llamar a `setResult` para devolver los resultados

Este método toma dos parámetros, el código de resultado y los datos de respuesta

Por norma general, los datos son devueltos en forma de URI, indicando el contenido seleccionado. Es posible devolver una colección de datos extra en el Intent con `putExtra`

Si una actividad es cerrada (botón atrás), el código de resultado es `RESULT_CANCELED` y el Intent es `null`

```
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.selector_layout);

  final ListView listView = (ListView)findViewById(R.id.listView1);

  Button okButton = (Button) findViewById(R.id.ok_button);
  okButton.setOnClickListener(new View.OnClickListener() {
    public void onClick(View view) {
      long selected_horse_id = listView.getSelectedItemId();

      Uri selectedHorse = Uri.parse("content://horses/" + selected_horse_id);
      Intent result = new Intent(Intent.ACTION_PICK, selectedHorse);
      setResult(RESULT_OK, result);
      finish();
    }
  });

  Button cancelButton = (Button) findViewById(R.id.cancel_button);
  cancelButton.setOnClickListener(new View.OnClickListener() {
    public void onClick(View view) {
      setResult(RESULT_CANCELED);
      finish();
    }
  });
}
```

Intent: gestionar respuestas

Cuando una sub-actividad se cierra, se ejecuta un evento en la actividad lanzadora, donde se procesa la respuesta

Este método, `onActivityResult`, acepta tres parámetros:

`requestCode`: código utilizado para lanzar la actividad.

`resultCode`: indica cómo ha finalizado la sub-actividad. **`Activity.RESULT_OK`** o **`Activity.RESULT_CANCELED`**

`data`: **`Intent`** que contiene los datos de respuesta. Puede incluir un URI con el contenido seleccionado, y datos extra.

```
private static final int SELECT_HORSE = 1;
  private static final int SELECT_GUN = 2;

  Uri selectedHorse = null;
  Uri selectedGun = null;

  @Override
  public void onActivityResult(int requestCode,
                               int resultCode,
                               Intent data) {

    super.onActivityResult(requestCode, resultCode, data);

    switch(requestCode) {
      case (SELECT_HORSE):
        if (resultCode == Activity.RESULT_OK)
          selectedHorse = data.getData();
        break;

      case (SELECT_GUN):
        if (resultCode == Activity.RESULT_OK)
          selectedGun = data.getData();
        break;

      default: break;
    }
}
```

Intent: acciones nativas

Las aplicaciones nativas, también utilizan Intents para lanzar actividades

Algunas acciones predefinidas...

http://developer.android.com/reference/android/content/Intent.html#constants

```
if (somethingWeird && itDontLookGood) {
    Intent intent =
        new Intent(Intent.ACTION_DIAL, Uri.parse(“tel:555-2368”));

    startActivity(intent);
}
```

Intent Resolution: proceso

1. Android lista todos los Intent Filters disponibles en los paquetes instalados.
2. Los filtros que no cumplen las condiciones definidas en la acción o categorías (todas), son eliminados de la lista.
3. Todos los datos incluidos en el Intent, son comparados con los datos definidos en el filtro (`mimetype, scheme, host, path`).
4. Cuando se inicia una actividad de manera implícita, y más de un componente cumple las condiciones, todas las posibilidades son ofrecidas al usuario.
    * En el caso de los Broadcast Receivers, cada uno de ellos recibe el Intent.

Intent Resolution: obtener Intent

Cuando una actividad es iniciada de manera implícita, es necesario obtener la acción y los datos sobre los que trabajar

```
@Override
public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.main);

    Intent intent = getIntent();
    String action = intent.getAction();
    Uri data = intent.getData();
}
```

Intent Resolution: obtener Intent

En algunos casos, la actividad va a seguir recibiendo Intents una vez creada, por lo que el anterior método no es suficiente

```
@Override
public void onNewIntent(Intent newIntent) {
   // TODO React to the new Intent
   super.onNewIntent(newIntent);
}
```

Ejercicio

Demo

A fatal error 0E has ocurred at 0028:C0011E36 in VXD VMM(01) +
00010E36. The current application will be terminated.

* Press any key to terminate the current application.
* Press CTRL+ALT+DEL again to restart your computer. You wil lose any
  unsaved information in all your applications.

Press any key to continue _

Broadcast events

Fundamentos

Broadcast events

Al igual que podemos escuchar a eventos de Broadcast, podemos enviarlos de una forma anónima

Esto permite crear aplicaciones totalmente desacopladas

Android utiliza este sistema de mensajes de manera extensiva

Broadcast: lanzar eventos

Tan sencillo como construir un Intent y utilizar el método `sendBroadcast` para enviarlo

```
private IntentFilter filter =
      new IntentFilter(LifeformDetectedReceiver.NEW_LIFEFORM);

Intent intent = new Intent(LifeformDetectedReceiver.NEW_LIFEFORM);
intent.putExtra(LifeformDetectedReceiver.EXTRA_LIFEFORM_NAME,
                detectedLifeform);
intent.putExtra(LifeformDetectedReceiver.EXTRA_LONGITUDE,
                currentLongitude);
intent.putExtra(LifeformDetectedReceiver.EXTRA_LATITUDE,
                currentLatitude);

sendBroadcast(intent);
```

Broadcast Receivers

Son necesarios para escuchar los Broadcast, y actuar en consecuencia

Es necesario que estén registrados en el manifiesto, especificando que tipo de Intents escucha (Intent Filter)

No es necesario que la aplicación esté en ejecución para reaccionar ante los Intents

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

public class MyBroadcastReceiver extends BroadcastReceiver {
   @Override
   public void onReceive(Context context, Intent intent) {
   //TODO: React to the Intent received.
   }
}
```

```
public class LifeformDetectedReceiver extends BroadcastReceiver {
  public final static String EXTRA_LIFEFORM_NAME = "EXTRA_LIFEFORM_NAME";
  public final static String EXTRA_LATITUDE = "EXTRA_LATITUDE";
  public final static String EXTRA_LONGITUDE = "EXTRA_LONGITUDE";

  public static final String
    ACTION_BURN = "com.paad.alien.action.BURN_IT_WITH_FIRE";

  public static final String
    NEW_LIFEFORM = "com.paad.alien.action.NEW_LIFEFORM";

  @Override
  public void onReceive(Context context, Intent intent) {
    Uri data = intent.getData();
    String type = intent.getStringExtra(EXTRA_LIFEFORM_NAME);
    double lat = intent.getDoubleExtra(EXTRA_LATITUDE, 0);
    double lng = intent.getDoubleExtra(EXTRA_LONGITUDE, 0);
    Location loc = new Location("gps");
    loc.setLatitude(lat);
    loc.setLongitude(lng);
    if (type.equals("facehugger")) {
      Intent startIntent = new Intent(ACTION_BURN, data);
      startIntent.putExtra(EXTRA_LATITUDE, lat);
      startIntent.putExtra(EXTRA_LONGITUDE, lng);

      context.startService(startIntent);
    }
  }
}
```

Broadcast Receivers

Los Bradcast Receivers que interactúan con la UI, generalmente son registrados en el código

Estos únicamente van a responder a eventos lanzados por componentes registrados dentro de la misma aplicación

Broadcast Receivers

```
private IntentFilter filter =
  new IntentFilter(LifeformDetectedReceiver.NEW_LIFEFORM);

private LifeformDetectedReceiver receiver =
  new LifeformDetectedReceiver();

@Override
public void onResume() {
  super.onResume();

  // Register the broadcast receiver.
  registerReceiver(receiver, filter);
}

@Override
  public void onPause() {
    // Unregister the receiver
    unregisterReceiver(receiver);

    super.onPause();
  }
```

Broadcast Intents ordenados

Cuando el orden en el que se procesan los Intents es importante, es posible utilizar el método `sendOrderedBroadcast`

Utilizando este método, los Intents son enviados según el orden de prioridad indicado en el manifiesto

```
<receiver
   android:name=”.MyOrderedReceiver”
   android:permission=”com.paad.MY_BROADCAST_PERMISSION”>
   <intent-filter
      android:priority=”100”>
      <action android:name=”com.paad.action.ORDERED_BROADCAST” />
   </intent-filter>
</receiver>
```

Sticky Intents

Variación de Intents que mantienen el último valor asociado a su último Broadcast

Éste valor almacenado es enviado cada vez que un nuevo Broadcast Receiver es registrado

La gran mayoría de los estados del sistema utilizan este tipo de Intents, ya mejora la eficiencia del paso de mensajes

```
IntentFilter battery = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
Intent currentBatteryCharge = registerReceiver(null, battery);
```

Local Broadcast Manager

Fue creado para facilitar el proceso de creación, registro y envío Intents dentro de la propia aplicación

Su visibilidad es mucho más reducida, y por lo tanto, más eficiente

Además, los Intents no son enviados al exterior, por lo que no existe riesgo de revelar información sensible

Igualmente nuestra aplicación no puede recibir Intents del exterior

Local Broadcast Manager

```
LocalBroadcastManager lbm = LocalBroadcastManager.getInstance(this);


lbm.registerReceiver(new BroadcastReceiver() {
   @Override
   public void onReceive(Context context, Intent intent) {
      // TODO Handle the received local broadcast
   }
}, new IntentFilter(LOCAL_ACTION));



lbm.sendBroadcast(new Intent(LOCAL_ACTION));
```

Intent Filters

Es importante definir las acciones y datos que los componentes de la aplicación son capaces de gestionar

Al registrar una actividad, servicio o receiver (como potencial gestor de Intents) es importante restringir su visibilidad, si es necesario

Tenemos la posibilidad de utilizar los siguientes filtros

Intent Filters

`action`: utilizar el atributo **`android:name`** para especificar el nombre de la acción a servir. Todos los filtros deben contener al menos una etiqueta de acción.

`category`: utilizar el atributo **`android:name`** para describir en qué circunstancias la acción debe ser ejecutada. Se pueden incluir tantas como sean necesarias.

`DEFAULT`: acción principal para el tipo de datos definido.

`LAUNCHER`: la actividad aparece en el lanzador de aplicaciones.

`HOME`: alternativa a la pantalla de inicio principal.

`ALTERNATIVE`: alternativa a la acción principal.

`BROWSABLE`: ejecutar la acción dentro del navegador.

Intent Filters

`data`: especificar el tipo de datos compatibles con el componente.

`android:host`: nombre de host válido.

`android:mimetype`: tipos de datos.

`android:path`: path válido apara el URI.

`android:port`: un puerto válido para el host.

`android:scheme`: requiere un esquema en particular.

Intent Filters

```
<intent-filter>
   <action
      android:name=”com.paad.earthquake.intent.action.SHOW_DAMAGE” />
   <category android:name=”android.intent.category.DEFAULT”/>
   <category
      android:name=”android.intent.category.SELECTED_ALTERNATIVE”/>
   <data android:mimeType=”vnd.earthquake.cursor.item/*”/>
</intent-filter>
```

```
<activity android:name=”.MyBlogViewerActivity”>
   <intent-filter>
      <action android:name=”android.intent.action.VIEW” />
      <category android:name=”android.intent.category.DEFAULT” />
      <category android:name=”android.intent.category.BROWSABLE” />
      <data android:scheme=”http”
      android:host=”blog.radioactiveyak.com”/>
   </intent-filter>
</activity>
```

Native Broadcast Intents

Muchos de los servicios del sistema lanzan eventos cuando se producen cambios

`ACTION_BOOT_COMPLETED`: el dispositivo ha completado la secuencia de arranque.

`ACTION_CAMERA_BUTTON`: el botón de la cámara es pulsado.

`ACTION_MEDIA_EJECT`: el usuario ha decidido desconectar el almacenamiento externo (antes de completarse).

`ACTION_MEDIA_MOUNTED, ACTION_MEDIA_UNMOUNTED.`

`ACTION_SCREEN_ON, ACTION_SCREEN_OFF.`

Ejercicio:

Monitorizar eventos del sistema