Lezione 23 Intent Filter, Broadcast Event, Mappe, Geocoding, Servizi Location-Based

Intent Filter

Se l'intento una volta creato dal mittente deve essere ricevuto dal destinatario. Il destinatario deve indicare al sistema Android che è in grado di ricevere uno o più intenti. Usando gli Intent Filter la nostra applicazione può indicare al sistema Android a quali Intenti vuole/può rispondere.

```
<activity android:name=".CrashViewer" android:label="Crash View">
<intent-filter>
<action android:name="it.unive.dsi.intent.action.SHOW_CRASH"></action>
<category android:name="android.intent.category.DEFAULT"/>
<category
android:name="android.intent.category.ALTERNATIVE_SELECTED"/>
<data android:mimeType="vnd.earthquake.cursor.item/*"/>
</intent-filter>
</activity>
```

action: descrive il tipo di azione. Alcune azioni del sistema android sono:

- ACTION_ANSWER Inzia un'attività che risponde a una chimata telefonica;
- ACTION_CALL inizia una chiamanta corrispondente al numero descritto dall'Uri. (meglio usare ACTION_DIAL)
- ACTION_DELETE Inizia un'attività per eliminare il dato specificato nell'Uri (per esempio eliminare una ruga da un content provider);
- ACTION_DIAL effettua una chimata al numero specificato nell'Uri.
- ACTION_EDIT Inizia un'attività per editate il dato specificato nell'Uri.
- ACTION_INSERT Inizia un'attività per inserire un elemento nel Cursor specificato nell'Uri dell'intento. L'attività chiamata dovrebbe restituire un intento in cui l'Uri specifica l'elemento inserito.
- ACTION_PICK Inizia un'attività che permette di scegliere un elemento da il content provider specificato nel Uri. L'attività chiamanta dovrebbe ritornare un intento con l'Uri dell'elemento selezionato.
- ACTION_SEARCH Inizia un'attività che effettua la ricerca del termine passato su SearchManager.QUERY.
- ACTION_SENDTO Inizia un'attività per inviare un messaggio al contatto specificato nell'Uri.
- ACTION_SEND Inizia un'attività che invia i dati specificati nell'Intent. Il destinatario sarà individuato dalla nuova attività.
- ACTION_VIEW Inizia un'attività che visualizza il dato specificato nell'Uri.

- ACTION_WEB_SEARCH Inizia un'attività che effettua una ricerca sul Web del dato passato nell'Uri.

category l'attibuto android:name specifia sotto che circostanze un'azione sarà servita. Le categorie specificate possono essere più di una. Le categorie possono essere create dal programmatore oppure è possibile usare quelle di Android che sono:

- ALTERNATIVE Le azioni alternative all'azione standard sull'oggetto.
- SELECTED_ALTERNATIVE: Simile alla categoria ALTERNATIVE ma verrà risolta in una singola selezione
- BROWSABLE: specifica azioni disponibili dal Browser
- DEFAULT Per selezionare l'azione di default su un componente e per poter usare gli intenti espliciti.
- GADGET si specifica un'attività che può essere inclusa in un'altra attività.
- HOME Specificando questa categoria e non specificando l'azione, si propone un'alternativa allo schermo home nativo.
- LAUNCHER si specifica un'attività che può essere eseguita dal launcher del sistema Android.

data Questo tag permette di specificare che tipi di dati il componente può manipolare. Si possono specificare più tag di questo titpo. Sintassi: (<scheme>://<host>:<port>/<path>)

- android:scheme: uno schema (esempio: content or http).
- android:host: un hostname valido (es. google.com).
- android:port: la porta dell'host.
- android:path specifica un path dell'Uri (es. /transport/boats/).
- android:mimetype: Permette di specificare un dipo di dati che il componente è in grado di gestire. Per esempio <type android:value="vnd.android.cursor.dir/*"/> indica che il componente è in grado di gestire ogni Cursore di Android (per la definizione di Cursore, vedere i Content Provider).

Broadcast Event

Se un intento è pensato per attivare una singola attività, i broadcast event sono pensati per attivare molte attività contemporaneamente. Alcuni Broadcast Event del sistema Android sono:

`ACTION_BATTERY_LOW:` Batteria scarica.

`ACTION_HEADSET_PLUG` : Cuffiette collegate o scollegate dal terminale.

`ACTION_SCREEN_ON:` Lo schermo è stato acceso.

`ACTION_TIMEZONE_CHANGED:` Il fuso orario è stato cambiato.

Esempio di creazione di un nuovo Broadcast Event (CRASH_DETECTED):

```
Package it.unive.dsi.crash;
// l'attività o servizio che invia l'evento
public class Crash extend Activity{

final public static String
CRASH_DETECTED=”it.unive.dsi.action.CRASH_DETECTED”;

...
Intent intent = new Intent(CRASH_DETECTED);
//aggiungo alcuni parametri che descrivono l'evento
intent.putExtra("crashType", “car”);
intent.putExtra("level", 3);
intent.putExtra("description", “...”);
//invio l'evento a tutti gli ascolatori (BroadcastReceiver)
sendBroadcast(intent);
```

Broadcast Receiver

Ovviamente se non ci sono ascoltatori l'evento non genera nessun effetto. Per definire un BroadcastReceiver bisogna:

1) estendere la classe BroadcastReceiver e sovrascrivere il metodo onReceive. Es.:

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

2) poi bisogna registrare su Android il BroadcastReceiver ovvero aggingere nel manifesto dell'applicazione che contiene il BroadcastReceiver che c'é un receiver e che eventi gestisce. Es.

```
<receiver android:name=".CrashBroadcastReceiver">
<intent-filter>
<action android:name="it.unive.dsi.Crash.CRASH_DETECTED"/>
</intent-filter>
</receiver>
```

Per i BroadcastReceiver è anche possibile creare e registrare il receiver dinamicamente (e quindi non necessariamente per tutto il tempo di vita dell'applicazione il Broadcast Receiver sarà attivo):

```
// Create and register the broadcast receiver.
IntentFilter filter = new
IntentFilter(it.unive.dsi.Crash.CRASH_DETECTED);
CrashBroadcastReceiver receiver = new CrashBroadcastReceiver();
registerReceiver(receiver, filter);
...
unregisterReceiver(receiver);
```

Una delle caratteristiche più interessanti delle applicazioni sviluppate su Android è la possibilità di

integrare facilmente posizionamento GPS, Mappe, geolocalizzazione etc.

Geolocalizzazione

Come configurare l'emulatore

Per poter verificare con il simulatore il cambiamenti di posizione del dispositivo mobile, bisogna configurare l'emulatore tramite la funzionalità “Location controls”.

Ci sono due modalità per indicare al simulatore le posizioni da usare durante la simulazione:

1. tramite file KML (generato by Google Earth);
2. tramite file GPX (generato da dispositivi GPS);
3. manualmente inviando una posizione alle volta.

Selezionare il tipo di localizzatore

Un dispositivo mobile può essere fornito di più dispositivi di localizzazione. Solitamente i due più usati sono:

1. tramite GPS
2. tramite Rete (GSM o WIFI).

Ogni tipo di localizzatore ha le sue peculiarità (precisione della localizzazione, consumo, costo etc.). Quindi per qualche applicazione può bastare un precisione bassa ma richiedere bassi consumi, in altre sarà necessario una localizzazione precisa e magari si è disposti a sopportare consumi o costi più alti.

Le API Android permettono di gestire diversi tipi di localizzatori tramite diversi “Location Provider”. Per gestire i diversi Location Provider si utilizza il “Location Manager” attraverso la classe “LocationManager.

Accedere Location Manager

Prima di iniziare a fare qualsiasi operazione con i Location Provider bisogna recuperare l'oggetto Location Manager nel seguente modo:

```
String serviceString = Context.LOCATION_SERVICE;
LocationManager locationManager;
locationManager = (LocationManager)getSystemService(serviceString);
```

Selezionare un Location Provider

Il Location Manager prevede varie modalità per selezionare un Location Provider. La più semplice è quella che prevede che il programmatoe già conosca il Location Provider più adatto e che quindi usi uno dei seguenti nomi di Location Provider:

```
LocationManager.GPS_PROVIDER
```

```
LocationManager.NETWORK_PROVIDER
```

Oppure possiamo richiedere tramite il Location Manager l'elenco di tutti i Location Provider presenti nel dispositivo tramite il metodo statico getProviders (l'argomento permette di specificare se devono venire ritornati solo i Location Manager attivati oppure anche quelli presenti ma non attivi).

```
List<String> providers = locationManager.getProviders(enabledOnly);

//select the provider based on name
```

Alternativamente il Location Manager metta a disposizione la possibilità di cercare il Location Provider che più si avvicina alle esigenze dell'applicazione e/o utente.

```
Criteria criteria = new Criteria();
criteria.setAccuracy(Criteria.ACCURACY_COARSE);
criteria.setPowerRequirement(Criteria.POWER_LOW);
criteria.setAltitudeRequired(false);
criteria.setBearingRequired(false);
criteria.setSpeedRequired(false);
criteria.setCostAllowed(true);

String bestProvider = locationManager.getBestProvider(criteria, true);
```

o alternativamente

```
List<String> matchingProviders =
locationManager.getProviders(criteria,
false);
```

Nel primo caso (getBestProvider) nel caso più provider soddisfino i criteri, viene scelto quello con precisione maggionre. Nel caso nessuno soddisfi i criteri, vengono rilassati nell'ordine i criteri:

1. consumo
2. precisione
3. possibilità di ritornare orientamento, velocità e altitudine.

Recupera la posizione

Per recuperare la posizione del device fornita da un Location Provider bisogna usare il metodo `getLastKnonwPosition` dell'oggetto location provider:

```
String provider = ...vedi sopra...
Location location = locationManager.getLastKnownLocation(provider);
```

Per invocare il metodo `getLastKnownLocation` bisogna che il manifesto dell'applicazione richieda il permesso di accedere alla posizione del device. Il permesso da richiedere può essere per la posizione precisa (fine) oppure approssimata (coarse). Il GPS fornisce una posizione precisa mentre la rete GSM fornisce una posizione approssimata. Ecco i due esempi:

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

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

Anche se l'applicazione deve per forza richiedere i permessi per accedere alla posizione del device e' inoltre il caso di osservare le seguenti regole per preservare la privacy dell'utente:

1. Tracciare la locazione solo quando indispensabile all'applicazione;
2. Avvertire l'utente quando si sta tracciando la posizione e come tale posizione viene usate e memorizzata;
3. Permettere all'utente di disabilitare gli aggiornamenti della posizione.

La posizione ritornata può non essere aggiornata (vedi sotto)

L'oggetto `Location` ritornato include tutte le informazioni disponibili quando di utilizza quel dato Location Provider. Esempi di informazioni disponibili sono: latitudine, longitudine, orientamento, altitudine, velocità e tempo in cui è stata presa la posizione. Tutti queste informazioni sono disponibili tramite metodi get. Eventuali altre informazioni sono disponibili tramite l'oggetto `Bundle`.

Esempio

Aggiungiamo al file manifest.xml la richiesta del permesso di accedere alla posiozione accurata del device.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="it.unive.dsi.android">
<application
android:icon="@drawable/icon">
<activity
android:name=".GeoApp"
android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
</manifest>
```

Modifichiamo il Layout main.xml in modo da aggiungere un attributo android:ID al controllo TextView control in modo da poter accedervi dall'attivita'.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
<TextView
android:id="@+id/myLocationText"
```

```
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/hello"
/>
</LinearLayout>
```

Sovrascriviamo il metodo onCreate dell'attivita' GeoApp:

```
package it.unive.dsi.android;
import android.app.Activity;
import android.content.Context;
import android.location.Location;
import android.location.LocationManager;
import android.os.Bundle;
import android.widget.TextView;
public class GeoApp extends Activity {
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
LocationManager locationManager;
String context = Context.LOCATION_SERVICE;
locationManager = (LocationManager)getSystemService(context);
String provider = LocationManager.GPS_PROVIDER;
Location location =
locationManager.getLastKnownLocation(provider);
updateWithNewLocation(location);
}

private void updateWithNewLocation(Location location) {
String desc;
TextView myLocationText;
myLocationText = (TextView)findViewById(R.id.myLocationText);
if (location != null) {
desc=”Lat: “ + location.getLatitude(); //double
desc+= “Long:” + location.getLongitude(); //double
desc+=”Time: “+location.getTime(); //long ms dal 1970

if (location.hasAccuracy())
desc+=”Acc: ”+location.getAccuracy(); //in metri
if (location.hasAltitude())
desc+=”Alt: “+ location.getAltitude(); //in metri WGS84
if (location.hasBearing())
desc+=”Dir: “+location.getBearing(); //in gradi (Nord=0)
if (location.hasSpeed())
desc+=”Vel: “+location.getSpeed(); //in metri al secondo

Bundle b = location.getExtras();
if (b!=null)
for (String key:b.keySet()){
desc+= key + b.get(key); // solo satellites
}
} else {
latLongString = "No location found";
}
```

```
myLocationText.setText(latLongString);
}
}
```

L'applicazione appena vista non aggiorna la nuova posizione in caso di aggiornamenti e in realta' non visualizza la posizione corretta se qualche altra applicazione non ha gia' richiesto la posizione. Per avere un'attivita' che aggiorna la posizione visualizzata al cambiamento di posizone del device, bisogna usare un LocationListener e registrarlo attraverso il metodo requestLocationUpdates.

Tale metodo ha come parametri 1) il nome del Location Provider (o un insieme di criteri per individuare il provider da usare) 2) il tempo minimo che deve passare tra due aggiornamenti (per ridurre il consumo di batterie del dispositivo GPS e processire) 3) la distanza minima tra due aggiornamenti (come sopra) e 4) l'oggetto Listener.

Esempio:

```
String provider = LocationManager.GPS_PROVIDER;
int t = 5000; // ms
int distance = 5; // metri
LocationListener myLocationListener = new LocationListener() {
public void onLocationChanged(Location location) {
updateWithNewLocation(location);
}
public void onProviderDisabled(String provider){
updateWithNewLocation(null);
}
public void onProviderEnabled(String provider){
...
}
public void onStatusChanged(String provider, int status,
Bundle extras){
...
}
};
locationManager.requestLocationUpdates(provider, t, distance,
myLocationListener);
```

Quando il tempo minimo e la distanza minima sono stati superati, il relativo evento invochera' il metodo `onLocationChanged`.

Per fermare gli aggiornamenti della posizione e' possibile chiamare il metodo `removeUpdates`.

Dato che i moduli GPS consumano molta potenza, per minimizzare tale consumo bisognerebbe disabilitare gli aggiornamenti della posizione ogniqulvolta non necessario. In particolar modo quando l'applicazione non e' visibile e gli aggiornamenti di posizione servono solo per aggiornare l'interfaccia utente.

Proximity Alerts

I LocationListener reagiscono a ogni cambiamento di posizione (filtrato solo per tempo minimo e distanza). Talvolta c'e' la necessita' che l'applicazione reagisca solo quando il device entra o esca da una specifica locazione. I Proximity Alerts attivano l'applicazione solo quando il device entra o esce da una particolare locazione. Internamente, Android puo' usare differenti Provider in dipendenza di quanto vicino si trola alla locazione da monitorare e in questo modo e' possibile risparmiare energia.

Per impostare un Proximity Alert bisognaselezionare il centro (usando longitudine e latudine), un raggio

attorno al centro e una scadenza temporale dell'alert. L'alert verra' inviato sia che il device entri nel cerchio che ne esca. Quando si attiva un proximity alert, questo fa scattare degli Intenti e molto comunemente dei Broadcast Intent. Per specificare l'intento da scattare, si usa la classe PendingIntent che incorpora un Intento nel seguente modo:

```
Intent intent = new Intent(MY_ACTION);
PendingIntent pendingIntent = PendingIntent.getBroadcast(this, -1,
intent, 0);
```

Il prossimo esempio, imposta un proximity alert che non scade mai e che viene fatto scattare quando il device si allontana piu' di 10 metri dall'obiettivo.

```
private static String
DSI_PROXIMITY_ALERT = "it.unive.dsi.android.alert";
private void setProximityAlert() {
String locService = Context.LOCATION_SERVICE;
LocationManager locationManager;
locationManager = (LocationManager)getSystemService(locService);
double lat = 45.479272;
double lng = 12.255374;
float r = 100f; // raggio in metri
long time= -1; // non scade
Intent int = new Intent(DSI_PROXIMITY_ALERT);
PendingIntent proIntent = PendingIntent.getBroadcast(this, -1,int,0);
locationManager.addProximityAlert(lat, lng, r, time, proIntent);
}
```

Quando il LocationManager rileva che il device ha superato la circonferenza (verso il centro o verso l'esterno), l'intento incapsulato nel PendingIntent viene fatto scattare e la chiave `LocationManager.KEY_PROXIMITY_ENTERING` impostata a vero o falso in relazione alla direzione (entrata=true, uscita=false).

Per gestire i proximity alert bisogna creare un BroadcastReceiver del tipo:

```
public class ProximityIntentReceiver extends BroadcastReceiver {
@Override
public void onReceive (Context context, Intent intent) {
String key = LocationManager.KEY_PROXIMITY_ENTERING;
Boolean entering = intent.getBooleanExtra(key, false);
//effettuare le azioni opportune
}
}
```

Per cominciare a ricevere gli alert bisogna registrare il BroadcastReceiver nel seguente modo:

```
IntentFilter filter = new IntentFilter(DSI_PROXIMITY_ALERT);
registerReceiver(new ProximityIntentReceiver(), filter);
```