Oltre gli intents: Intent filter e Broadcast receiver

Programmazione Mobile

A.A. 2021/22

M.O. Spata

Intent filter

- Gli **intent filter** sono utilizzati per registrare Activity, Services e Broadcast Receivers dichiarando l'azione che sono in grado di eseguire su un particolare tipo di dati.
  Possono inoltre essere utilizzati per registrare l'interesse di Broadcast Receivers nei confronti di Broadcast Intent relativi a determinate azioni e/o eventi.
- Utilizzando **Intent Filter** i diversi componenti di una applicazione possono dichiarare che sono in grado di rispondere alle richieste di una o più azioni da parte di qualsiasi applicazione installata sul dispositivo.
- Per registrare un componente di un'applicazione come potenziale gestore di intent, si aggiunge un tag ***intent-filter*** al nodo manifest del componente, utilizzando i seguenti tag (e relativi attributi):

Tag per registrare un app ad un intent

- **action**: si utilizza l'attributo ***android:name*** per specificare il nome dell'azione da realizzare. Ogni intent filter deve avere almeno un *tag action*.
- **category**: si utilizza l'attributo ***android:name*** per specificare in quali circostanze l'azione dovrebbe essere eseguita.

Tag category: ALTERNATIVE

- Ogni tag intent filter può includere diversi tag category. È possibile specificare proprie category o utilizzare i valori standard forniti da Android e qui elencati:
  - **ALTERNATIVE** Questa categoria specifica che la relativa azione dovrebbe essere disponibile come alternativa alla azione eseguita di default su un elemento del tipo di dati specificato. Per esempio, dove l'azione predefinita per un contatto è quello di vederlo, l'alternativa potrebbe essere per modificarlo.

Tag category: SELECTED_ALTERNATIVE

- Ogni tag intent filter può includere diversi tag category. È possibile specificare proprie category o utilizzare i valori standard forniti da Android e qui elencati:
  - **SELECTED_ALTERNATIVE** Simile alla categoria ALTERNATIVE, ma tale categoria sarà sempre ridotta ad una singola azione utilizzando il processo intent resolver; SELECTED_ALTERNATIVE viene utilizzato quando un elenco di possibilità è richiesto.

Tag category: BROWSABLE

- Ogni tag intent filter può includere diversi tag category. È possibile specificare proprie category o utilizzare i valori standard forniti da Android e qui elencati:
  - **BROWSABLE** Specifica un 'azione disponibile all'interno del browser. Quando un Intent è richiamato dall'interno del browser includerà sempre la category BROWSABLE. Se si desidera che un'applicazione risponda alle azioni attivate all'interno del browser (ad esempio, intercettare i link ad un particolare sito web), è necessario includere tale categoria.

Tag category: DEFAULT

- Ogni tag intent filter può includere diversi tag category. È possibile specificare proprie category o utilizzare i valori standard forniti da Android e qui elencati:
  - **DEFAULT** Impostare questa category per impostare l'esecuzione di un componente come l'azione predefinita per il tipo di dati specificato nell'Intent filter. **E' necessaria anche per le attività che vengono lanciate con intent espliciti.**

Tag category: GADGET

- Ogni tag intent filter può includere diversi tag category. È possibile specificare proprie category o utilizzare i valori standard forniti da Android e qui elencati:
  - **GADGET** Impostando la category gadget si specifica che tale attività può funzionare anche se inserita all'interno di un'altra attività.

Tag category: HOME

- Ogni tag intent filter può includere diversi tag category. È possibile specificare proprie category o utilizzare i valori standard forniti da Android e qui elencati:
  - **HOME** Impostando al category home senza specificare un'azione, di fatta si indica una alternativa all'home screen del sistema.

Tag category: LAUNCHER

- Ogni tag intent filter può includere diversi tag category. È possibile specificare proprie category o utilizzare i valori standard forniti da Android e qui elencati:
  - **LUNCHER** utilizzando questa categoria si fa in modo che una aactivity figuri nell'application luncher.

Tag data

- **data**: Il tag *data* consente di specificare su quali tipi di dati il componente può agire; è possibile siano presenti diversi tag data a seconda dei casi. Per specificare i dati che il componente supporta è possibile utilizzare qualsiasi combinazione dei seguenti attributi:
  - **android:host**: Specifica un hostname valido (ad esempio google.com).
  - **android:mimetype**: Consente di specificare il tipo di dati che il componente è in grado di manipolare.
  - **android:path** Specifica un ''percorso'' valido per l'URI
  - **android:port** specifica le porte valide per l'host indicato
  - **android:scheme** richiede uno schema particolare (ad esempio, content o http).

```
<activity android:name=".EarthquakeDamageViewer" android:label="View Damage">
  <intent-filter>
    <action android:name="com.paad.earthquake.intent.action.SHOW_DAMAGE"></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>
```

Activity ed Intent resolution

- Quando si utilizza **startActivity** con avvio implicito, l'intent indicato come argomento della funzione di solito si risolve in una singola attività.
- Se ci sono molteplici activity in grado di eseguire l'azione indicata sui dati forniti, l'utente sarà invitato a scegliere fra più alternative.
- L'**intent resolution** è il processo che ha l'obiettivo di trovare l'intent filter più adatto.

Intent resolution: passo 1

L’intent resolution avviene mediante il seguente processo:

1. Android mette insieme un elenco di tutti gli intent filter resi disponibili dai pacchetti installati.

Intent resolution: passo 2

L’intent resolution avviene mediante il seguente processo:

2. Gli intent filter che non corrispondono con l’action o la category associata all'intent in procinto di essere risolto vengono rimossi dalla lista.

   2.1. Si avrà corrispondenza di action se l'intent filter include l'azione specificata oppure non ha alcuna action definita. Non ci sarà corrispondenza solo nel caso in cui nell'intent filter ci saranno una o più action definite, e nessuna di esse corrisponde a quella specificata dall'intent.

   2.2. La corrispondenza tra le category è più rigorosa. Gli intent filter devono includere tutte le category definite nell'intent da risolvere.

Intent resolution: passo 3

- 3. Infine, ogni parte dei dati URI dell'intent viene confrontata con il tag data dell'intent filter. Eventuali disallineamenti provocheranno la rimozione dell'intent filter dalla lista. Se non viene specificato alcun valore di dati nell'intent filter provocherà corrispondenza con tutti i valori dati dell'intent.

Intent resolution: passo 4

- 4. Quando si inizia un'activity implicitamente, se più di un componente viene risolto da questo processo sarà l'utente a dover scegliere.

Priorità activity

- I componenti delle applicazioni Android native prendono parte al processo di risoluzione esattamente nello stesso modo di quelli delle applicazioni terze parti.
- Non hanno una priorità più alta, e possono essere completamente sostituiti da nuove activity che dichiarano intent filter per l'esecuzione delle stesse azioni.

Dati di una Intent

- l componente di un'applicazione avviata attraverso un intent implicito, ha bisogno di trovare l'action richiesta ed i dati sui quali operare.
- Per estrarre l'intent si utilizza il metodo **getIntent** - di solito all'interno del metodo onCreate
- Per estrarre le informazioni necessarie si utilizzano le funzioni **getData** e **getAction** per trovare rispettivamente i dati e la action associati all'intent. Inoltre si utilizza il metodo **get<type>Extra** per estrarre eventuali dati extra memorizzati nell'intent.

Esempio

- Nel caso in cui si voglia delegare l'esecuzione dell'action richiesta al componente che segue nella lista stilata dal intent resolution process si può utilizzare il metodo: **startNextMatchingActivity(intent);**
- Ciò permette di aggiungere ulteriori condizioni per restringere l'uso dell'applicazione oltre a quelle imposte dagli intent filter

```
@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 intent = getIntent();
if (isDuringBreak)
  startNextMatchingActivity(intent);
```

Intent filters e menu

- Android permette di estendere applicazioni esistenti con funzionalità aggiunte attraverso nuovi componenti e pacchetti; utilizzando Intent Filter si possono infatti popolare i menu in modo dinamico in fase di esecuzione
- Il metodo ***addIntentOptions*** disponibile nella classe *menu* consente di specificare un **intent** che descrive i dati su cui si deve agire attraverso la voce di menu. Android risolve questo **intent** e restituisce tutte le **action** specificate negli Intent Filter che corrispondono ai dati specificati.

Intent filters e menu

- Verrà creata una nuova voce di menu per ogni *action*; il testo sarà determinato dalla etichetta dei corrispondenti intent filter.
- Questo meccanismo può essere altresì usato per fornire **action** anonimamente ad applicazioni già esistenti.
  - Basterà utilizzare opportunamente gli intent filter che si devono dichiarare all'interno del manifest dell'applicazione descrivendo le *action* (le azioni in grado di fare) e i *data* (i dati su cui fare le azioni).
  - Deve essere inoltre specificata la *category* che in questo caso può essere una qualsiasi combinazione di ALTERNATIVE e/o SELECTED_ALTERNATIVE.
  - Inoltre è bene specificare una etichetta *android:label*, il cui valore verrà visualizzato come voce di menu.

Esempio

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="edu.lap2.altraactivity"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".AltraActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />

            </intent-filter>
            <intent-filter android:label="menuAltraActivity">
                <action android:name="android.intent.action.VIEW" />
                <category android:name="android.intent.category.ALTERNATIVE"></category>
                <category android:name="android.intent.category.SELECTED_ALTERNATIVE"></category>
                <data android:mimeType="vnd.android.cursor.dir/person"></data>
            </intent-filter>

        </activity>

    </application>
    <uses-sdk android:minSdkVersion="7" />

</manifest>
```

Intent filters e menu

- Per aggiungere voci di menu in modo dinamico ed anonimo in fase di esecuzione, si deve utilizzare il metodo **addIntentOptions** dell'oggetto Menu;
- a tale metodo si deve passare in un **intent** che specifica i dati su cui si devono fare azioni.
- In genere questa chiamata viene fatta all'interno di uno dei metodi delle activity onCreateOptionsMenu o onCreateContextMenu.
- L'intent creato sarà utilizzato per ritrovare i componenti con l'intent filter che forniscono action per i dati specificati.

Esempio

- L'intent è utilizzato anche per trovare le action, quindi potrebbe non essere assegnata ad esso alcuna action; saranno invece specificati i dati su cui eseguire le azioni e la category (CATEGORY_ALTERNATIVE e/o CATEGORY_SELECTED_ALTERNATIVE).

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // TODO Auto-generated method stub

    int order = Menu.FIRST;

    menu.add(0, order, order++, "Fai qualcosa");

    Intent intent = new Intent(Intent.ACTION_VIEW);
    intent.setData(Uri.parse("content://contacts/people"));
    intent.addCategory(Intent.CATEGORY_ALTERNATIVE);
    menu.addIntentOptions(1, 1, Menu.CATEGORY_ALTERNATIVE, this.getComponentName(), null,
            intent, 0, null);

    return true;
}
```

Sendbroadcast, Receivers

- Abbiamo già detto che gli intent implementano un meccanismo di message passing e come tale permettono di inviare messaggi al di là del processo che li istanzia.
- Utilizzando il metodo ***sendBroadcast*** gli intent sono in grado di inviare in broadcast messaggi strutturati che anonimamente possono essere ricevuti da altri componenti.
- Tali componenti dovranno implementare oggetti **Broadcast Receivers** per ascoltare e gestire tali messaggi.

Intent broadcast

- Gli **Intent broadcast** vengono utilizzati per notificare gli eventi di sistema o di una qualunque applicazione, ciò permette di estendere il modello di programmazione event-driven a più applicazioni.
- Ciò contribuisce a rendere le applicazioni aperte; per una applicazione è possibile reagire ad intent broadcast senza dover modificare il codice l'originale.
- Nelle applicazioni si può rimanere in ascolto di Intent Broadcast per sostituire o migliorare applicazioni native, o reagire ai cambiamenti del sistema.

Intent broadcast

- Android utilizza ampiamente Intent broadcast per trasmettere eventi di sistema (livello di carica della batteria, connessioni di rete e le chiamate in arrivo).
- Per fare il broadcast di un intent basterà costruirlo e richiamare la funzione **sendBroadcast** con parametro l'intent stesso

Esempio

- Come accade per l'avvio di activity è possibile fare il broadcast di intent espliciti o impliciti.

```
Intent explicitIntent = new Intent(this, MyBroadcastReceiver.class);
intent.putExtra(LifeformDetectedReceiver.EXTRA_LIFEFORM_NAME,
                            detectedLifeForm);
intent.putExtra(LifeformDetectedReceiver.EXTRA_LATITUDE,
                            mLatitude);
intent.putExtra(LifeformDetectedReceiver.EXTRA_LONGITUDE,
                            mLongitude);
sendBroadcast(explicitIntent);
```

- Intent esplicito verrà ricevuto solo dalla classe MyBroadcastReceiver

Esempio

- Gli Intent impliciti verranno potenzialmente ricevuti da più broadcast receivers

```
Intent intent = new Intent(LifeFormDetectedReceiver.NEW_LIFEFORM_ACTION);
intent.putExtra(LifeformDetectedReceiver.EXTRA_LIFEFORM_NAME,
                            detectedLifeForm);
intent.putExtra(LifeformDetectedReceiver.EXTRA_LATITUDE,
                            mLatitude);
intent.putExtra(LifeformDetectedReceiver.EXTRA_LONGITUDE,
                            mLongitude);
sendBroadcast(intent);
```

Broadcast receiver

- Un **Broadcast receiver** è un componente in grado di ascoltare broadcast intents.
- Affinché possa ricevere questo tipo di messaggi è necessario che esso sia registrato via codice o attraverso il manifest file, in questo caso si parlerà di **manifest receivers**.
- Per creare un receivers si deve estendere la classe **BroadcastReceiver** e ridefinire la funzione **OnReceive**

Esempio

```
package edu.pm.broadrec;

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

public class myReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {

    }
}
```

Deadline di elaborazione di un broadcast Intent

- Quando viene «lanciato» un broadcast intent, il metodo onReceive sarà eseguito nel Thread principale dell'app che contiene il relativo broadcast receiver.
- L'elaborazione di quest'ultima funzione non dovrà superare i 10 secondi altrimenti l'app sarà considerata non rispondente e si proverà a terminarla

Broadcast receiver

- La registrazione di broadcast receiver che rispondono a intent broadcast lanciati dalla stessa applicazione di solito vengono registrati da codice.
- I broadcast receiver che si registrano in questo modo potranno rispondere solo se l'app (o meglio il componente che lo registra) è in esecuzione.
- Particolarmente usato quando il receiver si occupa di aggiornare la UI di una activity. (registrazione onStart, deregistrazione onStop)

Esempio

```
public class MainActivity extends AppCompatActivity {


    myReceiver myrec;
    IntentFilter filter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        myrec = new myReceiver();
        filter = new IntentFilter( action: "myactionbr");
    }

    @Override
    protected void onStart() {
        super.onStart();
        registerReceiver(myrec, filter);
    }

    @Override
    protected void onStop() {
        unregisterReceiver(myrec);
        super.onStop();
    }
}
```