Anatomia di una applicazione

Elementi di base

Gli elementi di base per costruire una applicazione Android sono cinque:

- Activity
- Intent
- Broadcast Receiver
- Service
- Content Provider

Activity (1/3)

- Le Activity sono l'elemento di base più comune.
- Rappresentano blocchi logici dell'applicazione ed interagiscono con l'utente mediante i dispositivi di input dello smartphone.
- Serve una Activity per ogni schermata dell'applicazione (es. una per comporre un messaggio, una per consultare la rubrica e una per le configurazioni).
- Vengono derivate dalla classe base android.app.Activity.

Activity (2/3)

- La propria Activity derivata mostrerà una interfaccia utente composta da View e risponderà ad eventi.
- Spostarsi da una schermata all'altra fa avviare una nuova Activity.
- In certi casi l'Activity chiamata restituirà un valore all'Activity chiamante (ad esempio una schermata che permette di selezionare una foto restituirà al chiamate la foto scelta).
- Quando si apre una nuova schermata, quella precedente è messa in pausa e inserita in una pila (*activity stack*).

Activity (3/3)

- L'utente può quindi navigare avanti e indietro tra le schermate aperte nell'activity stack.
- Per alcune schermate può essere inappropriato rimanere nell'activity stack, e in questo caso è possibile rimuoverle.
- Android mantiene un activity stack per ogni applicazione lanciata dalla schermata Home.

Activity - Fragment

- Con Android 3.0 sono stati introdotti i Fragment.
- Un Fragment costituisce una porzione di UI di una data Activity con le seguenti proprietà:
  - E' caratterizzata da un proprio ciclo di vita
  - Possiede un proprio input
  - Può essere aggiunto e rimosso mentre l'Activity è in esecuzione
- Per es., una Activity potrebbe includere molti Fragment quando il display è abbastanza ampio, oppure solo alcuni in caso contrario.

Intent 1/4

- L'Intent rappresenta un messaggio in grado di attivare i tre componenti-base di una applicazione: Activity, Service e Broadcast Receiver
- Un Intent descrive cosa una applicazione vuole che venga fatto.
- La struttura dati dell'Intent si compone dell'azione da effettuare e dei dati su cui agire.
- Valori tipici per l'azione sono MAIN (la schermata principale dell'applicazione), VIEW, PICK, EDIT etc.

Intent 2/4

Un Intent può essere Esplicito o Implicito:

- **Esplicito**: all'interno del costruttore dell'Intent specifichiamo la classe da eseguire (tipicamente una Activity).
- Nell'esempio seguente è istanziato un Intent in cui è dichiarata una Activity da eseguire.

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

Intent 3/4

- **Implicito**: rappresenta l'essenza della filosofia Android. Un Intent implicito dichiarata l'azione da eseguire pur senza sapere chi la eseguirà.
- Nel seguente esempio è dichiarato un Intent mediante cui avviare una Activity che visualizzerà il sito definito nell'URI

```
Uri uri = Uri.parse("http://www.google.com");
Intent it  = new Intent(Intent.ACTION_VIEW,uri);
startActivity(it);
```

Intent 4/4

- In generale, potrebbe essere necessario richiedere specifici ”permission” per eseguire un particolare Intent
- Tali permessi devono essere dichiarati nel manifest.xml, ad esempio:

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

IntentFilter 1/2

- Una classe correlata si chiama IntentFilter.
- Una Activity che è in grado di mostrare le informazioni di contatto per una persona pubblicherà un IntentFilter nel quale dirà che sa come gestire l'azione VIEW quando essa è applicata a dati che rappresentano una persona.
- Le Activity pubblicano i loro IntentFilter nel file AndroidManifest.xml.
- La navigazione da una schermata all'altra viene effettuata risolvendo gli Intent.

IntentFilter

Un IntentFilter sempre presente nella nostra applicazione è il seguente:

```
<activity android:name=".IntentSample"
  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
  android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
 </activity>
```

- La action dichiara che si tratta di un *entry point*, mentre la category indica che questo *entry point* può essere listato nell'Application Launcher
- Ovvero, la nostra Activity si rende disponibile nella home per essere eseguita mediante click/tap sull'icona relativa

Intent - Navigazione

Per navigare avanti una Activity chiama

```
startActivity(myIntent);
```

Il sistema guarda quindi gli IntentFilter per tutte le applicazioni installate e sceglie l'Activity il cui IntentFilter è più appropriato per myIntent.

La nuova Activity è informata dell'Intent, e viene quindi lanciata.

Intent - Esecuzione a run-time

Il fatto di risolvere gli Intent al tempo di esecuzione solo quando **startActivity()** viene chiamata comporta almeno due benefici principali:

- Le Activity possono riutilizzare funzionalità di altri componenti semplicemente facendo una richiesta sotto forma di un Intent.
- Le Activity possono essere sostituite in ogni momento da una nuova Activity con un IntentFilter equivalente.

Rif: http://developer.android.com/reference/android/content/Intent.html

BroadcastReceiver 1/2

- Il secondo elemento di base di una applicazione è la classe BroadcastReceiver definita all'interno del package android.content.
- Viene utilizzato quando si vuole che parte del codice della propria applicazione venga eseguito in risposta ad un evento esterno (es. il telefono squilla, o una connessione dati diventa disponibile, oppure quando è mezzanotte).
- Affinché un BroadcastReceiver venga chiamato non è necessario che la relativa applicazione sia in esecuzione.

BroadcastReceiver 2/2

- I BroadcastReceiver non hanno interfaccia grafica, ma possono usare il NotificationManager per avvisare l'utente che è successo qualcosa di interessante.
- Anche i BroadcastReceiver come gli IntentFilter sono registrati nel file AndroidManifest.xml, ma possono essere anche registrati dal sorgente usando la chiamata Context.registerReceiver().
- Le applicazioni possono lanciare i broadcast alle altre con Context.sendBroadcast().

PendingIntent 1/2

- Un PendingIntent è un oggetto che contiene un Intent di un'azione da effetturare con esso
- Passando un PendingIntent ad un'altra application, le garantiamo il permesso di eseguire l'operazione che abbiamo stabiito come se l'altra applicazione fosse proprio la nostra applicazione

Pending Intent 2/2

Le istanze di questa classe sono create con :

- **getActivity(Context, int, Intent, int**) :consente al componente che riceve tale pending intent di lanciare una nuova activity tramite Context.startActivity(Intent).
- **getBroadcast(Context, int, Intent, int)**: consente al componente che riceve tale pending intent di inviare un annuncio broadcast tramite Context.sendBroadcast(Intent).
- **getService(Context, int, Intent, int)**: consente al componente che riceve tale pending intent di avviare un nuono service tramiteContext.startService(Inten

Service

- Un altro elemento di base delle applicazioni Android è il Service, ovvero del codice senza interfaccia grafica e sono definiti all'interno del package android.app.
- Un buon esempio è la riproduzione di un brano musicale: poiché si vuole che la riproduzione continui anche dopo che l'utente ha cambiato schermata, non la si può lasciare eseguire ad una Activity ma bisogna chiamare Context.startService() per eseguirla in background.

Cosa non è un service

- Un Service non è un processo separato: in altre parole l'applicazione e il suo servizio sono eseguiti all'interno dello stesso processo (se non altrimenti specificato)
- Non è un thread: non è uno strumento mediante cui fare eseguire delle attività al di fuori del thread principale

Cosa è un service

- Un Service è uno strumento mediante cui l'applicazione vuol comunicare al sistema che una data operazione deve essere eseguita in background anche quando l'utente non sta interagendo direttamente con essa.
- Uno strumento mediante il quale l'applicazione mette a disposizione di altre applicazioni alcune delle proprie funzionalità. Ciò può essere realizzato invocando una Context.bindService() mediante la quale viene stabilita una connessione tra l'applicazione richiedente ed il servizio richiesto.

Content Provider

- L'ultimo elemento di base è il Content Provider, definito all'interno della classe Android.Content
- Solitamente le applicazioni memorizzano i loro dati all'interno del file-system oppure in un database SQLite, ma se i dati di una applicazione devono essere condivisi con le altre bisogna usare la classe ContentProvider.
- Un ContentProvider è una classe che implementa un insieme standard di metodi per permettere alle altre applicazioni di salvare e recuperare dati.

Conclusioni

In questa lezione abbiamo visto brevemente le quattro classi alla base di ogni applicazione Android: Activity, Intent, BroadcastReceiver, Service e ContentProvider.