Oltre le Activities: Intents

Programmazione Mobile

A.A. 2021/22

M.O. Spata

Intent per avviare Activity o Services

- Le applicazioni mobile nella maggior parte delle piattaforme vengono eseguite nella propria "**sandbox**". Ovvero sono isolate dalle altre applicazioni e hanno accesso limitato all'hardware e ai componenti nativi.
- Abbiamo già evidenziato che anche se le applicazioni Android vengono eseguite in "**sandbox**" esse possono utilizzare **Intent** per scambiare dati con altri componenti della stessa applicazione e con altre applicazioni, ovvero fare partire **Activity** o **Services** sia esplicitamente che utilizzando metodi impliciti a runtime.

Intents

- Gli Intents sono utilizzati come un meccanismo di message-passing tra componenti della stessa applicazione, e tra applicazioni diverse.
- Gli Intents possono essere utilizzati per:
  - Dichiarare l'intenzione che una activity o un service sarà avviato per eseguire un'azione con (o su) una specifica porzione di dati
  - Messaggi di broadcast per comunicare che un evento (o azione) si è verificato
  - Avviare esplicitamente un particolare servizio o attività

Intent: impliciti ed espliciti

- E' possibile utilizzare **Intents** per permettere l'interazione tra uno qualsiasi dei componenti dell'applicazione installata sul dispositivo Android, indipendentemente dall'applicazione di cui fanno parte.
- *Questo trasforma il dispositivo da una piattaforma contenente una collezione di componenti indipendenti in un unico sistema interconnesso*.
- Uno degli usi più comuni per **Intents** è quello di avviare nuove attività e ciò può essere fatto:
  - in modo esplicito (specificando la classe)
  - implicitamente (richiedendo una azione da eseguire su una porzione di dati). In questo caso l'azione non deve essere effettuata necessariamente da una attività all'interno dell'applicazione chiamante

Usi degli Intent

- Gli Intents possono anche essere usati per trasmettere messaggi attraverso il sistema.
- Qualsiasi applicazione può registrare Broadcast Receivers per ascoltare, e reagire a questi messaggi (creazione di applicazioni event driven).
- Gli Intents di Android possono essere usati altresì per trasmettere eventi di sistema, come cambiamenti nello stato di connessione Internet o i livelli di carica della batteria.

Accoppiamento di Intent con applicazioni native

- Le applicazioni native di Android, come il combinatore telefonico ed il gestore di SMS, registrano componenti che sono in grado di ascoltare Intents broadcast specifici - come ''telefonata in arrivo'' oppure ''messaggio SMS ricevuto''- e reagiscono di conseguenza.
- L'utilizzo di Intents per propagare le azioni - anche all'interno della stessa applicazione - permette il disaccoppiamento dei componenti, ovvero la sostituzione senza soluzione di continuità di elementi di una applicazione.

Intent per lanciare Activity

- L'uso più comune degli Intent è quello di legare e far comunicare tra loro i componenti dell'applicazione.
- Gli Intent vengono utilizzati per avviare, ed effettuare quindi le transizioni tra Activity.
- Per aprire una nuova Activity si deve effettuare una chiamata alla funzione startActivity che riceve come parametro per l'appunto un Intent:

```
startActivity(myIntent);
```

Intent resolution

- L'intent può specificare esplicitamente la classe Activity da aprire, o includere un'azione per richiedere l'apertura di una Activity in grado di eseguirla.
- In quest'ultimo caso a run-time toccherà ad Android scegliere l'attività più opportuna; per fare questo il framework utilizza un processo noto come risoluzione di intents (**intent resolution**).
- Il metodo **startActivity** trova e inizia la singola Activity che meglio corrisponde all'Intent specificato.

startActivityForResult

- Quando si utilizza **startActivity** l'applicazione non riceverà alcuna notifica al termine dell'activity avviata. Per monitorare i feedback dall'activity lanciata, si deve utilizzare il metodo **startActivityForResult** descritto in dettaglio nel seguito.
- Abbiamo già visto che le applicazioni consistono in una serie di schermate o istanze di Activity, che devono essere specificate nell'application manifest. Per collegarle tra loro, dunque in questo file deve essere esplicitamente indicata l'attività da aprire.

Esempio

- Per selezionare in modo esplicito una classe activity si deve creare un nuovo intent, specificando il contesto dell'applicazione corrente e la classe activity da lanciare, da passare appunto alla funzione start activity.

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

- Dopo la chiamata alla funzione startActivity, la nuova activity (in questo caso MyOtherActivity) sarà creata e diventerà visibile ed attiva (ovvero sarà al top dell'activity stack).

Chiusura Activity

- Se viene richiamato il metodo **finish** nella nuova activity o viene premuto il tasto back del dispositivo la nuova activity verrà chiusa e rimossa dallo stack.
- In alternativa è possibile passare all'activity precedente o ad un’altra activity chiamando nuovamente la funzione startActivity.

Intent impliciti

- Gli intent impliciti rappresentano un meccanismo che permette di richiedere azioni e servizi a componenti di applicazioni sconosciute. Ovvero è possibile richiedere al sistema di avviare una activity che sia in grado di compiere una data azione senza conoscere di quale applicazione faccia parte.
- Quando si realizza un nuovo intent implicito da usare con la funzione startActivity si deve pertanto indicare l'azione da svolgere e facoltativamente si può fornire l’URI (Uniform Resource Identifier) dei dati necessari per eseguire l'azione specificata.

Intent impliciti

- È anche possibile inviare i dati aggiuntivi per l'activity target con l'aggiunta di dati extra per l'intent.
- Quando si utilizzano intent impliciti per avviare un'activity, Android a run time sceglierà la classe più adatta ad eseguire le azioni richieste sui dati specificati. Ciò significa che è possibile creare applicazioni che utilizzano le funzionalità di altre.
- Ad esempio, per consentire agli utenti di effettuare chiamate dalla propria applicazione è possibile implementare un dialer nuovo, oppure si potrebbe usare un intent implicito che richieda l'azione (dialing) da compiere su un numero di telefono (URI)

Esempio

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

- Tramite questo intent Android lancerà un'attività che prevede l'azione composizione su un numero telefonico; in questo caso l'attività dialer.
- Nei casi in cui più activity sono in grado di eseguire una determinata azione vengono presentate le diverse scelte all'utente.

Intent di applicazioni native e di terze parti

- Le numerose applicazioni Android native forniscono diverse activity per effettuare azioni su specifici tipi di dati.
- Le applicazioni terze-parti possono essere registrare per supportare e svolgere nuove azioni o per fornire alternative alle azioni native.

Valori di ritorno delle activity

- Una activity iniziata attraverso la funzione startActivity è indipendente dalla activity che ha richiamato la funzione e non darà alcun feedback quando si chiuderà.
- Tuttavia, è possibile avviare un'activity come sub-activity in maniera tale che essa sia connessa alla classe parent.
- Una sub-activity quando si chiude è in grado di attivare una funzione handler (event handler) all'interno della sua parent activity.

Sub-activity

- Le sub-activity rappresentano una ottima soluzione nelle situazioni in cui un'activity sta fornendo i dati di input (ad esempio, un utente seleziona un elemento da un elenco) ad un' altra.
- Le Sub-activity sono in realtà activity lanciate in un modo diverso, come tali essi devono essere registrate nell'application manifest.
- Di fatto qualsiasi attività registrata nel manifest file può essere aperta come una sub-Activity comprese le activity di sistema o di altre applicazioni terze parti

startActivityForResult

- L'avvio di una sub-activity avviene utilizzando il metodo startActivityForResult.
- Tale metodo funziona in modo analogo a startActivity, ma con una differenza importante oltre all'intent (starting esplicito o implicito) deve essere passato come parametro anche un request code; si tratta di un valore utilizzato per identificare univocamente la sub-activity che finendo ha restituito un risultato

Esempio

- Esempio sub-activity con avvio esplicito

```
private static final int SHOW_SUBACTIVITY = 1;

Intent intent = new Intent(this, MyOtherActivity.class);
startActivityForResult(intent, SHOW_SUBACTIVITY);
```

- Esempio sub-activity con avvio implicito

```
private static final int PICK_CONTACT_SUBACTIVITY = 2;

Uri uri = Uri.parse("content://contacts/people");
Intent intent = new Intent(Intent.ACTION_PICK, uri);
startActivityForResult(intent, PICK_CONTACT_SUBACTIVITY);
```

setResult

- Prima della chiamata a finish ovvero subito prima del suo return una sub-activity chiama la funzione **setResult** al fine di restituire un risultato all'activity chiamante.
- Il metodo setResult accetta due parametri: il result code e lo stesso result, rappresentato da un Intent.
- Il result code è il risultato dell'esecuzione della sub-activity (generalmente **Activity.RESULT_OK** o **Activity.RESULT_CANCELED**). E' comunque possibile utilizzare i codici personalizzati; setResult supporta qualsiasi valore intero.

Esempio

- L'intent restituito come risultato spesso include un URI a un pezzo di contenuto (come il contatto selezionato, un numero di telefono, o un file multimediale) e una collezione di dati extra utilizzata per restituire informazioni aggiuntive.
- Il codice seguente è tratto da un metodo onCreate di una sub-Activity, e mostra come attraverso i due pulsanti OK e Cancel, la sub- activity potrebbe restituire risultati diversi all'attività chiamante

```
Button okButton = (Button) findViewById(R.id.ok_button);
  okButton.setOnClickListener(new View.OnClickListener() {
  public void onClick(View view) {
    Uri data = Uri.parse("content://horses/" + selected_horse_id);

    Intent result = new Intent(null, data);
    result.putExtra(IS_INPUT_CORRECT, inputCorrect);
    result.putExtra(SELECTED_PISTOL, selectedPistol);
    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, null);
    finish();
  }
});
```

Chiusura di una Activity o sub-Activity

- Se l'attività è chiusa dall'utente premendo il tasto Back, o la funzione finish e cioè senza una previa chiamata a setResult, il result code sarà impostato a RESULT_CANCELED e l'intent result a null.
- Quando si chiude una sub-activity viene generato l'evento onActivityResult nell'attività chiamante e verrà pertanto richiamata l'omonima funzione di callback.
- Per gestire i risultati restituiti dalla sub-activity sarà necessario ridefinire questo metodo.

Parametri per una sub-activity

- **onActivityResult** riceve una serie di parametri:
  - **request code**: il codice che è stato usato per lanciare la sub activity appena finita.
  - **result code**: Il risultato impostato dalla sub-activity. Può essere qualsiasi numero intero, ma di solito sarà o Activity.RESULT_OK o Activity.RESULT_CANCELED.
  - **data**: un Intent utilizzato per contenere il pacchetto dei aggiuntivi restituiti dalla sub-activity.

Esempio

- Di seguito viene mostrato il codice per l'esecuzione del gestore di eventi **onActivityResult** all'interno di una activity:

```
private static final int SHOW_SUB_ACTIVITY_ONE = 1;
private static final int SHOW_SUB_ACTIVITY_TWO = 2;

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

 super.onActivityResult(requestCode, resultCode, data);

 switch(requestCode) {
   case (SHOW_SUB_ACTIVITY_ONE) : {
     if (resultCode == Activity.RESULT_OK) {
       Uri horse = data.getData();
       boolean inputCorrect = data.getBooleanExtra(IS_INPUT_CORRECT, false);
       String selectedPistol = data.getStringExtra(SELECTED_PISTOL);
     }
     break;
   }
   case (SHOW_SUB_ACTIVITY_TWO) : {
     if (resultCode == Activity.RESULT_OK) {
       // TODO: Handle OK click.
     }
     break;
   }
 }
}
```

Activity Intents

- La seguente lista, non esaustiva, mostra alcune delle azioni native disponibili nella classe Intents come costanti string di tipo statico.
- Durante la creazione di Intents impliciti è possibile utilizzare queste azioni, denominata Activity Intents, per avviare activity e sub-activity all'interno delle applicazioni.
  - ACTION_ANSWER Apre un'attività che gestisce le chiamate in entrata.
  - ACTION_CALL Apre un combinatore telefonico e avvia subito una chiamata utilizzando il numero fornito attraverso l'URI nell'oggetto Intent. In generale è meglio utilizzare ACTION_DIAL.
  - ACTION_DELETE inizia un'attività che consente di eliminare i dati specificati i dati l'intento di URI.
  - ACTION_DIAL Apre un dialer con il numero da comporre fornito attraverso un URI nell'Inten. Per default è gestito dal combinatore telefonico nativo di Android.
  - ACTION_EDIT richiede una attività che sia in grado di modificare i dati specificati attraverso URI dell'oggetto Intent.

Activity Intents

- ACTION_INSERT Apre una attività in grado di inserire nuovi elementi nell'insieme specificato nell'Intent verso l'URI . Quando viene chiamato come un sub-activity dovvrebbe restituire un URI cghe rappresenta l'elemento appena inserito.
- ACTION_PICK lancia un sub-activity che permette di scegliere un oggetto dal Content Provider specificato dall'URI dell'Intent. Una volta chiuso dovrebbe restituire un URI dell'elemento scelto. L'activity avviata dipende dai dati che devono essere selezionati: per esempio, passando content://contacts/people verrà richiamata la lista dei contatti nativa.
- ACTION_SEARCH lancia l'activity utilizzata per eseguire una ricerca. Il termne da ricercare si deve fornire sottoforma di stringa attraverso gli extra dell'intent ed utilizzado la chiave SearchManager.QUERY.
- ACTION_SENDTO lancia una activity in grado di inviare un messaggio al contatto indicato nell'Intent attraverso l'URI.
- ACTION_SEND lancia una activity che invia i dati specificati nell'Intent. Il destinatario contatto deve essere scelto tramite l'Activity selezionata. Si può utilizzare il metodo setType per impostare il tipo MIME dei dati trasmessi

Activity Intents

- I dati stessi devono essere conservati come un extra utilizzando le chiavi EXTRA_TEXT o EXTRA_STREAM, a seconda del tipo. Nel caso di e-mail, le applicazioni native di Android accetteranno extra attraverso le chiavi EXTRA_EMAIL, EXTRA_CC, EXTRA_BCC ed EXTRA_SUBJECT. Utilizzare ACTION_SEND solo per inviare dati a un destinatario remoto (non un'altra applicazione sullo dispositivo).
- ACTION_VIEW Richiede la visualizzazione dei dati forniti nell'Intento attraverso l'URI nella maniera più opportuna. Nativamente ad esempio indirizzi http: saranno aperti nel browser, numeri specificati con tel: saranno aperti attraverso il dialer per chiamare il numero, gli indirizzi geo: saranno visualizzati con Google Maps.
- ACTION_WEB_SEARCH Apre una activity che esegue una ricerca web basata sul testo fornito nell'intent attraverso l'URI.