

ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
Dipartimento di Informatica – Scienza e Ingegneria (DISI)
C.d.S. in Ingegneria e Scienze Informatiche, Campus di Cesena

Programmazione in Android

Activity

Angelo Croatti
a.croatti@unibo.it

Sistemi Embedded e Internet of Things
A.A. 2019 – 2020

Outline

- 1 Overview
- 2 Activity Lifecycle
 - Callbacks
- 3 Gestione delle Activity
 - Creazione
 - Avvio
 - Terminazione
- 4 Back Stack



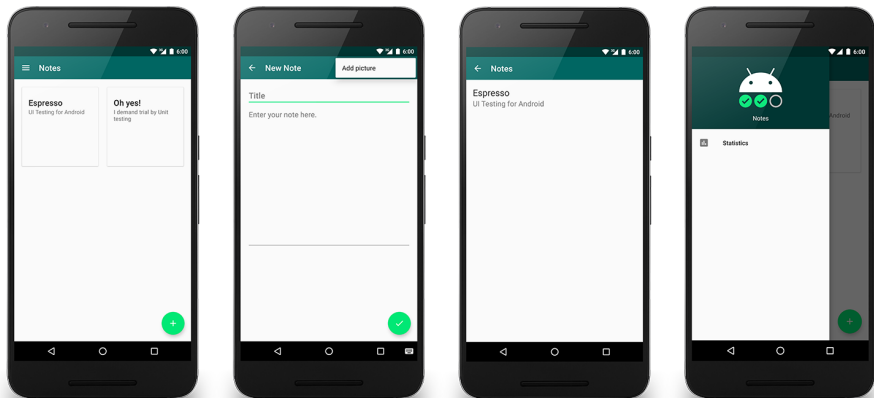
Activity (Overview)

- Componente che abilita l'interazione dell'utente con l'applicazione attraverso una specifica interfaccia grafica
- Tipicamente, ciascuna activity occupa l'intera dimensione dello schermo del device

Note

- Generalmente, un'applicazione Android consiste in una collezione di Activity tra loro interconnesse
- In un'applicazione Android deve sempre esistere un'activity principale (**Main Activity**), avviata dal sistema contestualmente alla richiesta di avvio dell'applicazione

Android App e Activity (esempio)



Outline

- 1 Overview
- 2 Activity Lifecycle**
 - Callbacks
- 3 Gestione delle Activity
 - Creazione
 - Avvio
 - Terminazione
- 4 Back Stack



Activity Lifecycle I

- Comprendere (e gestire) il Lifecycle di ciascuna Activity è fondamentale per poter sviluppare applicazioni Android.
 - ▶ **È influenzato anche dal comportamento che hanno gli altri componenti del sistema.**

“Stato” di un’Activity

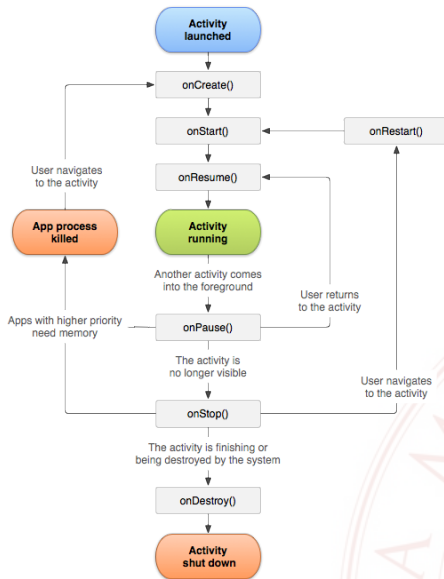
- **RESUMED (RUNNING)**: L’activity è in foreground e ha il focus.
- **PAUSED**: Caso in cui un’altra activity è in stato running, ma questa è ancora visibile.
- **STOPPED**: L’activity è in background, ovvero completamente oscurata da un’altra activity.

NOTA: Quando un’activity si trova in stato PAUSED o STOPPED, il sistema può eliminarla dalla memoria centrale in qualunque momento!

Activity Lifecycle II

- Il lifecycle di un'activity si compone di una serie di stati e di loop interni.
 - ▶ Definisce il comportamento dell'activity da quando viene lanciata a quando questa viene chiusa o interrotta dal sistema.
- Le transizioni da uno stato all'altro provocano l'invocazione automatica di uno o più metodi (callbacks).
 - ▶ Tali metodi possono essere ridefiniti per specificarne il comportamento.
 - ▶ In particolare: `onCreate()`, `onStart()`, `onResume()`, `onPause()`, `onStop()`, `onRestart()`, `onDestroy()`.

Activity Lifecycle III



Activity Lifecycle – Callback I

- **La ridefinizione di ciascuna callback deve obbligatoriamente richiamare l'implementazione originale corrispondente.**
 - ▶ Es.: La ridefinizione del metodo `onStart()` in una propria activity deve richiamare nel suo corpo lo stesso metodo definito nella super-class `Activity` con l'istruzione `super.onStart()`.

`onCreate(Bundle savedInstanceState)`

- Invocata quando l'activity è creata per la prima volta.
- Provvede al setup dell'activity.
- Riceve dal sistema l'eventuale stato precedentemente salvato dell'activity.

Activity Lifecycle – Callback II

onStart()

- Invocata subito prima che l'activity sia resa visibile all'utente.
- Prepara l'activity affinché possa essere posta nello stato di foreground.

onResume()

- Invocata dopo che l'activity è stata resa visibile all'utente ma prima che quest'ultimo possa cominciare ad interagire con essa.
- Al termine della sua esecuzione l'activity si trova nello stato RESUMED (o RUNNING).

Activity Lifecycle – Callback III

onPause()

- Invocata dal sistema sull'activity quando un'altra activity deve essere portata allo stato di RUNNING.
 - ▶ Implicazioni particolari se si considera il **multi-window mode** (presente da Android 7.0)
- Utilizzata generalmente per rilasciare le risorse eventualmente detenute dall'activity.
- Al termine della sua esecuzione l'activity si trova nello stato PAUSED.

onStop()

- Invocata quando l'activity non è più visibile all'utente.
- Al termine della sua esecuzione l'activity si trova nello stato STOPPED.

Activity Lifecycle – Callback IV

onRestart()

- Invocata quanto l'activity, precedentemente interrotta (ovvero che si trova nello stato STOPPED), deve essere riportata allo stato RUNNING.

onDestroy()

- Invocata immediatamente prima della fase di distruzione dell'activity, ovvero prima che sia rimossa dalla memoria centrale.
- Invocata sia che la distruzione dell'activity sia stata richiesta dall'utente, sia nel caso sia stata imposta dal sistema.

Outline

- 1 Overview
- 2 Activity Lifecycle
 - Callbacks
- 3 Gestione delle Activity
 - Creazione
 - Avvio
 - Terminazione
- 4 Back Stack



Creazione di un'Activity I

- Ogni nuova activity deve essere definita come sottoclasse di `android.app.Activity`.
- Deve implementare obbligatoriamente il metodo `onCreate()`.
 - ▶ Nel corpo del metodo deve essere associata all'activity la risorsa che definisce la relativa User Interface.
 - ▶ Si può chiamare il metodo `setContentView(int resID)` che richiede come parametro l'ID della risorsa specifica.
- Come best-practice, è consigliata l'implementazione anche del metodo `onPause()`.
 - ▶ Più che altro, per il rilascio di eventuali risorse occupate.
- L'implementazione di tutte le altre callback è opzionale.

Creazione di un'Activity II

Esempio

```
import android.app.Activity;

public class MyActivity extends Activity{

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

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

        //Release resources, make state persistent, ...
    }
}
```

Creazione di un'Activity III

- Ciascuna activity deve essere dichiarata nel File Manifest.
 - ▶ Deve necessariamente essere specificato un valore per l'attributo **android:name**, che deve contenere il nome completo della classe.
 - ▶ Possono essere aggiunti altri parametri per la configurazione dell'activity.

Esempio

```
<manifest> <!-- Manifest tag properties omitted -->
  <application>
    <activity
      android:name="com.example.MyActivity"
      android:label="@string/app_name"/>
    </application>
  </manifest>
```


Dichiarazione della Main Activity

- Tra le activity specificate nel File Manifest, una di questa deve essere *etichettata* come **Main Activity**, specificando un Intent Filter predefinito, con i seguenti parametri:
 - ▶ azione: `android.intent.action.MAIN`
 - ▶ categoria: `android.intent.category.LAUNCHER`

Esempio

```
<activity
  android:name="com.example.MainActivity"
  android:label="@string/app_name">

  <intent-filter>
    <action android:name="android.intent.action.MAIN"/>
    <category android:name="android.intent.category.LAUNCHER"/>
  </intent-filter>

</activity>
```

Avvio di un'Activity

- Si richiama il metodo `startActivity(Intent intent)` fornito dalla classe `Activity`.
 - ▶ Accetta come parametro un `Intent` creato con esplicito riferimento all'activity da avviare.
 - ▶ Può contenere una serie di parametri di configurazione (**flags**) per istruire il sistema su come avviare l'activity.
 - ▶ Può trasmettere alcuni dati (**extras**) alla nuova activity.

Esempio – Avvio dell'istanza di `NewActivity`

```
Intent intent = new Intent(this, NewActivity.class);  
intent.putExtra("MESSAGE_FROM_CREATOR", "I'm your creator!");  
startActivity(intent);
```

Esempio – Recupero del dato trasmesso

```
//Within the NewActivity instance onCreate() callback  
String msg = getIntent().getStringExtra("MESSAGE_FROM_CREATOR");
```

Avvio di un'Activity (con risultato) I

- Può essere necessario avviare un'activity con lo scopo di svolgere un certo compito e voler attendere un risultato prodotto alla sua terminazione, retro-propagato all'activity chiamante.
 - ▶ Es. Richiedere all'utente una serie di dati, compilazione di una form.
- Si utilizza il metodo `startActivityForResult(Intent intent, int reqID)`, associando un ID arbitrario alla richiesta.
- Per poter intercettare il risultato deve essere implementato il metodo `onActivityResult()`
 - ▶ Richiamato dal sistema tutte le volte che è disponibile il risultato di un'activity avviata esplicitamente con lo scopo di produrre un risultato.
 - ▶ I relativi parametri di input consentono di verificare lo stato del risultato e a quale richiesta tali informazioni fanno riferimento.

Avvio di un'Activity (con risultato) II

Esempio – Avvio dell'istanza di CheckLoginActivity

```
public class MyActivity extends Activity{

    private static final int REQUEST_CODE = 12345;

    public void startButtonClick(View view){
        Intent i = new Intent(this, CheckLoginActivity.class);
        startActivityForResult(i, REQUEST_CODE);
    }

    @Override
    protected void onActivityResult(int reqID, int res, Intent data){
        if (reqID == REQUEST_CODE && res == Activity.RESULT_OK){
            //do something
            data.getStringExtra("username")
        }
    }
}
```

Avvio di un'Activity (con risultato) III

- Il risultato, prodotto dall'activity avviata allo scopo, può essere retro-propagato attraverso un intent.
 - ▶ Si aggiungono tutti gli extras necessari ad un intent
 - ▶ Si richiama il metodo `setResult(int resCode, Intent data)` prima di terminare l'activity.
 - ▶ Possibili tipi di risultato: `Activity.RESULT_OK`, `Activity.RESULT_CANCELED`, ...

Esempio – Restituzione del risultato

```
public void closeButtonClick(View view){  
    Intent returnIntent = new Intent();  
    returnIntent.putExtra("username", "mario.rossi");  
  
    setResult(Activity.RESULT_OK, returnIntent);  
    finish();  
}
```

Terminazione di un'Activity

- **In genere la terminazione delle activity è lasciata al sistema!**
 - ▶ ...in relazione al lifecycle e alla disponibilità di risorse computazionali;
 - ▶ ...per garantire all'utente una migliore esperienza d'uso.
- Alternativamente, un'activity può terminare se stessa chiamando il metodo `finish()` fornito dalla classe Activity.

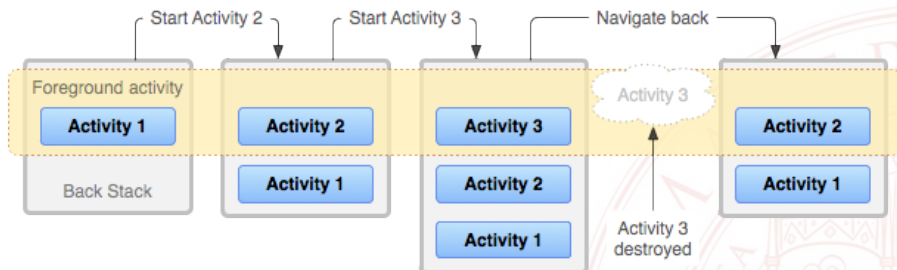
Outline

- 1 Overview
- 2 Activity Lifecycle
 - Callbacks
- 3 Gestione delle Activity
 - Creazione
 - Avvio
 - Terminazione
- 4 Back Stack



Activity Back Stack I

- Ciascuna applicazione contiene generalmente più activity.
- Solo una può essere nello stato RUNNING e visibile all'utente.
- Tutte le altre (non terminate) sono inserite in uno stack LIFO di activity chiamato **Back Stack**.







Activity Back Stack II

- Salvo diversa specifica, le activity sono inserite nello stack secondo l'ordine di attivazione/avvio.
 - ▶ Tramite le flag dell'intent è possibile imporre modalità di gestione delle activity sullo stack.
- Alla base dello stack è sempre presente l'activity che rappresenta la Home Screen del sistema.
- Alla pressione del tasto **back**, l'activity in foreground viene rimossa dallo stack e quella precedente viene portata nello stato di foreground.

» developer.android.com/guide/components/tasks-and-back-stack

» developer.android.com/guide/components/recents

Riferimenti - Risorse Online

-  **Android Developers - Guide**
» <https://developer.android.com/guide/>
-  **Android Developers - API Reference**
» <https://developer.android.com/reference/>
-  **Android Developers - Samples**
» <https://developer.android.com/samples/>
-  **Android Developers - Design & Quality**
» <https://developer.android.com/design/>



Riferimenti - Libri

-  Zigurd Mednieks, Laird Dornin, G. Blake Meike, Masumi Nakamura
Programming Android
O'Reilly, 2011
-  Chris Haseman, Kevin Grant
Beginning Android Programming: Develop and Design
Peachpit Press, 2013
-  Ronan Schwarz, Phil Dutson, James Steele, Nelson To
The Android Developer's Cookbook : Building Applications with the Android SDK
Addison-Wesley, 2013
-  Theresa Neil
Mobile Design Pattern Gallery: UI Patterns for Smartphone App
O'Reilly, Second Edition, 2014