5-Graphical User Interface - part III

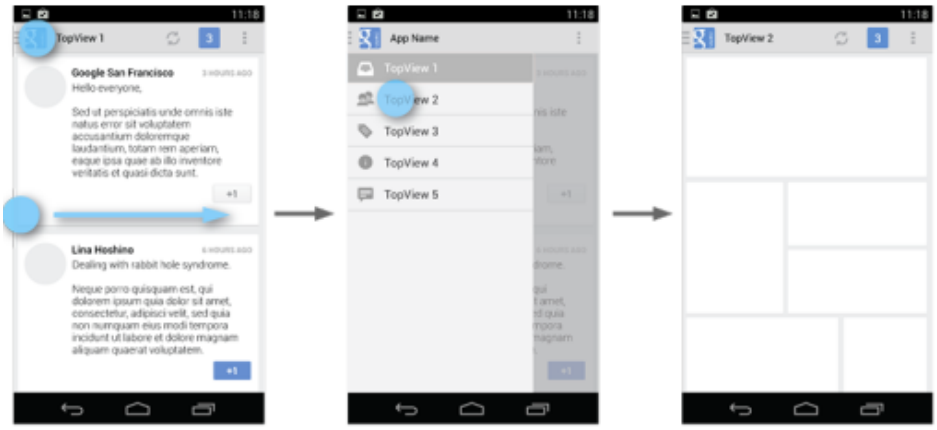
Navigation Drawer

Il navigation drawer è un pannello che si apre facendo una transizione dal bordo sinistro dello schermo e mostra le opzioni principali di navigazione dell'applicazione.

L'utente può far apparire il navigation drawer trascinando dal bordo sinistro dello schermo o toccando l'icona dell'applicazione nella action bar, mentre il navigation drawer si espande, copre il contenuto ma non la action bar.

> Nota
> Se si utilizza una Toolbar, è possibile modificare questo comportamento e sovrapporre anche la ActionBar/ToolBar.

> Nota
> Il menu overflow nell'angolo in alto a destra con le voci standard per le impostazioni e l'aiuto rimane visibile.



When to Use the Navigation Drawer

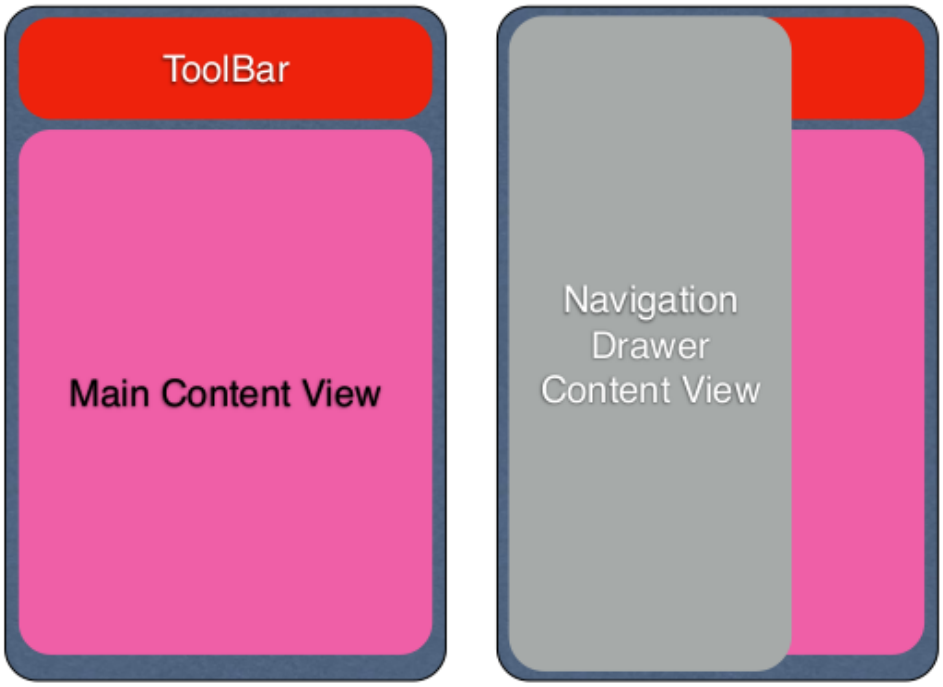
Il navigation drawer non è una sostituzione generale per la navigazione di primo livello tramite spinners o schede (tabs), La struttura dell'applicazione dovrebbe guidare la scelta del pattern da utilizzare per il passaggio di primo livello.

> Nota
> Un buon esempio in cui i navigation drawer funzionano meglio è quando hai più di 3 viste di primo livello,

> infatti i navigation drawer sono ideali per visualizzare contemporaneamente un gran numero di destinazioni di navigazione

> Nota
> Bisogna utilizzare il navigation drawer se si ha più di 3 viste di primo livello diverse, in caso contrario, è meglio utilizzare schede fisse (fixed tabs) per l'organizzazione di primo livello per facilitare la scoperta e l'interazione

Navigation Drawer Structure



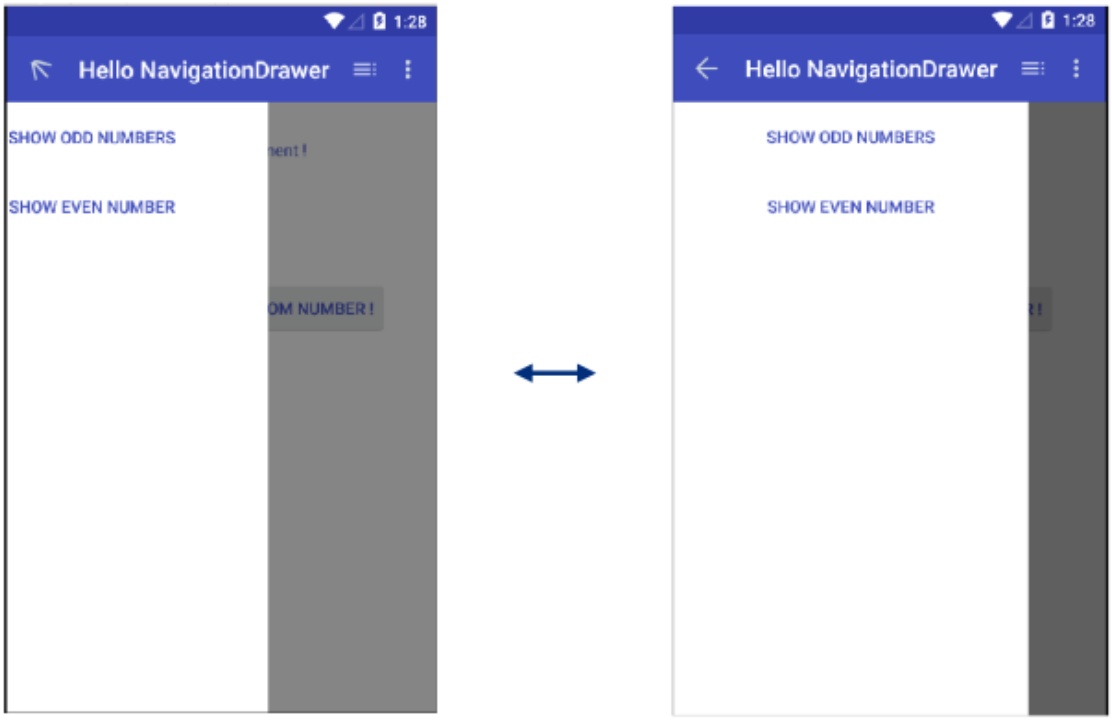
Navigation Drawer Structure (XML)

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
                                xmlns:tools="http://schemas.android.com/tools"
                                android:layout_width="match_parent"
                                android:layout_height="match_parent"
                                android:orientation="vertical" >
                                <!-- toolbar -->
        <androidx.appcompat.widget.Toolbar android:id="@+id/my_awesome_toolbar"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:background="?attr/colorPrimary"

android:minHeight="?attr/actionBarSize" />
        <!-- struttura navigation drawer -->

        <androidx.drawerlayout.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/drawer_layout"

android:layout_width="match_parent"

android:layout_height="match_parent" >
                <!-- contenitore principale -->
                <FrameLayout android:id="@+id/container"
                                                android:layout_width="match_parent"
                                                android:layout_height="match_parent"

tools:context="com.mobdev.multifragment.MainActivity"
                                                tools:ignore="MergeRootFrame" >
                </FrameLayout>
                <!-- The navigation drawer -->
                <LinearLayout android:id="@+id/navigation_drawer"
```

```
                                                android:layout_width="match_parent"
                                                android:layout_height="match_parent"
                                                android:layout_gravity="start"
                                                android:background="@color/grey_white_1000"
                                                android:orientation="vertical" >
                <!-- The navigation drawer -->
                </LinearLayout>
        </androidx.drawerlayout.widget.DrawerLayout>
</LinearLayout>
```

Navigation Drawer Structure (Code)

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.navigation_drawer_layout);
        if (savedInstanceState == null) {
                randomFragment = new RandomNumberFragment();
                getSupportFragmentManager().beginTransaction().add(R.id.container,
randomFragment).commit();
        }
        Toolbar toolbar = (Toolbar)findViewById(R.id.my_awesome_toolbar);
        setSupportActionBar(toolbar);

        // binding tra le toolbar

        mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);

        // gestisce come il drawer è mostrato/nascosto
        toggle = new ActionBarDrawerToggle(this, mDrawerLayout, toolbar,
R.string.drawer_open, R.string.drawer_close);

        toggle.setDrawerIndicatorEnabled(true);
        mDrawerLayout.setDrawerListener(toggle);
        // metodo custom e interno per impsotare il contenuto del drawe come nel file XML
        setupDrawerUI();
}


// binding tra le toolbar
@Override
protected void onPostCreate(Bundle savedInstanceState) {
        super.onPostCreate(savedInstanceState);
        toggle.syncState();
}
```



ViewPager

I ViewPagers hanno incorporato i gesti di scorrimento per passare tra le pagine, e mostrano di default le animazioni di scorrimento dello schermo. Le ViewPages utilizzano un PagerAdapter, per fornire nuove pagine da visualizzare, che utilizzerà i Fragment per definire il contenuto di ogni pagina.

Il ViewPager è un ViewGroup che funziona in modo simile alle AdapterViews (come ListView e Gallery), è stato rilasciato come parte della Compatibility Package revision 3 e funziona con Android 1.6 e versioni successive

> Nota
> Se si utilizza il ViewPager in un layout XML, bisogna assicurarsi di utilizzare il riferimento completo della classe

Implementation

1. Bisogna impostare la vista del contenuto sul layout con il ViewPager
2. Bisogna crea una classe che estende la classe astratta `FragmentStatePagerAdapter` e implementa il metodo `getItem()` per fornire istanze di `ScreenSlidePageFragment` come nuove pagine. Il pager adapter richiede anche che viene implementato il metodo `getCount()`, che restituisce il numero di pagine che l'adapter adnrà a creare (5 nell'esempio)
3. Bisogna collegare il PagerAdapter al ViewPager
4. Bisogna gestire il pulsante di ritorno del dispositivo spostandosi all'indietro nello stack virtuale dei fragment, se l'utente si trova già sulla prima pagina, tornare indietro allo stack delle attività precedente

```
<androidx.viewpager.widget.ViewPager
xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/pager"

android:layout_width="match_parent"

android:layout_height="match_parent" />
```

Implementation of the FragmentPagerAdapter

```
public class AppSectionsPagerAdapter extends FragmentPagerAdapter {
        public AppSectionsPagerAdapter(FragmentManager fm) {
                super(fm);
        }
        /**
        * Returns the Fragment associated to the element in position 'i'
        */
        @Override
        public Fragment getItem(int i) {
                switch (i) {
                        case 0:
                                return new BookmarkGridFragment();
                        case 1:
                                return new BookmarkListFragment();
                        default:
                                return new BookmarkGridFragment();
                }
        }

        /**
        * Returns the number of element in the ViewPager
        */
        @Override
        public int getCount() {
                return 2;
        }
        /**
        * Returns the title (as CharSequence) of the Page at position 'position'
        */
        @Override
        public CharSequence getPageTitle(int position) {
                switch (position) {
                        case 0:
                                return "Add";
                        case 1:
                                return "Top";
                        case 2:
                                return "My List";
                        default:
                                return "No Title !";
```

```
                }
        }
}
```

Implementation

```
// crea il ViewPager Adapter
mAppSectionsPagerAdapter = new AppSectionsPagerAdapter(getSupportFragmentManager());

// SConfigurare il ViewPager, collegando l'adapter e impostando
// un listener per quando l'utente scorre tra le sezioni
mViewPager = (ViewPager) findViewById(R.id.pager);
mViewPager.setAdapter(mAppSectionsPagerAdapter);

mViewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {

        @Override
        public void onPageScrolled(int position, float positionOffset, int
positionOffsetPixels) {..}

        // Listener per reagire quando una pagina specifica è stata selezionata dall'utente
        @Override
        public void onPageSelected(int position) {..}

        @Override
        public void onPageScrollStateChanged(int state) {..}
});
```

Android & Tabs (TabLayout)

TabLayout implementa sia schede fisse che schede scorrevoli, in cui le schede non hanno una dimensione uniforme e possono scorrere lateralmente.

Le schede possono essere inserite programmaticamente

Layout XML del TabLayout posizionato sotto la barra degli strumenti

```
<com.google.android.material.tabs.TabLayout android:id="@+id/tabLayout"

android:scrollbars="horizontal"

android:layout_below="@+id/toolbar"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:background="?attr/colorPrimary" />
```

Recupera il riferimento all'oggetto e aggiungi 4 diverse schede

```
TabLayout tabLayout = (TabLayout) findViewById(R.id.tabLayout);
tabLayout.setTabGravity(TabLayout.GRAVITY_FILL);
tabLayout.addTab(tabLayout.newTab().setText("Tab 1"));
tabLayout.addTab(tabLayout.newTab().setText("Tab 2"));
tabLayout.addTab(tabLayout.newTab().setText("Tab 3"));
tabLayout.addTab(tabLayout.newTab().setText("Tab 4"));
```

dipendenze di grandle

```
dependencies {
        compile 'com.android.support:support-v4:23.1.1'
        compile 'com.android.support:appcompat-v7:23.1.1'
        compile 'com.android.support:design:23.1.1'
}
```

Listener

```

// Gestisce le callback di diversi componenti.
// Per ciascun metodo, puoi avere informazioni
```

```
// sulla scheda associata e la sua posizione tramite il parametro "Tab".
tabLayout.setOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {
        @Override
        public void onTabSelected(TabLayout.Tab tab) {
                int position = tab.getPosition();
        }
        @Override
        public void onTabUnselected(TabLayout.Tab tab) {
        }
        @Override
        public void onTabReselected(TabLayout.Tab tab) {
        }
});
```

Integrating TabLayout & ViewPager

Crea un associazione tra ViewPager e TabLayout

```
//TabLayout with ViewPager
tabLayout = (TabLayout) findViewById(R.id.tabLayout);
tabLayout.setTabGravity(TabLayout.GRAVITY_FILL);
tabLayout.setupWithViewPager(mViewPager);
```

> Note
> La libreria recupera automaticamente i nomi delle pagine e il conteggio tramite il PagerAdapter e gestisce gli eventi sia su adapter che sulle schede (tabs)

TabLayout Customization

```
// icona e testo
tabLayout.getTabAt(0).setIcon(tabIcons[0]);

// newl ViewPagerAdapter
@Override
public CharSequence getPageTitle(int position) {
        // "disattiva" il titolo
        return null;
}
```

Dialogs

Un dialogo è una piccola finestra che appare sopra all'activity corrente (popup), la classe `Dialog` è la classe base per la creazione di dialoghi.

È possibile utilizzare anche una delle seguenti sottoclassi:

- `AlertDialog`, che comunica informazioni all'utente e consente azioni rapide
- `ProgressDialog`, mostra il progresso di un'attività
- `DatePickerDialog`, consente all'utente di inserire una data
- `TimePickerDialog`, consente all'utente di inserire un'ora

I dialoghi vengono utilizzati per notifiche che devono interrompere l'utente e per eseguire brevi attività che riguardano direttamente l'applicazione in corso (come una barra di avanzamento o una richiesta di accesso).

> Nota
> È possibile personalizzare il dialogo, si può estendere l'oggetto `Dialog` di base o una delle sottoclassi elencate sopra e definire un nuovo layout

Alert Dialog

Un dialogo che può gestire zero, uno, due o tre pulsanti e/o una lista di elementi selezionabili che possono includere caselle di controllo o pulsanti radio. Un `AlertDialog` è in grado di costruire la maggior parte delle interfacce utente dei dialoghi ed è il tipo di dialogo consigliato.

```
// this indica il contesto
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setMessage("Are you sure you want to exit?")
                .setCancelable(false) // richiede la scelta dell'utente
                .setPositiveButton("Yes", new DialogInterface.OnClickListener() { //
concatenzaione
                        public void onClick(DialogInterface dialog, int id) {
```

```
                        MyActivity.this.finish();
                }
                })
                .setNegativeButton("No", new DialogInterface.OnClickListener() { //
concatenazione
                        public void onClick(DialogInterface dialog, int id) {
                                dialog.cancel();
                        }
                });
AlertDialog alert = builder.create();
alert.show();
```

È anche possibile creare un AlertDialog con una lista di elementi selezionabili utilizzando il metodo `setItems()`

```
final CharSequence[] items = {"Red", "Green", "Blue"};

AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("Pick a color");
builder.setItems(items, new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int item) {
        // handle selection …
        }
});
AlertDialog alert = builder.create();
```

Si può utilizzare i metodi `setMultiChoiceItems()` e `setSingleChoiceItems()`, è possibile creare una lista di elementi a scelta multipla (caselle di controllo) o elementi a scelta singola (pulsanti radio) all'interno del dialogo.

Progress Dialog

```
ProgressDialog dialog = ProgressDialog.show(MyActivity.this, "", "Loading. Please wait...",
true);
```

Se si desidera creare una barra di avanzamento che mostri il progresso di caricamento con maggiore precisione

```
ProgressDialog progressDialog;
progressDialog = new ProgressDialog(mContext);
progressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
progressDialog.setMessage("Loading...");
progressDialog.setCancelable(false);
// ...
progressDialog.show();
```

Si può incrementare la quantità di progresso visualizzata nella barra chiamando `setProgress(int)` con un valore che rappresenta la percentuale totale completata finora o `incrementProgressBy(int)` con un valore incrementale da aggiungere alla percentuale totale completata finora. Si può anche specificare il valore massimo in casi specifici in cui la vista percentuale non è utile.

Custom Dialog

È anche possibile personalizzare il design di un dialogo, per creare un layout per la finestra del dialogo con elementi di layout e widget.

Una volta istanziato il Dialog, si può impostare il layout personalizzato come vista del contenuto del dialogo utilizzando `setContentView(int)`, passandogli l'ID delle risorse del layout ed utilizzando il metodo `findViewById(int)` sull'oggetto del dialogo, è possibile recuperare e modificare il suo contenuto.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
                                android:id="@+id/layout_root"
                                android:orientation="horizontal"
                                android:layout_width="fill_parent"
                                android:layout_height="fill_parent"
                                android:padding="10dp">

        <ImageView android:id="@+id/image"
                                        android:layout_width="wrap_content"
                                        android:layout_height="fill_parent"
                                        android:layout_marginRight="10dp"
        />
        <TextView android:id="@+id/text"
                                android:layout_width="wrap_content"
                                android:layout_height="fill_parent"
                                android:textColor="#FFF"
        />
</LinearLayout>
```

```
Context mContext = getApplicationContext();
Dialog dialog = new Dialog(mContext);

dialog.setContentView(R.layout.custom_dialog);
dialog.setTitle("Custom Dialog");

TextView text = (TextView) dialog.findViewById(R.id.text);
text.setText("Hello, this is a custom dialog!");
ImageView image = (ImageView)
dialog.findViewById(R.id.image);
image.setImageResource(R.drawable.android);
```

Toast Notifications

Una notifica Toast è un messaggio che appare (per un periodo limitato di tempo) sulla superficie della vista attiva, occupando lo spazio necessario per il messaggio, mantendendo però la activity corrente visibile ed utilizzabile.

La notifica si dissipa automaticamente senza accettare eventi di interazione, può essere creata e visualizzata sia da un'activity che da un service.

> Nota
> Se si crea una ToastNotification da un service, essa appare di fronte all'activity attualmente in primo piano

Prima di tutto, bisogna istanziare un oggetto Toast utilizzando uno dei metodi `makeText()` che richiede tre parametri:

- Il Context dell'applicazione
- Il messaggio di testo
- La durata della notifica Toast.

E restituisce un oggetto Toast correttamente inizializzato e utilizzando il metodo `show()` per mostrarlo su schermo.

```
Context context = getApplicationContext();
CharSequence text = "Hello toast!";
int duration = Toast.LENGTH_SHORT;
Toast toast = Toast.makeText(context, text, duration);
toast.show();
```

Lo stesso risultato può essere ottenuto in un modo più breve e più semplice utilizzando una singola linea di codice

```
Toast.makeText(context, "Hello toast!", Toast.LENGTH_SHORT).show();
```

Position

Una notifica Toast standard appare vicino alla parte inferiore dello schermo, centrata orizzontalmente. Si può cambiare questa posizione utilizzando il metodo `setGravity(int, int, int)`, che accetta tre parametri:

- Una costante Gravity
- Uno spostamento dell'asse x
- Uno spostamento dell'asse y

```
toast.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0);
```

Per esempio con il codice precedente la notifica toast apparirà nell'angolo in alto a sinistra.

Custom Toast Notification

C'è anche la possibilità di creare un layout personalizzato per la notifica Toast ma non bisogna farlo.

Come per il precedente elemento dell'interfaccia utente, si può definire un layout personalizzato definendo una vista, sia in XML che nel codice dell'applicazione, e passare l'oggetto View radice al metodo `setView(View)`.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
                              android:id="@+id/toast_layout_root"
                              android:orientation="horizontal"
                              android:layout_width="fill_parent"
                              android:layout_height="fill_parent"
                              android:padding="10dp"
                              android:background="#DAAA">
       <ImageView android:id="@+id/image"
                              android:layout_width="wrap_content"
```

```
                                android:layout_height="fill_parent"
                                android:layout_marginRight="10dp"
        />
        <TextView android:id="@+id/text"
                                android:layout_width="wrap_content"
                                android:layout_height="fill_parent"
                                android:textColor="#FFF"
        />
</LinearLayout>
```

> Nota
> L'ID dell'elemento `LinearLayout` è "toast_layout", bisgna utilizzare questo ID per caricare il layout del XML

Per creare una notifica Toast personalizzata utilizzando un layout XML personalizzato, si può recuperare il `LayoutInflater` con `getLayoutInflater()` (o `getSystemService()`) e quindi infilare il layout dall'XML utilizzando `inflate(int, ViewGroup)`, dove:

- Il primo parametro è l'ID delle risorse del layout
- Il secondo parametro è la view radice

> Nota
> Si può utilizzare questo layout inserito per trovare altri oggetti View nel layout, e quindi catturare e definire il contenuto per gli elementi ImageView e TextView

Infine, creare una nuova Toast con `Toast(Context)`, impostare alcune proprietà della toast, e successivamente chiamare `setView(View)` passando il layout inserito.
Ora si può visualizzare la toast con il layout personalizzato chiamando `show()`.

```
LayoutInflater inflater = getLayoutInflater();
View layout = inflater.inflate(R.layout.toast_layout,(ViewGroup)
findViewById(R.id.toast_layout_root));

ImageView image = (ImageView) layout.findViewById(R.id.image);
image.setImageResource(R.drawable.android);
TextView text = (TextView) layout.findViewById(R.id.text);
text.setText("Hello! This is a custom toast!");

Toast toast = new Toast(getApplicationContext());
toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
toast.setDuration(Toast.LENGTH_LONG);
toast.setView(layout);
toast.show();
```

Snackbars

Gli Snackbar (simili ai Toast) forniscono un feedback leggero su un'operazione mostrando un breve messaggio nella parte inferiore dello schermo sui dispositivi mobili (e in basso a sinistra su dispositivi più grandi).

> Nota
> Questo tipo di feedback appare sopra tutti gli altri elementi sullo schermo e ne può essere visualizzato solo uno alla volta.

Gli Snackbar consentono anche di aggiungere un pulsante per azioni rapide, scompaio automaticamente dopo un certo periodo di tempo o dopo un'interazione dell'utente altrove sullo schermo o se l'utente scorre via il Snackbar dallo schermo.

La classe `Snackbar` sostituisce il Toast, che è ancora disponibile e supportato, ma non è il modo preferito per visualizzare messaggi rapidi e brevi.

grandle

```
implementation 'com.google.android.material:material:1.1.0'
```

```
Snackbar mySnackbar = Snackbar.make(view, stringId, duration);
```

La vista a cui collegare lo Snackbar, può essere la vista con cui è stata attivata la visualizzazione dello Snackbar, come un pulsante che è stato premuto o una card che è stata scorrere.

L'ID della risorsa del messaggio che si desidera visualizzare (può essere testo formattato o non formattato).

La durata per mostrare il messaggio è configurato mediante i valori accettati che sono:

- `LENGTH_SHORT`
- `LENGTH_LONG`
- `LENGTH_INDEFINITE`, che consente di mostrare lo Snackbar fino a quando non viene chiamata un'azione diretta di chiusura

Snackbars & Actions

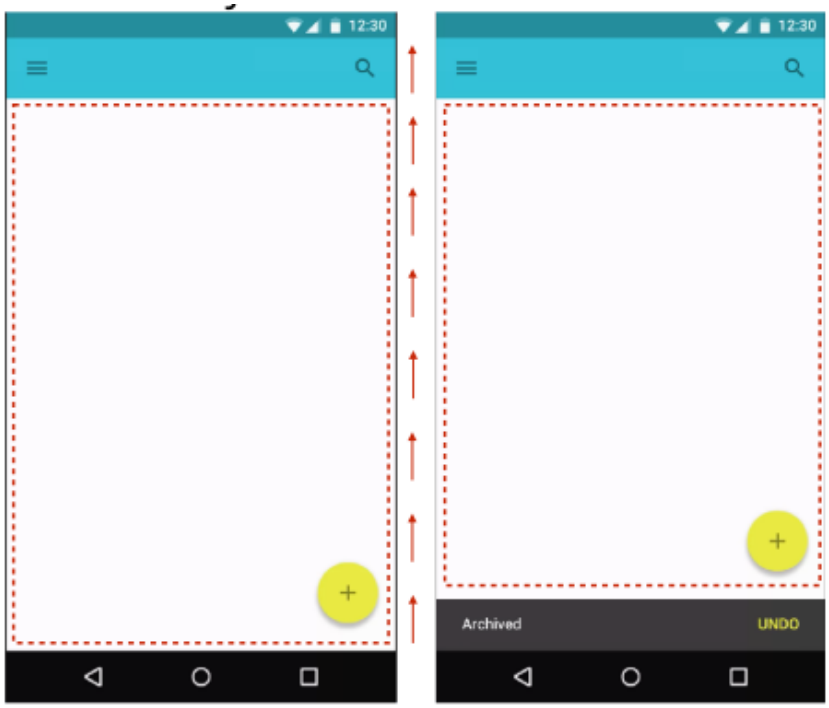
```
Snackbar mySnackbar = Snackbar.make(findViewById(R.id.myCoordinatorLayout),

R.string.email_archived, Snackbar.LENGTH_SHORT);
                                                                    // undo_string = ID

della stringa visualizzata per l'azione associata dello Snackbar
                                                                    // View.OnClickListener

per gestire il clic dell'utente sull'azione

mySnackbar.setAction(R.string.undo_string, new View.OnClickListener {
                                                                            @Override
                                                                            public void

onClick(View v) {
                                                                                    // Code

to undo the user's last action
                                                                            }
                                                                    );

mySnackbar.show();
```

Coordinator Layout

Il CoordinatorLayout è un nuovo layout introdotto con la libreria Android Design Support, che stende le capacità e le funzionalità del FrameLayout.

Se si aggiungono più figli a un FrameLayout, essi si sovrapporrebbero l'uno sull'altr, come per il FrameLayout, che dovrebbe essere utilizzato più spesso per contenere una singola vista figlia.

> Nota
> Una delle principali introduzioni fornite dal CoordinatorLayout è la capacità di coordinare le animazioni e le transizioni delle viste al suo interno (ad esempio, collegardolo a uno snackbar)




```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
                xmlns:android="http://schemas.android.com/apk/res/android"
                xmlns:app="http://schemas.android.com/apk/res-auto"
                xmlns:tools="http://schemas.android.com/tools"
                android:id="@+id/myCoordinatorLayout"
                android:layout_width="match_parent"
                android:layout_height="match_parent">
        <LinearLayout
                        android:layout_width="match_parent"
                        android:layout_height="match_parent"
```

```
                        […]
                        />
                <TextView
                                android:id="@+id/textView"
                                android:layout_width="wrap_content"
                                android:layout_height="wrap_content"
                                android:text="@string/helloSnackbar"
                                […] />
                <Button
                                android:id="@+id/triggerButton"
                                android:layout_width="wrap_content"
                                android:layout_height="wrap_content"
                                android:text="@string/triggerSnackBarButtonText"
                                […] />
        </LinearLayout>

        <com.google.android.material.floatingactionbutton.FloatingActionButton
                android:id="@+id/fab"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="bottom|right"
                android:layout_margin="20dp"
                android:src="@android:drawable/ic_popup_disk_full"/>

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

```
Snackbar.make(findViewById(R.id.myCoordinatorLayout),R.string.action,Snackbar.LENGTH_SHORT).s
how();
```