3-Graphical User Interface - part I

Official Design Principles

Keep it brief

Utilizza frasi brevi con parole semplici.

Pictures are faster than words

Considera l'uso di immagini per spiegare le idee, attirano l'attenzione delle persone e possono essere molto più efficienti delle parole.

Decide for me but let me have the final say

Fai la scelta migliore ed agisci senza esitare invece di chiedere prima, troppe scelta e decisioni rendono le persone infelici.
Nel caso in cui sbagli, prevedi la possibilità di annullare.

Only show what I need when I need it

Le persone si sentono sopraffatte quando vedono troppo in una sola volta.
Suddividi compiti e informazioni in piccoli pezzi facili da "digerire", nascondi le opzioni che non sono essenziali al momento ed insegnale alle persone man mano che procedono.

I should always know where I am

Dai alle persone la sicurezza di conoscere bene dove si trovano. Rendi distinti i luoghi nella tua applicazione e utilizza transizioni per mostrare le relazioni tra le schermate.
Fornisci feedback sulle attività in corso.

If it looks the same, it should act the same

Evita le modalità, che sono situazioni in cui gli elementi visivi appaiono simili ma agiscono in modo diverso sulla stessa input.

It's not my fault

Sii gentile nel chiedere alle persone di apportare correzioni, vogliono sentirsi intelligenti/smart quando usano la tua app.
Se qualcosa va storto, fornisci istruzioni chiare per recuperare, ma risparmia loro i dettagli tecnici.
Se puoi risolvere il problema dietro le quinte, ancora meglio.

Make important things fast

Non tutte le azioni sono uguali. Decidi cosa è più importante nella tua applicazione e rendilo facile da trovare e veloce da utilizzare.

Do not reinvent the wheel!

Non sei da solo!
In ogni istante, in qualche parte del mondo, qualcuno si trova ad affrontare gli stessi problemi di sviluppo software/design.
Sai di non voler reinventare la ruota (o una gomma a terra), quindi guarda intorno e concentra l'attenzione sugli aspetti rilevanti della tua applicazione!

Use native components and themes

Una piattaforma mobile ricca e completa come Android fornisce molti componenti nativi utili, testati e ben noti.
Dovresti utilizzarli durante le fasi di progettazione e sviluppo.

Get inspired! Sometimes copying is ok

Guardati intorno e impara dalle applicazioni esistenti!
Se non sai come progettare la tua applicazione o una sezione specifica, dai un'occhiata alle applicazioni ben note e ufficiali per capire come sono progettate e composte.

Use online free tools & resources

Se non sei un designer, affidati a strumenti e risorse esistenti per migliorare e semplificare l'interfaccia utente dell'applicazione.
Mantieni tutto semplice e chiaro.

Material Design

Material Design è una guida completa per il design visivo, del movimento e dell'interazione su diverse piattaforme e dispositivi. Creare un linguaggio visivo che sintetizzi i principi classici del buon design con l'innovazione e le possibilità della tecnologia e della scienza.

Sviluppare un singolo sistema sottostante che consenta un'esperienza unificata su diverse piattaforme e dimensioni dei dispositivi, i principi per i dispositivi mobili sono fondamentali, ma il tocco, la voce, il mouse e la tastiera sono tutti metodi di input di prima classe

Enviroment

L'ambiente materiale è uno spazio tridimensionale, e tutti gli oggetti hanno le dimensioni x, y e z. L'asse z è perpendicolare al piano del display

> Nota
> Se è positivo si estende che va verso lo spettatore (uscente dallo schermo)

Ogni foglio di materiale occupa una singola posizione lungo l'asse z e ha uno spessore standard di `1dp`.
All'interno dell'ambiente di materiale, le luci virtuali illuminano la scena e consentono agli oggetti di proiettare ombre.

Specifications

Il Material Design definisce ogni aspetto del design dell'applicazione:

- Stile (colore, icone, immagini, tipografia)
- Layout
- Componenti
- Patterns (Errori, gesture, formato dei dati)
- Usabilità
- Animazioni

Main components and features

Android fornisce i seguenti elementi per costruire app con il design materiale:

- Un nuovo tema con la ridisegnazione e l'adattamento dei principali elementi e modelli di sviluppo esistenti di Android, come:
  - ActionBar/ToolBar
  - Tabs & ViewPager
  - Navigation Drawer
- Nuovi widget per visualizzazioni complesse:
  - Lists
  - Cards
- Nuove API per ombre e animazioni personalizzate

Setup

Il material theme è definito come:

- Versione scura, `@android:style/Theme.Material`
- Versione chiara, `@android:style/Theme.Material.Light`

> Nota
> È disponibile da Android 5.0 (API 21) e successivi, per includere e utilizzare temi con stili di material design nelle versioni precedenti, è necessario utilizzare le `v7 Support Libraries`

Customize the color palette

I colori di base del tema possono essere personalizzati per adattarsi al marchio o alle preferenze dell'applicazione, è possibile specificare i colori personalizzati utilizzando gli attributi del tema nel file `style.xml` o creando direttamente un nuovo tema ereditato da `material` nel file `themes.xml`

Esempio

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <style name="AppTheme" parent="android:Theme.Material">
                <item name="android:colorPrimary">
                        @color/primary
                </item>
                <item name="android:colorPrimaryDark">
                        @color/primary_dark
                </item>
                <item name="android:colorAccent">
                        @color/accent
                </item>
        </style>
</resources>
```

Maintaining Compatibility

Per consentire l'utilizzo delle ultime progettazioni su piattaforme più datate, Google ha ampliato le `Support Libraries v7 (r21)`, includendo un importante aggiornamento di AppCompat, oltre alle nuove librerie `RecyclerView`, `CardView` e `Palette`.

AppCompact era un backport dell'API `ActionBar` di Android 4.0 per dispositivi che eseguono Gingerbread, fornendo uno strato API comune sopra l'implementazione retroportato e l'implementazione del framework.

È possibile configurare un applicazione per utilizzare il tema material seguendo questi passi:

1. Definire un tema che eredita da un tema precedente (come `Holo`) nel file `res/values/styles.xml`
2. Definire un tema con lo stesso nome che eredita dal tema dei materiali nel file `res/values-v21/styles.xml`
3. Impostare questo tema come tema dell'applicazione nel file `manifest`

Le transizioni di Lollipop non sono retrocompatibili, quindi al momento si potrebbe non essere in grado di vedere transizioni fluide su dispositivi pre-21

Esempio

file `res/values/style.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <style name="AppBaseTheme" parent="Theme.AppCompany.Light">
                <!--
                        stile
                -->
        </style>
</resources>
```

file `res/values/themes.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <style name="AppTheme" parent="AppTheme.Base"/>
        <style name="AppTheme.Base" parent="Theme.AppCompany.Light">
                <item name="colorPrimary">
                        @color/primary
                </item>
                <item name="colorPrimaryDark">
                        @color/primary_dark
                </item>
                <item name="android:textColorPrimary">
                        @color/blue_grey_100
                </item>
                <item name="android:windowNoTitle">
                        true
                </item>
                <item name="windowActionBar">
                        true
                </item>
        </style>
</resources>
```

file `res/values-21/themes.xml`

```
<resources>
        <style name="AppTheme" parent="AppTheme.Base"/>
                <item name="android:windowContentTransition">
                        true
                </item>
                <item name="android:windowAllowEnterTransitionOverlap">
                        true
                </item>
                <item name="android:windowAllowReturnTransitionOverlap">
                        true
                </item>
                <item name="android:wondowSharedElementEnterTransition">
                        @android:transition/move
                </item>
                <item name="android:wondowSharedElementExitTransition">
                        @android:transition/move
                </item>
        </style>
</resources>
```

file `AndroidManifest.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
        xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.mobdev.hellomaterial"
        android:versionCode="1"
        android:versionName="1.0" >
        [..]
        <application
                android:allowBackup="true"
                android:icon="@drawable/ic_launcher"
                android:label="@string/app_name"
                android:theme="@style/AppTheme" >

                <activity
                        android:name=".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>
        </application>
</manifest>
```

file `/res/values/colors.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <color name="red_50">#fde@dc</color>
        <color name="red_100">#f9bdbb</color>
        <color name="red_200">#f69988</color>
        <color name="red_300">#f36c60</color>
        <color name="red_400">#e84e40</color>
        <color name="red_500">#e51c23</color>
        <color name="red_600">#dd191d</color>
        <color name="red_700">#d01716</color>
        <color name="red_800">#c41411</color>
        <color name="red_900">#b0120a</color>
        <color name="red_A100">#ff7997</color>
        <color name="red_A200">#ff5177</color>
        <color name="red_A400">#ff2d6f</color>
        <color name="red_A700">#e00032</color>
        [...]
</resources>
```

Activity

Un'activity è sia un'unità di interazione dell'utente (tipicamente associata a una View) che un'unità di esecuzione, all'interno del metodo sovrascritto `onCreate(...)`, lo sviluppatore può impostare la View di layout associata all'activity chiamando `setContentView(R.layout.<nome_file_layout>)`, utilizzando l'oggetto condiviso R per specificare il file di risorse contenente il layout.

Una volta impostata la vista radice, è possibile recuperare gli oggetti associati agli elementi dell'interfaccia utente tramite il metodo `findViewById(R.id.<ID_risorsa>)`.

Un oggetto view come `TextView` può essere utilizzato per aggiornare l'interfaccia utente, ottenere valori dai campi di input o impostare listener degli eventi utilizzando i metodi forniti per il recupero e l'impostazione dei valori.

Esempio

vista

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/
                            apk/res/android"
                            android:layout_width="fill_parent"
                            android:layout_height="fill_parent"
                            android:orientation="vertical" >

        <TextView
        android:id="@+id/mainTextView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />

</LinearLayout>
```

controller

```
public class HelloWorldActivity extends Activity {
        /** Called when the activity is first created. */
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        }
}
```

accesso e modifica all'elemento della UI

```
TextView mainTextView = (TextView)findViewById(R.id.mainTextView);
mainTextView.setText("Hello World ! :D");
```

Your Activity Class

```
public class HelloWorldActivity extends Activity {
        /** Called when the activity is first created. */
        @Override public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                (setContentView(R.layout.main);
        }
}
```

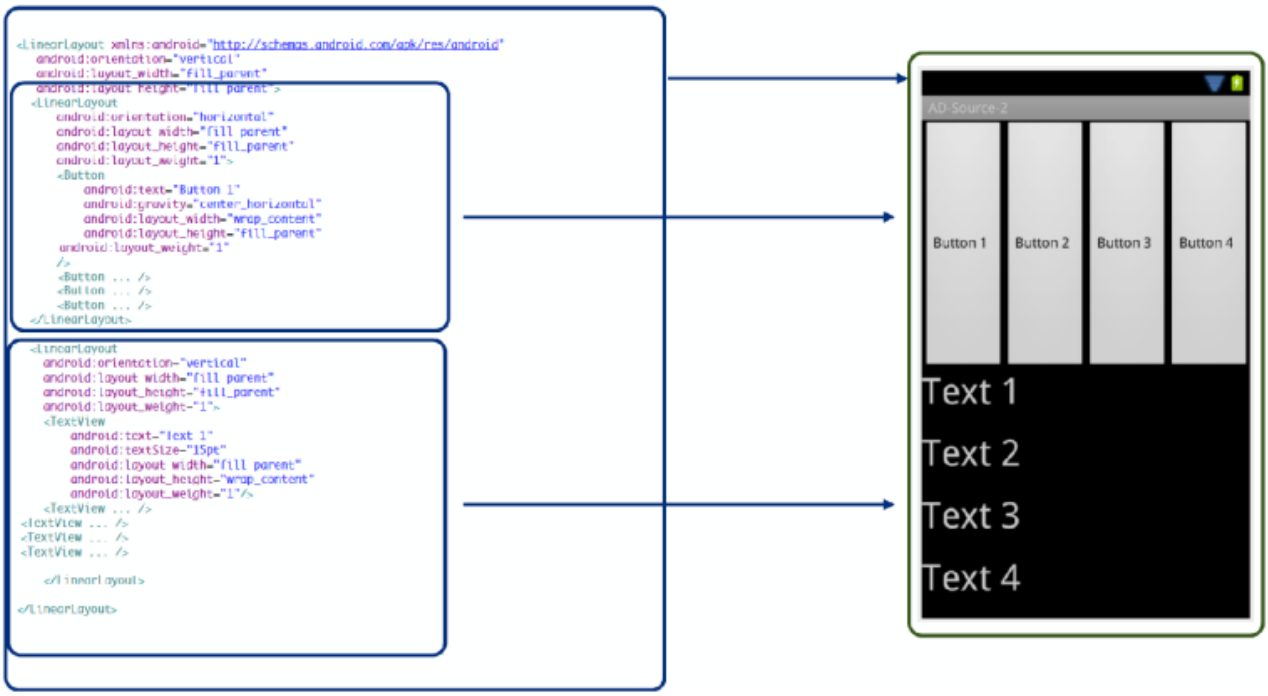
Layout

Linear Layout

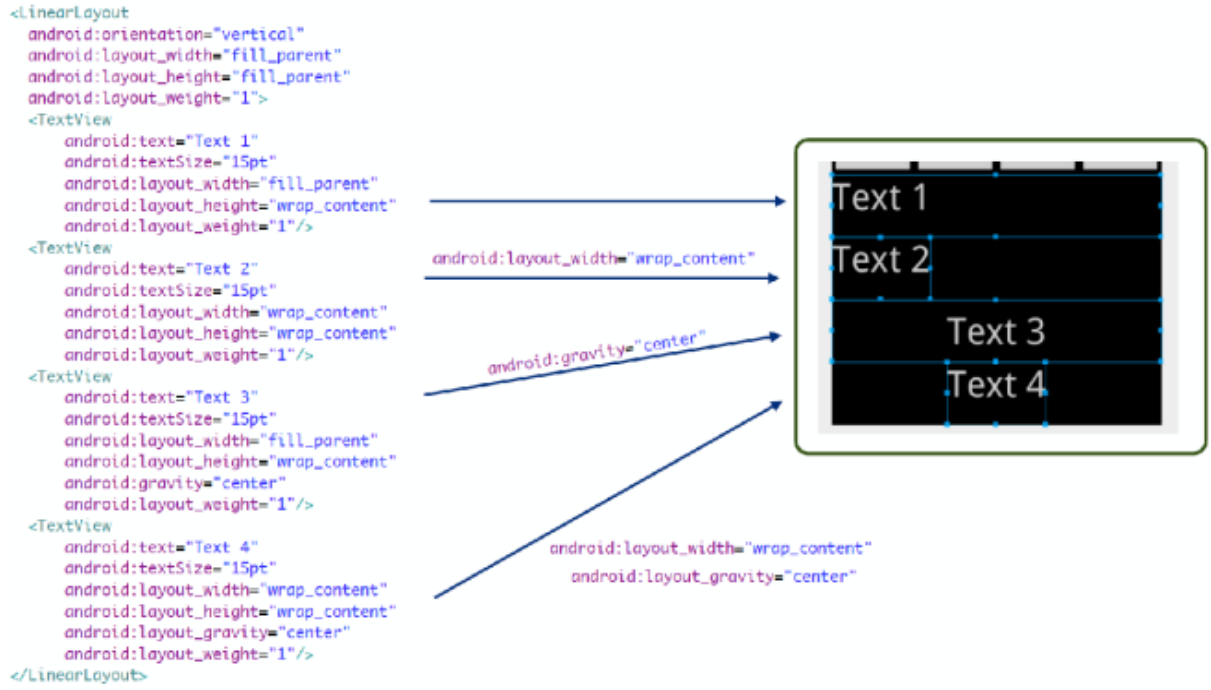
Un layout che dispone i suoi figli in una singola colonna o una singola riga è chiamato `LinearLayout`, lo spazio disponibile viene diviso tra i figli del layout, che vengono disposti in sequenza.
La dimensione degli elementi figli può essere specificata utilizzando attributi come `layout_weight` o `layout_width`/`layout_height`.

La direzione della riga può essere impostata chiamando `setOrientation()` nel codice o utilizzando l'attributo `android:orientation="vertical"` nel file XML del layout, se si imposta la direzione a `"vertical"`, gli elementi figli verranno disposti in una singola colonna, se si imposta la direzione a `"horizontal"`, gli elementi figli verranno disposti in una singola riga.

> Nota
> L'orientamento di default è orizzontale



Parametri



Relative Layout

Il RelativeLayout è un ViewGroup che mostra gli elementi view figli in posizioni relative, la posizione di una view può essere specificata in:

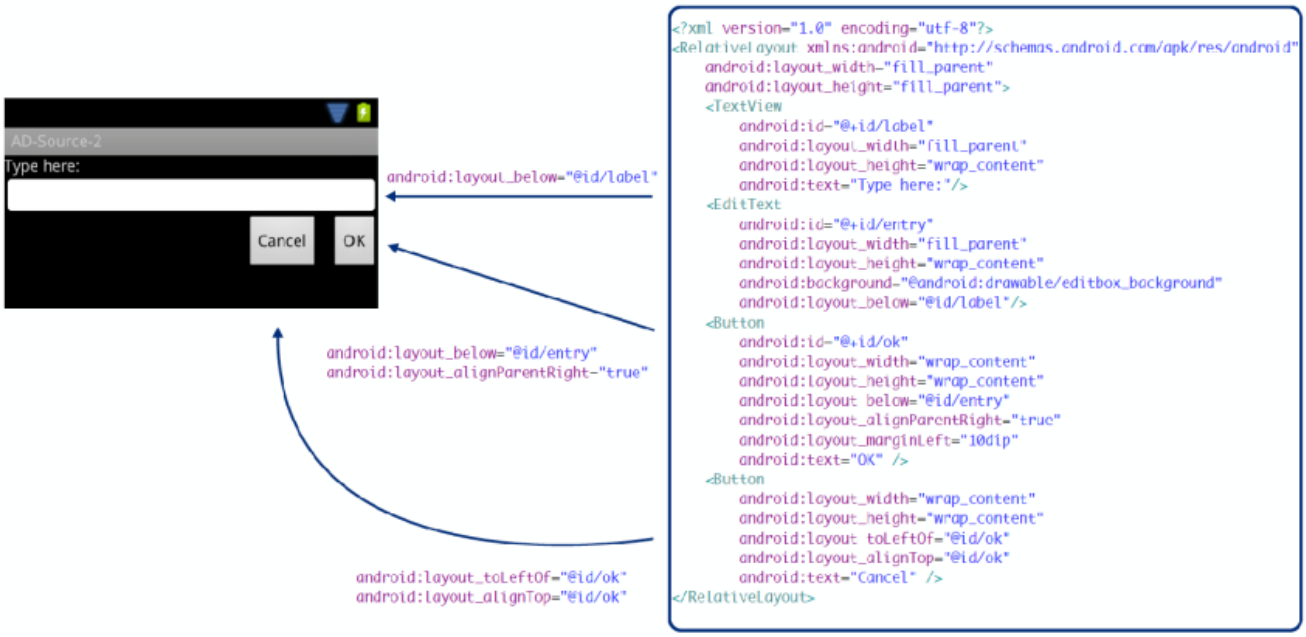
- Relazione agli elementi fratelli (ad esempio a sinistra di o sotto un determinato elemento)
- In posizioni relative all'area RelativeLayout (ad esempio allineato in basso, a sinistra o al centro)

> Nota
> Un RelativeLayout è un'utilità molto potente per progettare un'interfaccia utente poiché può eliminare i gruppi di View nidificati, se ci si trova a utilizzare diversi gruppi LinearLayout nidificati, potresti essere in grado di sostituirli con un singolo RelativeLayout

Quando si utilizza un RelativeLayout, è possibile utilizzare gli attributi `android:layout_*`, come `layout_below`, `layout_alignParentRight` e `layout_toLeftOf`, per descrivere come si desidera posizionare ciascuna view, ciascuno di questi attributi definisce un tipo diverso di posizione

relativa.

> Nota
> Alcuni attributi utilizzano l'ID della risorsa di una view sibling per definire la propria posizione relativa



Frame Layout

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
                                xmlns:tools="http://schemas.android.com/tools"
                                android:id="@+id/container"
android:layout_width="match_parent"
                                android:layout_height="match_parent"
                                tools:context="com.mobdev.hellofragment.MainActivity"
                                tools: ignore="MergeRoot Frame" />
```

FrameLayout è progettato per occupare un'area dello schermo e visualizzare un singolo elemento, infatti i generale dovrebbe essere utilizzato per rappresentare una singola view figlia.

> Nota
> Quando il FrameLayout opera più figli può essere utilizzato per sovrapporli

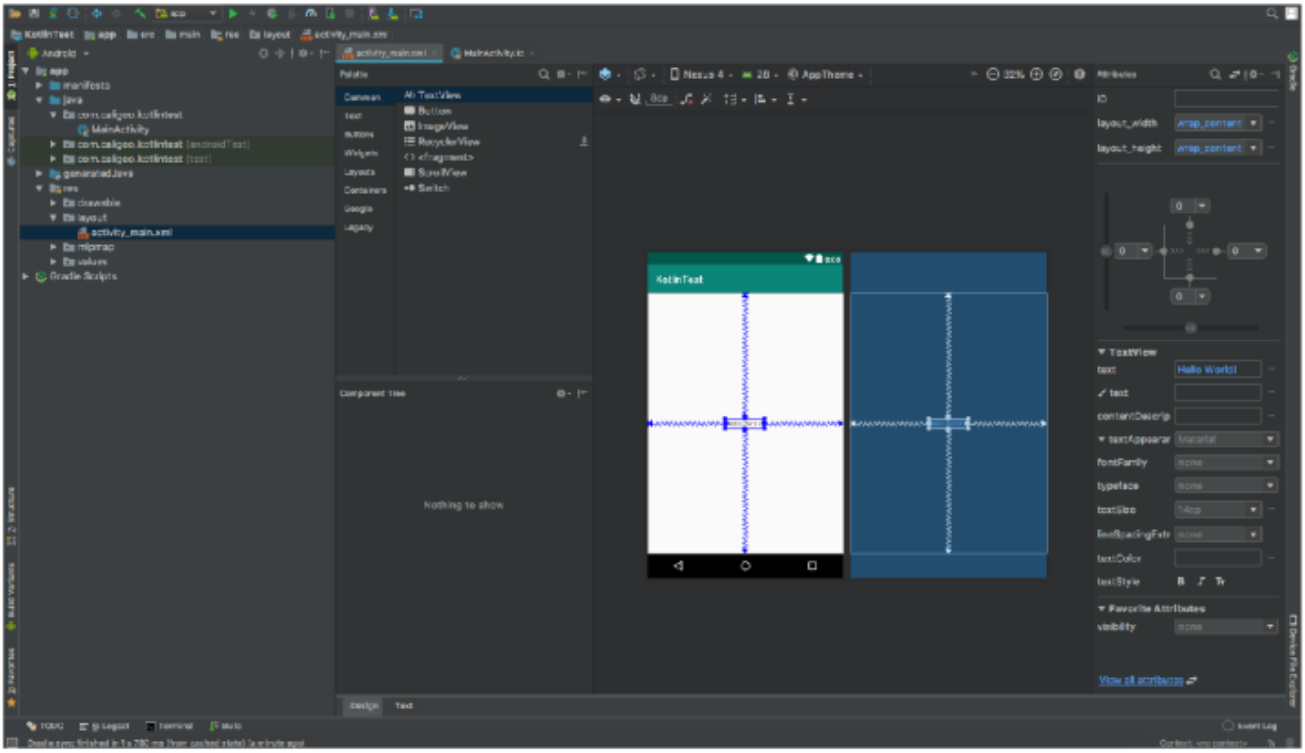
È particolarmente utile come contenitore generico per un Fragment senza la necessità di definire file XML multipli.

Constraint Layout

Il ConstraintLayout consente di creare layout ampi e complessi con una gerarchia di visualizzazione piatta (senza gruppi di visualizzazione nidificati), è simile, ma più flessibile e facile da usare con l'Editor di Layout di Android Studio, a RelativeLayout in quanto tutte le viste vengono posizionate in base alle relazioni tra le viste fratello e il layout principale.

Tutta la potenza di ConstraintLayout è disponibile direttamente dagli strumenti visuali dell'Editor di Layout. Puoi costruire il tuo layout con ConstraintLayout completamente trascinandolo senza dover modificare l'XML.

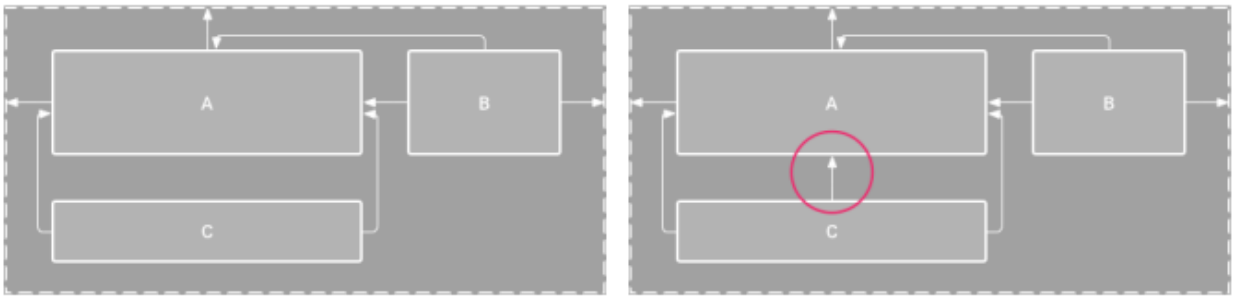
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
                                xmlns:android="http://schemas.android.com/apk/res/android"
                                xmlns:app="http://schemas.android.com/apk/res-auto"
                                android:layout_width="match_parent"
                                android:layout_height="match_parent">
        [...]
</android.support.constraint.ConstraintLayout>
```

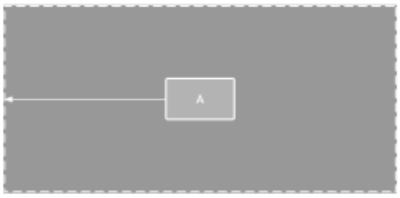


Constraints overview

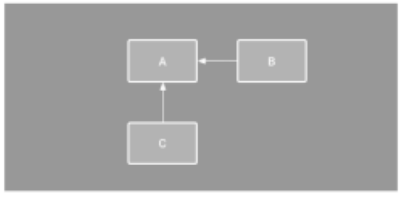
Per la posizione di una vista in ConstraintLayout, è necessario aggiungere almeno un vincolo orizzontale e uno verticale alla vista, ogni vincolo rappresenta una connessione o un allineamento con un'altra vista, il layout padre o una guida invisibile.

> Nota
> Ogni vincolo definisce la posizione della vista lungo l'asse verticale o orizzontale, ogni vista deve avere almeno un vincolo per ciascun asse, ma spesso ne sono necessari di più

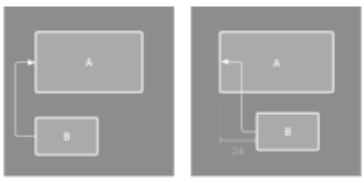




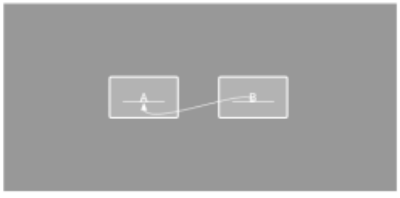
Parent position



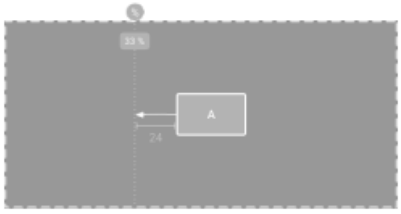
Order position



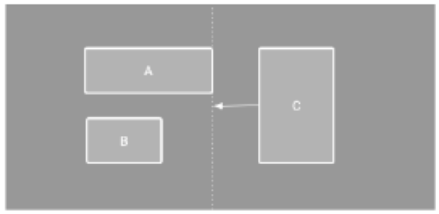
Alignment



Baseline alignment



Constraint to a guideline



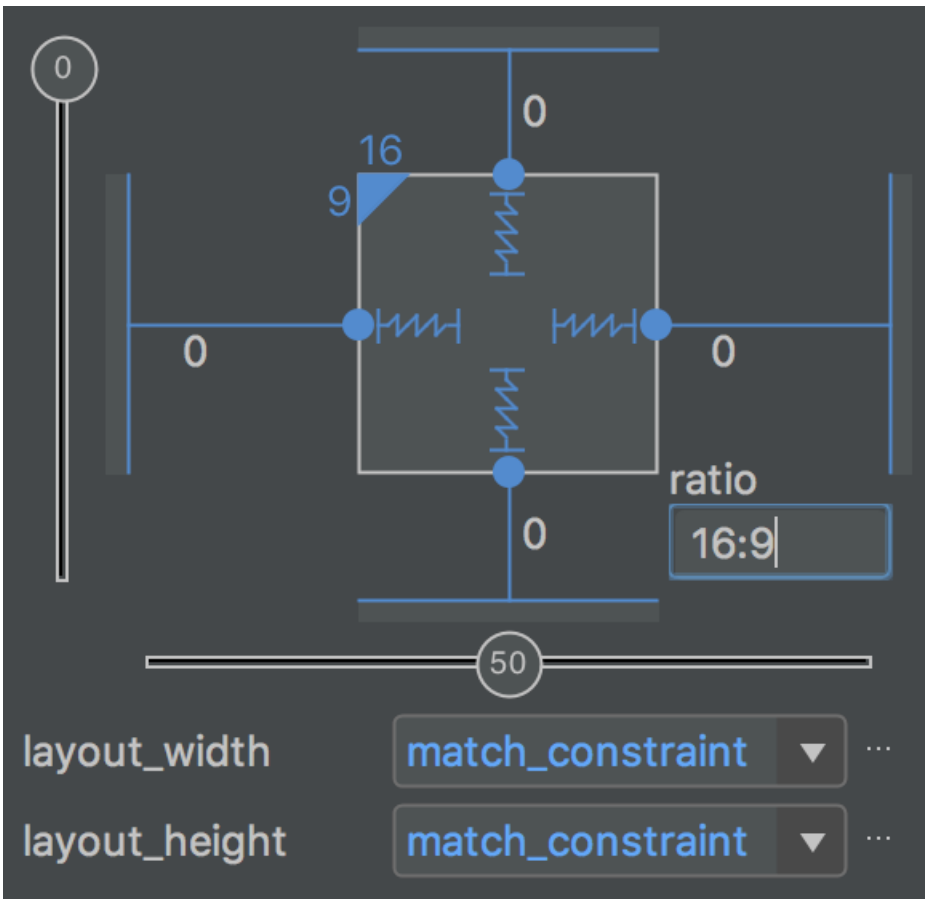
Constraint to a barrier

Adjust the view size

Puoi utilizzare gli handler agli angoli per ridimensionare una vista, ma questo imposta rigidamente le dimensioni, quindi la vista non si ridimensionerà in base a diversi contenuti o dimensioni dello schermo. Per selezionare una modalità di dimensionamento diversa, clicca sulla vista e apri la finestra degli attributi sul lato destro dell'editor.

Puoi cambiare il modo in cui vengono calcolate l'altezza e la larghezza:

- Fixed, si può specificare una dimensione specifica nella casella di testo sottostante o ridimensionando la vista nell'editor
- Wrap Content, la vista si espande solo quanto è necessario per adattarsi ai suoi contenuti
- Match Constraints, la vista si espande il più possibile per soddisfare i vincoli su ciascun lato (dopo aver considerato i margini della vista), tuttavia si può modificare questo comportamento con gli attributi e i valori seguenti:
  - `layout_constraintWidth_default`,. specifica il comportamento predefinito della larghezza della vista
    - `spread`, la vista si espande per riempire lo spazio disponibile tra i vincoli
    - `wrap`, la vista si espande solo quanto necessario per adattarsi ai suoi contenuti
    - `percent`, la larghezza della vista è specificata come percentuale della larghezza del genitore
  - `layout_constraintWidth_min`, specifica la larghezza minima della vista
  - `layout_constraintWidth_max`, specifica la larghezza massima della vista
  - `layout_constraintWidth_percent`, specifica la larghezza della vista come percentuale della larghezza del genitore



ImageView

```
<ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView1"
        android:layout_centerHorizontal="true" android:layout_marginTop="14dp"
        android:src="@drawable/ic_action_android" />
```

A runtime

```
ImageView imageView = (ImageView)findViewById(R.id. imageView);
imageView.setImageDrawable(androidDrawable);
imageView.setImageBitmap(bm);
imageView.setImageResource(R.drawable.ic_action_android);
```

Eventi di input

Le view/widget e in generale l'interfaccia utente (UI) vengono utilizzati non solo per mostrare contenuti e informazioni, ma hanno anche un ruolo importante nel consentire all'utente di interagire con l'applicazione eseguendo azioni.
Per essere informato degli eventi di input dell'utente, lo sviluppatore deve fare una delle due cose:

- Definire un listener degli eventi e registrarlo con la vista, la classe `View` contiene una collezione di interfacce nidificate chiamate `OnSomeEventListener`, ognuna con un metodo di callback chiamato `onSomeEvent()`
  - Ad esempio `View.OnClickListener` per gestire i "click", `View.OnTouchListener` per gestire i tocchi dello schermo, `View.OnKeyListener` per gestire la pressione di una tasto
  - Per essere notificati quando la vista viene "cliccata", bisogna implementare l'interfaccia `OnClickListener` e definire il suo metodo di callback `onClick()`, quindi registrarlo nella view utilizzando `setOnClickListener()`.
- Sovrascrivere un metodo di callback esistente per la view

> Nota
> Il secondo punto descrive ciò che si dovrebbe fare quando si implementa la propria classe `View` personalizzata e si desideri ascoltare eventi specifici che si verificano al suo interno

```

// Create an anonymous implementation of OnclickListener
private OnClickListener mCorkyListener = new OnClickListener() {
        public void onClick(View v) {
                // ...
        }
};

protected void onCreate(Bundle savedValues) {
        // Capture our button from layout
        Button button = (Button) findViewById(R.id.corky);
        // Register the onClick listener with the implementation above
button.setOnClickListener (mCorkyListener);
}
```

Bottoni

```
<Button
        android:id="@+id/randomButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/infoTextView" android:layout_centerHorizontal="true"
        android:text="Click me to get a random number !" android:textColor="@color/primary"
/>
```

```
// Recupera un oggetto Button dall'interfaccia utente (UI) e imposta un listener inline.
Button randomButton = (Button)findViewById(R.id.randomButton);
```

Button Events

Android si basa massivamente sul principio/design pattern di callback. I metodi di questo tipo sono molto utili per gestire eventi che possono occorrere in un momento non specifico.

> Nota
> Un metodo di callback è un metodo che viene passato come argomento di un altro metodo per essere successivamente invocato

In Java i metodi di callback sono implementati mediante interfacce

```
public class MyActivity extends Activity {
        protected void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.content_layout_id);


        final Button button = (Button) findViewById(R.id.button_id);

        button.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {
                // Perform action on click
                }
        });
}
```

ImageButton

```
<ImageButton
        android:id="@+id/imageButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/infoButton"
        android:layout_centerHorizontal="true"
        android:src="@drawable/ic_launcher" />
```


```
ImageButton imageButton = (ImageButton)findViewById(R.id.imageButton);
imageButton.setImageDrawable(androidDrawable);
imageButton.setImageBitmap(bm);
imageButton.setImageResource(R.drawable.ic_launcher);
imageButton.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
                // ...
        }
});
```

Launch/Finish Activity

Il metodo `startActivity(Intent)` viene utilizzato per avviare una nuova attività, che verrà posizionata in cima allo stack delle attività.

> Nota
> Il metodo prende un singolo argomento, un intent, che descrive l'attività da eseguire

```
startActivity(new Intent(getApplicationContext(), MyActivity.class));
```

A volte si desidera ottenere un risultato da un'attività quando terminal, ad esempio è possibile avviare un'attività che permette all'utente di selezionare una persona da un elenco di contatti, quando l'attività termina restituisce la persona selezionata. Per fare ciò, si chiama il metodo `startActivityForResult(Intent, int)` con un secondo parametro intero che identifica la chiamata, il risultato ottenuto verrà restituito attraverso il metodo `onActivityResult(int, int, Intent)`.

Quando un'attività termina, può chiamare `setResult(int)` per restituire dati al suo genitore (attività chiamante).

> Nota
> Deve essere sempre fornito un codice di risultato, che può essere uno dei risultati standard `RESULT_CANCELED`, `RESULT_OK` o qualsiasi altro valore personalizzato che inizia con `RESULT_FIRST_USER`

Inoltre opzionalmente può restituire un intent contenente dati aggiuntivi desiderati.

> Nota
> Tutte queste informazioni verranno visualizzate nel metodo `onActivityResult()` dell'attività genitore, insieme all'identificatore intero che è stato fornito originariamente

Quando l'attività ha terminato il suo compito, lo sviluppatore può chiamare il metodo `finish()` per chiuderla, l'`ActivityResult` viene quindi trasmesso a chiunque l'abbia lanciata tramite `onActivityResult()`.

Start an Activity with parameters

```
Bundle bundle = new Bundle();
bundle.putInt("bookmarkPosition", position);

Intent newIntent = new Intent(getApplicationContext(), EditBookmarkActivity.class);
newIntent.putExtras(bundle);
startActivityForResult(newIntent);
```

Utilizzando il metodo `startActivity(Intent)` o `startActivityForResult(Intent, result)`, è possibile avviare una nuova attività, che verrà posizionata in cima allo stack delle attività.

Con la seconda versione, è possibile specificare un parametro intero aggiuntivo che identifica la chiamata, il risultato verrà restituito attraverso il metodo `onActivityResult(int, int, Intent)`.

> Nota
> La classe `Bundle` è una mappa chiave/valore che consente di specificare parametri associati a un intent (attraverso il metodo `putExtras(bundle)`) che si desidera inviare alla nuova activity

Nella activity di destinazione, si può recuperare il Bundle in arrivo ed estrarre i valori inviati utilizzando la giusta chiave:

```
Bundle bundle = this.getIntent().getExtras();
bookmarkPosition = bundle.getInt("bookmarkPosition");
```

Fragments

Un fragment è una porzione dell'interfaccia utente in un'activity con un comportamento specifico e potenzialmente diverso, è possibile:

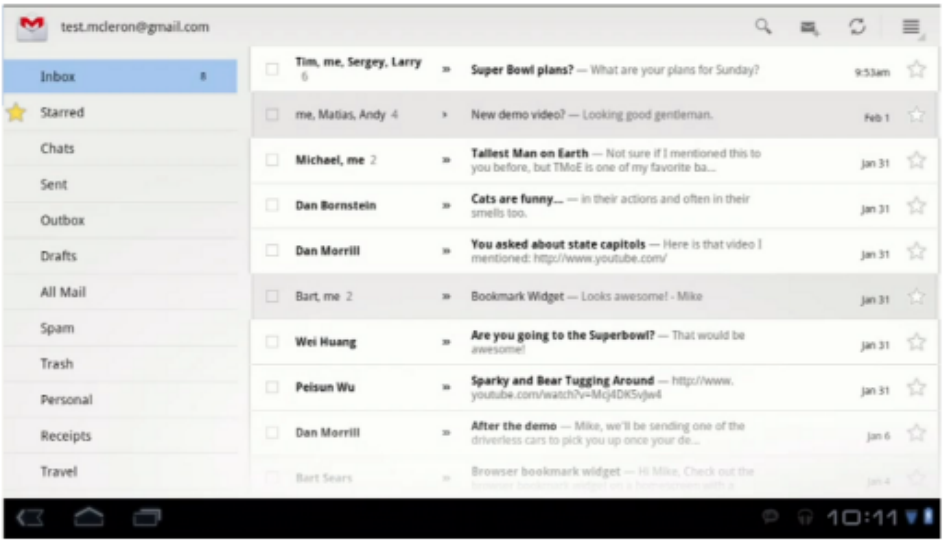
- Combinare più frammenti in un'unica attività per creare un'interfaccia utente con più pannelli
- Riutilizzare un frammento in più attività

Essenzialmente un fragment può essere visto come una sezione modulare di un'attività che:

- Ha il proprio ciclo di vita
- Riceve i propri eventi di input
- Può essere aggiunto o rimosso dinamicamente mentre l'attività è in esecuzione

Philosophy

Consideriamo questo esempio:



Un'applicazione di posta elettronica può utilizzare un frammento per mostrare l'elenco delle e-mail ricevute a sinistra e un altro frammento per visualizzare l'e-mail selezionata a destra, entrambi i frammenti appaiono in una sola attività, affiancati l'uno all'altro.

> Nota
> Ogni frammento ha il proprio set di metodi di callback del ciclo di vita e gestisce i propri eventi di input dell'utente

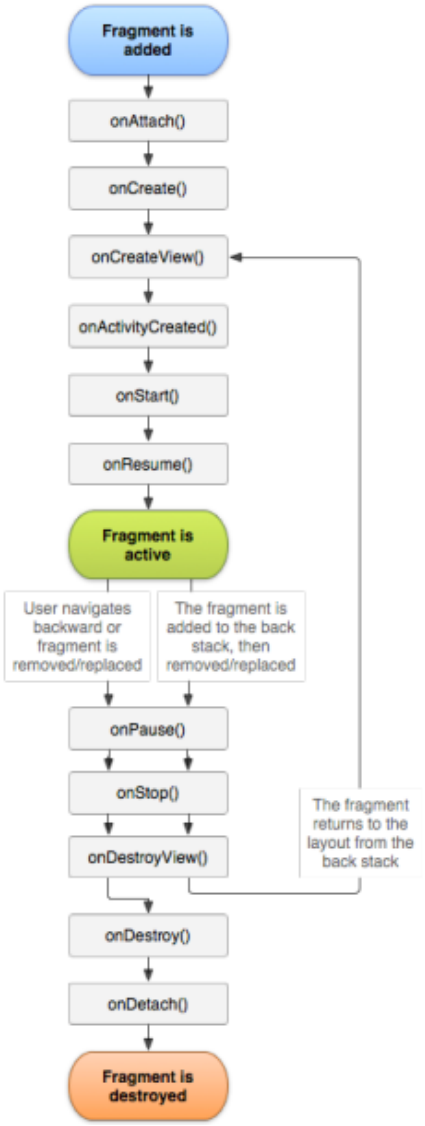
Utilizzando i frammenti, l'utente può (nella stessa attività) aggiornare la lista delle email mentre sta leggendo un messaggio già selezionato senza cambiare l'interfaccia utente.

Lifecycle

Per creare un frammento, è necessario creare una sottoclasse di `Fragment` (o utilizzare una sottoclasse esistente).

La classe `Fragment` è simile a una classe `Activity`, contiene metodi di callback come `onCreate()`, `onStart()`, `onPause()` e `onStop()`. È possibile definire il comportamento di un fragment ma è necessario implementare almeno i seguenti metodi del ciclo di vita:

- `onCreate()`, metodo chiamato durante la creazione del frammento, bisognerebbe inizializzare i componenti essenziali del frammento che si desidera conservare quando il frammento viene messo in pausa o interrotto, per poi riprenderne l'esecuzione
- `onCreateView()`, metodo chiamato dal sistema quando è il momento per il frammento di disegnare la sua interfaccia utente per la prima volta. Per fare ciò il frammento deve restituire una view, attraverso questo metodo, che rappresenta il layout principale del frammento. È possibile restituire `null` se il frammento non fornisce un'interfaccia utente
- `onPause()`, metodo dal sistema come primo feedback che l'utente sta lasciando il frammento (anche se non significa sempre che il frammento sta per essere distrutto). Questo è generalmente il punto in cui si dovrebbe salvare eventuali modifiche che devono essere conservate oltre la sessione utente corrente



Code

```
public class ExampleFragment extends Fragment {
        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
                // Inflate the layout for this fragment
                View view = inflater.inflate(R.layout.example_fragment, container, false);
                return view;
        }
}
```

Adding a fragment to an Activity using XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
                              android:orientation="horizontal"
                              android:layout_width="match_parent"
                              android:layout_height="match_parent">
        <fragment android:name="com.example.news.ArticleListFragment"
                              android:id="@+id/list"
                              android:layout_weight="1"
                              android:layout_width="0dp"
                              android:layout_height="match_parent" />
        <fragment android:name="com.example.news.ArticleReaderFragment"
                              android:id="@+id/viewer"
                              android:layout_weight="2"
                              android:layout_width="0dp"
                              android:layout_height="match_parent" />
</LinearLayout>
```

Adding a fragment to an Activity programmatically

Mentre un'attività è in esecuzione, si può aggiungere uno o più frammenti al layout esistente dell'attività. Per fare ciò bisogna semplicemente specificare un ViewGroup in cui posizionare il frammento.

> Nota
> La gestione e l'interazione tra un'attività e un frammento sono definite e controllate dalla classe FragmentManager.

Per effettuare transazioni di frammenti in un'attività (aggiungere, rimuovere o sostituire un frammento), bisogna utilizzare le API di `FragmentTransaction`.

```
FragmentManager fragmentManager = getFragmentManager();
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
ExampleFragment fragment = new ExampleFragment();
fragmentTransaction.add(R.id.fragment_container, fragment);
fragmentTransaction.commit();
```

La transazione deve essere avviata utilizzando `beginTransaction()`, si può quindi aggiungere un frammento usando il metodo `add()`, specificando il frammento da aggiungere e la vista in cui inserirlo.
Il primo argomento passato a `add()` è il ViewGroup in cui il frammento dovrebbe essere posizionato, specificato tramite l'ID della risorsa, e il secondo parametro è il frammento da aggiungere.
Una volta apportate le modifiche con `FragmentTransaction`, bisogna chiamare `commit()` affinché le modifiche abbiano effetto.

Application communication with Fragments

Due frammenti diversi non possono comunicare direttamente poiché potrebbero non essere presenti contemporaneamente nel layout dell'attività, pertanto, essi comunicano tramite l'attività a cui appartengono. Sebbene un frammento sia implementato come un oggetto indipendente da un'attività e possa essere utilizzato all'interno di più attività, una data istanza di un frammento è direttamente legata all'attività che lo contiene, in particolare il frammento può accedere all'istanza dell'attività utilizzando `getActivity()` e facilmente svolgere compiti come trovare una vista nel layout dell'attività.

```
View listView = getActivity().findViewById(R.id.list);
```

Allo stesso modo, un'attività può chiamare i metodi nel frammento acquisendo un riferimento al frammento dal FragmentManager, utilizzando `findFragmentById()` o `findFragmentByTag()`.

```
ExampleFragment fragment =
(ExampleFragment)getFragmentManager().findFragmentById(R.id.example_fragment);
```

Fragments & Android Compatibility

Android 3.0 (Honeycomb) introdusse alcune modifiche fondamentali all'interfaccia utente, soprattutto attraverso l'introduzione dell'API Fragment.

Per supportare le versioni precedenti di Android, Google ha rilasciato una libreria chiamata "AndroidX" (precedentemente noto come "Android Support Library" o "Android Compatibility package"). Questa libreria fornisce il supporto per l'API dei frammenti e altre importanti nuove funzionalità per dispositivi che risalgono fino ad Android 1.6.

> Nota
> Per supportare i frammenti nelle versioni precedenti, bisognerebbe utilizzare la libreria di supporto dei frammenti

Ecco i passaggi:

1. Cambia l'attività in modo che estenda la classe FragmentActivity quando vuoi utilizzare i frammenti
2. Cambia la chiamata al metodo `getFragmentManager()` in `getSupportFragmentManager()`
3. Organizza o aggiorna gli import e il codice, `import android.support.v4.app.Fragment;`, `import android.support.v4.app.Fragment;TabHost`

Access UI Components (Activity vs Fragment)

```
<Button
        android:id="@+id/randomButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
/>
```

Activity

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.single_fragment_activity);
        //Retrieve a Button UI Object
        Button randomButton = (Button)findViewById(R.id.randomButton);
}
```

Fragment

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View view = inflater.inflate(R.layout.random_number_fragment, container, false);
        Button randomButton = (Button)view.findViewById(R.id.randomButton); // invocato sulla
vista non sull'attività
        return view;
}
```

Access Context Reference

Puoi accedere alle risorse del bundle dell'applicazione utilizzando l'oggetto `Context`.

> Nota
> Le classi `Activity`, `Service`, `BroadcastReceiver` e `ContentProvider` estendono `Context`

All'interno di un activity si può fare una cosa tipo

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.single_fragment_activity);
        Context context = this;
}
```

> Nota
> La classe `Fragment` non estende `Context`

All'interno di un fragment bisogna riferirsi all'activity padre per riuscire a riferirsi al context

```
Context context = getActivity();
```

Access Resources

Quando hai un riferimento al Context della tua applicazione, puoi recuperare le risorse disponibili (stringe disegnabili, colori, assets, configurazioni, layouts, ecc..).

Disegnabile

```
Context context = getActivity(); //if inside a fragment
Drawable androidDrawable = context.getResources().getDrawable(R.drawable.ic_action_android);
imageView.setImageDrawable(androidDrawable);
```

Stringa

```
Context context = getActivity(); //if inside a fragment
String helloWorldString = context.getResources().getString(R.string.hello_world);
textView.setText(helloWorldString);
```