Implementare una interfaccia utente

Introduzione

In questa lezione vedremo come implementare una interfaccia utente sullo schermo di Android.

Vedremo gli elementi di base visualizzabili sullo schermo ed i layout che permettono di disporre i vari elementi dell'interfaccia grafica.

Gerarchia di elementi dello schermo

- Come visto più volte durante questo corso l'elemento di base di una applicazione sono gli oggetti della classe android.app.Activity.
- Una Activity può fare molte cose, ma di per sé non ha associata una schermata.
- Per assegnare ad una activity una interfaccia grafica si lavora con oggetti del tipo View e Viewgroup.

android.view.View

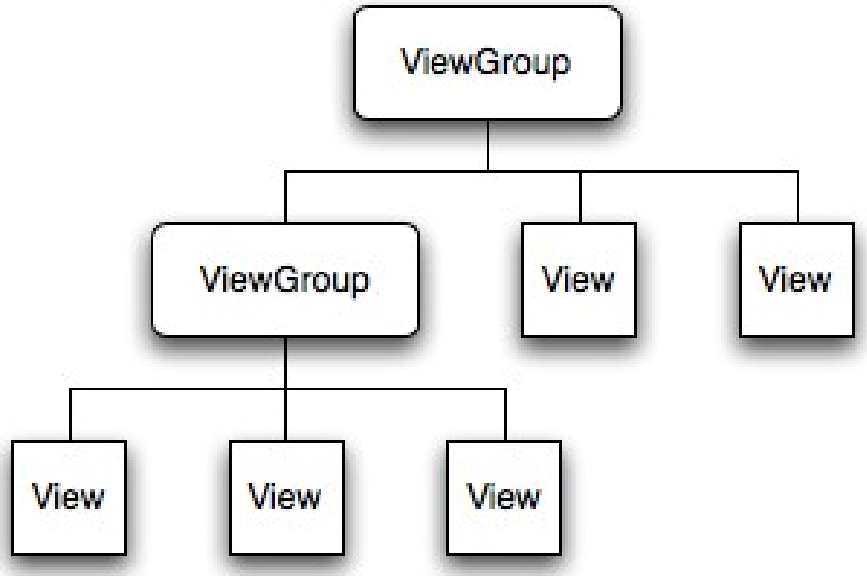
- La classe View è una struttura dati le cui proprietà memorizzano il layout e il contenuto di una specifica area rettangolare sullo schermo.
- L'oggetto View gestisce le dimensioni, il layout, il disegno, il focus, lo scorrimento, i tasti e i tocchi per lo schermo.
- La classe View serve come classe di base per diversi widget già implementati (Text, EditText, InputMethod, MovementMethod, Button, RadioButton, Checkbox e ScrollView).

android.view.ViewGroup

- Come il nome stesso suggerisce, un ViewGroup è un oggetto il cui compito è quello di contenere un insieme di altri View e ViewGroup.
- In tal modo schermate complesse possono essere gestite come se fossero una unica entità.
- La classe ViewGroup serve come classe di base per diversi tipi comuni di layout già implementati.

GUI strutturata ad albero

- L'interfaccia grafica di una Activity è costituita da un albero di View e ViewGroup.
- Per collegare l'albero allo schermo, l'Activity invoca la propria setContentView() passando come argomento il nodo radice.



Disegno dell'albero

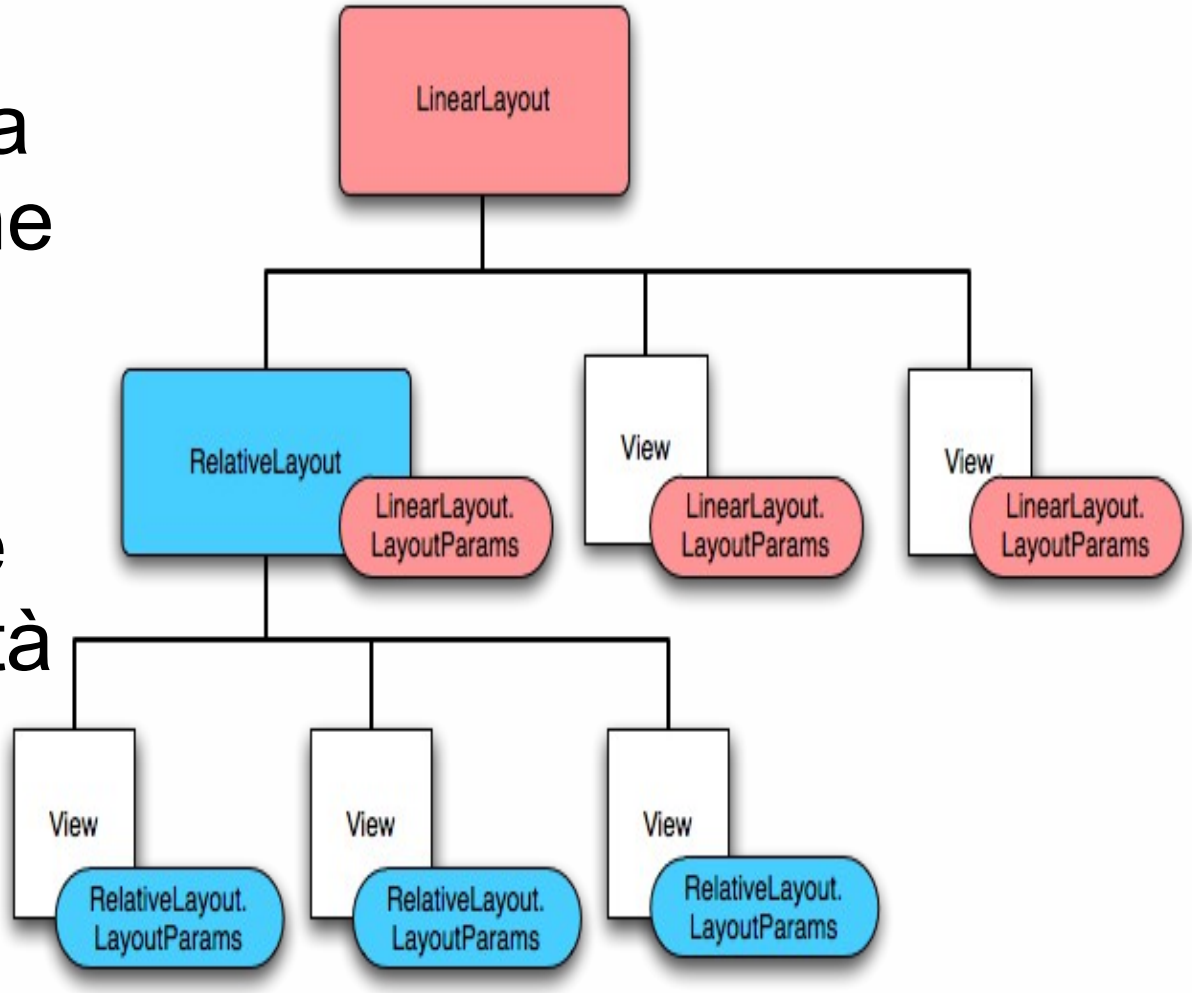
- Una volta che Android ha il riferimento all'oggetto del nodo radice può lavorare direttamente con esso per invalidare, misurare e disegnare l'albero.
- Quando l'Activity diventa attiva e riceve il focus, il sistema lo notifica all'Activity che richiede quindi al nodo radice di misurare e disegnare l'albero.
- Il nodo radice richiede quindi ad ognuno dei nodi figli di disegnare se stessi, e così via.

Disegno dei figli

- Ogni ViewGroup ha la responsabilità di misurare lo spazio disponibile, disporre i suoi figli e invocare il metodo Draw() su ognuno dei figli per fare in modo che si disegnino.
- Il figlio può quindi richiedere una dimensione e una posizione nel genitore, ma la decisione finale su dove e quanto grande possa essere il figlio spetta all'oggetto genitore.

ViewGroup.LayoutParams

- Ogni classe ViewGroup usa una classe innestata che estende LayoutParams.
- Questa sottoclasse contiene le proprietà che definiscono dimensione e posizione dei figli.

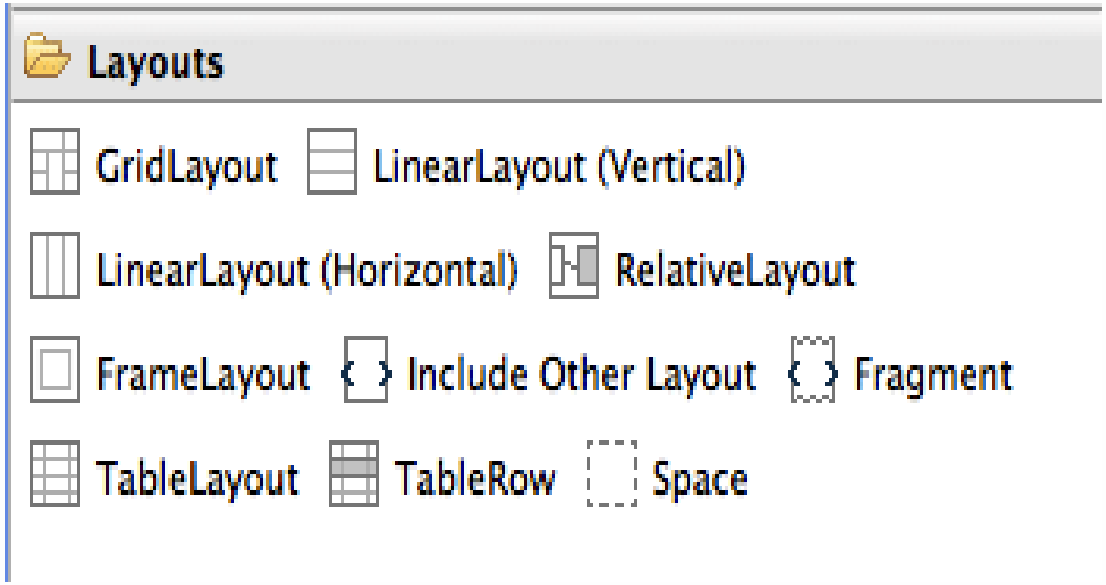


Dimensioni

- Tutti i ViewGroup includono larghezza e altezza; molti includono anche margini e bordi.
- A volte potrebbe essere necessario impostare larghezza e altezza a un valore fisso, mentre nella maggior parte dei casi si potrebbe voler richiedere a un ViewGroup di assumere le dimensioni degli oggetti in essi contenuti oppure la dimensione massima possibile.

Tipi di layout comuni

- FrameLayout
- LinearLayout
- TableLayout
- RelativeLayout
- GridLayout

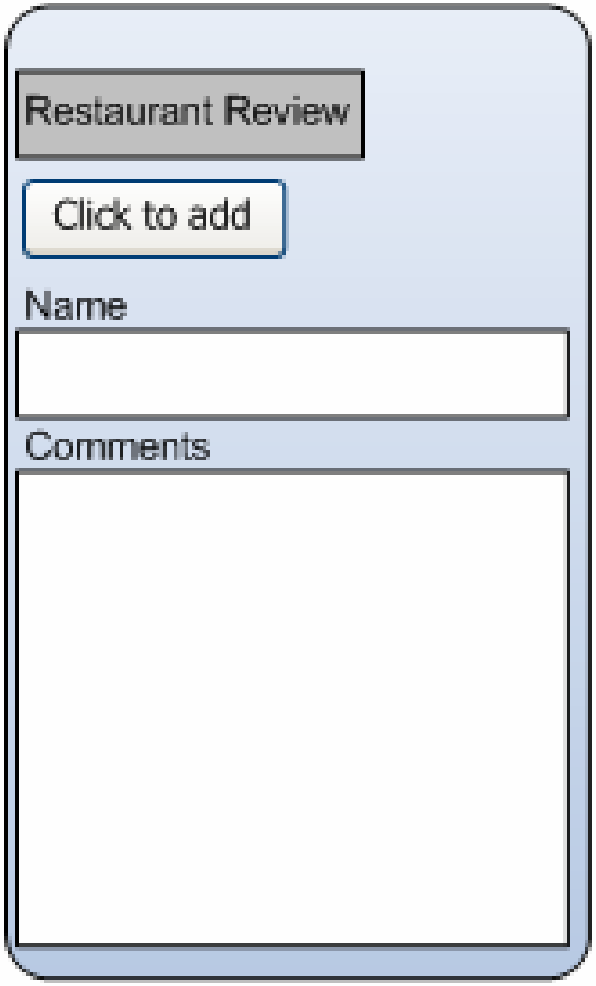


FrameLayout 

- FrameLayout è l'oggetto di layout più semplice.
- E' uno spazio vuoto riservato sullo schermo che può poi essere riempito con un singolo oggetto.
- Tutti gli oggetti figli vengono allineati all'angolo in alto a sinistra.
- I figli aggiunti successivamente andranno quindi a coprire (totalmente o parzialmente) i figli aggiunti in precedenza.

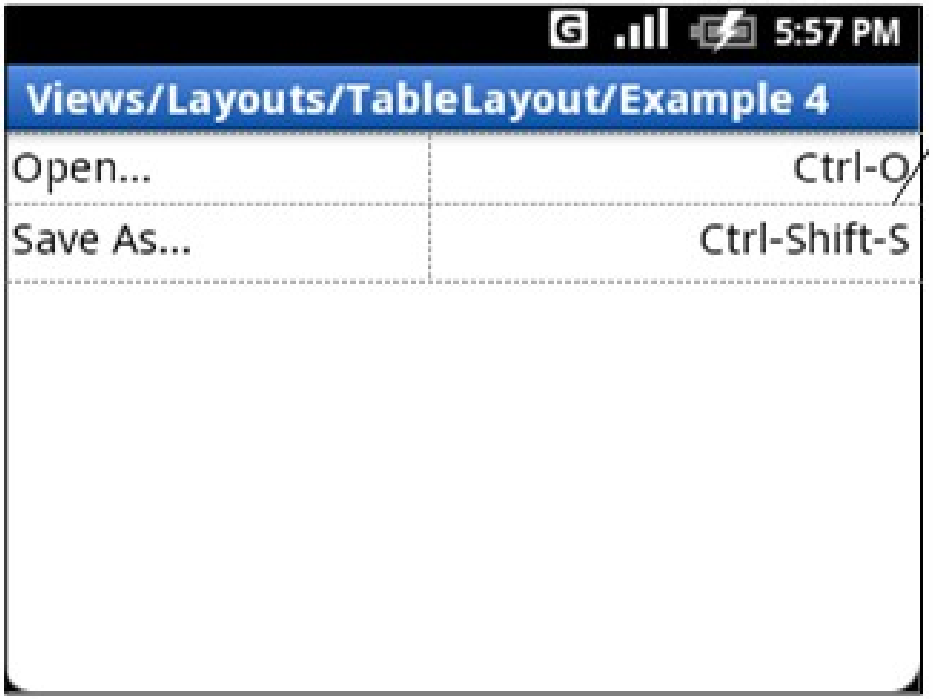
LinearLayout

- Un LinearLayout allinea tutti i suoi figli in una direzione, in orizzontale o verticale, a seconda di quale proprietà viene settata.
- Si possono definire gravità e peso in modo che un oggetto (es. Comments) si espanda.



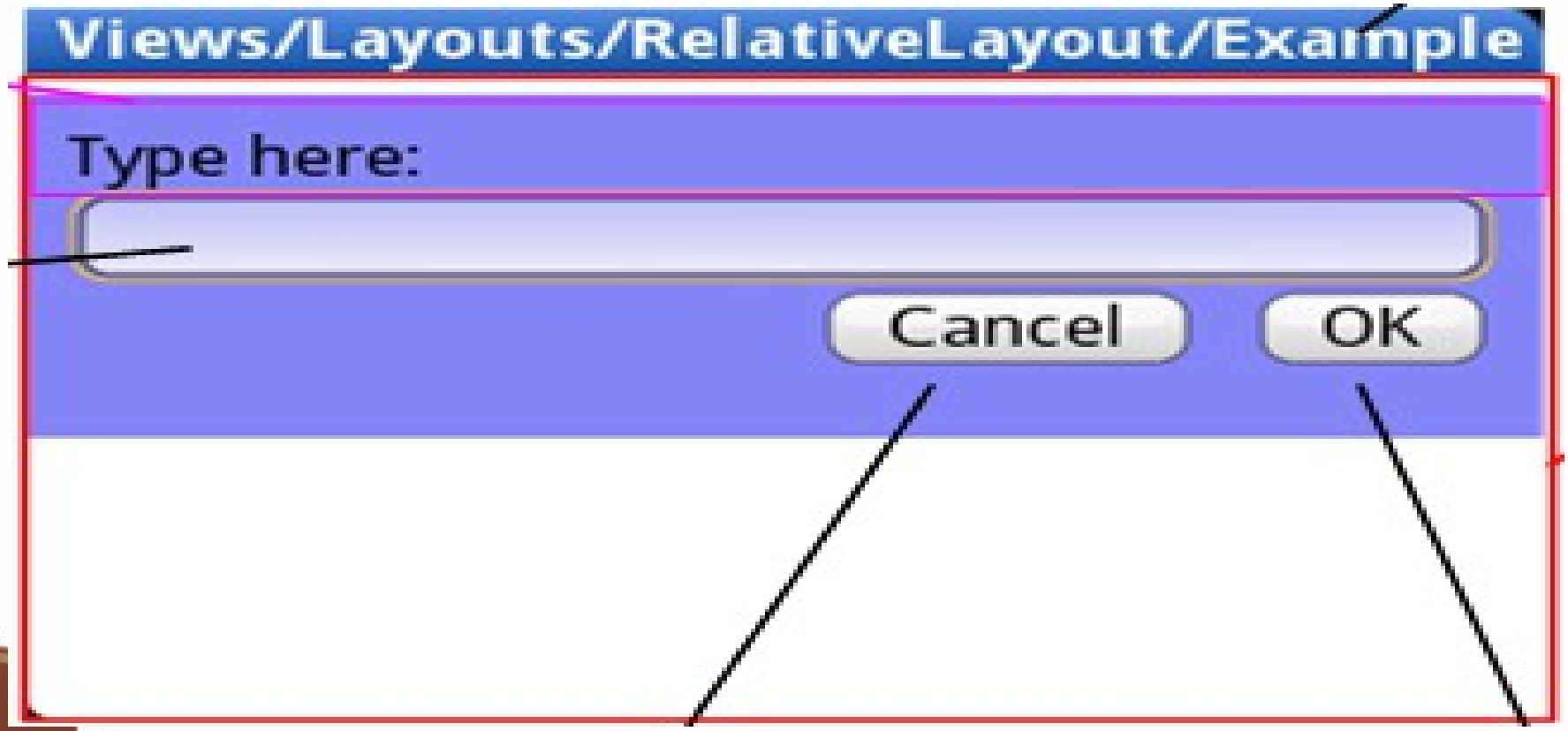
TableLayout

- Il TableLayout dispone i figli in righe e colonne.
- E' formato da oggetti di tipo TableRow, formati da celle che a loro volta contengono una sola View.



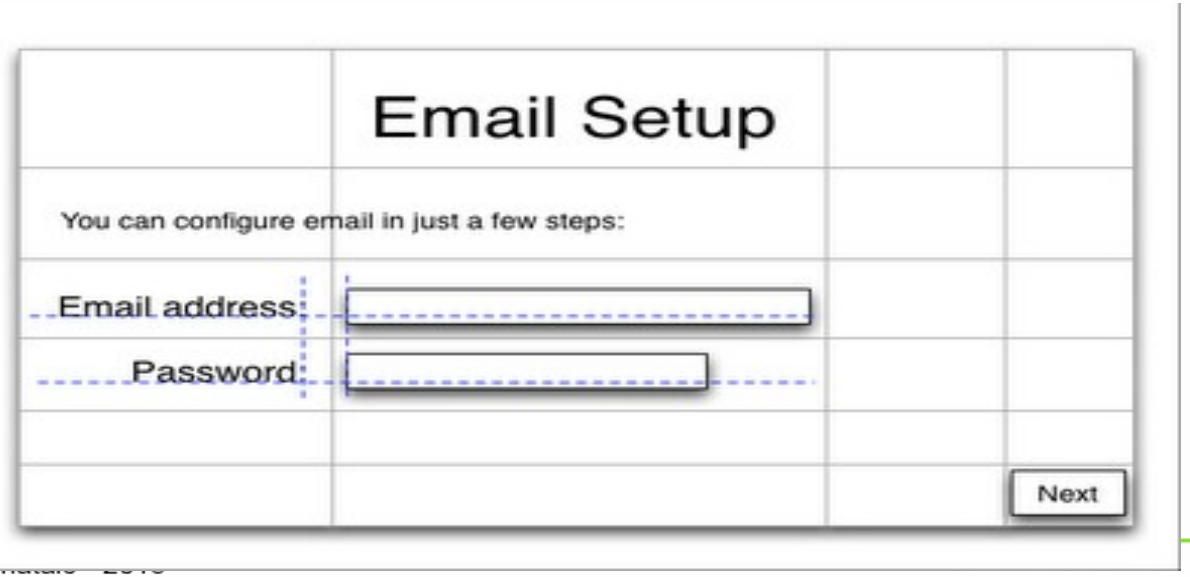
RelativeLayout

- Il RelativeLayout permette ai figli di specificare la loro posizione relativa a quella di un altro, identificato tramite un ID.



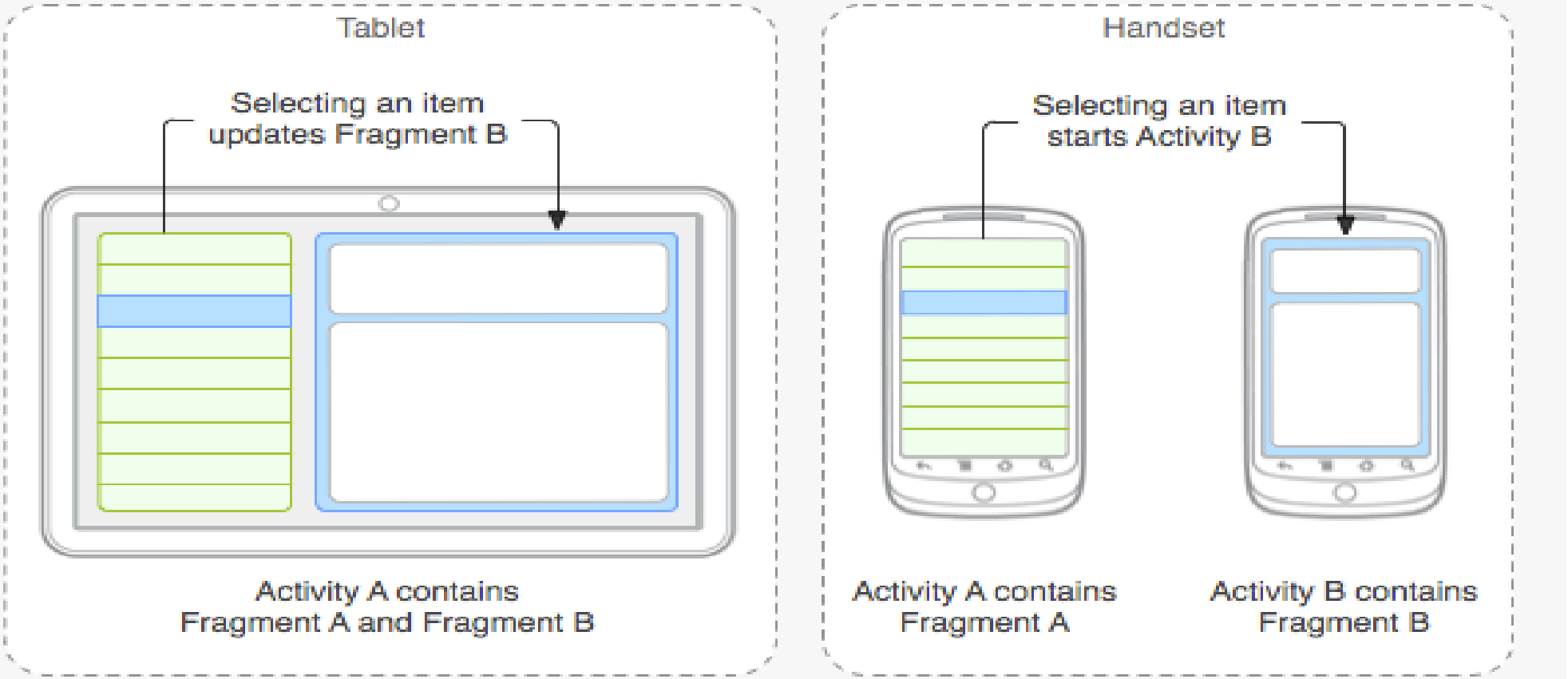
GridLayout

- Il GridLayout (disponibile a partire dalla 4.0) consente di definire una griglia fatta di righe, colonne e celle all'interno delle quali inserire i widget.
- La differenza con il TableLayout è che gli oggetti all'interno delle celle si adattano alle variazioni nell'interfaccia (es. Dimensione dei font) e utilizza meno memoria.



Fragment

- Sono stati introdotti a partire dalla 3.0
- La classe base per i Fragments è android.app.Fragment
- Facilitano il riuso di componenti in differenti layout: per es. , si può definire un layout di tipo single-pane per smartphone e multi-pane tablet.



from: developer.android.com

Fragment

```
DetailFragment fragment = (DetailFragment)
getFragmentManager().
    findFragmentById(R.id.detail_frag);
if (fragment==null || ! fragment.isInLayout()) {
  // start new Activity
  }
else {
  fragment.update(...);
}
```

Esistono fondamentalmente due approcci:

- Si utilizza una sola Activity che ad es. mostra due Fragment per i Tablet ed un fragment per gli Smartphone
- Facilitano il riuso di componenti in differenti layout: per es. , si può definire un layout di tipo single-pane per smartphone e multi-pane tablet.

Creazione della UI

- Mediante l'ADT è possibile creare l'intera UI della nostra applicazione, con i layout, i widget e le altre risorse, senza scrivere alcuna riga di codice
- Per le icone fare riferimento a:

  http://developer.android.com/design/style/iconography.html
- Esistono anche altri tool di supporto tra i quali:
  - **DroidDraw** per sviluppare la UI
  - **Android UI Utils** composte da:
    - **UI Prototyping Stencils**, per la prototipazione
    - **Android Asset Studio**, per la generazione di Icone
    - **Android Icon Templates**, raccolta di Icon Template in formato PhotoShop

Conclusioni

In questa lezione abbiamo visto le caratteristiche di base della gestione dell'interfaccia utente da parte di Android.

In particolare abbiamo visto l'organizzazione gerarchica degli oggetti e i 5 principali tipi di layout usati per disporre gli oggetti sullo schermo.