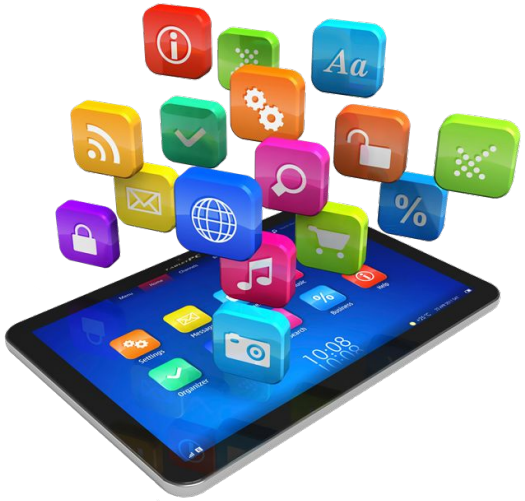
UNIVERSITÀ
DEGLI STUDI
DE L'AQUILA

disim



Mobile Programming Laboratory

ANDROID
Layouts

Teachers

Ing. Tarquini Francesco, Ph.D

Ph.D in Computer Science Engineering

francesco.tarquini@univaq.it

Ing. D'Errico Leonardo

Ph.D Student in Computer Science Engineering

leonardo.derrico@graduate.univaq.it

Teaching Materials

Available on MOODLE platform

http://www.didattica.univaq.it

Google Drive Repository

https://drive.google.com/drive/folders/1ISqZfn0i9Ub3eWNXbvW00rd0hD9ya8OL?usp=sharing

Topics

- Identifiers
- Layouts
  - RelativeLayout
  - LinearLayout
  - FrameLayout
  - TableLayout
  - ConstraintLayout

Identifiers

The developer can declare for every items add in layout XML file, layout or containers or generic view, an identifier.

This id is used:

- to define the layout attributes constraints between item included in the same XML file
- to link the XML item to right Java instance in Activity class

The developer, to declare the identifier in XML file, have to use the attribute **id**

@+id/your_identifier_name

If in XML file you want use an identified item the developer have to use as value of the constraint

@id/item_identifier_name (without the plus)

Layouts

Since the first version of Android the RelativeLayout and LinearLayout have been introduced.

These items were the most used to define a layout, but in the last two years Google recommended to use ConstraintLayout instead RelativeLayout to have better performances.

Instead the TableLayout permits to create easily a structure as a table using the TableRow as a row of the table.

Another common layout is the FrameLayout, used especially to implement the Fragment.

RelativeLayout

The RelativeLayout permits to define a position of its children declaring its:

- margins
- constraints with the parent
- constraints between the other children

The layout allows its children to use the follow attributes:

With the parent:

layout_centerInParent
layout_centerHorizontal, layout_centerVertical
layout_alignParentTop, layout_alignParentBottom
layout_alignParentStart (layout_alignParentLeft), layout_alignParentEnd (layout_alignParentRight)

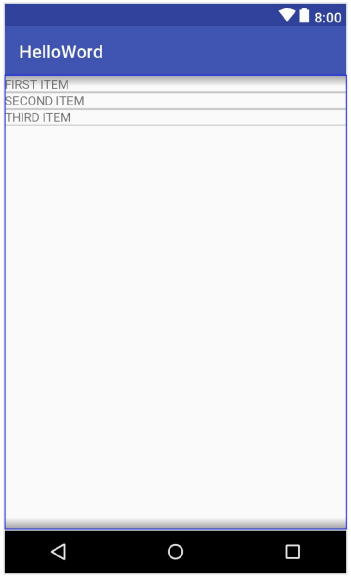
Between the other children

layout_below, layout_above
layout_toStartOf (layout_toLeftOf), layout_toEndOf (layout_toRightOf)

LinearLayout

The LinearLayout is available since Android 1.0 and aligns all children in a single direction, vertically or horizontally.

For default the direction is horizontally and the developer can change it declaring the "**orientation**" attribute (**horizontal** or **vertical**)




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

  <TextView
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:text="FIRST ITEM"/>

  <TextView
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:text="SECOND ITEM"/>

  <TextView
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:text="THIRD ITEM"/>

</LinearLayout>
```

FrameLayout

The FrameLayout is designed to block out an area on the screen to display a single item.

FrameLayout should be used to hold a single child view, because it can be difficult to organize child views in a way that's scalable to different screen sizes without the children overlapping each other.

Child views are drawn in a stack, with the most recently added child on top.

Typically, the FrameLayout is used to manage the Fragment.

TableLayout

A layout that arranges its children into rows and columns.

A TableLayout consists of a number of **TableRow** objects, each defining a row.

The width of a column is defined by the row with the widest cell in that column. However, a TableLayout can specify certain columns as shrinkable or stretchable by calling "**columnShrinkable**" or "**columnStretchable**".

The children of a TableLayout cannot specify the **layout_width** attribute. Width is always MATCH_PARENT.

The **layout_height** attribute can be defined by a child; default value is **WRAP_CONTENT**.

If the child is a TableRow, then the height is always **WRAP_CONTENT**.

ConstraintLayout

ConstraintLayout is available as a support library that you can use on Android systems starting with API level 9.

A ConstraintLayout is a ViewGroup which allows you to position and size widgets in a flexible way.

There are currently various types of constraints that you can use:

- Relative positioning
- Margins
- Centering positioning
- Circular positioning
- Visibility behavior
- Dimension constraints
- Chains
- Virtual Helpers objects
- Optimizer