

#4 界面编程（上）



界面设计

系统和用户之间进行交互和信息交换的媒介，主要作用是实现信息内部形式与人类可接受形式之间的转换。

人机交互实践中，一个好的界面设计不仅让软件变得更加有个性和创意，同时还能让软件的操作变得舒适自由，充分体现软件的定位和特点。

界面设计

UI(User Interface)

应用软件的操作逻辑、人机交互、界面的整体设计

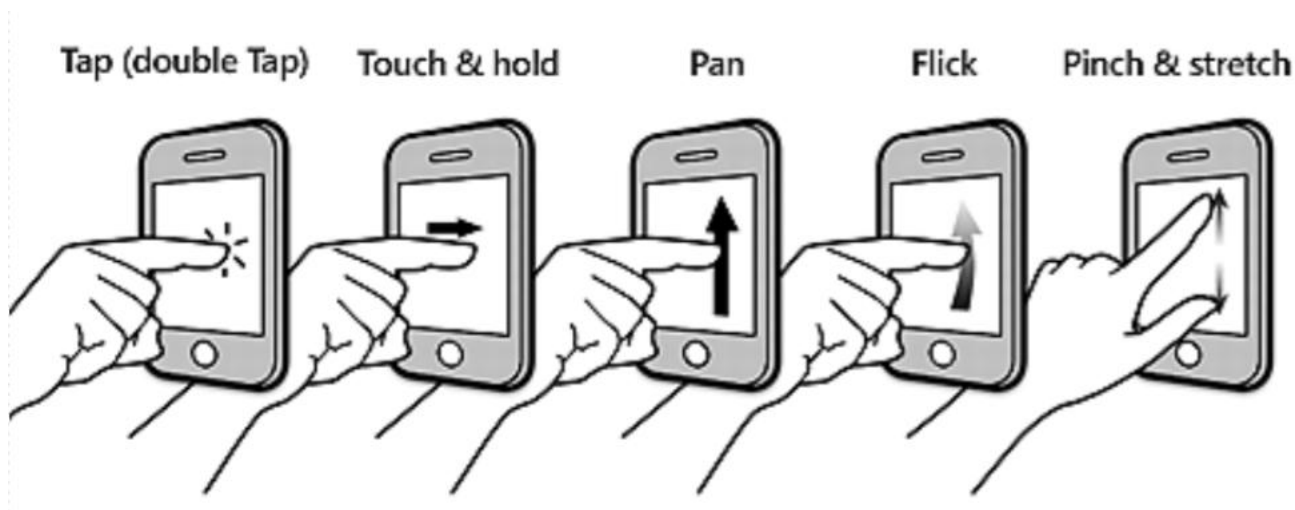
ID(Interaction Design)

人、环境与设备的关系和行为,以及传达这种行为的元素的设计

GUI(Graphical User Interface)

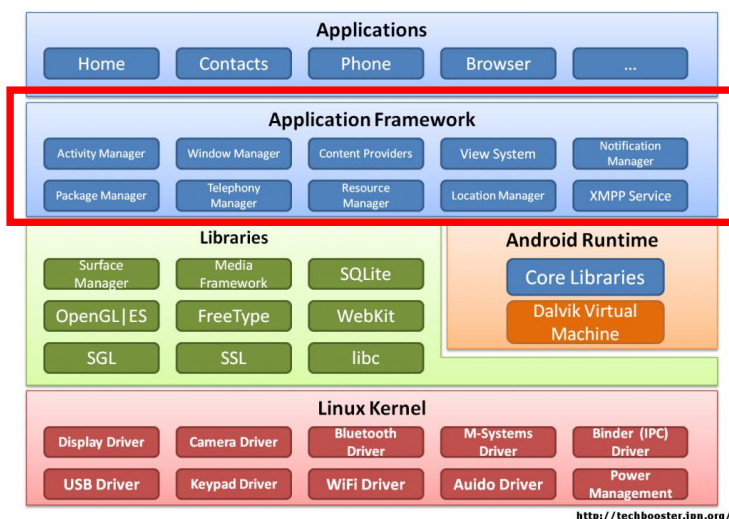
图形用户界面

手势操作



- **Tap:** 用户用单个手指轻击触摸屏,随后马上与触摸屏开
- **Double tap:** 短 时间,两次tap操作
- **Touch & hold:** 用户用手指轻击触摸屏,保持不动
- **Pan:** 用户手指轻击触摸屏,在不离开的情况下,移动手指
- **Flick:** 用户手指轻击触摸屏,然后迅速的滑动,同时离开屏幕(想象一下, 你用手指在弹一个东西)
- **Pinch & stretch:** 用户用两个手指分别往两个方向移动。

Framework



Android app framework

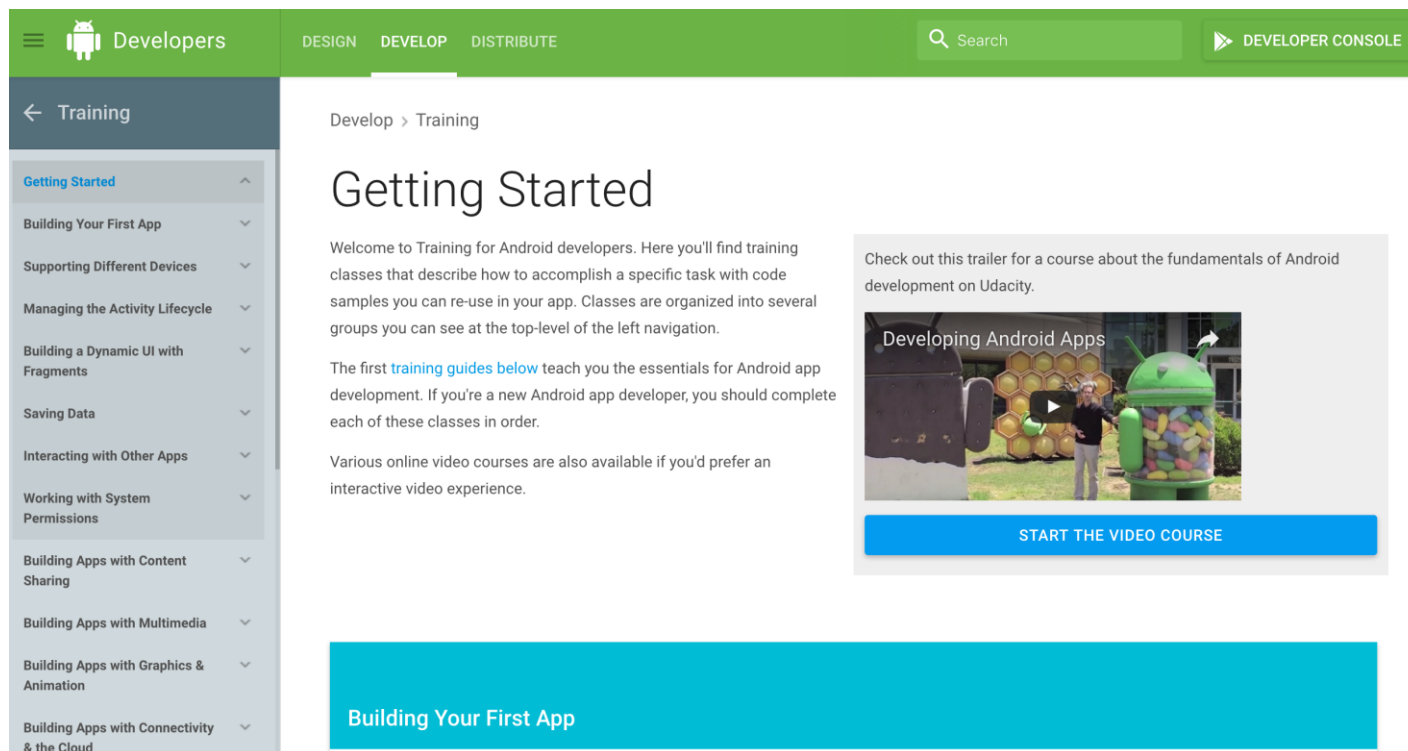


某种应用的半成品,就是一组组件,供你选用完成你自己的系统。
简单说就是使用别人搭好的舞台,你来做表演。而且框架一般都是成熟的,不断升级的软件。

Framework

帮助文档的学习

<http://developer.android.com/training/index.html>



The screenshot shows the Android Developer Training page. The top navigation bar is green with the Android logo, 'Developers', and tabs for 'DESIGN', 'DEVELOP', and 'DISTRIBUTE'. A search bar and 'DEVELOPER CONSOLE' link are on the right. The left sidebar is dark blue with a 'Training' header and a list of topics: 'Getting Started' (highlighted), 'Building Your First App', 'Supporting Different Devices', 'Managing the Activity Lifecycle', 'Building a Dynamic UI with Fragments', 'Saving Data', 'Interacting with Other Apps', 'Working with System Permissions', 'Building Apps with Content Sharing', 'Building Apps with Multimedia', 'Building Apps with Graphics & Animation', and 'Building Apps with Connectivity & the Cloud'. The main content area has a green header 'Develop > Training' and a large 'Getting Started' section. It includes a welcome message, a paragraph about training classes, a paragraph about training guides, and a paragraph about video courses. A video player titled 'Developing Android Apps' is shown with a 'START THE VIDEO COURSE' button. A blue banner at the bottom says 'Building Your First App'.

Developers

DESIGN DEVELOP DISTRIBUTE

Search

DEVELOPER CONSOLE

← Training

Getting Started

Building Your First App

Supporting Different Devices

Managing the Activity Lifecycle

Building a Dynamic UI with Fragments

Saving Data

Interacting with Other Apps

Working with System Permissions

Building Apps with Content Sharing

Building Apps with Multimedia

Building Apps with Graphics & Animation

Building Apps with Connectivity & the Cloud

Develop > Training

Getting Started

Welcome to Training for Android developers. Here you'll find training classes that describe how to accomplish a specific task with code samples you can re-use in your app. Classes are organized into several groups you can see at the top-level of the left navigation.

The first [training guides below](#) teach you the essentials for Android app development. If you're a new Android app developer, you should complete each of these classes in order.

Various online video courses are also available if you'd prefer an interactive video experience.

Check out this trailer for a course about the fundamentals of Android development on Udacity.

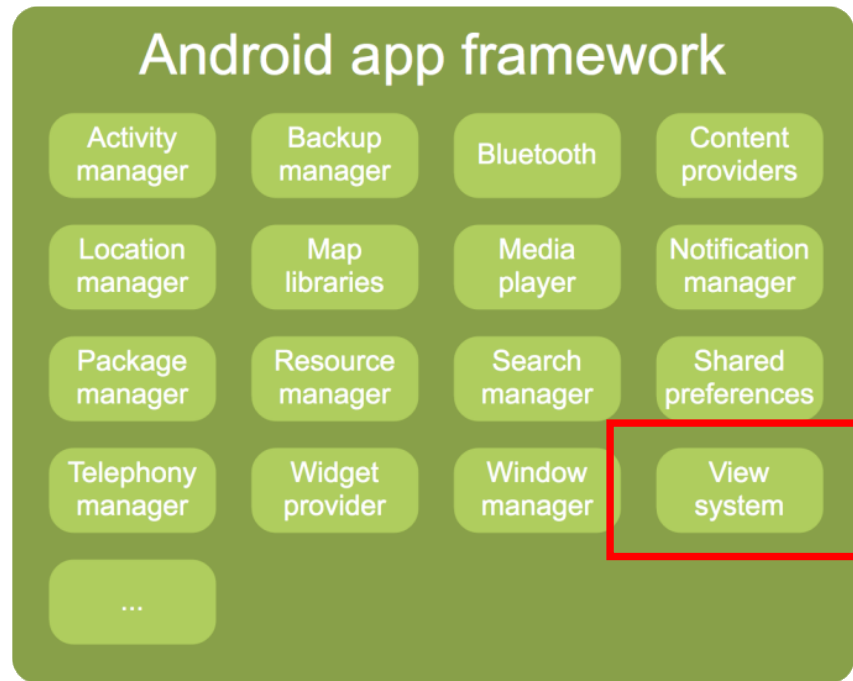
Developing Android Apps

START THE VIDEO COURSE

Building Your First App

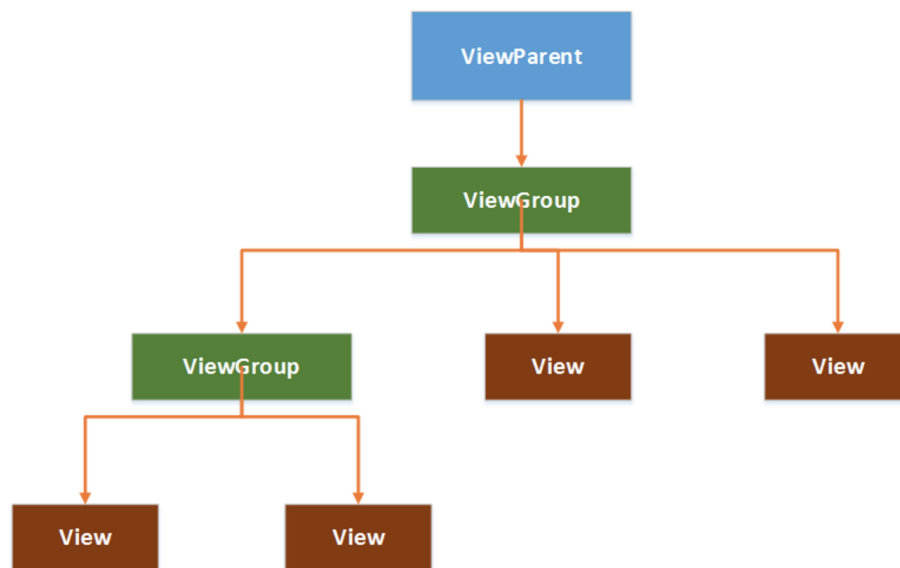
View

视图View作为界面基本元素，由View System管理。



View树结构

在Android中，视图控件大致被分为两类，即ViewGroup和View，ViewGroup控件作为父控件，包含并管理着子View，通过ViewGroup和View便形成了控件树，各个ViewGoup对象和View对象就是控件树中的节点。

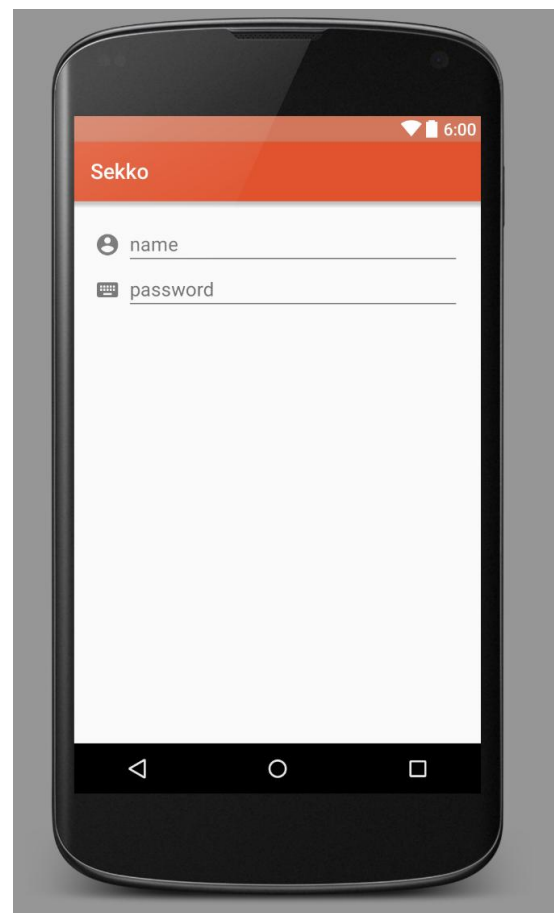


View组件

- View类是最基本的一个UI类,基本上所有的高级UI组件都是继承View类而实现。
 - 一个视图在屏幕占据一块矩形区域,负责渲染这块矩形区域,也可以处理此区域发生的事件,可以设置是否可见,是否可以获取焦点。
-
- ◆ TextView(文本框)
 - ◆ RadioButton(单选框)
 - ◆ Checkbox(检查框)
 - ◆ Button(按钮)
 - ◆ List(列表)
 - ◆ EditText(编辑框)

界面编程 支持可视化编辑与脚本编辑

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginLeft="20dp"
        android:layout_marginBottom="20dp"
        android:layout_marginTop="20dp"
        android:orientation="vertical"
        android:layout_marginRight="20dp">
        <TableLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent">
            <TableRow>
                <ImageView
                    android:layout_height="wrap_content"
                    android:layout_width="wrap_content"
                    android:src="@drawable/ic_account_circle_black_24dp"
                    android:layout_gravity="center_vertical"
                    android:layout_marginRight="5dp"/>
                <EditText
                    android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:layout_weight="1.0"
                    android:id="@+id/login_name"
                    android:hint="@string/login_name"/>
            </TableRow>
            <TableRow>
                android:layout_width="match_parent"
                android:layout_height="wrap_content">
                <ImageView
                    android:layout_height="wrap_content"
                    android:layout_width="wrap_content"
                    android:src="@drawable/ic_keyboard_black_24dp"
                    android:layout_gravity="center_vertical"
                    android:layout_marginRight="5dp"/>
                <EditText
                    android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:layout_weight="1.0"
                    android:id="@+id/login_password"
                    android:hint="@string/login_password"/>
            </TableRow>
        </TableLayout>
    </LinearLayout>
</LinearLayout>
```



xml文件

效果预览

界面度量单位

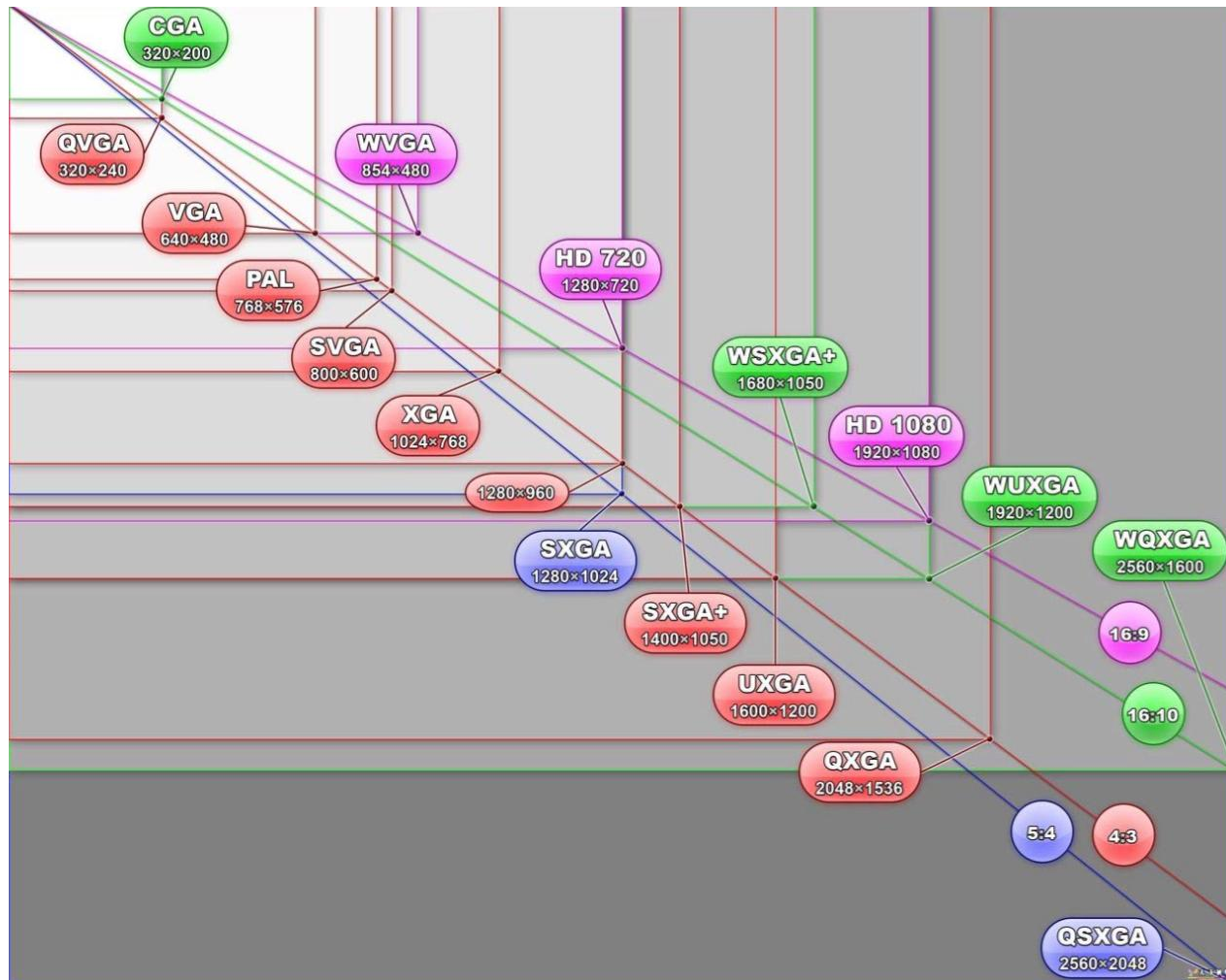
通常以像素为单位进行用户界面设计。例如:图片大小为 80×32 像素。然而, 若在一个每英寸点数(dpi)更高的新显示器上运行该程序, 则用户界面会显得很很小。在有些情况下, 用户界面可能会小到难以看清内容。由此建议采用与分辨率无关的度量单位来开发程序解决该问题。Android 应用开发支持不同的度量单位。

界面度量单位

- dp: 与密度无关像素,一种基于屏幕密度的抽象单位
 $px = dp * density / 160$,则当屏幕密度为160时, $px = dp$
- dip: 与dp相同
- sp: 与dp类似,但是可以根据用户的字体大小首选项进行缩放;
主要用于字体显示;
- px: pixels(像素).屏幕上的点;
- pt: (磅)标准的长度单位, $1pt = 1/72$ 英寸,用于印刷业,非常简单易用;

sp所为文字大小的单位,**dip**作为其他元素的单位。

各种分辨率格式



Android UI控件

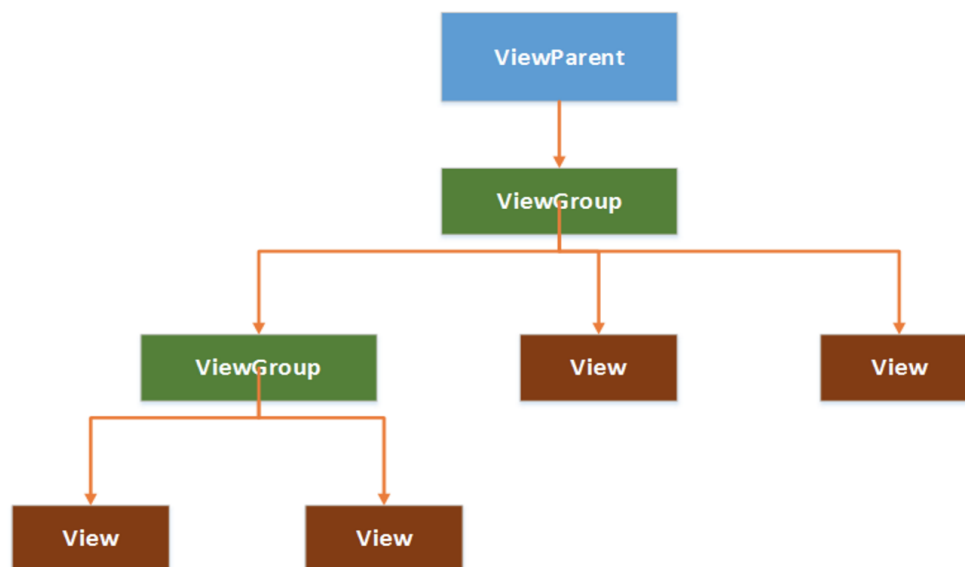
Android提供的UI控件分别包括了几种布局Layout和多种组件(widget),例如Button(按钮)、TextView(文本)、EditText(文本编辑框)、ListView(列表)、CheckBox(复选框)、RadioButton(单选按钮)、Spinner(下拉列表)以及AutoCompleteTextView(带自动补全的文本框)、图片切换器(ImageSwitcher)等等。

另外还有一些较复杂且常用的控件例如时间日期选择控件和缩放控制控件。当然,开发人员更可以自己创建一些控件供应用程序使用,只要按照一定的标准去自定义视图对象或者直接在已有控件上进行扩展和合并即可。

Android UI布局

Android工作界面主要由容器和控件构成,为了规范控件在容器中的显示,设计人员通常需要规定控件在界面的显示方式,这就是**布局文件**。

ViewGroup通过各种**Layout**, 控制所属View的显示层次与位置。

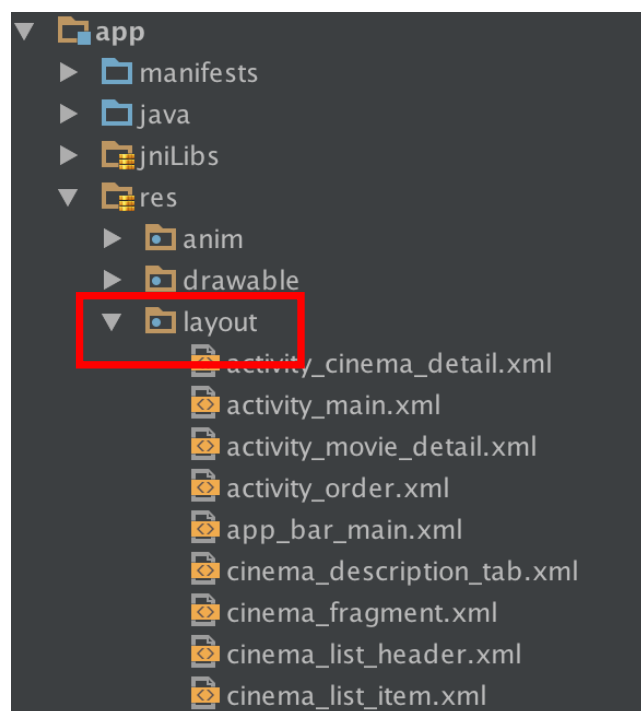


Android UI布局

布局的作用

在布局中,通过设置控件或者容器的属性来规定控件的显示方式

布局文件



布局的实现之一(布局文件)

在Android应用程序中,界面通常都是通过布局文件设定。该文件采用XML 文件格式。每个应用程序默认包含一个主界面的布局文件。该文件位于项目 文件中res目录下的layout子目录中。单击选项卡选择布局文件或是界面设计面板。

布局方式

Android为开发人员提供了两种声明的方式

- **XML**文件中声明**UI**元素

使用xml语法来声明UI视图, 这些xml文件存放在项目树下的/res/layout目录下, Android为每种View都提供了很多属性,通过设置这些属性来达到定制用户界面的目的。

- **Java**代码中实时声明**UI**元素

对于第一种方法中提到的用于定制View的xml属性, 基本上每一种可以在xml中设置的属性都对应了一个Java方法, 可以在Java代码中使用这些方法来声明。

布局方式

Xml声明

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/layout">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/app_name">
    </TextView>

</LinearLayout>
```

布局方式

JAVA声明

```
LinearLayout layout = (LinearLayout)findViewById(R.id.layout);
TextView tv= new TextView(this);
tv.setLayoutParams(new LinearLayout.LayoutParams
    (LinearLayout.LayoutParams.MATCH_PARENT,
    LinearLayout.LayoutParams.WRAP_CONTENT));
tv.setText(R.string.app_name);
layout.addView(tv);
```

布局方式

XML 布局文件

优点

- 直观简洁,可读性强;
- 实现了UI界面和逻辑代码的分离

缺点

- 难以动态布局

JAVA 布局

优点

- 动态布局

缺点

- 抽象模糊,可读性比较差;
- 耦合性强,数据的表现和逻辑错杂

布局layout

Android系统提供的布局

1. 线性布局(LinearLayout)
2. 表格布局(TableLayout)
3. 相对布局(RelativeLayout)
4. 绝对布局(AbsoluteLayout)
5. 框架布局(FrameLayout)
6. 网格布局(GridLayout)

线性布局LinearLayout

线性布局是按照水平或垂直的顺序将子元素(可以是控件或布局)依次按照顺序排列，每一个元素都位于前面一个元素之后。线性布局分为两种：水平方向和垂直方向的布局。分别通过属性 `android:orientation= "vertical/horizontal"` 来设置。

特点：此布局里面可以放多个控件,但是一行/列只能放一个控件。

- 布局文件:<LinearLayout> 标签进行配置;
- 代码实现:Android.widget.LinearLayout类对象的实例

线性布局LinearLayout

属性说明

gravity: 控制布局中视图的位置。

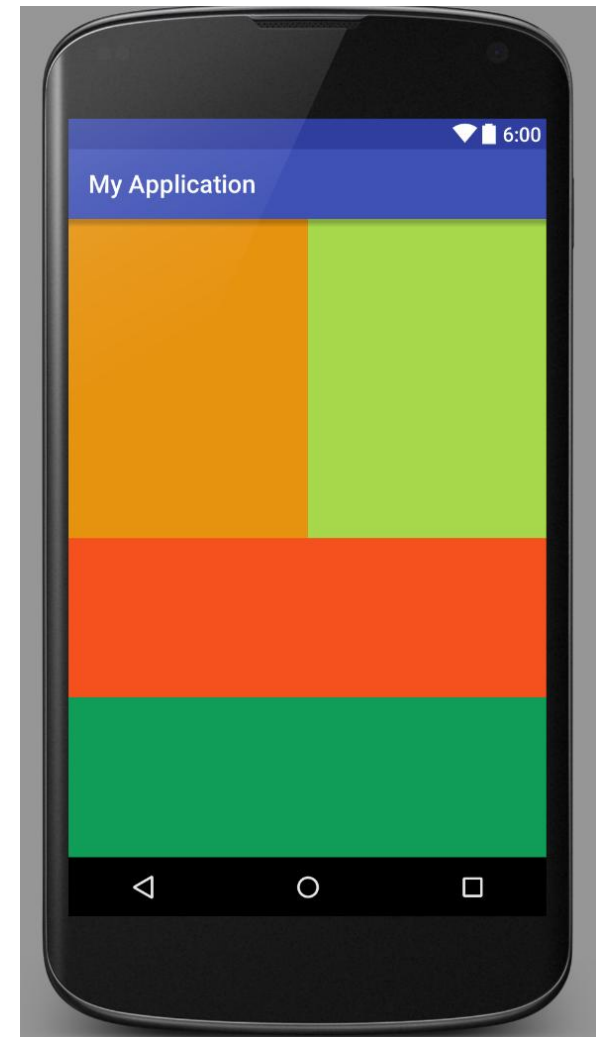
orientation: "vertical "/" horizontal"

width / height: "fill_parent" (填充整个幕) /wrap_content" (根据内容调整)

layout_weight: 设置所占比例的权重。在线性布局中,每个视图都有一个android:layout_weight值,若没有显式的声明则为默认值0,表示按照视图的实际大小在屏幕上显示。当该属性被赋予一个大于零的值时,则将父容器中的可用空间进行分割,分割的大小则根据每个视图的android:layout_weight值来确定,权值越大所占比例越大。

线性布局LinearLayout layout_weight

```
<?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">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:orientation="horizontal"
        android:layout_weight="1">
        <TextView
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:background="@android:color/keyguard_text_color_soundon"
            android:layout_weight="1"/>
        <TextView
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:background="@android:color/keyguard_text_color_unlock"
            android:layout_weight="1"/>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:orientation="vertical"
        android:layout_weight="1">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:background="@android:color/perms_costs_money"
            android:layout_weight="1"/>
        <TextView
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:background="@android:color/user_icon_5"
            android:layout_weight="1"/>
    </LinearLayout>
</LinearLayout>
```



线性布局LinearLayout layout_gravity

android:layout_gravity是用来设置该view相对于父view的位置

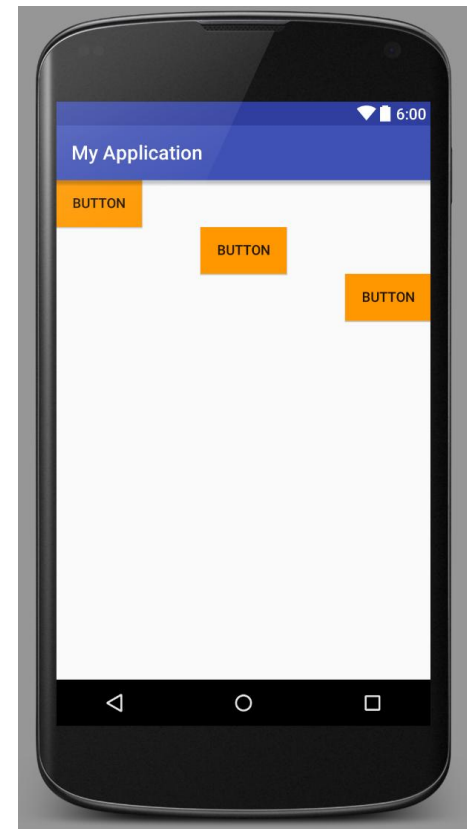
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/btn"
        android:background="@android:color/user_icon_7"
        android:layout_gravity="start"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/btn"
        android:background="@android:color/user_icon_7"
        android:layout_gravity="center_horizontal"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/btn"
        android:background="@android:color/user_icon_7"
        android:layout_gravity="end"/>

</LinearLayout>
```



线性布局LinearLayout gravity

android:gravity用于设置View中内容相对于View组件的对齐方式

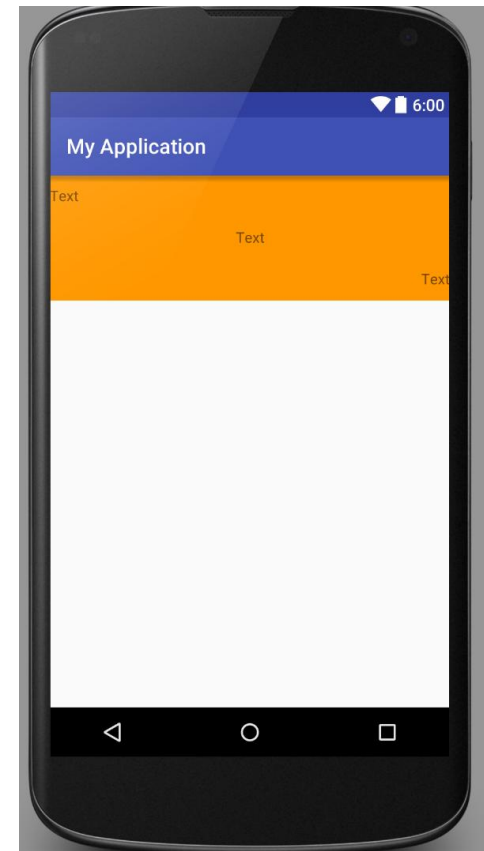
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="40dp"
        android:text="@string/text"
        android:background="@android:color/user_icon_7"
        android:gravity="start|center_vertical"/>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="40dp"
        android:text="@string/text"
        android:background="@android:color/user_icon_7"
        android:gravity="center_horizontal|center_vertical"/>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="40dp"
        android:text="@string/text"
        android:background="@android:color/user_icon_7"
        android:gravity="end|center_vertical"/>

</LinearLayout>
```



相对布局RelativeLayout

相对布局是一个容器,允许其子元素指定它们相对于**其它元素**或**父元素**的位置(通过元素的id来指定是相对于哪个元素的位置)。

可以通过向右对齐、向上或者向下对齐、至于屏幕中央等形式来排列界面中的元素。元素的相对关系跟顺序有关,如果第一个元素在屏幕 的中央,那么相对于这个元素的其它元素将以屏幕中央的相对位置来排列。如果要在xml中指定某个元素的相对位置,那么在定义这个元素之 前,必须先定义它要相对应的元素。

相对布局RelativeLayout

- 让子元素指定它们相对于其他元素的位置(通过ID来指定)或相对于父布局对象,跟AbsoluteLayout绝对坐标布局是个相反。
- 在RelativeLayout布局里的控件包含丰富的排列属性:

Layout_above: 控件在指定控件的上方

Layout_below: 控件在指定控件的下方

Layout_toStartOf

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="40dp"
    android:text="@string/text"
    android:id="@+id/text1"
    android:layout_alignBottom="@+id/text2"
    android:layout_alignEnd="@+id/text2"
    android:layout_alignLeft="@+id/text2"
    android:layout_alignParentBottom="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentStart="true"
    android:layout_alignParentTop="true"
    android:layout_alignRight="@+id/text2"
    android:layout_alignStart="@+id/text2"
    android:layout_alignTop="@+id/text2"
    android:layout_alignWithParentIfMissing="true"
    android:layout_below="@+id/text2"
    android:layout_centerInParent="true"
    android:layout_centerVertical="true"
    android:layout_margin="10dp"
    android:layout_marginBottom="10dp"
    android:layout_marginEnd="10dp"
    android:layout_marginLeft="10dp"
    android:layout_marginRight="10dp"
    android:layout_marginStart="10dp"
    android:layout_marginTop="10dp"
    android:layout_toEndOf="@+id/text2"
    android:layout_toLeftOf="@+id/text2"
    android:layout_toRightOf="@+id/text2"
    android:layout_toStartOf="@+id/text2"
/>
```

相对布局RelativeLayout

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

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/btn"
        android:textSize="20sp" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/button1"
        android:layout_toRightOf="@id/button1"
        android:text="@string/btn"
        android:textSize="20sp" />
```

```
<Button
    android:id="@+id/button3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/button2"
    android:layout_toLeftOf="@id/button2"
    android:text="@string/btn"
    android:textSize="20sp" />

<Button
    android:id="@+id/button4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@id/button2"
    android:layout_toRightOf="@id/button2"
    android:text="@string/btn"
    android:textSize="20sp" />

<Button
    android:id="@+id/button5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/button2"
    android:layout_toRightOf="@id/button2"
    android:text="@string/btn"
    android:textSize="20sp" />
</RelativeLayout>
```

My Application

BUTTON

BUTTON

BUTTON

BUTTON

BUTTON

相对布局RelativeLayout

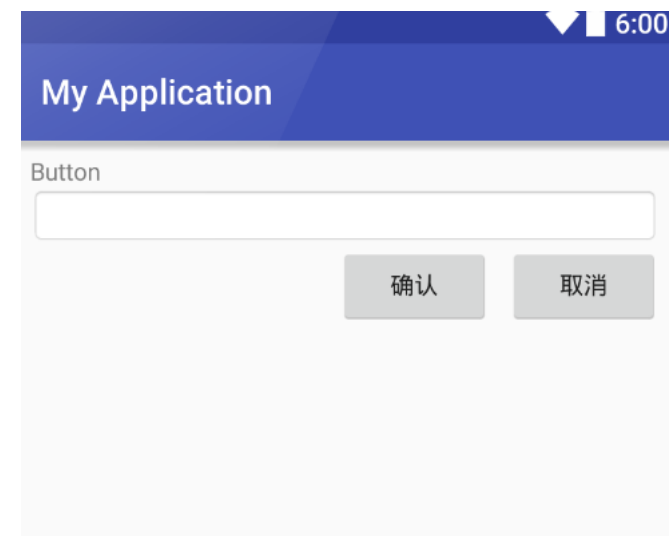
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:padding="10dip">

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

    <!--这个EditText放置在上边id为label的TextView的下边-->
    <EditText
        android:id="@+id/entry"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/label"
        android:background="@android:drawable/editbox_background" />

    <!--取消按钮和容器的右边齐平,并且设置左边的边距为10dip-->
    <Button
        android:id="@+id/cancel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_below="@id/entry"
        android:layout_marginLeft="10dip"
        android:text="@string/cancel" />

    <!--确定按钮在取消按钮的左侧,并且和取消按钮的高度齐平-->
    <Button
        android:id="@+id/ok"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@id/cancel"
        android:layout_toLeftOf="@id/cancel"
        android:text="@string/ok" />
</RelativeLayout>
```



表格布局TableLayout

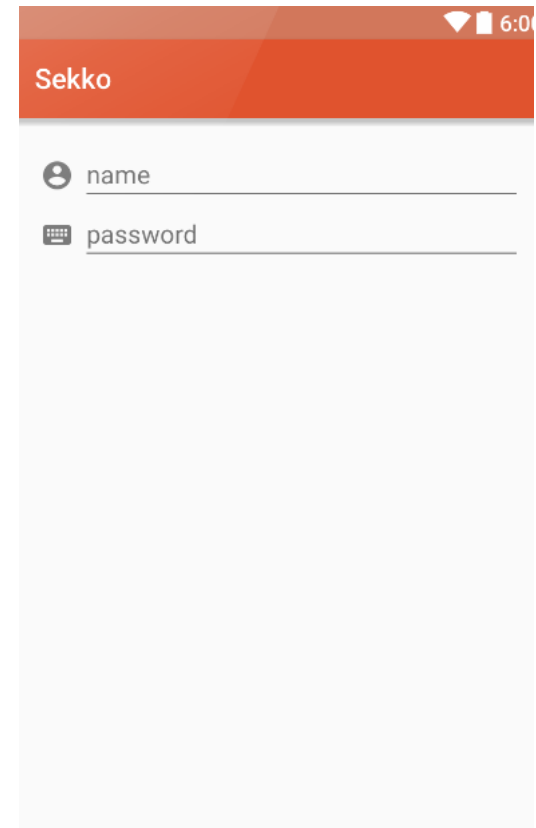
与TableRow配合使用, 类似HTML中的Table。

此布局里面可以放多个控件,但是一行(列)只能放一个控件

- 子元素放入到行与列中
- 不显示行、列或是单元格边界线
- 单元格不能横跨行,如HTML 中一样

表格布局TableLayout

```
<TableLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TableRow>
        <ImageView
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:src="@drawable/ic_account_circle_black_24dp"
            android:layout_gravity="center_vertical"
            android:layout_marginRight="5dp"/>
        <EditText
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1.0"
            android:id="@+id/login_name"
            android:hint="@string/login_name"/>
    </TableRow>
    <TableRow>
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <ImageView
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:src="@drawable/ic_keyboard_black_24dp"
            android:layout_gravity="center_vertical"
            android:layout_marginRight="5dp"/>
        <EditText
            android:layout_width="0dp"
            android:layout_weight="1.0"
            android:layout_height="wrap_content"
            android:id="@+id/login_password"
            android:hint="@string/login_password"/>
    </TableRow>
</TableLayout>
```



绝对布局AbsoluteLayout

指明子元素确切的屏幕(X,Y)坐标

(0,0)是左上角,下移或右移时,坐标值增加

允许元素重叠(但是不推荐)

- 一般建议不使用AbsoluteLayout 除非有很好的理由来使用
- 因为它相当严格并且在不同的设备显示中不能很好地工作

绝对布局AbsoluteLayout



* 已经不建议使用

绝对布局AbsoluteLayout

AbsoluteLayout

public class AbsoluteLayout

extends [ViewGroup](#)

[java.lang.Object](#)

↳ [android.view.View](#)

↳ [android.view.ViewGroup](#)

↳ [android.widget.AbsoluteLayout](#)

▼ Known Direct Subclasses

[WebView](#)

This class was deprecated in API level 3.

Use [FrameLayout](#), [RelativeLayout](#) or a custom layout instead.

* 已经不建议使用

帧布局FrameLayout

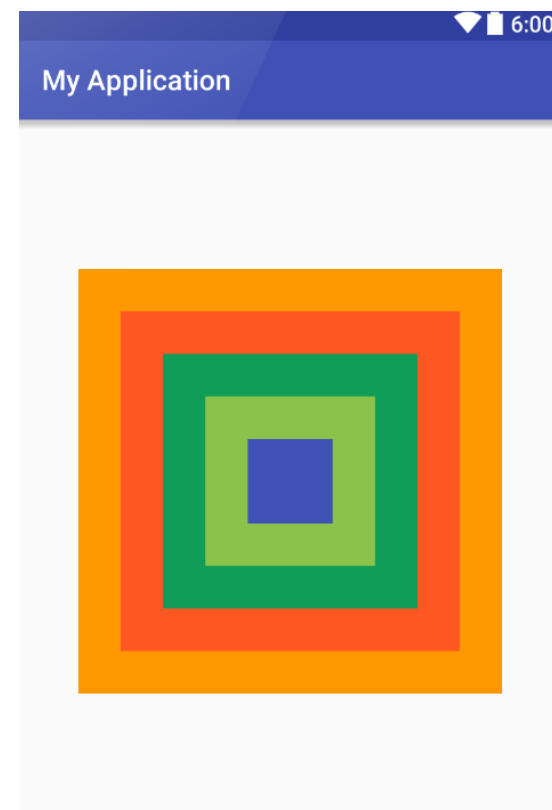
添加到此布局中的视图都以层叠方式展示,最后一个添加到框架布局中的视图显示在最上层,第一个添加的放在最底层(类似堆栈)。

- 布局文件:<FrameLayout> 标签
- Java代码:android.widget.FrameLayout对象实例

一个帧布局中只能有效的显示一个元素。主要用于选项卡视图和图像切换器

帧布局FrameLayout

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:id="@+id/textview1"
        android:layout_width="300dp"
        android:layout_height="300dp"
        android:layout_gravity="center"
        android:background="@android:color/user_icon_7"/>
    <TextView
        android:id="@+id/textview2"
        android:layout_width="240dp"
        android:layout_height="240dp"
        android:layout_gravity="center"
        android:background="@android:color/user_icon_8"/>
    <TextView
        android:id="@+id/textview3"
        android:layout_width="180dp"
        android:layout_height="180dp"
        android:layout_gravity="center"
        android:background="@android:color/user_icon_5"/>
    <TextView
        android:id="@+id/textview4"
        android:layout_width="120dp"
        android:layout_height="120dp"
        android:layout_gravity="center"
        android:background="@android:color/user_icon_6"/>
    <TextView
        android:id="@+id/textview5"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:layout_gravity="center"
        android:background="@android:color/user_icon_2"/>
</FrameLayout>
```



网格布局GridLayout

LinearLayout 嵌套布局

以前Android中最常用的布局类是LinearLayout, 它能将它的子元素们水平排列或垂直排列。当界面布局比较复杂的时候, 也可以利用它嵌套一系列分割出来的LinearLayout子布局来实现, 嵌套的层数通常不宜太深, 只适合于许多简单布局的情形。

嵌套布局有很多显著的缺点,总结起来有这三个方面:

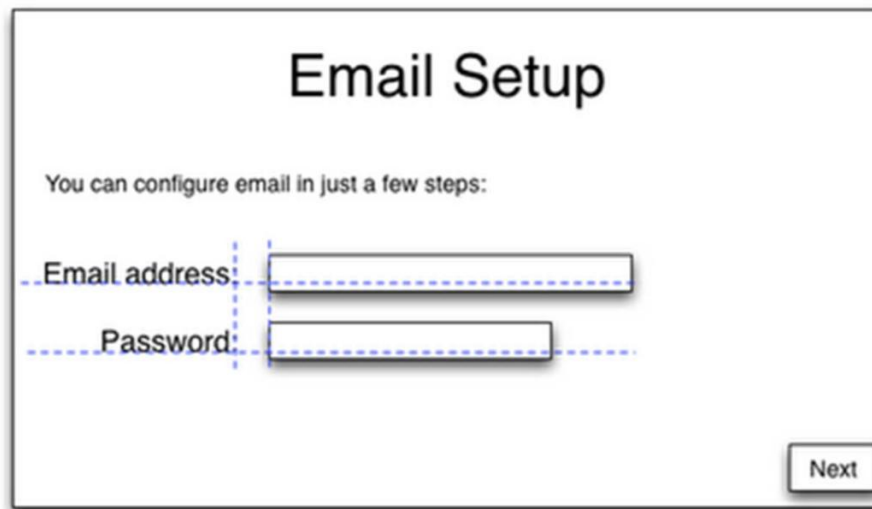
- 1.无法同时在水平和竖直方向对齐;
- 2.嵌套太深影响性能;
- 3.不适用于那些支持自由编辑的设计工具;

网格布局GridLayout

在API Level14中加入网格布局类型, 能将一个视图按照网格的形式进行划分, 并且以“格”为单位来为子视图分配空间, 一个子视图可以占用一格也可以占用多格 (通过属性rowSpan和columnSpan参数进行设置)

网格布局GridLayout

应用实例

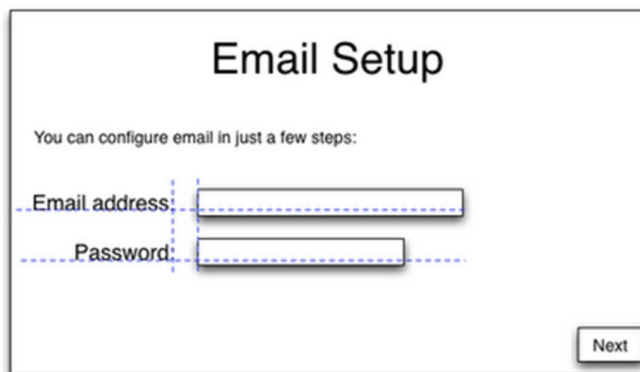


The image shows a dialog box titled "Email Setup". Inside, there is a text label "You can configure email in just a few steps:". Below this, there are two input fields. The first input field is preceded by the label "Email address". The second input field is preceded by the label "Password". Blue dashed lines illustrate the alignment: a vertical line aligns the right edge of the "Email address" label with the right edge of the first input field, and a horizontal line aligns the bottom of the "Email address" label with the bottom of the "Password" label. A "Next" button is located in the bottom right corner of the dialog box.

当文字字体和“Email address”标签文字本身改变的时候,希望标签与它右边的组件的底部基线对齐,同时让它的右边缘与它下方的标签的右边缘对齐。

网格布局GridLayout

- 若用嵌套线性布局做这个会很困难,因为标签本身会去和其他组件在水平和竖直方向上自动对齐。
- 若用表格布局,这种方式会把包含的元素以行和列的形式进行排列,每行为一个TableRow对象,也可以是一个View对象,而在TableRow中还可以继续添加其他的控件,每添加一个子控件就成为一列。但是使用这种布局可能会出现**不能将控件占据多个行或列**的问题,而且渲染速度也不能得到很好的保证



The image shows a dialog box titled "Email Setup". Inside the dialog, there is a message: "You can configure email in just a few steps:". Below this message, there are two input fields. The first field is labeled "Email address:" and the second field is labeled "Password:". Both labels are aligned to the left of their respective input fields. A "Next" button is located at the bottom right corner of the dialog box. The dialog box has a thin border and a light gray background.

网格布局GridLayout

GridLayout通过将容器自身的真实区域切割成行列单元来解决上述问题。如下图,在使用GridLayout之后,“Email address” 标签可以同时属于那底部基线对齐的一行和那右边缘对齐的一列。

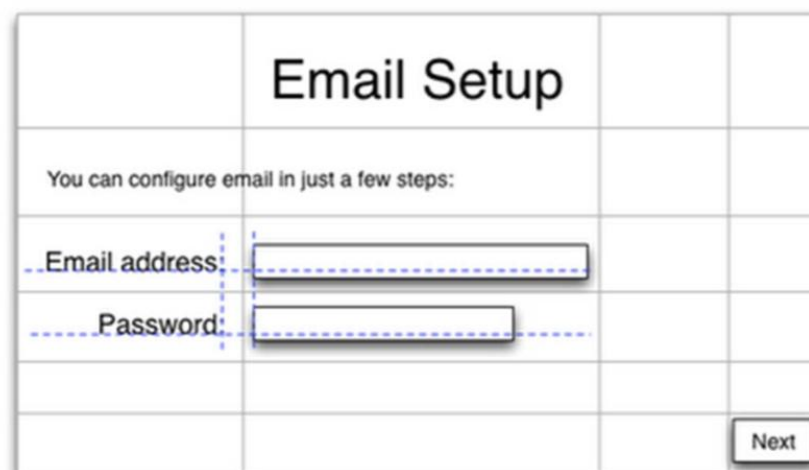
GridLayout用一组无限细的直线将它的绘图区域分割成行、列、单元。它支持行、列拼接合并,这就使得一个子元素控件能够排布在一系列连续单元格组成的矩形区域。

Email Setup			
You can configure email in just a few steps:			
Email address:	<input type="text"/>		
Password:	<input type="password"/>		
			Next

网格布局GridLayout

GridLayout通过将容器自身的真实区域切割成行列单元来解决上述问题。如下图,在使用GridLayout之后,“Email address” 标签可以同时属于那底部基线对齐的一行和那右边缘对齐的一列。

GridLayout用一组无限细的直线将它的绘图区域分割成行、列、单元。它支持行、列拼接合并,这就使得一个子元素控件能够排布在一系列连续单元格组成的矩形区域。



The image shows a screenshot of an "Email Setup" dialog box. The dialog is divided into a grid of cells by thin lines. The title "Email Setup" is in the top row, spanning two columns. Below the title, there is a text label "You can configure email in just a few steps:". The main part of the form consists of two rows of input fields. The first row has a label "Email address:" followed by a text input field. The second row has a label "Password:" followed by a password input field. A "Next" button is located in the bottom right corner of the dialog. Dashed blue lines are overlaid on the grid, highlighting the alignment of the "Email address:" label and its corresponding input field.

网格布局GridLayout

与LinearLayout的相似性

GridLayout的所有XML API与LinearLayout有着一致的语法规则,所以已经使用过LinearLayout的话,上手 GridLayout也是应该很容易的。事实上,它们之间是非常相似的,相似到直接将XML文件中的标签名从 LinearLayout改到GridLayout而无需做其他改变,就可以实现与LinearLayout中相似的UI布局。

网格布局GridLayout

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:alignmentMode="alignBounds"
    android:columnCount="4"
    android:columnOrderPreserved="false"
    android:useDefaultMargins="true">

    <TextView
        android:text="Email setup"
        android:textSize="32sp"
        android:layout_columnSpan="4"
        android:layout_gravity="center_horizontal"/>

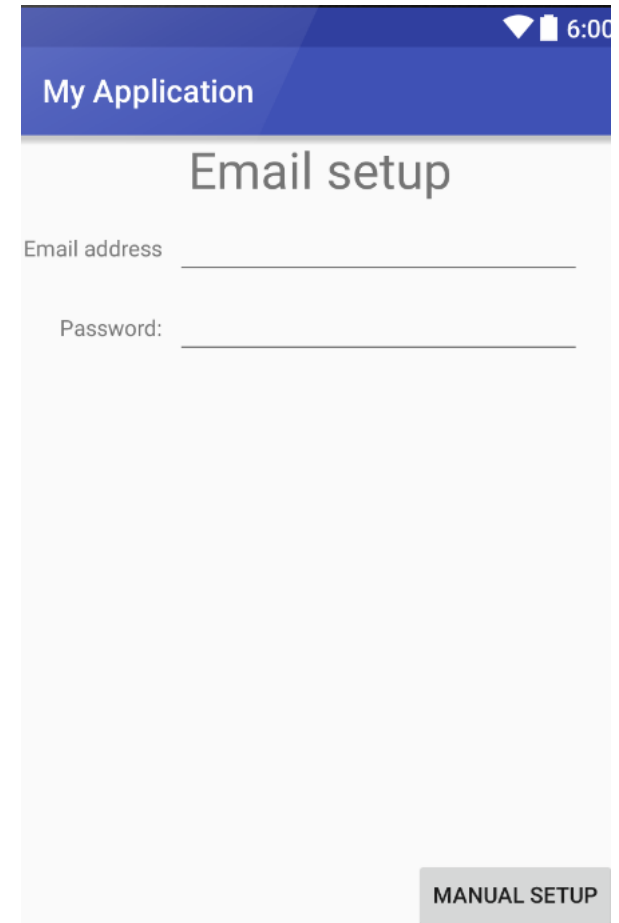
    <TextView
        android:text="Email address"
        android:layout_gravity="end"/>

    <EditText
        android:ems="12"/>

    <TextView
        android:text="Password:"
        android:layout_column="0"
        android:layout_gravity="end"/>

    <EditText
        android:ems="12"/>

    <Button
        android:text="Manual setUp"
        android:layout_row="5"
        android:layout_column="3"/>
</GridLayout>
```

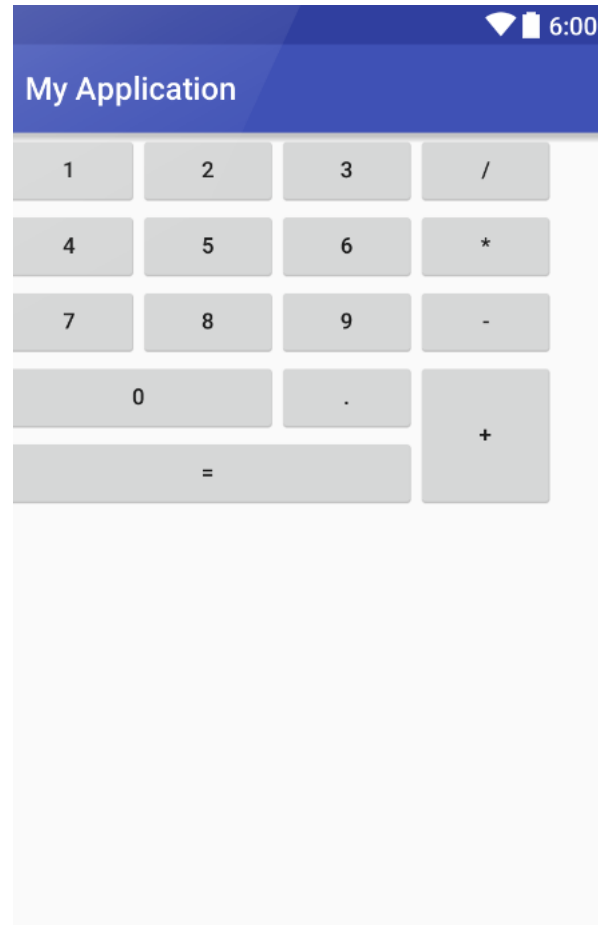


网格布局GridLayout

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:columnCount="4"
    android:rowCount="5">
    <Button
        android:text="1"/>
    <Button
        android:text="2"/>
    <Button
        android:text="3"/>
    <Button
        android:text="/">
    <Button
        android:text="4"/>
    <Button
        android:text="5"/>
    <Button
        android:text="6"/>
    <Button
        android:text="*"/>
```

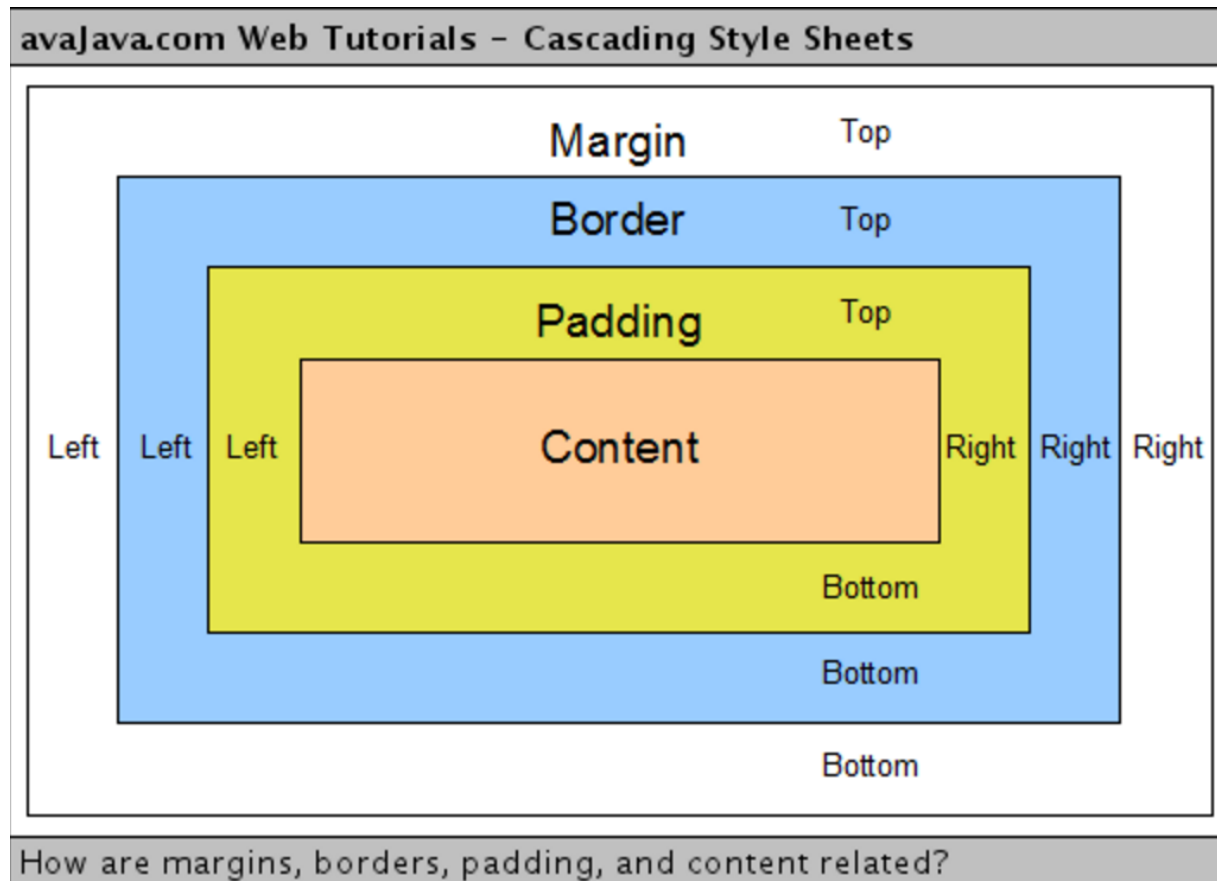
```
<Button
    android:text="7"/>
<Button
    android:text="8"/>
<Button
    android:text="9"/>
<Button
    android:text="-"/>
<Button
    android:text="0"
    android:layout_columnSpan="2"
    android:layout_gravity="fill"/>
<Button
    android:text="."/>
<Button
    android:text="+"
    android:layout_rowSpan="2"
    android:layout_gravity="fill"/>
<Button
    android:text="="
    android:layout_columnSpan="3"
    android:layout_gravity="fill"/>
</GridLayout>
```

网格布局GridLayout



Margin与padding

类似CSS



Margin与padding

padding设置View中的内容在上下左右四个方向距离边缘的距离

layout_margin设置View距离其它View或父容器边缘的距离。

其它

- 引用其他的布局文件

```
<include layout="@layout/content_main" />
```