|  |
| --- |
| 线性布局是Android开发中最常见的一种布局方式，它是按照垂直或者水平方向来布局，通过“android:orientation”属性可以设置线性布局的方向。属性值有垂直（vertical）和水平（horizontal）两种。  RelativeLayout是按照组件之间的相对位置来布局，比如在某个组件的左边，右边，上面和下面等，相对于父view、或者平行view的位置。  帧布局是从屏幕的左上角（0,0）坐标开始布局，多个组件层叠排列，第一个添加的组件放到最底层，最后添加到框架中的视图显示在最上面。上一层的会覆盖下一层的控件。  表格布局是一个ViewGroup以表格显示它的子视图（view）元素，即行和列标识一个视图的位置。  绝对布局是Android 中最为不常用的一种布局方式，通过指定子组件的确切X,Y坐标来确定组件的位置。特别是在屏幕旋转时，往往会出问题，而且多个元素的时候，计算比较麻烦。  约束布局ConstraintLayout 是一个ViewGroup，它的出现主要是为了解决布局嵌套过多的问题，以灵活的方式定位和调整小部件。ConstraintLayout则是使用约束的方式来指定各个控件的位置和关系的，它有点类似于RelativeLayout，但远比RelativeLayout要更强大。 |

## 1、5种布局

在Android中常用的布局有五种：  
 线性布局（LinearLayout）：按照垂直或者水平方向布局的一种组件。  
 相对布局（RelativeLayout）：相对某个组件的布局方式。  
 帧布局（FrameLayout）：组件从屏幕左上方布局组件一层一层。  
 表格布局（TableLayout）：按照行列方式布局组件类似于表格。  
 绝对布局（AbsoluteLayout）：按坐标来布局某个组件。  
 布局管理器本身就是一种组件，既可以当普通的组件使用，它们也是继承ViewGroup类的。

## 线性布局

线性布局是Android开发中最常见的一种布局方式，它是按照垂直或者水平方向来布局，通过“android:orientation”属性可以设置线性布局的方向。属性值有垂直（vertical）和水平（horizontal）两种。

（1）常用的属性：  
android:orientation：可以设置布局的方向 vertical、horizontal。  
android:gravity:用来控制组件的对齐方式。  
layout\_weight：控制各个组件在布局中的相对大小。（引出水平平分的问题。）  
实例代码：  
<?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="wrap\_content"   
        android:orientation="vertical">   
        <EditText   
            android:layout\_width="match\_parent"   
            android:layout\_height="wrap\_content"/>   
    </LinearLayout>   
    <LinearLayout   
        android:layout\_width="match\_parent"   
        android:layout\_height="wrap\_content"   
        android:orientation="horizontal"   
        android:gravity="left">   
        <Button   
            android:layout\_height="wrap\_content"   
            android:layout\_width="wrap\_content"   
            android:text="ok"/>   
        <Button   
            android:layout\_height="wrap\_content"   
            android:layout\_width="wrap\_content"   
            android:text="cancel"/>      
     </LinearLayout>   
</LinearLayout> 

## 3、相对布局

RelativeLayout是按照组件之间的相对位置来布局，比如在某个组件的左边，右边，上面和下面等。  
实例代码：   
<?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="wrap\_content"   
    android:padding="10dp">

//padding填补、填料，内边距 margin外布局 aligin 对齐  
    <TextView      
        android:id="@+id/tv1"   
        android:layout\_width="wrap\_content"     
        android:layout\_height="wrap\_content"     
        android:layout\_marginBottom="20dp"   
        android:text="测试:"/>   
    <EditText   
        android:id="@+id/tx1"   
        android:layout\_width="match\_parent"   
        android:layout\_height="wrap\_content"   
        android:layout\_below="@id/tv1"/>   
    <Button   
        android:id="@+id/btn1"   
        android:layout\_height="wrap\_content"   
        android:layout\_width="wrap\_content"   
        android:layout\_below="@id/tx1"   
        android:layout\_alignParentRight="true"   
        android:text="ok" />   
    <Button   
        android:id="@+id/btn2"   
        android:layout\_height="wrap\_content"   
        android:layout\_width="wrap\_content"   
        android:layout\_below="@id/tx1"   
        android:layout\_toLeftOf="@id/btn1"   
        android:layout\_marginRight="20dp"   
        android:text="cancel"/>  
</RelativeLayout> 

## 4、帧布局

帧布局是从屏幕的左上角（0,0）坐标开始布局，多个组件层叠排列，第一个添加的组件放到最底层，最后添加到框架中的视图显示在最上面。上一层的会覆盖下一层的控件。

实例代码：  
<?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:layout\_width="350dp"     
        android:layout\_height="350dp"     
        android:background="#ff0000"/>   
    <TextView      
        android:layout\_width="250dp"     
        android:layout\_height="250dp"     
        android:background="#00ff00"/>   
    <TextView      
        android:layout\_width="150dp"     
        android:layout\_height="150dp"     
        android:background="#0000ff"/>   
</FrameLayout> 

## 表格布局

表格布局是一个ViewGroup以表格显示它的子视图（view）元素，即行和列标识一个视图的位置。  
表格布局常用的属性如下：  
android:collapseColumns：隐藏指定的列  
android:shrinkColumns：收缩指定的列以适合屏幕，不会挤出屏幕  
android:stretchColumns：尽量把指定的列填充空白部分  
android:layout\_column:控件放在指定的列  
android:layout\_span:该控件所跨越的列数  
  
实例代码：  
<?xml version="1.0" encoding="utf-8"?>   
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"   
    android:layout\_width="fill\_parent"   
    android:layout\_height="fill\_parent">   
    <TableRow>   
        <Button android:text="测试1"/>   
        <Button android:text="测试2" />   
        <Button android:text="测试3"/>   
    </TableRow>   
    <TableRow>   
        <Button android:text="测试4"/>   
        <Button android:layout\_span="2"   
            android:text="测试5"/>   
    </TableRow>       
</TableLayout>

## 6、绝对布局

绝对布局是Android 中最为不常用的一种布局方式，通过指定子组件的确切X,Y坐标来确定组件的位置，在Android2.0 API文档中标明该类已经过期，可以使用FrameLayout或者RelativeLayout来代替。所以这里我也就不再详细介绍。

## 7、约束布局ConstraintLayout

（1）在过去，几乎所有情形下，您都需要一个深度嵌套的布局，因此，ConstraintLayout 应当成为您优化性能和易用性的不二之选。

（2）Systrace 工具可以测试布局的绘制过程的性能：测量、布局、绘制。

可以通过工具测试一下，不同的布局的性能比较。

通常是约束不觉对嵌套层数很深的，优化性能较明显。但也不能一概而论。

（3）ConstraintLayout的使用说明

（4）传统界面布局需要XML代码完成，约束布局特别适合拖拽控件。

1）将默认的RelativeLayout转换成ConstraintLayout布局。

2）拖拽。之后是添加约束，一共可以在四个方向给控件添加约束。

3）删除约束。 删除控件所有的约束。 删除所有控件的约束。

4）Inspector 调整控件的布局位置。

三种模式：wrap content、 固定值、any size（相当于match parent）

1. Guidelines 相当于是引导线，比如两个控件（注册、登录）整体居中。
2. 自动添加约束的方式主要有两种。

一种叫Autoconnect：自动添加约束，不对的地方再更改。

一种叫Inference：一键生成添加所有控件的约束。