

# Android 基础开发

---

## 第二章 第二节 Android界面布局



Java与移动智能设备开发



# 教学目标

- 掌握在Android中创建五种常见的布局。



# 目录



1 Android用户界面布局简介

2 Android中线性布局的使用

3 Android中相对布局的使用

4 Android中表格布局的使用

5 Android中网格布局的使用

6 Android中帧布局的使用



# 问题引入

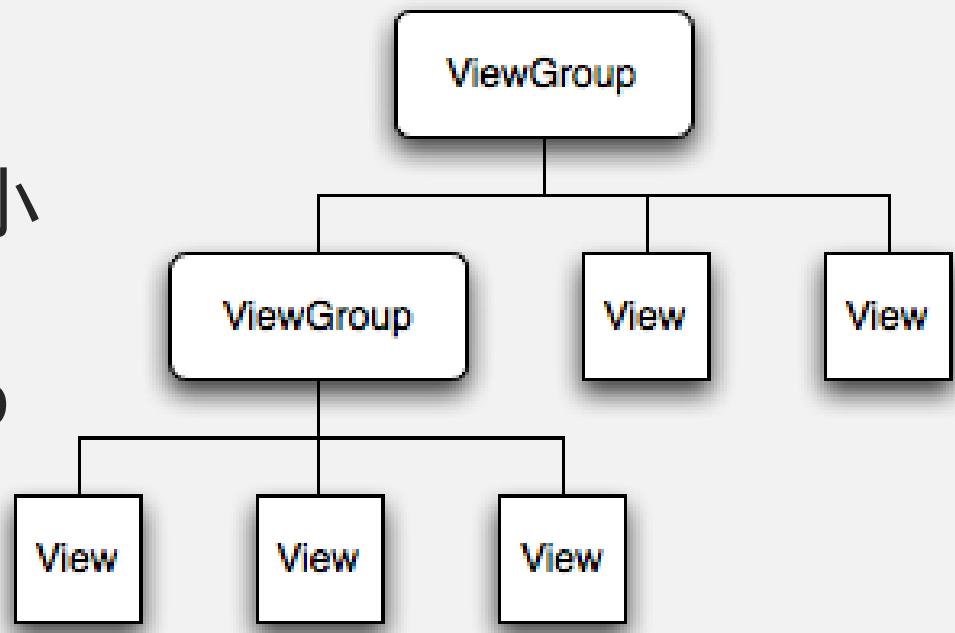
- 布局定义了一个用户界面（UI）中的视觉结构





# Android中视图层次结构

- Android视图层次结构
  - Android中视图按照树形结构进行设计（**视图树**）；而**视图树由View或ViewGroup构成**。
  - View：视图控件，界面可操作的最小可视化元素。
  - ViewGroup：由View或ViewGroup组成的元素组。





# Android中视图层次结构

- Android视图层次结构

```
<?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"
    android:background="#ffffffff" >
    ViewGroup布局
    <TextView
        android:text="@string/username_Label"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:textColor="#000000" />
    View控件
    <EditText
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:id="@+id/username"
        android:hint="@string/username_hint" />
</LinearLayout>
```



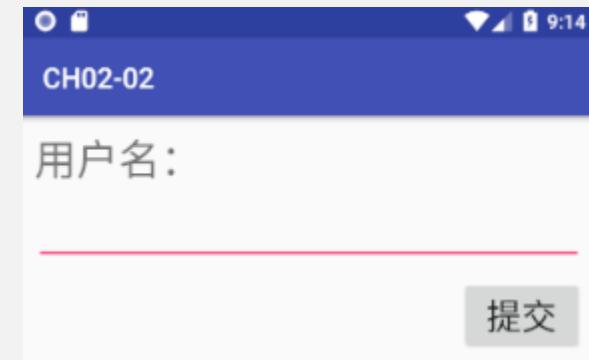
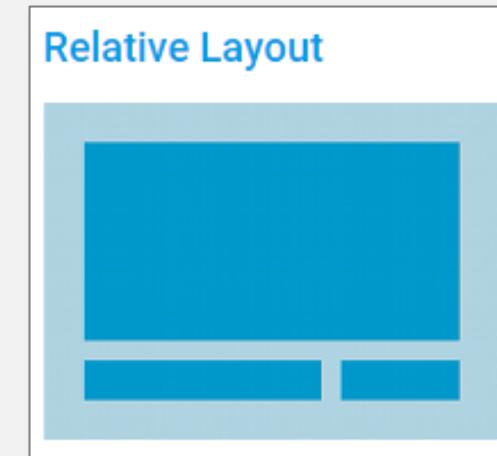
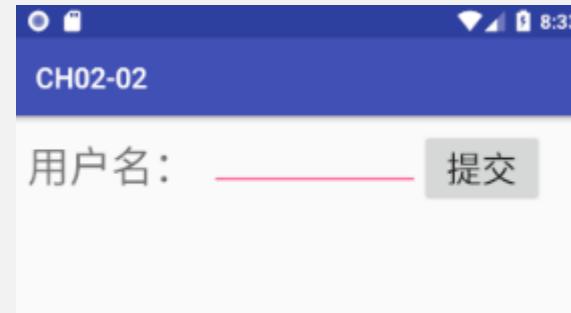
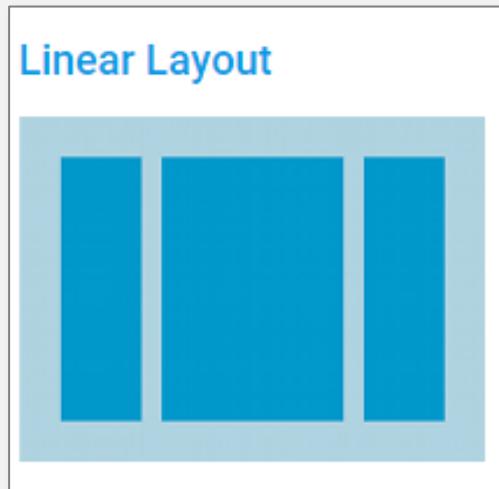
# Android界面布局简介

- Android界面布局：控制子视图对象（View对象或 ViewGroup对象）在界面中的显示方式（即如何显示这些View控件或ViewGroup）。
- Android中内置的常用布局方式有：
  - LinearLayout：线性布局
  - RelativeLayout：相对布局
  - TableLayout：表格布局
  - GridLayout：网格布局
  - FrameLayout：帧布局
  - .....



# Android界面布局简介

- LinearLayout
  - 让所有子视图按照单一方向排列，垂直的或者水平的
- RelativeLayout
  - 让子视图的位置和其他视图位置相关





# Android界面布局简介

- TableLayout
  - 通过表格的形式布局子视图
- GridLayout
  - 将其子视图放在网格中，并且子视图可以占据一个或多个连续的网格
- FrameLayout
  - 为每个子视图创建一个空白区域（称为一帧），每个子视图占据一帧





# 目录



1 Android用户界面布局简介

**2 Android中线性布局的使用**

3 Android中相对布局的使用

4 Android中表格布局的使用

5 Android中网格布局的使用

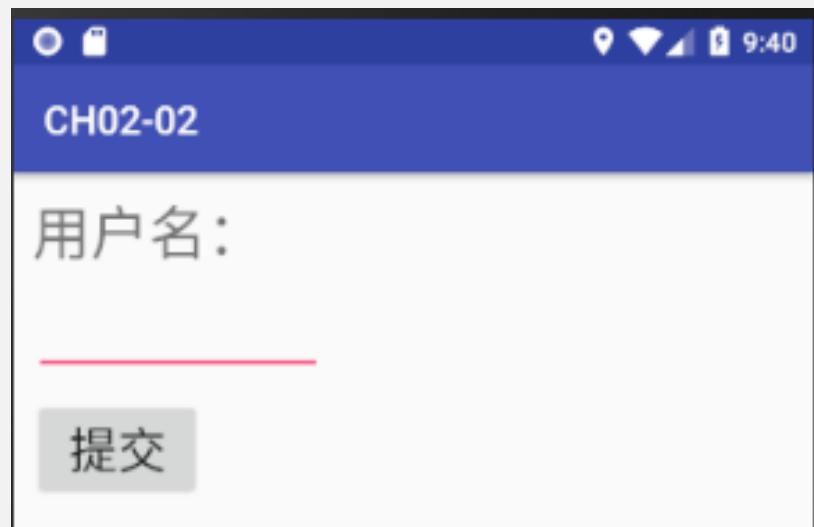
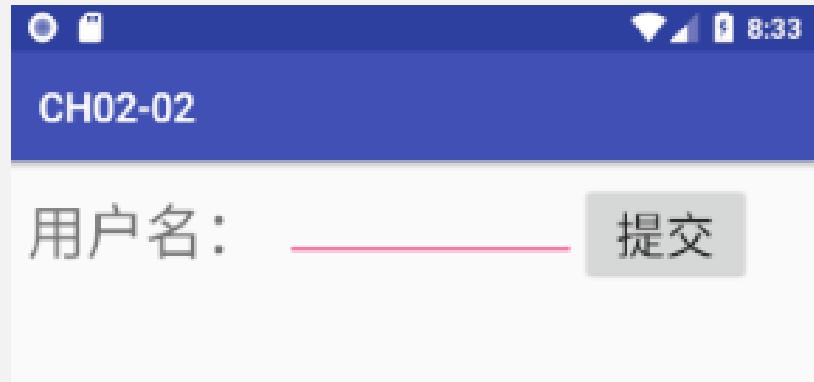
6 Android中帧布局的使用



# Android中线性布局的使用

- 线性布局

- 线性布局（ LinearLayout ）是一种重要的界面布局中，也是经常使用到的一种界面布局
- 在线性布局中，所有的子元素都按照垂直或水平的顺序在界面上排列
  - 如果垂直排列，则每行仅包含一个界面元素
  - 如果水平排列，则每列仅包含一个界面元素





# Android中创建线性布局

- Android中布局创建的方式有两种：
  - 通过XML文件
  - 通过Java代码

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:orientation="horizontal"
    tools:context="com.example.shuangying.myapplication.MainActivity" >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/textviewstr1" />
    <EditText
        android:layout_width="100dp"
        android:layout_height="wrap_content" />
    <Button
```



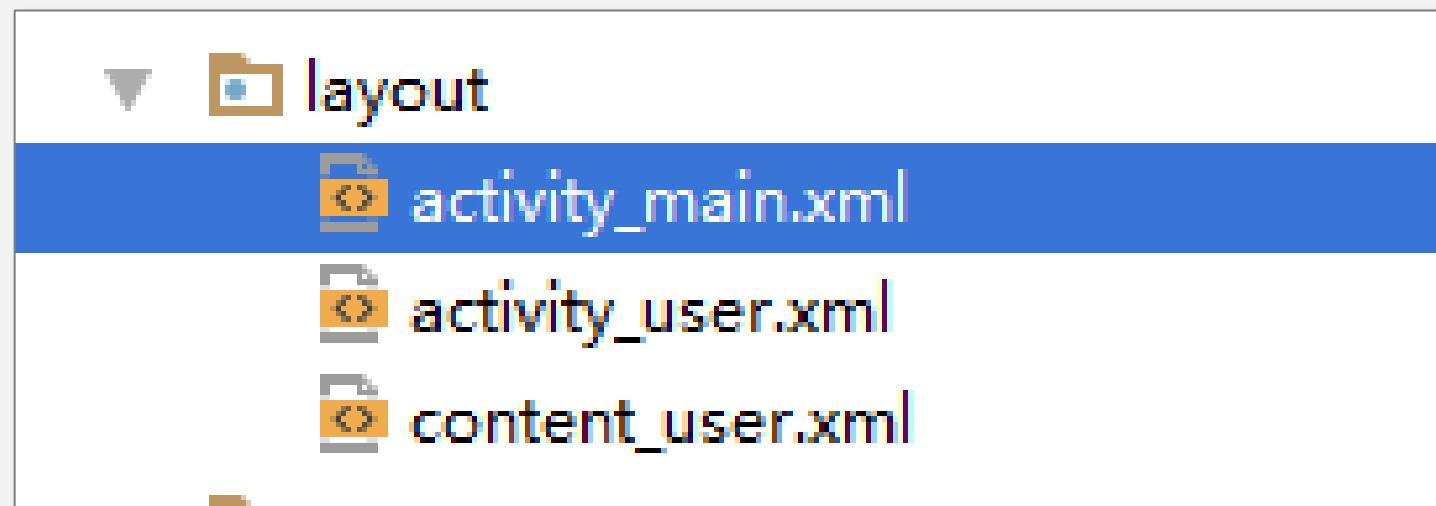
# 使用XML文件创建布局

- 使用XML文件创建用户界面布局的基本流程
  1. 建立XML文件 ( res / layout / \*\*\*.xml文件 )
  2. 在XML文件中设置界面布局
    - 选择根元素 (一般为布局方式)
    - 添加View控件或ViewGroup控件 (嵌套添加)
  3. 在Activity中设置布局文件 ( setContentView方法 )



# Step1：创建XML布局文件

- res/layout/main.xml 文件
  - 文件名必须是**小写字母、数字或下划线**





## Step2：在XML文件中设置界面

- res/layout/main.xml 文件
  - 选择根元素：一般为布局对象，表明界面上整体上采用的布局方式
  - 添加子元素：可以View控件，或ViewGroup控件（布局对象的嵌套使用）
  - 设置元素属性



# XML文件中布局元素的常用属性

- LinearLayout元素的XML属性

属性名	属性值	备注
layout_width layout_height	fill_parent match_parent	布局元素的宽度/高度占满父元素的宽度/高度空间
	wrap_content	布局元素的宽度/高度为其内容宽度/高度
	数值	数值表示的距离单位
orientation	关键字	布局方向，线性水平布局或垂直布局
layout_weight	数值	使用在View控件中，表示当前LinearLayout剩余空间在View控件中的分配情况

[https://developer.android.com/reference/android/widget/LayoutProperties.html#attr\\_android:layout\\_weight](https://developer.android.com/reference/android/widget/LayoutProperties.html#attr_android:layout_weight)



# Step3：在Activity中显示视图

- src/包名/\*\*Activity.java 文件
  - **setContentView(R.layout.布局文件名);**

```
public class MainActivity extends Activity {  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
    }  
}
```



# Android中创建线性布局

- Android中布局创建的方式有两种：
  - 通过XML文件
  - 通过Java代码

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    LinearLayout layout= new LinearLayout(this);  
    layout.setOrientation(LinearLayout.VERTICAL);  
    LinearLayout.LayoutParams params =  
        new LinearLayout.LayoutParams(  
            ViewGroup.LayoutParams.MATCH_PARENT,  
            LinearLayout.LayoutParams.WRAP_CONTENT);  
    TextView tv = new TextView(this);  
    tv.setLayoutParams(params);  
    tv.setText("this is TextView:");  
    layout.addView(tv);  
    setContentView(layout);  
}
```



# 使用Java代码创建界面布局

- 基本格式：
  - 先创建布局元素的对象（ LinearLayout ）
  - 设置布局属性
  - 为布局元素添加子元素（ View 控件或其它布局元素 ）
  - 使用 `setContentView( )` 方法加载布局对象



# Step1：创建布局元素

- 在Activity的onCreate( )回调函数中

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    // setContentView(R.layout.main);  
  
    // 1. 创建布局对象  
    LinearLayout layout = new LinearLayout(this);  
    layout.setOrientation(LinearLayout.VERTICAL);
```



## Step2：设置布局属性

- 在Activity的onCreate()回调函数中

```
// 2.设置布局属性  
LayoutParams params  
    = new LinearLayout.LayoutParams(  
        LayoutParams.MATCH_PARENT,  
        LayoutParams.WRAP_CONTENT);
```



## Step3：添加布局子元素

- 在Activity的onCreate()回调函数中

```
// 3.创建视图控件  
TextView tv = new TextView(this);  
tv.setText("This is a TextView");  
tv.setLayoutParams(params);
```

```
// 把视图控件添加到layout布局对象中  
layout.addView(tv);
```



# Step4：加载布局对象

- 在Activity的onCreate( )回调函数中

```
// 4.为当前Activity显示界面视图  
setContentView(layout);
```

```
}
```



# Android中创建线性布局

- Android中通过XML文件创建布局：
  - 优点：界面与逻辑控制代码相分离，同一个布局文件可适用于多个Activity
  - 缺点：在程序运行前确定界面的布局形式，运行中不易更改
- Android中通过Java代码创建布局：
  - 优点：在程序运行过程中确定界面的布局形式，界面可伴随程序运行过程中修改
  - 缺点：界面与逻辑控制代码在一起，同一个布局文件仅能用于当前Activity



# 目录

- 1 Android用户界面布局简介
- 2 Android中线性布局的使用
- 3 **Android中相对布局的使用**
- 4 Android中表格布局的使用
- 5 Android中网格布局的使用
- 6 Android中帧布局的使用

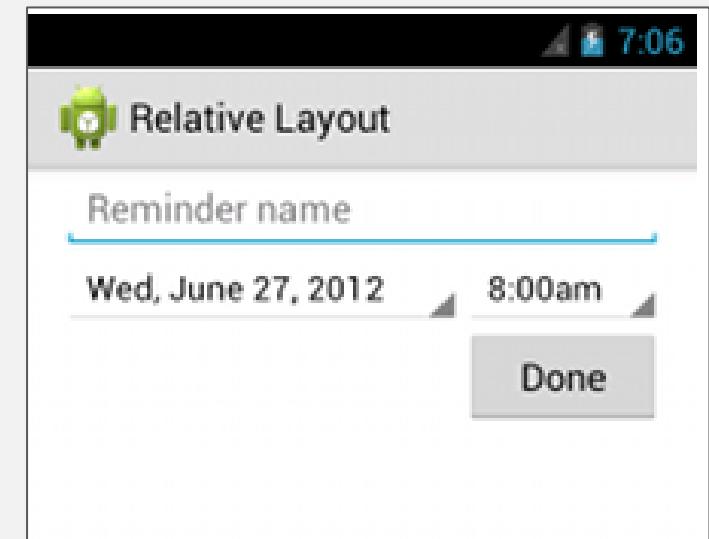




# Android中相对布局的使用

- 相对布局

- 相对布局（RelativeLayout）是一种非常灵活的布局方式，能够通过指定界面元素与其它元素的相对位置关系，确定界面中所有元素的布局位置
- 特点：能够最大程度保证在各种屏幕类型的手机上正确显示界面布局





# XML文件中布局元素的常用属性

- RelativeLayout元素的XML属性
  - 以下属性均使用在RelativeLayout元素的**子元素**中

属性名	属性值	备注
layout_toLeftOf	ID值	当前控件位于给定ID元素控件的左边
layout_alignLeft	ID值	当前控件与给定ID元素控件的左边对齐
layout_alignParentLeft	true/false	当前控件是否与父元素的左侧对齐
layout_below	ID值	当前控件位于给定ID元素控件的下方
layout_alignTop	ID值	当前控件位于给定ID元素控件的上边对齐
layout_alignParentTop	true/false	当前控件是否与父元素的上侧对齐

<https://developer.android.com/reference/android/widget/RelativeLayout.html>



# 相对布局示例

- 相对布局
  - 相对布局在main.xml文件的代码示例如下：

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <RelativeLayout android:id="@+id/RelativeLayout01"
3.     android:layout_width="fill_parent"
4.     android:layout_height="fill_parent"
5.     xmlns:android="http://schemas.android.com/apk/res/android">
6.     <TextView android:id="@+id/label"
7.             android:layout_height="wrap_content"
8.             android:layout_width="match_parent"
9.             android:text="用户名: ">
10.    </TextView>
11.    <EditText android:id="@+id/entry"
12.              android:layout_height="wrap_content"
13.              android:layout_width="match_parent"
14.              android:layout_below="@id/label">
15.    </EditText>
```



# 相对布局示例

```
16.    <Button android:id="@+id/cancel"  
17.        android:layout_height="wrap_content"  
18.        android:layout_width="wrap_content"  
19.        android:layout_alignParentRight="true"  
20.        android:layout_marginLeft="10dip"  
21.        android:layout_below="@id/entry"  
22.        android:text="取消" >  
23.    </Button>  
24.    <Button android:id="@+id/ok"  
25.        android:layout_height="wrap_content"  
26.        android:layout_width="wrap_content"  
27.        android:layout_toLeftOf="@id/cancel"  
28.        android:layout_alignTop="@id/cancel"  
29.        android:text="确认">  
30.    </Button>  
31.</RelativeLayout>
```



# 目录

1 Android用户界面布局简介

2 Android中线性布局的使用

3 Android中相对布局的使用

4 **Android中表格布局的使用**

5 Android中网格布局的使用

6 Android中帧布局的使用



# Android中表格布局的使用

- 表格布局
  - 表格布局（TableLayout）也是一种常用的界面布局，继承了LinearLayout，采用**行和列**的形式来管理UI组件
    - 表格的边界对用户是不可见的
    - 表格布局还支持嵌套，可以将另一个表格布局放置在前一个表格布局的单元格中，也可以在表格布局中添加其他界面布局，例如线性布局、相对布局等等



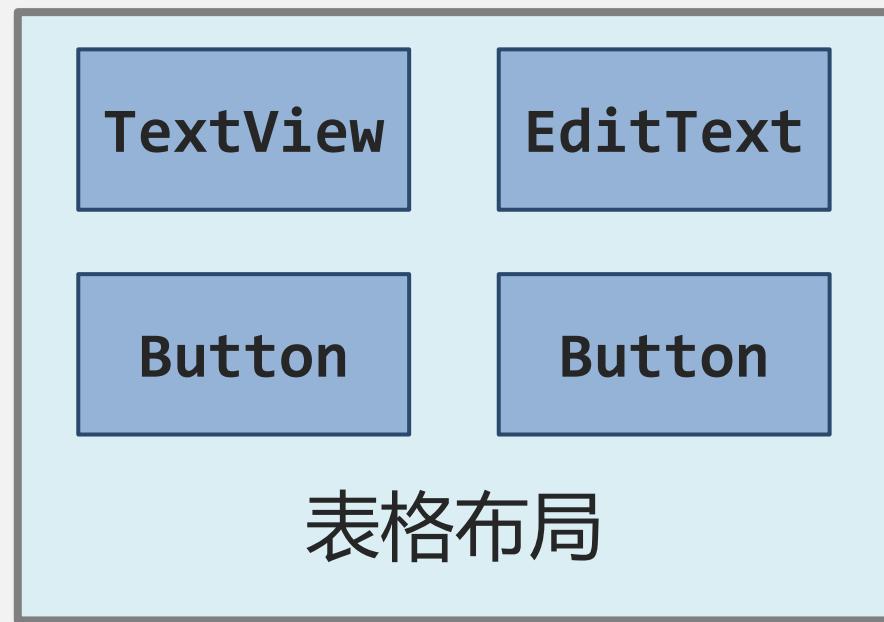
# Android中表格布局的使用

- 表格布局

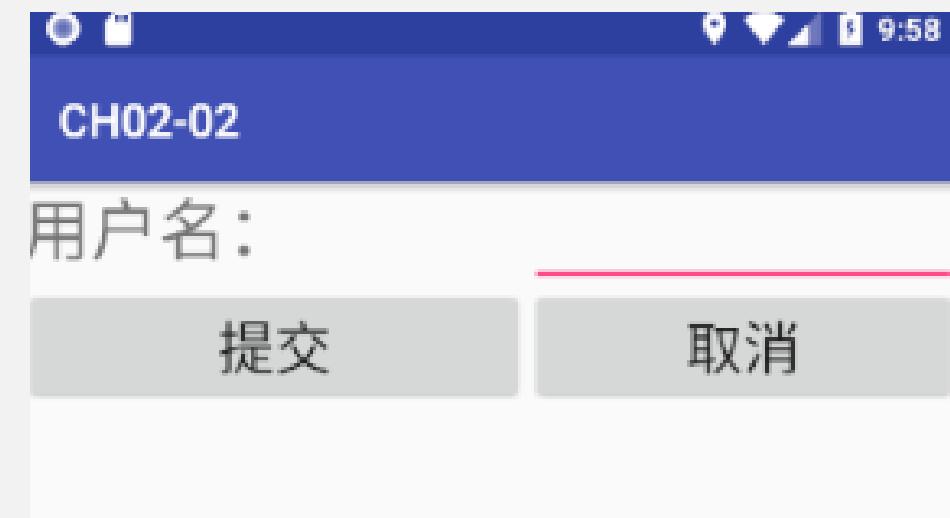
- 表格布局示意图

- 表格布局效果图

TableRow



TableRow





# XML文件中布局元素的常用属性

- TableLayout元素的XML属性
  - 以下属性均使用在TableLayout元素中

属性名	属性值	备注
stretchColumns	ID值	设置自动伸展哪些列，列ID从0开始，多格列的话用“,”分隔
shrinkColumns	ID值	设置自动收缩哪些列，列ID从0开始，多格列的话用“,”分隔
collapseColumns	ID值	设置隐藏哪些列，列ID从0开始，多个列的话用“,”分隔
layout_column	ID值	设置当前控件在哪一列
layout_span	数值	设置当前控件占据几列

<https://developer.android.com/reference/android/widget/TableLayout.html>



# 表格布局示例





# 表格布局示例

- 表格布局
  - 表格布局在main.xml文件的代码示例如下：

```
1. <TableLayout android:layout_width="match_parent"
2.     android:layout_height="match_parent"
3.     android:stretchColumns="0,1,2">
4.     <Button android:layout_width="wrap_content"
5.             android:layout_height="wrap_content"
6.             android:text="我占据一行"/>
7.     <TableRow>
8.         <Button android:layout_width="wrap_content"
9.                 android:layout_height="wrap_content"
10.                android:text="第0列"/>
11.        <Button android:layout_width="wrap_content"
12.                android:layout_height="wrap_content"
13.                android:text="第1列"/>
14.    </TableRow>
```



# 表格布局示例

```
15. <TableRow>
16.     <Button android:layout_width="wrap_content"
17.             android:layout_height="wrap_content"
18.             android:text="第0列"/>
19.     <Button android:layout_width="wrap_content"
20.             android:layout_height="wrap_content"
21.             android:layout_span="2"
22.             android:text="我占据两列"/>
23. </TableRow>
24. <TableRow>
25.     <Button android:layout_width="wrap_content"
26.             android:layout_height="wrap_content"
27.             android:layout_column="2"
28.             android:text="我在第2列"/>
29. </TableRow>
30.</TableLayout>
```



# 目录



1 Android用户界面布局简介

2 Android中线性布局的使用

3 Android中相对布局的使用

4 Android中表格布局的使用

5 Android中网格布局的使用

6 Android中帧布局的使用



# Android中网格布局的使用



- 网格布局 ( GridLayout ) 是Android4.0新增的布局管理器
  - 它把整个容器划分成 $\text{rows} \times \text{columns}$ 个网格
  - 每个网格可以放置一个组件
  - 也可以设置一个组件横跨多少列、或者设置一个组件纵跨多少行



# XML文件中布局元素的常用属性

- GridLayout的XML属性

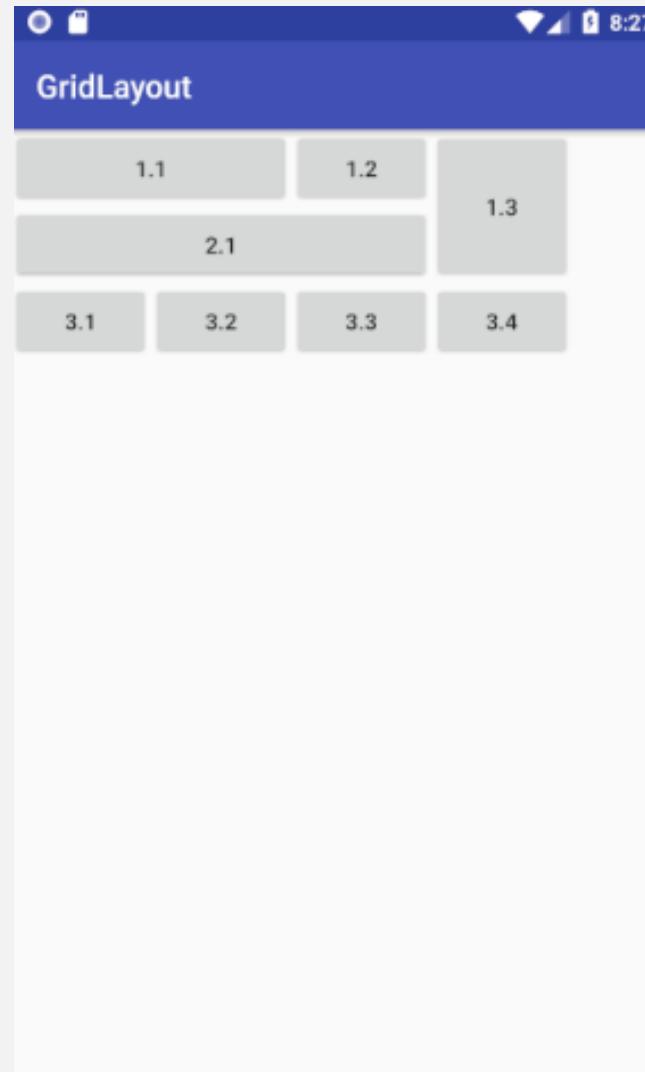
属性名	属性值	备注
android:columnCount	数值	设置网格列的数量
android:rowCount	数值	设置网格行的数量

- GridLayout中子视图的XML属性

属性名	属性值	备注
android:layout_columnSpan	数值	设置子组件在GridLayout中横向跨几列
android:layout_rowSpan	数值	设置子组件在GridLayout中纵向跨几行



# 网格布局示例





# 网格布局示例

```
<GridLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:columnCount="4"  
        android:rowCount="3">  
    <Button  
        android:layout_columnSpan="2"  
        android:layout_gravity="fill"  
        android:text="1.1"/>  
    <Button  
        android:text="1.2"/>  
    <Button  
        android:text="1.3"  
        android:layout_rowSpan="2"  
        android:layout_gravity="fill"/>
```



# 网格布局示例

```
<Button  
        android:layout_columnSpan="3"  
        android:layout_gravity="fill"  
        android:text="2.1"/>  
<Button  
        android:text="3.1"/>  
<Button  
        android:text="3.2"/>  
<Button  
        android:text="3.3"/>  
<Button  
        android:text="3.4"/>  
  
</GridLayout>
```



# 目录

1 Android用户界面布局简介

2 Android中线性布局的使用

3 Android中相对布局的使用

4 Android中表格布局的使用

5 Android中网格布局的使用

6 Android中帧布局的使用



# Android中帧布局的使用

- 帧布局
  - 帧布局（FrameLayout）又称为框架布局，是最简单的界面布局，所有放在布局内的控件，都按照层次堆叠在屏幕左上角。
  - 如果有多个控件，后放置的子元素将遮挡先放置的控件，即默认情况下FrameLayout里的控件是左上角对齐。
  - FrameLayout 就像画布，固定从屏幕的左上角开始填充图片，文字等。



# XML文件中布局元素的常用属性

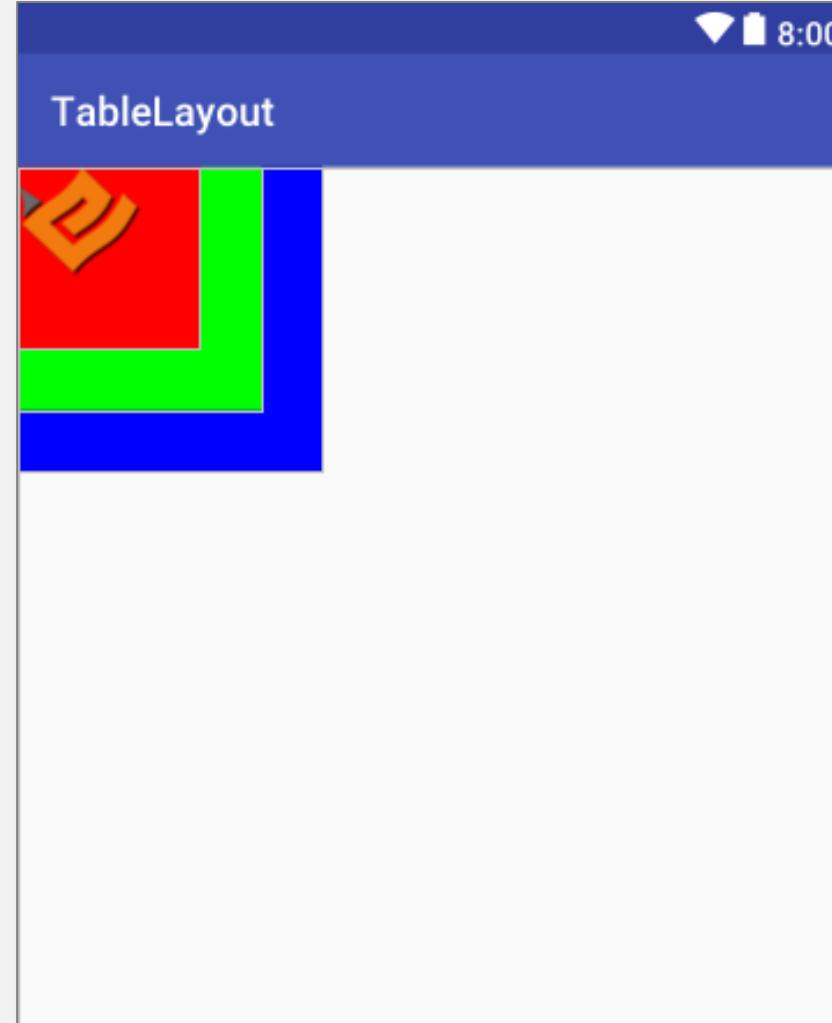
- FrameLayout元素的XML属性
  - 前景图像：永远处于框架布局最上层，直接面对用户的图像，就是不会被覆盖的图片。
  - 以下属性均使用在FrameLayout元素中

属性名	属性值	备注
foreground	图片	设置前景图像的图片
foregroundGravity	位置	设置前景图像的位置

<https://developer.android.com/reference/android/widget/FrameLayout.html>



# 框架布局示例





# 框架布局示例

- 框架布局
  - 框架布局在main.xml文件的代码示例如下：

```
1. <FrameLayout android:layout_width="match_parent"
2.       android:layout_height="match_parent"
3.       android:foreground="@drawable/logo"
4.       android:foregroundGravity="top|left">
5.       <Button android:layout_width="150dp"
6.               android:layout_height="150dp"
7.               android:background="#0000FF"/>
8.       <Button android:layout_width="120dp"
9.               android:layout_height="120dp"
10.              android:background="#00FF00"/>
11.       <Button android:layout_width="90dp"
12.               android:layout_height="90dp"
13.               android:background="#FF0000"/>
14. </FrameLayout>
```



# 目录

1 Android用户界面布局简介

2 Android中线性布局的使用

3 Android中相对布局的使用

4 Android中表格布局的使用

5 Android中网格布局的使用

6 Android中帧布局的使用



# 页面布局练习





# 页面布局练习





# 页面布局练习





## 补充：

- `minSdkVersion`：当前APK所能安装的最低手机API Level
- `compileSdkVersion`：编译当前APK的sdk版本，默认使用当前有API Level中最高的
- `targetSdkVersion`：此APK兼容的最新的手机API Level，同`compileSdkVersion`



# Android中的单位（补）

- 屏幕尺寸：3.2寸、4寸-4.8寸、5寸、7寸、10寸
- 屏幕的分辨率：
- 480x800, 480x854, 540x960, 720x1280, 800x1280
- 屏幕密度：点数/inch



# Android中的单位（补）

- density值表示每个英寸有多少个显示点
- 分辨率：屏幕长宽的显示点数
  - QVGA屏density=120
  - WQVGA屏density=120
  - HVGA屏density=160
  - WVGA屏density=240



# Android中的单位（补）

- px(pixels)-像素：不同的设备显示效果相同。
- dip(device independent pixels)-设备独立像素：这个和设备的硬件有关，一般我们为了支持WCGA、HVGA和QVGA推荐使用这个，不依赖像素，**等同于dp**。
- pt(points)-磅： $1\text{pt} = 1/72\text{英寸}$
- in(inches)-英寸
- mm(millimeters)-毫米
- sp(scaled pixels)-放大像素：主要用于字体显示



# Android中的单位（补）

- 字体大小一般使用sp，此单位的字体能够根据用户的设置而自动缩放
- 空间等相对距离一般使用dp，随密度变化而变化
- px与实际像素有关，与密度有关，不建议使用



Thank you!

