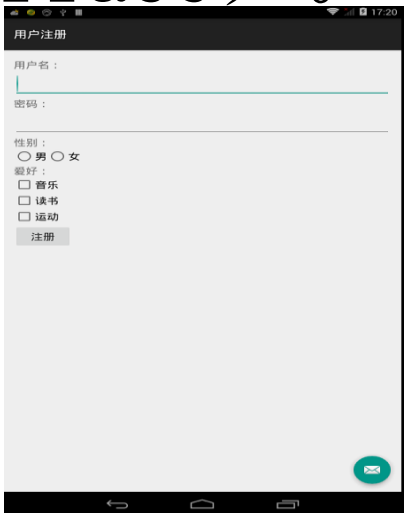
Android App的开发-用户界面基础

本章大纲

- Android用户界面的工作机制
- 基本视图控件的使用
- 事件监听器的使用
- UI布局

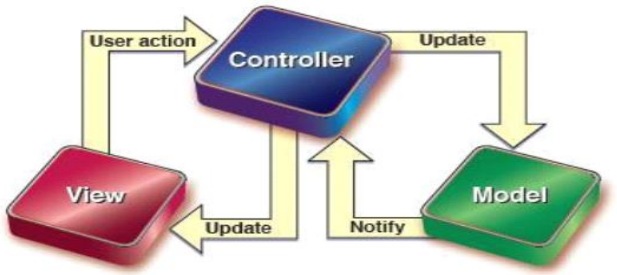
用户界面简介

- 在Android应用中，每一个屏幕就是一个Activity，每个Activity由一个布局来决定如何显示，这就是（User Interface）。



Android中UI工作机制

- Android用户界面采用MVC（Model-View-Controller）框架来接收用户动作、显示UI界面与及处理数据等工作。
  - 控制器：处理用户的数据。
  - 视图：显示用户界面，与用户交互。
  - 模型：数据模型。



MVC 关系图

Android中UI工作机制

- Android用户界面MVC模式
  - 控制器层
    - 控制器负责接受用户动作请求（如按键动作或触摸屏动作等），调用指定模型处理用户请求（如读取数据库、发送网络请求等），响应用户结果（如返回视图界面等）。
    - 在Android系统中，控制器的责任由Activity承担，意味着Activity负责接收用户请求、调用模型方法、响应用户界面等操作（Activity不应承担过多业务逻辑（应交给模型层））。

Android中UI工作机制

- Android用户界面MVC模式
  - 模型层
    - 模型层负责对数据的操作、对网络服务等的操作。
    - 在Android中，数据库/文件操作、ContentProvider、网络访问等等充当模型层。

Android中UI工作机制

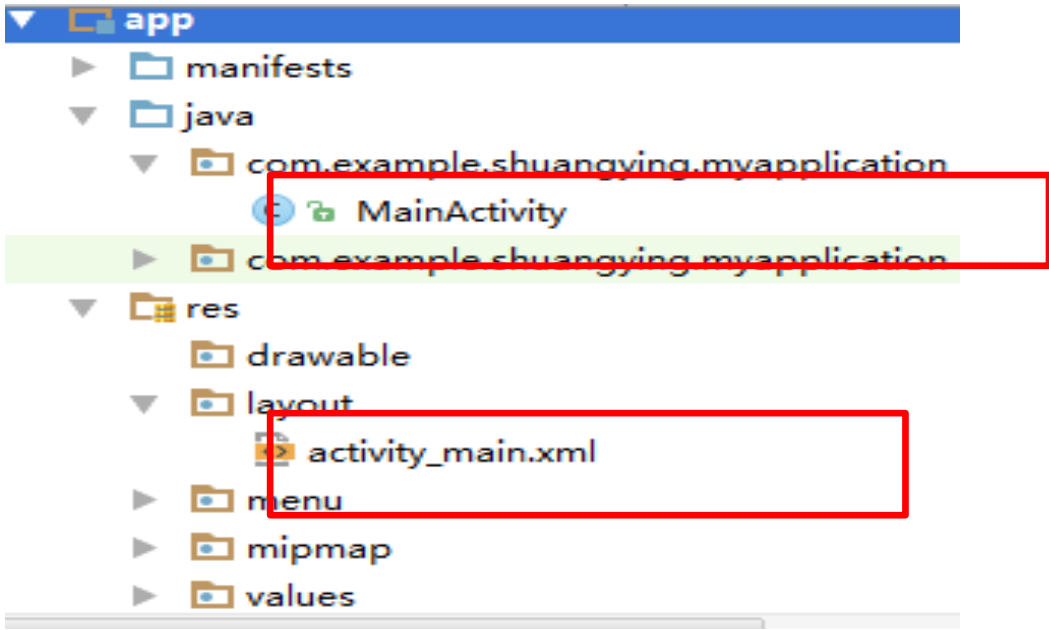
- Android用户界面MVC模式
  - 视图层
    - 视图层主要负责用户界面（UI）的展示。
    - 在Android中使用XML布局文件实现视图层和模型层的分离。

Android中如何实现MVC分离

- Android中视图层与控制层的分离

控制器层

视图层



Android中MVC如何整合到一起

- Android视图层与控制器层、模型层的整合
  - 在Activity文件，使用setContentView( )方法，确定当前Activity显示哪个视图

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

用户界面文件

Android界面原理

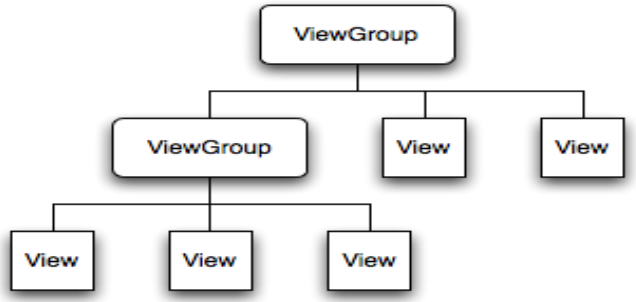
1. 分析Activity，编写布局
2. 在Activity中建立相应对象，设置属性
3. 在Activity中建立对象，设置相应监听器方法
4. 设计逻辑

本章大纲

- Android用户界面的工作机制
- 基本视图控件的使用
- 事件监听器的使用
- UI布局

Android中视图层的使用

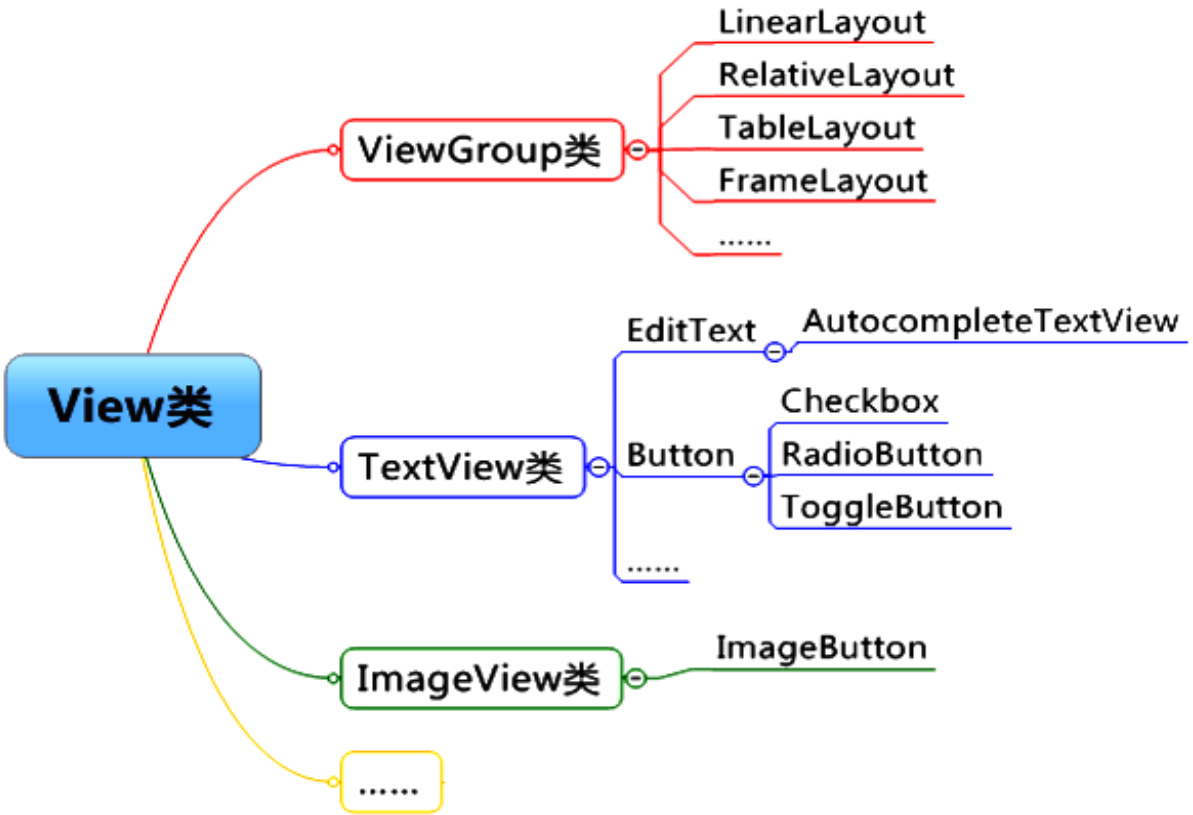
- Android视图层简介
  - 视图层采用视图树（View Tree）模型：用户界面中的界面元素以树型结构组织在一起，整个视图界面为一个视图树模型。
  - 视图树：由View控件或ViewGroup构成。



Android中视图层的使用

- Android视图层简介
  - View控件是界面的最基本的可视单元，是Android视图界面的基类。
    - 例如：文本（TextView）、输入框（EditText）……
  - ViewGroup是由其它View或ViewGroup组成的显示单元，继承自View类。
    - ViewGroup功能：提供了一种布局方法，可以按照该布局定制视图的外观和顺序
    - 例如：LinerLayout、FrameLayout……

View类及其子类的层次关系



在Android中创建视图界面

- Android中创建用户视图界面基本流程
  - 确定视图界面所采用的布局方式（暂用LinearLayout）
  - 为视图界面添加视图组件

在Android中创建视图界面

- Android中创建视图界面有3种方法：
  - 使用可视化编辑方式，创建用户视图界面
    - 最简单的布局方式，但不适合创建复杂布局
  - 使用XML代码方式，创建用户视图界面
    - 最常用的布局方式，但只能创建静态界面
    - 使用findViewById( )方法得到对象
  - 使用Java代码方式，动态创建用户视图界面
    - 最灵活的布局方式，但复杂度较大

在Android中创建视图界面

- 最基本的视图组件：
  - TextView
  - EditText
  - RadioButton
  - Checkbox
  - Button
  - ……

使用基本的视图组件

- XML布局中，视图控件常用的公有布局属性有：

属性名	属性值	说明
android:background	图片资源或颜色值	控件的背景
android:id	@+id/字符串	控件的标识符（在程序中可以通过id获得该控件引用）
android:layout_width	match_parent（铺满父容器） wrap_content（由内容决定） 数据值（固定值）	控件在其父容器中的显示宽度
android:layout_height		控件在其父容器中的显示高度
android:layout_gravity	关键字	控件在其父容器中的相对位置
android:layout_margin	数值	控件在其父容器中与界面四周的页边距
android:padding	数值	控件四周的填充（内边距）

https://developer.android.google.cn/reference/android/view/View

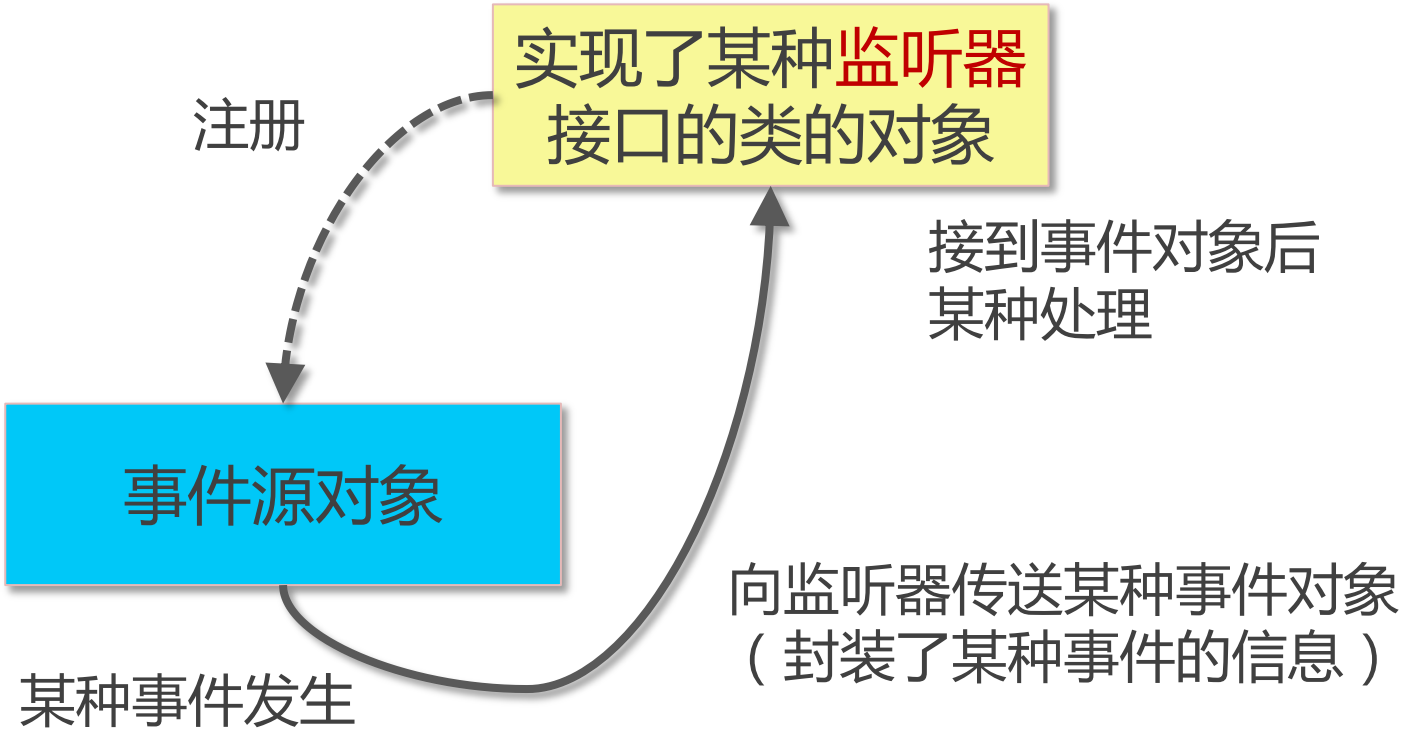
本章大纲

- Android用户界面的工作机制
- 基本视图控件的使用
- 事件监听器的使用
- UI布局

Android中常见的事件监听器

- 使用findViewById获取UI控件对应的对象后，不但可以设置相应属性，而且可以设置相应的事件监听
- 例如：
  - Button点击事件
  - 控件得到焦点，失去焦点事件
  - View的长按事件
  - 屏幕的触摸事件
  - 键盘事件等

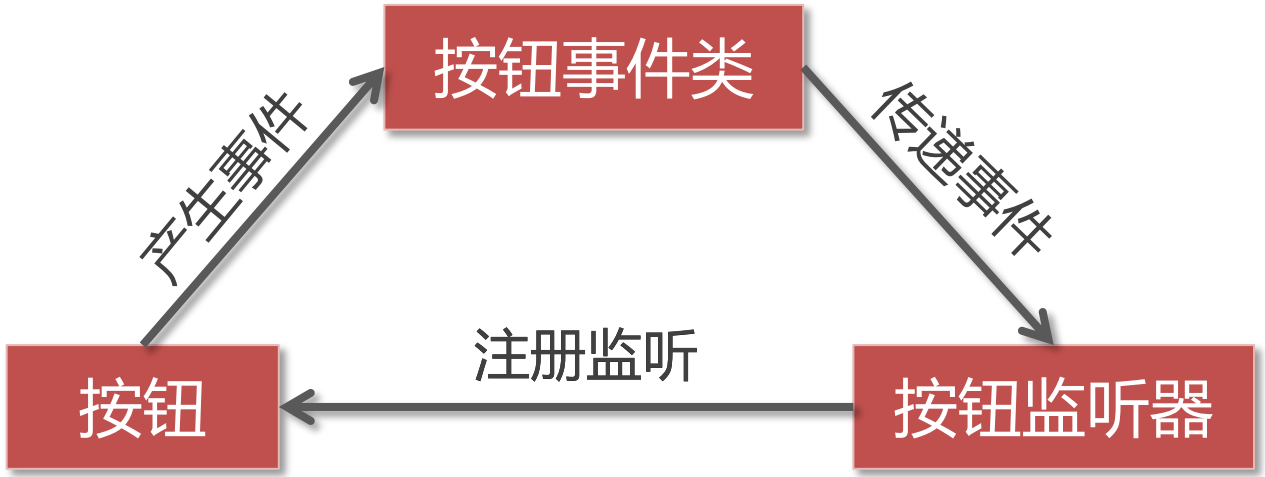
事件监听器实现


实现了某种监听器
接口的类的对象
注册
接到事件对象后
某种处理
事件源对象
向监听器传送某种事件对象
（封装了某种事件的信息）
某种事件发生

事件监听器实现

- 事件的处理步骤为：
  - 事件源上触发一个事件。比如用户按下鼠标、按下按钮等
  - 系统会自动产生对应的事件对象EventObject，并通知所有授权的事件监听者
  - 事件监听者中有对应的事件处理方法来处理该事件

事件监听器实现



基本视图控件的事件监听器

- 为视图控件绑定事件监听器的步骤
  - 获得视图控件对象
  - 设置事件监听类型
  - 绑定事件监听器

```
控件对象.setOn事件类型Listener(new On事件类型Listener() {
    public 返回类型 on事件类型(View v, ...) {
        // TODO Code Here
    }
});
```

基本视图控件的事件监听器

- Button控件的事件监听器

```
btn.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
    }
});
```

视图控件的常用事件类型

- TextView控件
  - Click、LongClick、Touch、CreateContext、FocusChange、Key、……
- EditText控件：继承父类（TextView）
- Button：继承父类（TextView）
- CheckBox/RadioGroup：继承TextView
  - CheckedChange
- ……

为视图控件绑定事件监听器

- 实例：扩充用户注册实例，实现用户注册信息的数据校验
  - 当“用户名”/“密码”输入完毕后校验用户输入信息，保证用户名/密码在6个字符以上，否则提示用户重新输入。

本章大纲

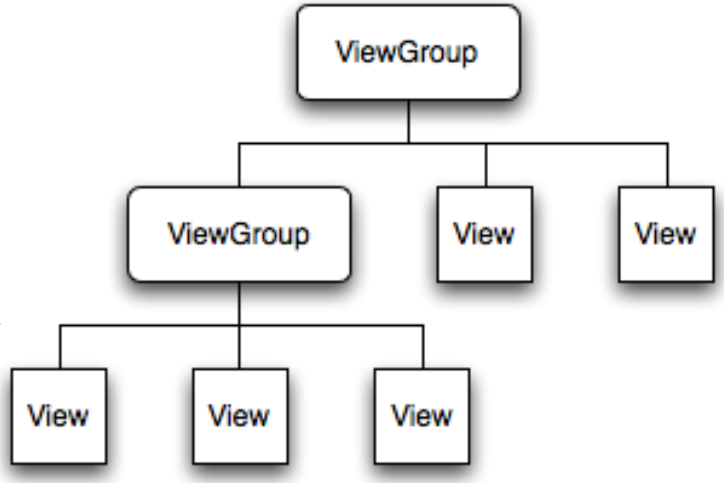
- Android用户界面的工作机制
- 基本视图控件的使用
- 事件监听器的使用
- UI布局

UI界面


芳菲四月天 拒绝宅 出门嗨！
规则
芳菲四月天
不要宅 出门嗨
¥50春日大礼包
抢
解春乏 洗浴桑拿
安然纳米
单人汗蒸+艾灸，节假日通用，免费WiFi
¥66.8 ¥98
抢
安然纳米汗蒸
67.2km
超优惠双人汗蒸，节假日通用，免费WiFi
¥45.8 立减4元 ¥76
抢
黄冈本周桑拿/洗浴/汗蒸好评排行top6
安然纳米汗蒸（八一路店）
67.8km
单人汗蒸1次，节假日通用，免费WiFi，免费停车位
¥20 ¥38
抢
黄冈本周桑拿/洗浴/汗蒸人气top7
安然纳米
68km
双人汗蒸，节假日通用，免费WiFi，免费停车位
¥38 ¥76
抢

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="#ffffff" >

    <TextView
        android:text="@string/username_label"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:textColor="#000000" />

    <EditText
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:id="@+id/username"
        android:hint="@string/username_hint" />

</LinearLayout>
```

ViewGroup布局

View控件

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（文件名必须是小写字母、数字或下划线）
  - 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）

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（ConstraintLayout ）
  - 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

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：
  - ConstraintLayout：约束布局
  - LinearLayout：线性布局
  - FrameLayout：帧布局
  - TableLayout：表格布局
  - ……

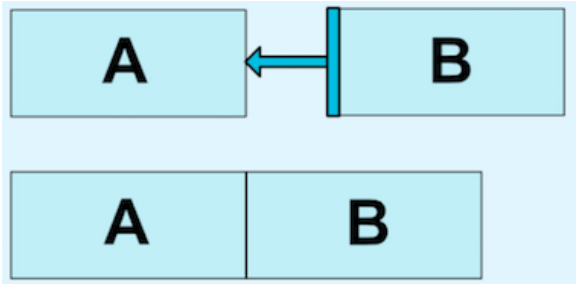
Android界面布局简介

- ConstraintLayout 约束布局

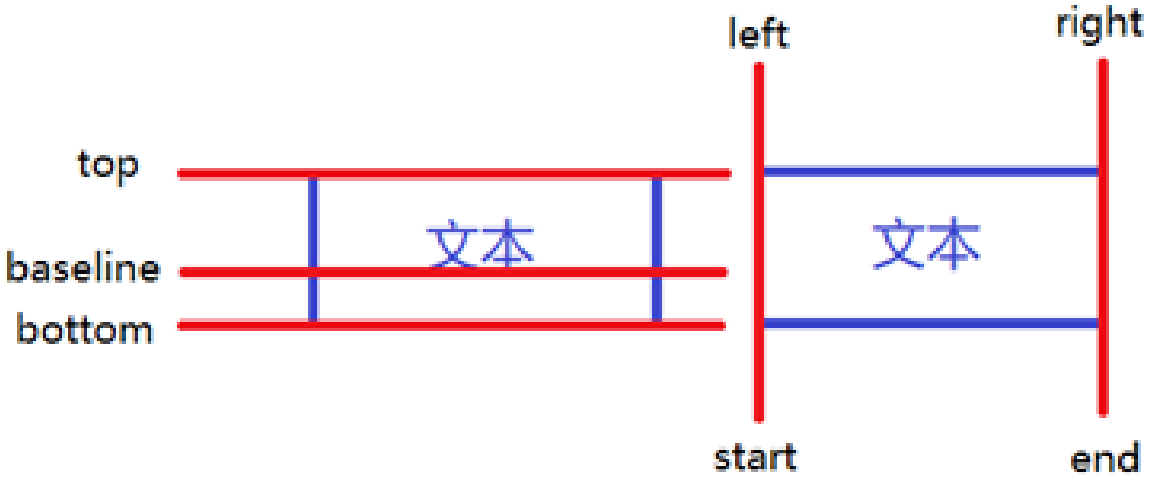
使用起来比RelativeLayout更灵活，性能更出色，而且ConstraintLayout可以按照比例约束控件位置和尺寸，能够更好地适配屏幕大小不同的机型。

例：控件B要放在控件A的右侧，可以使用 layout_constraintLeft_toRightOf属性。

```
<Button android:id="@+id/buttonA" ... />
<Button android:id="@+id/buttonB" ...
    app:layout_constraintLeft_toRightOf="@+id/buttonA" />
```



Android界面布局简介



Android中线性布局的使用

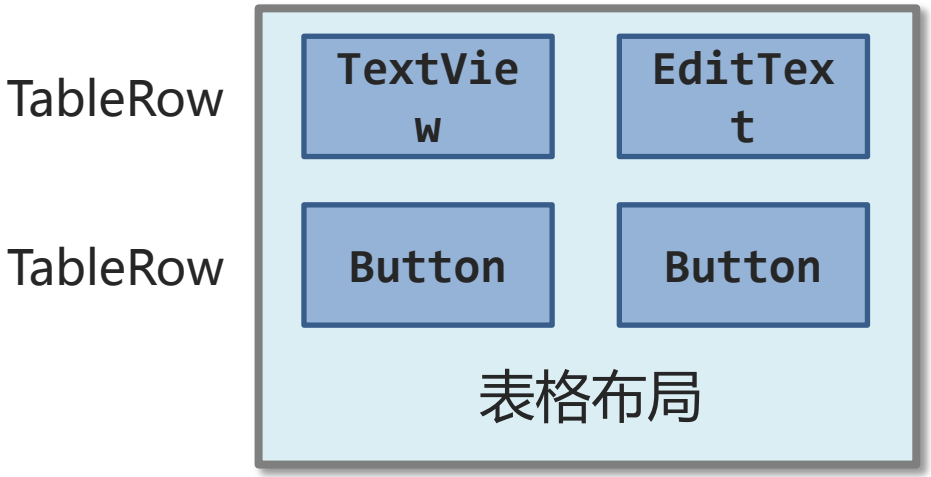
- 线性布局
  - 线性布局（LinearLayout）是一种重要的界面布局中，也是经常使用到的一种界面布局
  - 在线性布局中，所有的子元素都按照垂直或水平的顺序在界面上排列
    - 如果垂直排列，则每行仅包含一个界面元素
    - 如果水平排列，则每列仅包含一个界面元素

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
    - TableRow中的组件个数就决定了该行有多少列,而列的宽度由该列中最宽的单元格决定

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

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>
```

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。

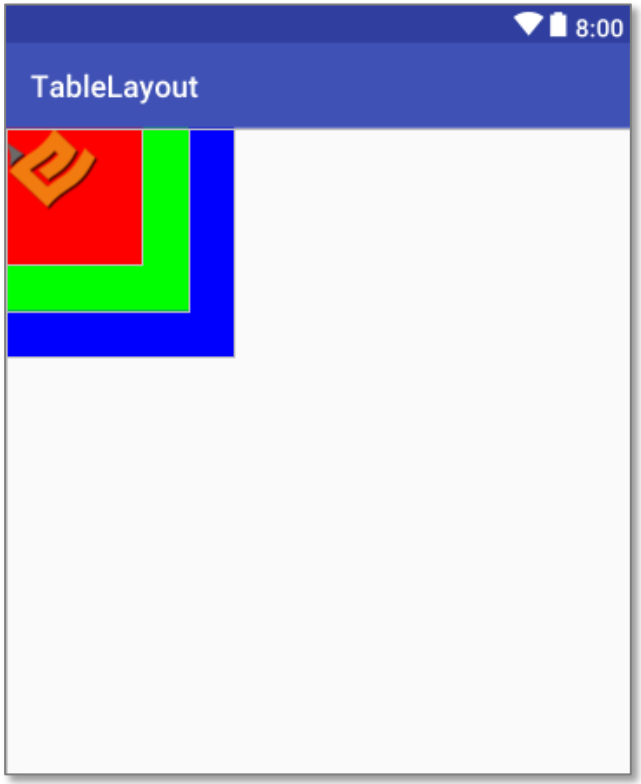
XML文件中布局元素的常用属性

- 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>
```