# 布局管理 (xml)

|  |  |
| --- | --- |
| 约束布局 | 通过相对定位排列 |
| 线性布局 | 以水平或垂直方向排列 |
| 帧布局 | 帧里的控件(层)叠加 |
| 表格布局 | 表格形式排列 |

## 线性布局（LinearLayout）

主要以水平或垂直方式来显示界面中的控件。 当控件水平排列时，显示顺序依次为从左到右，当控件垂直排列时，显示 顺序依次为从上到下。线性布局中，每行或每列中只允许有一个子视图或 控件。

android:gravity： 设置内部控件的显示位置。 

android:orientation：

设置内部控件的排列方向，常量horizontal （默认值）表示水平排列，vertical表示垂直排列。 

android:layout\_weight：设置内部控件在LinearLayout中所占的权重。

## 约束布局（ConstraintLayout）

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。从 Android Studio 2.3 起，官方的模板默认使用 ConstraintLayout。

相对定位：

app:layout\_constraintLeft\_toRightOf="@+id/TextView1"

把TextView2的左边约束到TextView1的右边

layout\_constraintLeft\_toLeftOf 把TextView2的左边约束到TextView1的左边  
layout\_constraintRight\_toLeftOf 把TextView2的右边约束到TextView1的左边  
layout\_constraintRight\_toRightOf 把TextView2的右边约束到TextView1的右边  
layout\_constraintTop\_toTopOf 把TextView2的顶部约束到TextView1的顶部  
layout\_constraintTop\_toBottomOf 把TextView2的顶部约束到TextView1的底部  
layout\_constraintBottom\_toTopOf 把TextView2的底部约束到TextView1的顶部  
layout\_constraintBottom\_toBottomOf 把TextView2的底部约束到TextView1的底部  
layout\_constraintBaseline\_toBaselineOf 文本下划线对齐  
layout\_constraintStart\_toEndOf 把TextView2的起点约束到TextView1的终点  
layout\_constraintStart\_toStartOf 把TextView2的起点约束到TextView1的起点  
layout\_constraintEnd\_toStartOf 把TextView2的终点约束到TextView1的起点  
layout\_constraintEnd\_toEndOf 把TextView2的终点约束到TextView1的终点

角度定位：

app:layout\_constraintCircle="@+id/TextView1" TextView2的中心在TextView1的中心的  
app:layout\_constraintCircleAngle="120"（角度） 120度  
app:layout\_constraintCircleRadius="150dp"（距离） 距离为150dp

边距：

android:layout\_marginStart  
android:layout\_marginEnd  
android:layout\_marginLeft  
android:layout\_marginTop  
android:layout\_marginRight  
android:layout\_marginBottom

## 帧布局(FrameLayout)

默认情况下，控件位于帧布局的左上角。可通过控件的 android:layout\_gravity属性控制其位置。  android:layout\_gravity属性可设置为下列值：

top：控件位于布局顶部。 

bottom：控件位于布局底部，单独使用时等价于“left|bottom”。 

left：控件位于布局左侧。

right：控件位于布局右侧，单独使用时等价于“top|right”。 

center：控件位于布局中心。 

center\_vertical：控件位于垂直方向上的中间位置，单独使用时等价于“left| center\_vertical”。

center\_horizontal：控件位于水平方向上的中间位置，单独使用时等价于“top| center\_ horizontal”。

android:visibility:

View.VISIALBE 可见，显示到页面 

View.INVISIABLE 不可见，但是还是占用位置 

View.GONE 隐藏，不可见并且不占用位置

## 表格布局（TableLayout）

表格布局（TableLayout）是以表格形式排列控件的，通过行和列将界面 划分为多个单元格，每个单元格都可以添加控件。

表格布局需要和TableRow配合使用，每一行都由TableRow对象组成，因此TableRow的数量决定表格的行数。而表格的列数是由包含最多控件的TableRow决定的，例如第1个TableRow有两个控件，第2个TableRow有 三个控件，则表格列数为3。

android:stretchColumns 设置该列被拉伸

android:shrinkColumns 设置该列被收缩

android:collapseColumns 设置该列被隐藏

android:layout\_column 设置该单元显示位置

android:layout\_span 设置该单元格占据几列，默认为1列

# 简单控件 (xml)

## 基本属性

android:id="@+id/textView" //控件ID，可以在java代码中通过findViewById()找到

android:layout\_width="wrap\_content" //控件宽度 单位dp “wrap\_content”代表宽度随内容大小而改变

android:layout\_height="wrap\_content" //控件高度 同上

android:layout\_gravity="center" //内容居中

## TextView(文本框)

 TextView是用于显示文字(字符串)的控件，可在代码中通过设置属性改变 文字的大小、颜色、样式等。

android:text="Hello World!" //显示的文本内容

android:textColor="#D81B60" //文本颜色

android:textSize="26sp" //文本字体大小 单位sp

android:textStyle="nomal" //字体样式 "nomal"无效果 "bold"加粗 "italic"斜体

android:background="" //控件背景颜色

## EditText(输入文本框)

 EditText是可以进行编辑操作的文本框，将用户信息传递给Android程序。 还可以为EditText控件设置监听器，用来测试用户输入的内容是否合法。

android:id="@+id/editText" //控件ID，可以在java代码中通过findViewById()找到

android:layout\_width="wrap\_content" //控件宽度 单位dp “wrap\_content”代表宽度随内容大小而改变

android:layout\_height="wrap\_content" //控件高度 单位dp “wrap\_content”代表高度随内容大小而改变

android:hint="请输入用户名" //框内提示文字

android:ems="10" //每行限制输入十个

tools:layout\_editor\_absoluteX="31dp" //控件的x轴

tools:layout\_editor\_absoluteY="521dp" //控件的y轴

android:inputType=" " //限制输入文本类型 如下表

|  |  |  |  |
| --- | --- | --- | --- |
| none 没有限制 | **textCapCharacters 大写字母** | **textMultiLine多行输入** | **textShortMessage 短讯** |
| Text 文本 | **textCapWords首字母大写** | **textImeMultiLine输入法多行** | **textLongMessage长信息** |
| textUri 网址 | **textCapSentences 仅第一个字母大写** | **textNoSuggestions 不提示** | **textPersonName 人名** |
| textFilte文本筛选 | **textAutoCorrect 文本自动更正** | **textEmailAddress 邮件地址** | **textPostalAddress 地址** |
| Number 数字 | **textAutoComplete 文本自动完成** | **textEmailSubject 邮件主题** | **textPassword 密码** |
| Phone拨号键盘 | **textWebEditText 作为网页表单文本** | **textPhonetic 拼音** | **textVisiblePassword密码可见** |
| Datetime时间日期 | **Date日期** | **Time时间** | **numberSigned带符号数字格式** |
|  |  |  | **numberDecimal带小数点的浮点格式** |

## Button(按钮)

用于响应用户的一系列点击事件，使程序更加流畅和完整。

android:id="@+id/button"

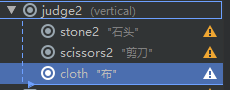
android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Button"

android:onClick="click"/>

## RadioGroup(单选按钮组)和RadioButton(单选按钮)

单选按钮组中包含一个或多个单选按钮

一个单选按钮组只能选中一个单选按钮

通过拖动,可以将按钮收纳至组内

RadioGroup属性:

android:orientation="vertical" //单选按钮对齐方式 horizontal>>水平 vertical>>垂直

## progressBar进度条

style设置进度条样式

style="@style/Widget.AppCompat.ProgressBar.Horizontal" //横向进度条,可以设置是否模糊显示

android:indeterminate="false" //设置模糊搜索true开启false关闭

//模糊显示就是不显示进度,只是显示正在运行 只有横向进度条才可以精确显示

Widget.ProgressBar.Horizontal

//横向进度条（精确模式或模糊模式，这取决于Android:indeterminate）。

Widget.ProgressBar

//中号的圆形进度条（模糊模式）。

Widget.ProgressBar.Small

//小号的圆形进度条（模糊模式）。

Widget.ProgressBar.Large

//大号的圆形进度条（模糊模式）。

Widget.ProgressBar.Inverse

//中号的圆形进度条（模糊模式），该样式适用于亮色背景（例如白色）。

Widget.ProgressBar.Small.Inverse

//小号的圆形进度条（模糊模式），该样式适用于亮色背景（例如白色）。

Widget.ProgressBar.Large.Inverse

//大号的圆形进度条（模糊模式），该样式适用于亮色背景（例如白色）。

android:animationResolution //设置动画间隔时间

android:max //设置进度条最大值

android:progress //设置初始值

android:progressDrawable //设置在精确模式下ProgressBar第一个进度条的Drawable资源。

android:secondaryProgress //设置在精确模式下ProgressBar第二个进度条的Drawable资源。

## SeekBar拖动条

android:max //设置拖动条最大值

## BottomNavigationView 底部导航栏

改变选中后的颜色

新建color类型的xml文件，选择选中的颜色和未选中颜色

<?xml version="1.0" encoding="utf-8"?>

<selector xmlns:android="http://schemas.android.com/apk/res/android">

<!--选择状态下的颜色-->

<item android:color="@color/Select" android:state\_enabled="true" android:state\_checked="true" />

<!--未选择状态下的颜色-->

<item android:color="@color/Unchecked" android:state\_enabled="true" android:state\_checked="false" />

</selector>

在布局中引用

app:itemTextColor="@color/tab\_text"

改变选中后的图标

在drawable文件夹下新建xml，选择选中时图片和未选中图片

<?xml version="1.0" encoding="utf-8"?>

<selector xmlns:android="http://schemas.android.com/apk/res/android">

<!--选择状态下的图片-->

<item android:drawable="@mipmap/examination2" android:state\_checked="true"/>

<!--未选择状态下的图片-->

<item android:drawable="@mipmap/examination1" android:state\_checked="false"/>

</selector>

在布局文件中找到

app:menu="@menu/bottom\_nav\_menu"

打开bottom\_nav\_menu文件

修改对应item的icon路径

# 基础 (Java)

## 声明和引用

private Button button1,button2; //声明需要引用的控件(首字母必须大写)可声明多个

//私有的 控件类型 控件名字 //在类名(class)下创建

button = findViewById(R.id.button); //引用xml中的控件

//控件名字=BfindViewById(R.id. xml中的控件ID)

private Button button=findViewById(R.id.bottom); //可以在页面方法里引用,也可以在创建控件时引用

## 点击事件

button.setOnClickListener(new View.OnClickListener() { //button点击事件

@Override

public void onClick(View v) {

}

## 获取输入

String e1 = editText1.getText().toString(); //使用一个String类型数据接收editText1输入框的数据

//getText 获取文本 toString转换成字符串

## TextView操作

t1.setText(“123”)； //将t1的文本框内容改为 123 值可以是基本类型或引用类型

t1.setVisibility(View.INVISIBLE); //将t1设置为 不可见>>INVISIBLE 可见>>VISIBLE

## ImageView操作

me2.setImageResource(R.drawable.st); //将me2图片改成drawable文件夹下的st图片

## Toast(提示弹窗)

Toast a =Toast.makeText(MainActivity.this,"欢迎您"+zh,Toast.LENGTH\_LONG).show();//创建一个名为a的Toast

//makeText方法生成文本 括号内的值分别为

// (显示的页面.this , 显示的文字(可以使用变量用+号连接) , 显示的时长 LONG长 SHORT短).show显示

## RadioGroup(单选按钮组)和RadioButton(单选按钮)

声明时只需要声明按钮组

judge2.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

@Override

public void onCheckedChanged(RadioGroup radioGroup, int i) {

//选中其中一个单选按钮会返回一个专属的ID i是传入值,也就是选中的按钮ID

}

});//单选按钮的点击事件

RadioButton radioButton = judge2.findViewById(i); //获取了ID之后用这个ID创建方法

String j = (String) radioButton.getText(); //用j接受选中的按钮的文本

点击单选按钮 >> 产生返回值ID >> 创建方法 >> 使用ID找到选中的按钮的文本内容 >> 判断内容执行方法

## ProgressBar(进度条)

setMax(); //设置精确进度条的最大值

setProgress(); //设置进度条的值

getProgress() ; //获取当前进度值

## SeekBar拖动条

大部分方法继承继承于ProgressBar

setProgress(); //设置进度条的值

getProgress() ; //获取当前进度值

setMax(); //设置精确进度条的最大值

SeekBar的监听事件

seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {

@Override

public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {

//拖动改变时

//progress是当前进度值

}

@Override

public void onStartTrackingTouch(SeekBar seekBar) {

//点击时

}

@Override

public void onStopTrackingTouch(SeekBar seekBar) {

//松开时

}

});

# **Inernt意图 (Java)**

显式和隐式

显式通常用于应用内跳转

隐式用于应用间跳转

## 页面跳转

Intent intent = new Intent(MainActivity.this,Game1.class);

// Intent 意图名字= new Intent() 创建一个意图,意图是从this前面的界面跳到class前面的页面

startActivity(intent); //执行意图

finish(); //关闭当前页面,减少程序运行压力

## 数据传递

Intent intent = new Intent(MainActivity.this,Game1.class); //创建意图,指定数据从哪里传到哪里

intent.putExtra("number",123); //意图名字.putExtra(值的名字,值)

//值的名字带双引号,值是引用类型带双引号(字符串)

startActivity(intent); //执行意图

## 数据接收

Intent intent = getIntent(); //接收意图

String e1 = intent.getStringExtra("e1"); //用getStringExtra方法将（值的名字）的值放入e1里

# 动画

Android动画分为：

逐帧动画（Frame Animation）

补间动画（ Tweened animation ）

属性动画（Property Animation）

属性动画填补了逐帧和补间的短板,适用于所有对象,可以自定义各种动画

原理是通过不断赋值改变对象属性,实现平滑显示动画

而属性动画有两个类ValueAnimator 类 & ObjectAnimator 类

## ValueAnimator

通过不断控制值的变化，再不断手动赋给对象的属性，从而实现动画效果.

而ValueAnimator有三个重要方法:

ValueAnimator.ofInt（int values）

// 步骤1：设置动画属性的初始值 & 结束值

ValueAnimator anim = ValueAnimator.ofInt(0, 3);

// ofInt（）作用有两个

// 1. 创建动画实例

// 2. 将传入的多个Int参数进行平滑过渡:此处传入0和1,表示将值从0平滑过渡到1

// 如果传入了3个Int参数 a,b,c ,则是先从a平滑过渡到b,再从b平滑过渡到C，以此类推

// ValueAnimator.ofInt()内置了整型估值器,直接采用默认的.不需要设置，即默认设置了如何从初始值 过渡到 结束值

// 步骤2：设置动画的播放各种属性

anim.setDuration(500); //设置动画运行的时长

anim.setStartDelay(500); // 设置动画延迟播放时间

anim.setRepeatCount(0); // 设置动画重复播放次数 = 重放次数+1

// 动画播放次数 = infinite时,动画无限重复

anim.setRepeatMode(ValueAnimator.REST0ART);

// 设置重复播放动画模式

// ValueAnimator.RESTART(默认):正序重放

// ValueAnimator.REVERSE:倒序回放

// 步骤3：将改变的值手动赋值给对象的属性值：通过动画的更新监听器

// 设置 值的更新监听器

// 即：值每次改变、变化一次,该方法就会被调用一次

anim.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {

@Override

public void onAnimationUpdate(ValueAnimator animation) {

int currentValue = (Integer) animation.getAnimatedValue();

// 获得改变后的值

System.out.println(currentValue);

// 输出改变后的值

// 步骤4：将改变后的值赋给对象的属性值，下面会详细说明

View.setproperty（currentValue）；

// 步骤5：刷新视图，即重新绘制，从而实现动画效果

View.requestLayout();

}

});

anim.start();

// 启动动画

}

总结方法

setDuration(long duration) //设置动画时间

start() //启动动画

setStartDelay(long startDelay) //延迟启动

pause() //暂停

resume() //继续

end() //终止并直接到终点

cancel() //终止不到终点,直接停下

改变控件属性方法

text.setTextColor(color); //文字颜色

text.setTranslationY(Y); //文字Y轴位置,用于上下移动

ValueAnimator.ofFloat（float values）

同上，估值器的数据类型不同

ValueAnimator.ofObject（int values）

同上，估值器的数据类型不同

ValueAnimator.ofArgb(int color)

改变颜色的动画事件

alueAnimator maincolor1=ValueAnimator.ofArgb(color1,color2,color3,color4,color5);

//创建动画,从color1过度到color5

maincolor1.setDuration(5000);

//时间5000ms

maincolor1.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {

@Override

public void onAnimationUpdate(ValueAnimator animation) {

//获取当前颜色值

int color= (int) animation.getAnimatedValue();

//设置文字颜色

text.setTextColor(color);

}

});

//开始动画

maincolor1.start();

# Adapter(适配器)

## 适配器点击事件

先让适配器实现点击事件接口

implements View.OnClickListener

然后在适配器中添加代码

private ListAdapter.OnItemClickListener mOnItemClickListener;

public interface OnItemClickListener {

void onItemClick(View view, int position);

}

public void setOnItemClickListener(ListAdapter.OnItemClickListener listener) {

this.mOnItemClickListener = listener;

}

@Override

public void onClick(View v) {

if (mOnItemClickListener != null) {

//注意这里使用getTag方法获取position

mOnItemClickListener.onItemClick(v, (int) v.getTag());

}

}

在onCreateViewHolder方法中开启监听

每一项的布局.setOnClickListener(this);

在onBindViewHolder（绑定控件）中绑定Tag

holder.itemView.setTag(position);

最后设置点击事件，根据传来的position进行判断又或者传值

该代码要在recyclerView设置适配器后，再进行修改

适配器.setOnItemClickListener(new 适配器.OnItemClickListener() {

@Override

public void onItemClick(View view, int position) {

switch ((String) 数组.get(position)) {

case "考试大纲":

openOutline();

break;

case "作文模板":

openTemplate();

break;

case "音标教程":

openCourse();

break;

case "翻译":

openTranslate();

break;

}

}

});