带你了解 Android 约束布局 ConstraintLayout

ConstraintLayout是Android新推出的一个布局，其性能更好，连官方的hello world都用ConstraintLayout来写了。所以极力推荐使用ConstraintLayout来编写布局。

本文主要介绍一下如何使用代码来编写ConstraintLayout布局。

好了，开始我们的征程。

### ConstraintLayout简介

ConstraintLayout，可以翻译为约束布局，在2016年Google I/O 大会上发布。我们知道，当布局嵌套过多时会出现一些性能问题。之前我们可以去通过RelativeLayout或者GridLayout来减少这种布局嵌套的问题。现在，我们可以改用ConstraintLayout来减少布局的层级结构。ConstraintLayout相比RelativeLayout，其性能更好，也更容易使用，结合Android Studio的布局编辑器可以实现拖拽控件来编写布局等等。

如果我们是新建工程，则Android Studio会默认帮我们加入ConstraintLayout的依赖了。如果我们是改造旧项目，可以在build.gradle中添加以下依赖：

**implementation 'com.android.support.constraint:constraint-layout:1.1.2'**

然后就可以使用ConstraintLayout了。

### 相对位置

要在ConstraintLayout中确定view的位置,必须至少添加一个水平和垂直的约束。每一个约束表示到另一个view，父布局，或者不可见的参考线的连接或者对齐。如果水平或者垂直方向上没有约束，那么其位置就是0。

我们先来看个例子：

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

**<android.support.constraint.ConstraintLayout**

**xmlns:android="http://schemas.android.com/apk/res/android"**

**xmlns:app="http://schemas.android.com/apk/res-auto"**

**android:layout\_width="match\_parent"**

**android:layout\_height="match\_parent">**

**<Button**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:text="左对齐"**

**app:layout\_constraintLeft\_toLeftOf="parent"**

**app:layout\_constraintTop\_toTopOf="parent"/>**

**<Button**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:text="右对齐"**

**app:layout\_constraintRight\_toRightOf="parent"**

**app:layout\_constraintTop\_toTopOf="parent"/>**

**<Button**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:text="水平居中"**

**app:layout\_constraintLeft\_toLeftOf="parent"**

**app:layout\_constraintRight\_toRightOf="parent"/>**

**<Button**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:text="垂直居中"**

**app:layout\_constraintBottom\_toBottomOf="parent"**

**app:layout\_constraintTop\_toTopOf="parent"/>**

**<Button**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:text="底部对齐"**

**app:layout\_constraintBottom\_toBottomOf="parent"**

**app:layout\_constraintLeft\_toLeftOf="parent"/>**

**<Button**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:text="水平居中+垂直居中"**

**app:layout\_constraintBottom\_toBottomOf="parent"**

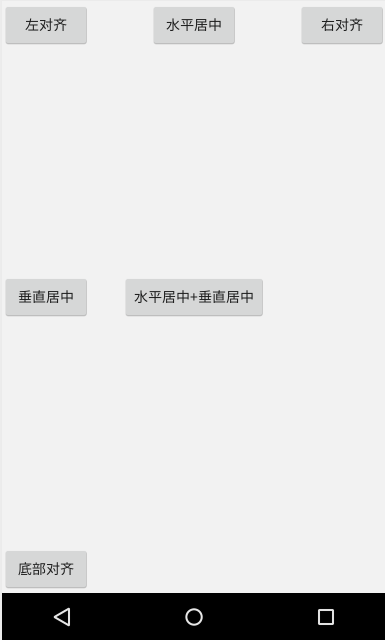
**app:layout\_constraintLeft\_toLeftOf="parent"**

**app:layout\_constraintRight\_toRightOf="parent"**

**app:layout\_constraintTop\_toTopOf="parent"/>**

**</android.support.constraint.ConstraintLayout>**

其显示效果如下图所示：

****

**相对位置.png**

上面例子中app:layout\_constraintLeft\_toLeftOf="parent" 表示view的左边对齐父布局的左边。

app:layout\_constraintRight\_toRightOf="parent"则表示view的右边对齐父布局的右边。

其他同理，就不一一说明。

**相对位置的属性**  
下面是ConstraintLayout确定位置的属性：

**layout\_constraintLeft\_toLeftOf**

**layout\_constraintLeft\_toRightOf**

**layout\_constraintRight\_toLeftOf**

**layout\_constraintRight\_toRightOf**

**layout\_constraintTop\_toTopOf**

**layout\_constraintTop\_toBottomOf**

**layout\_constraintBottom\_toTopOf**

**layout\_constraintBottom\_toBottomOf**

**layout\_constraintBaseline\_toBaselineOf**

**layout\_constraintStart\_toEndOf**

**layout\_constraintStart\_toStartOf**

**layout\_constraintEnd\_toStartOf**

**layout\_constraintEnd\_toEndOf**

这些属性的值即可以是parent，也可以是某个view的id。

**居中显示**  
如果一个view满足以下

**app:layout\_constraintLeft\_toLeftOf="parent"**

**app:layout\_constraintRight\_toRightOf="parent"**

即view的左边对齐父布局的左边，view的右边对齐父布局的右边，除非这个view的大小刚好充满整个父布局；否则的话，就是水平居中显示了。我们可以理解为有两个力，它们左右互搏，view只能给扯到中间了。

再来两个例子，把上面的属性基本都涉及到了，看下估计就懂了，就不逐一说明了。

* 例子一：水平方向的相对位置  
  这里主要关注水平方式的属性即可。  
  布局代码：

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

**<android.support.constraint.ConstraintLayout**

**xmlns:android="http://schemas.android.com/apk/res/android"**

**xmlns:app="http://schemas.android.com/apk/res-auto"**

**xmlns:tools="http://schemas.android.com/tools"**

**android:layout\_width="match\_parent"**

**android:layout\_height="match\_parent">**

**<Button**

**android:id="@+id/btn\_center"**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:background="#00f"**

**android:text="水平参照物"**

**app:layout\_constraintBottom\_toBottomOf="parent"**

**app:layout\_constraintLeft\_toLeftOf="parent"**

**app:layout\_constraintRight\_toRightOf="parent"**

**app:layout\_constraintTop\_toTopOf="parent"/>**

**<Button**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:background="#f00"**

**android:text="Left\_toLeftOf"**

**app:layout\_constraintBottom\_toTopOf="@id/btn\_center"**

**app:layout\_constraintLeft\_toLeftOf="@id/btn\_center"/>**

**<Button**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:background="#0f0"**

**android:text="Right\_toLeftOf"**

**app:layout\_constraintBottom\_toTopOf="@id/btn\_center"**

**app:layout\_constraintRight\_toLeftOf="@id/btn\_center"/>**

**<Button**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:background="#0f0"**

**android:text="Right\_toRightOf"**

**app:layout\_constraintRight\_toRightOf="@id/btn\_center"**

**app:layout\_constraintTop\_toBottomOf="@id/btn\_center"/>**

**<Button**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:background="#f00"**

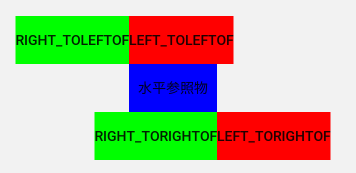
**android:text="Left\_toRightOf"**

**app:layout\_constraintLeft\_toRightOf="@id/btn\_center"**

**app:layout\_constraintTop\_toBottomOf="@id/btn\_center"/>**

**</android.support.constraint.ConstraintLayout>**

显示效果如下：

****

**水平方向的相对位置.png**

* 例子二：竖直方向的相对位置  
  这里主要关注竖直方向的属性即可。  
  布局代码：

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

**<android.support.constraint.ConstraintLayout**

**xmlns:android="http://schemas.android.com/apk/res/android"**

**xmlns:app="http://schemas.android.com/apk/res-auto"**

**xmlns:tools="http://schemas.android.com/tools"**

**android:layout\_width="match\_parent"**

**android:layout\_height="match\_parent">**

**<Button**

**android:id="@+id/btn\_center"**

**android:layout\_width="wrap\_content"**

**android:layout\_height="100dp"**

**android:background="#00f"**

**android:text="竖直参照物"**

**app:layout\_constraintBottom\_toBottomOf="parent"**

**app:layout\_constraintLeft\_toLeftOf="parent"**

**app:layout\_constraintRight\_toRightOf="parent"**

**app:layout\_constraintTop\_toTopOf="parent"/>**

**<Button**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:background="#f00"**

**android:text="Top\_toTopOf"**

**app:layout\_constraintTop\_toTopOf="@id/btn\_center"**

**app:layout\_constraintRight\_toLeftOf="@id/btn\_center"/>**

**<Button**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:background="#0f0"**

**android:text="Bottom\_toTopOf"**

**app:layout\_constraintBottom\_toTopOf="@id/btn\_center"**

**app:layout\_constraintRight\_toLeftOf="@id/btn\_center"/>**

**<Button**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:background="#0f0"**

**android:text="Top\_toBottomOf"**

**app:layout\_constraintLeft\_toRightOf="@id/btn\_center"**

**app:layout\_constraintTop\_toBottomOf="@id/btn\_center"/>**

**<Button**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:background="#f00"**

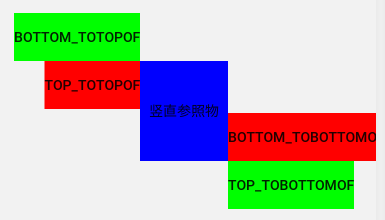
**android:text="Bottom\_toBottomOf"**

**app:layout\_constraintLeft\_toRightOf="@id/btn\_center"**

**app:layout\_constraintBottom\_toBottomOf="@id/btn\_center"/>**

**</android.support.constraint.ConstraintLayout>**

显示效果如下：

****

**竖直方向的相对位置.png**

### 尺寸约束

view中使用warp\_content或者固定值等等是没有问题的。但是ConstraintLayout中不支持MATCH\_PARENT这个值，如果需要实现跟MATCH\_PARENT同样的效果，可以使用0dp来代替，其表示MATCH\_CONSTRAINT,即适应约束。其跟MATCH\_PARENT还是有区别的，后面的例子（宽高比那一小节）会提到。  
我们来看下例子：

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

**<android.support.constraint.ConstraintLayout**

**xmlns:android="http://schemas.android.com/apk/res/android"**

**xmlns:app="http://schemas.android.com/apk/res-auto"**

**android:layout\_width="match\_parent"**

**android:layout\_height="match\_parent">**

**<Button**

**android:id="@+id/btn\_center"**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:text="wrap\_content"**

**app:layout\_constraintLeft\_toLeftOf="parent"**

**app:layout\_constraintRight\_toRightOf="parent"**

**app:layout\_constraintTop\_toTopOf="parent"/>**

**<Button**

**android:id="@+id/btn\_1"**

**android:layout\_width="200dp"**

**android:layout\_height="wrap\_content"**

**android:text="具体数值：200dp"**

**app:layout\_constraintLeft\_toLeftOf="parent"**

**app:layout\_constraintRight\_toRightOf="parent"**

**app:layout\_constraintTop\_toBottomOf="@id/btn\_center"/>**

**<Button**

**android:layout\_width="0dp"**

**android:layout\_height="wrap\_content"**

**android:text="0dp(MATCH\_CONSTRAINT)"**

**app:layout\_constraintLeft\_toLeftOf="parent"**

**app:layout\_constraintRight\_toRightOf="parent"**

**app:layout\_constraintTop\_toBottomOf="@id/btn\_1"/>**

**</android.support.constraint.ConstraintLayout>**

其效果如下：

****

**尺寸大小.png**

**宽高比**

在ConstraintLayout中，还可以将宽定义成高的一个比例或者高定义成宽的比率。首先，需要将宽或者高设置为0dp（即MATCH\_CONSTRAINT），即要适应约束条件。然后通过layout\_constraintDimensionRatio属性设置一个比率即可。这个比率可以是一个浮点数，表示宽度和高度之间的比率；也可以是“宽度：高度”形式的比率。比如：

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

**<android.support.constraint.ConstraintLayout**

**xmlns:android="http://schemas.android.com/apk/res/android"**

**xmlns:app="http://schemas.android.com/apk/res-auto"**

**android:layout\_width="match\_parent"**

**android:layout\_height="match\_parent">**

**<Button**

**android:layout\_width="wrap\_content"**

**android:layout\_height="0dp"**

**android:text="-------------------宽高比2：1-------------------"**

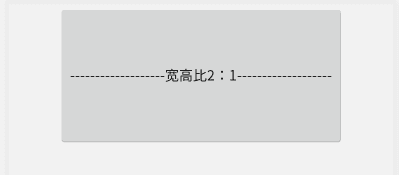
**app:layout\_constraintDimensionRatio="2:1"**

**app:layout\_constraintLeft\_toLeftOf="parent"**

**app:layout\_constraintRight\_toRightOf="parent"/>**

**</android.support.constraint.ConstraintLayout>**

这里将按钮的高度设置为宽度的一半了。如下图所示：

****

**宽高比1.png**

如果宽和高都设置为0dp（MATCH\_CONSTRAINT），那么layout\_constraintDimensionRatio的值需要先加一个"W,"或者"H,"来表示约束宽度或高度。如下：

**<Button**

**android:layout\_width="0dp"**

**android:layout\_height="0dp"**

**app:layout\_constraintBottom\_toBottomOf="parent"**

**app:layout\_constraintDimensionRatio="H,16:9"**

**app:layout\_constraintLeft\_toLeftOf="parent"**

**app:layout\_constraintRight\_toRightOf="parent"**

**app:layout\_constraintTop\_toTopOf="parent"/>**

这里例子是说，首先宽度将满足父布局的约束，然后将按照16：9的比例设置高度。

### 百分比宽高

ConstraintLayout还能使用百分比来设置view的宽高。  
要使用百分比，宽或高同样要设置为0dp（MATCH\_CONSTRAINT）。  
然后设置以下属性即可：

**app:layout\_constraintWidth\_default="percent" //设置宽为百分比**

**app:layout\_constraintWidth\_percent="0.3" //0到1之间的值**

**或**

**app:layout\_constraintHeight\_default="percent" //设置高为百分比**

**app:layout\_constraintHeight\_percent="0.3" //0到1之间的值**

例子如下：

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

**<android.support.constraint.ConstraintLayout**

**xmlns:android="http://schemas.android.com/apk/res/android"**

**xmlns:app="http://schemas.android.com/apk/res-auto"**

**android:layout\_width="match\_parent"**

**android:layout\_height="match\_parent">**

**<Button**

**android:layout\_width="0dp"**

**android:layout\_height="wrap\_content"**

**android:text="宽50%"**

**app:layout\_constraintHeight\_default="percent"**

**app:layout\_constraintLeft\_toLeftOf="parent"**

**app:layout\_constraintWidth\_percent="0.5"/>**

**</android.support.constraint.ConstraintLayout>**

效果如下：

****

**百分比宽高.png**

### 位置偏向

如果想让view的位置偏向某一侧，可以使用以下的两个属性来设置：

**layout\_constraintHorizontal\_bias //水平偏向**

**layout\_constraintVertical\_bias //竖直偏向**

其值同样也是0到1之间。  
比如，以下例子为横向偏向左侧30%，默认的居中效果就是50％：

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

**<android.support.constraint.ConstraintLayout**

**xmlns:android="http://schemas.android.com/apk/res/android"**

**xmlns:app="http://schemas.android.com/apk/res-auto"**

**android:layout\_width="match\_parent"**

**android:layout\_height="match\_parent">**

**<Button**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:text="左边偏向30%"**

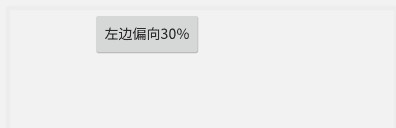
**app:layout\_constraintHorizontal\_bias="0.3"**

**app:layout\_constraintLeft\_toLeftOf="parent"**

**app:layout\_constraintRight\_toRightOf="parent"/>**

**</android.support.constraint.ConstraintLayout>**

显示效果如下：

****

**左边偏向.png**

### 权重

LinearLayout中可以设置权重，ConstraintLayout同样也有这玩意。  
通过设置以下两个属性：

**app:layout\_constraintHorizontal\_weight //水平权重**

**app:layout\_constraintVertical\_weight //竖直权重**

然后将相连的view两两约束好即可。如下：

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

**<android.support.constraint.ConstraintLayout**

**xmlns:android="http://schemas.android.com/apk/res/android"**

**xmlns:app="http://schemas.android.com/apk/res-auto"**

**android:layout\_width="match\_parent"**

**android:layout\_height="match\_parent">**

**<Button**

**android:id="@+id/btn\_1"**

**android:layout\_width="0dp"**

**android:layout\_height="wrap\_content"**

**android:text="权重为1"**

**app:layout\_constraintHorizontal\_weight="1"**

**app:layout\_constraintLeft\_toLeftOf="parent"**

**app:layout\_constraintRight\_toLeftOf="@id/btn\_2"/>**

**<Button**

**android:id="@+id/btn\_2"**

**android:layout\_width="0dp"**

**android:layout\_height="wrap\_content"**

**android:text="权重为2"**

**app:layout\_constraintHorizontal\_weight="2"**

**app:layout\_constraintLeft\_toRightOf="@id/btn\_1"**

**app:layout\_constraintRight\_toLeftOf="@id/btn\_3"/>**

**<Button**

**android:id="@+id/btn\_3"**

**android:layout\_width="0dp"**

**android:layout\_height="wrap\_content"**

**app:layout\_constraintHorizontal\_weight="2"**

**android:text="权重为2"**

**app:layout\_constraintLeft\_toRightOf="@id/btn\_2"**

**app:layout\_constraintRight\_toRightOf="parent"/>**

**</android.support.constraint.ConstraintLayout>**

显示效果如下：

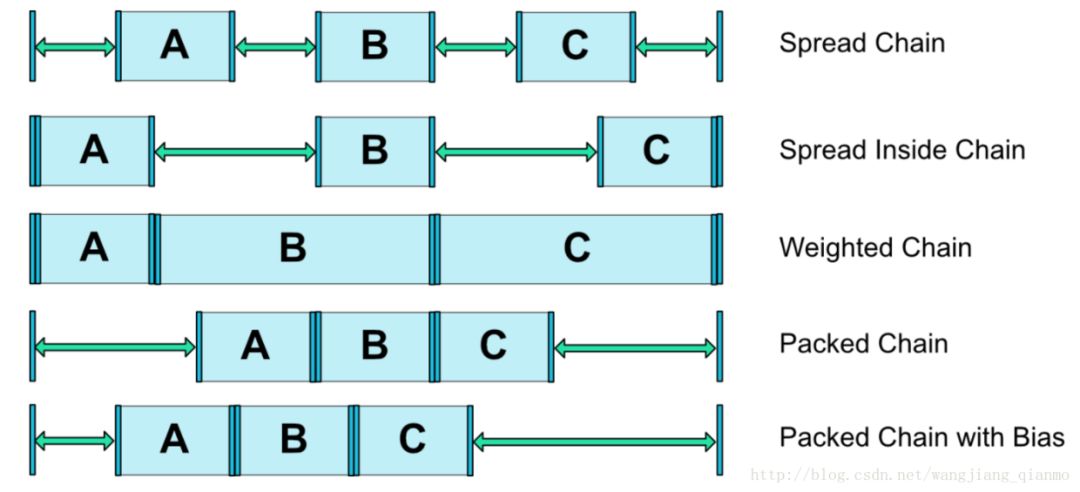


权重.png

### 链

上面这种将相连的view两两约束好的实际上就形成了链。在ConstraintLayout中可以实现各种不同的链，权重链是其中一种。整个链由链中的第一个view（链头）上设置的属性控制。

官网上一共有5种样式的链：

****

**链样式.png**

可以通过以下属性来设置链的样式：

**app:layout\_constraintHorizontal\_chainStyle="spread|spread\_inside|packed"**

下面简单说下上面5种样式的实现：

1. Spread Chain  
   默认样式就是spread，所以可以不用设置。然后宽或高非0即可。  
   效果如下图：



Spread Chain.png

所以：Spread Chain = spread + 宽或高非0

1. Spread Inside Chain  
   这里设置以下属性

**app:layout\_constraintHorizontal\_chainStyle="spread\_inside"**

然后宽度同样为非0，效果如下：



**spread\_inside.png**

所以：Spread Inside Chain = spread\_inside + 宽或高非0

1. Weighter Chain  
   即权重链，具体可以查看上一小节。  
   Weighter Chain = spread + 宽或高为0 + 权重值
2. Packed Chain  
   Packed是一种view聚拢起来的效果，这里设置以下属性

**app:layout\_constraintHorizontal\_chainStyle="packed"**

宽度同样为非0，效果如下：

packed.png  
所以：Packed Chain = packed + 宽或高非0

1. Packed Chain with bias  
   就是Packed Chain再加一个偏向属性，偏向属性可以看下前面的内容。  
   效果如下图所示：



**packed+bias.png**

所以：Packed Chain with bias = Packed Chain + bias

### Guideline辅助线

Guideline可以用来辅助布局，通过Guideline能创建出一条条的水平线或者垂直线，该线不会显示到界面上，但是能够利用这些线条来添加约束去完成界面的布局。  
Guideline主要的属性有：

**android:orientation="horizontal|vertical"**

**app:layout\_constraintGuide\_begin="30dp"**

**app:layout\_constraintGuide\_end="30dp"**

**app:layout\_constraintGuide\_percent="0.5"**

android:orientation=”horizontal|vertical”表示是水平或垂直引导线。

app:layout\_constraintGuide\_begin=”30dp”，如果是水平引导线，则距离布局顶部30dp，如果是垂直引导线，则距离布局左边30dp。

app:layout\_constraintGuide\_end=”30dp”，如果是水平引导线，则距离布局底部30dp，如果是垂直引导线，则距离布局右边30dp。

app:layout\_constraintGuide\_percent=”0.5”，如果是水平引导线，则距离布局顶部为整个布局高度的50%，如果是垂直引导线，则距离布局左边文这个布局宽度的50%。

来看个例子：

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

**<android.support.constraint.ConstraintLayout**

**xmlns:android="http://schemas.android.com/apk/res/android"**

**xmlns:app="http://schemas.android.com/apk/res-auto"**

**android:layout\_width="match\_parent"**

**android:layout\_height="match\_parent">**

**<android.support.constraint.Guideline**

**android:id="@+id/guideline\_h"**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:orientation="horizontal"**

**app:layout\_constraintGuide\_percent="0.5"/>**

**<android.support.constraint.Guideline**

**android:id="@+id/guideline\_v"**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:orientation="vertical"**

**app:layout\_constraintGuide\_percent="0.5"/>**

**<Button**

**app:layout\_constraintLeft\_toLeftOf="@id/guideline\_v"**

**app:layout\_constraintTop\_toTopOf="@id/guideline\_h"**

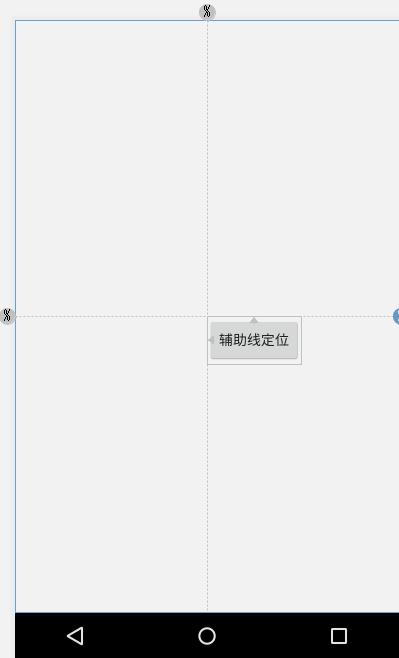
**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:text="辅助线定位"/>**

**</android.support.constraint.ConstraintLayout>**

如下图所示：

****

**辅助线定位.png**

本节基本把ConstraintLayout的常用属性都介绍了一遍了。通过使用ConstraintLayout能够减少布局嵌套等，可以有效的提升性能。