ConstraintLayout是Android Studio 2.2中主要的新增功能之一，也是Google在去年的I/O大会上重点宣传的一个功能。我们都知道，在传统的Android开发当中，界面基本都是靠编写XML代码完成的，虽然Android Studio也支持可视化的方式来编写界面，但是操作起来并不方便，我也一直都不推荐使用可视化的方式来编写Android应用程序的界面。

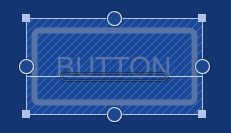
而ConstraintLayout就是为了解决这一现状而出现的。它和传统编写界面的方式恰恰相反，ConstraintLayout非常适合使用可视化的方式来编写界面，但并不太适合使用XML的方式来进行编写。当然，可视化操作的背后仍然还是使用的XML代码来实现的，只不过这些代码是由Android Studio根据我们的操作自动生成的。

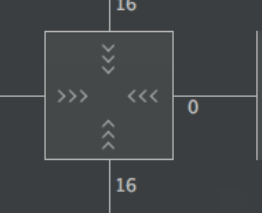
另外，ConstraintLayout还有一个优点，它可以有效地解决布局嵌套过多的问题。我们平时编写界面，复杂的布局总会伴随着多层的嵌套，而嵌套越多，程序的性能也就越差。ConstraintLayout则是使用约束的方式来指定各个控件的位置和关系的，它有点类似于RelativeLayout，但远比RelativeLayout要更强大。

ConstraintLayout的基本用法很简单，比如我们想要向布局中添加一个按钮，那么只需要从左侧的Palette区域拖一个Button进去就可以了。

虽说现在Button已经添加到界面上了，但是由于我们还没有给Button添加任何的约束，因此Button并不知道自己应该出现在什么位置。现在我们在预览界面上看到的Button位置并不是它最终运行后的实际位置，如果一个控件没有添加任何约束的话，它在运行之后会自动位于界面的左上角。

每个控件的约束都分为垂直和水平两类，一共可以在四个方向上给控件添加约束，如下图所示。

  
上图中Button的上下左右各有一个圆圈，这圆圈就是用来添加约束的，我们可以将约束添加到ConstraintLayout，也可以将约束添加到另一个控件。

位于Inspector最中间的那个正方形区域，它是用来控制控件大小的。一共有三种模式可选，每种模式都使用了一种不同的符号表示，点击符号即可进行切换。

* http://img.blog.csdn.net/20170131170403918 表示wrap content，这个我们很熟悉了，不需要进行什么解释。
* http://img.blog.csdn.net/20170131173104320 表示固定值，也就是给控件指定了一个固定的长度或者宽度值。
* http://img.blog.csdn.net/20170131173220664 表示any size，它有点类似于match parent，但和match parent并不一样，是属于ConstraintLayout中特有的一种大小控制方式，下面我们来重点讲解一下。

那有的朋友可能会问了，any size和match parent有什么区别呢？其实最大的区别在于，match parent是用于填充满当前控件的父布局，而any size是用于填充满当前控件的约束规则。举个例子更好理解，如果我们有一个新的Button，它的其中一个约束是添加到当前这个Button上的，那么any size的效果也会发生改变，

Guidelines

现在你已经对ConstraintLayout比较熟悉，并且能使用ConstraintLayout来编写一些简单的界面了。不过目前有一个问题可能还比较头疼，刚才我们已经实现了让一个按钮居中对齐的功能，如果我们想让两个按钮共同居中对齐该怎么实现呢？

其实这个需求很常见，比如说在应用的登录界面，都会有一个登录按钮和一个注册按钮，不管它们是水平居中也好还是垂直居中也好，但肯定都是两个按钮共同居中的。

想要实现这个功能，仅仅用我们刚刚学的那些知识是不够的，这需要用到ConstraintLayout中的一个新的功能，Guidelines。

然后我们希望让这两个按钮在水平方向上居中显示，在垂直方向上都距离底部64dp，那么就需要先添加一个垂直方向上的Guideline，如下图所示。

我来对上图中的操作进行一下解释。首先点击通知栏中的Guidelines图标可以添加一个垂直或水平方向上的Guideline，这里我们需要的是垂直方向上的。而Guideline默认是使用的dp尺，我们需要选中Guideline，并点击一下最上面的箭头图标将它改成百分比尺，然后将垂直方向上的Guideline调整到50%的位置，这样就将准备工作做好了。

接下来我们开始实现让两个按钮在水平方向上居中显示，并距离底部64dp的功能，

自动添加约束

不过如果界面中的内容变得复杂起来，给每个控件一个个地添加约束也是一件很繁琐的事情。为此，ConstraintLayout中支持自动添加约束的功能，可以极大程度上简化那些繁琐的操作。

自动添加约束的方式主要有两种，一种叫Autoconnect，一种叫Inference，我们先来看第一种。

想要使用Autoconnect，首先需要在工具栏中将这个功能启用，默认情况下Autoconnect是不启用的，如下图所示。

http://img.blog.csdn.net/20170131203249059

Autoconnect可以根据我们拖放控件的状态自动判断应该如何添加约束，比如我们将Button放到界面的正中央，那么它的上下左右都会自动地添加上约束，如下图所示。

可以看到，只需要将Button拖放到界面上，Autoconnect会判断我们的意图，并自动给控件添加约束。不过Autoconnect是无法保证百分百准确判断出我们的意图的，如果自动添加的约束并不是你想要的话，还可以在任何时候进行手动修改。总之，可以把它当成一个辅助工具，但不能完全靠它去添加控件的约束。

以上是Autoconnect的用法，接下来我们看一下Inference的用法。Inference也是用于自动添加约束的，但它比Autoconnect的功能要更为强大，因为AutoConnect只能给当前操作的控件自动添加约束，而Inference会给当前界面中的所有元素自动添加约束。因而Inference比较适合用来实现复杂度比较高的界面，它可以一键自动生成所有的约束。

下面我们就通过一个例子来演示一下Inference的用法，比如界面上现在有两个TextView，两个EditText，和两个Button，如下图所示。

接下来我们先将各个控件按照界面设计的位置进行摆放，摆放完成之后点击一下工具栏上的Infer Constraints按钮，就能为所有控件自动添加约束了，如下图所示。