刘坤的技术博客

- BlueBox
- 所有文章

嗨，我是刘坤，一名来自中国的 IOS 开发者，现就职于杭州阿里，花名'念纪'，沉淀技术，寻求创意

GitHub RSS

友情链接

- Casatwy Taloyum
- gf&zj的盗梦空间
- 明弈

Autolayout使用小结（一）

最近做项目时，因为iPhone6和iPhone6Plus的兼容，我们启用了Autolayout. 以前是因为不用也能满足需求，也是因为懒，没有认真使用，只是了解过。经过一段时间的使用，做下总结，希望给大家些帮助哈。

以前我写过IOS自动布局之Autoresizing是关于Autoresizing的介绍，在简单的布局上比较有用，今天总结的是更强大的Autolayout.

本文不是从零开始入门级的介绍，主要介绍了学习过程中遇到一些问题的解决方案，希望给大家帮助，如果初学者请先看看苹果文档，玩玩Autolayout.

PS: Autolayout的强大是毋庸质疑的，当你熟悉了它之后，你肯定会喜欢上它，布局将会比使用frame的绝对坐标时还方便。如果还没有用Autolayout，这已经是最后的时机啦，再不学就out了。

1. 使用途径介绍

还是先介绍下我们可以用哪些途径来使用Autolayout吧,这里只做介绍，具体如何使用，大家可以点击对应链接进入文档学习。

1.1 Nib

最直观的使用就是在xib文件中使用Autolayout. 这种方式最直观，初学者建议先在xib中拖一拖，建立布局的概念。

1.2 Masonry

这是一个第三方的非常好用的开源框架，以前我以为Autolayout只能用xib了，否则代码量太大了，但是Masonry让用代码写Autolayout成为可能，而且非常强大，清晰好用，示例：

```
1 UIEdgeInsets padding = UIEdgeInsetsMake(10, 10, 10, 10);
2
3 [view1 mas_makeConstraints:^(MASConstraintMaker *make) {
4     make.top.equalTo(superview.mas_top).with.offset(padding.top);
5     make.left.equalTo(superview.mas_left).with.offset(padding.left);
6     make.bottom.equalTo(superview.mas_bottom).with.offset(-padding.bottom);
7     make.right.equalTo(superview.mas_right).with.offset(-padding.right);
8 }];
```

1.3 Virtual Format Language

这是苹果推出的一种在代码中使用Autolayout的语法，使用特定的string来创建Constraint，如`[button]-[textField]`代表button和textField间隔标准间距（一般是8） Virtual Format Language 个人感觉没有Masonry好理解和使用。

1.4 手动添加Constraint

Autolayout的本质就是通过给view添加约束，系统通过这些约束来计算view的真实frame. 约束就是类`NSLayoutConstraint`的实例。这里牵涉到Constraint的原理，稍作解释下，理解原理对学习总是好的。那么，什么是Constraint呢？ 看下面的公式：

$$y = m \cdot x + b$$

每一个`NSLayoutConstraint`都代表了上面的这个关系式，其中：

x, y 代表view的attribute， attribute可以分为left right top bottom leading trailing width height centerX centerY baseline m 是倍数， 即multiplier b 常数， 即 constant

从这个公式我们可以了解我们在创建Constraint时，设置的值是如何工作的，比如： x代表 view1.width, y代表view2.width, m = 2, b = 0; 这个约束就代表view2的宽是view1的2倍。

从公式中也可以了解到Constraint只能代表线性关闭，不过只是布局貌似没必要更复杂的关系吧。

有没有发现attribute中有了left和right， 但是还有leading和trailing. 用过的同学会发现这两对没啥区别，为什么会有这2种呢，原因是不是世界上所有的文字都是从left到right。 如果文字是从右到左的话，leading就对应right， training就对应left了。

手动创建Constraint的API:

```
1 +(instancetype)constraintWithItem:(id)view1
2                         attribute:(NSLayoutAttribute)attr1
3                         relatedBy:(NSLayoutRelation)relation
4                            toItem:(id)view2
5                         attribute:(NSLayoutAttribute)attr2
6                        multiplier:(CGFloat)multiplier
7                          constant:(CGFloat)c;
```

理解了上面的公式，再看这个API是不是不一样了呢。

- 通过分析，我们决定采用前两种方式来进行Autolayout的开发。所以无论你是不是代码党，都可以玩转Autolayout哦。

2. 那些不用手动添加的约束

2.1 AutoresizingMark

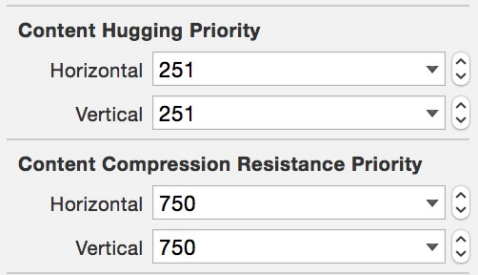
当我们使用Autoresizing时，运行时会自动根据autoresizingMask的特性为view添加Constraint， 我们可以通过下面这个API来决定启用还是关闭这一特性（NO为关闭）：

```
1 - (void)setTranslatesAutoresizingMaskIntoConstraints:(BOOL)flag NS_AVAILABLE_IOS(6_0);
```

这个属性在xib中默认是NO, 在不用xib时默认是YES, 也就是如果使用代码创建Constraint时，需要设置该属性为NO, Masonry自动帮我们做了这一操作，所以用Masonry不需要设置改属性。

2.2 contentHugging & contentCompression

在xib中设置Constraint时，你会看到这块：



在UIView的API中也可以看到这几个方法：

```
1 - (UILayoutPriority)contentHuggingPriorityForAxis:(UILayoutConstraintAxis)axis NS_AVAILABLE_IOS(6_0);
2 - (void)setContentHuggingPriority:(UILayoutPriority)priority forAxis:(UILayoutConstraintAxis)axis NS_AVAILABLE_IOS(6_0);
3 - (UILayoutPriority)contentCompressionResistancePriorityForAxis:(UILayoutConstraintAxis)axis NS_AVAILABLE_IOS(6_0);
4 - (void)setContentCompressionResistancePriority:(UILayoutPriority)priority forAxis:(UILayoutConstraintAxis)axis NS_AVAILABLE_IOS(6_0);
```

那么这两个东西到底是什么呢？可以这样形象的理解一下：

contentHugging: 抱住使其在“内容大小”的基础上不能继续变大 contentCompression: 撑住使其在在其“内容大小”的基础上不能继续变小

这两个属性分别可以设置水平方向和垂直方向上的，而且一个默认优先级是250， 一个默认优先级是750. 因为这两个很有可能与其他Constraint冲突，所以优先级较低。

3. 巧用Constraint优先级

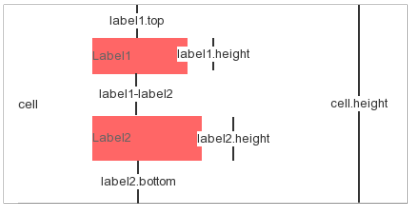
设置Constraint的过程其实就是做算术题的过程，这里提醒大家， xib中有Suggest Constraint的功能，但是大家不要用，特别是在学习Autolayout阶段，Suggest Constraint自动补全的Constraint一般不能直接用，除非你view很少。

如果设置Constraint来确定一个view的frame呢，你可以里面想到： * 1. frame法,像定义frame一样定义约束，就是设置view.left view.top view.width view.height. 但是实际中很少这样用。 * 2. edge法， 设置view.left view.top view.bottom view.right。 * 3. 居中法， 设置 centerX centerY width height.

等等方法，然而事实使用中你可能不是只有一个view，有可能有很多的view，相互直接会有很多的约束，约束一多就会容易发生逻辑冲突，这个时候就会发现Constraint优先级的作用了。

Constraint还有个属性叫priority， 即优先级，一般是0 ~ 1000之间的整数。 1000代表是必需的， 0则不会生效。理解和使用好优先级是熟练使用Autolayout所必备的。

3.1 使用优先级避免冲突的Constraint



举个栗子， `UITableViewCell` 会默认设置 `cell.height` 这个约束， 如果你像上图中设置了垂直方向上的Constraint，在IOS7上是有crash的风险的哦。下面讲原因：

> label1.top + label1.height + (label1-label2) + lable2.height +label2.bottom ≠ cell.height

如果上面不等式发生，那么就会产生Constraint冲突，在IOS8上，苹果应该做了优化，不会crash，只是在控制台会报警告，在IOS7上就会crash了。

啥，你说这些值你都提前算正确就不会出问题了？ 不要太自信，还有一种东西叫“浮点值不准”。 所以最好尽量避免这样的冲突，解决方案也很简单： 选择一个constraint降低优先级，比如把label2.bottom的优先级调到 998. 这样就解决了约束冲突的问题，当约束发生冲突时，低优先级的约束会被舍弃。

3.2 使用优先级决定view的布局

假设在同一水平方向上平行的 Label1 和 Label2。当在水平屏幕宽度不足以展示两个Label的全部内容时，怎么决定哪个Label先展示不全呢，就是通过设置起contentCompression的水平方向上的优先级就可以解决。比如当 Label1.contentCompression < Label2.contentCompression 时， 那么Label1就会展示为 “内容…” 这种。

4. 小结

Autolayout看再多的教程资料不如自己动手一试，所以本文不做详细介绍，旨在分析理解和问题。 （一）就到这吧， 下一篇将会介绍 《自动适应高度Label和Cell的方案》， 《如何在ScrollView中使用Autolayout》， 《Autolayout时的动画》。

Comments