



博客

Loader's Blog

人不会死在绝境，却往往栽在十字路口

目录视图

个人资料



元斌

+ 关注

发私信



访问：430600次

积分：5810

等级：BLOG > 5

排名：第3721名

原创：78篇

转载：0篇

译文：2篇

评论：554条

联系我

邮箱：qibin0506@gmail.comgithub：<https://github.com/qibin0506>codercard：<http://qibin.codercard.net/>

QQ群



一脸懵逼

Android开发交流群

文章分类

android (65)

CoordinatorLayout高级用法-自定义Behavior

标签：behavior

快速回复

我要收藏

2015-12-14 09:40

13855人阅读

评论

分类：android (64)

版权声明：本文来自Loader's Blog，未经博主允许不得转载。

目录(?)

[+]

在新的support design中，CoordinatorLayout可以说是最重要的一个控件了，CoordinatorLayout给我们带来了一种新的事情behavior，你是不是还记得我们在使用CoordinatorLayout的时候，一些子view需要一段，

```
1 app:layout_behavior="@string/appbar_scrolling_view_behavior"
```

这样的xml配置？当时我们不知道这是干嘛的，直接照用就行了，后来发现这玩意是一个类！而且我们还可以自定义！所以，今一下如何自定义Behavior，之后的博客可能会看一下CoordinatorLayout是怎么处理这个Behavior的。

认识Behavior

Behavior是CoordinatorLayout的一个抽象内部类

```
1 public abstract static class Behavior<V extends View> {
2     public Behavior() {
3     }
4
5     public Behavior(Context context, AttributeSet attrs) {
6     }
7     ...
8 }
```

有一个泛型是指定的我们应用这个Behavior的View的类型，例如上面的appbar_scrolling_view_behavior对应的字符串其实是 `Android.support.design.widget.AppBarLayout$ScrollingViewBehavior`，这个 `ScrollingViewBehavior` 内部类指定的泛型是 `View`，所以任何的View都可以使用，我们在自定义的时候，如果不是特殊的行为，也可以直接指定泛型 `View`。

在自定义Behavior的时候，我们需要关心的两组四个方法，为什么分为两组呢？看一下下面两种情况

1. 某个view监听另一个view的状态变化，例如大小、位置、显示状态等
2. 某个view监听CoordinatorLayout里的滑动状态

golang (9)

java (3)

web (1)

杂谈 (2)

博客专栏

设计模式

文章：8篇

阅读：39294

Android新技术

文章：15篇

阅读：158745

Android源码解析

文章：7篇

阅读：36804

打造android ORM框架

文章：7篇

阅读：12192

文章存档

2017年03月 (1)

2016年12月 (1)

2016年11月 (2)

2016年10月 (1)

2016年09月 (2)

展开

友情链接

鸿洋_

Aggie的博客

梁肖技术中心

极客导航

CN22

文章搜索

阅读排行

RecyclerView添加Header的正.. (24112)

CoordinatorLayout高级用法-... (13839)

是时候来了解android7了:sho... (13454)

AndroidSupportDesign之Ta... (13342)

高逼格UI-ASD(Android Supp... (12563)

Android Bottom Sheet详解 (12253)

你所不知道的Activity转场动画... (11838)

RecyclerView的高级用法——... (11451)

Android自定义View—仿雷达... (11126)

打造史上最容易使用的Tab指示... (10259)

评论排行

Android路由实现 (43)

RecyclerView添加Header的正.. (38)

是时候来了解android7了:sho... (36)

对于第一种情况，我们关心的是：

layoutDependsOn 和 onDependentViewChanged 方法，

对于第二种情况，我们关心的是：

onStartNestedScroll 和 onNestedPreScroll 方法。

对于这几个方法什么意思，我们需要干什么，稍候我们就能了解到。

初步自定义

现在我们就来根据第一种情况尝试自定义一个Behavior，这里我们实现一个简单的效果，让一个View根据另一个View上下移动

首先我们来自定义一个Behavior，起名为 DependentBehavior

```
1 public class DependentBehavior extends CoordinatorLayout.Behavior<View> {
2
3     public DependentBehavior(Context context, AttributeSet attrs) {
4         super(context, attrs);
5     }
6
7     @Override
8     public boolean layoutDependsOn(CoordinatorLayout parent, View child, View dependency) {
9         return super.layoutDependsOn(parent, child, dependency);
10    }
11
12    @Override
13    public boolean onDependentViewChanged(CoordinatorLayout parent, View child, View dependency) {
14        ViewCompat.offsetLeftAndRight();
15        return super.onDependentViewChanged(parent, child, dependency);
16    }
17 }
```

注意一下，带有参数的这个构造必须要重载，因为在CoordinatorLayout里利用反射去获取这个Behavior的时候就是拿的这个方法 layoutDependsOn 和 onDependentViewChanged，这两个方法的参数都是一样的，解释一下，第一个不用说，就是当前的CoordinatorLayout，第二个是我们设置这个Behavior的View，第三个是我们关心的那个View。如何知道关心的哪个呢？ layoutDependsOn 的返回值决定了—

这里我们关心一个TextView好了，所以 layoutDependsOn 可以这么写，

```
1 @Override
2 public boolean layoutDependsOn(CoordinatorLayout parent, View child, View dependency) {
3     return dependency instanceof TextView;
4 }
```

现在设置好了关心谁，接下来就是在这个View状态发生变化的时候，我们现在的View该做些什么了，恩，这里肯定是在 onDependentViewChanged 方法里了。我们的任务就是获取dependency距离底部的距离，并且设置给child,很简单。

```
1 @Override
2 public boolean onDependentViewChanged(CoordinatorLayout parent, View child, View dependency) {
3     int offset = dependency.getTop() - child.getTop();
4     ViewCompat.offsetTopAndBottom(child, offset);
5     return true;
6 }
```

首先我们先获取两个View的top值的差，然后让child的位置位移一下就ok啦，如此简单，那这个简单的Behavior如何用呢？

```
1 <android.support.design.widget.CoordinatorLayout
2     xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:fitsSystemWindows="true"
8     tools:context="org.loader.mybehavior.MainActivity">
9
10    <TextView
11        android:id="@+id/dependent"
12        android:layout_width="100dp"
13        android:layout_height="100dp"
14        android:background="#FFFF0000"
15        android:gravity="center"
```

打造史上最容易使用的Tab指示..	(26)
Android Bottom Sheet详解	(22)
Android打造不一样的EmptyV..	(20)
高逼格UI-ASD(Android Supp..	(19)
利用githubpages创建你的个...	(18)
Android自定义Transition动画	(18)
Android官方数据绑定框架Da...	(17)

最新评论

你所不知道的Activity转场动画——Activi...
xie592030956 : 好多文字和给出的例子不一致的。。。

是时候来了解android7了:shortcuts(快捷...
开斌 : @yumi0629:不会，长按会出现shor cut， 拖动的时候才会出现删除。

是时候来了解android7了:shortcuts(快捷...
吉原拉面 : 这个是不是会和桌面图标长按删除有冲突？

RecyclerView添加Header的正确方式
某猿 : 完美解决我的问题 多谢博主！

Android Material Design动画
海峰-清欢 : 赞

RecyclerView添加Header的正确方式
guoshijie1990 : 好用！比网上其他的乱七八糟的简单明了多了。

RecyclerView添加Header的正确方式
dxio cn : 最后封装的MyHolder中，是否少了判断？if(itemView == mHeaderView) ...

Android路由实现
开斌 : @Huang_Cai_Yuan:不想给你回。。。。

Android路由实现
编码很酷 : Router.router(ActivityRule.AC TIVITY_SCHEME + "shop...

Android路由实现
jasoncol_521 : @qibin0506:好的

```
16         android:textColor="@android:color/white"
17         android:layout_gravity="top|left"
18         android:text="dependent"/>
19
20     <TextView
21         android:layout_width="100dp"
22         android:layout_height="100dp"
23         android:background="#FF00FF00"
24         android:gravity="center"
25         android:textColor="@android:color/white"
26         android:layout_gravity="top|right"
27         app:layout_behavior="org.loader.mybehavior.DependentBehavior"
28         android:text="auto"/>
29
30 </android.support.design.widget.CoordinatorLayout>
```

注意，第二个TextView我们设置了 app:layout_behavior="org.loader.mybehavior.DependentBehavior"

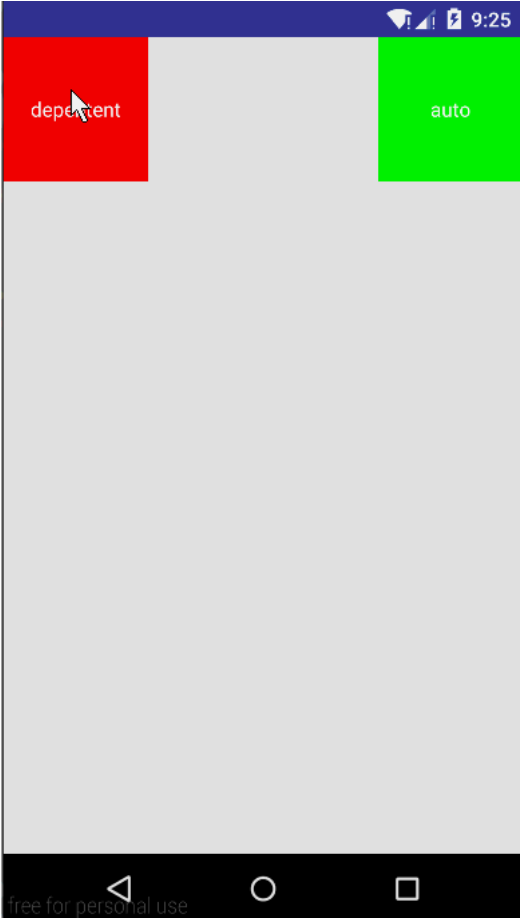
值正好是我们定义的那个 DependentBehavior。

```
1 final TextView dependent = (TextView) findViewById(R.id.dependent);
2 dependent.setOnClickListener(new View.OnClickListener() {
3     @Override
4     public void onClick(View v) {
5         ViewCompat.offsetTopAndBottom(v, 5);
6     }
7 });
```

快速回复

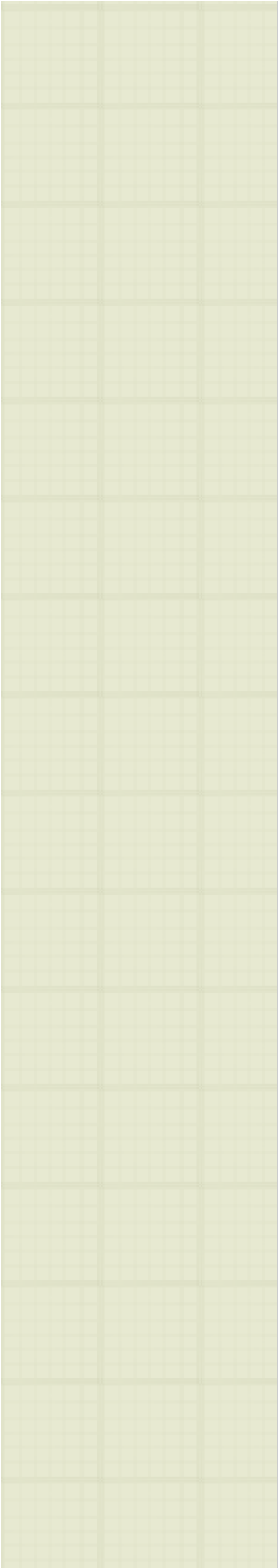
☆ 我要收藏

在Activity中，我们每次点击第一个TextView都会去改变一下它的位置，下面让我们来看看另一个TextView的位置改变了没有。



Scroll Behavior

在学会了如何自定义Behavior后，我们接着来实现上面说的第二种情况-滑动。为了演示这种Behavior的定义，我们还是来做个跟随另一个ScrollView滑动。恩，先来看看效果吧，



快速回复

我要收藏

从效果中我们可以看出，第二个ScrollView明显是在跟随第一个进行滑动，现在让我们用自定义Behavior的形式实现它。
创建一个Behavior，起名叫ScrollBehavior，

```
1
2 public class ScrollBehavior extends CoordinatorLayout.Behavior<View> {
3
4     public ScrollBehavior(Context context, AttributeSet attrs) {
5         super(context, attrs);
6     }
7
8     @Override
9     public boolean onStartNestedScroll(CoordinatorLayout coordinatorLayout, View child, View directTarget, View target, int dx, int dy, boolean isFromUserGesture) {
10         return super.onStartNestedScroll(coordinatorLayout, child, directTarget, target, nestedScrollAxes);
11     }
12
13     @Override
14     public void onNestedPreScroll(CoordinatorLayout coordinatorLayout, View child, View target, int dx, int dy, boolean isFromUserGesture, boolean consumed) {
15         super.onNestedPreScroll(coordinatorLayout, child, target, dx, dy, consumed);
16     }
17
18     @Override
19     public boolean onNestedPreFling(CoordinatorLayout coordinatorLayout, View child, View target, float velocityX, float velocityY) {
20         return super.onNestedPreFling(coordinatorLayout, child, target, velocityX, velocityY);
21     }
22 }
```

和你想的一样，我们覆写了 `onStartNestedScroll` 和 `onNestedPreScroll` 方法，但是除了这两个方法外，我们还覆写了 `onNestedPreFling` 方法。为什么呢？估计大家已经猜出来了，这里是处理fling动作的，你想想，我们在滑动松开手的时候，ScrollView是不是还继续滑动一会，那个ScrollView也要继续滑动一会，这种效果，`onNestedPreFling` 就派上用场了。

好，接下来我们来实现代码，首先来看看 `onStartNestedScroll`，这里的返回值表明这次滑动我们要不要关心，我们要关心什么样的。

```
1 @Override
2 public boolean onStartNestedScroll(CoordinatorLayout coordinatorLayout, View child, View directTarget, View target, int dx, int dy, boolean isFromUserGesture) {
3     return nestedScrollAxes != null && (dx != 0 && nestedScrollAxes.indexOfAxisType(ScrollAxis.HORIZONTAL) != -1 || dy != 0 && nestedScrollAxes.indexOfAxisType(ScrollAxis.VERTICAL) != -1);
4 }
```

```
4         return (nestedScrollAxes & ViewCompat.SCROLL_AXIS_VERTICAL) != 0;
5     }
6 }
```

现在我们准备好了关心的滑动事件了，那如何让它滑动起来呢？还是要看 `onNestedPreScroll` 的实现

```
1 @Override
2 public void onNestedPreScroll(CoordinatorLayout coordinatorLayout, View child, View target, int dx, int
3     super.onNestedPreScroll(coordinatorLayout, child, target, dx, dy, consumed);
4     int leftScrolled = target.getScrollY();
5     child.setScrollY(leftScrolled);
6 }
```

也很简单，让child的scrollY的值等于目标的scrollY的值就ok啦，那fling呢？更简单，

```
1 @Override
2 public boolean onNestedFling(CoordinatorLayout coordinatorLayout, View child, View target, float veloci
3     ((NestedScrollView) child).fling((int) velocityY);
4     return true;
5 }
```

直接将现在的y轴上的速度传递传递给child，让他fling起来就ok了。



定义好了Behavior，就得在xml中使用了，使用方法和前面的一样。

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/andr
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:fitsSystemWindows="true"
7     android:orientation="vertical">
8
9     <android.support.v4.widget.NestedScrollView
10         android:layout_gravity="left"
11         android:layout_width="wrap_content"
12         android:background="#FF00FF00"
13         android:layout_height="match_parent">
14
15         <LinearLayout
16             android:layout_width="wrap_content"
17             android:layout_height="wrap_content"
18             android:orientation="vertical">
19
20             <TextView
21                 android:layout_width="wrap_content"
22                 android:layout_height="wrap_content"
23                 android:paddingTop="50dp"
24                 android:paddingBottom="50dp"
25                 android:textColor="@android:color/white"
26                 android:text="contentLeft"/>
27
28             <TextView
29                 android:layout_width="wrap_content"
30                 android:layout_height="wrap_content"
31                 android:paddingTop="50dp"
32                 android:paddingBottom="50dp"
33                 android:textColor="@android:color/white"
34                 android:text="contentLeft"/>
35
36             <TextView
37                 android:layout_width="wrap_content"
38                 android:layout_height="wrap_content"
39                 android:paddingTop="50dp"
40                 android:paddingBottom="50dp"
41                 android:textColor="@android:color/white"
42                 android:text="contentLeft"/>
43
44             <TextView
45                 android:layout_width="wrap_content"
46                 android:layout_height="wrap_content"
47                 android:paddingTop="50dp"
48                 android:paddingBottom="50dp"
49                 android:textColor="@android:color/white"
50                 android:text="contentLeft"/>
51
52             <TextView
```

快速回复

☆ 我要收藏

```
51         android:layout_width="wrap_content"
52         android:layout_height="wrap_content"
53         android:paddingTop="50dp"
54         android:paddingBottom="50dp"
55         android:textColor="@android:color/white"
56         android:text="contentLeft"/>
57     <TextView
58         android:layout_width="wrap_content"
59         android:layout_height="wrap_content"
60         android:paddingTop="50dp"
61         android:paddingBottom="50dp"
62         android:textColor="@android:color/white"
63         android:text="contentLeft"/>
64
65
66     </LinearLayout>
67
68 </android.support.v4.widget.NestedScrollView>
69
70 <android.support.v4.widget.NestedScrollView
71     android:layout_gravity="right"
72     android:layout_width="wrap_content"
73     android:background="#FFFF0000"
74     android:layout_height="match_parent"
75     app:layout_behavior="org.loader.mybehavior.ScrollBehavior">
76
77     <LinearLayout
78         android:layout_width="wrap_content"
79         android:layout_height="wrap_content"
80         android:orientation="vertical">
81         <TextView
82             android:layout_width="wrap_content"
83             android:layout_height="wrap_content"
84             android:paddingTop="50dp"
85             android:paddingBottom="50dp"
86             android:textColor="@android:color/white"
87             android:text="contentRight"/>
88
89         <TextView
90             android:layout_width="wrap_content"
91             android:layout_height="wrap_content"
92             android:paddingTop="50dp"
93             android:paddingBottom="50dp"
94             android:textColor="@android:color/white"
95             android:text="contentRight"/>
96
97         <TextView
98             android:layout_width="wrap_content"
99             android:layout_height="wrap_content"
100            android:paddingTop="50dp"
101            android:paddingBottom="50dp"
102            android:textColor="@android:color/white"
103            android:text="contentRight"/>
104
105         <TextView
106             android:layout_width="wrap_content"
107             android:layout_height="wrap_content"
108             android:paddingTop="50dp"
109             android:paddingBottom="50dp"
110             android:textColor="@android:color/white"
111             android:text="contentRight"/>
112
113         <TextView
114             android:layout_width="wrap_content"
115             android:layout_height="wrap_content"
116             android:paddingTop="50dp"
117             android:paddingBottom="50dp"
118             android:textColor="@android:color/white"
119             android:text="contentRight"/>
120
121         <TextView
122             android:layout_width="wrap_content"
123             android:layout_height="wrap_content"
124             android:paddingTop="50dp"
125             android:paddingBottom="50dp"
126             android:textColor="@android:color/white"
127             android:text="contentRight"/>
128
129     </LinearLayout>
```

 快速回复 我要收藏

```
129         </android.support.v4.widget.NestedScrollView>
130
131         </android.support.design.widget.CoordinatorLayout>
```

第二个ScrollView的layout_behavior我们指定为 org.loader.mybehavior.ScrollBehavior，现在就可以看到上面的效果了。

ok，最后是文章中demo的代码下载：<http://download.csdn.net/detail/qibin0506/9352989>

顶

15

踩

0

- 上一篇 初探Java8lambda表达式
- 下一篇 源码看CoordinatorLayout.Behavior原理

我的同类文章

android (64)		快速回复
		我要收藏
• 是时候来了了解android7了:通知直接回复	2016-12-26 阅读 4081	• Android路由实现 2016-11-28
• Android自定义Transition动画	2016-11-21 阅读 3287	• 是时候来了了解android7了:shortcuts(快... 2016-10-21
• RecyclerView自定义LayoutManager,...	2016-09-27 阅读 8683	• 是时候来了了解android7了:多窗口支持 2016-08-22
• 来仿一仿retrofit	2016-07-25 阅读 5790	• 打造Material Design风格的TabBar 2016-05-08
• Android Bottom Sheet详解	2016-03-28 阅读 12257	• ubuntu下安装AndroidStudio 2016-02-25
• 源码看CoordinatorLayout.Behavior原...	2015-12-22 阅读 9610	

参考知识库



Android知识库
32835 关注 | 2675 收录



.NET知识库
3627 关注 | 833 收录

猜你在找

- Android开发高级组件与框架——...
- 【Android APP开发】Android高级...
- Android App性能调优、内存泄露...
- Android APP开发之真机调试环境...
- Android5.0新特征详解(Material...
- UWP开发自定义Behavior的使用
- 自定义Behavior之ToolBar上滑Ta...
- Material Design系列自定义Beha...
- 自定义behavior实现UC首页
- 在ASPNETAtlas中创建自定义的Be...

查看评论

- 名字被取了
两个列表滑动的时候可能会错位
11楼 2016-12-26 14:00
- gdut_song
我用CoordinatorLayout、AppBarLayout和CollapsingToolBarLayout，没有使用toolBar，然而App BarLayout外部设置的View滚动时，CollapsingToolBarLayout里的view没有收缩，郁闷了。我试试你的方法
10楼 2016-12-26 13:50
- 岁月0_0静好
发布两小时后被转载 <http://www.07net01.com/2015/12/1017279.html>
9楼 2016-12-26 13:40



litefish

8楼 2017-04-06 14:08

看了很多CoordinatorLayout的文章，这个算是非常好的



upperLucky

7楼 2017-04-06 14:08

博主学习这些知识都是自己看的源码还是参考别人的博客呢



亓斌

Re: 2017-04-06 14:08

回复upperLucky：我都是首先搞会怎么用，然后必要的话去源码看原理。



红灰李

6楼 2017-04-06 14:08

博主 有个问题想请教一下 第一个例子中Behavior中定义了一个View依赖于某一个TextView 如果CoordinatorLayout中有多个TextView 的时候 被设置了Behavior的View要依赖于哪一个



litefish

Re: 2017-04-06 14:08

回复红灰李：都依赖，一个view可以依赖于多个view

快速回复



亓斌

Re: 2017-04-06 14:08

回复红灰李：layoutDependsOn

我要收藏



fromVillageCoolBoy

5楼 2017-04-06 14:08

自己在做折叠效果里的移动toolbar透明效果，请教下，能实现么



fly_androidBoy

4楼 2017-04-06 14:08

我一直不明白 是什么把两个ScrollView联系起来的, 只在一个ScrollView加了 behavior,为什么 他会关联到 第一个ScrollView 而不是关联其他的View



亓斌

Re: 2017-04-06 14:08

回复fly_androidBoy：看下一篇博客



fly_androidBoy

3楼 2017-04-06 14:08

我一直不明白 是什么把两个ScrollView联系起来的, 只在一个ScrollView加了 behavior,为什么 他会关联到 第一个ScrollView 而不是关联其他的View



mzhua78

Re: 2017-04-06 14:08

回复fly_androidBoy：设置了behavior的view在onNestedPreScroll是接收CoordinatorLayout的事件的，CoordinatorLayout下面的实现了NestedScrollChild的可滚动的View滚动了会传递给CoordinatorLayout，然后CoordinatorLayout就分发下去了，所以不需要设置各个子View之间的关系，不知道这样理解对不对



帝丹11

2楼 2017-04-06 14:08

简单易懂，棒



kailaisi

1楼 2017-04-06 14:08

这个有什么一些常见的应用场景么？能给介绍一下么？



亓斌

Re: 2017-04-06 14:08

回复kailaisi：CoordinatorLayout最核心的东西就在Behavior上。

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题

Hadoop

AWS

移动游戏

Java

Android

iOS

Swift

智能硬件

Docker

OpenStack

VP

IE10

Eclipse

CRM

JavaScript

数据库

Ubuntu

NFC

WAP

jQuery

BI

HTML5

Spring

Apac

HTML

SDK

IIS

Fedora

XML

LBS

Unity

Splashtop

UML

components

Windows Mobile

Ra

Cassandra

CloudStack

FTC

coremail

OPhone

CouchBase

云计算

iOS6

Rackspace

Web App

S

Compuware

大数据

apttech

Perl

Tornado

Ruby

Hibernate

ThinkPHP

HBase

Pure

Solr

An

Cloud Foundry

Redis

Scala

Django

Bootstrap

公司简介 | 招贤纳士 | 广告服务 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服

杂志客服

微博客服

webmaster@csdn.net

400-600-2320

北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 | 江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2016, CSDN.NET, All Rights Reserved

 快速回复

 我要收藏

http://blog.csdn.net/qibin0506/article/details/50290421

9/9