**CoordinatorLayout使用（三）：NestedScrollView & 嵌套滑动事件**

**简单复习**

前面大体大体了解了  
Behavior 和 CoordinatorLayout 直接的关系和使用  
自定义Behavior的通用流程  
了解 绑定的方式， 事件流  
上一篇唯一的例子，是事件流中 **CoordinatorLayout关联事件**

而其中， 事件流中，嵌套滑动事件 中  
我们 只是了解了 **NestedScrollingChild** ， **NestedScrollingParent** 的理论关系

注意： **嵌套滑动事件 不需要指定DependOn， 所有NestedScrollingChild的滑动， Parent默认都可以获得**

通过源码，可以提前了解知道

* **CoordinatorLayout 其实是一个 NestedScrollingParent**

CoordinatorLayout

* 传递滑动事件的，其实相当于 NestedScrollingChild
* 最后真正消费事件的，其实是 Behavior的子类（自定义的，系统的）

**NestedScrollView简单了解**

有一个类，叫 NestedScrollView：

NestedScrollView

直接从源码看，可以知道， 它既是一个 **NestedScrollingChild** 也是一个 **NestedScrollingParent**  
换句话说， 即可以 **接收事件**， 也可以 **处理并且发送给Behavior子类绑定的View**

参考下官网：  
[https://developer.android.com/reference/android/support/v4/widget/NestedScrollView.html](https://link.jianshu.com/?t=https://developer.android.com/reference/android/support/v4/widget/NestedScrollView.html)  
可以发现，其实就是一个ScrollView  
并且可以在老版本，新版本的android下面使用。  
默认是开启的。

**嵌套滑动事件 简单实例**

注意： 这里是用 事件流中 嵌套滑动事件 去处理的

我们可以用NestedScrollView做事件发送，给外面的Parent发事件， 再传递给Behavior子类绑定的View  
简单的思路：

* NestedScrollView直接会发送事件
* CoordinatorLayout也就是外面parent的会自动接收
* 我们只需要写一个Behavior子类来消费即可

我们先看一下 嵌套滑动事件 方式的 Behavior

自定义简单的Behavior **DodoBehavior1scroll**

package com.aohuan.dodo.coordinator.utils;

import android.content.Context;

import android.support.design.widget.CoordinatorLayout;

import android.support.v4.view.ViewCompat;

import android.support.v4.widget.NestedScrollView;

import android.util.AttributeSet;

import android.view.View;

/\*\*

\* Created by dodo 2390183798 on 2016/10/31.

\* 参考： http://blog.csdn.net/qibin0506/article/details/50290421

\*

\* 对应的一起滑动的

\* 原理也简单， 是上下滑动， 就设置对应的y值为 Main View的y值

\*

\*/

public class DodoBehavior1scroll extends CoordinatorLayout.Behavior<View> {

public DodoBehavior1scroll(Context context, AttributeSet attrs) {

super(context, attrs);

}

@Override

public boolean onStartNestedScroll(CoordinatorLayout coordinatorLayout, View child, View directTargetChild, View target, int nestedScrollAxes) {

// return (nestedScrollAxes & ViewCompat.SCROLL\_AXIS\_VERTICAL) != 0;

return nestedScrollAxes == ViewCompat.SCROLL\_AXIS\_VERTICAL;

// return false;

}

// @Override

// public void onNestedPreScroll(CoordinatorLayout coordinatorLayout, View child, View target, int dx, int dy, int[] consumed) {

// super.onNestedPreScroll(coordinatorLayout, child, target, dx, dy, consumed);

// int followScrolled = target.getScrollY();

// child.setScrollY(followScrolled);

// }

@Override

public void onNestedScroll(CoordinatorLayout coordinatorLayout, View child, View target, int dxConsumed, int dyConsumed, int dxUnconsumed, int dyUnconsumed) {

super.onNestedScroll(coordinatorLayout, child, target, dxConsumed, dyConsumed, dxUnconsumed, dyUnconsumed);

int followScrolled = target.getScrollY();

child.setScrollY(followScrolled);

}

@Override

public boolean onNestedFling(CoordinatorLayout coordinatorLayout, View child, View target, float velocityX, float velocityY, boolean consumed) {

if(child instanceof NestedScrollView){

((NestedScrollView) child).fling((int)velocityY);

}

return true;

}

}

很好理解， 也就是3个方法  
（具体参数说明，看上一篇Behavior子类获得事件，对应View变化，这里不单独介绍了）

* boolean onStartNestedScroll
  + 判断是否接收后续事件
  + 我们的例子由于是竖直方向的滑动监听（直接true包含横向也行，后面不会获取对应的值）
* void onNestedScroll
  + 对应滑动的时候，处理的事情
  + 当然，这里换成void onNestedPreScroll 效果是差不多的， 具体只是2个方法有先后顺序而已
* boolean onNestedFling
  + 对应的滑动较快，也就是fling事件触发的时候调用
  + 这里不能换成 onNestedPreFling，替换后，会有卡顿，暂时不纠结为什么

这里layout，也很简单  
就CoordinatorLayout中，包含 2个 NestedScrollView ， 一个Behavior

**activity\_main4.xml**

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

<RelativeLayout 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.design.widget.CoordinatorLayout

android:layout\_width="match\_parent"

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

<!--app:layout\_behavior="com.aohuan.dodo.coordinator.utils.DodoBehavior1scroll"-->

<android.support.v4.widget.NestedScrollView

android:id="@+id/ns0"

android:layout\_width="150dp"

android:layout\_height="match\_parent"

android:layout\_gravity="top|right"

android:background="#666666">

<!--<com.aohuan.dodo.coordinator.view.NoScrollListView-->

<!--android:id="@+id/list0"-->

<!--android:layout\_width="150dp"-->

<!--android:layout\_height="match\_parent"-->

<!--android:layout\_gravity="top|right" />-->

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

style="@style/TextAppearance.AppCompat.Display3"

android:text="A\nB\nC\nD\nE\nF\nG\nH\nI\nJ\nK\nL\nM\nN\nO\nP\nQ\nR\nS\nT\nU\nV\nW\nX\nY\nZ"/>

</android.support.v4.widget.NestedScrollView>

<android.support.v4.widget.NestedScrollView

android:id="@+id/ns1"

android:layout\_width="150dp"

android:layout\_height="match\_parent"

android:layout\_gravity="top|left"

android:background="#888888"

app:layout\_behavior="com.aohuan.dodo.coordinator.utils.DodoBehavior1scroll">

<!--<com.aohuan.dodo.coordinator.view.NoScrollListView-->

<!--android:id="@+id/list1"-->

<!--android:layout\_width="150dp"-->

<!--android:layout\_height="match\_parent"-->

<!--android:layout\_gravity="top|left" />-->

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

style="@style/TextAppearance.AppCompat.Display3"

android:text="A\nB\nC\nD\nE\nF\nG\nH\nI\nJ\nK\nL\nM\nN\nO\nP\nQ\nR\nS\nT\nU\nV\nW\nX\nY\nZ"/>

</android.support.v4.widget.NestedScrollView>

</android.support.design.widget.CoordinatorLayout>

</RelativeLayout>

如果是TextView，Activity就不用填充数据了，这里就不贴对应的代码了

看一下效果

效果

我们可以得到，

* 右边 滑动，左边随着滑动
* 左边单独滑动，右边不动

和前面提到的逻辑是一样的， 因为Parent会传递给左边

**再添加一个NestedScrollView**

我们知道 NestedScrollView 可以发送事件给外面的Parent，  
也就是CoordinatorLayout  
那如果我们再添加一个 NestedScrollView  
那应该都可以发送滑动事件

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

<android.support.design.widget.CoordinatorLayout 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.v4.widget.NestedScrollView

android:id="@+id/ns0"

android:layout\_width="100dp"

android:layout\_height="match\_parent"

android:layout\_gravity="top|right"

android:background="#666666">

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

style="@style/TextAppearance.AppCompat.Display3"

android:text="A\nB\nC\nD\nE\nF\nG\nH\nI\nJ\nK\nL\nM\nN\nO\nP\nQ\nR\nS\nT\nU\nV\nW\nX\nY\nZ\nB\nC\nD\nE\nF\nG\nH\nI\nJ\nK\nL\nM\nN\nO\nP\nQ\nR\nS\nT\nU\nV\nW\nX\nY\nZ"/>

</android.support.v4.widget.NestedScrollView>

<android.support.v4.widget.NestedScrollView

android:id="@+id/ns1"

android:layout\_width="100dp"

android:layout\_height="match\_parent"

android:layout\_gravity="top|center\_horizontal"

android:background="#009900">

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

style="@style/TextAppearance.AppCompat.Display3"

android:text="A\nB\nC\nD\nE\nF\nG\nH\nI\nJ\nK\nL\nM\nN\nO\nP\nQ\nR\nS\nT\nU\nV\nW\nX\nY\nZ\nB\nC\nD\nE\nF\nG\nH\nI\nJ\nK\nL\nM\nN\nO\nP\nQ\nR\nS\nT\nU\nV\nW\nX\nY\nZ"/>

</android.support.v4.widget.NestedScrollView>

<android.support.v4.widget.NestedScrollView

android:id="@+id/ns2"

android:layout\_width="100dp"

android:layout\_height="match\_parent"

android:layout\_gravity="top|left"

android:background="#888888"

app:layout\_behavior="com.aohuan.dodo.coordinator.utils.DodoBehavior1scroll">

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

style="@style/TextAppearance.AppCompat.Display3"

android:text="A\nB\nC\nD\nE\nF\nG\nH\nI\nJ\nK\nL\nM\nN\nO\nP\nQ\nR\nS\nT\nU\nV\nW\nX\nY\nZ\nB\nC\nD\nE\nF\nG\nH\nI\nJ\nK\nL\nM\nN\nO\nP\nQ\nR\nS\nT\nU\nV\nW\nX\nY\nZ"/>

</android.support.v4.widget.NestedScrollView>

</android.support.design.widget.CoordinatorLayout>

其他一样，只是给最左边的添加了Behavior

我们来看看效果

效果

我们如果给左边2个都添加上对应的Behavior  
也就是里面任意一个滑动，左边2个都会跟着移动  
这里就不贴代码了  
我们来看看效果

效果

**简单的变动**

我们经常可以看见一些滑动后，慢慢出现一个Button按钮  
应该是用的系统的，或者自己写的  
按这个思路，简单写一个demo  
大体也就是滑动

* 到一定距离以后，显示按钮
* 再一定距离以后，隐藏按钮

大体layout

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

<android.support.design.widget.CoordinatorLayout 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.v4.widget.NestedScrollView

android:id="@+id/ns0"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:layout\_gravity="top|right"

android:background="#666666">

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

style="@style/TextAppearance.AppCompat.Display3"

android:text="A\nB\nC\nD\nE\nF\nG\nH\nI\nJ\nK\nL\nM\nN\nO\nP\nQ\nR\nS\nT\nU\nV\nW\nX\nY\nZ\nB\nC\nD\nE\nF\nG\nH\nI\nJ\nK\nL\nM\nN\nO\nP\nQ\nR\nS\nT\nU\nV\nW\nX\nY\nZ"/>

</android.support.v4.widget.NestedScrollView>

<Button

android:id="@+id/btn"

android:visibility="gone"

android:layout\_width="wrap\_content"

android:layout\_height="100dp"

android:background="#888888"

android:layout\_gravity="bottom|center\_horizontal"

android:text=" Dodo Follow "

app:layout\_behavior="com.aohuan.dodo.coordinator.utils.DodoMoveBigerBehavior"

/>

</android.support.design.widget.CoordinatorLayout>

也就只有一个 发送滑动事件的NestedScrolling  
外面一样是 NestedScrolling 的 Parent  
再有一个绑定Behavior的按钮， 接收和消费 滑动事件

对应的Behavior

package com.aohuan.dodo.coordinator.utils;

import android.animation.ObjectAnimator;

import android.content.Context;

import android.support.design.widget.CoordinatorLayout;

import android.support.v4.view.ViewCompat;

import android.support.v4.widget.NestedScrollView;

import android.util.AttributeSet;

import android.util.Log;

import android.view.View;

import android.view.animation.Animation;

import com.aohuan.dodo.coordinator.view.DodoMoveView;

/\*\*

\* Created by dodo on 2016/11/1.

\* qq: 2390183798

\*

\*

\* 根据MainView竖直方向的滑动， 设置绑定View的宽度

\* 原理也简单， 只要是竖直滑动， 动态设置宽， 添加是否可见，以及简单动画，即可

\*/

public class DodoMoveBigerBehavior extends CoordinatorLayout.Behavior<View> {

public DodoMoveBigerBehavior(Context context, AttributeSet attrs) {

super(context, attrs);

}

@Override

public boolean onStartNestedScroll(CoordinatorLayout coordinatorLayout, View child, View directTargetChild, View target, int nestedScrollAxes) {

return nestedScrollAxes == ViewCompat.SCROLL\_AXIS\_VERTICAL;

}

@Override

public void onNestedScroll(CoordinatorLayout coordinatorLayout, View child, View target, int dxConsumed, int dyConsumed, int dxUnconsumed, int dyUnconsumed) {

super.onNestedScroll(coordinatorLayout, child, target, dxConsumed, dyConsumed, dxUnconsumed, dyUnconsumed);

int followScrolled = target.getScrollY();

setBiger(child, followScrolled);

}

@Override

public boolean onNestedFling(CoordinatorLayout coordinatorLayout, View child, View target, float velocityX, float velocityY, boolean consumed) {

int followScrolled = target.getScrollY();

setBiger(child, followScrolled);

return true;

}

@Override

public void onStopNestedScroll(CoordinatorLayout coordinatorLayout, View child, View target) {

super.onStopNestedScroll(coordinatorLayout, child, target);

int followScrolled = target.getScrollY();

setBiger(child, followScrolled);

}

@Override

public boolean onNestedPreFling(CoordinatorLayout coordinatorLayout, View child, View target, float velocityX, float velocityY) {

int followScrolled = target.getScrollY();

setBiger(child, followScrolled);

return super.onNestedPreFling(coordinatorLayout, child, target, velocityX, velocityY);

}

final int MinWidth = 300;

final int MaxWidth = 450;

private void setBiger(View v, int y) {

CoordinatorLayout.MarginLayoutParams layoutParams = (CoordinatorLayout.MarginLayoutParams) v.getLayoutParams();

layoutParams.width = 200 + y/10;

if(layoutParams.width >= MinWidth && layoutParams.width <= MaxWidth){

v.setVisibility(View.VISIBLE);

}else{

v.setVisibility(View.GONE);

}

// doJudge(v, layoutParams.width);

v.setLayoutParams(layoutParams);

}

private int oldY = 0;

private boolean isMinBiggerAni = false;

private boolean isMaxBiggerAni = false;

private void doJudge(View v, int y){

Log.e("ani", "y : " +y + " == oldY : " + oldY);

if(oldY < y){

if(y >= MinWidth && !isMinBiggerAni){

isMinBiggerAni = !isMinBiggerAni;

v.setVisibility(View.VISIBLE);

doAnimat(v, true);

Log.e("ani", "now");

}

if(y >= MaxWidth && !isMaxBiggerAni){

isMaxBiggerAni = !isMaxBiggerAni;

doAnimat(v, false);

Log.e("ani", "now");

}

}

if(oldY > y){

if(y < MinWidth && isMinBiggerAni){

isMinBiggerAni = !isMinBiggerAni;

doAnimat(v, false);

Log.e("ani", "now");

}

if(y < MaxWidth && isMaxBiggerAni){

isMaxBiggerAni = !isMaxBiggerAni;

doAnimat(v, true);

v.setVisibility(View.VISIBLE);

Log.e("ani", "now");

}

}

oldY = y;

}

private void doAnimat(View v, boolean isBigger){

ObjectAnimator fViewScaleXAnim = ObjectAnimator.ofFloat(v,"scaleX",isBigger?0f:1f, isBigger?1f:0f);

fViewScaleXAnim.setDuration(500);

fViewScaleXAnim.start();

}

}

和前面的demo类似， 只是简单的修改

我们来看一下效果  
这里变大，是为了让我们感觉对应的滑动变大的关联

直接感觉

再简单添加一个动画  
看看效果

添加简单动画

这里只是为了理解 **嵌套滑动事件**  
应该会有一些bug  
自己就不继续了

**简单回顾**

这里几个demo，只是上一节理论的实例  
用NestedScrollView简单理解了 NestedScrolling的嵌套滑动事件  
由于在文章链接里面  
[卌梓的文章](https://link.jianshu.com/?t=https://segmentfault.com/a/1190000006657044)  
找到一张图，感觉说得很清楚，自己就不画图了，贴别人的  
（对应的**触摸事件** 换成 **嵌套滑动事件** 即可 ）

其他的内容，后续一起学习  
具体代码，可以见  
[https://github.com/2954722256/use\_little\_demo](https://link.jianshu.com/?t=https://github.com/2954722256/use_little_demo)  
对应 coordinator 的 Module

下一篇我们可以了解  
[和Toolbar的简单使用](https://www.jianshu.com/p/70a2b14df549)

**小礼物**