

[\(/plus/list.php?tid=31\)](#) [\(/plus/list.php?tid=12\)](#) [\(/ask\)](#) [\(/about.html\)](#)[tid=31\)](#) [lid=12\)](#)[APP \(/appdown.html\)](#)[搜索](#)[登录 \(/member/login.php\)](#) [注册 \(/member/reg_new.php\)](#)[首页 \(http://www.jcodecraeer.com/\)](#) › [安卓开发 \(/plus/list.php?tid=16\)](#) › [android开发 \(/plus/list.php?tid=18\)](#)

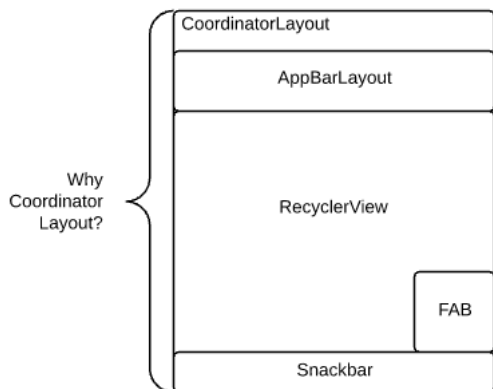
拦截一切的CoordinatorLayout Behavior

泡在网上的日子 / 文 发表于2016-02-24 11:04 第17313次阅读 Behavior (/tags.php?/Behavior/), 拦截 (/tags.php?/拦截/), CoordinatorLayout (/tags.php?/CoordinatorLayout/)
(<http://www.jiathis.com/share>)

编辑推荐: 稀土掘金 (<https://juejin.im/>), 这是一个针对技术开发者的一个应用, 你可以在掘金上获取最新最优质的技术干货, 不仅仅是Android知识、前端、后端以至于产品和设计都有涉猎, 想成为全栈工程师的朋友不要错过!

原文: Intercepting everything with CoordinatorLayout Behaviors (<https://medium.com/google-developers/intercepting-everything-with-coordinatorlayout-behaviors-8c6adc140c26#.o4gj1s87y>).

如果没有深入CoordinatorLayout (http://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.html?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog), 你注定无法在探索Android Design Support Library (http://android-developers.blogspot.com/2015/05/android-design-support-library.html?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog)的路上走多远 - Design Library中的许多view都需要一个CoordinatorLayout。但是为什么呢? CoordinatorLayout本身并没有做太多事情: 和标准的framework视图一起使用时, 它就跟一个普通的FrameLayout差不多。那么它的神奇之处来自于哪里呢? 答案就是CoordinatorLayout.Behavior (http://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.Behavior.html?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog)。通过为CoordinatorLayout的直接子view设置一个Behavior, 就可以拦截touch events, window insets, measurement, layout, 和 nested scrolling等动作。Design Library大量利用了Behaviors来实现你所看到的功能。



创建一个Behavior

创建一个behavior很简单: 继承Behavior即可。

```

1.  public class FancyBehavior<V extends View>
2.      extends CoordinatorLayout.Behavior<V> {
3.      /**
4.       * Default constructor for instantiating a FancyBehavior in code.
5.       */
6.      public FancyBehavior() {
7.      }
8.      /**
9.       * Default constructor for inflating a FancyBehavior from layout.
10.     */
11.     * @param context The {@link Context}.
12.     * @param attrs The {@link AttributeSet}.
13.     */
14.     public FancyBehavior(Context context, AttributeSet attrs) {
15.         super(context, attrs);
16.         // Extract any custom attributes out
17.         // preferably prefixed with behavior_ to denote they
18.         // belong to a behavior
19.     }
20. }

```

注意这个类设置的是普通View，这意味着你可以把FancyBehavior设置给任何View类。但是，如果你只允许让Behavior设置给一个特定类型的View，则需要这样写：

```

1.  public class FancyFrameLayoutBehavior
2.      extends CoordinatorLayout.Behavior<FancyFrameLayout>

```

这可以省去把回调方法中收到的view参数转换成正确类型的步骤 - 效率第一嘛。

可以使用Behavior.setTag() (http://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.Behavior.html?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog#setTag%28android.view.View,%20java.lang.Object%29) (http://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.Behavior.html?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog#getTag%28android.view.View%29) 来保存临时数据，还可以使用onSaveInstanceState() (http://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.Behavior.html?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog#onSaveInstanceState%28android.support.design.widget.CoordinatorLayout%29) (http://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.Behavior.html?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog#onRestoreInstanceState%28android.support.design.widget.CoordinatorLayout%29) 来保存跟Behavior相关的实例的状态。我建议让Behaviors尽可能的轻，但是这些方法让状态化Behaviors成为可能。

设置Behavior

当然了，Behaviors并不会对自身做任何事情 - 它们需要被设置在一个CoordinatorLayout的子view上之后才会被实际调用。设置Behaviors主要有三种方式：程序中动态设置、xml布局文件设置和使用注解设置。

在程序中设置Behavior

当你认为Behavior是一个被设置在CoordinatorLayout每个子view上的附加数据时，你就不会对Behavior其实是保存在每个view的LayoutParams中感到奇怪了（如果你已经阅读了我们 关于布局的文章 (<https://medium.com/@ianhlake/layouts-attributes-and-you-9e5a4b4fe32c>)） - 这也是为什么Behaviors需要声明在CoordinatorLayout的直接子View上的原因，因为只有那些子View才存有CoordinatorLayout.LayoutParams（根据自己的理解翻译的）。

```

1.  FancyBehavior fancyBehavior = new FancyBehavior();
2.  CoordinatorLayout.LayoutParams params =
3.      (CoordinatorLayout.LayoutParams) yourView.getLayoutParams();
4.  params.setBehavior(fancyBehavior);

```

这里你会发现我们使用的是默认的空构造函数。但这并不是说你就不能使用任何参数 - 如果你想，代码里面，万事皆有可能。

在xml里设置Behavior

当然，每次都在代码里面把所有事情做完会显得有点乱。就跟多数自定义的LayoutParam一样，这里也有相应的layout_ attribute 与之对应。那就是layout_behavior 属性：

```
1. <FrameLayout
2.     android:layout_height="wrap_content"
3.     android:layout_width="match_parent"
4.     app:layout_behavior=".FancyBehavior" />
```

这里与前面不同的是，被调用的构造函数总是FancyBehavior(Context context, AttributeSet attrs)。因此，你可以在xml属性中声明你想要的其他自定义属性。如果你想让开发者能够通过xml自定义Behavior的功能，这点是很重要的。

注意：类似于由父类负责解析和解释的layout_ 属性命名规则，使用behavior_ prefix来指定被专门Behavior使用的某个属性。

例子（译者结合评论做的补充）：

```
1. <FrameLayout
2.     android:layout_width="match_parent"
3.     android:layout_height="match_parent"
4.     app:layout_behavior=".MaxWidthBehavior"
5.     app:behavior_maxWidth="400dp" />
```

自动设置一个Behavior

如果你正在创建一个需要一个自定义Behavior的自定义View（就如Design Library中的许多控件那样），那么你很可能希望view默认就设置了那个Behavior，而不需要每次都通过xml或者代码去手动指定。为此，你只需在自定义View类的最上面设置一个简单的注解：

```
1. @CoordinatorLayout.DefaultBehavior(FancyFrameLayoutBehavior.class)
2. public class FancyFrameLayout extends FrameLayout {
3. }
```

你会发现你的Behavior会随着默认的构造函数被调用，这非常类似于与通过程序设置Behavior。注意任何 layout_behavior属性所代表的Behavior都会重写 DefaultBehavior (<http://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.DefaultBehavior.html>?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog)。

拦截 Touch Events

一旦你设置好了所有的behavior，你就该准备做点实际工作了。Behavior能做的事情之一就是拦截触摸事件。

如果没有CoordinatorLayout，我们通常会被牵涉进 ViewGroup的子类中，就像 Managing Touch Events training (<http://developer.android.com/training/gestures/viewgroup.html>?

utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog)一文所讨论的那样。但是如果有了CoordinatorLayout，CoordinatorLayout就会把它onInterceptTouchEvent() (http://developer.android.com/reference/android/view/ViewGroup.html?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog#onInterceptTouchEvent%28android.view.M的参数（主要是MotionEvent）和调用传递到Behavior的onInterceptTouchEvent() (http://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.Behavior.html?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog#onInterceptTouchEvent%28android.support让你的Behavior有一次拦截触摸事件的机会。如果返回true，你的Behavior则会通过onTouchEvent() (http://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.Behavior.html?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog#onTouchEvent%28android.support.design收到所有的后续触摸事件 - 而View完全不知道发生了什么事情。这也是SwipeDismissBehavior (http://developer.android.com/reference/android/support/design/widget/SwipeDismissBehavior.html?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog) 在view上的工作原理。

ps：我以前专门分析过SwipeDismissBehavior，和这段话基本一致。另外CoordinatorLayout其实是遍历了一遍自己的直接子View，一个一个的调用子view中的Behavior，见：SwipeDismissBehavior用法及实现原理 (<http://www.jcodecraeer.com/a/anzhuokaifa/androidkaifa/2015/1103/3650.html>)。

不过还有一个更粗暴的触摸拦截：拦截所有的交互。只需在 `blocksInteractionBelow()`

([http://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.Behavior.html?](http://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.Behavior.html?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog#blocksInteractionBelow%28android.support)

`utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog#blocksInteractionBelow%28android.support`
返回true即可（我们这个视图下的其他视图将获取不到任何Touch事件）。当然，你可能希望在交互被阻止的情况下能有一些视觉效果 - 这就是为什么`blocksInteractionBelow()`实际上默认依赖 `getScrimOpacity()`

([http://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.Behavior.html?](http://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.Behavior.html?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog#getScrimOpacity%28android.support)

`utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog#getScrimOpacity%28android.support`
的值 - 返回一个非零将在View之上绘制一层overlay颜色并且屏蔽所有的交互。

拦截Window Insets

假设你读了Why would I want to fitSystemWindows? blog (https://medium.com/google-developers/why-would-i-want-to-fitssystemwindows-4e26d9ce1eec?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog)。那里深入讨论了fitSystemWindows到底干什么的，但是它归纳为：window insets 需要避免在 system windows（比如status bar 和 navigation bar）的下面绘制。

Behaviors在这里也有拦截的机会 - 如果你的View是fitSystemWindows="true"的，那么任何依附着的Behavior都将得到onApplyWindowInsets()

([http://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.Behavior.html?](http://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.Behavior.html?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog#onApplyWindowInsets%28android.support)

`utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog#onApplyWindowInsets%28android.support`
调用，且优先级高于View自身。

注意：如果你的Behavior并没有消费掉整个 window insets，它应该通过ViewCompat.dispatchApplyWindowInsets()

([http://developer.android.com/reference/android/support/v4/view/ViewCompat.html?](http://developer.android.com/reference/android/support/v4/view/ViewCompat.html?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog#dispatchApplyWindowInsets%28android.view.View,%20andri)

`utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog#dispatchApplyWindowInsets%28android.view.View,%20andri`
递insets，以确保任何子view都能有机会看到这个WindowInsets。

拦截Measurement 和 layout

测量与布局（Measurement and layout）是 安卓如何绘制View ([http://developer.android.com/guide/topics/ui/how-android-draws.html?](http://developer.android.com/guide/topics/ui/how-android-draws.html?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog)

`utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog`)的关键组成部分。因此对于能够拦截一切的Behavior来说，它应该能在第一时间拦截测量和布局才是合情合理的。这要通过onMeasureChild()

([http://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.Behavior.html?](http://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.Behavior.html?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog#onMeasureChild%28android.support)

`utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog#onMeasureChild%28android.support`
设计

([http://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.Behavior.html?](http://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.Behavior.html?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog#onLayoutChild%28android.support)

`utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog#onLayoutChild%28android.support`
调来完成。

比如，我们找来任意一个普通的ViewGroup，并向它添加一个maxWidth：

```

1.  /*
2.   * Copyright 2015 Google Inc.
3.   *
4.   * Licensed under the Apache License, Version 2.0 (the "License");
5.   * you may not use this file except in compliance with the License.
6.   * You may obtain a copy of the License at
7.   *
8.   *     http://www.apache.org/licenses/LICENSE-2.0
9.   *
10.  * Unless required by applicable law or agreed to in writing, software
11.  * distributed under the License is distributed on an "AS IS" BASIS,
12.  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13.  * See the License for the specific language governing permissions and
14.  * limitations under the License.
15.  */
16.
17.  package com.example.behaviors;
18.
19.  import android.content.Context;
20.  import android.content.res.TypedArray;
21.  import android.support.design.widget.CoordinatorLayout;
22.  import android.util.AttributeSet;
23.  import android.view.ViewGroup;
24.
25.  import static android.view.View.MeasureSpec;
26.
27.  /**
28.   * Behavior that imposes a maximum width on any ViewGroup.
29.   *
30.   * <p />Requires an attrs.xml of something like
31.   *
32.   * <pre>
33.   * <declare-styleable name="MaxWidthBehavior_Params">
34.   *     <attr name="behavior_maxWidth" format="dimension"/>
35.   * </declare-styleable>
36.   * </pre>
37.   */
38.  public class MaxWidthBehavior<V extends ViewGroup> extends CoordinatorLayout.Behavior<V> {
39.      private int mMaxWidth;
40.
41.      public MaxWidthBehavior(Context context, AttributeSet attrs) {
42.          super(context, attrs);
43.          TypedArray a = context.obtainStyledAttributes(attrs,
44.              R.styleable.MaxWidthBehavior_Params);
45.          mMaxWidth = a.getDimensionPixelSize(
46.              R.styleable.MaxWidthBehavior_Params_behavior_maxWidth, 0);
47.          a.recycle();
48.      }
49.
50.      @Override
51.      public boolean onMeasureChild(CoordinatorLayout parent, V child,
52.          int parentWidthMeasureSpec, int widthUsed,
53.          int parentHeightMeasureSpec, int heightUsed) {
54.          if (mMaxWidth <= 0) {
55.              // No max width means this Behavior is a no-op
56.              return false;
57.          }
58.          int widthMode = MeasureSpec.getMode(parentWidthMeasureSpec);
59.          int width = MeasureSpec.getSize(parentWidthMeasureSpec);
60.
61.          if (widthMode == MeasureSpec.UNSPECIFIED || width > mMaxWidth) {
62.              // Sorry to impose here, but max width is kind of a big deal
63.              width = mMaxWidth;
64.              widthMode = MeasureSpec.AT_MOST;
65.              parent.onMeasureChild(child,
66.                  MeasureSpec.makeMeasureSpec(width, widthMode), widthUsed,
67.                  parentHeightMeasureSpec, heightUsed);
68.              // We've measured the View, so CoordinatorLayout doesn't have to
69.              return true;
70.          }
71.
72.          // Looks like the default measurement will work great
73.          return false;
74.      }
75.  }

```


当嵌套滚动（或者flinging）结束，你将得到一个onStopNestedScroll()

(http://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.Behavior.html?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog#onStopNestedScroll%28android.support.design.widget.CoordinatorLayout.Behavior.html)

回调。这标志着滚动的结束 - 迎接在下一个滚动之前的onStartNestedScroll() 调用。

比如，当向下滚动的时候隐藏FloatingActionButton，向上滚动的时候显示FloatingActionButton - 这只牵涉到重写

onStartNestedScroll() 和 onNestedScroll()，就如在ScrollAwareFABBehavior

(https://github.com/ianhanniballake/cheesesquare/blob/scroll_aware_fab/app/src/main/java/com/support/android/designlibdemo/ScrollAwareFABBehavior.java)中所看到的那样。

这只是开始

Behavior每个单独的部分都很有趣，当他们结合起来就会发生很神奇的事情。为了了解更多的高级behavior，我强烈鼓励你去查看Design Library的源码 - Android SDK Search Chrome extension (https://chrome.google.com/webstore/detail/android-sdk-search/hgcbfeicehlpmgmnhnkjbjoldkfhoi?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog)是我探索AOSP源码时最喜欢的资源（虽然包含在 <android-sdk>/extras/android/m2repository中的源码总是最新的）。

在了解Behavior能做哪些事情这点上打下了坚实的基础后，让我知道你们是如何使用它们创建更优秀的app的。

要了解更多，请参与在 Google+ post (https://plus.google.com/+AndroidDevelopers/posts/WojFEkDdFNe?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog)

上的讨论并关注 Android Development Patterns Collection (https://plus.google.com/collection/sLR0p?utm_campaign=adp_series_coordinatorlayoutbehavior_021716&utm_source=medium&utm_medium=blog) !



他们收藏了这篇文章



相关文章

关于CoordinatorLayout与Behavior的一点分析 (/a/anzhuokaifa/androidkaifa/2015/1201/3738.html)	2015-12-01
CoordinatorLayout 自定义Behavior并不难，由简到难手把手带你撸三款! (/a/anzhuokaifa/androidkaifa/2016/0824/6565.html)	2016-08-24
CoordinatorLayout与快速返回的实现 (/a/anzhuokaifa/androidkaifa/2015/0818/3315.html)	2015-08-18
CoordinatorLayout与WebView一起使用 (/a/anzhuokaifa/androidkaifa/2017/0412/7810.html)	2017-04-12
来来来，随老夫撸一个支付宝玩玩——自定义Behavior的正确打开方式 (/a/anzhuokaifa/androidkaifa/2017/0725/8240.html)	2017-07-25
Android Support Design 中 CoordinatorLayout 与 Behaviors 初探 (/a/anzhuokaifa/androidkaifa/2015/0707/3149.html)	2015-07-07
自定义CoordinatorLayout的Behavior实现知乎和简书快速返回效果 (/a/anzhuokaifa/androidkaifa/2015/0913/3447.html)	2015-09-13
在CoordinatorLayout中实现底部导航栏的滚动隐藏 (/a/anzhuokaifa/androidkaifa/2017/0428/7887.html)	2017-05-06
CoordinatorLayout与滚动的处理 (/a/anzhuokaifa/androidkaifa/2015/0717/3196.html)	2017-04-28


上一篇：**【有人@我】Android中高亮变色显示文本中的关键字 (/a/anzhuokaifa/androidkaifa/2016/0224/3990.html)**

应该是好久没有写有关技术类的文章了，前天还有人在群里问我，说群主很长时间没有分享干货了，今天分享一篇Android中TextView在大段的文字内容中如何让关键字高亮变色的文章，希望对大家有所帮助，我终于在歪路上回归正途了。这个篇文章在平时应该还算比较


下一篇：[深入理解LayoutInflater.inflate\(\)](#) (/a/anzhuokaifa/androidkaifa/2016/0225/3992.html)

原文链接：<https://www.bignerdranch.com/blog/understanding-androids-layoutinflater-inflate/> 译文链接：<http://blog.chengdazhi.com/index.php/110> 由于我们很容易习惯公式化的预置代码，有时我们会忽略很优雅的细节。LayoutInflater以及它在Fragment


发表评论



benjamin (/member/index.php?uid=benjamin) · 2017-01-05
写的非常好。学习了。
(/member/index.php?uid=benjamin) 0 回复




网友101.81.123.128 · 2016-03-01
网友61.190.38.18 的原帖：
翻译腔太重，看的糊里糊涂
有原文地址啊。。
0 0 回复



网友61.190.38.18 · 2016-03-01
翻译腔太重，看的糊里糊涂
0 0 回复




网友58.41.216.63 · 2016-02-28
网友124.128.9.122 的原帖：
小鄧子的原帖：
哇哦，特别期待这一篇，感谢泡网，感谢菠萝哥。
我也有两篇介绍Behavior的博客：<http://blog.csdn.net/qibin0506/article/details/50290421>
和<http://blog.csdn.net/qibin0506/article/details/50377592>
看过你写的，非常棒，第二篇还有点不理解
0 0 回复




网友124.128.9.122 · 2016-02-24
小鄧子 的原帖：
哇哦，特别期待这一篇，感谢泡网，感谢菠萝哥。
我也有两篇介绍Behavior的博客：<http://blog.csdn.net/qibin0506/article/details/50290421>
和<http://blog.csdn.net/qibin0506/article/details/50377592>
0 0 回复



网友182.18.19.162 · 2016-02-24
赞~
0 0 回复



泡在网上的日子 (/member/index.php?uid=jianghejie) · 2016-02-24
(/member/index.php?uid=jianghejie) 小鄧子的原帖：
哇哦，特别期待这一篇，感谢泡网，感谢菠萝哥。
我觉得没翻译好
0 0 回复



小鄧子 (/member/index.php?uid=小鄧子) · 2016-02-24
哇哦，特别期待这一篇，感谢泡网，感谢菠萝哥。
(/member/index.php?uid=小鄧子) 0 0 回复
小鄧子

推荐文章

NumberProgressBar: 一个简约性感的数字ProgressBar (</a/anzhuokaifa/androidkaifa/2014/0813/1645.html>)

终于等到你Depth-LIB-Android (</a/anzhuokaifa/androidkaifa/2016/0429/4200.html>)

App开发架构指南 (谷歌官方文档译文) (</a/anzhuokaifa/androidkaifa/2017/0523/7963.html>)

Android插件化原理解析——概要 (</a/anzhuokaifa/androidkaifa/2016/0227/4005.html>)

将Eclipse代码导入到Android Studio的两种方式 (</a/anzhuokaifa/androidkaifa/2015/0104/2259.html>)

SQLite: 一个响应式的数据查询框架 (</a/anzhuokaifa/androidkaifa/2015/0306/2552.html>)

赞助商



([https://www.trustauth.cn/marketing/freesssl.html?](https://www.trustauth.cn/marketing/freesssl.html?utm_source=jcodecraeer&utm_medium=social&utm_campaign=freessslnew&utm_content=freessslnew)

[utm_source=jcodecraeer&utm_medium=social&utm_campaign=freessslnew&utm_content=freessslnew](https://www.trustauth.cn/marketing/freesssl.html?utm_source=jcodecraeer&utm_medium=social&utm_campaign=freessslnew&utm_content=freessslnew))

Copyright 2011 - 2016 jcodecraeer.com All Rights Reserved.

蜀ICP备12021840号-1

本站文章用于学习交流

本站CDN / 存储服务由又拍云  又拍云 (<https://console.upyun.com/register/?invite=H1NEyK4L->

[&utm_source=Referral&utm_medium=jcode&utm_content=dex](https://console.upyun.com/register/?invite=H1NEyK4L-&utm_source=Referral&utm_medium=jcode&utm_content=dex))提供

新浪微博 (<http://weibo.com/u/2711441293>) qq群—161644793 qq群二98711210

网站地图 (</sitemap/>)