



(/plus/list.php?tid=16) (/plus/list.php?tid=18) (/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)

## 完全掌握Android Data Binding

泡在网上的日子 / 文 发表于2015-06-03 23:39 第80464次阅读 数据绑定 (/tags.php?/数据绑定/)

(<http://www.jiathis.com/share>)

42

来源 <https://github.com/LyndonChin/MasteringAndroidDataBinding>

(<https://github.com/LyndonChin/MasteringAndroidDataBinding>)

**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！

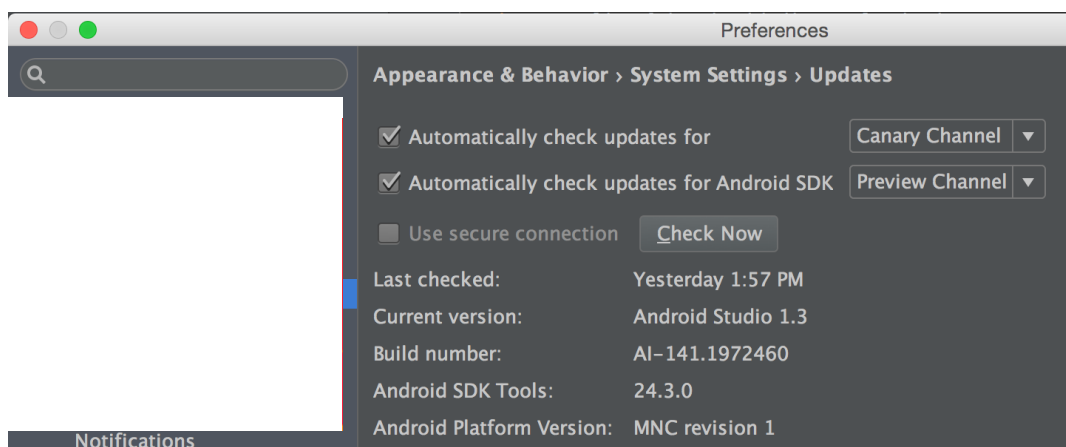
本教程是跟着 Data Binding Guide (<https://developer.android.com/tools/data-binding/guide.html>) 学习过程中得出的一些实践经验，同时修改了官方教程的一些错误，每一个知识点都有对应的源码，争取做到实践与理论相结合。

Data Binding 解决了 Android UI 编程中的一个痛点，官方原生支持 MVVM 模型可以让我们在不改变既有代码框架的前提下，非常容易地使用这些新特性。其实在此之前，已经有些第三方的框架（RoboAndroid ([http://robobinding.github.io/RoboBinding/getting\\_started.zh.html](http://robobinding.github.io/RoboBinding/getting_started.zh.html))) 可以支持 MVVM 模型，无耐由于框架的侵入性太强，导致一直没有流行起来。

## 准备

### Android Studio 更新到 1.3 版本

打开 Preferences，找到 **Appearance & Behavior** 下的 **Updates** 选项，把 **Automatically Check updates for** 修改成 **Canary Channel**。



## 注意

*Data Binding 是一个 support 包，因此与 Android M 没什么关系，可以不用下载 Android MNC Preview 的 SDK。*

## 新建一个 Project

修改 Project 的 build.gradle (<https://github.com/LyndonChin/MasteringAndroidDataBinding/blob/master/build.gradle>)，为 build script 添加一条依赖，Gradle 版本为 1.2.3。

```
1. classpath 'com.android.tools.build:gradle:1.2.3'
2. classpath 'com.android.databinding:dataBinder:1.0-rc0'
```

为用到 Data Binding 的模块添加插件，修改对应的 build.gradle

(<https://github.com/LyndonChin/MasteringAndroidDataBinding/blob/master/app/build.gradle>)。

```
1. apply plugin: 'com.android.databinding'
```

## 注意

如果 Module 用到的 **buildToolsVersion** 高于 **22.0.1**，比如 **23 rc1**，那 `com.android.databinding:dataBinder` 的版本要改为 **1.3.0-beta1**，否则会出现如下错误：



## 基础

工程创建完成后，我们通过一个最简单的例子来说明 Data Binding 的基本用法。

## 布局文件

使用 Data Binding 之后，xml 的布局文件就不再单纯地展示 UI 元素，还需要定义 UI 元素用到的变量。所以，它的根节点不再是一个 ViewGroup，而是变成了 layout，并且新增了一个节点 data。

```
1. <layout xmlns:android="http://schemas.android.com/apk/res/android">
2.     <data>
3.     </data>
4.     <!--原先的根节点 (Root Element) -->
5.     <LinearLayout>
6.         ....
7.     </LinearLayout>
8. </layout>
```

要实现 MVVM 的 ViewModel 就需要把数据与 UI 进行绑定，data 节点就为此提供了一个桥梁，我们先在 data 中声明一个 variable，这个变量会为 UI 元素提供数据（例如 TextView 的 android:text），然后在 Java 代码中把“后台”数据与这个 variable 进行绑定。

如果要用一个表格来展示用户的基本信息，用 Data Binding 应该怎么实现呢？

## 数据对象

添加一个 POJO 类 - User，非常简单，四个属性以及他们的 getter 和 setter。

```
1.  public class User {
2.      private final String firstName;
3.      private final String lastName;
4.      private String displayName;
5.      private int age;
6.
7.      public User(String firstName, String lastName) {
8.          this.firstName = firstName;
9.          this.lastName = lastName;
10.     }
11.
12.     public User(String firstName, String lastName, int age) {
13.         this(firstName, lastName);
14.         this.age = age;
15.     }
16.
17.     public int getAge() {
18.         return age;
19.     }
20.
21.     public String getFirstName() {
22.         return firstName;
23.     }
24.
25.     public String getLastName() {
26.         return lastName;
27.     }
28.
29.     public String getDisplayName() {
30.         return firstName + " " + lastName;
31.     }
32.
33.     public boolean isAdult() {
34.         return age >= 18;
35.     }
36. }
```

稍后，我们会新建一个 User 类型的变量，然后把它跟布局文件中声明的变量进行绑定。

## 定义 Variable

再回到布局文件，在 data 节点中声明一个变量 user。

```
1.  <data>
2.      <variable name="user" type="com.liangfeizc.databindingsamples.basic.User" />
3.  </data>
```

其中 type 属性就是我们在 Java 文件中定义的 User 类。

当然，data 节点也支持 import，所以上面的代码可以换一种形式来写。

```
1.  <data>
2.      <import type="com.liangfeizc.databindingsamples.basic.User" />
3.      <variable name="user" type="User" />
4.  </data>
```

然后我们刚才在 build.gradle 中添加的那个插件 - com.android.databinding 会根据 xml 文件的名称 **Generate** 一个继承自 ViewDataBinding 的类。

例如，这里 xml 的文件名叫 activity\_basic.xml，那么生成的类就是 ActivityBasicBinding。

## 注意

java.lang.\* 包中的类会被自动导入，可以直接使用，例如要定义一个 String 类型的变量：

```
1.  <variable name="firstName" type="String" />
```

## 绑定 Variable

修改 BaseActivity 的 onCreate 方法，用 DataBindingUtil.setContentView() 来替换掉 setContentView()，然后创建一个 user 对象，通过 binding.setUser(user) 与 variable 进行绑定。

```
1.  @Override
2.  protected void onCreate(Bundle savedInstanceState) {
3.      super.onCreate(savedInstanceState);
4.      ActivityBasicBinding binding = DataBindingUtil.setContentView(
5.          this, R.layout.activity_basic);
6.      User user = new User("fei", "Liang");
7.      binding.setUser(user);
8.  }
```

## 注意

ActivityBasicBinding 类是自动生成的，所有的 set 方法也是根据 variable 名称生成的。例如，我们定义了两个变量。

```
1.  <data>
2.      <variable name="firstName" type="String" />
3.      <variable name="lastName" type="String" />
4.  </data>
```

那么就会生成对应的两个 set 方法。

```
1.  setFirstName(String firstName);
2.  setLastName(String lastName);
```

## 使用 Variable

数据与 Variable 绑定之后，xml 的 UI 元素就可以直接使用了。

```
1.  <TextView
2.      android:layout_width="wrap_content"
3.      android:layout_height="wrap_content"
4.      android:text="@{user.lastName}" />
```

至此，一个简单的数据绑定就完成了，可参考完整代码

(<https://github.com/LyndonChin/MasteringAndroidDataBinding/tree/master/app/src/main/java/com/liangfeizc/databindingsamples/basic>)

## 高级用法

### 使用类方法

首先为类添加一个静态方法

```
1.  public class MyStringUtils {
2.      public static String capitalize(final String word) {
3.          if (word.length() > 1) {
4.              return String.valueOf(word.charAt(0)).toUpperCase() + word.substring(1);
5.          }
6.          return word;
7.      }
8.  }
```

然后在 xml 的 data 节点中导入：

```
1. <import type="com.liangfeizc.databindingsamples.utils.MyStringUtils" />
```

使用方法与 Java 语法一样：

```
1. <TextView
2.     android:layout_width="wrap_content"
3.     android:layout_height="wrap_content"
4.     android:text="@{StringUtils.capitalize(user.firstName)}" />
```

## 类型别名

如果我们在 data 节点导入了两个同名的类怎么办？

```
1. <import type="com.example.home.data.User" />
2. <import type="com.example.detail.data.User" />
3. <variable name="user" type="User" />
```

这样一来出现了两个 User 类，那 user 变量要用哪一个呢？不用担心，import 还有一个 alias 属性。

```
1. <import type="com.example.home.data.User" />
2. <import type="com.example.detail.data.User" alias="DetailUser" />
3. <variable name="user" type="DetailUser" />
```

## Null Coalescing 运算符

```
1. android:text="@{user.displayName ?? user.lastName}"
```

就等价于

```
1. android:text="@{user.displayName != null ? user.displayName : user.lastName}"
```

## 属性值

通过 \${} 可以直接把 Java 中定义的属性值赋值给 xml 属性。

```
1. <TextView
2.     android:text="@{user.lastName}"
3.     android:layout_width="wrap_content"
4.     android:layout_height="wrap_content"
5.     android:visibility="@{user.isAdult ? View.VISIBLE : View.GONE}"/>
```

## 使用资源数据

这个例子，官方教程有错误，可以参考Android Data Binder 的一个bug (<http://blog.csdn.net/feelang/article/details/46342699>)，完整代码在此 ([https://github.com/LyndonChin/MasteringAndroidDataBinding/blob/master/app/src/main/res/layout/activity\\_resource.xml](https://github.com/LyndonChin/MasteringAndroidDataBinding/blob/master/app/src/main/res/layout/activity_resource.xml))。

```
1. <TextView
2.     android:padding="@{large? (int)@dimen/largePadding : (int)@dimen/smallPadding}"
3.     android:background="@android:color/black"
4.     android:textColor="@android:color/white"
5.     android:layout_width="wrap_content"
6.     android:layout_height="wrap_content"
7.     android:text="@string/hello_world" />
```

## Observable Binding

本来这一节的标题应该叫双向绑定，但是很遗憾，现在的 Data Binding 暂时支持单向绑定，还没有达到 Angular.js 的威力。

要实现 Observable Binding，首先得有一个实现了 `android.databinding.Observable` 的类，为了方便，Android 原生提供了已经封装好的一个类 - `BaseObservable`，并且实现了监听器的注册机制。

我们可以直接继承 `BaseObservable`。

```
1.  public class ObservableUser extends BaseObservable {
2.      private String firstName;
3.      private String lastName;
4.      @Bindable
5.      public String getFirstName() {
6.          return firstName;
7.      }
8.      @Bindable
9.      public String getLastName() {
10.         return lastName;
11.     }
12.     public void setFirstName(String firstName) {
13.         this.firstName = firstName;
14.         notifyPropertyChanged(BR.firstName);
15.     }
16.     public void setLastName(String lastName) {
17.         this.lastName = lastName;
18.         notifyPropertyChanged(BR.lastName);
19.     }
20. }
```

`BR` 是编译阶段生成的一个类，功能与 `R.java` 类似，用 `@Bindable` 标记过 `getter` 方法会在 `BR` 中生成一个 `entry`，当我们

通过代码可以看出，当数据发生变化时还是需要手动发出通知。通过调用 `notifyPropertyChanged(BR.firstName)` 来通知系统 `BR.firstName` 这个 `entry` 的数据已经发生变化，需要更新 UI。

除此之外，还有一种更细粒度的绑定方式，可以具体到成员变量，这种方式无需继承 `BaseObservable`，一个简单的 **POJO** 就可以实现。

```
1.  public class PlainUser {
2.      public final ObservableField<String> firstName = new ObservableField<>();
3.      public final ObservableField<String> lastName = new ObservableField<>();
4.      public final ObservableInt age = new ObservableInt();
5.  }
```

系统为我们提供了所有的 **primitive type** 所对应的 **Observable** 类，例如 `ObservableInt`、`ObservableFloat`、`ObservableBoolean` 等等，还有一个 `ObservableField` 对应着 **reference type**。

剩下的数据绑定与前面介绍的方式一样，具体可参考 `ObservableActivity`

(<https://github.com/LyndonChin/MasteringAndroidDataBinding/blob/master/app/src/main/java/com/liangfeizc/databindingsamples/observable/Observe>

## 带 ID 的 View

**Data Binding** 有效降低了代码的冗余性，甚至完全没有必要再去获取一个 View 实例，但是情况不是绝对的，万一我们真的就需要了呢？不用担心，只要给 View 定义一个 ID，**Data Binding** 就会为我们生成一个对应的 `final` 变量。

```
1.  <TextView
2.      android:id="@+id/firstName"
3.      android:layout_width="wrap_content"
4.      android:layout_height="wrap_content" />
```

上面代码中定义了一个 ID 为 `firstName` 的 `TextView`，那么它对应的变量就是

```
1. public final TextView firstName;
```

具体代码可参考 `ViewWithIDsActivity.java`

(<https://github.com/LyndonChin/MasteringAndroidDataBinding/blob/master/app/src/main/java/com/liangfeizc/databindingsamples/viewids/ViewWithIDs>

。

## ViewStubs

xml中的 `ViewStub` 经过 binding 之后会转换成 `ViewStubProxy`, 具体代码可参考 `ViewStubActivity.java`

(<https://github.com/LyndonChin/MasteringAndroidDataBinding/blob/master/app/src/main/java/com/liangfeizc/databindingsamples/viewstub/ViewStubAr>

简单用代码说明一下, xml 文件与之前的代码一样, 根节点改为 `layout`, 在 `LinearLayout` 中添加一个 `ViewStub`, 添加 **ID**。

```
1. <layout xmlns:android="http://schemas.android.com/apk/res/android">
2.     <LinearLayout
3.         ...>
4.         <ViewStub
5.             android:id="@+id/view_stub"
6.             android:layout="@layout/view_stub"
7.             ... />
8.     </LinearLayout>
9. </layout>
```

在 Java 代码中获取 binding 实例, 为 `ViewStubProy` 注册 `ViewStub.OnInflateListener` 事件, 搞定!

```
1. binding = DataBindingUtil.setContentView(this, R.layout.activity_view_stub);
2. binding.viewStub.setOnInflateListener(new ViewStub.OnInflateListener() {
3.     @Override
4.     public void onInflate(ViewStub stub, View inflated) {
5.         ViewStubBinding binding = DataBindingUtil.bind(inflated);
6.         User user = new User("fee", "lang");
7.         binding.setUser(user);
8.     }
9. });
```

## Dynamic Variables

完整代码可以参考 `dynamic`

(<https://github.com/LyndonChin/MasteringAndroidDataBinding/tree/master/app/src/main/java/com/liangfeizc/databindingsamples/dynamic>)

以 `RecyclerView` 为例, Adapter 的 **DataBinding** 需要动态生成, 因此我们可以在 `onCreateViewHolder` 的时候创建这个 **DataBinding**, 然后在 `onBindViewHolder` 中获取这个 **DataBinding**。

```

1.  public static class BindingHolder extends RecyclerView.ViewHolder {
2.      private ViewDataBinding binding;
3.      public BindingHolder(View itemView) {
4.          super(itemView);
5.      }
6.      public ViewDataBinding getBinding() {
7.          return binding;
8.      }
9.      public void setBinding(ViewDataBinding binding) {
10.         this.binding = binding;
11.     }
12. }
13. @Override
14. public BindingHolder onCreateViewHolder(ViewGroup viewGroup, int i) {
15.     ViewDataBinding binding = DataBindingUtil.inflate(
16.         LayoutInflater.from(viewGroup.getContext()),
17.         R.layout.list_item,
18.         viewGroup,
19.         false);
20.     BindingHolder holder = new BindingHolder(binding.getRoot());
21.     holder.setBinding(binding);
22.     return holder;
23. }
24. @Override
25. public void onBindViewHolder(BindingHolder holder, int position) {
26.     User user = users.get(position);
27.     holder.getBinding().setVariable(BR.user, user);
28.     holder.getBinding().executePendingBindings();
29. }

```

注意此处 `DataBindingUtil` 的用法：

```

1.  ViewDataBinding binding = DataBindingUtil.inflate(
2.      LayoutInflater.from(viewGroup.getContext()),
3.      R.layout.list_item,
4.      viewGroup,
5.      false);

```

## Attribute setters

有了 **Data Binding**，即使属性没有在 `declare-styleable` 中定义，我们也可以通过 xml 进行赋值操作。为了演示这个功能，我自定义了一个 `View - UserView`

(<https://github.com/LyndonChin/MasteringAndroidDataBinding/blob/master/app/src/main/java/com/liangfeizc/databindingsamples/attributesetters/UserView.java>)

其中 `R.styleable.UserView` (<https://github.com/LyndonChin/MasteringAndroidDataBinding/blob/master/app/src/main/res/values/styles.xml>) 中只定义了一个 `age` 属性，其中 `firstName` 和 `lastName` 只有对应的两个 `setter` 方法。

只要有 `setter` 方法就可以这样为属性赋值：

```

1.  <com.liangfeizc.databindingsamples.attributesetters.UserView
2.      android:layout_width="match_parent"
3.      android:layout_height="wrap_content"
4.      android:paddingLeft="@dimen/largePadding"
5.      app:onClickListener="@{activity.clickListener}"
6.      app:firstName="@{@string/firstName}"
7.      app:lastName="@{@string/lastName}"
8.      app:age="27" />

```

`onClickListener` 也是同样道理，只不过我们是在 `Activity` 中定义了一个 `Listener`。

## 转换器 (Converters)

具体代码可参考 `ConversionsActivity.java`

(<https://github.com/LyndonChin/MasteringAndroidDataBinding/blob/master/app/src/main/java/com/liangfeizc/databindingsamples/converters/ConversionsActivity.java>)

在 xml 中为属性赋值时，如果变量的类型与属性不一致，通过 **DataBinding** 可以进行转换。



```

1. <Button
2.     android:onClick="toggleIsError"
3.     android:text="@{isError.get() ? @color/red : @color/white}"
4.     android:layout_width="match_parent"
5.     android:layout_height="wrap_content" />

```

只需要定义一个标记了 `@BindingConversion` 的静态方法即可：

```

1. @BindingConversion
2. public static int convertColorToString(int color) {
3.     switch (color) {
4.         case Color.RED:
5.             return R.string.red;
6.         case Color.WHITE:
7.             return R.string.white;
8.     }
9.     return R.string.app_name;
10. }

```

至此，官网所介绍的用法都在代码中实践过了，如果你喜欢，请为我点赞：)

- 我的微博 (<http://weibo.com/u/1670598115>)
- 个人博客 (<http://www.liangfeizc.com>)
- twitter (<https://twitter.com/JpRyouhi>)
- facebook (<https://www.facebook.com/fee.lang.zc>)



收藏(43)

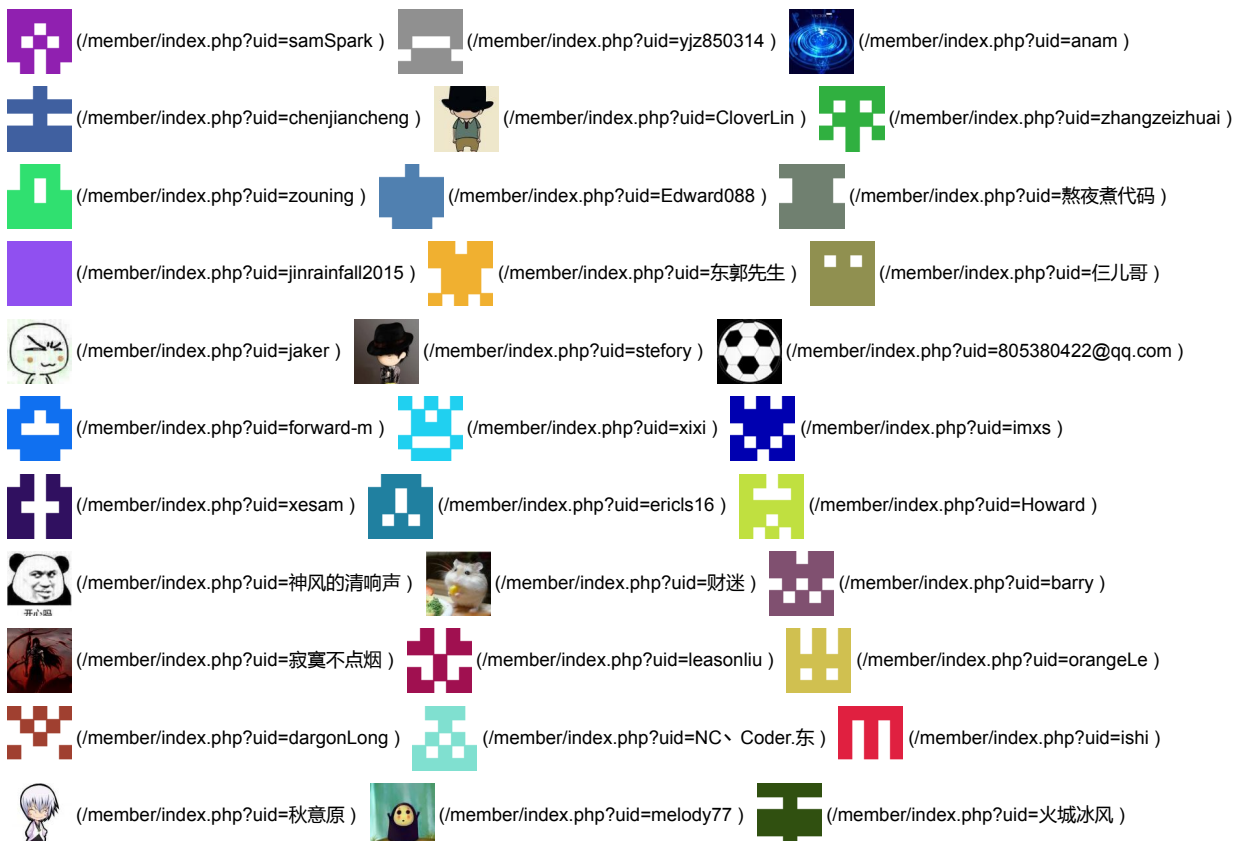


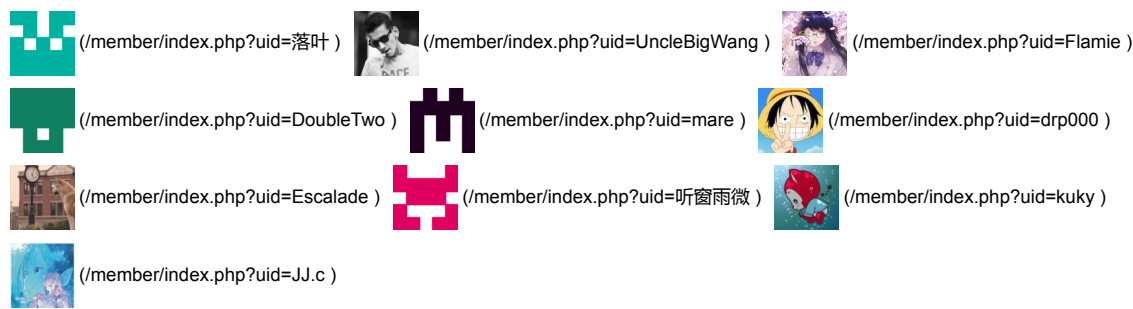
赞(304)



踩(4)

他们收藏了这篇文章





相关文章

如何在安卓数据绑定（Data Binding）的时候使用动画 (/a/anzhuokaifa/androidkaifa/2015/0602/2974.html)

2015-06-02

上一篇：[从Android的角度看Storyboard \(/a/anzhuokaifa/androidkaifa/2015/0603/2991.html\)](#)

由于先入为主的原因，Android 开发者很容易搞混某些 iOS 术语，导致理解上出现偏差。举个栗子，Objective-C 中有一个关键词 interface，虽然 Java中也有 interface，但两者的含义是完全不同的，除此之外，还有很多概念相同，但是叫法不同的术语，比如 Clos

下一篇：[Rajawali3D基础教程-一个地球旋转的例子 \(/a/anzhuokaifa/androidkaifa/2015/0604/2995.html\)](#)

这篇文章将帮助你在安卓中使用Rajawali 3D库实现一个基本的3D场景。关于最新版本的教程不是很多，有一些改动是需要注意的。在过去，Rajawali是在一个activity子类和fragment子类中渲染3D和2D场景。自从上一个官方版本0.9的发布之后，Rajawali是使用 Rajawal

发表评论

- 网友116.24.93.149 · 2016-05-29

windows一万年就有了，而且比这好用。notifyPropertyChanged纯属抄袭windows

1

46

回复
- 网友116.24.93.149 · 2016-05-29

windows一万年就有了，而且比这好用。

0

14

回复
- 网友101.41.92.232 · 2016-05-19

zero bananas  
one banana  
two bananas  
few bananas  
many bananas  
other bananas

bananaCount只对应 1 和 other，而且如果 需要在other 中 显示出具体的值需要 “%d bananas”，xml，里面需要传入两个参数 android:text="@{@plurals/banana(ban

0

1

回复

网友60.191.2.202 · 2016-05-18

感谢分享，基本概念都很清楚！

0

0

回复

网友221.218.8.88 · 2016-04-13

赞赞赞，现在还看不懂，正在慢慢学习中

1

0

回复

网友122.224.77.194 · 2015-07-16

android:visibility="@{user.isAdult ? View.VISIBLE : View.GONE}"  
为啥没有效果？

1

1




回复

网友183.60.177.228 · 2015-07-10

真棒！！

http://www.jcodecraeer.com/a/anzhuokaifa/androidkaifa/2015/0603/2992.html

10/11

 0  0  回复






**泡在网上的日子 (/member/index.php?uid=jianghejie) . 2015-06-29**

(/member/index.php?uid=jianghejie)的原帖：  
想问问找不到databinding包怎么解决

找不到databinding包怎么解决




是用的studio吗？要用现在的新api最好还是用studio好配置些

 1  1  回复



**xyp584520 (/member/index.php?uid=xyp584520) . 2015-06-29**




(/member/index.php?uid=xyp584520)想问问找不到databinding包怎么解决

 0  0  回复



**万事屋 (/member/index.php?uid=万事屋) . 2015-06-13**

(/member/index.php?uid=万事屋)Android stdio 更新蛋疼。

 0  4  回复

万事屋)

总: 2 页/14 条评论 1 2 下一页

推荐文章

- 使用ShareActionProvider分享数据 (/a/anzhuokaifa/androidkaifa/2014/1121/2029.html)
- InstaMaterial：正确处理RecyclerView动画 (/a/anzhuokaifa/androidkaifa/2015/1217/3782.html)
- Android View绘制13问13答 (/a/anzhuokaifa/androidkaifa/2016/0328/4094.html)
- InstaMaterial 概念设计(第八部分) - 拍照功能 (/a/anzhuokaifa/androidkaifa/2015/0506/2841.html)
- 你说你看到了沉浸模式，你可能只是见到鬼了 (/a/anzhuokaifa/androidkaifa/2015/0616/3047.html)
- Android一款UI体验好于NumberPicker的自定义控件NumberPickerVie (/a/anzhuokaifa/androidkaifa/2016/0629/4397.html)

赞助商

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)提供  
新浪微博 (http://weibo.com/u/2711441293) qq群一161644793 qq群二98711210  
网站地图 (/sitemap/) 网站统计 (http://tongji.baidu.com/hm-web/welcome/ico?s=2f2ac530df20294f718580cea710780e)