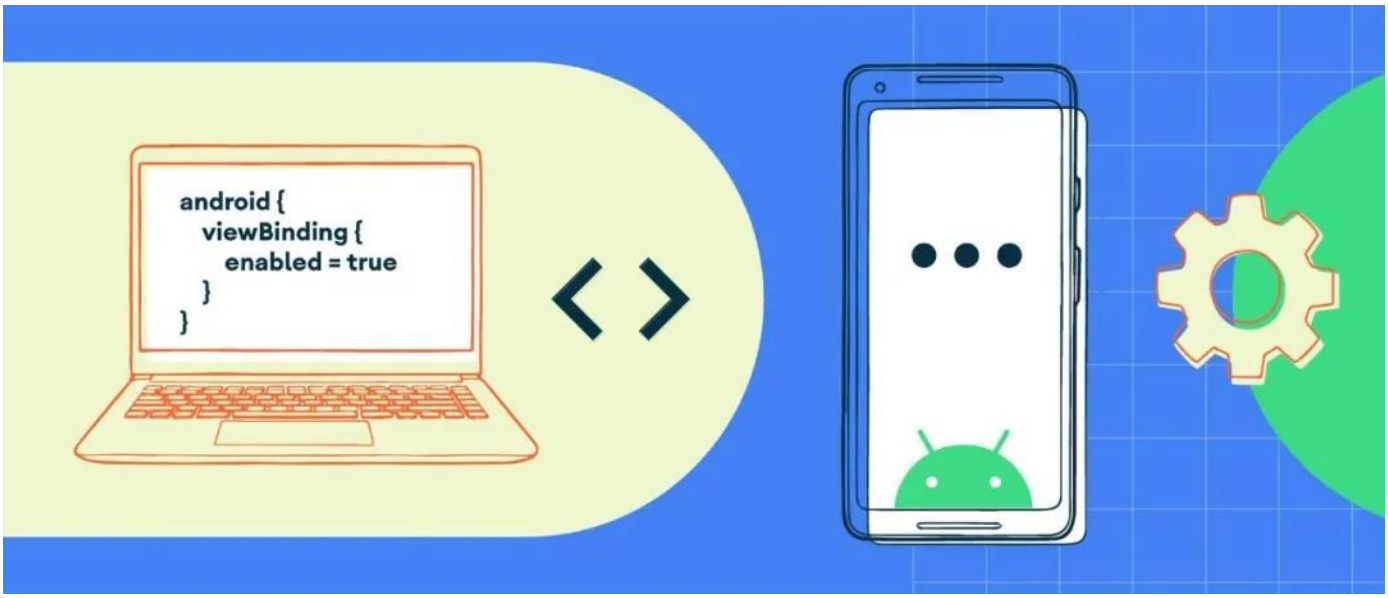
使用视图绑定替代 findViewById

谷歌开发者
已认证的官方帐号

已关注

29 人赞同了该文章



从 Android Studio 3.6 开始，视图绑定能够通过生成绑定对象来替代findViewById，从而可以帮您简化代码、移除 bug，并且从 findViewById 的模版代码中解脱出来。

本文梗概

- 在 build.gradle 中就可以方便快捷地开启视图绑定且无须额外引入依赖库
- 视图绑定会为 Module 中的每一个布局文件生成一个绑定对象 (activity_awesome.xml → ActivityAwesomeBinding.java)
- 布局文件中每一个带有 id 的视图都会在绑定对象中有一个对应的属性，这个属性将拥有正确的类型，并且空安全
- 视图绑定完美支持 Java 和 Kotlin 编程语言

腾讯视频：

使用视图绑定替代 findViewById_腾讯视频
v.qq.com

- Bilibili 视频链接

https://www.bilibili.com/video/av95393

开启视图绑定无须引入额外依赖，从 Android Studio 3.6 开始，视图绑定将会内建于 Android Gradle 插件中。需要打开视图绑定的话，只需要在build.gradle文件中配置viewBinding选项:

```
// 需要 Android Gradle Plugin 3.6.0
android {
    viewBinding {
        enabled = true
    }
}
```

在 Android Studio 4.0 中，viewBinding变成属性被整合到了buildFeatures选项中，所以配置要改成:

```
// Android Studio 4.0
android {
    buildFeatures {
        viewBinding = true
    }
}
```

配置完成后，视图绑定就会为所有布局文件自动生成对应的绑定类。无须修改原有布局的 XML 文件，视图绑定将根据您现有的布局自动完成所有工作。

> 视图绑定将会根据现有的 XML 文件，为 Module 内所有的布局文件生成绑定对象。

您可以在任何需要填充布局的地方使用绑定对象，比如Fragment、Activity、甚至是RecyclerViewAdapter（或者说是ViewHolder中）。

在 Activity 中使用视图绑定

假如您有一个布局文件名叫 activity_awesome.xml，其中包含了一个按钮和两个文本视图。视图绑定会为这个布局生成一个名叫 ActivityAwesomeBinding 的类，布局文件中所有拥有 id 的视图，都会在这个类中有一个对应的属性:

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    val binding = ActivityAwesomeBinding.inflate(layoutInflater)

    binding.title.text = "Hello"
    binding.subtext.text = "Concise, safe code"
```

△ 在 Activity 中使用视图绑定

使用视图绑定时，无须再调用findViewById方法，只要直接调用绑定对象中的对应属性即可。

布局的根视图（无论有没有 id）都会自动生成一个名为root的属性。在Activity的onCreate方法中，要将root传入setContentView方法，从而让 Activity 可以使用绑定对象中的布局。

> **一个常见的错误用法是**: 在开启了视图绑定的同时，依然在 setContentView(...) 中传入布局的 id 而不是绑定对象。这将造成同一布局被填充两次，同时监听器也会被添加到错误的布局对象中。
>
> **解决方案**: 在 Activity 中使用视图绑定时，一定要将绑定对象的 root 属性传入 setContentView() 方法中。

使用绑定对象编写安全性更佳的代码

findViewById是许多用户可见 bug 的来源: 我们很容易传入一个布局中根本不存在的 id，从而导致空指针异常而崩溃；由于此方法类型不安全，也很容易使人写出像findViewById<TextView>(R.id.image) 这样的，导致类型转换错误的代码。为了解决这些问题，视图绑定把findViewById替换成了更加简洁和安全的实现。

视图绑定有下面两个特性:

- 类型安全: 因为视图绑定总是会基于布局中的视图生成类型正确的属性。所以如果您在布局中放入了一个 TextView ，视图绑定就会暴露一个 TextView 类型的属性给您。
- 空安全: 视图绑定会检测某个视图是不是只在一些配置下存在，并依据结果生成带有 @Nullable 注解的属性。所以即使在多种配置下定义的布局文件，视图绑定依然能够保证空安全。

由于生成的绑定类是普通的 Java 类，并且其中添加了 Kotlin 友好的注解，所以 Java 和 Kotlin 都可以使用视图绑定。

视图绑定生成的代码是怎样的

如前文所说，视图绑定会生成一个包含替代 findViewById 功能的 Java 类。它会为 Module 下的每一个布局的 XML 文件生成一个对应的绑定对象，并根据源文件为其命名，比如 activity_awesome.xml 对应的绑定对象为ActivityAwesomeBinding.java。

生成代码的逻辑被优化为，当您在 Android Studio 中编辑 XML 布局文件时，只会更新所修改布局对应的绑定对象，同时这些工作会在内存中运行，从而使这个过程可以迅速完成，这意味着您的修

让我们通过一个示例 XML 布局所生成的代码，来了解一下视图绑定究竟生成了什么。

```
public final class ActivityAwesomeBinding implements ViewBinding {
  @NonNull
  private final ConstraintLayout rootView;
  @NonNull
  public final Button button;
  @NonNull
  public final TextView subtext;
  @NonNull
  public final TextView title;
```

△ 视图绑定生成的属性。可以看到它们都是类型安全以及空安全的

视图绑定会根据每个拥有 id 的视图生成类型正确的属性。他也会为根布局生成rootView属性并通过getRoot暴露给您。视图绑定没有添加任何额外的逻辑，他只是把视图属性暴露给您，从而帮您在不使用findViewById的情况下也能调用它们。这样一来便保证了生成文件简洁性（当然也避免了拖慢构建速度）。

如果您正在使用 Kotlin，视图绑定的生成类也已经对互操作进行了优化。通过@Nullable和@NonNull注解的使用，Kolin 可以正确的将属性暴露为空安全类型。如果想要了解更多关于两种语言的互操作问题，请查阅文档:在 Kotlin 中调用 Java。

```
  private ActivityAwesomeBinding(@NonNull ConstraintLayout rootView, @NonNull B
      @NonNull TextView subtext, @NonNull TextView title) { … }

  @NonNull
  public static ActivityAwesomeBinding inflate(@NonNull LayoutInflater inflater
    /* 编辑过: 移除了重载方法 inflate(inflater, parent, attachToParent) 的调用*/
    View root = inflater.inflate(R.layout.activity_awesome, null, false);
    return bind(root);
  }
```

视图绑定会生成inflate方法作为生成一个绑定对象实例的主要方式。在ActivityAwesomeBinding.java中，视图绑定生成了一个只有一个参数的inflate方法，该方法通过将parent设定为空值来指定当前视图不会绑定到父视图中；视图绑定也暴露了一个有三个参数的inflate方法，来让您在需要的时候传入parent和attachToParent参数。

真正神奇的地方是 bind 方法的调用。这里会填充视图并绑定所有的属性，同时做一些错误检测并生成清晰的错误提示。

```
    TextView subtext = rootView.findViewById(R.id.subtext);
    TextView title = rootView.findViewById(R.id.title);
    if (button != null && subtext != null && title != null) {
      return new ActivityAwesomeBinding((ConstraintLayout) rootView, button, su
    }
    throw new NullPointerException("Missing required view […]");
  }
```

△ 自动生成的 bind 方法的简化版本

bind 是绑定对象中最复杂的一个方法，它通过调用 findViewById 来绑定每个视图。既然编译器可以通过 XML 布局文件知道每个属性的类型和为空的可能性，那他就可以安全的调用 findViewById。

请注意，视图绑定生成的真正的bind方法要来的更长，并且其中使用了一个标记 break 语句来优化字节码，您可以查看Jake Wharton 撰写的这篇文章来了解更多优化有关的内容。在每个绑定对象中，都会暴露三个静态方法来创建绑定对象实例，下面是每个方法使用场景的简要说明:

- **inflate(inflater)** -- 在例如 Activity onCreate 方法里，这类没有父视图需要被传入的场合使用
- **inflate(inflater, parent, attachToParent)** -- 在 Fragment 或 RecyclerView Adapter (或者说 ViewHolder 中) ，这类您需要传递父级 ViewGroup 给绑定对象时使用。
- **bind(rootView)** -- 在您已经获得对应视图，并且只想通过视图绑定来避免使用findViewById 时使用。这个方法在使用视图绑定改造和重构现有代码时非常有用。
- 示例 XML 布局gist.github.com/objcode...
- 在 Kotlin 中调用 Javakotlinlang.org/docs/ref...
- Jake Wharton 撰写的这篇文章jakewharton.com/optimiz...

对使用 <include> 标签引入的布局会发生什么影响

前面已经讲过，视图绑定会为 Module 下的每一个布局文件生成一个绑定对象，这个说法在布局文件被另一个布局文件使用 <include> 引入时依然适用。

```
<!-- activity_awesome.xml -->
<androidx.constraintlayout.widget.ConstraintLayout>
    <include android:id="@+id/includes" layout="@layout/included_buttons"
</androidx.constraintlayout.widget.ConstraintLayout>

<!-- included_buttons.xml -->
<androidx.constraintlayout.widget.ConstraintLayout>
    <Button android:id="@+id/include_me" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

在使用引入布局的时候，视图绑定会创建一个被引入布局绑定对象的引用。注意<include>标签有一个id: android:id="@+id/includes"。这里的逻辑跟使用普通视图一样，<include>标签也需要有一个 id 才能在绑定对象中生成对应的属性。

> include 标签必须有一个 id，才能生成对应的属性。

```
public final class ActivityAwesomeBinding implements ViewBinding {
 ...

 @NonNull
 public final IncludedButtonsBinding includes;
```

视图绑定会在ActivityAwesomeBinding中生成一个IncludedButtonsBinding的引用。

结合数据绑定来使用视图绑定

视图绑定只是findViewById的取代方案，如果您希望在 XML 中自动绑定视图，可以使用数据绑定库。数据绑定和视图绑定可以生成同样的组件，它们可以同时工作。

在两者都被开启时，使用<layout>标签的布局会由数据绑定来生成绑定对象；而其余的布局则由视图绑定生成绑定对象。

> 您可以在同一 Module 中同时使用数据绑定和视图绑定。

我们之所以开发视图绑定作为数据绑定的补充，是因为许多开发者反映说，希望有一个轻量的解决方案，能在数据绑定之外替代findViewById——视图绑定提供的正是这一功能。

视图绑定对比 Kotlin 合成方法与 ButterKnife

关于视图绑定，一个最常见的问题是: "我是否应该用视图绑定替代 Kotlin 合成方法或 ButterKnife ? " 二者都是目前十分成功的组件库，有许多应用使用它们解决findViewById的问题。

对于大多数应用来说，我们推荐尝试使用视图绑定来替代这两个库，因为视图绑定可以提供更加安全和准确的视图映射方式。

△ 视图绑定空安全、只引用当前布局中的视图、支持 Java 和 Kotlin，同时也更简洁

> 虽然 ButterKnife 会在运行时校验可空与不可空，但是编译器并不会检查您匹配的视图是否在存在于您的布局之中。为了安全性与更简洁代码，我们推荐尝试使用视图绑定。

点击这里 **了解更多有关视图绑定的信息**

发布于 03-12

Android 开发　Android　Android Studio

Android中使用ViewStub提高布局性能

技术小黑屋　　发表于技术小黑屋...

如何用Android Studio 优雅的开发System App

单金伟

详解LayoutInflater.inflate()

LayoutInflater.inflate()这个方法，大家一定很熟悉——在给fragment添加布局文件，或者在RecyclerView的Adapter中为item添加布局时，都会用到。inflate()这个方法需要最多三个参数：resour...

靖然是你　　发表于靖然有这样...

4 条评论

切换为时间排序

写下你的评论...

一只少年 03-13

目前在使用kotlin进行开发，已经支持直接获取控件id，那么有必要在项目中开启viewbinding吗?

赞

Skyphone 回复 **一只少年** 03-14

完全没必要，直接用kotlin就行

1

8gd3694zgbdn1u77 04-24

我能说这东西很难用吗，一个不小心就找不到binding类

1

逝水枫华 05-06

直接迁移到kotlin即可，bonding 这东西并不好用

赞