模仿Android微信小程序，实现小程序独立任务视图的效果

原创 郭霖 郭霖 2022-07-26 08:00 发表于江苏

收录于合集

#我的原创 88 #android 8 #微信 1 #小程序 1

▲

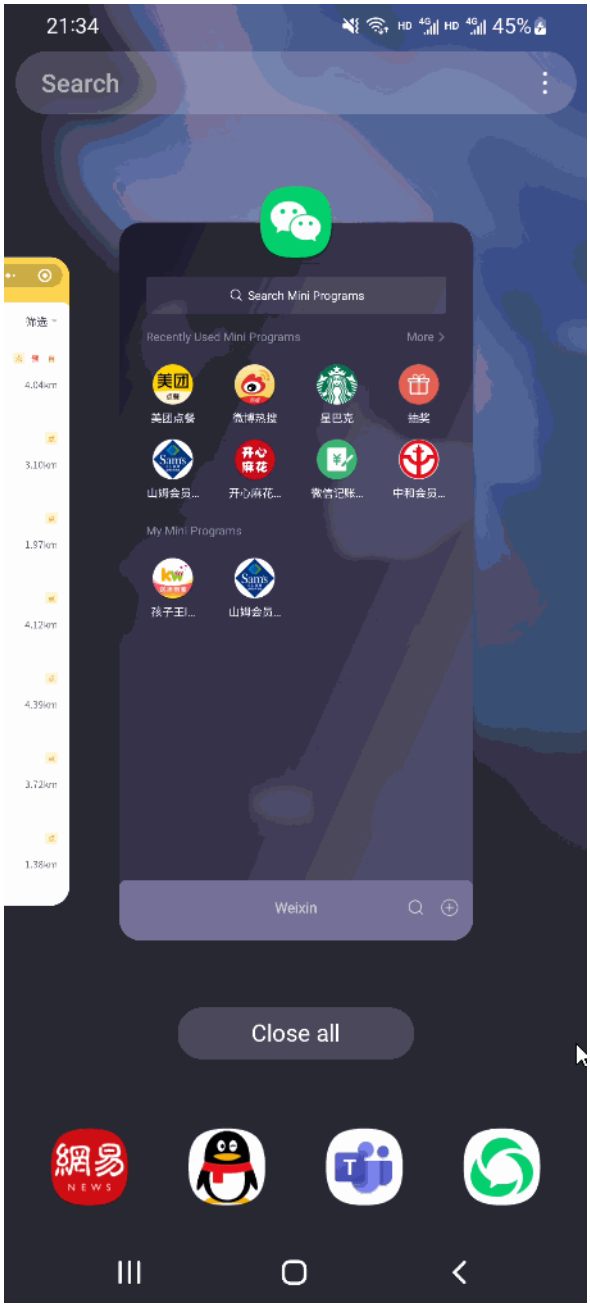
大家好，久违的原创又来了。

今天跟大家分享一个非常有趣的技术，如何在我们的App中实现类似于微信小程序的功能。

哈哈开个玩笑，如果我能徒手实现一套微信小程序系统的话，早就被腾讯挖过去当架构师了。

小程序相信现在所有人都使用过的对吧，很多人甚至天天都在使用。小程序特别的方便，无需下载，无需安装，在微信当中打开就能立刻使用。随取随用，随用随走，也不占用任何手机的存储空间。

而Android上的微信小程序做得格外的像一个真正的应用程序。为什么这么说呢？因为Android上的每个微信小程序甚至还能拥有自己的任务视图，就像是一个真正的独立应用程序一样。点击手机任务栏键可以看到如下界面：



上图中美团外卖、微博热搜、星巴克都是小程序。

拥有独立的任务视图的话，就可以更加方便地在多个小程序或微信本体之间进行快速切换，在这点上Android的体验要比iOS更好。

那么问题来了，这种依附于其他程序的小程序是如何做到拥有一个独立的任务视图的呢？

本篇文章我们就来一探究竟。

事实上，这是一个很基础的功能。有多基础呢？任何一位Android开发者在入门时都一定学过这个知识：Launch Mode。

因此，我就不在这里对Launch Mode进行展开讲解了。如果你真的从来没有听说过Launch Mode，建议参考《第一行代码 第3版》第3章的内容。

我们都知道，Android中Activity的启动模式一共有4种：standdard、singleTop、singleTask和singleInstance。

从字面意思上来看，singleTask表示的就是要启用一个单独的任务来存放当前Activity。但假如你把一个Activity声明成了singleTask，你会发现并不能得到我们想要的效果，所有的Activity仍然是放在同一个任务当中的。

这是因为，singleTask还会关联一个叫taskAffinity的属性，只有被声明成singleTask的Activity，且它的taskAffinity值也是独立的，那么这个Activity才会被放在一个单独的任务当中。

而默认情况下，每个Activity的taskAffinity属性值都是当前应用程序的包名，也就是说它们的值都是相同的，所以才不能得到我们想要的效果。

那么解决方法也很简单，给每一个要启用独立任务视图的Activity都赋值一个不同的taskAffinity值即可。

接下来我们就开始动手实践一下吧。

首先创建一个叫MiniProgramTest的项目。

接下来创建3个空的Activity，分别给它们起名为FirstActivity、SecondActivity和ThirdActivity。

然后编辑项目的activity_main.xml布局文件，在里面加入3个按钮，分别用于启动FirstActivity、SecondActivity和ThirdActivity：

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/first_btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="启动第一行代码"
        app:layout_constraintVertical_chainStyle="packed"
        app:layout_constraintBottom_toTopOf="@+id/second_btn"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/second_btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="启动第二行代码"
        app:layout_constraintVertical_chainStyle="packed"
        app:layout_constraintBottom_toTopOf="@+id/third_btn"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/first_btn" />

    <Button
        android:id="@+id/third_btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="启动第三行代码"
        app:layout_constraintVertical_chainStyle="packed"
        app:layout_constraintBottom_toBottomOf="parent"
```

```
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/second_btn" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

布局文件定义好了之后，接下来修改MainActivity的代码，加入启动逻辑：

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val firstBtn = findViewById<Button>(R.id.first_btn)
        val secondBtn = findViewById<Button>(R.id.second_btn)
        val thirdBtn = findViewById<Button>(R.id.third_btn)

        firstBtn.setOnClickListener {
            val intent = Intent(this, FirstActivity::class.java)
            startActivity(intent)
        }
        secondBtn.setOnClickListener {
            val intent = Intent(this, SecondActivity::class.java)
            startActivity(intent)
        }
        thirdBtn.setOnClickListener {
            val intent = Intent(this, ThirdActivity::class.java)
            startActivity(intent)
        }
    }
}
```

代码非常简单，点击哪个按钮就去启动相应的Activity就可以了。

但如果仅仅是这样，FirstActivity、SecondActivity和ThirdActivity一定与MainActivity是存放在同一个任务当中的。

因此下面我们就要去编写最核心的代码了，修改AndroidManifest.xml文件，如下所示：

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.miniprogramtest">

    <application
        ...>

        <activity
            android:name=".FirstActivity"
            android:exported="false"
            android:label="第一行代码"
            android:launchMode="singleTask"
            android:taskAffinity="com.example.miniprogramtest.first"
            />

        <activity
            android:name=".SecondActivity"
            android:exported="false"
            android:label="第二行代码"
            android:launchMode="singleTask"
            android:taskAffinity="com.example.miniprogramtest.second" />

        <activity
            android:name=".ThirdActivity"
            android:exported="false"
            android:label="第三行代码"
            android:launchMode="singleTask"
            android:taskAffinity="com.example.miniprogramtest.third"
            />
        ...
    </application>

</manifest>
```

可以看到，这里我们将FirstActivity、SecondActivity和ThirdActivity的launchMode都设置成了singleTask，并且给它们都指定了一个不同的taskAffinity。

现在运行一下程序，并分别点击界面上的3个按钮，然后按下手机任务栏键，我们就能看到如下效果了：

有没有觉得很神奇？明明都是同一个App中的3个Activity，现在我们竟然可以让它们在3个独立的任务视图中显示，是不是感觉就好像是微信小程序一样？

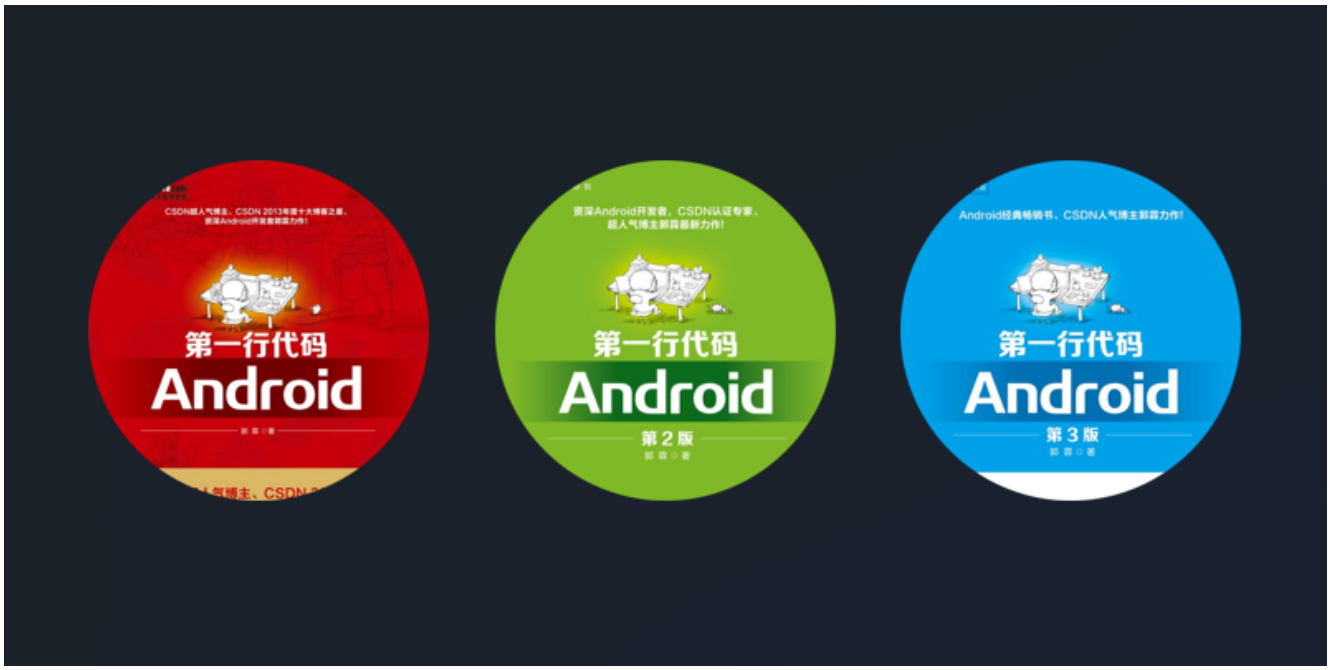
不过，虽然FirstActivity、SecondActivity和ThirdActivity都拥有独立的任务视图了，它们和微信小程序还有一个非常明显的差距。

因为每个程序都有自己专属的应用Logo，小程序也不例外。就像我们在最开始的图片中看到的一样，美团小程序有美团的Logo，微博小程序有微博的Logo，星巴克小程序有星巴克的Logo。

而目前，FirstActivity、SecondActivity和ThirdActivity显示的都是MiniProgramTest这个项目的Logo，这使得它们看上去仍然不像是一个独立的应用程序。

下面我们就开始着手优化这部分问题。

首先，这里我准备了3张图片first_line.png、second_line.png、third_line.png，分别用于作为FirstActivity、SecondActivity和ThirdActivity的Logo：



接下来，编辑FirstActivity、SecondActivity和ThirdActivity的代码，在里面加入如下逻辑：

```
class FirstActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)
```

```
        setContentView(R.layout.activity_first)
        setCustomTaskDescription()
    }

    private fun setCustomTaskDescription() {
        val taskDescription = ActivityManager.TaskDescription(
            "FirstActivity",
            BitmapFactory.decodeResource(resources, R.drawable.first_line)
        )
        setTaskDescription(taskDescription)
    }
}

class SecondActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_second)
        setCustomTaskDescription()
    }

    private fun setCustomTaskDescription() {
        val taskDescription = ActivityManager.TaskDescription(
            "SecondActivity",
            BitmapFactory.decodeResource(resources, R.drawable.second_line)
        )
        setTaskDescription(taskDescription)
    }
}

class ThirdActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_third)
        setCustomTaskDescription()
    }

    private fun setCustomTaskDescription() {
        val taskDescription = ActivityManager.TaskDescription(
            "ThirdActivity",
            BitmapFactory.decodeResource(resources, R.drawable.third_line)
        )
        setTaskDescription(taskDescription)
    }
}
```

这3段代码的逻辑基本都是相同的。

核心部分就是调用了setCustomTaskDescription()方法来给当前Activity设置一个自定义的TaskDescription。

所谓TaskDescription就是给当前的任务设置一个描述，描述中可以包含任务的名称和图标。

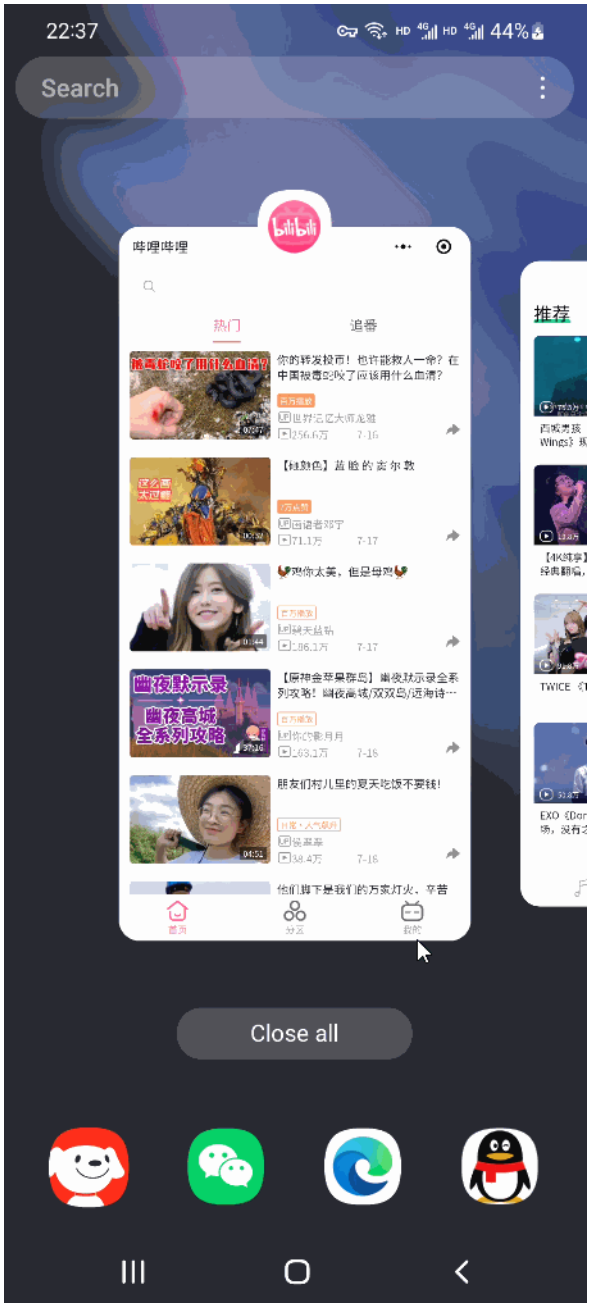
那么这里我们给FirstActivity、SecondActivity和ThirdActivity分别设置了不同的TaskDescription，这样在任务视图当中，就可以看到各不相同的应用Logo了，如下图所示：

其实到这里为止，我们就把微信小程序的外壳搭建得差不多了。剩下的部分，当然也是最难的部分，就是在这个壳子里面添加小程序的内容了。这部分的技术以前端为主，并不是我擅长的领域，我也讲不了，因此就不再继续向下延伸了。

不过或许还有些朋友会存在这样的疑惑：目前我们的技术实现方案是给每个小程序定义一个单独的Activity（FirstActivity、SecondActivity和ThirdActivity），而微信小程序却可以有无限多个，我们显然不可能在AndroidManifest.xml文件中注册无限个Activity，那么微信又是如何实现的呢？

其实这只是一个美丽的误会，因为微信小程序并不是可以有无限多个，只是你平时没有注意这个小细节而已。

我们通过做个实验来验证一下吧，观察下图中的效果：



可以看到，这里我事先依次按照顺序打开了哔哩哔哩、QQ音乐、微博热搜、京东购物、星巴克，这5个小程序。

这个时候回到微信当中，再打开一个顺丰速运小程序。

再次回到任务视图列表界面，你会发现现在多了一个顺丰速运的小程序，而最早打开的哔哩哔哩小程序却从任务视图列表中消失不见了。

由此可以看出，微信其实在AndroidManifest.xml文件中也只是放置了5个占位的Activity。当你尝试打开第6个小程序时，最先打开的那个小程序就会被回收，将它的容器提供给第6个小程序使用。

好了，本篇文章到这里就结束了。内容其实非常的简单，但是已经把在Android上如何实现小程序外层的架子讲明白了。至于如何实现小程序最核心的内容部分，那就要看各位架构师的水准了。

我们下期再见。

推荐阅读：

我的新书，《第一行代码 第3版》已出版！

再学一遍android:fitsSystemWindows属性

一个Android沉浸式状态栏上的黑科技

收录于合集 #我的原创 88

上一篇
Android 13运行时权限变更一览

下一篇
一个Android沉浸式状态栏上的黑科技

阅读原文

喜欢此内容的人还喜欢

SpringBoot接入轻量级分布式日志框架GrayLog

码猿技术专栏

详解微服务编排

51CTO技术栈

一篇文章带你了解 Android 国际化

是刘航啊