Kotlin 写 Android 单元测试（四），Robolectric 在 JVM 上测试安卓相关代码


johnnyshieh.me/posts/unit-test-robolectric-android-on-jvm

Kotlin 写 Android 单元测试系列文章：

Kotlin 写 Android 单元测试（一），单元测试是什么以及为什么需要

Kotlin 写 Android 单元测试（二），JUnit 4 测试框架和 kotlin.test 库的使用

Kotlin 写 Android 单元测试（三），Mockito mocking 框架的使用

Kotlin 写 Android 单元测试（四），Robolectric 在 JVM 上测试安卓相关代码

未完待续...

通过前面几篇文章，我们知道可以使用 JUnit 4 和 Mockito 测试框架来测试纯 Java 业务逻辑，但是无法在 JVM 上测试 Android 相关代码。因为 Android 代码需要运行在 Android 平台的虚拟机 Dalvik 或 ART 上，不能直接在 Java 虚拟机（JVM）上直接运行。而我们用 Android Studio 编写 Android 代码时只需要下载 JDK 和 Android SDK，在项目的 `External Libraries` 可以看到编译所需要的 Android API，实际上就是 android.jar，这样 Android 代码就能正常开发编译了。

但是 android.jar 里的类只是个壳，里面的方法都是 `throw RuntimeException("stub!!");`，所以在 Android Studio 中可以正常开发编译，但是在 JDK 中 JVM 下运行的话会抛 RuntimeException。我们的 app 代码能在 Dalvik 或 ART 上运行，可以运行是因为它们把 android.jar 里面替换为 Android 的系统实现，所以才能正常运行。

这种情况如何做 Android 代码的单元测试呢？一种方式是使用 Android 官方提供的 Instrumentation 框架，不过测试代码还是不能在 JVM 上运行，只能在模拟器或者真机上运行，这种方式相当于编一个测试版的 apk，传到模拟器或者真机上再运行，显然速度不可能快，不方便做单元测试。那有没有什么方式可以直接 Android Studio 的开发环境 JVM 中运行 Android 代码呢，Robolectric 框架就可以解决这个问题。

> 本文是基于 Robolectric 3.5.1，Android Studio 3.0 环境

1. Robolectric

官网：http://robolectric.org/

Robolectric 重新实现了 android.jar，使得 Android 相关的测试代码都可以直接在 JVM 上运行。对于 Android 中的类 `XXX`，它们都实现了 `ShadowXXX`，例如 ShadowActivity、ShadowView 等，在调用 Activity 代码，Robolectric 会拦截并实际上调用 ShadowActivity

的同名方法。而且 Robolectric 还处理了布局加载、资源加载等 Android 运行需要的东西，可以在 JVM 中测试 Android 代码在真机上运行的样子。

1.1 Gradle 引入

```
testImplementation 'junit:junit:4.12'
testImplementation 'org.robolectric:robolectric:3.5.1'
// 如果用到 multidex 和 support-v4 包的话，还需要引入 robolectric 对应
的模块
testImplementation 'org.robolectric:shadows-multidex:3.5.1'
testImplementation 'org.robolectric:shadows-supportv4:3.5.1'
android {
  testOptions {
    unitTests {
      includeAndroidResources = true
    }
  }
}
```

注意：Robolectric 3.3 以上的版本都需要 Android Studio 的版本在 3.0 以上。

如果是 Linux 或者 Mac 用户，还需要在 Run -> Edit Configuration... 的窗口中，在左侧边栏选择 Defaults -> Android JUnit，然后将右侧的 working directory 值改为 `$MODULE_DIR$` 。

1.2 测试实例

下面这个 Activity 点击按钮后会跳转到登录页。

```
class WelcomeActivity : Activity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.welcome_activity)

        findViewById(R.id.login).setOnClickListener { startActivity(Intent(this,
LoginActivity::class.java)) }
    }
}
```

再看下测试代码：

```
@RunWith(RobolectricTestRunner::class)
class WelcomeActivityTest {
    @Test
    fun clickLogin() {
        val activity =
Robolectric.setupActivity(WelcomeActivity::class.java)
        activity.findViewById(R.id.login).performClick()

        val expectedIntent = Intent(activity, LoginActivity::class.java)
        val actualIntent =
ShadowApplication.getInstance().nextStartedActivity
        assertEquals(expectedIntent.component,
actualIntent.component)
    }
}
```

上面测试代码中 `@RunWith(RobolectricTestRunner::class)` 保证 Robolectric 框架生效，这样才能在调用 Android 代码时转到 Robolectric 的 Shadow 实现。而且 Robolectric 会在测试框架执行一开始创建 application 实例，在 AndroidManifest.xml 中定义的 Application 类也会被反射创建实例，并执行 onCreate 生命周期。

`ShadowApplication.getInstance().nextStartedActivity` 是 Robolectric 提供方便测试的方法，可以获取最近一个启动的 Activity 的 Intent。

1.3 测试版的 Application

Robolectric 会自动识别出 AndroidManifest.xml 中定义的 Application 类，并且初始化，但是我们在做单元测试的时候，只需要测试项目代码的逻辑，不测试第三方库，例如网络请求、图片加载、数据库读写。我们测试的是调用第三方库时，所传递的参数是否符合预期，所以不需要初始化第三方库，而且 Robolectric 不支持加载第三方 native 库。

写单元测试时，有时需要使用 Mockito 框架 Mock 出一些对象，如果使用 Dagger 2 依赖注入框架的话，最好直接把 Module 里的依赖对象换成 Mock 对象，这时就需要改写 Application 的初始化逻辑。

值得庆幸的是，**Robolectric 支持测试版的 Application**，方法也很简单，所以可以用测试版的 Application 来修改初始化逻辑。如果在 AndroidManifest.xml 中定义的为 MyApplication，那么在测试文件下，同样包名，新建一个加上 `Test` 前缀的 Application 即可：

```
class TestMyApplication : MyApplication(),
TestLifecycleApplication {
    override fun onCreate() {
        ...
    }
    override fun beforeTest(method: Method) {
    }
    override fun prepareTest(test: Any) {
    }
    override fun afterTest(method: Method) {
    }
}
```

这样 Robolectric 就好创建 TestMyApplication 实例作为应用的 Application。

2. 小结

总的来说，Robolectric 让我们很方便地测试 Android 相关代码，不过在一开始时使用时会遇到一些问题，坚持一下就过去了。使用过程中遇到什么问题，可以在网络下搜索下，或者断点看源码，当然可以在文章下面留言。我在使用过程就遇到一个问题：Application 实例创建后，Robolectric 会先注册广播监听器，然后在调用 Application 的 onCreate 方法，因为项目中的广播监听器初始化时用到了 onCreate 执行后才会生成的属性，结果就抛出空异常。

到目前为主，已经介绍了大体上在 Kotlin 下写 Android 单元测试需要用到的 JUnit、Mockito、Roblectric 测试框架，之后会写一些实际项目过程中单元测试的常见问题的解决方案。