安卓单元测试(八)：Junit Rule的使用

chriszou.com/2016/07/09/junit-rule

July 9, 2016

JUnit Rule是什么

一个JUnit Rule就是一个实现了 `TestRule` 的类，这些类的作用类似于 `@Before` 、 `@After` ，是用来在每个测试方法的执行前后执行一些代码的一个方法。如果你不清楚 `@Before` 、 `@After` 这些Annotation的意思，Chances are你还不了解Junit的使用，建议先看这篇文章。
那为什么不直接用这些annotation呢？这是因为它们都只能作用于一个类，如果同一个setup需要在两个类里面同时使用，那么你就要在两个测试类里面定义相同的 `@Before` 方法，然后里面写相同的代码，这就造成了代码重复。有的人说你可以用继承啊，首先我想说，我很讨厌继承这个东西，所以如果可以不用继承的话，我就不会用；再次我想说，如果你不讨厌继承的话，从现在开始，你也应该慢慢的讨厌它了。
此外，JUnit Rule还能做一些 `@Before` 这些Annotation做不到的事情，那就是他们可以动态的获取将要运行的测试类、测试方法的信息。这个在接下来的一个例子里面可以看到。

怎么用JUnit Rule？

使用框架自带的Rule

很多测试框架比如JUnit、Mockito自带给我们很多已经实现过好了的JUnit Rule，我们可以直接拿来用。比如 `Timeout` ， `TemporaryFolder` ，等等。这些Rule的使用方法非常简单。定义一个这些类的public field，然后用 `@Rule` 修饰一下就好了。比如

```
public class ExampleTest {
   @Rule
   public Timeout timeout = new Timeout(1000);  //使用Timeout这个 Rule，
   @Test
   public void testMethod1() throws Exception {
      //your tests
   }
   @Test
   public void testMethod2() throws Exception {
      //your tests2
   }
   //other test methods
}
```

那么，对于上面这个 `ExampleTest` 的每一个测试方法。它们的运行时间都不能超过1秒钟，不然就会标志为失败。而它的实现方式就是在每个方法测试之前都会记录一下时间戳，然后开始倒计时，1秒钟之后如果方法还没有运行结束，就把结果标记为失败。
这里需要注意的一点是Rule需要是**public** field

实现自己的Rule

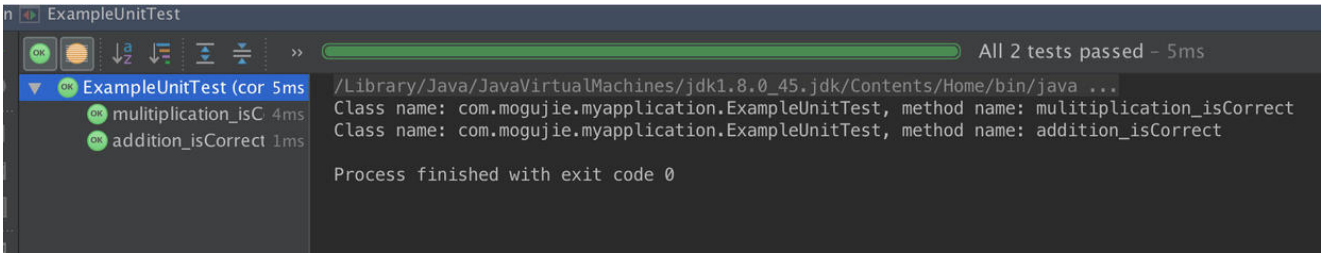
当然，如果只能用框架自带的Rule，那功能未免太受限了，JUnit Rule最强大的地方在于，我们可以自己写满足我们自己需要的Rule。所以现在的问题是怎么写这个Rule。简单来说，写一个Rule就是implement一个 `TestRule` interface，实现一个叫 `apply()` 的方法。这个方法需要返回一个 `Statement` 对象。下面给一个例子，这个 Rule的作用是，在测试方法运行之前，记录测试方法所在的类名和方法名，然后在测试方法运行之后打印出来，至于怎么在测试方法运行前后做这些事情，下面例子中的注释里面说的很清楚。

```
public class MethodNameExample implements TestRule {
    @Override
    public Statement apply(final Statement base, final Description description) {
        return new Statement() {
            @Override
            public void evaluate() throws Throwable {
                //想要在测试方法运行之前做一些事情，就在base.evaluate()之前做
                String className = description.getClassName();
                String methodName = description.getMethodName();
                base.evaluate();  //这其实就是运行测试方法
                //想要在测试方法运行之后做一些事情，就在base.evaluate()之后做
                System.out.println("Class name: "+className +", method name: "+methodName);
            }
        };
    }
}
```

这个Rule这样就算写好了，现在来试试，用这个 Rule的方法跟使用自带的Rule的用法是一样的，写一个public field，用 `@Rule` 修饰一下就好了。

```
public class ExampleUnitTest {
    @Rule
    public MethodNameExample methodNameExample = new MethodNameExample();
    @Test
    public void addition_isCorrect() throws Exception {
        assertEquals(4, 2 + 2);
    }
    @Test
    public void mulitiplication_isCorrect() throws Exception {
        assertEquals(4, 2 * 2);
    }
}
```

运行结果如下：



在右边的框框可以看到，把测试方法的方法名和所在的类名打印出来了。
上面的例子对于TestRule的实现应该说的比较清楚，但是看起来没多大用。下面给另外的一个例子，大家或许会觉得这个东西更有用一点。在安卓里面，我们经常在很多地方需要用到 `Context` 这个东西。我们的做法是将这个东西保存在一个类里面，作为静态变量存在：

```
public class ContextHolder {
    private static Context sContext;
    public static void set(Context context) {
        sContext = context;
    }
    public static Context get() {
        return sContext;
    }
}
```

然后在自定义的 `Application#onCreate()` 里面调一下 `ContextHolder.set()` 将这个context初始化。
如果你的当前项目是一个library，那么你在测试环境下是没有这个 `Application` 的，Robolectric会给你造一个Application，放在 `RuntimeEnvironment.application` 里面。所以在测试环境下你可以使用这个instance来将 `ContextHolder` 初始化。在这种情况下，你就可以用一个Rule来实现这样的效果，在用到 `Context` 的测试方法运行之前将 `ContextHolder` 初始化：

```
public class ContextRule implements TestRule {
    @Override
    public Statement apply(Statement base, Description description) {
        ContextHolder.set(RuntimeEnvironment.application);
        return base;
    }
}
```

这样，在运行一些用到context的测试方法之前，你就可以使用这个Rule来给Context赋值了。
其实，最常用的Rule之一就是结合 `@Mock` 之类的Annotation快速的创建mock，但是这点我想作为一篇单独的文章写一下。原因之一是因为它涉及到的东西不仅仅是Rule，还有其它的一些东西。更重要的原因是，我希望大家能知道这个东西，而不是被这篇文章淹没。请关注下一篇文章吧！

小结

JUnit Rule的介绍就到这里，应该说比较简单，却是非常有帮助。希望这篇文章能帮助到大家了解这个东西。
这篇文章的代码放在这个github repo里面。
如果你对安卓单元测试感兴趣，欢迎加入我们的交流群，因为群成员超过100人，没办法扫码加入，请关注公众号获取加入方法。