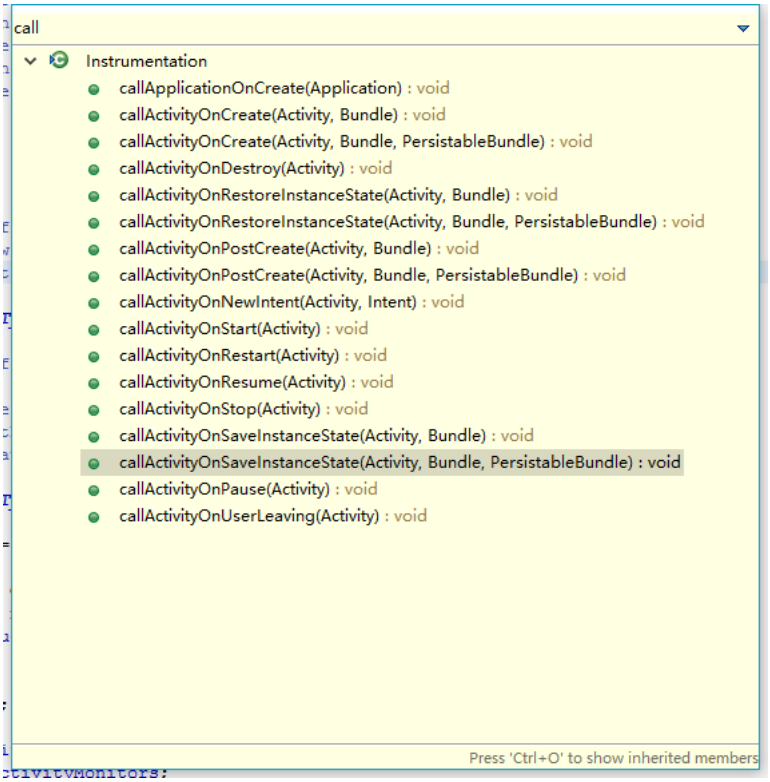
测试基础 Android Instrumentation 框架简单说明

土拨鼠 · 2016年12月06日 · 最后由 Hi Hydra 回复于 2018年08月29日 · 4827 次阅读

本帖已被设为精华帖!

目录

提到 android 自动化测试的时候经常会提到Instrumentation，但实际上 Instrumentation 是什么呢，很多人可能认为 Instrumentation 就是 android 的测试框架，实际上当启动一个 app 的时候都会实例化一个 Instrumentation 对象，且 Instrumentation 在每个 Activity 跳转的时候都会用到且其内部类 ActivityMonitor 会监控 activity 的，，只是我们不直接使用它；另外 Activity 的生命周期方法也是通过它来调用的：



在自动化测试过程中我们不是直接使用 Instrumentation 而且使用其子类 InstrumentationTestRunner，在测试工程的 AndroidManiFest.xml 里面配置如下：

```
<instrumentation
    android:name="android.test.InstrumentationTestRunner"
    android:label="InstrumentationApp"
    android:targetPackage="com.instrumentation.app" >
</instrumentation>
```

上面两种方式在应用启动的时候初始化的 Instrumentation 对象是不同的；点击 app 图标启动 app 初始化的是默认值 Instrumentation 的对象，通过**adb shell instrument**方式启动 app 的时候初始化的是 AndroidManiFest.xml 里面配置的 InstrumentationTestRunner 对象，以上两种初始化方式都可以在阅读 android 源码的时候看到，这里提供一种直观简单的方式来验证我们的猜想。

获取 Instrumentation 对象

查看 Activity.java 源码可知在 Activity 中存在 Instrumentation 的成员变量：

```
private Instrumentation mInstrumentation;
```

所以我们大致步骤就是通过反射获取 mInstrumentation 成员变量。
首先新建一个 Android project， 包名这里是：com.instrumentation.app，创建一个名为 MainActivity 的 Activity，

```
package com.instrumentation.app;

import java.lang.reflect.Field;
import android.app.Activity;
import android.util.Log;

public class MainActivity extends Activity {
    private String LOG_STR = "debug";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Field mInstrumentation;
        Object value = null;
        try {
            mInstrumentation = this.getClass().getSuperclass().getDeclaredField("mInstrumentation");
            mInstrumentation.setAccessible(true);
            value = mInstrumentation.get(this);
        } catch (NoSuchFieldException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        } catch (IllegalArgumentException e) {
            e.printStackTrace();
        }
        Log.d(LOG_STR, "Instrumentation: "+value.getClass().getName());
    }
}
```

打包工程安装到设备，点击 app 图标打开此时查看 ddms 的 log 显示：

```
debug            Instrumentation: android.app.Instrumentation
```

此时显示的是默认的 Instrumentation 对象，当我们在此工程的 AndroidManiFest.xml 中配置如下：

```
<instrumentation
    android:name="android.test.InstrumentationTestRunner"
    android:label="InstrumentationApp"
    android:targetPackage="com.instrumentation.app" >
</instrumentation>
```

此工程中新建一个测试类：AppDemoTest

```
import android.test.ActivityInstrumentationTestCase2;

public class AppDemoTest extends ActivityInstrumentationTestCase2<MainActivity>{
    public AppDemoTest() {
        super(MainActivity.class);
    }

    public void testApp(){
        getActivity();
        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

打包工程安装到设备，通过命令行启动 case:

> adb shell am instrument -e class com.instrumentation.app/AppDemoTest
> com.instrumentation.app/android.test.InstrumentationTestRunner

查看 ddms 的 log 显示:

```
debug            Instrumentation: android.test.InstrumentationTestRunner
```

另外对比 InstrumentationTestRunner 可以发现其主要实现了 Instrumentation 的 onCreate() 和 onStart() 方法，用于解析命令行传入的参数和执行 case。所以当我们想实际测试报告的输出也只要继续扩展 InstrumentationTestRunner 就可以了。

扩展：关于 robotium 框架对 Instrumentation 的利用

通过 robotium 获取当前的 Activity 对象

在看 robotium 框架源码之前需要分析 Instrumentation 启动 activity 的过程：

```
public void addMonitor(ActivityMonitor monitor) {
    synchronized (mSync) {
        if (mActivityMonitors == null) {
            mActivityMonitors = new ArrayList();
        }
        mActivityMonitors.add(monitor);
    }
}

public ActivityResult execStartActivity(
        Context who, IBinder contextThread, IBinder token, Activity target,
        Intent intent, int requestCode, Bundle options) {
    IApplicationThread whoThread = (IApplicationThread) contextThread;
    if (mActivityMonitors != null) {
        synchronized (mSync) {
            final int N = mActivityMonitors.size();
            for (int i=0; i<N; i++) {
                final ActivityMonitor am = mActivityMonitors.get(i);
                if (am.match(who, null, intent)) {
                    am.mHits++;
                    if (am.isBlocking()) {
                        return requestCode >= 0 ? am.getResult() : null;
                    }
                    break;
                }
            }
        }
    }
    try {
        intent.migrateExtraStreamToClipData();
        intent.prepareToLeaveProcess();
        int result = ActivityManagerNative.getDefault()
            .startActivity(whoThread, who.getBasePackageName(), intent,
                    intent.resolveTypeIfNeeded(who.getContentResolver()),
                    token, target != null ? target.mEmbeddedID : null,
                    requestCode, 0, null, options);
        checkStartActivityResult(result, intent);
    } catch (RemoteException e) {
    }
    return null;
}
public void callActivityOnResume(Activity activity) {
    activity.mResumed = true;
    activity.onResume();

    if (mActivityMonitors != null) {
        synchronized (mSync) {
            final int N = mActivityMonitors.size();
            for (int i=0; i<N; i++) {
                final ActivityMonitor am = mActivityMonitors.get(i);
                am.match(activity, activity, activity.getIntent());
            }
        }
    }
}
```

以上通过 addMonitor() 方法将指定的 ActivityMonitor 对象添加到 list 中，在启动一个 activity 的过程中（代码仅供参考，实际流程复杂的多）会通过 for 循环对列表所有的 ActivityMonitor 对象调用 match 方法对其成员变量进行赋值操作，因此当此时调用 ActivityMonitor 的 getLastActivity 方

法就可以获取刚启动的 activity 对象；
再看 Robotium 的 solo.getCurrentActivity() 最终实际调用在 robotium 框架中的 ActivityUtils 的 getCurrentActivity(true, true);

```
/**
 * Returns the current {@code Activity}.
 *
 * @param shouldSleepFirst whether to sleep a default pause first
 * @param waitForActivity whether to wait for the activity
 * @return the current {@code Activity}
 */

public Activity getCurrentActivity(boolean shouldSleepFirst, boolean waitForActivity) {
    if(shouldSleepFirst){
        sleeper.sleep();
    }
    if(!config.trackActivities){
        return activity;
    }

    if(waitForActivity){
        waitForActivityIfNotAvailable();
    }
    if(!activityStack.isEmpty()){
        activity=activityStack.peek().get();
    }
    return activity;
}

/**
 * Waits for an activity to be started if one is not provided
 * by the constructor.
 */

private final void waitForActivityIfNotAvailable(){
    if(activityStack.isEmpty() || activityStack.peek().get() == null){

        if (activityMonitor != null) {
            Activity activity = activityMonitor.getLastActivity();
            while (activity == null){
                sleeper.sleepMini();
                activity = activityMonitor.getLastActivity();
            }
            addActivityToStack(activity);
        }
        else if(config.trackActivities){
            sleeper.sleepMini();
            setupActivityMonitor();
            waitForActivityIfNotAvailable();
        }
    }
}

/**
 * This is were the activityMonitor is set up. The monitor will keep check
 * for the currently active activity.
 */

private void setupActivityMonitor() {
    if(config.trackActivities){
        try {
            IntentFilter filter = null;
            activityMonitor = inst.addMonitor(filter, null, false);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

大概流程（不画图了

）

- robotium 中通过 Instrumentation 对象初始化一个 ActivityMonitor 对象，activityMonitor = inst.addMonitor(filter, null, false);
- 一旦有 activity 启动则 activityMonitor 就会调用 match 方法给其成员变量赋值，此时调用 activityMonitor.getLastActivity() 方法获取最新的 activity 对象并保存到 Stack 中；
- 调用 ActivityUtils 的 getCurrentActivity(true, true)，从取出栈顶的对象即为最新的 activity 对象；

扩展：关于 Instrumentation 对 uiautomator 的利用

Instrumentation 是无法跨应用的，因此跨应用方案会单独采用 uiautomator 框架来做，但是 Google 在 API>=18 中通过 Instrumentation 提供获取 UiAutomation 对象来进行跨应用操作：

```
public UiAutomation getUiAutomation() {
    if (mUiAutomationConnection != null) {
        if (mUiAutomation == null) {
            mUiAutomation = new UiAutomation(getTargetContext().getMainLooper(),
                    mUiAutomationConnection);
            mUiAutomation.connect();
        }
        return mUiAutomation;
    }
    return null;
}
```

示例：Instrumentation 中调用 uiautomation 对象进行跨应用操作，回到桌面点击 “设置” 进入设置界面。

```
public void testApp(){
    getActivity();
    try {
        Thread.sleep(5000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    UiAutomation uiAutomation = getInstrumentation().getUiAutomation();
    //回到桌面
    uiAutomation.performGlobalAction(AccessibilityService.GLOBAL_ACTION_HOME);
    try {
        Thread.sleep(3000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    //获取当前界面的顶层AccessibilityNode信息
    AccessibilityNodeInfo accessibilityNodeInfo = uiAutomation.getRootInActiveWindow();
    List<AccessibilityNodeInfo> list = accessibilityNodeInfo.findAccessibilityNodeInfosByText("设置");
    int size = list.size();
    Log.d("debug", "size: "+size);
    if(size != 0){
        //获取“设置”控件信息
        AccessibilityNodeInfo settingAccessibilityNodeInfo = list.get(0);
        //执行点击操作
        settingAccessibilityNodeInfo.performAction(AccessibilityNodeInfo.ACTION_CLICK);
    }
    try {
        Thread.sleep(3000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

扩展：解决通过 Instrumentation 进行自动化测试依赖被测应用权限问题

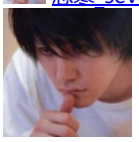
主要是利用 AIDL 提供远程调用，具体源码参考：https://github.com/hao-shen/AndToolsTest
「原创声明：保留所有权利，禁止转载」
17 个赞
共收到 **7** 条回复 时间 点赞

EndTest #1 · 2016年12月06日

写的很好。

喜力 #2 · 2016年12月06日

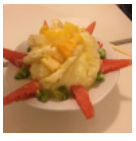

思寒_seveniruby 将本帖设为了精华贴 12月06日 17:18


思寒_seveniruby #4 · 2016年12月06日

加精理由: 提供了对 android 基本原理的解释. 对 instrumentation 的理解也很正确. 这是优秀工程师的基本功. 在源码分析上体现了作者优秀的探索精神.


testly #5 · 2016年12月06日

感谢分享。受教了



Michael_Wang #6 · 2016年12月06日

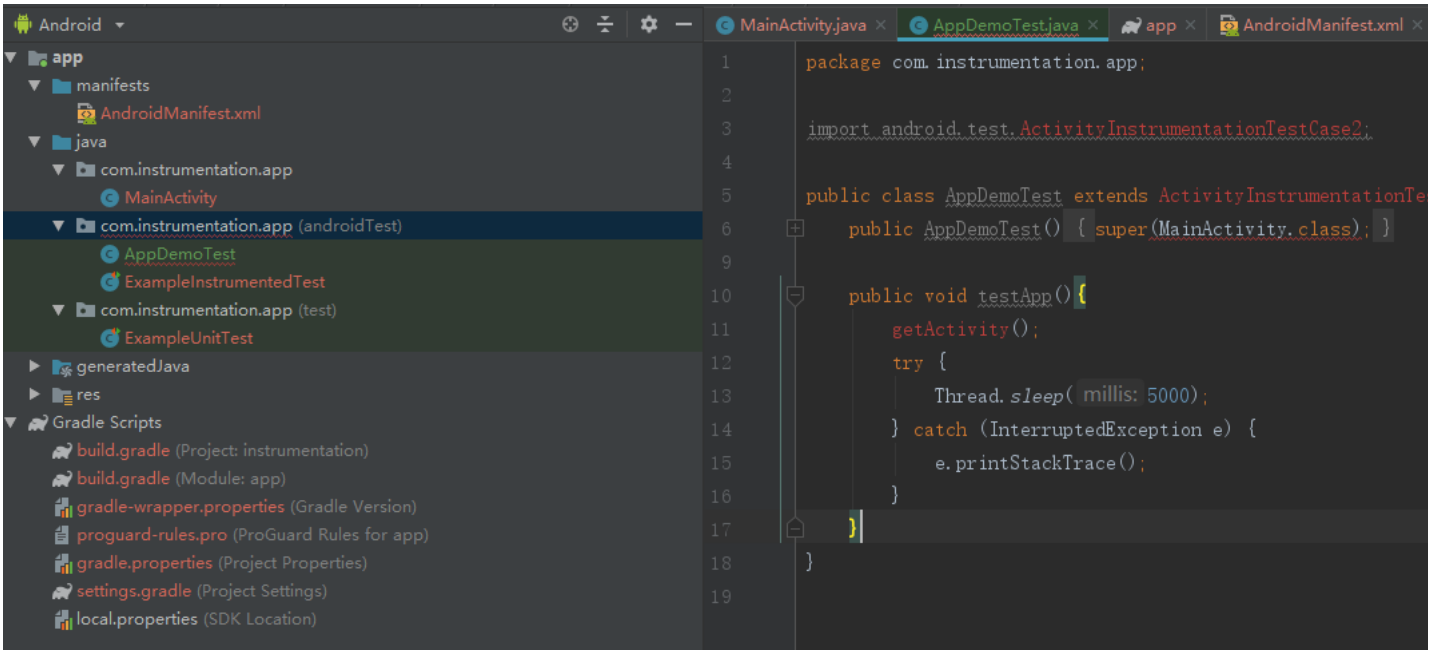
厉害

扫地僧 #7 · 2016年12月06日

感谢分享朴实无华的好文

Hi Hydra #8 · 2018年08月29日

新人路过, 代码复制过来为什么报错啊, 那个`ActivityInstrumentationTestCase2`导不进去, 不知道怎么回事, 求大神指导



需要 登录 后方可回复, 如果你还没有账号请点击这里 注册。

土拨鼠
@haos

失业

17 个赞

共收到 **7** 条回复

有新回复！ 点击这里立即载入

关于 / 活跃用户 / 中国移动互联网测试技术大会 / 反馈 / Github / API / 帮助推广

TesterHome社区，测试之家，由众多测试工程师组织和维护的技术社区，致力于帮助新人成长，提高测试地位，推进质量发展。Inspired by RubyChina

友情链接 WeTest腾讯质量开放平台 / InfoQ / 掘金 / SegmentFault / 测试窝 / 百度测试吧 / IT大咖说

简体中文 / 正體中文 / English