

UI Automator

本章大纲

- UI Automator 介绍
- UI Automator 环境搭建
- UI Automator 主要的对象类

UIAutomator 介绍

UIAutomator是Google提供的自动化测试框架，可以通过它来编写UI自动化测试用例。

- `ui automator viewer` - 一个图形界面工具来扫描和分析应用的UI控件。
- `ui automator` - 一个测试的Java库，包含了创建UI测试的各种API和执行自动化测试的引擎。支持所有的Android事件操作，可以通过断言和截图验证正确性。

UIAutomator 介绍



Developers

Develop > Testing Support Library

Developer Console

Tools Help

Build System

Performance Tools

Testing Tools

Testing Support
Library

API Reference

monkey

monkeyrunner

Support Library

Data Binding Library

Revisions

NDK

test actions and assertions run more reliably.

To learn more about using Espresso, see the [API reference](#) and [Testing UI for a Single App](#) training.

UI Automator

The UI Automator testing framework provides a set of APIs to build UI tests that perform interactions on user apps and system apps. The UI Automator APIs allows you to perform operations such as opening the Settings menu or the app launcher in a test device. The UI Automator testing framework is well-suited for writing *black box*-style automated tests, where the test code does not rely on internal implementation details of the target app.

The key features of the UI Automator testing framework include:

- A viewer to inspect layout hierarchy. For more information, see [UI Automator Viewer](#).
- An API to retrieve state information and perform operations on the target device. For more information, see [Access to device state](#).
- APIs that support cross-app UI testing. For more information, see [UI Automator APIs](#).

Requires Android 4.3 (API level 18) or higher.

UI Automator Viewer

The `uiautomatorviewer` tool provides a convenient GUI to scan and analyze the UI components currently displayed on an Android device. You can use this tool to inspect the layout hierarchy and view the properties of UI components that

UIAutomator 介绍

UI Automator测试框架提供了一组api来构建UI测试执行用户程序和系统程序交互。UI Automator api允许执行操作，如打开设置菜单或在测试设备应用程序启动器。UI Automator测试框架非常适合写黑盒自动化测试，在测试代码不依赖于目标应用程序的内部实现细节。

UI Automator 特点

- 优点:

- ✓ 支持跨应用

- ✓ 可以对所有操作进行自动化，操作简单；

- ✓ 不需要对被测程序进行重签名，可以测试所有设备上的程序，比如某APP，拨号，发信息等等

- ✓ 对于控件定位，要比robotium简单一点点

- 缺点:

- ✓ 权限较低，无法像Instrumentation一样获取应用的较高权限

- ✓ uiautomator需要android level 18以上才可以使用，因为在level 18及以上的API里面才带有uiautomator工具

- ✓ 如果想要使用resource-id定位控件，则需要level 18及以上才可以

UI Automator 工作流程

1. 安装要测试的应用到手机中，分析应用的UI界面元素 并确保被测试应用的各个控件可以被测试工具获取到。
2. 创建知道测试案例来模拟应用中的用户操作步骤。
3. 编译测试案例代码为Jar包并复制该Jar包到安装了待测应用的测试手机中。
4. 运行测试并查看结果
5. 修改任何发现的bug，然后修复并重新测试。

本章大纲

- UI Automator 介绍
- UI Automator 环境搭建
- UI Automator 主要的对象类

UI Automator环境搭建

在当前module中修改build.gradle文件，增加

uiautomator的依赖

androidTestImplementation

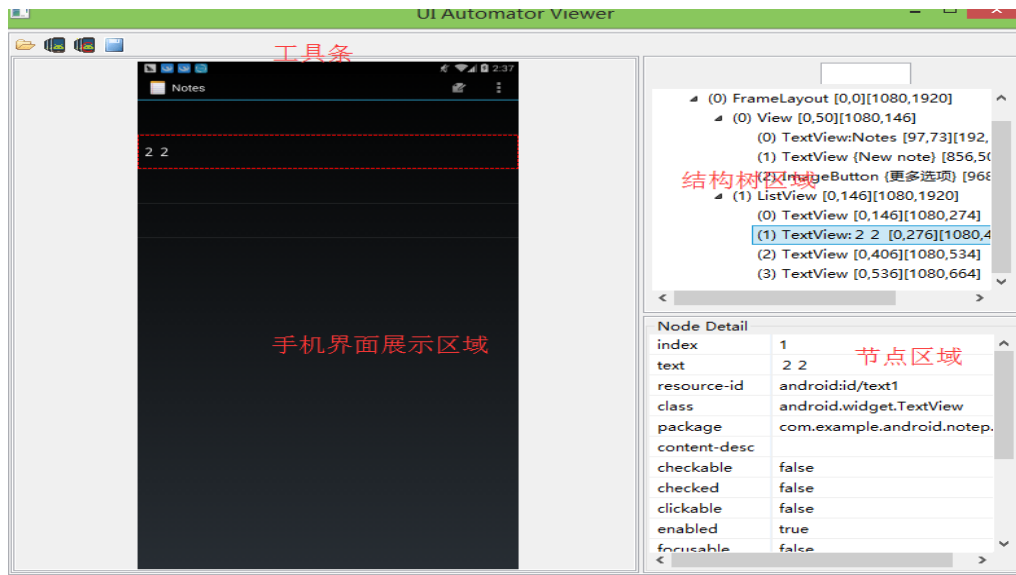
'com.android.support.test.uiautomator:uiautomator

-v18:2.1.3'

UI Automator演示实例

Ui Automator Viewer工具介绍

AndroidSDK\tools\bin uiautomatorviewer.bat



第一个实例

```
public class Demol {  
    private UiDevice mDevice;  
    @Test  
    public void demol() throws UiObjectNotFoundException {  
        mDevice = UiDevice.getInstance(InstrumentationRegistry.getInstrumentation());  
        mDevice.pressHome();  
        |  UiObject contact=mDevice.findObject(new UiSelector().text("联系人"));  
        contact.click();  
    }  
}
```

本章大纲

- UI Automator 介绍
- UI Automator 环境搭建
- UI Automator 主要的对象类

UI Automator 主要的对象类

UiDevice: 与设备相关的, 物理按键, 坐标点击

UiSelector: 定位控件

UiObject: 具体的界面控件

UiCollection: 一组控件的集合

UiScrollable: 可以滚动的控件

UiDevice类

检查设备不同的状态，屏幕尺寸，进行设备的操作，点击菜单键，Home等。

`device.pressHome()`;按Home

`device.pressDelete()`;按Delete

`device.pressMenu()`;菜单

`device.pressEnter()`;回车

`device.getDisplayHeight()`;获取屏幕高度

`device.getDisplayWidth()`;获取屏幕宽度

`device.click(100, 200)`;在坐标处进行点击操作

UiDevice类

`device.swipe(100, 100, 100, 500, 5)`。滑动操作，步长为5, 毫秒，步长越长，速度越慢。

`device.drag(100, 100, 200, 300, 5)`拖动操作

`device.pressKeyCode(KeyEvent.KEYCODE_A)`；输入小写a

`device.pressKeyCode(KeyEvent.KEYCODE_A, 1)`；输入大写A

`device.takeScreenshot(new File(“/sdcard/test.png”))`截屏

`device.click(198, 106)`；

激活状态	metaState
META_key未被激活	0
Shift或CapsLock被激活	1
ALT被激活	2
ALT、Shift或CapsLock同时被激活	3

UiDevice类

`device.wakeUp()`;

唤醒屏幕操作，如果屏幕已经是亮的，则不起任何作用，否则将唤醒屏幕

`device.isScreenOn()`;

判断当前屏幕是否是亮的，并将返回值赋给布尔类型变量status。

`sleep()`;

如果屏幕已经是关闭的，则不起任何作用，否则将关闭屏幕

`device.getCurrentPackageName()`;获取当前界面包名

`device.openNotification()`;打开通知栏

`device.openQuickSettings()`;打开快速设置

`device.dumpWindowHierarchy(“test.xml”)`;获取当前的布局文件，保存在
/data/local/tmp/test.xml。

UiDevice类

```
private UiDevice device;
```

```
@Before
```

```
public void startUp() throws UiObjectNotFoundException {
```

```
    device =
```

```
    UiDevice.getInstance(InstrumentationRegistry.getInstrumentat  
ion());
```

```
    device.pressHome();
```

```
    UiSelector s1= new UiSelector().text("Notes");
```

```
    UiObject object1=device.findObject(s1);
```

```
    object1.click();
```

```
}
```

UiSelector类

UiSelector类代表搜索UI控件的条件，可以在当前的界面上查询和获取特定元素的句柄。若找到多个元素，则返回布局层次结构的第一个匹配元素。

//创建一个以“Demo”开头的UiSelector 对象

```
UiSelector ww = new
```

```
UiSelector().textStartsWith("Demo");
```

//以特定的条件创建一个对象实例

```
UiObject obj = new UiObject(ww);
```

```
obj.click();
```

UiSelector类

//复合属性进行查询

```
UiObject appItem1 = new UiObject(new  
UiSelector().className("android.widget.TextView").text(  
"微信"));
```

```
UiObject appItem2= new UiObject(new  
UiSelector().focused(true).className("android.widget.Te  
xtView"));
```

```
UiSelector ui = new  
UiSelector().className("android.widget.TextView").insta  
nce(3);
```

UiSelector类

- 按resource id定位

```
new UiObject(new
```

```
UiSelector().resourceId("com.android.calculator2:id/digit_7"));
```

- 按text定位

```
new UiSelector().text( "") : 全匹配
```

```
new UiSelector().textContains( "") : 包含某文本
```

```
new UiSelector().textStartsWith( "") : 以某文本开头
```

```
new UiSelector().textMatches( "" ) : 正则表达式模式
```

```
new UiSelector().textMatches(".*系.*")
```

UiSelector类

- 按index定位

```
new UiSelector().index(1)
```

- 按class name定位

```
new UiSelector().className("android.view.view").text("登录")
```












- 按description定位










```
new UiSelector().description("APPs")
```









- 按package name定位

```
new UiSelector().packageName("com.example.todo").text("请输入用户名")
```

UiSelector类

```
m  checkable(boolean val) UiSelector
m  checked(boolean val) UiSelector
m  childSelector(UiSele... UiSelector
m  className(Class<T> t... UiSelector
m  className(String cla... UiSelector
m  classNameMatches(Str... UiSelector
m  clickable(boolean val) UiSelector
m  description(String d... UiSelector
m  descriptionContains... UiSelector
m  descriptionMatches(... UiSelector
m  descriptionStartsWith UiSelector
```

```
m  enabled(boolean val) UiSelector
m  focusable(boolean val) UiSelector
m  focused(boolean val) UiSelector
m  fromParent(UiSelecto... UiSelector
m  index(int index) UiSelector
m  instance(int instance) UiSelector
m  longClickable(boolea... UiSelector
m  packageName(String n... UiSelector
m  packageNameMatches(... UiSelector
```

```
m  resourceId(String id) UiSelector
m  resourceIdMatches(S... UiSelector
m  scrollable(boolean v... UiSelector
m  selected(boolean val) UiSelector
m  text(String text) UiSelector
m  textContains(String ... UiSelector
m  textMatches(String r... UiSelector
m  textStartsWith(Strin... UiSelector
```

UiObject类

UiObject类代表一个UI元素对象，为创建UiObject实例，需要通过UiSelector类查找UiObject，待找到实例后，通过实例的方法进行

@Test

```
public void demo2() throws UiObjectNotFoundException {  
    mDevice = UiDevice.getInstance(InstrumentationRegistry.getInstrumentation());  
    mDevice.pressHome();  
    UiSelector qq = new UiSelector().text("QQ");  
    UiObject obj = mDevice.findObject(qq);  
    obj.dragTo(334, 991, 10);|
```

```
}  
    UiSelector qq = new UiSelector().className("android.view.View").instance(0);  
    UiObject obj = mDevice.findObject(qq);  
    obj.swipeRight(10);
```

```
UiObject mobile = mDevice.findObject  
    (new UiSelector().className("android.widget.EditText").text("请输入手机号"));  
mobile.setText("13910102020");
```

UiCollection

继承于UiObject，它用于枚举一个容器用户界面元素的目的，可以通过其提供的一些方法获取容器内的子元素对象。当界面存在多个控件而无法用UiSelector描述目标控件的唯一性，或需要对界面元素进行遍历操作时，可以使用UiCollection来进行。

通过以下三个方法来获得查找的对象。

```
getChildByDescription(UiSelector childPattern, String text)
```

```
getChildByInstance(UiSelector childPattern, int instance)
```

```
getChildByText(UiSelector childPattern, String text)
```


UiCollection

@Test

```
public void UiCollectionDemo() throws UiObjectNotFoundException {  
    mDevice = UiDevice.getInstance(InstrumentationRegistry.getInstrumentation());  
    UiCollection u = new UiCollection  
        (new UiSelector().className("android.view.View").instance(0));  
    int count=u.getChildCount(new UiSelector().className("android.widget.TextView"));  
    System.out.println(u.getBounds().toString());  
    UiObject object = u.getChildByText  
        (new UiSelector().className("android.widget.TextView"), "ToDoList");  
    object.click();  
}
```

UiWatcher

用来处理脚本执行过程中遇到的一些异常情况，例如：在执行过程中突然打来电话，打乱了正在执行的步骤，需要通过UiWatcher来监听处理这种情况。Ui Automator使用`checkForCondition()`调用设备上所有已经启动的监听

@Test

```
public void UiWatcherDemo() throws UiObjectNotFoundException, RemoteException, InterruptedException;
```

//先要注册监听器

```
mDevice.registerWatcher("phone", new UiWatcher() {
```

```
    public boolean checkForCondition() {
```

```
        UiObject call = mDevice.findObject(new UiSelector().text("来电"));
```

```
        UiObject reject = mDevice.findObject(new UiSelector().text("拒绝"));
```

```
        UiObject view = mDevice.findObject(new UiSelector().className("android.view.View"));
```

@Test

```
public void UiWatcherDemo() throws UiObjectNotFoundException, RemoteException,  
InterruptedException {
```

```
    //先要注册监听器
```

```
    device.registerWatcher("phone", new UiWatcher() {
```

```
        public boolean checkForCondition() {
```

```
            UiObject call = device.findObject(new UiSelector().text("来电"));
```

```
            UiObject reject = device.findObject(new UiSelector().text("拒绝"));
```

```
            UiObject view = device.findObject  
                (new UiSelector().className("android.view.View"));
```

```
            if (call.exists()) {
```

```
                Log.i("", "电话监听器被触发啦！！！！");
```

```
            }
```

```
            if (reject.exists()) {
```

```
                Log.i("", "存在挂断电话图标");
```

```
                try {
```

```
                    device.swipe(reject.getBounds().left, reject.getBounds().top,
```

```
573, 569, 20);
```

```
                    return true;
```

```
                } catch (UiObjectNotFoundException e) {
```

```
                    e.printStackTrace();
```

```
                }
```

```
            }
```

```
        return false;
```

UiWatcher

```
UiSelector zhihu = new UiSelector().text("知乎");
UiObject obj = device.findObject(zhihu);
Thread.sleep(3000);
obj.click();
UiObject mobile = device.findObject
    (new UiSelector().className("android.widget.EditText").
    text("请输入手机号"));
mobile.setText("13910102020");
device.removeWatcher("phone");
device.hasAnyWatcherTriggered();
device.hasWatcherTriggered("phone");
}
```

UiScrollable

是UiCollection的子类，用来专门处理滚动事件的对象，其提供了丰富多样的滚动处理方法

```
getChildByText (UiSelector childPattern, String  
text)
```

```
getChildByText (UiSelector childPattern, String  
text, boolean allowScrollSearch)
```

区别在于allowScrollSearch为真，效果一样，为假，不允许滚动查找。

Configuration

配置基础类，用以控制测试过程的事件等待超时、控件可见超时等

延时项	默认延时	说明	API
动作	3s	设置延时	<code>setActionAcknowledgmentTimeout(long timeout)</code>
		获取默认延时	<code>getActionAcknowledgmentTimeout()</code>
键盘输入	0s	设置延时	<code>setKeyInjectionDelay(long delay)</code>
		获取默认延时	<code>getKeyInjectionDelay()</code>
滚动	200ms	设置延时	<code>setScrollAcknowledgmentTimeout(long timeout)</code>
		获取默认延时	<code>getScrollAcknowledgmentTimeout()</code>
空闲	10s	设置延时	<code>setWaitForIdleTimeout(long timeout)</code>
		获取默认延时	<code>getWaitForIdleTimeout()</code>
组件查找	10s	设置延时	<code>setWaitForSelectorTimeout(long timeout)</code>
		获取默认延时	<code>getWaitForSelectorTimeout()</code>

```
setActionAcknowledgmentTimeout  
setKeyInjectionDelay(long ... Cor  
setScrollAcknowledgmentTimeout  
setToolType(int toolType) Cor  
setWaitForIdleTimeout(long... Cor  
setWaitForSelectorTimeout(... Cor
```

Configuration

- `Configurator con = Configurator.getInstance() ;`
`con.setWaitForSelectorTimeout(5000) ;`

adb 启动 UI Automator

adb shell am :使用此命令可以从cmd控制台启动 activity, services;发送 broadcast等等

运行所有的用例: `adb shell am instrument -w com.example.think.uiautomatordemo.test/android.support.test.runner.AndroidJUnitRunner`

运行一个类中的所有用例: `adb shell am instrument -w -r -e class com.edu.uidemo.test com.example.think.uiautomatordemo.test/android.support.test.runner.AndroidJUnitRunner`

adb 启动 UI Automator

运行类中的某个方法

```
adb shell am instrument -w -r    -e debug false -e class  
com.example.think.uiautomatordemo.Demo1#demo  
com.example.think.uiautomatordemo.test/android.support.test.runner.AndroidJUnitRunner
```

运行多个类的所有用例: adb shell am instrument -w -r -e
debug false -e class
com.edu.uidemo.test.Demo1,com.edu.uidemo.test.Demo2
com.example.think.uiautomatordemo.test/android.support.test.runner.AndroidJUnitRunner