Uiautomator 简介

[一、Uiautomator API 1](#_Toc389740425)

[1、UiDevice 1](#_Toc389740426)

[2、UiSelector 2](#_Toc389740427)

[3、UiObject 2](#_Toc389740428)

[4、UiCollection 3](#_Toc389740429)

[5、UiScrollable 3](#_Toc389740430)

[二、Uiautomator Demo 4](#_Toc389740431)

本文主要介绍uiautomator的主要API，同时根据魔图实例来分析各个方法的使用

# 一、Uiautomator API

uiautomator API在 uiautomator.jar 文件中（AndroidSDK>4.1）。这些API分别如下：

其中getUiDevice().openNotification();

new UiObject(new UiSelector().resourceId(""));要求AndroidSDK>4.3

## 1、UiDevice

UiDevice代表设备状态。在测试中，可以通过UiDevice实例主要进行以下两个方面的操作：

1. 检测设备的各种属性，例如当前的屏幕方向以及屏幕尺寸
   1. getUiDevice().getDisplayRotation()
   2. getUiDevice().getDisplayHeight()
2. 通过UiDevice实例来执行设备级别的操作，例如 把设备设置为横屏或者竖屏、按下Home按键等。
   1. getUiDevice().pressHome()
   2. getUiDevice().pressMenu()
   3. getUiDevice().pressBack()

## 2、UiSelector

UiSelector代表一个搜索UI控件的条件。如果发现多个满足条件的控件则会返回第一个控件。 返回的结果为UiObject对象。在构造UiSelector的时候可以组合使用多个属性来定位具体的控件。如果没有找到控件则会抛出 UiAutomatorObjectNotFoundException 异常。还可以使用childSelector()函数来嵌套UiSelector 对象。

例如，下面的代码演示了如何在当前界面中查找第一个ListView中的带有文本属性为Apps的子控件。

UiObject appItem = new UiObject(new UiSelector()

.className("android.widget.ListView").instance(1)

.childSelector(new UiSelector().text("Apps")));

## ****3、UiObject****

UiObject代表一个UI控件。通过UiSelector来查找UiObject。   
如下示例代码演示了如何查找当前显示界面中的取消按钮和确认按钮：

UiObject cancelButton = new UiObject(new UiSelector().text("Cancel"));

UiObject okButton = new UiObject(new UiSelector().text("OK"));

查找到的UiObject实例可以在其他测试代码中重用。需要注意的是：每次使用UiObject做操作的时候uiautomator 都会在当前屏幕重新查找该控件。

例子：如下代码uiautomator 工具在当前界面查找文本内容为“OK”的控件。如果存在并且可用则模拟用户点击该控件。

if(okButton.exists() && okButton.isEnabled())

{

okButton.click();

}

例子：还可以限制仅仅查找特定类型的控件。例如 如下代码只查找文本为“Cancel”和“OK”的android.widget.Button类型控件。

UiObject cancelButton = new UiObject(new UiSelector().text("Cancel")

.className("android.widget.Button"));

UiObject okButton = new UiObject(new UiSelector().text("OK")

.className("android.widget.Button"));

## ****4、UiCollection****

UiCollection代表控件的集合。获取UiCollection的方式和UiObject一样，通过 UiSelector查找。 UiCollection对应Android系统中的ViewGroup以及子控件。   
例子：如下代码演示如何通过UiSelector来获取包含视频集合的UiCollection。

UiCollection videos = new UiCollection(new UiSelector()

.className("android.widget.FrameLayout"));

例子：如果每个视频是放到LinearLayout中的，则可以通过如下方式获取视频的数目：

int count = videos.getChildCount(new UiSelector()

.className("android.widget.LinearLayout"));

例子：如果需要查找标签为“Cute Baby Laughing”的视频，并点击。则可以通过如下方式：

UiObject video = videos.getChildByText(new UiSelector()

.className("android.widget.LinearLayout"), "Cute Baby Laughing");

video.click();

例子：同样还可以模拟其他用户操作。例如，模拟选择视频的操作如下：

UiObject checkBox = video.getChild(new UiSelector()

.className("android.widget.Checkbox"));

if(!checkBox.isSelected()){

checkbox.click();

}

## ****5、UiScrollable****

UiScrollable代表可滚动的控件。可以用UiScrollable来模拟水平或者垂直滚动的UI元素。如果需要操作的元素在屏幕外需要滚动屏幕才能看到的情况下需要使用UiScrollable。

例子，下面的代码显示了如何模拟滚动到“Settings ”菜单并点击“About tablet”菜单的操作。

UiScrollable settingsItem = new UiScrollable(new UiSelector()

.className("android.widget.ListView"));

UiObject about = settingsItem.getChildByText(new UiSelector()

.className("android.widget.LinearLayout"), "About tablet");

about.click()

其他API参考uiautomator api文档。

# 二、Uiautomator Demo

#### 一个简单的uiautomator 测试案例

如下是一个简单的测试案例代码，模拟了点击Home键回到主屏，然后点击所以应用按钮打开所有应用列表，并滚动到时钟应用。打开时钟应用 并选择闹铃界面的第一个闹钟设置，修改该设置的开关。然后返回到时钟界面再进入倒计时界面。

package com.uia.example.my;

import android.widget.ListView;

import android.widget.Switch;

import com.android.uiautomator.core.UiObject;

import com.android.uiautomator.core.UiObjectNotFoundException;

import com.android.uiautomator.core.UiScrollable;

import com.android.uiautomator.core.UiSelector;

import com.android.uiautomator.testrunner.UiAutomatorTestCase;

public class LaunchSettings extends UiAutomatorTestCase {

// TODO 重要注意： 在运行该测试代码的时候 需要先把手机语言环境设置为英文。

public void testDemo() throws UiObjectNotFoundException {

//模拟 HOME 键点击事件

getUiDevice().pressHome();

// 现在打开了主屏应用，模拟点击所有应用按钮操作来启动所有应用界面。

// 如果你使用了uiautomatorviewer来查看主屏，则可以发现“所有应用”按钮的

// content-description 属性为“Apps”。可以使用该属性来找到该按钮。

UiObject allAppsButton = new UiObject(new UiSelector().description ("Apps"));

//模拟点击所有应用按钮，并等待所有应用界面起来

allAppsButton.clickAndWaitForNewWindow();

//1——下面的步骤可以省略，原因未知

// 在所有应用界面，时钟应用位于Apps tab界面中。下面模拟用户点击Apps tab操作。

// 找到 Apps tab 按钮

UiObject appsTab = new UiObject(new UiSelector().text ("Apps"));

// 模拟点击 Apps tab.

appsTab.click();

//1——上面的步骤可以省略，原因未知</span>

// 然后在 Apps tab界面，模拟用户滑动到时钟应用的操作。

// 由于Apps界面是可以滚动的，所以用

// UiScrollable 对象.

UiScrollable appViews = new UiScrollable(new UiSelector().scrollable (true));

//设置滚动模式为水平滚动(默认为垂直滚动)

appViews.setAsHorizontalList();

if(allAppsButton.exists() && allAppsButton.isEnabled()) {

// allAppsButton在当前界面已经不可见了 所以这里不会执行</span>

allAppsButton.click();

}

// 查找时钟应用并点击

UiObject settingsApp = appViews.getChildByText(New UiSelector().className(android.widget.TextView.class.getName()),"Clock");

settingsApp.clickAndWaitForNewWindow();

// 验证当前显示 的应用包名为时钟

UiObject settingsValidation = new UiObject(new UiSelector().packageName("com.google.android.deskclock"));

// 如果不存在则出错提示

assertTrue ("Unable to detect Clock", settingsValidation.exists());

// 模拟点击时间tab

UiObject clock = new UiObject(new UiSelector().description ("Clock"));

clock.clickAndWaitForNewWindow();

// 模拟点击下方的闹钟图标

UiObject alarms = new UiObject(new UiSelector().description ("Alarms"));

alarms.clickAndWaitForNewWindow();

UiScrollable list = new UiScrollable (new UiSelector().className (ListView.class.getName()));

if(list.getChildCount() > 0) {//如有过设置闹钟，则使用第一个闹钟，且index是从0计数

UiObject listIndex0 = list. getChild (new UiSelector().index (0));

UiObject switchBtn = listIndex0.getChild(new UiSelector().className (Switch.class.getName()));

boolean isChecked = switchBtn.isChecked();

switchBtn.click();

}

// 模拟点击返回键，此处有可能直接退出clock应用

getUiDevice().pressBack();

UiObject timer = new UiObject(new UiSelector().description ("Timer"));

timer.clickAndWaitForNewWindow();

}

}

##### 打包测试代码并在测试机上运行

如下步骤打包测试代码并在测试机运行的步骤：

1.创建打包测试代码的Build脚本。通过如下命令来生成Build脚本：

<android-sdk>/tools/android create uitest-project -n <name> -t 1 -p <path>

<name>是包含测试代码的项目名称，path是包含测试代码的项目文件路径。-t后面的1是Android sdk版本id。

**注意**： 在创建Build脚本之前，已经创建好了测试项目并编写好了测试代码。只不过该测试项目还没有包含打包脚本所以无法打包运行。在创建Build脚本的时 候，<name>属性就是测试项目的名称、<path>就是已经创建的测试项目在电脑中的文件夹路径。 如果电脑上安装了多个版本的Android sdk，则需要运行**/tools/android list target** 来查看每个SDK的id。选择4.1以上的id即可。

2. 设置ANDROID\_HOME 环境变量。

Windows：set ANDROID\_HOME=

Unix：export ANDROID\_HOME=

3. 打开命令行创建，导航到第一步中的目录中，运行 ant build 来打包。

4. 通过adb push命令把上一步打包出来的jar文件复制到测试手机中。

*adb push <path>/bin/<name>.jar /data/local/tmp/*

*类似如下代码：*

*adb push ~/dev/workspace/LaunchSettings/bin/LaunchSettings.jar /data/local/tmp/*

##### 运行uiautomator 测试

##### 下面是运行 LaunchSettings.jar jar包中测试代码的命令。测试代码位于com.uia.example.my包中。

*adb shell uiautomator runtest LaunchSettings.jar -c com.uia.example.my.LaunchSettings*

关于uiautomator 的更多信息参考这里：<http://developer.android.com/tools/help/uiautomator/index.html>