**UiAutomator认识学习**

1. UiAutomator概念
2. Google官方提供android 4.1版本以上自带的的一个UI测试框架。用作UI测试，可以获取界面上的所有view元素和控件，也可以用作功能性测试，点击界面上的控件，形似手工点击控件
3. 测试人员不需要知道APP源代码的情况下就可以用UIAutomator对程序进行UI测试
4. 可以对整个系统做UI测试，不局限于APP
5. 环境配置

方法一：1. Android SDK

2. ANT

3.Eclipse

方法二：1.IDT（eclipse/SDK）

2.ANT

三． UI组件分析

1. SDK/tools/中uiautomatorviewer 工具对UI进行分析
2. 点击Device Screenshot可以获取当前界面的UI组件
3. UIAutomator工程建立
4. 在IDT /Eclipse中新建android Application project
5. 导入junit 4（单元测试框架）
6. 在scr下面新建class，包会自动生成，或是先建立包在建立class
7. API
8. UiDevice:通过UiDevice对象对设备控制，如点亮屏幕、锁屏、旋转方向、截图等，点击home、back、menu等。

通过getUiDevice()来获取UiDevice对象。

UiDevice=getUiDevice()

如：getUiDevice.pressHome();点击home键

截图、滑动

按键事件

锁屏和唤醒

点击、拖拽、层级

旋转

获取信息、清除文本

等待、监听器

UiDevice

通知栏、快速设置

1. UiObject:UI里面的某个元素或控件。UiObject的构造函数里是传入一个UiSelector对象，通过UiSelector来查找我们需要的元素或控件；

如：UiObject appsButton=new UiObject(new UiSelector().description(“apps”));

拿到UiObject对象后可以作如下操作

对象操作

文本输入与清除

手势操作

获取子类

getChild

getChildCount

点击或长按 click/longclick

对象是否存在 exist

属性 获取属性：getxxxx 属性判断：

isxxxx

拖动或滑动 Drag/swipe

1. UiSelector:通过UiSelector对象定位UI元素，如果发现多个条件满足的控件则会返回第一个控件，使用UiSelector的时候可以组合使用多个属性来定位具体的控件，还可以使用ChildSelector()来嵌套UiSelector对象

UiSelector

文件

text(String)

特有属性

Clicked(boolean)

resourceId(string)

父控件、子控件

描述

Description(String)

类名、包名、索引

1. UiScrollable：可滚动的控件，如果操作的控件需要滚动就可以用UiScrollable

如： UiScrollable appsList=new UiScrollable(new UiSelector().className(“android.view.View”));

appsList.setAsHorizontalList();

滑动坐标是的偏移量

UiScrollable

设置水平或者垂直滚动方向

滚动到某个对象

向前或向后滚动

获取或设置滚动次数

获取列表子元素

快速滚动

补充：

UiCollection :代表所有控件的集合，可以用来获取所有控件的个数，获取子元素对象

1. 获取子元素：

1>> public UiObject getChildByDescription(UiSelector childPattern,String text)

从UiSelector里选择符合制定条件的text子元素

2>> public UiObject getChildByinstance(UiSelector childPattern,String text)

从UiSelector里选择符合制定条件的instance子元素

3>> public UiObject getChildBytext(UiSelector childPattern,String text)

从UiSelector里选择符合制定条件的text子元素

1. 获取子元素个数：

Public int getChildCount(UiSelector childPattern)

1. 运行测试工程
2. Android create uitest-project –n xxx –t <android-sdk-id> -p <存放工程路径>

-n 生成的jar包名

-t后 接API大于16的SDK 的id （cd /sdk/tools/ 输入android list）

-p 是case工程目录

如：android create uitest-project -n dog -t 1 -p D:\

1. 执行完上述会生成一个build.xml文件，输入ant build

（要在workspace下）

1. 执行完会生成test.jar包，这个包会生成在工程bin目录下面，然后执行

adb push xxx.jar /data/local/tmp

1. 执行 adb shell uiautomator runtest xxx.jar –c 包名.类名

UiAutomator 运行学习

1. Android uiaotomator主要模块
2. JDK(Java development kit) Java编译环境
3. SDK(software development kit) android编译环境(API>15)
4. Eclipse Java编程软件
5. ADT(android development tools) 在Eclipse下开发工具的升级下载工具
6. ANT(advanced network tools) android的编译打包工具
7. API(application programming interface) 应用程序接口
8. JUnit3/4 自主进行单元测试，可以根据单元测试生成单元测试报告
9. Android uiautomator逻辑结构

ADT

UiAutomator

ANT

JAR

XML

JDK

Eclipse

SDK

Platforms

JUnit 3/4

Platform-tools

Tools

UIViewer

API

Device

DDMS

1. Android uiautomator环境配置

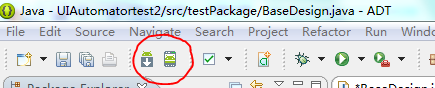
ADT、SDK环境配置

新建 SDK\_HOME= SDK绝对路径

系统变量path加入;%SDK\_HOME%tools;%SDK\_HOME%platform-tools;

查看是否成功：Android –help

Eclipse界面可以看到如下2个图标



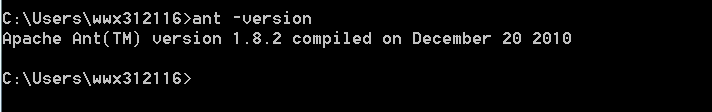
（直接用已经配置好的IDT软件直接编写软件）

ANT环境配置

新建ANT系统变量 ANT\_HOME= ANT绝对路径

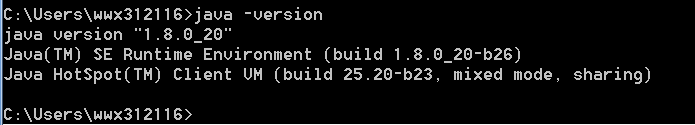
Path下放入%ANT\_HOME%bin

查看是否成功 ：Ant –version

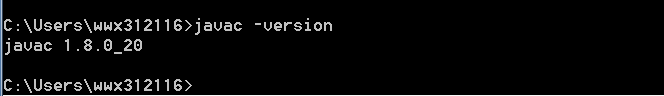


附：

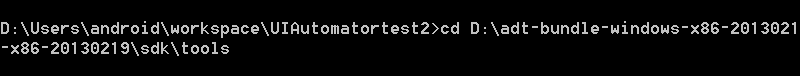
Java环境配置好



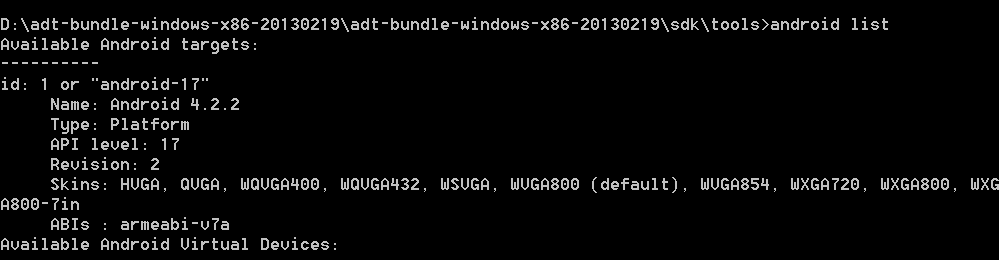
Javac环境配置好



1. Android uiautomator 运行步骤
2. 进入/SDK/tools目录下(后要继续输入d:，进入这个目录 ,微软电脑特性)

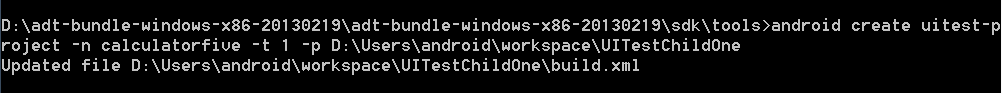


1. 列出>15的API端口

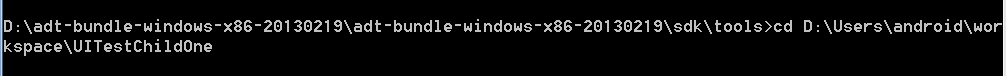


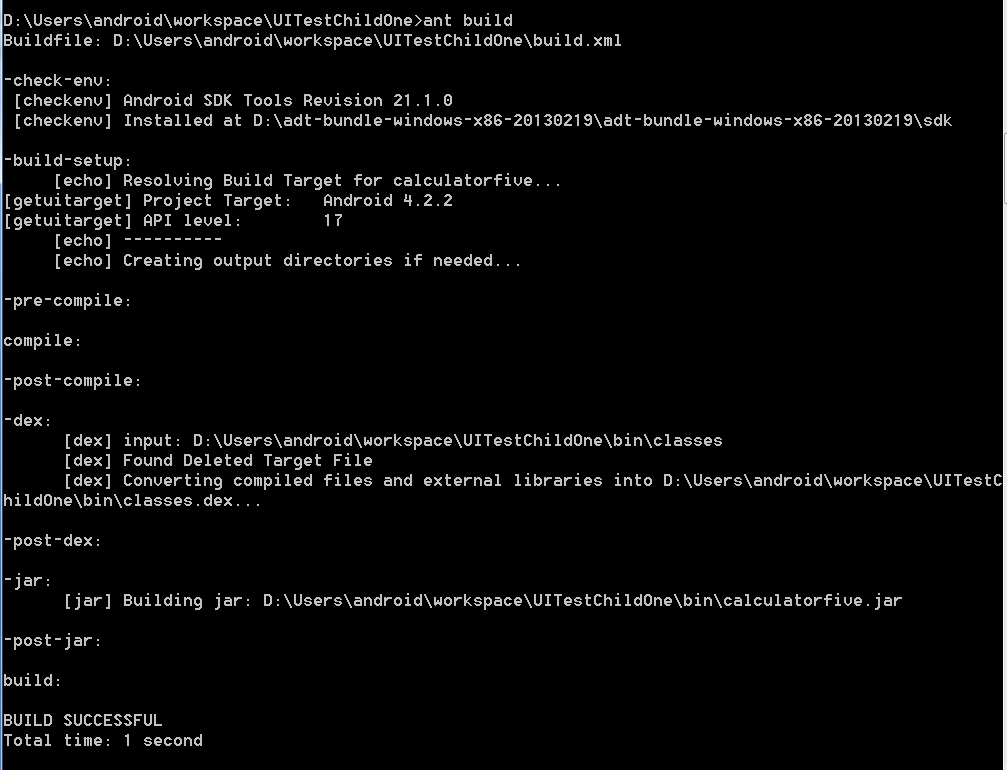
1. 选定API\_ID为1，生成build.xml文件

(若同位置已经存在build.xml文件，显示updated file;若无同文件，显示 added file)

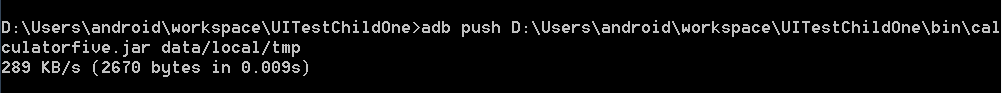


1. 进入workspace/工程名，用ant编译成jar架构包





1. 把工程架构包push到测试机data/local/tmp下



1. 运行工程架构包



附：大致流程

Adb push project

cd workspace/project

Adb runtest

ant build

Create build.xml

API

Android list

Cd SDK/tools

Src/ new package/ new class

New UI Project

Add platforms

Add JUnit4

Add uiautomator.jar

Add android.jar

UiAutomator深入学习

1. API-UiDevice学习
2. click

|  |  |  |
| --- | --- | --- |
| boolean | Click(int x,int y) | 指定坐标点点击 |

UiDevice.getInstance().click(300,300);

或 getUiDevice().click(300,300);

下列写法只适合同一个类，不适合调用其它类。上面写法正常

1. freezeRotation

|  |  |  |
| --- | --- | --- |
| void | freezeRotation | 禁用传感器与旋转，冻结当前旋转状态 |

UiDevice.getInstance().freezeRotation();

锁定手机竖屏，不让手机旋转

1. getCurrentPackageName

|  |  |  |
| --- | --- | --- |
| string | getCurrentPackageName() | 返回当前应用包名 |

String QQpackagenameString = UiDevice.getInstance().getCurrentPackageName();

返回当前QQ应用包名

1. getDisplayHeight与getDisplayWidth

|  |  |  |
| --- | --- | --- |
| Int | getDisplayHeight() | 获取显示器的高度 |
| Int | getDisplayWidth() | 获取显示器的宽度 |

Int height = UiDevice.getInstance().getDisplayHeight();

Int width = UiDevice.getInstance().getDisplayWidth();

获取当前手机的屏幕宽度与高度，及手机屏幕的分辨率

1. getDisplayRotation

|  |  |  |
| --- | --- | --- |
| Int | getDisplayRotation() | 返回当前的显示旋转度数，0,90,180,270 |

返回值为整数 0：旋转0度 1：旋转90度

2：旋转180度 3：旋转270度

Such as:

Switch(UiDevice.getInstance().getDisplayRotation()){

case 0:

System.out.printIn(“当前旋转0度”);

break；

case 1:

System.out.pintln(“当前选准90度”)；

Break;

case 2:

System.out.printIn(“当前旋转180度”);

Break；

case 3:

System.out.printIn(“当前旋转270度”);

default:

break；

}

1. getDisplaySizeDp

|  |  |  |
| --- | --- | --- |
| Point | getDisplaySizeDp() | 返回显示DP大小 |

DP：DP称DIP：device independent pixels(设备独立像素)

Point aPoint = getUiDevice().getDisplaySizeDp();

System.out.printIn(“width:”+aPoint.x);

System.out.printIn(“height:”+aPoint.y);

1. getProductName

|  |  |  |
| --- | --- | --- |
| String | getProductName() | 返回当前设备产品名 |

获取当前设备产品名

String product = UiDevice.getInstance().getProductName();

1. 监听器

|  |  |  |
| --- | --- | --- |
| Void | registerWatcher(string name, UiWatcher watcher) | 注册一个监听器，当前运行制定步骤被打断时，处理中断异常 |
| Void | removeWatcher(string name) | 移除之前注册的监听器 |
| Void | resetWatcherTriggers() | 重置一个监听器 |
| Void | runWatchers() | 强制运行所有监听器 |
| Boolean | hasAnyWatcherTriggered() | 检查是否有监听器触发过 |
| Boolean | hasWatcherTriggered(sting watchername) | 检查特定监听器是否触发过 |

Such as:

循环打开百度关闭百度应用过程中来电话，使用监听器挂断电话，让脚本继续执行。

//注册监听器，处理来电

UiDevice.getInstance().registWatcher(“hangUpThePhone”,new UiWatcher(){

Public Boolean checkForCondition(){

UiObject text = new UiObject(new UiSelector().text(“来电”));

UiObject call = new UiObject(new UiSelector().className(“”));

If(text.exists()){

Try{

Call.swipeRight(10);

}catch (UiObjectNotFoundException e){

e.printStackTrace();

}

Return true;

}else{

Return false;

}

}

})

//测试脚本

UiObject baiduObject = new UiObject(new UiSelector().text(“百度”))；

for(int i=0;i<=100;i++){

baiduObject.clickAndWaitForNewWindow();

sleep(2000);

UiDevice.getInstance().pressHome();

Sleep(1000);

}

1. 按键事件

|  |  |  |
| --- | --- | --- |
| Boolean | pressBack() | 短按返回 |
| Boolean | pressDPadCenter() | 轨迹球 |
| Boolean | pressDPadDown() | 轨迹球 |
| Boolean | pressDPadLeft() | 轨迹球 |
| Boolean | pressDPadRight() | 轨迹球 |
| Boolean | pressDPadUp() | 轨迹球 |
| Boolean | pressDelete() | 短按删除 |
| Boolean | pressEnter() | 短按回车 |
| Boolean | pressHome() | 短按home |
| Boolean | pressKeyCode(int keybode,int metaState) | 短按键盘代码 |
| Boolean | pressKeyCode(int keyCode) | 短按键盘代码 |
| Boolean | pressMenu() | 短按MENU键 |
| Boolean | pressRecentApps() | 短按最近应用按键 |
| Boolean | pressSearch() | 短按搜索键 |

按home键 UiDevice.getInstance().pressHome();

按menu键 UiDevice.getInstance().pressMenu();

按back 键 UiDevice.getInstance().pressBack();

pressKeyCode(int keyCode) 短按键盘代码，键盘上每个按键对用一个整数代码。

字母和数字键的键码值(keyCode)   
按键 键码 按键 键码 按键 键码 按键 键码   
A 65 J 74 S 83 1 49   
B 66 K 75 T 84 2 50   
C 67 L 76 U 85 3 51   
D 68 M 77 V 86 4 52   
E 69 N 78 W 87 5 53   
F 70 O 79 X 88 6 54   
G 71 P 80 Y 89 7 55   
H 72 Q 81 Z 90 8 56   
I 73 R 82 0 48 9 57   
  
数字键盘上的键的键码值(keyCode) 功能键键码值(keyCode)   
按键 键码 按键 键码 按键 键码 按键 键码   
0 96 8 104 F1 112 F7 118   
1 97 9 105 F2 113 F8 119   
2 98 \* 106 F3 114 F9 120   
3 99 + 107 F4 115 F10 121   
4 100 Enter 108 F5 116 F11 122   
5 101 - 109 F6 117 F12 123   
6 102 . 110   
7 103 / 111   
  
控制键键码值(keyCode)   
按键 键码 按键 键码 按键 键码 按键 键码   
BackSpace 8 Esc 27 Right Arrow 39 -\_ 189   
Tab 9 Spacebar 32 Down Arrow 40 .> 190   
Clear 12 Page Up 33 Insert 45 /? 191   
Enter 13 Page Down 34 Delete 46 `~ 192   
Shift 16 End 35 Num Lock 144 [{ 219   
Control 17 Home 36 ;: 186 \| 220   
Alt 18 Left Arrow 37 =+ 187 ]} 221   
Cape Lock 20 Up Arrow 38 ,< 188 '" 222

1. 旋转

|  |  |  |
| --- | --- | --- |
| void | freezeRotation() | 禁止传感器和冻结装备物理旋转在其当前旋转状态 |
| void | unfreezeRotation() | 启动传感器，允许旋转 |
| Boolean | isNaturalOrientation | 检查是否竖屏 |
| Void | setOrientationLeft() | 禁用传感器，向左旋转，并且固定 |
| Void | SetOrientationNatural() | 禁用传感器，旋转到自然默认状态，并且固定 |
| Void | SetOrientationRight() | 禁用传感器，向右旋转，并且固定 |

Such as:

测试横屏界面

UiDevice.getInstance().setOrientationLeft();

判断当前是否竖屏，不是则返回竖屏状态

if(!UiDevice.getInstance().isNaturalOrientation()){UiDevice.getInstance().setOrientationNatural();}

1. 锁屏与唤醒

|  |  |  |
| --- | --- | --- |
| Void | Sleep() | 锁屏，按电源键 |
| Void | WakeUp() | 唤醒，按电源键 |
| Void | isScreenOn() | 检查屏幕是否唤醒 |

Such as:

屏幕如果没有唤醒，则进行唤醒

if(!UiDevice.getInstance().isScreenOn()){UiDevice.getInstance().wakeUp();}

1. 等待对象

|  |  |  |
| --- | --- | --- |
| Void | waitForIdle(long timeout) | 等待当前应用程序处于空闲状态 |
| Void | waitForIdle() | 等待当前应用程序处于空闲状态 |
| Void | waitForWindowUpdate(sting packageName,long timeout) | 等待窗口内容更新时间发生 |

Such as:

点击界面坐标打开应用，点开应用后点击界面”搜索”按钮

UiDevice.getInstance().click(580.654);

//超时等待时间为0~500000ms

UiDevice.getInstance().waitForIdle(50000);

UiObject searchObject =new UiObject(new UiSelector().text(“搜索”))；

searchObject.click();

1. 截图

|  |  |  |
| --- | --- | --- |
| Boolean | takeScreenshot(File storagepath) | 当前窗口截图存储为png，file文件路径 |
| Boolean | takeScreenshot(File storagePath,float scale,int quality) | 当前窗口截图为png格式图片，自定义缩放比例和图片质量 |

Such as:

截图保存到SD卡下，图片文件名为test1

UiDevice.getInstance().takeScreenshot(new File(“/sdcard/test1.png”));

1. 拖拽与滑动

|  |  |  |
| --- | --- | --- |
| Boolean | swipe(Point[] segments,int segmentSteps) | 在点阵列中滑动，5ms一步 |
| Boolean | swipe(int startX,int startY,int endX,int endY,int steps) | 坐标滑动屏幕 |
| Boolean | Drag(int startX,int startY,int endX,int endY,int steps) | 从一个坐标点拖拽到另一个坐标点 |

Such as:

从左向右扫动屏幕：

int height=UiDevice.getInstance().getDisplayHeight();

int width=UiDevice.getInstance().getDisplayWidth();

UiDevice.getInstance().swipe(50,height/2,width-50,height/2,10);

1. 通知栏·快速设置

|  |  |  |
| --- | --- | --- |
| Boolean | openNotification() | 打开通知栏 |
| Boolean | openQuickSettings() | 打开快速设置 |

1. 窗口布局结构

|  |  |  |
| --- | --- | --- |
| Void | dumpWindowHierarchy(string fileName) | 用于调试转储当前窗口的布局层次结构。保存data/local/tmp |
| Void | setCompressedLayoutHeirarchy(Boolean compressed) | 启动或是禁用布局层次压缩 |

1. API-UiSelector学习
2. 文本

|  |  |
| --- | --- |
| text（String text） | 文本 |
| textContains(String text) | 文本包含 |
| textMatches(Srting regex) | 文本正则 |
| textStartsWith(String text) | 文本开始字符 |

1. 描述

|  |  |
| --- | --- |
| description(String desc) | 描述 |
| descriptionContains(String desc) | 描述包含 |
| descriptionMatches(String regex) | 描述正则 |
| descriptionStartsWith(String desc) | 描述开始字符匹配 |

1. 类名

|  |  |
| --- | --- |
| childSelector(UiSelector selector) | 子类 |
| className(String classname) | 类名 |
| classNameMatches(String regex) | 正则类名 |
| fromParent(UiSelector selector) | 父类 |

1. 索引

|  |  |
| --- | --- |
| index(int index) | 编号 |
| Instance(int instance) | 索引 |

1. 特有属性

|  |  |
| --- | --- |
| checked(Boolean val) | 选择属性 |
| clickable(Boolean val) | 可点击属性 |
| enabled(Boolean val) | Enable属性 |
| focusable(Boolean val) | 焦点属性 |
| focused(Boolean val) | 当前焦点属性 |
| longClickable(Boolean val) | 长按属性 |
| Scrollable(Boolean val) | 滚动属性 |
| Selected(Boolean val) | 可选属性 |

1. 包名

|  |  |
| --- | --- |
| packageName(String name) | 包名 |
| packageNameMatches(String regex) | 包名正则 |

1. 资源ID

|  |  |
| --- | --- |
| resourceId(String id) | 资源ID |
| resourceIdMatches(String regex) | 资源ID正则 |

1. API-UiObject 学习

(UiObject) 获取对象

操作对象

按照一定条件(uiselector)获取对象

1. 点击与长按

|  |  |  |  |
| --- | --- | --- | --- |
| 点击 | Boolean | click() | 点击对象 |
| Boolean | clickAndWaitForNewWindow(long timeout) | 点击对象，等待新窗口出现，参数为等待时间 |
| Boolean | clickAndWaitForNewWindow() | 点击对象，等待新窗口出现 |
| Boolean | clickBottomRight() | 点击对象右下角 |
| Boolean | clickTopLeft() | 点击对象左上角 |
| 长按 | Boolean | longClick() | 长按对象 |
| Boolean | longClickBottomRight() | 长按对象右下角 |
| Boolean | longClickTopLeft() | 长按对象左上角 |

UiObject ok = new UiObject(new UiSelector().text(“OK”));//生命对象

ok.click();//操作点击对象

1. 拖拽与滑动

|  |  |  |  |
| --- | --- | --- | --- |
| 拖拽 | boolean | dragTo(UiObject destObj,int steps) | 拖拽对象到另一个对象位置上，步长可设置拖动速度 |
| boolean | dragTo(int destX,int destY,int steps) | 拖拽对象到屏幕某个坐标位置上，步长可设置拖动速度 |
| 滑动 | boolean | swipeDown(int steps) | 拖动对象往下滑动 |
| boolean | swipeLeft(int steps) | 拖动对象往左滑动 |
| boolean | swipeRight(int steps) | 拖动对象往右滑动 |
| boolean | swipeUp(int steps) | 拖动对象往上滑动 |

UiObject jiedianhua =new UiObject(new UiSelector().textStartsWith(“下拉接听”))；

Jiedianhua.swipeDown(10);

1. 输入文本域清除文本

|  |  |  |  |
| --- | --- | --- | --- |
| 输入文本  清除文本 | boolean | setText(String text) | 在对象中输入文本 |
| void | clearTextField() | 清除编辑框中的文本 |

//短信编辑框中输入文本内容

UiObject inputmess = new UiObject(new UiSelector().text(“请输入内容”))；

Inputmess.setText(“短信内容”);

//输入完后清除短信编辑框文本内容

UiObject clear = new UiObject(new UiSelector().text(“请输入内容”));

clear.clearTextField();

1. 获取对象属性

|  |  |  |  |
| --- | --- | --- | --- |
| 获  取  对  象  属  性 | Rect | getBounds() | 获取对象矩阵坐标，矩形左上角坐标与右下角坐标 |
| UiObject | getChild(UiSelector selector) | 获取对象的子类对象，可以递归获取子孙中某个对象 |
| int | getChildCount() | 获得下一级子类数量 |
| String | getClassName() | 获得对象类名属性的类名文本 |
| String | getContentDescription() | 获取对象的描述属性的描述文本 |
| UiObject | getFromParent(UiSelector selector) | 从父类获取子类，按照uiselector获取兄弟类 |
| String | getPackageName() | 获取包名 |
| Final UiSelector | getSelector() | 获的selector用于调试 |
| String | getText() | 获取对象的文本 |
| Rect | getVisibleBounds() | 返回可见视图的范围 |

UiObject gettextObject =new UiObject(new UiSelector().className(“android.widget.TextView”));//获取第一个textview

String textString = gettextObject.getText();//获取textview的文本

1. 属性判断

|  |  |  |  |
| --- | --- | --- | --- |
| 属  性  判  断 | boolean | isCheckable() | 检查对象的checkable属性是否为true |
| boolean | isChecked() | 检查对象的checked属性是否为true |
| boolean | isClickable() | 检查对象的clickable属性是否为true |
| boolean | isEnabled() | 检查对象的enabled属性是否为true |
| boolean | isFocusable() | 检查对象的focusable属性是否为true |
| boolean | isFocused() | 检查对象的focused属性是否为true |
|  | boolean | isLongClickable() | 检查对象的longclickable属性是否为true |
| boolean | isScrollable() | 检查对象的scrollable属性是否为true |
| boolean | isSelected() | 检查对象的selector属性是否为true |

//声明开关对象

UiObject inputtextObject = new UiObject(new UiSelector().className(“android.widget.Switch”));

If(inputtextObject.isChecked()){inputtextObject.click();}//如果开关是开,则点击关闭

1. 手势操作

|  |  |  |  |
| --- | --- | --- | --- |
| 手  势  操  作 | boolean | performMultiPointerGesture(PointerCoords…touches) | 执行单手指触控手势，可定义任意手势与形状 |
| boolean | performTwoPointerGesture(Point startPoint1,Point startPoint2,Point endPoint1,Point endPoint2,int steps) | 执行任意两个手指触控手势，模拟两个手指手势 |
| boolean | pinchIn(int percent,int steps) | 手势操作，两点向内收缩 |
| boolean | pinchOut(int percent,int steps) | 手势操作，两点向外张开 |

UiObject fram = new UiObject(new UiSelector().classname(“android.widget.FrameLayout”));

PointerCoords p = new PointerCoords();

p.x =500;p.y =500 ; p.pressure =1 ;p.size =1;

PointerCoords p1=new PointerCoords();

P1.x =600 ;p1.y =600 ;p1.pressure =1 ;p1.size=1;

PointerCoords p2 = new PointerCoords();

P2.x =500 ;p2.y =500 ;p2.pressure=1;p2.size =1;

Fram.performMultiPointerGesture(p,p1,p2);

1. 判断对象是否存在与出现消失

|  |  |  |  |
| --- | --- | --- | --- |
| 判断对  象是  否存在 | boolean | waitForExists(long timeout) | 等待对象出现 |
| boolean | waitUntilGone(long timeout) | 等待对象消失 |
| boolean | exists() | 检查对象是否存在 |

//打开计算器

Runtime.getRuntime().exec(“am start –n com.android.calculator2/com.android.calculator2.Calculator”);

//计算器界面”+”对象

UiObject addObject =new UiObject(new UiSelector().text(“+”));

//等待检查对象出现

addObject.waitForExists(10000);

1. API-UiScrollable 学习

UiObject

UiCollection

UiScrollable

继承

继承

基本格式

UiScrollable 对象名 =new UiScrollable(new UiScrollable().选择对象条件)；

对象名.操作函数

1. 快速滚动

|  |  |  |
| --- | --- | --- |
| Boolean | flingBackward() | 以步长为5快速向后滑动 |
| Boolean | flingForward() | 以步长为5快速向前滑动 |
| Boolean | flingToBeginning(int maxSwipes) | 自定义扫动次数以步长为5快速滑动到开启 |
| Boolean | flingToEnd(int maxSwipes) | 自定义扫动次数以步长为5快速滑动到结束 |

//获取滚动对象

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

listScrollable.flingToEnd(2);//只滑动2次滚动到结尾

1. 获取列表子元素

|  |  |  |
| --- | --- | --- |
| UiObject | getChildByDescription(UiSelector childPattren,String text,Boolean allowScrollSearch) | 是否允许滚动查找获取具备uiselector条件与文本描述的uiobject对象 |
| UiObject | getChildByDescription(UiSelector childPattern,String text) | 默认滚动获取具备uiselector条件与文本描述条件的uiobject对象 |
| UiObject | getChildByInstance(UiSelector childPattern, int instance) | 获取具备uiselector条件的子集，再从子集中按照实例筛选想要的元素(不滚动) |
| UiObject | getChildByText(UiSeletor childPattern,String text,Boolean allowScrollSearch) | 是否允许滚动获取具备uiselector条件与文本条件的uiobject对象 |
| UiObject | getChildByText(UiSelector childPattern,String text) | 默认滚动获取具备uiselector条件与文本条件的uiobject对象 |

UiScrollable listScrollable = new UiScrollable(new UiSelector().scrollable(true));//获取滚动对象元素

//不滚动获取符合条件的子元素对象

UiObject yidongObject = listScrollable.getChildByDescription(new UiSelector().classname(“android.widget.TextView”),”10086”,false);

yidongObject.clickAndWaitForNewWindow();//点击等待新界面出现

1. 获取与设置最大滚动次数常量值

|  |  |  |
| --- | --- | --- |
| int | getMaxSearchSwipes() | 获取执行搜索滑动过程中最大滑动次数，默认常量为30 |
| UiScrollable | setMaxSearchSwipes(int swipes) | 设置最大可扫动次数 |

列表滚动20次才能到结尾位置，需要操作的元素位于19页，设置默认最大滚动次数为10

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

listScrollable.setMaxSearchSwipes(10);

listScrollable.scrollTextIntoView(“zzzz”);

1. 滑动区域校准常量设置与获取

|  |  |  |
| --- | --- | --- |
| double | getSwipeDeadZonePercentage() | 默认常量值为0.1,10% |
| UiScrollable | setSwipeDeadZonePercentage(double swipeDeadZonePercentage) | 设置一个部件的大小，在滑动时，视为无接触区的百分比 |

校准常量指的是，滑动操作坐标，取偏移比例

int swiperAreaAdjust = (int)(rect.height()\*getSwipeDeadZonePercentage());//计算坐标偏移量

downX = rect.centerX();

downY = rect.top +swipeAreaAdjust;

upX = rect.centerX();

upY = rect.bottom – swipeAreaAdjust;

1. 向前与向后滚动

|  |  |  |
| --- | --- | --- |
| boolean | scrollBackward(int steps) | 自定义步长向后滑动 |
| boolean | scrollBackward() | 以默认步长55向后滑动 |
| boolean | scrollDescriptionIntoView(String text) | 滚动到描述所在位置，并且尽量让它居于屏幕中央 |
| boolean | scrollForward() | 以默认步长55向前滚动 |
| boolean | scrollForward(int steps) | 自定义步长向前滚动 |

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

listScrollable.scrollBackward(10);//向后滚动

1. 滚动到某个对象

|  |  |  |
| --- | --- | --- |
| boolean | scrollabletoView(UiSelector selector) | 滚动到条件元素所在位置，并且尽量让其居于屏幕中央 |
| boolean | scrollIntoView(UiObject obj) | 滚动到对象所在位置，并且尽量让其居于屏幕中央 |
| boolean | scrollTextIntoView(String text) | 滚动到文本对象所在位置，并且尽量让其居于屏幕中央 |
| boolean | scrollToBeginning(int maxSwipes) | 滚动到开始位置 |
| boolean | scrollToBeginning(int maxSwipes,int steps) | 自定义扫动距离与步长滚动到开始位置 |
| boolean | scrollToEnd(int maxSwipes,int steps) | 自定义扫动距离与步长滚动到结束位置 |
| boolean | scrollToEnd(int maxSwipes) | 自定义扫动距离滚动到结束位置 |

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

listScrollable.scrollDescriptionIntoView(“jjjj”);

1. 设置滚动方向

|  |  |  |
| --- | --- | --- |
| UiScrollable | setAsHorizontalList() | 设置滚动方向设置为水平滚动 |
| UiScrollable | setAsVerticalList() | 设置滚动方向设置为纵向方向 |

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

listScrollable.setAsHorizontalList();//设置滚动方向为横向滚动

listScrollable.scrollForward();//往前滚动一次

listScrollable.setAsVerticalList();//设置滚动方向为纵向滚动

listScrollable.scrollTextIntoView(“我的收藏”);//滚动到我要查找的元素