**创建及编译UiAutomator测试工程：**

1) Eclipse中创建Java工程，如在$HOME/workspace创建MultimediaTest工程

2) 右键点击项目名称，Build Path -> Configure Build Path... -> Add External JARs 选择PC上Android SDK根目录下platforms/android-19目录下的android.jar、uiautomator.jar两个包

3）创建测试类(extends UiAutomatorTestCase)，编写测试用例。

4）进入工程所在目录，执行android create uitest-project -n <name> -t 1 -p <path>

eg. 进入$HOME/workspace目录，执行

android create uitest-project -n MultimediaTest -t 1 -p MultimediaTest

5）进入工程目录，ant clean build。

eg. MultimediaTest目录下执行ant clean build 生成 bin/MultimediaTest.jar

**执行测试用例：**

1）将编译到的jar包push到终端内

2）执行用例

adb shell uiautomator runtest /storage/sdcard0/MultimediaTest.jar -c com.chinatsp.test.multimedia.PlaysourceTest #测试类PlaysourceTest

adb shell uiautomator runtest /storage/sdcard0/MultimediaTest.jar -e snap false #测试jar中所有测试类。 -e snap false 自定义参数 snap = false

# jar：若指定为相对路径，则默认其在终端/data/local/tmp目录下。

**UI Automator Viewer工具位置**: $Android\_SDK\_HOME/tools/uiautomatorviewer

**UiAutomatorTestCase**

extends junit.framework.TestCase。 Inherited Methods:TestCase/Assert/Object/Test。

public Bundle getParams (); // 获取参数。uiautomator执行时-e指定的参数

public UiDevice getUiDevice (); // 获取当前设备

public void sleep (long ms);

**UiDevice**

设备相关。提供获取设备相关信息的接口、模拟用户触屏/按键操作接口。

public boolean click (int x, int y); //点击。 拖动事件drag，API17中没有此接口，API18才开始有。可使用UiScrollable来实现列表滑动。

public boolean pressBack (); //BACK键

public boolean pressHome (); //HOME键

public boolean pressKeyCode (int keyCode); //按键值

public String getCurrentPackageName (); //获取当前包名，即当前被测APP的包名

public int getDisplayHeight (); //像素

public int getDisplayWidth ();

public String getProductName (); // sirfsocv7

public boolean takeScreenshot (File storePath); //截屏

public void registerWatcher (String name, UiWatcher watcher); 注册UiWatcher

public void removeWatcher (String name);

public void runWatchers(); // 强制执行已注册的所有UiWatcher的checkForCondition()。

public boolean hasAnyWatcherTriggered (); //是否被触发。若UiWatcher被执行且checkForCondition()返回true则认为其被触发。

public boolean hasWatcherTriggered (String watcherName);

public void resetWatcherTriggers ();

**UiSelector**

UiSelector对象可以理解为一种条件对象，描述的是一种条件，可以配合UiObject使用得到某个符合条件的控件对象。

public UiSelector ();

所有的方法都是public的，且都返回UiSelector类的对象。

文本方面的方法：

　　1.text(String text) 文本 // public UiSelector text(String text);

　　2.textContains(String text) 文本包含

　　3.textMatches(String regex) 文本正则

　　4.textStartsWith(String text) 文本开始字符

描述方面的方法：

　　1.description(String desc) 描述

　　2.descriptionContains(String desc) 描述包含

　　3.descriptionMatches(String regex) 描述正则

　　4.descriptionStartsWith(String desc) 描述开始字符

类名方面的方法：

　　1.childSelector(UiSelector selector) 子类

　　2.className(String className) 类名

索性、实例方面的方法：

　　1.index(int index) 编号

　　2.instance(int instantce) 索引

特有属性：

　　1.checked(boolean val) 选择属性

　　2.chickable(boolean val) 点击属性

　　3.enabled(boolean val) enabled属性

　　4.focusable(boolean val) 焦点属性

　　5.longClickable(boolean val) 长按属性

　　6.scrollable(boolean val) 滚动属性

　　7.selected(boolean val) 选择属性

包名方面的方法：

　　1.packageName(String name) 包名

　　2.packageNameMatches(String regex) 包名正则

资源ID方面的方法：

　　1.resourceId(String id) 资源ID

　　2.resourceIdMatches(String regex) 资源ID正则

**UiObject**

控件对象。a representation of a view.

public UiObject (UiSelector selector); //eg. UiObject icon = new UiOjbect(new UiSelector().text("多媒体"));

public boolean click () throws UiObjectNotFoundException; // 触发控件点击事件。若控件未找到则抛出异常UiObjectNotFoundException

public boolean clickAndWaitForNewWindow (long timeout) throws UiObjectNotFoundException; //触发控件点击事件并等待新窗口跳转。若控件未找到则抛出异常UiObjectNotFoundException。

比如：启动新的Activity、弹出一个菜单、弹出一个对话框。

params: timeout：超时，mills

public boolean clickAndWaitForNewWindow () throws UiObjectNotFoundException; //default timeout

public boolean longClick () throws UiObjectNotFoundException; //长按

public boolean exists (); //控件是否存在

public UiObject getChild (UiSelector selector) throws UiObjectNotFoundException; // 获取子控件

public boolean isCheckable () throws UiObjectNotFoundException;

public boolean isChecked () throws UiObjectNotFoundException;

public boolean isEnabled () throws UiObjectNotFoundException;

public boolean isFocusable () throws UiObjectNotFoundException;

public boolean isFocused () throws UiObjectNotFoundException;

public boolean isScrollable () throws UiObjectNotFoundException;

public boolean isSelected () throws UiObjectNotFoundException;

public boolean setText (String text) throws UiObjectNotFoundException; //输入字符。此操作会先清空原有内容。

public boolean swipeDown (int steps) throws UiObjectNotFoundException; //滑动。 类似UiScrollable中的scrollToBeginning/scrollToEnd/scrollForward/scrollBackward

public boolean swipeLeft (int steps) throws UiObjectNotFoundException; // 向左滑动，查看列表右半部分。

public boolean swipeRight (int steps) throws UiObjectNotFoundException;

public boolean swipeUp (int steps) throws UiObjectNotFoundException;

public boolean waitForExists (long timeout); // 等待控件变为可见。若timeout(mills)内变为可见，则返回true。

public boolean waitUntilGone (long timeout);// 等待控件变为不可见。若timeout(mills)内变为不可见，则返回true。

protected AcessibilityNodeInfo findAccessibilityNodeInfo(long timeout); //未找到则返回null。timeout时间内检测当前UiSelector是否存在，若未存在执行runWatcher，等待1S后继续检测当前UiSelector，循环直到检测到当前UiSelector或超时。

**UiScrollable**

A UiCollection that supports searching for items in scrollable layout elements

public UiScrollable (UiSelector container); // eg. UiScrollable list = new UiScrollable(new UiSelector().scrollable(true));

public UiScrollable setAsHorizontalList (); // 横列表

public UiScrollable setAsVerticalList (); // 纵列表

public boolean scrollBackward () throws UiObjectNotFoundException; //滚动距离与列表显示高度(纵列表)/宽度(横列表)有关。 Return: true if scrolled, and false if can't scroll anymore

public boolean scrollBackward (int steps) throws UiObjectNotFoundException;

public boolean scrollForward () throws UiObjectNotFoundException;

public boolean scrollForward (int steps) throws UiObjectNotFoundException;

public boolean scrollToBeginning (int maxSwipes) throws UiObjectNotFoundException;

// 相当于重复操作scrollBackward，直到滚动到列表开头或操作maxSwipes次。

public boolean scrollToEnd (int maxSwipes) throws UiObjectNotFoundException;

public boolean scrollToBeginning (int maxSwipes, int steps) throws UiObjectNotFoundException;

public boolean scrollToEnd (int maxSwipes, int steps) throws UiObjectNotFoundException;

public UiScrollable setMaxSearchSwipes (int swipes); // Sets the maximum number of scrolls allowed when performing a scroll action in search of a child element. default=30

public int getMaxSearchSwipes ();

public boolean scrollIntoView (UiSelector selector) throws UiObjectNotFoundException; // 向前滑动forward，直到符合selector的控件已找到或结束(maxSearchSwipes)。true：表示找到。

public boolean scrollIntoView (UiObject obj) throws UiObjectNotFoundException;

public boolean scrollTextIntoView (String text) throws UiObjectNotFoundException;

public boolean scrollDescriptionIntoView (String text) throws UiObjectNotFoundException;

**UiWatcher**

UiDevice上注册UiWatcher，测试被打断场景。（如测试中有来电界面、弹出对话框。。。）

public abstract boolean checkForCondition ();

官方手册：sdk/docs/tools/help/uiautomator/index.html

深入研究UiAutomator：

1. uiautomator 二进制程序源代码：

bsp/android/frameworks/testing/uiautomator/cmds/

1. UiDevice、UiObject等类源代码：

bsp/android/frameworks/testing/uiautomator/library/