**Android 自动化测试工具Robotium 之Solo类**

package com.robotium.solo;

import java.lang.reflect.Method;  
import java.util.ArrayList;  
import junit.framework.Assert;  
import android.app.Activity;  
import android.app.Instrumentation;  
import android.content.pm.ActivityInfo;  
import android.graphics.PointF;  
import android.os.Environment;  
import android.view.KeyEvent;  
import android.view.View;  
import android.webkit.WebView;  
import android.widget.AbsListView;  
import android.widget.Button;  
import android.widget.CheckBox;  
import android.widget.CheckedTextView;  
import android.widget.CompoundButton;  
import android.widget.DatePicker;  
import android.widget.EditText;  
import android.widget.ImageButton;  
import android.widget.ImageView;  
import android.widget.ProgressBar;  
import android.widget.RadioButton;  
import android.widget.ScrollView;  
import android.widget.SlidingDrawer;  
import android.widget.Spinner;  
import android.widget.TextView;  
import android.widget.ListView;  
import android.widget.TimePicker;  
import android.widget.ToggleButton;  
import android.app.Instrumentation.ActivityMonitor;

/\*\*  
\* Robotium测试主入口，提供给各类测试使用  
\* Main class for development of Robotium tests.  
\* Robotium has full support for Views, WebViews, Activities, Dialogs, Menus and Context Menus.  
\*   
\* Robotium can be used in conjunction with Android test classes like  
\* ActivityInstrumentationTestCase2 and SingleLaunchActivityTestCase.  
\*  
\* @author Renas Reda, renas.reda@robotium.com  
\*/

public class Solo {  
// 断言工具类  
protected final Asserter asserter;  
// view获取工具类  
protected final ViewFetcher viewFetcher;  
// check类控件工具类  
protected final Checker checker;  
// 点击工具类  
protected final Clicker clicker;  
// 按动作工具类  
protected final Presser presser;  
// 控件搜索工具类  
protected final Searcher searcher;  
// activity操作工具类  
protected final ActivityUtils activityUtils;  
// 弹框操作工具类  
protected final DialogUtils dialogUtils;  
// 文本输入工具类  
protected final TextEnterer textEnterer;  
// 屏幕方向操作工具类  
protected final Rotator rotator;  
// 带滚动条控件擦做工具类  
protected final Scroller scroller;  
// 等待工具类  
protected final Sleeper sleeper;  
// 手动划屏操作工具类  
protected final Swiper swiper;  
// 手指点击操作工具类  
protected final Tapper tapper;  
// View等待工具类  
protected final Waiter waiter;  
// 设置类控件操作工具类  
protected final Setter setter;  
// View属性获取工具类  
protected final Getter getter;  
// WebView操作工具类  
protected final WebUtils webUtils;  
// 按键事件发送工具类  
protected final Sender sender;  
// 截图操作工具类  
protected final ScreenshotTaker screenshotTaker;  
// Instrument,用于发送各类事件  
protected final Instrumentation instrumentation;  
// 放大动作工具类  
protected final Zoomer zoomer;  
// 网站地址  
protected String webUrl = null;  
// 相关属性配置  
private final Config config;  
// 横屏  
public final static int LANDSCAPE = ActivityInfo.SCREEN\_ORIENTATION\_LANDSCAPE; // 0  
// 竖屏  
public final static int PORTRAIT = ActivityInfo.SCREEN\_ORIENTATION\_PORTRAIT; // 1  
// 右方向键  
public final static int RIGHT = KeyEvent.KEYCODE\_DPAD\_RIGHT;  
// 左方向键  
public final static int LEFT = KeyEvent.KEYCODE\_DPAD\_LEFT;  
// 向上方向键  
public final static int UP = KeyEvent.KEYCODE\_DPAD\_UP;  
// 向下方向键  
public final static int DOWN = KeyEvent.KEYCODE\_DPAD\_DOWN;  
// 回车按钮  
public final static int ENTER = KeyEvent.KEYCODE\_ENTER;  
// Menu按钮  
public final static int MENU = KeyEvent.KEYCODE\_MENU;  
// DEL按钮  
public final static int DELETE = KeyEvent.KEYCODE\_DEL;  
// 关闭  
public final static int CLOSED = 0;  
// 打开  
public final static int OPENED = 1;

/\*\*  
\* 构造函数,使用默认配置  
\* Constructor that takes the Instrumentation object and the start Activity.  
\*  
\* @param instrumentation the {@link Instrumentation} instance  
\* @param activity the start {@link Activity} or {@code null}  
\* if no Activity is specified  
\*/

public Solo(Instrumentation instrumentation, Activity activity) {  
this(new Config(), instrumentation, activity);  
}

/\*\*  
\* 带指定配置，不带 activity构造函数  
\* Constructor that takes the Instrumentation and Config objects.  
\*  
\* @param instrumentation the {@link Instrumentation} instance  
\* @param config the {@link Config} instance  
\*/

public Solo(Instrumentation instrumentation, Config config) {  
this(config, instrumentation, null);  
}

/\*\*  
\* 构造函数，包含3个参数  
\* Private constructor.  
\*  
\* @param config the {@link Config} instance  
\* @param instrumentation the {@link Instrumentation} instance  
\* @param activity the start {@link Activity} or {@code null}  
\* if no Activity is specified  
\*/

private Solo(Config config, Instrumentation instrumentation, Activity activity) {  
this.config = config;  
this.instrumentation = instrumentation;  
this.sleeper = new Sleeper();  
this.sender = new Sender(instrumentation, sleeper);  
this.activityUtils = new ActivityUtils(instrumentation, activity, sleeper);  
this.viewFetcher = new ViewFetcher(activityUtils);  
this.screenshotTaker = new ScreenshotTaker(config, activityUtils, viewFetcher, sleeper);  
this.dialogUtils = new DialogUtils(activityUtils, viewFetcher, sleeper);  
this.webUtils = new WebUtils(config, instrumentation,activityUtils,viewFetcher, sleeper);  
this.scroller = new Scroller(config, instrumentation, activityUtils, viewFetcher, sleeper);  
this.searcher = new Searcher(viewFetcher, webUtils, scroller, sleeper);  
this.waiter = new Waiter(activityUtils, viewFetcher, searcher,scroller, sleeper);  
this.setter = new Setter(activityUtils);  
this.getter = new Getter(instrumentation, activityUtils, waiter);  
this.asserter = new Asserter(activityUtils, waiter);  
this.checker = new Checker(viewFetcher, waiter);  
this.clicker = new Clicker(activityUtils, viewFetcher,sender, instrumentation, sleeper, waiter, webUtils, dialogUtils);  
this.zoomer = new Zoomer(instrumentation);  
this.swiper = new Swiper(instrumentation);  
this.tapper = new Tapper(instrumentation);  
this.rotator = new Rotator(instrumentation);  
this.presser = new Presser(viewFetcher, clicker, instrumentation, sleeper, waiter, dialogUtils);  
this.textEnterer = new TextEnterer(instrumentation, clicker, dialogUtils);  
// 进行初始化  
initialize();  
}

/\*\*  
\* 配置静态类，用于设置Robotium的一些属性  
\* Config class used to set the scroll behaviour, default timeouts, screenshot filetype and screenshot save path.  
\*

\* Example of usage:  
\*

\* public void setUp() throws Exception { \* Config config = new Config(); \* config.screenshotFileType = ScreenshotFileType.PNG; \* config.screenshotSavePath = Environment.getExternalStorageDirectory() + "/Robotium/"; \* config.shouldScroll = false; \* solo = new Solo(getInstrumentation(), config); \* getActivity(); \* } \*

\*  
\* @author Renas Reda, renas.reda@robotium.com  
\*/

public static class Config {

/\*\*  
\* get is set assert enter click等方法的默认超时时间10s  
\* The timeout length of the get, is, set, assert, enter and click methods. Default length is 10 000 milliseconds.  
\*/  
public int timeout\_small = 10000;

/\*\*  
\* waitFor方法的默认超时时间20s  
\* The timeout length of the waitFor methods. Default length is 20 000 milliseconds.  
\*/  
public int timeout\_large = 20000;

/\*\*  
\* 截图存储路径.默认为/sdcard/Robotium-Screenshots/  
\* The screenshot save path. Default save path is /sdcard/Robotium-Screenshots/.  
\*/  
public String screenshotSavePath = Environment.getExternalStorageDirectory() + “/Robotium-Screenshots/”;

/\*\*  
\* 截图类型，默认为jpg  
\* The screenshot file type, JPEG or PNG. Use ScreenshotFileType.JPEG or ScreenshotFileType.PNG. Default file type is JPEG.  
\*/  
public ScreenshotFileType screenshotFileType = ScreenshotFileType.JPEG;

/\*\*  
\* get is set enter type click方法操作时，默认对scroll类型的控件拖动滚动条  
\* Set to true if the get, is, set, enter, type and click methods should scroll. Default value is true.  
\*/  
public boolean shouldScroll = true;

/\*\*  
\* 设置是否使用JavaScript执行WebElement 点击动作，默认是false  
\* Set to true if JavaScript should be used to click WebElements. Default value is false.  
\*/  
public boolean useJavaScriptToClickWebElements = false;

/\*\*  
\* 截图枚举类型jpg png  
\* The screenshot file type, JPEG or PNG.  
\*  
\* @author Renas Reda, renas.reda@robotium.com  
\*  
\*/  
public enum ScreenshotFileType {  
JPEG, PNG  
}  
}

/\*\*  
\* 构造函数  
\* Constructor that takes the instrumentation object.  
\*  
\* @param instrumentation the {@link Instrumentation} instance  
\*/

public Solo(Instrumentation instrumentation) {  
// 传递activity参数为null  
this(new Config(), instrumentation, null);  
}

/\*\*  
\* 获取ActivityMonitor对象  
\* Returns the ActivityMonitor used by Robotium.  
\*  
\* @return the ActivityMonitor used by Robotium  
\*/

public ActivityMonitor getActivityMonitor(){  
// 获取ActivityUtils的ActivityMonitor属性  
return activityUtils.getActivityMonitor();  
}

/\*\*  
\* 所以当前界面中的所有View  
\* Returns an ArrayList of all the View objects located in the focused  
\* Activity or Dialog.  
\*  
\* @return an {@code ArrayList} of the {@link View} objects located in the focused window  
\*/

public ArrayList getViews() {  
try {  
// 获取所有的View  
return viewFetcher.getViews(null, false);  
} catch (Exception e) {  
e.printStackTrace();  
return null;  
}  
}

/\*\*  
\* 获取指定parent View中的所有View  
\* Returns an ArrayList of the View objects contained in the parent View.  
\*  
\* @param parent the parent view from which to return the views  
\* @return an {@code ArrayList} of the {@link View} objects contained in the specified {@code View}  
\*/

public ArrayList getViews(View parent) {  
try {  
// 获取指定parent中的所有的View  
return viewFetcher.getViews(parent, false);  
} catch (Exception e) {  
e.printStackTrace();  
return null;  
}  
}

/\*\*  
\* 获取指定view的mParent属性  
\* Returns the absolute top parent View of the specified View.  
\*  
\* @param view the {@link View} whose top parent is requested  
\* @return the top parent {@link View}  
\*/

public View getTopParent(View view) {  
View topParent = viewFetcher.getTopParent(view);  
return topParent;  
}

/\*\*  
\* 等待指定的text内容出现，超时时间20s  
\* Waits for the specified text to appear. Default timeout is 20 seconds.  
\*  
\* @param text the text to wait for, specified as a regular expression  
\* @return {@code true} if text is displayed and {@code false} if it is not displayed before the timeout  
\*/

public boolean waitForText(String text) {  
return (waiter.waitForText(text) != null);  
}

/\*\*  
\* 等待指定text内容出现minimumNumberOfMatches次，可以设置超时时间  
\* text 指定文本内容  
\* minimumNumberOfMatches 指定的次数  
\* timeout 超时时间,单位ms  
\* Waits for the specified text to appear.  
\*  
\* @param text the text to wait for, specified as a regular expression  
\* @param minimumNumberOfMatches the minimum number of matches that are expected to be found. {@code 0} means any number of matches  
\* @param timeout the the amount of time in milliseconds to wait  
\* @return {@code true} if text is displayed and {@code false} if it is not displayed before the timeout  
\*/

public boolean waitForText(String text, int minimumNumberOfMatches, long timeout) {  
return (waiter.waitForText(text, minimumNumberOfMatches, timeout) != null);  
}

/\*\*  
\* 等待指定text内容出现minimumNumberOfMatches次，可以设置超时时间,是否刷新列表类控件  
\* text 指定文本内容  
\* minimumNumberOfMatches 指定的次数  
\* timeout 超时时间,单位ms  
\* scroll 是否对可滑动控件进行滑动安装  
\* Waits for the specified text to appear.  
\*  
\* @param text the text to wait for, specified as a regular expression  
\* @param minimumNumberOfMatches the minimum number of matches that are expected to be found. {@code 0} means any number of matches  
\* @param timeout the the amount of time in milliseconds to wait  
\* @param scroll {@code true} if scrolling should be performed  
\* @return {@code true} if text is displayed and {@code false} if it is not displayed before the timeout  
\*/

public boolean waitForText(String text, int minimumNumberOfMatches, long timeout, boolean scroll) {  
return (waiter.waitForText(text, minimumNumberOfMatches, timeout, scroll) != null);  
}

/\*\*  
\* 等待指定text内容出现minimumNumberOfMatches次，可以设置超时时间,是否刷新列表类控件,是否只查找可见控件  
\* text 指定文本内容  
\* minimumNumberOfMatches 指定的次数  
\* timeout 超时时间,单位ms  
\* scroll 是否对可滑动控件进行滑动操作  
\* onlyVisible 是否只对可见的进行查找  
\* Waits for the specified text to appear.  
\*  
\* @param text the text to wait for, specified as a regular expression  
\* @param minimumNumberOfMatches the minimum number of matches that are expected to be found. {@code 0} means any number of matches  
\* @param timeout the the amount of time in milliseconds to wait  
\* @param scroll {@code true} if scrolling should be performed  
\* @param onlyVisible {@code true} if only visible text views should be waited for  
\* @return {@code true} if text is displayed and {@code false} if it is not displayed before the timeout  
\*/

public boolean waitForText(String text, int minimumNumberOfMatches, long timeout, boolean scroll, boolean onlyVisible) {  
return (waiter.waitForText(text, minimumNumberOfMatches, timeout, scroll, onlyVisible, true) != null);  
}

/\*\*  
\* 通过id查找对应的第一个view,超时20s  
\* Waits for a View matching the specified resource id. Default timeout is 20 seconds.  
\*  
\* @param id the R.id of the {@link View} to wait for  
\* @return {@code true} if the {@link View} is displayed and {@code false} if it is not displayed before the timeout  
\*/

public boolean waitForView(int id){  
return waitForView(id, 0, Timeout.getLargeTimeout(), true);  
}

/\*\*  
\* 通过id查找对应的minimumNumberOfMatches个view,可设置超时  
\* id 传入的 View id  
\* minimumNumberOfMatches id对应的控件数量  
\* timeout 超时时间，单位ms  
\* scroll 是否对可滑动控件进行滑动操作  
\* Waits for a View matching the specified resource id.  
\*  
\* @param id the R.id of the {@link View} to wait for  
\* @param minimumNumberOfMatches the minimum number of matches that are expected to be found. {@code 0} means any number of matches  
\* @param timeout the amount of time in milliseconds to wait  
\* @return {@code true} if the {@link View} is displayed and {@code false} if it is not displayed before the timeout  
\*/

public boolean waitForView(int id, int minimumNumberOfMatches, int timeout){  
return waitForView(id, minimumNumberOfMatches, timeout, true);  
}

/\*\*  
\* 通过id查找对应的minimumNumberOfMatches个view,可设置超时,可设置是否对列表类控件滑动刷新  
\* id 传入的 View id  
\* minimumNumberOfMatches id对应的控件数量  
\* timeout 超时时间，单位ms  
\* scroll 是否可以进行滑动操作  
\* Waits for a View matching the specified resource id.  
\*  
\* @param id the R.id of the {@link View} to wait for  
\* @param minimumNumberOfMatches the minimum number of matches that are expected to be found. {@code 0} means any number of matches  
\* @param timeout the amount of time in milliseconds to wait  
\* @param scroll {@code true} if scrolling should be performed  
\* @return {@code true} if the {@link View} is displayed and {@code false} if it is not displayed before the timeout  
\*/

public boolean waitForView(int id, int minimumNumberOfMatches, int timeout, boolean scroll){  
// index从0开始因此减一  
int index = minimumNumberOfMatches-1;  
// 对于小于1的，直接修正为0  
if(index < 1)  
index = 0;

return (waiter.waitForView(id, index, timeout, scroll) != null);  
}

/\*\*  
\* 等待指定类型的View出现  
\* viewClass class类型  
\* Waits for a View matching the specified class. Default timeout is 20 seconds.  
\*  
\* @param viewClass the {@link View} class to wait for  
\* @return {@code true} if the {@link View} is displayed and {@code false} if it is not displayed before the timeout  
\*/

public boolean waitForView(final Class viewClass){

return waiter.waitForView(viewClass, 0, Timeout.getLargeTimeout(), true);  
}

/\*\*  
\* 等待指定的view出现  
\* Waits for the specified View. Default timeout is 20 seconds.  
\*  
\* @param view the {@link View} object to wait for  
\* @return {@code true} if the {@link View} is displayed and {@code false} if it is not displayed before the timeout  
\*/

public boolean waitForView(View view){  
return waiter.waitForView(view);  
}

/\*\*  
\* 等待指定的view出现，可以设置超时和是否滑动刷新，可滑动控件  
\* view 指定的view  
\* timeout 超时时间,单位ms  
\* scroll 是否可以滑动刷新  
\* Waits for the specified View.  
\*  
\* @param view the {@link View} object to wait for  
\* @param timeout the amount of time in milliseconds to wait  
\* @param scroll {@code true} if scrolling should be performed  
\* @return {@code true} if the {@link View} is displayed and {@code false} if it is not displayed before the timeout  
\*/

public boolean waitForView(View view, int timeout, boolean scroll){  
// 默认设置不区分控件是否可见  
boolean checkIsShown = false;  
// 如果设置了不可滑动，那么只查找可见的  
if(!scroll){  
checkIsShown = true;  
}

return waiter.waitForView(view, timeout, scroll, checkIsShown);  
}

/\*\*  
\* 等待指定类型的minimumNumberOfMatches个view出现，可设置超时时间  
\* viewClass 指定的类型  
\* minimumNumberOfMatches 指定的数量  
\* timeout 超时时间，单位ms  
\* Waits for a View matching the specified class.  
\*  
\* @param viewClass the {@link View} class to wait for  
\* @param minimumNumberOfMatches the minimum number of matches that are expected to be found. {@code 0} means any number of matches  
\* @param timeout the amount of time in milliseconds to wait  
\* @return {@code true} if the {@link View} is displayed and {@code false} if it is not displayed before the timeout  
\*/

public boolean waitForView(final Class viewClass, final int minimumNumberOfMatches, final int timeout){  
// 因为计数从0,开始因此数量减一  
int index = minimumNumberOfMatches-1;  
// 设置的值小于1,修正为0  
if(index < 1)  
index = 0;

return waiter.waitForView(viewClass, index, timeout, true);  
}

/\*\*  
\* 等待指定类型的minimumNumberOfMatches个view出现，可设置超时时间,是否可滑动  
\* viewClass 指定的类型  
\* minimumNumberOfMatches 指定的数量  
\* timeout 超时时间，单位ms  
\* scroll 是否可以滑动刷新  
\* Waits for a View matching the specified class.  
\*  
\* @param viewClass the {@link View} class to wait for  
\* @param minimumNumberOfMatches the minimum number of matches that are expected to be found. {@code 0} means any number of matches  
\* @param timeout the amount of time in milliseconds to wait  
\* @param scroll {@code true} if scrolling should be performed  
\* @return {@code true} if the {@link View} is displayed and {@code false} if it is not displayed before the timeout  
\*/

public boolean waitForView(final Class viewClass, final int minimumNumberOfMatches, final int timeout,final boolean scroll){  
// 因为计数从0,开始因此数量减一  
int index = minimumNumberOfMatches-1;  
// 小于1,修正为0  
if(index < 1)  
index = 0;

return waiter.waitForView(viewClass, index, timeout, scroll);  
}

/\*\*  
\* 等待WebView中的指定条件的WebElement出现，超时20s  
\* Waits for a WebElement matching the specified By object. Default timeout is 20 seconds.  
\*  
\* @param by the By object. Examples are: {@code By.id("id")} and {@code By.name("name")}  
\* @return {@code true} if the {@link WebElement} is displayed and {@code false} if it is not displayed before the timeout  
\*/

public boolean waitForWebElement(By by){  
return (waiter.waitForWebElement(by, 0, Timeout.getLargeTimeout(), true) != null);  
}

/\*\*  
\* 等待WebView中的指定条件的WebElement出现，可设置超时时间，是否需要滑动  
\* by 指定的条件  
\* timeout 超时时间，单位 ms  
\* scroll 是否需要滑动  
\* Waits for a WebElement matching the specified By object.  
\*  
\* @param by the By object. Examples are: {@code By.id("id")} and {@code By.name("name")}  
\* @param timeout the the amount of time in milliseconds to wait  
\* @param scroll {@code true} if scrolling should be performed  
\* @return {@code true} if the {@link WebElement} is displayed and {@code false} if it is not displayed before the timeout  
\*/

public boolean waitForWebElement(By by, int timeout, boolean scroll){  
return (waiter.waitForWebElement(by, 0, timeout, scroll) != null);  
}

/\*\*  
\* 等待WebView中的指定条件的WebElement出现minimumNumberOfMatches次，可设置超时时间，是否需要滑动  
\* by 指定的条件  
\* minimumNumberOfMatches 指定的数量  
\* timeout 超时时间，单位 ms  
\* scroll 是否需要滑动  
\* Waits for a WebElement matching the specified By object.  
\*  
\* @param by the By object. Examples are: {@code By.id("id")} and {@code By.name("name")}  
\* @param minimumNumberOfMatches the minimum number of matches that are expected to be found. {@code 0} means any number of matches  
\* @param timeout the the amount of time in milliseconds to wait  
\* @param scroll {@code true} if scrolling should be performed  
\* @return {@code true} if the {@link WebElement} is displayed and {@code false} if it is not displayed before the timeout  
\*/

public boolean waitForWebElement(By by, int minimumNumberOfMatches, int timeout, boolean scroll){  
return (waiter.waitForWebElement(by, minimumNumberOfMatches, timeout, scroll) != null);  
}

/\*\*  
\* 按照给定的Condition判断条件进行等待操作，可设置超时时间  
\* condition 配置的判定规则  
\* timeout 超时时间，单位 ms  
\* Waits for a condition to be satisfied.  
\*  
\* @param condition the condition to wait for  
\* @param timeout the amount of time in milliseconds to wait  
\* @return {@code true} if condition is satisfied and {@code false} if it is not satisfied before the timeout  
\*/

public boolean waitForCondition(Condition condition, final int timeout){  
return waiter.waitForCondition(condition, timeout);  
}

/\*\*  
\* 查找指定文本内容的EditText类型View是否出现  
\* text 指定的文本内容  
\* Searches for a text in the EditText objects currently displayed and returns true if found. Will automatically scroll when needed.  
\*  
\* @param text the text to search for  
\* @return {@code true} if an {@link EditText} displaying the specified text is found or {@code false} if it is not found  
\*/

public boolean searchEditText(String text) {  
return searcher.searchWithTimeoutFor(EditText.class, text, 1, true, false);  
}

/\*\*  
\* 查找指定文本内容的Button类型View是否出现  
\* text 指定的文本内容  
\* Searches for a Button displaying the specified text and returns {@code true} if at least one Button  
\* is found. Will automatically scroll when needed.  
\*  
\* @param text the text to search for. The parameter will be interpreted as a regular expression  
\* @return {@code true} if a {@link Button} displaying the specified text is found and {@code false} if it is not found  
\*/

public boolean searchButton(String text) {  
return searcher.searchWithTimeoutFor(Button.class, text, 0, true, false);  
}

/\*\*  
\* 查找指定文本内容的Button类型View是否出现,可配置是否只查找可见控件  
\* text 指定的文本内容  
\* onlyVisible 是否可见  
\* Searches for a Button displaying the specified text and returns {@code true} if at least one Button  
\* is found. Will automatically scroll when needed.  
\*  
\* @param text the text to search for. The parameter will be interpreted as a regular expression  
\* @param onlyVisible {@code true} if only {@link Button} visible on the screen should be searched  
\* @return {@code true} if a {@link Button} displaying the specified text is found and {@code false} if it is not found  
\*/

public boolean searchButton(String text, boolean onlyVisible) {  
return searcher.searchWithTimeoutFor(Button.class, text, 0, true, onlyVisible);  
}

/\*\*  
\* 查找指定文本内容的ToggleButton类型View是否出现  
\* Searches for a ToggleButton displaying the specified text and returns {@code true} if at least one ToggleButton  
\* is found. Will automatically scroll when needed.  
\*  
\* @param text the text to search for. The parameter will be interpreted as a regular expression  
\* @return {@code true} if a {@link ToggleButton} displaying the specified text is found and {@code false} if it is not found  
\*/

public boolean searchToggleButton(String text) {  
return searcher.searchWithTimeoutFor(ToggleButton.class, text, 0, true, false);  
}

/\*\*  
\* 查找指定文本内容的Button类型View是否有minimumNumberOfMatches个出现  
\* text 指定文本内容  
\* minimumNumberOfMatches 指定的数量  
\* Searches for a Button displaying the specified text and returns {@code true} if the  
\* searched Button is found a specified number of times. Will automatically scroll when needed.  
\*  
\* @param text the text to search for. The parameter will be interpreted as a regular expression  
\* @param minimumNumberOfMatches the minimum number of matches expected to be found. {@code 0} matches means that one or more  
\* matches are expected to be found  
\* @return {@code true} if a {@link Button} displaying the specified text is found a specified number of times and {@code false}  
\* if it is not found  
\*/

public boolean searchButton(String text, int minimumNumberOfMatches) {  
return searcher.searchWithTimeoutFor(Button.class, text, minimumNumberOfMatches, true, false);  
}

/\*\*  
\* 查找指定文本内容的Button类型View是否有minimumNumberOfMatches个出现,可配置是否只查询可见的  
\* text 指定文本内容  
\* minimumNumberOfMatches 指定的数量  
\* onlyVisible 设置是否只是可见的  
\* Searches for a Button displaying the specified text and returns {@code true} if the  
\* searched Button is found a specified number of times. Will automatically scroll when needed.  
\*  
\* @param text the text to search for. The parameter will be interpreted as a regular expression  
\* @param minimumNumberOfMatches the minimum number of matches expected to be found. {@code 0} matches means that one or more  
\* matches are expected to be found  
\* @param onlyVisible {@code true} if only {@link Button} visible on the screen should be searched  
\* @return {@code true} if a {@link Button} displaying the specified text is found a specified number of times and {@code false}  
\* if it is not found  
\*/

public boolean searchButton(String text, int minimumNumberOfMatches, boolean onlyVisible) {  
return searcher.searchWithTimeoutFor(Button.class, text, minimumNumberOfMatches, true, onlyVisible);  
}

/\*\*  
\* 查找指定文本内容的ToggleButton类型View是否有minimumNumberOfMatches个出现  
\* text 指定文本内容  
\* minimumNumberOfMatches 指定的数量  
\* Searches for a ToggleButton displaying the specified text and returns {@code true} if the  
\* searched ToggleButton is found a specified number of times. Will automatically scroll when needed.  
\*  
\* @param text the text to search for. The parameter will be interpreted as a regular expression  
\* @param minimumNumberOfMatches the minimum number of matches expected to be found. {@code 0} matches means that one or more  
\* matches are expected to be found  
\* @return {@code true} if a {@link ToggleButton} displaying the specified text is found a specified number of times and {@code false}  
\* if it is not found  
\*/

public boolean searchToggleButton(String text, int minimumNumberOfMatches) {  
return searcher.searchWithTimeoutFor(ToggleButton.class, text, minimumNumberOfMatches, true, false);  
}

/\*\*  
\* 查找指定文本内容是否出现,超时5s  
\* Searches for the specified text and returns {@code true} if at least one item  
\* is found displaying the expected text. Will automatically scroll when needed.  
\*  
\* @param text the text to search for. The parameter will be interpreted as a regular expression  
\* @return {@code true} if the search string is found and {@code false} if it is not found  
\*/

public boolean searchText(String text) {  
return searcher.searchWithTimeoutFor(TextView.class, text, 0, true, false);  
}

/\*\*  
\* 查找指定文本内容是否出现,超时5s,可配置是否只查找可见的  
\* Searches for the specified text and returns {@code true} if at least one item  
\* is found displaying the expected text. Will automatically scroll when needed.  
\*  
\* @param text the text to search for. The parameter will be interpreted as a regular expression  
\* @param onlyVisible {@code true} if only texts visible on the screen should be searched  
\* @return {@code true} if the search string is found and {@code false} if it is not found  
\*/

public boolean searchText(String text, boolean onlyVisible) {  
return searcher.searchWithTimeoutFor(TextView.class, text, 0, true, onlyVisible);  
}

/\*\*  
\* 查找指定文本内容是否出现minimumNumberOfMatches次,超时5s  
\* Searches for the specified text and returns {@code true} if the searched text is found a specified  
\* number of times. Will automatically scroll when needed.  
\*  
\* @param text the text to search for. The parameter will be interpreted as a regular expression  
\* @param minimumNumberOfMatches the minimum number of matches expected to be found. {@code 0} matches means that one or more  
\* matches are expected to be found  
\* @return {@code true} if text is found a specified number of times and {@code false} if the text  
\* is not found  
\*/

public boolean searchText(String text, int minimumNumberOfMatches) {  
return searcher.searchWithTimeoutFor(TextView.class, text, minimumNumberOfMatches, true, false);  
}

/\*\*  
\* 查找指定文本内容是否出现minimumNumberOfMatches次,超时5s,可设置是否可以滑动可滑动控件进行查找  
\* Searches for the specified text and returns {@code true} if the searched text is found a specified  
\* number of times.  
\*  
\* @param text the text to search for. The parameter will be interpreted as a regular expression.  
\* @param minimumNumberOfMatches the minimum number of matches expected to be found. {@code 0} matches means that one or more  
\* matches are expected to be found  
\* @param scroll {@code true} if scrolling should be performed  
\* @return {@code true} if text is found a specified number of times and {@code false} if the text  
\* is not found  
\*/

public boolean searchText(String text, int minimumNumberOfMatches, boolean scroll) {  
return searcher.searchWithTimeoutFor(TextView.class, text, minimumNumberOfMatches, scroll, false);  
}

/\*\*  
\* 查找指定文本内容是否出现minimumNumberOfMatches次,超时5s,可设置是否可以滑动可滑动控件进行查找,是否只查找可见控件  
\* Searches for the specified text and returns {@code true} if the searched text is found a specified  
\* number of times.  
\*  
\* @param text the text to search for. The parameter will be interpreted as a regular expression.  
\* @param minimumNumberOfMatches the minimum number of matches expected to be found. {@code 0} matches means that one or more  
\* matches are expected to be found  
\* @param scroll {@code true} if scrolling should be performed  
\* @param onlyVisible {@code true} if only texts visible on the screen should be searched  
\* @return {@code true} if text is found a specified number of times and {@code false} if the text  
\* is not found  
\*/

public boolean searchText(String text, int minimumNumberOfMatches, boolean scroll, boolean onlyVisible) {  
return searcher.searchWithTimeoutFor(TextView.class, text, minimumNumberOfMatches, scroll, onlyVisible);  
}

/\*\*  
\* 设置屏幕方向横向或者纵向  
\* Sets the Orientation (Landscape/Portrait) for the current Activity.  
\*  
\* @param orientation the orientation to set. Solo.{@link #LANDSCAPE} for landscape or  
\* Solo.{@link #PORTRAIT} for portrait.  
\*/

public void setActivityOrientation(int orientation)  
{  
activityUtils.setActivityOrientation(orientation);  
}

/\*\*  
\* 获取当前焦点所在activity  
\* Returns the current Activity.  
\*  
\* @return the current Activity  
\*/

public Activity getCurrentActivity() {  
return activityUtils.getCurrentActivity(false);  
}

/\*\*  
\* 检查当前activity name是否是设置的,异常提醒message  
\* message 与传入name不一致时的提示  
\* name activity name  
\* Asserts that the Activity matching the specified name is active.  
\*  
\* @param message the message to display if the assert fails  
\* @param name the name of the {@link Activity} that is expected to be active. Example is: {@code "MyActivity"}  
\*/

public void assertCurrentActivity(String message, String name)  
{  
asserter.assertCurrentActivity(message, name);  
}

/\*\*  
\* 检查当前activity 类型是否是设置的,异常提醒message  
\* message 与传入name不一致时的提示  
\* activityClass activity类型  
\* Asserts that the Activity matching the specified class is active.  
\*  
\* @param message the message to display if the assert fails  
\* @param activityClass the class of the Activity that is expected to be active. Example is: {@code MyActivity.class}  
\*/

@SuppressWarnings("unchecked")  
public void assertCurrentActivity(String message, @SuppressWarnings("rawtypes") Class activityClass)  
{  
asserter.assertCurrentActivity(message, activityClass);

}

/\*\*  
\* 检查当前activity 名字是否是设置的,异常提醒message,可设置是否是最新的  
\* message 与传入name不一致时的提示  
\* name activity name  
\* isNewInstance 为true则等待最新出现的activity,为false则直接获取activity堆栈的栈顶activity做比较  
\* Asserts that the Activity matching the specified name is active, with the possibility to  
\* verify that the expected Activity is a new instance of the Activity.  
\*  
\* @param message the message to display if the assert fails  
\* @param name the name of the Activity that is expected to be active. Example is: {@code "MyActivity"}  
\* @param isNewInstance {@code true} if the expected {@link Activity} is a new instance of the {@link Activity}  
\*/

public void assertCurrentActivity(String message, String name, boolean isNewInstance)  
{  
asserter.assertCurrentActivity(message, name, isNewInstance);  
}

/\*\*  
\* 检查当前activity 类型是否是设置的,异常提醒message,可设置是否是最新的  
\* message 与传入name不一致时的提示  
\* activityClass activity 类型  
\* isNewInstance 为true则等待最新出现的activity,为false则直接获取activity堆栈的栈顶activity做比较  
\* Asserts that the Activity matching the specified class is active, with the possibility to  
\* verify that the expected Activity is a new instance of the Activity.  
\*  
\* @param message the message to display if the assert fails  
\* @param activityClass the class of the Activity that is expected to be active. Example is: {@code MyActivity.class}  
\* @param isNewInstance {@code true} if the expected {@link Activity} is a new instance of the {@link Activity}  
\*/

@SuppressWarnings("unchecked")  
public void assertCurrentActivity(String message, @SuppressWarnings("rawtypes") Class activityClass,  
boolean isNewInstance) {  
asserter.assertCurrentActivity(message, activityClass, isNewInstance);  
}

/\*\*  
\* 检查当前内存是否达到lowmem状态  
\* Asserts that the available memory is not considered low by the system.  
\*/

public void assertMemoryNotLow()  
{  
asserter.assertMemoryNotLow();  
}

/\*\*  
\* 等待弹框出现  
\* Waits for a Dialog to open. Default timeout is 20 seconds.  
\*  
\* @return {@code true} if the {@link android.app.Dialog} is opened before the timeout and {@code false} if it is not opened  
\*/

public boolean waitForDialogToOpen() {  
return dialogUtils.waitForDialogToOpen(Timeout.getLargeTimeout(), true);  
}

/\*\*  
\* 等待弹框关闭  
\* Waits for a Dialog to close. Default timeout is 20 seconds.  
\*  
\* @return {@code true} if the {@link android.app.Dialog} is closed before the timeout and {@code false} if it is not closed  
\*/

public boolean waitForDialogToClose() {  
return dialogUtils.waitForDialogToClose(Timeout.getLargeTimeout());  
}

/\*\*  
\* 等待弹框出现，可设置超时时间，单位 ms  
\* Waits for a Dialog to open.  
\*  
\* @param timeout the amount of time in milliseconds to wait  
\* @return {@code true} if the {@link android.app.Dialog} is opened before the timeout and {@code false} if it is not opened  
\*/

public boolean waitForDialogToOpen(long timeout) {  
return dialogUtils.waitForDialogToOpen(timeout, true);  
}

/\*\*  
\* 等待弹框关闭，可设置超时时间，单位 ms  
\* Waits for a Dialog to close.  
\*  
\* @param timeout the amount of time in milliseconds to wait  
\* @return {@code true} if the {@link android.app.Dialog} is closed before the timeout and {@code false} if it is not closed  
\*/

public boolean waitForDialogToClose(long timeout) {  
return dialogUtils.waitForDialogToClose(timeout);  
}

/\*\*  
\* 按回退按钮  
\* Simulates pressing the hardware back key.  
\*/

public void goBack()  
{  
sender.goBack();  
}

/\*\*  
\* 点击屏幕上的指定坐标点  
\* Clicks the specified coordinates.  
\*  
\* @param x the x coordinate  
\* @param y the y coordinate  
\*/

public void clickOnScreen(float x, float y) {  
sleeper.sleep();  
clicker.clickOnScreen(x, y);  
}

/\*\*  
\* 点击屏幕上的指定坐标点，可以设置点击次数API要求14+  
\* Clicks the specified coordinates rapidly a specified number of times. Requires API level >= 14.  
\*  
\* @param x the x coordinate  
\* @param y the y coordinate  
\* @param numberOfClicks the number of clicks to perform  
\*/

public void clickOnScreen(float x, float y, int numberOfClicks) {  
// 14以下版本抛出异常  
if (android.os.Build.VERSION.SDK\_INT < 14){  
throw new RuntimeException("clickOnScreen(float x, float y, int numberOfClicks) requires API level >= 14");

}  
tapper.generateTapGesture(numberOfClicks, new PointF(x, y));  
}

/\*\*  
\* 长按屏幕上的指定点  
\* Long clicks the specified coordinates.  
\*  
\* @param x the x coordinate  
\* @param y the y coordinate  
\*/

public void clickLongOnScreen(float x, float y) {  
clicker.clickLongOnScreen(x, y, 0);  
}

/\*\*  
\* 长按屏幕上的指定点 可设置长按时间，单位 ms  
\* Long clicks the specified coordinates for a specified amount of time.  
\*  
\* @param x the x coordinate  
\* @param y the y coordinate  
\* @param time the amount of time to long click  
\*/

public void clickLongOnScreen(float x, float y, int time) {  
clicker.clickLongOnScreen(x, y, time);  
}

/\*\*  
\* 点击指定文本内容的Button  
\* Clicks a Button displaying the specified text. Will automatically scroll when needed.  
\*  
\* @param text the text displayed by the {@link Button}. The parameter will be interpreted as a regular expression  
\*/

public void clickOnButton(String text) {  
clicker.clickOn(Button.class, text);

}

/\*\*  
\* 点击指定的第index个ImageButton.  
\* Clicks an ImageButton matching the specified index.  
\*  
\* @param index the index of the {@link ImageButton} to click. 0 if only one is available  
\*/

public void clickOnImageButton(int index) {  
clicker.clickOn(ImageButton.class, index);  
}

/\*\*  
\* 点击指定文本内容的ToggleButton  
\* Clicks a ToggleButton displaying the specified text.  
\*  
\* @param text the text displayed by the {@link ToggleButton}. The parameter will be interpreted as a regular expression  
\*/

public void clickOnToggleButton(String text) {  
clicker.clickOn(ToggleButton.class, text);  
}

/\*\*  
\* 点击指定文本内容的菜单项  
\* Clicks a MenuItem displaying the specified text.  
\*  
\* @param text the text displayed by the MenuItem. The parameter will be interpreted as a regular expression  
\*/

public void clickOnMenuItem(String text)  
{  
clicker.clickOnMenuItem(text);  
}

/\*\*  
\* 点击指定文本内容的菜单项，可设置是否点击子级菜单  
\* Clicks a MenuItem displaying the specified text.  
\*  
\* @param text the text displayed by the MenuItem. The parameter will be interpreted as a regular expression  
\* @param subMenu {@code true} if the menu item could be located in a sub menu  
\*/

public void clickOnMenuItem(String text, boolean subMenu)  
{  
clicker.clickOnMenuItem(text, subMenu);  
}

/\*\*  
\* 点击指定的WebElement  
\* Clicks the specified WebElement.  
\*  
\* @param webElement the WebElement to click  
\*/

public void clickOnWebElement(WebElement webElement){  
if(webElement == null)  
Assert.fail("WebElement is null and can therefore not be clicked!");

clicker.clickOnScreen(webElement.getLocationX(), webElement.getLocationY());  
}

/\*\*  
\* 点击指定条件的第1个WebElement  
\* Clicks a WebElement matching the specified By object.  
\*  
\* @param by the By object. Examples are: {@code By.id("id")} and {@code By.name("name")}  
\*/

public void clickOnWebElement(By by){  
clickOnWebElement(by, 0, true);  
}

/\*\*  
\* 点击符合指定条件的第match个WebElement  
\* Clicks a WebElement matching the specified By object.  
\*  
\* @param by the By object. Examples are: {@code By.id("id")} and {@code By.name("name")}  
\* @param match if multiple objects match, this determines which one to click  
\*/

public void clickOnWebElement(By by, int match){  
clickOnWebElement(by, match, true);  
}

/\*\*  
\* 点击符合指定条件的第match个WebElement.可设置是否要滑动WebView查找 WebElement  
\* Clicks a WebElement matching the specified By object.  
\*  
\* @param by the By object. Examples are: {@code By.id("id")} and {@code By.name("name")}  
\* @param match if multiple objects match, this determines which one to click  
\* @param scroll {@code true} if scrolling should be performed  
\*/

public void clickOnWebElement(By by, int match, boolean scroll){  
clicker.clickOnWebElement(by, match, scroll, config.useJavaScriptToClickWebElements);  
}

/\*\*  
\* 点击Menu中的第n个Item,Item从左往右从上到下，按顺序排列,每行包含3个Item  
\* Presses a MenuItem matching the specified index. Index {@code 0} is the first item in the  
\* first row, Index {@code 3} is the first item in the second row and  
\* index {@code 6} is the first item in the third row.  
\*  
\* @param index the index of the {@link android.view.MenuItem} to press  
\*/

public void pressMenuItem(int index) {  
presser.pressMenuItem(index);  
}

/\*\*  
\* 点击Menu中的第n个Item,Item从左往右从上到下，按顺序排列,可设置每行的Item数量  
\* Presses a MenuItem matching the specified index. Supports three rows with a specified amount  
\* of items. If itemsPerRow equals 5 then index 0 is the first item in the first row,  
\* index 5 is the first item in the second row and index 10 is the first item in the third row.  
\*  
\* @param index the index of the {@link android.view.MenuItem} to press  
\* @param itemsPerRow the amount of menu items there are per row  
\*/

public void pressMenuItem(int index, int itemsPerRow) {  
presser.pressMenuItem(index, itemsPerRow);  
}

/\*\*  
\* 点击软件盘上当前焦点的下一个按键  
\* Presses the soft keyboard next button.  
\*/

public void pressSoftKeyboardNextButton(){  
presser.pressSoftKeyboardNextButton();  
}

/\*\*  
\* 点击第spinnerIndex个 Spinner的第itemIndex个Item  
\* spinnerIndex 指定的Spinner顺序  
\* itemIndex 指定的Item顺序,如果是正值，那么往下移动，负值往上移动  
\* Presses a Spinner (drop-down menu) item.  
\*  
\* @param spinnerIndex the index of the {@link Spinner} menu to use  
\* @param itemIndex the index of the {@link Spinner} item to press relative to the currently selected item.  
\* A Negative number moves up on the {@link Spinner}, positive moves down  
\*/

public void pressSpinnerItem(int spinnerIndex, int itemIndex)  
{  
presser.pressSpinnerItem(spinnerIndex, itemIndex);  
}

/\*\*  
\* 点击指定的view  
\* Clicks the specified View.  
\*  
\* @param view the {@link View} to click  
\*/

public void clickOnView(View view) {  
waiter.waitForView(view, Timeout.getSmallTimeout());  
clicker.clickOnScreen(view);  
}

/\*\*  
\* 点击指定的View,可设置是否需要等待view出现再点击  
\* view 指定的view  
\* immediately true 直接按照view解析的坐标点击，false 先等待view出现再点击  
\* Clicks the specified View.  
\*  
\* @param view the {@link View} to click  
\* @param immediately {@code true} if View should be clicked without any wait  
\*/

public void clickOnView(View view, boolean immediately){  
if(immediately)  
clicker.clickOnScreen(view);  
else{  
waiter.waitForView(view, Timeout.getSmallTimeout());  
clicker.clickOnScreen(view);  
}  
}

/\*\*  
\* 长按指定的view  
\* Long clicks the specified View.  
\*  
\* @param view the {@link View} to long click  
\*/

public void clickLongOnView(View view) {  
waiter.waitForView(view, Timeout.getSmallTimeout());  
clicker.clickOnScreen(view, true, 0);

}

/\*\*  
\* 长按指定的view.可设置长按时间，单位 ms  
\* Long clicks the specified View for a specified amount of time.  
\*  
\* @param view the {@link View} to long click  
\* @param time the amount of time to long click  
\*/

public void clickLongOnView(View view, int time) {  
clicker.clickOnScreen(view, true, time);

}

/\*\*  
\* 点击指定文本内容的View 或者 WebElement  
\* Clicks a View or WebElement displaying the specified  
\* text. Will automatically scroll when needed.  
\*  
\* @param text the text to click. The parameter will be interpreted as a regular expression  
\*/

public void clickOnText(String text) {  
clicker.clickOnText(text, false, 1, true, 0);  
}

/\*\*  
\* 点击指定文本内容的第match个View 或者 WebElement  
\* Clicks a View or WebElement displaying the specified text. Will automatically scroll when needed.  
\*  
\* @param text the text to click. The parameter will be interpreted as a regular expression  
\* @param match if multiple objects match the text, this determines which one to click  
\*/

public void clickOnText(String text, int match) {  
clicker.clickOnText(text, false, match, true, 0);  
}

/\*\*  
\* 点击指定文本内容的第match个View 或者 WebElement.可设置是否滑动刷新内容  
\* Clicks a View or WebElement displaying the specified text.  
\*  
\* @param text the text to click. The parameter will be interpreted as a regular expression  
\* @param match if multiple objects match the text, this determines which one to click  
\* @param scroll {@code true} if scrolling should be performed  
\*/

public void clickOnText(String text, int match, boolean scroll) {  
clicker.clickOnText(text, false, match, scroll, 0);  
}

/\*\*  
\* 长按指定文本内容的View 或者 WebElement  
\* Long clicks a View or WebElement displaying the specified text. Will automatically scroll when needed.  
\*  
\* @param text the text to click. The parameter will be interpreted as a regular expression  
\*/

public void clickLongOnText(String text)  
{  
clicker.clickOnText(text, true, 1, true, 0);  
}

/\*\*  
\* 长按指定文本内容的第match个View 或者 WebElement  
\* Long clicks a View or WebElement displaying the specified text. Will automatically scroll when needed.  
\*  
\* @param text the text to click. The parameter will be interpreted as a regular expression  
\* @param match if multiple objects match the text, this determines which one to click  
\*/

public void clickLongOnText(String text, int match)  
{  
clicker.clickOnText(text, true, match, true, 0);  
}

/\*\*  
\* 长按指定文本内容的第match个View 或者 WebElement.可设置是否滑动刷新内容  
\* Long clicks a View or WebElement displaying the specified text.  
\*  
\* @param text the text to click. The parameter will be interpreted as a regular expression  
\* @param match if multiple objects match the text, this determines which one to click  
\* @param scroll {@code true} if scrolling should be performed  
\*/

public void clickLongOnText(String text, int match, boolean scroll)  
{  
clicker.clickOnText(text, true, match, scroll, 0);  
}

/\*\*  
\* 长按指定文本内容的第match个View 或者 WebElement.可设置长按时间  
\* Long clicks a View or WebElement displaying the specified text.  
\*  
\* @param text the text to click. The parameter will be interpreted as a regular expression  
\* @param match if multiple objects match the text, this determines which one to click  
\* @param time the amount of time to long click  
\*/

public void clickLongOnText(String text, int match, int time)  
{  
clicker.clickOnText(text, true, match, true, time);  
}

/\*\*  
\* 长按指定text内容的第1个View,等待弹框出现，向下 按键index次，再点击回车键.确认  
\* Long clicks a View displaying the specified text and then selects  
\* an item from the context menu that appears. Will automatically scroll when needed.  
\*  
\* @param text the text to click. The parameter will be interpreted as a regular expression  
\* @param index the index of the menu item to press. {@code 0} if only one is available  
\*/

public void clickLongOnTextAndPress(String text, int index) {  
clicker.clickLongOnTextAndPress(text, index);  
}

/\*\*  
\* 点击第index个Button  
\* Clicks a Button matching the specified index.  
\*  
\* @param index the index of the {@link Button} to click. {@code 0} if only one is available  
\*/

public void clickOnButton(int index) {  
clicker.clickOn(Button.class, index);  
}

/\*\*  
\* 点击第index个RadioButton  
\* Clicks a RadioButton matching the specified index.  
\*  
\* @param index the index of the {@link RadioButton} to click. {@code 0} if only one is available  
\*/

public void clickOnRadioButton(int index) {  
clicker.clickOn(RadioButton.class, index);  
}

/\*\*  
\* 点击第index个CheckBox  
\* Clicks a CheckBox matching the specified index.  
\*  
\* @param index the index of the {@link CheckBox} to click. {@code 0} if only one is available  
\*/

public void clickOnCheckBox(int index) {  
clicker.clickOn(CheckBox.class, index);  
}

/\*\*  
\* 点击第index个EditText  
\* Clicks an EditText matching the specified index.  
\*  
\* @param index the index of the {@link EditText} to click. {@code 0} if only one is available  
\*/

public void clickOnEditText(int index) {  
clicker.clickOn(EditText.class, index);  
}

/\*\*  
\* 点击找到的第1个列表的第line行，并返回此行中的所有TextView类型的View  
\* Clicks the specified list line and returns an ArrayList of the TextView objects that  
\* the list line is displaying. Will use the first ListView it finds.  
\*  
\* @param line the line to click  
\* @return an {@code ArrayList} of the {@link TextView} objects located in the list line  
\*/

public ArrayList clickInList(int line) {  
return clicker.clickInList(line);  
}

/\*\*  
\* 点击指定的第index个列表的第line行，可设置是否长按,并返回此行中的所有TextView类型的View  
\* Clicks the specified list line in the ListView matching the specified index and  
\* returns an ArrayList of the TextView objects that the list line is displaying.  
\*  
\* @param line the line to click  
\* @param index the index of the list. {@code 0} if only one is available  
\* @return an {@code ArrayList} of the {@link TextView} objects located in the list line  
\*/

public ArrayList clickInList(int line, int index) {  
return clicker.clickInList(line, index, false, 0);  
}

/\*\*  
\* 点击指定的第1个列表的第line行,并返回此行中的所有TextView类型的View  
\* Long clicks the specified list line and returns an ArrayList of the TextView objects that  
\* the list line is displaying. Will use the first ListView it finds.  
\*  
\* @param line the line to click  
\* @return an {@code ArrayList} of the {@link TextView} objects located in the list line  
\*/

public ArrayList clickLongInList(int line){  
return clicker.clickInList(line, 0, true, 0);  
}

/\*\*  
\* 点击指定的第index个列表的第line行,并返回此行中的所有TextView类型的View  
\* Long clicks the specified list line in the ListView matching the specified index and  
\* returns an ArrayList of the TextView objects that the list line is displaying.  
\*  
\* @param line the line to click  
\* @param index the index of the list. {@code 0} if only one is available  
\* @return an {@code ArrayList} of the {@link TextView} objects located in the list line  
\*/

public ArrayList clickLongInList(int line, int index){  
return clicker.clickInList(line, index, true, 0);  
}

/\*\*  
\* 长按指定的第index个列表的第line行,并返回此行中的所有TextView类型的View,可设置长按时间  
\* Long clicks the specified list line in the ListView matching the specified index and  
\* returns an ArrayList of the TextView objects that the list line is displaying.  
\*  
\* @param line the line to click  
\* @param index the index of the list. {@code 0} if only one is available  
\* @param time the amount of time to long click  
\* @return an {@code ArrayList} of the {@link TextView} objects located in the list line  
\*/

public ArrayList clickLongInList(int line, int index, int time){  
return clicker.clickInList(line, index, true, time);  
}

/\*\*  
\* 点击指定id的ActionBar  
\* Clicks an ActionBarItem matching the specified resource id.  
\*  
\* @param id the R.id of the ActionBar item to click  
\*/

public void clickOnActionBarItem(int id){  
clicker.clickOnActionBarItem(id);  
}

/\*\*  
\* 点击 ActionBar的 Home或Up  
\* Clicks an ActionBar Home/Up button.  
\*/

public void clickOnActionBarHomeButton() {  
clicker.clickOnActionBarHomeButton();  
}

/\*\*  
\* 按住并且拖动到指定位置 fromX 起始X坐标 toX 终点X坐标 fromY 起始Y坐标 toY 终点Y坐标 stepCount 动作拆分成几步  
\* Simulate touching the specified location and dragging it to a new location.  
\*  
\*  
\* @param fromX X coordinate of the initial touch, in screen coordinates  
\* @param toX X coordinate of the drag destination, in screen coordinates  
\* @param fromY Y coordinate of the initial touch, in screen coordinates  
\* @param toY Y coordinate of the drag destination, in screen coordinates  
\* @param stepCount How many move steps to include in the drag  
\*/

public void drag(float fromX, float toX, float fromY, float toY,  
int stepCount) {  
// 隐藏软键盘  
dialogUtils.hideSoftKeyboard(null, false, true);  
// 拖动操作  
scroller.drag(fromX, toX, fromY, toY, stepCount);  
}

/\*\*  
\* 滚动条下滑  
\* Scrolls down the screen.  
\*  
\* @return {@code true} if more scrolling can be performed and {@code false} if it is at the end of  
\* the screen  
\*/

@SuppressWarnings("unchecked")  
public boolean scrollDown() {  
// 判断是否存在滑动类型的控件  
waiter.waitForViews(true, AbsListView.class, ScrollView.class, WebView.class);  
// 往下滑动  
return scroller.scroll(Scroller.DOWN);  
}

/\*\*  
\* 滑动到底部  
\* Scrolls to the bottom of the screen.  
\*/

@SuppressWarnings("unchecked")  
public void scrollToBottom() {  
// 判断是否存在滑动类型的控件  
waiter.waitForViews(true, AbsListView.class, ScrollView.class, WebView.class);  
// 往下滑动到底部  
scroller.scroll(Scroller.DOWN, true);  
}

/\*\*  
\* 往上滑动  
\* Scrolls up the screen.  
\*  
\* @return {@code true} if more scrolling can be performed and {@code false} if it is at the top of  
\* the screen  
\*/

@SuppressWarnings("unchecked")  
public boolean scrollUp(){  
// 判断是否存在滑动类型的控件  
waiter.waitForViews(true, AbsListView.class, ScrollView.class, WebView.class);  
// 往上滑动  
return scroller.scroll(Scroller.UP);  
}

/\*\*  
\* 往上滑动到顶部  
\* Scrolls to the top of the screen.  
\*/

@SuppressWarnings("unchecked")  
public void scrollToTop() {  
// 判断是否存在滑动类型的控件  
waiter.waitForViews(true, AbsListView.class, ScrollView.class, WebView.class);  
// 往上滑动到顶部  
scroller.scroll(Scroller.UP, true);  
}

/\*\*  
\* 指定列表滚动条往上拖动  
\* Scrolls down the specified AbsListView.  
\*  
\* @param list the {@link AbsListView} to scroll  
\* @return {@code true} if more scrolling can be performed  
\*/

public boolean scrollDownList(AbsListView list) {  
return scroller.scrollList(list, Scroller.DOWN, false);  
}

/\*\*  
\* 指定列表滚动条拖动到顶部  
\* Scrolls to the bottom of the specified AbsListView.  
\*  
\* @param list the {@link AbsListView} to scroll  
\* @return {@code true} if more scrolling can be performed  
\*/

public boolean scrollListToBottom(AbsListView list) {  
return scroller.scrollList(list, Scroller.DOWN, true);  
}

/\*\*  
\* 指定列表滚动条往下拖动  
\* Scrolls up the specified AbsListView.  
\*  
\* @param list the {@link AbsListView} to scroll  
\* @return {@code true} if more scrolling can be performed  
\*/

public boolean scrollUpList(AbsListView list) {  
return scroller.scrollList(list, Scroller.UP, false);  
}

/\*\*  
\* 指定列表滚动条拖动到底部  
\* Scrolls to the top of the specified AbsListView.  
\*  
\* @param list the {@link AbsListView} to scroll  
\* @return {@code true} if more scrolling can be performed  
\*/

public boolean scrollListToTop(AbsListView list) {  
return scroller.scrollList(list, Scroller.UP, true);  
}

/\*\*  
\* 拖动第index个列表类控件,往上拖动  
\* Scrolls down a ListView matching the specified index.  
\*  
\* @param index the index of the {@link ListView} to scroll. {@code 0} if only one list is available  
\* @return {@code true} if more scrolling can be performed  
\*/

public boolean scrollDownList(int index) {  
return scroller.scrollList(waiter.waitForAndGetView(index, ListView.class), Scroller.DOWN, false);  
}

/\*\*  
\* 拖动第index个列表类控件拖动到顶部  
\* Scrolls a ListView matching the specified index to the bottom.  
\*  
\* @param index the index of the {@link ListView} to scroll. {@code 0} if only one list is available  
\* @return {@code true} if more scrolling can be performed  
\*/

public boolean scrollListToBottom(int index) {  
return scroller.scrollList(waiter.waitForAndGetView(index, ListView.class), Scroller.DOWN, true);  
}

/\*\*  
\* 拖动第index个列表类控件,往下拖动  
\* Scrolls up a ListView matching the specified index.  
\*  
\* @param index the index of the {@link ListView} to scroll. {@code 0} if only one list is available  
\* @return {@code true} if more scrolling can be performed  
\*/

public boolean scrollUpList(int index) {  
return scroller.scrollList(waiter.waitForAndGetView(index, ListView.class), Scroller.UP, false);  
}

/\*\*  
\* 拖动第index个列表类控件拖动到底部  
\* Scrolls a ListView matching the specified index to the top.  
\*  
\* @param index the index of the {@link ListView} to scroll. {@code 0} if only one list is available  
\* @return {@code true} if more scrolling can be performed  
\*/

public boolean scrollListToTop(int index) {  
return scroller.scrollList(waiter.waitForAndGetView(index, ListView.class), Scroller.UP, true);  
}

/\*\*  
\* 拖动列表内容到指定的行  
\* Scroll the specified AbsListView to the specified line.  
\*  
\* @param absListView the {@link AbsListView} to scroll  
\* @param line the line to scroll to  
\*/

public void scrollListToLine(AbsListView absListView, int line){  
scroller.scrollListToLine(absListView, line);  
}

/\*\*  
\* 拖动指定的第index列表内容到指定的行  
\* Scroll a AbsListView matching the specified index to the specified line.  
\*  
\* @param index the index of the {@link AbsListView} to scroll  
\* @param line the line to scroll to  
\*/

public void scrollListToLine(int index, int line){  
scroller.scrollListToLine(waiter.waitForAndGetView(index, AbsListView.class), line);  
}

/\*\*  
\* 按照给定方向左右滑动，可指定滑动比例  
\* Scrolls horizontally.  
\*  
\* @param side the side to scroll; {@link #RIGHT} or {@link #LEFT}  
\* @param scrollPosition the position to scroll to, from 0 to 1 where 1 is all the way. Example is: 0.60.  
\*/

public void scrollToSide(int side, float scrollPosition) {  
switch (side){  
case RIGHT: scroller.scrollToSide(Scroller.Side.RIGHT, scrollPosition); break;  
case LEFT: scroller.scrollToSide(Scroller.Side.LEFT, scrollPosition); break;  
}  
}

/\*\*  
\* 上下滑动  
\* Scrolls horizontally.  
\*  
\* @param side the side to scroll; {@link #RIGHT} or {@link #LEFT}  
\*/

public void scrollToSide(int side) {  
switch (side){  
case RIGHT: scroller.scrollToSide(Scroller.Side.RIGHT, 0.60F); break;  
case LEFT: scroller.scrollToSide(Scroller.Side.LEFT, 0.60F); break;  
}  
}

/\*\*  
\* 对指定View进行左右滑动  
\* Scrolls a View horizontally.  
\*  
\* @param view the View to scroll  
\* @param side the side to scroll; {@link #RIGHT} or {@link #LEFT}  
\* @param scrollPosition the position to scroll to, from 0 to 1 where 1 is all the way. Example is: 0.60.  
\*/

public void scrollViewToSide(View view, int side, float scrollPosition) {  
switch (side){  
case RIGHT: scroller.scrollViewToSide(view, Scroller.Side.RIGHT, scrollPosition); break;  
case LEFT: scroller.scrollViewToSide(view, Scroller.Side.LEFT, scrollPosition); break;  
}  
}

/\*\*  
\* 对指定View进行左右滑动,可设置滑动比例  
\* Scrolls a View horizontally.  
\*  
\* @param view the View to scroll  
\* @param side the side to scroll; {@link #RIGHT} or {@link #LEFT}  
\*/

public void scrollViewToSide(View view, int side) {  
switch (side){  
case RIGHT: scroller.scrollViewToSide(view, Scroller.Side.RIGHT, 0.60F); break;  
case LEFT: scroller.scrollViewToSide(view, Scroller.Side.LEFT, 0.60F); break;  
}  
}

/\*\*  
\* 触发缩小放大手势动作,  
\* 开始点比结束点大缩小  
\* 开始点比结束点小放大  
\* API要求14  
\* Zooms in or out if startPoint1 and startPoint2 are larger or smaller then endPoint1 and endPoint2. Requires API level >= 14.  
\*  
\* @param startPoint1 First "finger" down on the screen  
\* @param startPoint2 Second "finger" down on the screen  
\* @param endPoint1 Corresponding ending point of startPoint1  
\* @param endPoint2 Corresponding ending point of startPoint2  
\*/

public void pinchToZoom(PointF startPoint1, PointF startPoint2, PointF endPoint1, PointF endPoint2)  
{  
// API未到14 抛出异常  
if (android.os.Build.VERSION.SDK\_INT < 14){  
throw new RuntimeException("pinchToZoom() requires API level >= 14");  
}  
zoomer.generateZoomGesture(startPoint1, startPoint2, endPoint1, endPoint2);  
}

/\*\*  
\* 划屏手势,2个触控点,API要求14  
\* startPoint1 开始点1  
\* startPoint2 开始点2  
\* endPoint1 结束点1  
\* endPoint2 结束点2  
\*  
\* Swipes with two fingers in a linear path determined by starting and ending points. Requires API level >= 14.  
\*  
\* @param startPoint1 First "finger" down on the screen  
\* @param startPoint2 Second "finger" down on the screen  
\* @param endPoint1 Corresponding ending point of startPoint1  
\* @param endPoint2 Corresponding ending point of startPoint2  
\*/

public void swipe(PointF startPoint1, PointF startPoint2, PointF endPoint1, PointF endPoint2)  
{  
// API未到14抛出异常  
if (android.os.Build.VERSION.SDK\_INT < 14){  
throw new RuntimeException("swipe() requires API level >= 14");  
}  
swiper.generateSwipeGesture(startPoint1, startPoint2, endPoint1,  
endPoint2);  
}

/\*\*  
\* 模拟画圈手势,每次移动步骤3.6度  
\* center1 第一个圈圆心 0-180度  
\* center2 第二个圈圆心 180-540度  
\* Draws two semi-circles at the specified centers. Both circles are larger than rotateSmall(). Requires API level >= 14.  
\*  
\* @param center1 Center of semi-circle drawn from [0, Pi]  
\* @param center2 Center of semi-circle drawn from [Pi, 3\*Pi]  
\*/

public void rotateLarge(PointF center1, PointF center2)  
{  
// API未到14抛出异常  
if (android.os.Build.VERSION.SDK\_INT < 14){  
throw new RuntimeException("rotateLarge(PointF center1, PointF center2) requires API level >= 14");  
}  
rotator.generateRotateGesture(Rotator.LARGE, center1, center2);  
}

/\*\*  
\* 模拟画圈手势,每次移动步骤36度  
\* center1 第一个圈圆心 0-180度  
\* center2 第二个圈圆心 180-540度  
\* Draws two semi-circles at the specified centers. Both circles are smaller than rotateLarge(). Requires API level >= 14.  
\*  
\* @param center1 Center of semi-circle drawn from [0, Pi]  
\* @param center2 Center of semi-circle drawn from [Pi, 3\*Pi]  
\*/

public void rotateSmall(PointF center1, PointF center2)  
{  
// API未到14抛出异常  
if (android.os.Build.VERSION.SDK\_INT < 14){  
throw new RuntimeException("rotateSmall(PointF center1, PointF center2) requires API level >= 14");  
}  
rotator.generateRotateGesture(Rotator.SMALL, center1, center2);  
}

/\*\*  
\* 设置第index个日期控件的日期  
\* Sets the date in a DatePicker matching the specified index.  
\*  
\* @param index the index of the {@link DatePicker}. {@code 0} if only one is available  
\* @param year the year e.g. 2011  
\* @param monthOfYear the month which starts from zero e.g. 0 for January  
\* @param dayOfMonth the day e.g. 10  
\*/

public void setDatePicker(int index, int year, int monthOfYear, int dayOfMonth) {  
setDatePicker(waiter.waitForAndGetView(index, DatePicker.class), year, monthOfYear, dayOfMonth);  
}

/\*\*  
\* 设置指定日期控件的日期  
\* Sets the date in the specified DatePicker.  
\*  
\* @param datePicker the {@link DatePicker} object.  
\* @param year the year e.g. 2011  
\* @param monthOfYear the month which starts from zero e.g. 03 for April  
\* @param dayOfMonth the day e.g. 10  
\*/

public void setDatePicker(DatePicker datePicker, int year, int monthOfYear, int dayOfMonth) {  
waiter.waitForView(datePicker, Timeout.getSmallTimeout());  
setter.setDatePicker(datePicker, year, monthOfYear, dayOfMonth);  
}

/\*\*  
\* 设置第index个时间控件的时间  
\* Sets the time in a TimePicker matching the specified index.  
\*  
\* @param index the index of the {@link TimePicker}. {@code 0} if only one is available  
\* @param hour the hour e.g. 15  
\* @param minute the minute e.g. 30  
\*/

public void setTimePicker(int index, int hour, int minute) {  
setTimePicker(waiter.waitForAndGetView(index, TimePicker.class), hour, minute);  
}

/\*\*  
\* 设置指定时间控件的时间  
\* Sets the time in the specified TimePicker.  
\*  
\* @param timePicker the {@link TimePicker} object.  
\* @param hour the hour e.g. 15  
\* @param minute the minute e.g. 30  
\*/

public void setTimePicker(TimePicker timePicker, int hour, int minute) {  
waiter.waitForView(timePicker, Timeout.getSmallTimeout());  
setter.setTimePicker(timePicker, hour, minute);  
}

/\*\*  
\* 设置第index个进度条的进度  
\* Sets the progress of a ProgressBar matching the specified index. Examples of ProgressBars are: {@link android.widget.SeekBar} and {@link android.widget.RatingBar}.  
\*  
\* @param index the index of the {@link ProgressBar}  
\* @param progress the progress to set the {@link ProgressBar}  
\*/

public void setProgressBar(int index, int progress){  
setProgressBar(waiter.waitForAndGetView(index, ProgressBar.class), progress);  
}

/\*\*  
\* 设置指定进度条的进度  
\* Sets the progress of the specified ProgressBar. Examples of ProgressBars are: {@link android.widget.SeekBar} and {@link android.widget.RatingBar}.  
\*  
\* @param progressBar the {@link ProgressBar}  
\* @param progress the progress to set the {@link ProgressBar}  
\*/

public void setProgressBar(ProgressBar progressBar, int progress){  
waiter.waitForView(progressBar, Timeout.getSmallTimeout());  
setter.setProgressBar(progressBar, progress);  
}

/\*\*  
\* 设置第index个开关的状态  
\* Sets the status of a SlidingDrawer matching the specified index. Examples of status are: {@code Solo.CLOSED} and {@code Solo.OPENED}.  
\*  
\* @param index the index of the {@link SlidingDrawer}  
\* @param status the status to set the {@link SlidingDrawer}  
\*/

public void setSlidingDrawer(int index, int status){  
setSlidingDrawer(waiter.waitForAndGetView(index, SlidingDrawer.class), status);  
}

/\*\*  
\* 设置指定开关的状态  
\* Sets the status of the specified SlidingDrawer. Examples of status are: {@code Solo.CLOSED} and {@code Solo.OPENED}.  
\*  
\* @param slidingDrawer the {@link SlidingDrawer}  
\* @param status the status to set the {@link SlidingDrawer}  
\*/

public void setSlidingDrawer(SlidingDrawer slidingDrawer, int status){  
waiter.waitForView(slidingDrawer, Timeout.getSmallTimeout());  
setter.setSlidingDrawer(slidingDrawer, status);  
}

/\*\*  
\* 设置第index个EditText的文本内容,在原内容上追加.如果传入空值，那么清空原内容  
\* Enters text in an EditText matching the specified index.  
\*  
\* @param index the index of the {@link EditText}. {@code 0} if only one is available  
\* @param text the text to enter in the {@link EditText} field  
\*/

public void enterText(int index, String text) {  
textEnterer.setEditText(waiter.waitForAndGetView(index, EditText.class), text);  
}

/\*\*  
\* 设置指定EditText的文本内容,在原内容上追加.如果传入空值，那么清空原内容  
\* Enters text in the specified EditText.  
\*  
\* @param editText the {@link EditText} to enter text in  
\* @param text the text to enter in the {@link EditText} field  
\*/

public void enterText(EditText editText, String text) {  
waiter.waitForView(editText, Timeout.getSmallTimeout());  
textEnterer.setEditText(editText, text);  
}

/\*\*  
\* 设置指定条件的WebElement的文本内容  
\* Enters text in a WebElement matching the specified By object.  
\*  
\* @param by the By object. Examples are: {@code By.id("id")} and {@code By.name("name")}  
\* @param text the text to enter in the {@link WebElement} field  
\*/

public void enterTextInWebElement(By by, String text){  
if(waiter.waitForWebElement(by, 0, Timeout.getSmallTimeout(), false) == null) {  
Assert.fail("WebElement with " + webUtils.splitNameByUpperCase(by.getClass().getSimpleName()) + ": '" + by.getValue() + "' is not found!");  
}  
webUtils.enterTextIntoWebElement(by, text);  
}

/\*\*  
\* 对第index个 EditText输入内容  
\* Types text in an EditText matching the specified index.  
\*  
\* @param index the index of the {@link EditText}. {@code 0} if only one is available  
\* @param text the text to type in the {@link EditText} field  
\*/

public void typeText(int index, String text) {  
textEnterer.typeText(waiter.waitForAndGetView(index, EditText.class), text);  
}

/\*\*  
\* 对指定的 EditText输入内容  
\* Types text in the specified EditText.  
\*  
\* @param editText the {@link EditText} to type text in  
\* @param text the text to type in the {@link EditText} field  
\*/

public void typeText(EditText editText, String text) {  
waiter.waitForView(editText, Timeout.getSmallTimeout());  
textEnterer.typeText(editText, text);  
}

/\*\*  
\* 对符合条件的第1个WebElement，进行文本内容输入  
\* Types text in a WebElement matching the specified By object.  
\*  
\* @param by the By object. Examples are: {@code By.id("id")} and {@code By.name("name")}  
\* @param text the text to enter in the {@link WebElement} field  
\*/

public void typeTextInWebElement(By by, String text){  
typeTextInWebElement(by, text, 0);  
}

/\*\*  
\* 对符合条件的第match个WebElement,输入文本内容  
\* Types text in a WebElement matching the specified By object.  
\*  
\* @param by the By object. Examples are: {@code By.id("id")} and {@code By.name("name")}  
\* @param text the text to enter in the {@link WebElement} field  
\* @param match if multiple objects match, this determines which one will be typed in  
\*/

public void typeTextInWebElement(By by, String text, int match){  
// 焦点切换到对应的WebElement  
clicker.clickOnWebElement(by, match, true, false);  
// 隐藏软键盘  
dialogUtils.hideSoftKeyboard(null, true, true);  
// 发送键盘内容  
instrumentation.sendStringSync(text);  
}

/\*\*  
\* 指定的WebElement,输入文本内容  
\* Types text in the specified WebElement.  
\*  
\* @param webElement the WebElement to type text in  
\* @param text the text to enter in the {@link WebElement} field  
\*/

public void typeTextInWebElement(WebElement webElement, String text){  
// 焦点到对应的WebElement  
clickOnWebElement(webElement);  
// 隐藏软键盘  
dialogUtils.hideSoftKeyboard(null, true, true);  
// 发送键盘内容  
instrumentation.sendStringSync(text);  
}

/\*\*  
\* 清空第index个EditText文本内容  
\* Clears the value of an EditText.  
\*  
\* @param index the index of the {@link EditText} to clear. 0 if only one is available  
\*/

public void clearEditText(int index) {  
// 输入"",达到清空效果  
textEnterer.setEditText(waiter.waitForAndGetView(index, EditText.class), "");  
}

/\*\*  
\* 清空指定EditText文本内容  
\* Clears the value of an EditText.  
\*  
\* @param editText the {@link EditText} to clear  
\*/

public void clearEditText(EditText editText) {  
waiter.waitForView(editText, Timeout.getSmallTimeout());  
textEnterer.setEditText(editText, "");  
}

/\*\*  
\* 清空指定条件WebElement文本内容  
\* Clears text in a WebElement matching the specified By object.  
\*  
\* @param by the By object. Examples are: {@code By.id("id")} and {@code By.name("name")}  
\*/

public void clearTextInWebElement(By by){  
webUtils.enterTextIntoWebElement(by, "");  
}

/\*\*  
\* 点击第index个图片  
\* Clicks an ImageView matching the specified index.  
\*  
\* @param index the index of the {@link ImageView} to click. {@code 0} if only one is available  
\*/

public void clickOnImage(int index) {  
clicker.clickOn(ImageView.class, index);  
}

/\*\*  
\* 返回第index个EditText  
\* Returns an EditText matching the specified index.  
\*  
\* @param index the index of the {@link EditText}. {@code 0} if only one is available  
\* @return an {@link EditText} matching the specified index  
\*/

public EditText getEditText(int index) {  
return getter.getView(EditText.class, index);  
}

/\*\*  
\* 返回第index个Button  
\* Returns a Button matching the specified index.  
\*  
\* @param index the index of the {@link Button}. {@code 0} if only one is available  
\* @return a {@link Button} matching the specified index  
\*/

public Button getButton(int index) {  
return getter.getView(Button.class, index);  
}

/\*\*  
\* 返回第index个TextView  
\* Returns a TextView matching the specified index.  
\*  
\* @param index the index of the {@link TextView}. {@code 0} if only one is available  
\* @return a {@link TextView} matching the specified index  
\*/

public TextView getText(int index) {  
return getter.getView(TextView.class, index);  
}

/\*\*  
\* 返回第index个ImageView  
\* Returns an ImageView matching the specified index.  
\*  
\* @param index the index of the {@link ImageView}. {@code 0} if only one is available  
\* @return an {@link ImageView} matching the specified index  
\*/

public ImageView getImage(int index) {  
return getter.getView(ImageView.class, index);  
}

/\*\*  
\* 返回第index个ImageButton  
\* Returns an ImageButton matching the specified index.  
\*  
\* @param index the index of the {@link ImageButton}. {@code 0} if only one is available  
\* @return the {@link ImageButton} matching the specified index  
\*/

public ImageButton getImageButton(int index) {  
return getter.getView(ImageButton.class, index);  
}

/\*\*  
\* 返回指定文本内容的TextView  
\* Returns a TextView displaying the specified text.  
\*  
\* @param text the text that is displayed, specified as a regular expression  
\* @return the {@link TextView} displaying the specified text  
\*/

public TextView getText(String text)  
{  
return getter.getView(TextView.class, text, false);  
}

/\*\*  
\* 返回指定文本内容的TextView,可设置是否可见  
\* Returns a TextView displaying the specified text.  
\*  
\* @param text the text that is displayed, specified as a regular expression  
\* @param onlyVisible {@code true} if only visible texts on the screen should be returned  
\* @return the {@link TextView} displaying the specified text  
\*/

public TextView getText(String text, boolean onlyVisible)  
{  
return getter.getView(TextView.class, text, onlyVisible);  
}

/\*\*  
\* 返回指定文本内容的Button  
\* Returns a Button displaying the specified text.  
\*  
\* @param text the text that is displayed, specified as a regular expression  
\* @return the {@link Button} displaying the specified text  
\*/

public Button getButton(String text)  
{  
return getter.getView(Button.class, text, false);  
}

/\*\*  
\* 返回指定文本内容的Button,可设置是否可见  
\* Returns a Button displaying the specified text.  
\*  
\* @param text the text that is displayed, specified as a regular expression  
\* @param onlyVisible {@code true} if only visible buttons on the screen should be returned  
\* @return the {@link Button} displaying the specified text  
\*/

public Button getButton(String text, boolean onlyVisible)  
{  
return getter.getView(Button.class, text, onlyVisible);  
}

/\*\*  
\* 返回指定文本内容的EditText  
\* Returns an EditText displaying the specified text.  
\*  
\* @param text the text that is displayed, specified as a regular expression  
\* @return the {@link EditText} displaying the specified text  
\*/

public EditText getEditText(String text)  
{  
return getter.getView(EditText.class, text, false);  
}

/\*\*  
\* 返回指定文本内容的EditText,可设置是否可见  
\* Returns an EditText displaying the specified text.  
\*  
\* @param text the text that is displayed, specified as a regular expression  
\* @param onlyVisible {@code true} if only visible EditTexts on the screen should be returned  
\* @return the {@link EditText} displaying the specified text  
\*/

public EditText getEditText(String text, boolean onlyVisible)  
{  
return getter.getView(EditText.class, text, onlyVisible);  
}

/\*\*  
\* 返回指定id的第一个View  
\* Returns a View matching the specified resource id.  
\*  
\* @param id the R.id of the {@link View} to return  
\* @return a {@link View} matching the specified id  
\*/

public View getView(int id){  
return getView(id, 0);  
}

/\*\*  
\* 返回第index指定id的View  
\* Returns a View matching the specified resource id and index.  
\*  
\* @param id the R.id of the {@link View} to return  
\* @param index the index of the {@link View}. {@code 0} if only one is available  
\* @return a {@link View} matching the specified id and index  
\*/

public View getView(int id, int index){  
// 查找指定条件的View  
View viewToReturn = getter.getView(id, index);  
// 未找到提示异常  
if(viewToReturn == null) {  
// 按照设置给出提示信息  
int match = index + 1;  
// match大于1说明要找的是第n个  
if(match > 1){  
Assert.fail(match + " Views with id: '" + id + "' are not found!");  
}  
// 标识只找一个  
else {  
Assert.fail("View with id: '" + id + "' is not found!");  
}  
}  
return viewToReturn;  
}

/\*\*  
\* 返回指定string id的第1个View  
\* Returns a View matching the specified resource id.  
\*  
\* @param id the id of the {@link View} to return  
\* @return a {@link View} matching the specified id  
\*/

public View getView(String id){  
return getView(id, 0);  
}

/\*\*  
\* 返回指定string id的第index个View  
\* Returns a View matching the specified resource id and index.  
\*  
\* @param id the id of the {@link View} to return  
\* @param index the index of the {@link View}. {@code 0} if only one is available  
\* @return a {@link View} matching the specified id and index  
\*/

public View getView(String id, int index){  
// 查找指定条件的View  
View viewToReturn = getter.getView(id, index);  
// 未找到提示异常  
if(viewToReturn == null) {  
// 按照设置给出提示信息  
int match = index + 1;  
// match大于1说明要找的是第n个  
if(match > 1){  
Assert.fail(match + " Views with id: '" + id + "' are not found!");  
}  
// 标识只找一个  
else {  
Assert.fail("View with id: '" + id + "' is not found!");  
}  
}  
return viewToReturn;  
}

/\*\*  
\* 返回指定类型的第index个View  
\* Returns a View matching the specified class and index.  
\*  
\* @param viewClass the class of the requested view  
\* @param index the index of the {@link View}. {@code 0} if only one is available  
\* @return a {@link View} matching the specified class and index  
\*/

public T getView(Class viewClass, int index){  
return waiter.waitForAndGetView(index, viewClass);  
}

/\*\*  
\* 返回指定条件的第index个WebElement  
\* Returns a WebElement matching the specified By object and index.  
\*  
\* @param by the By object. Examples are: {@code By.id("id")} and {@code By.name("name")}  
\* @param index the index of the {@link WebElement}. {@code 0} if only one is available  
\* @return a {@link WebElement} matching the specified index  
\*/

public WebElement getWebElement(By by, int index){  
int match = index + 1;  
WebElement webElement = waiter.waitForWebElement(by, match, Timeout.getSmallTimeout(), true);  
// 找不到按照设置条件给出提示  
if(webElement == null) {  
if(match > 1){  
Assert.fail(match + " WebElements with " + webUtils.splitNameByUpperCase(by.getClass().getSimpleName()) + ": '" + by.getValue() + "' are not found!");  
}  
else {  
Assert.fail("WebElement with " + webUtils.splitNameByUpperCase(by.getClass().getSimpleName()) + ": '" + by.getValue() + "' is not found!");  
}  
}  
return webElement;  
}

/\*\*  
\* 获取当前WebView的url地址  
\* Returns the current web page URL.  
\*  
\* @return the current web page URL  
\*/

public String getWebUrl() {  
// 获取当前WebView  
final WebView webView = waiter.waitForAndGetView(0, WebView.class);  
// 如果找不到，提示异常  
if(webView == null)  
Assert.fail("WebView is not found!");  
// 获取url地址  
instrumentation.runOnMainSync(new Runnable() {  
public void run() {  
webUrl = webView.getUrl();  
}  
});  
return webUrl;  
}

/\*\*  
\* 获取当前焦点所在activity中的所有可见view  
\* Returns an ArrayList of the Views currently displayed in the focused Activity or Dialog.  
\*  
\* @return an {@code ArrayList} of the {@link View} objects currently displayed in the  
\* focused window  
\*/

public ArrayList getCurrentViews() {  
return viewFetcher.getViews(null, true);  
}

/\*\*  
\* 获取当前焦点所在activity所有指定类型的view  
\* Returns an ArrayList of Views matching the specified class located in the focused Activity or Dialog.  
\*  
\* @param classToFilterBy return all instances of this class. Examples are: {@code Button.class} or {@code ListView.class}  
\* @return an {@code ArrayList} of {@code View}s matching the specified {@code Class} located in the current {@code Activity}  
\*/

public ArrayList getCurrentViews(Class classToFilterBy) {  
return viewFetcher.getCurrentViews(classToFilterBy);  
}

/\*\*  
\* 获取当前指定parent中的指定类型的view  
\* Returns an ArrayList of Views matching the specified class located under the specified parent.  
\*  
\* @param classToFilterBy return all instances of this class. Examples are: {@code Button.class} or {@code ListView.class}  
\* @param parent the parent {@code View} for where to start the traversal  
\* @return an {@code ArrayList} of {@code View}s matching the specified {@code Class} located under the specified {@code parent}  
\*/

public ArrayList getCurrentViews(Class classToFilterBy, View parent) {  
return viewFetcher.getCurrentViews(classToFilterBy, parent);  
}

/\*\*  
\* 获取当前WebView上所有的WebElement  
\* Returns an ArrayList of WebElements displayed in the active WebView.  
\*  
\* @return an {@code ArrayList} of the {@link WebElement} objects currently displayed in the active WebView  
\*/

public ArrayList getCurrentWebElements(){  
return webUtils.getCurrentWebElements();  
}

/\*\*  
\* 获取当前WebView上指定条件的WebElement  
\* Returns an ArrayList of WebElements displayed in the active WebView matching the specified By object.  
\*  
\* @param by the By object. Examples are: {@code By.id("id")} and {@code By.name("name")}  
\* @return an {@code ArrayList} of the {@link WebElement} objects currently displayed in the active WebView  
\*/

public ArrayList getCurrentWebElements(By by){  
return webUtils.getCurrentWebElements(by);  
}

/\*\*  
\* 检查第index个RadioButton是否是选择状态  
\* Checks if a RadioButton matching the specified index is checked.  
\*  
\* @param index of the {@link RadioButton} to check. {@code 0} if only one is available  
\* @return {@code true} if {@link RadioButton} is checked and {@code false} if it is not checked  
\*/

public boolean isRadioButtonChecked(int index)  
{  
return checker.isButtonChecked(RadioButton.class, index);  
}

/\*\*  
\* 检查指定文本内容的RadioButton是否是选择状态  
\* Checks if a RadioButton displaying the specified text is checked.  
\*  
\* @param text the text that the {@link RadioButton} displays, specified as a regular expression  
\* @return {@code true} if a {@link RadioButton} matching the specified text is checked and {@code false} if it is not checked  
\*/

public boolean isRadioButtonChecked(String text)  
{  
return checker.isButtonChecked(RadioButton.class, text);  
}

/\*\*  
\* 检查第index个CheckBox是否是选择状态  
\* Checks if a CheckBox matching the specified index is checked.  
\*  
\* @param index of the {@link CheckBox} to check. {@code 0} if only one is available  
\* @return {@code true} if {@link CheckBox} is checked and {@code false} if it is not checked  
\*/

public boolean isCheckBoxChecked(int index)  
{  
return checker.isButtonChecked(CheckBox.class, index);  
}

/\*\*  
\* 检查指定文本内容的ToggleButton是否是选择状态  
\* Checks if a ToggleButton displaying the specified text is checked.  
\*  
\* @param text the text that the {@link ToggleButton} displays, specified as a regular expression  
\* @return {@code true} if a {@link ToggleButton} matching the specified text is checked and {@code false} if it is not checked  
\*/

public boolean isToggleButtonChecked(String text)  
{  
return checker.isButtonChecked(ToggleButton.class, text);  
}

/\*\*  
\* 检查第index个ToggleButton是否是选择状态  
\* Checks if a ToggleButton matching the specified index is checked.  
\*  
\* @param index of the {@link ToggleButton} to check. {@code 0} if only one is available  
\* @return {@code true} if {@link ToggleButton} is checked and {@code false} if it is not checked  
\*/

public boolean isToggleButtonChecked(int index)  
{  
return checker.isButtonChecked(ToggleButton.class, index);  
}

/\*\*  
\* 检查指定文本内容的CheckBox是否是选择状态  
\* Checks if a CheckBox displaying the specified text is checked.  
\*  
\* @param text the text that the {@link CheckBox} displays, specified as a regular expression  
\* @return {@code true} if a {@link CheckBox} displaying the specified text is checked and {@code false} if it is not checked  
\*/

public boolean isCheckBoxChecked(String text)  
{  
return checker.isButtonChecked(CheckBox.class, text);  
}

/\*\*  
\* 检查指定文本内容可选择类控件是否被选中 CheckedTextView CompoundButton  
\* Checks if the specified text is checked.  
\*  
\* @param text the text that the {@link CheckedTextView} or {@link CompoundButton} objects display, specified as a regular expression  
\* @return {@code true} if the specified text is checked and {@code false} if it is not checked  
\*/

@SuppressWarnings("unchecked")  
public boolean isTextChecked(String text){  
// 查找是否存在 CheckedTextView CompoundButton  
waiter.waitForViews(false, CheckedTextView.class, CompoundButton.class);  
// 检查选中状态  
if(viewFetcher.getCurrentViews(CheckedTextView.class).size() > 0 && checker.isCheckedTextChecked(text))  
return true;  
// 检查选中状态  
if(viewFetcher.getCurrentViews(CompoundButton.class).size() > 0 && checker.isButtonChecked(CompoundButton.class, text))  
return true;

return false;  
}

/\*\*  
\* 检查指定的文本内容是否是选中状态 Spinner类型控件  
\* Checks if the specified text is selected in any Spinner located in the current screen.  
\*  
\* @param text the text that is expected to be selected, specified as a regular expression  
\* @return {@code true} if the specified text is selected in any {@link Spinner} and false if it is not  
\*/

public boolean isSpinnerTextSelected(String text)  
{  
return checker.isSpinnerTextSelected(text);  
}

/\*\*  
\* 指定的第index个Spinner的指定文本内容是否被选择  
\* Checks if the specified text is selected in a Spinner matching the specified index.  
\*  
\* @param index the index of the spinner to check. {@code 0} if only one spinner is available  
\* @param text the text that is expected to be selected, specified as a regular expression  
\* @return {@code true} if the specified text is selected in the specified {@link Spinner} and false if it is not  
\*/

public boolean isSpinnerTextSelected(int index, String text)  
{  
return checker.isSpinnerTextSelected(index, text);  
}

/\*\*  
\* 隐藏软键盘  
\* Hides the soft keyboard.  
\*/

public void hideSoftKeyboard() {  
dialogUtils.hideSoftKeyboard(null, true, false);  
}

/\*\*  
\* 发送按键  
\* Sends a key: Right, Left, Up, Down, Enter, Menu or Delete.  
\*  
\* @param key the key to be sent. Use {@code Solo.}{@link #RIGHT}, {@link #LEFT}, {@link #UP}, {@link #DOWN},  
\* {@link #ENTER}, {@link #MENU}, {@link #DELETE}  
\*/

public void sendKey(int key)  
{  
sender.sendKeyCode(key);  
}

/\*\*  
\* 返回到指定的名字的activity  
\* Returns to an Activity matching the specified name.  
\*  
\* @param name the name of the {@link Activity} to return to. Example is: {@code "MyActivity"}  
\*/

public void goBackToActivity(String name) {  
activityUtils.goBackToActivity(name);  
}

/\*\*  
\* 等待指定名字的activity出现，超时20s  
\* Waits for an Activity matching the specified name. Default timeout is 20 seconds.  
\*  
\* @param name the name of the {@code Activity} to wait for. Example is: {@code "MyActivity"}  
\* @return {@code true} if {@code Activity} appears before the timeout and {@code false} if it does not  
\*/

public boolean waitForActivity(String name){  
return waiter.waitForActivity(name, Timeout.getLargeTimeout());  
}

/\*\*  
\* 等待指定名字的activity出现，可设置超时时间，单位 ms  
\* Waits for an Activity matching the specified name.  
\*  
\* @param name the name of the {@link Activity} to wait for. Example is: {@code "MyActivity"}  
\* @param timeout the amount of time in milliseconds to wait  
\* @return {@code true} if {@link Activity} appears before the timeout and {@code false} if it does not  
\*/

public boolean waitForActivity(String name, int timeout)  
{  
return waiter.waitForActivity(name, timeout);  
}

/\*\*  
\* 等待指定类型的activity出现,超时20s  
\* Waits for an Activity matching the specified class. Default timeout is 20 seconds.  
\*  
\* @param activityClass the class of the {@code Activity} to wait for. Example is: {@code MyActivity.class}  
\* @return {@code true} if {@code Activity} appears before the timeout and {@code false} if it does not  
\*/

public boolean waitForActivity(Class activityClass){  
return waiter.waitForActivity(activityClass, Timeout.getLargeTimeout());  
}

/\*\*  
\* 等待指定类型的activity出现,可设置超时时间,单位 ms  
\* Waits for an Activity matching the specified class.  
\*  
\* @param activityClass the class of the {@code Activity} to wait for. Example is: {@code MyActivity.class}  
\* @param timeout the amount of time in milliseconds to wait  
\* @return {@code true} if {@link Activity} appears before the timeout and {@code false} if it does not  
\*/

public boolean waitForActivity(Class activityClass, int timeout)  
{  
return waiter.waitForActivity(activityClass, timeout);  
}

/\*\*  
\* 等待activity堆栈为空,可设置超时时间,单位 ms  
\* Wait for the activity stack to be empty.  
\*  
\* @param timeout the amount of time in milliseconds to wait  
\* @return {@code true} if activity stack is empty before the timeout and {@code false} if it is not  
\*/

public boolean waitForEmptyActivityStack(int timeout)  
{  
return waiter.waitForCondition(  
new Condition(){  
@Override  
public boolean isSatisfied() {  
return activityUtils.isActivityStackEmpty();  
}  
}, timeout);  
}

/\*\*  
\* 等待指定tag类型的Fragment出现，超时20s  
\* Waits for a Fragment matching the specified tag. Default timeout is 20 seconds.  
\*  
\* @param tag the name of the tag  
\* @return {@code true} if fragment appears and {@code false} if it does not appear before the timeout  
\*/

public boolean waitForFragmentByTag(String tag){  
return waiter.waitForFragment(tag, 0, Timeout.getLargeTimeout());  
}

/\*\*  
\* 等待指定tag类型的Fragment出现，可设置超时时间，单位 ms  
\* Waits for a Fragment matching the specified tag.  
\*  
\* @param tag the name of the tag  
\* @param timeout the amount of time in milliseconds to wait  
\* @return {@code true} if fragment appears and {@code false} if it does not appear before the timeout  
\*/

public boolean waitForFragmentByTag(String tag, int timeout){  
return waiter.waitForFragment(tag, 0, timeout);  
}

/\*\*  
\* 等待指定id的Fragment出现，超时时间20s  
\* Waits for a Fragment matching the specified resource id. Default timeout is 20 seconds.  
\*  
\* @param id the R.id of the fragment  
\* @return {@code true} if fragment appears and {@code false} if it does not appear before the timeout  
\*/

public boolean waitForFragmentById(int id){  
return waiter.waitForFragment(null, id, Timeout.getLargeTimeout());  
}

/\*\*  
\* 等待指定id的Fragment出现，可设置超时时间  
\* Waits for a Fragment matching the specified resource id.  
\*  
\* @param id the R.id of the fragment  
\* @param timeout the amount of time in milliseconds to wait  
\* @return {@code true} if fragment appears and {@code false} if it does not appear before the timeout  
\*/

public boolean waitForFragmentById(int id, int timeout){  
return waiter.waitForFragment(null, id, timeout);  
}

/\*\*  
\* 等待指定的logcat日志内容出现，超时时间20s 需要 android.permission.READ\_LOGS权限  
\* Waits for the specified log message to appear. Default timeout is 20 seconds.  
\* Requires read logs permission (android.permission.READ\_LOGS) in AndroidManifest.xml of the application under test.  
\*  
\* @param logMessage the log message to wait for  
\* @return {@code true} if log message appears and {@code false} if it does not appear before the timeout  
\*  
\* @see clearLog()  
\*/

public boolean waitForLogMessage(String logMessage){  
return waiter.waitForLogMessage(logMessage, Timeout.getLargeTimeout());  
}

/\*\*  
\* 等待指定的logcat日志内容出现，可设置超时时间，单位ms 需要 android.permission.READ\_LOGS权限  
\* Waits for the specified log message to appear.  
\* Requires read logs permission (android.permission.READ\_LOGS) in AndroidManifest.xml of the application under test.  
\*  
\* @param logMessage the log message to wait for  
\* @param timeout the amount of time in milliseconds to wait  
\* @return {@code true} if log message appears and {@code false} if it does not appear before the timeout  
\*  
\* @see clearLog()  
\*/

public boolean waitForLogMessage(String logMessage, int timeout){  
return waiter.waitForLogMessage(logMessage, timeout);  
}

/\*\*  
\* 清空logcat日志缓存  
\* Clears the log.  
\*/

public void clearLog(){  
waiter.clearLog();  
}

/\*\*  
\* 按照指定资源id，获取当前activity中的 String  
\* Returns a localized String matching the specified resource id.  
\*  
\* @param id the R.id of the String  
\* @return the localized String  
\*/

public String getString(int id)  
{  
return getter.getString(id);  
}

/\*\*  
\* 按照指定资源id，获取当前activity的String.  
\* Returns a localized String matching the specified resource id.  
\*  
\* @param id the id of the String  
\* @return the localized String  
\*/

public String getString(String id)  
{  
return getter.getString(id);  
}

/\*\*  
\* 等待指定时间，单位 ms  
\* Robotium will sleep for the specified time.  
\*  
\* @param time the time in milliseconds that Robotium should sleep  
\*/

public void sleep(int time)  
{  
sleeper.sleep(time);  
}

/\*\*  
\* solo生命周期结束，释放相关资源  
\* Finalizes the Solo object and removes the ActivityMonitor.  
\*  
\* @see #finishOpenedActivities() finishOpenedActivities() to close the activities that have been active  
\*/

public void finalize() throws Throwable {  
activityUtils.finalize();  
}

/\*\*  
\* 关闭所有打开的activity  
\* The Activities that are alive are finished. Usually used in tearDown().  
\*/

public void finishOpenedActivities(){  
activityUtils.finishOpenedActivities();  
}

/\*\*  
\* 截图 需要 android.permission.WRITE\_EXTERNAL\_STORAGE 权限  
\* Takes a screenshot and saves it in the {@link Config} objects save path (default set to: /sdcard/Robotium-Screenshots/).  
\* Requires write permission (android.permission.WRITE\_EXTERNAL\_STORAGE) in AndroidManifest.xml of the application under test.  
\*/

public void takeScreenshot(){  
takeScreenshot(null);  
}

/\*\*  
\* 截图可指定保存名字  
\* Takes a screenshot and saves it with the specified name in the {@link Config} objects save path (default set to: /sdcard/Robotium-Screenshots/).  
\* Requires write permission (android.permission.WRITE\_EXTERNAL\_STORAGE) in AndroidManifest.xml of the application under test.  
\*  
\* @param name the name to give the screenshot  
\*/

public void takeScreenshot(String name){  
takeScreenshot(name, 100);  
}

/\*\*  
\* 截图可指定保存名字,可指定图片质量0-100  
\* Takes a screenshot and saves the image with the specified name in the {@link Config} objects save path (default set to: /sdcard/Robotium-Screenshots/).  
\* Requires write permission (android.permission.WRITE\_EXTERNAL\_STORAGE) in AndroidManifest.xml of the application under test.  
\*  
\* @param name the name to give the screenshot  
\* @param quality the compression rate. From 0 (compress for lowest size) to 100 (compress for maximum quality)  
\*/

public void takeScreenshot(String name, int quality){  
screenshotTaker.takeScreenshot(name, quality);  
}

/\*\*  
\* 使用指定的名字保存图片，连续截图100张，质量80,截图间隔 400ms  
\* 需要 android.permission.WRITE\_EXTERNAL\_STORAGE 权限  
\* Takes a screenshot sequence and saves the images with the specified name prefix in the {@link Config} objects save path (default set to: /sdcard/Robotium-Screenshots/).  
\*  
\* The name prefix is appended with "\_" + sequence\_number for each image in the sequence,  
\* where numbering starts at 0.  
\*  
\* Requires write permission (android.permission.WRITE\_EXTERNAL\_STORAGE) in AndroidManifest.xml of the application under test.  
\*  
\* At present multiple simultaneous screenshot sequences are not supported.  
\* This method will throw an exception if stopScreenshotSequence() has not been  
\* called to finish any prior sequences.  
\* Calling this method is equivalend to calling startScreenshotSequence(name, 80, 400, 100);  
\*  
\* @param name the name prefix to give the screenshot  
\*/

public void startScreenshotSequence(String name) {  
startScreenshotSequence(name,  
80, // quality  
400, // 400 ms frame delay  
100); // max frames  
}

/\*\*  
\* 连续截图  
\* name 截图保存的图片名.会追加\_0---maxFrames-1  
\* quality 截图质量0-100  
\* frameDelay 每次截图时间间隔  
\* maxFrames 截图数量  
\* Takes a screenshot sequence and saves the images with the specified name prefix in the {@link Config} objects save path (default set to: /sdcard/Robotium-Screenshots/).  
\*  
\* The name prefix is appended with "\_" + sequence\_number for each image in the sequence,  
\* where numbering starts at 0.  
\*  
\* Requires write permission (android.permission.WRITE\_EXTERNAL\_STORAGE) in the  
\* AndroidManifest.xml of the application under test.  
\*  
\* Taking a screenshot will take on the order of 40-100 milliseconds of time on the  
\* main UI thread. Therefore it is possible to mess up the timing of tests if  
\* the frameDelay value is set too small.  
\*  
\* At present multiple simultaneous screenshot sequences are not supported.  
\* This method will throw an exception if stopScreenshotSequence() has not been  
\* called to finish any prior sequences.  
\*  
\* @param name the name prefix to give the screenshot  
\* @param quality the compression rate. From 0 (compress for lowest size) to 100 (compress for maximum quality)  
\* @param frameDelay the time in milliseconds to wait between each frame  
\* @param maxFrames the maximum number of frames that will comprise this sequence  
\*/

public void startScreenshotSequence(String name, int quality, int frameDelay, int maxFrames) {  
screenshotTaker.startScreenshotSequence(name, quality, frameDelay, maxFrames);  
}

/\*\*  
\* 停止连续截图  
\* Causes a screenshot sequence to end.  
\*  
\* If this method is not called to end a sequence and a prior sequence is still in  
\* progress, startScreenshotSequence() will throw an exception.  
\*/

public void stopScreenshotSequence() {  
screenshotTaker.stopScreenshotSequence();  
}

/\*\*  
\* 初始化默认最小最大超时  
\* Initialize timeout using 'adb shell setprop' or use setLargeTimeout() and setSmallTimeout(). Will fall back to the default values set by {@link Config}.  
\*/

private void initialize(){  
Timeout.setLargeTimeout(initializeTimeout("solo\_large\_timeout", config.timeout\_large));  
Timeout.setSmallTimeout(initializeTimeout("solo\_small\_timeout", config.timeout\_small));  
}

/\*\*  
\* 获取系统属性，如未设置，则使用默认值  
\* Parse a timeout value set using adb shell.  
\*  
\* There are two options to set the timeout. Set it using adb shell (requires root access):  
\*

\* 'adb shell setprop solo\_large\_timeout milliseconds'  
\*   
  
\* 'adb shell setprop solo\_small\_timeout milliseconds'  
\*   
\* Example: adb shell setprop solo\_small\_timeout 10000  
\*

\* Set the values directly using setLargeTimeout() and setSmallTimeout  
\*  
\* @param property name of the property to read the timeout from  
\* @param defaultValue default value for the timeout  
\* @return timeout in milliseconds  
\*/

@SuppressWarnings({ "rawtypes", "unchecked" })  
private static int initializeTimeout(String property, int defaultValue) {  
try {  
// 反射获取系统变量设置类  
Class clazz = Class.forName("android.os.SystemProperties");  
// 获取获取属性方法  
Method method = clazz.getDeclaredMethod("get", String.class);  
// 获取相关属性  
String value = (String) method.invoke(null, property);  
// 返回找到的值  
return Integer.parseInt(value);  
} catch (Exception e) {  
// 找不到试用默认值  
return defaultValue;  
}  
}  
}