ANDROID TESTING WITH ROBOTIUM DOCUMENTATION

1) Setting up Robotium

- Download Robotium jars from -

https://code.google.com/p/robotium/wiki/Downloads?tm=2

We need to reference the Robotium jar to our project. Right click on project select Build Path, and then click on Configure Build Path option. On Properties window click on Libraries tab and add Robotium latest jar into project.

- Copy Robotium Jars by creating the robotium folder inside the project

- Add robotium jar to the build path

- **Note**: In the latest Android SDK versions(17 or above) a *java.lang.NoClassDefFoundError*:*com.robotium.solo.Solo* error is shown if the Robotium jar is not exported . To fix the issue, after adding the Robotium jar go to the "Order & Export" tab and click the checkbox besides the Robotium Jar and then click "OK".

2) **Writing Test Cases with Robotium**

Basic Structure -

```
private Solo solo;

public  <ActivityName>Test() {
        super(<ActivityName>.class);
}

@Override
protected void setUp() throws Exception {
        super.setUp();
   setSolo(new Solo(getInstrumentation(),getActivity()));
        <ActivityName> = getActivity();
```

```
	}
```

Sample Test with Robotium on PhotoActivity

Test cases covered -

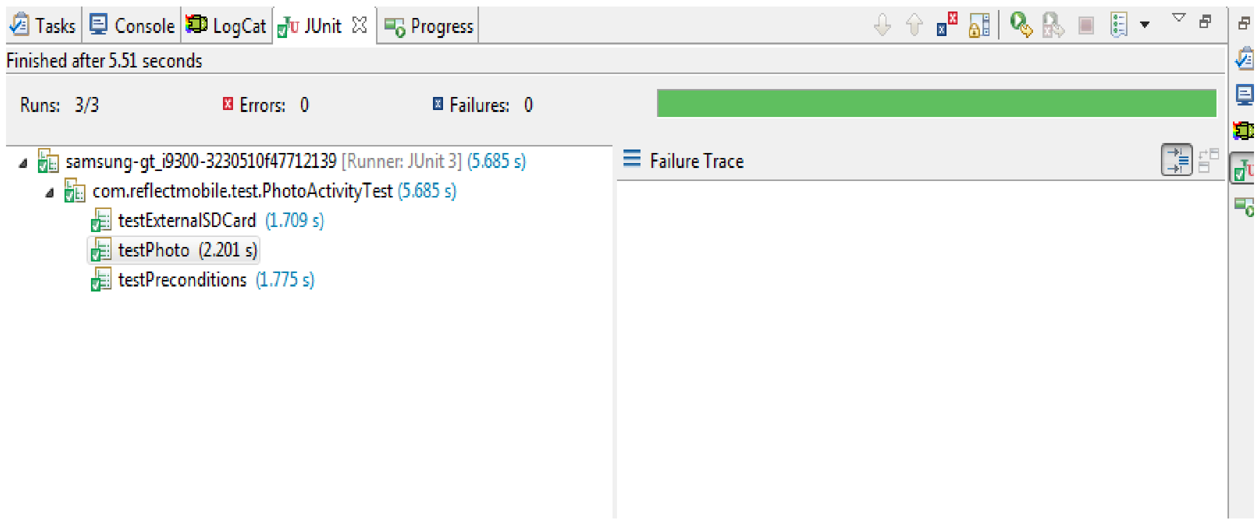
1) Null activity
2) Low memory on device
3) External SD Card mounted

```
/*************************************************************************************

package com.reflectmobile.test;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;

import android.content.Intent;
import android.os.Environment;
import android.test.ActivityInstrumentationTestCase2;
import android.util.Log;

import com.reflectmobile.activity.PhotoActivity;
import com.robotium.solo.Solo;

public class PhotoActivityTest extends
ActivityInstrumentationTestCase2<PhotoActivity> {

	private PhotoActivity photoActivity;
	private Solo solo;

	public PhotoActivityTest() {
		super(PhotoActivity.class);
	}

	@Override
```

```
      protected void setUp() throws Exception {
                  super.setUp();
         setSolo(new Solo(getInstrumentation(),getActivity()));
                  photoActivity = getActivity();

      }

      public void testPreconditions() {
                  assertNotNull("Photo Activity is null", photoActivity);
      }

      public void getSolo() {
                  return solo;
      }

      public void setSolo(Solo solo) {
                  this.solo = solo;
      }

      public void testPhoto(){
                   solo.assertMemoryNotLow();
                  /*assertNotNull("Intent should have triggered after button press",
                        triggeredIntent);
                     String data = triggeredIntent.getExtras().getString("result");
                     assertEquals("Incorrect result data passed via the intent",
                        "Testing Text", data);*/
      }

      public void testExternalSDCard(){
try {
   File sdCard = Environment.getExternalStorageDirectory();
   assertNotNull("SD Card mounted", sdCard);
   Log.d("can write", String.valueOf(sdCard.canWrite()));
   Log.d("ExternalStorageState", Environment.getExternalStorageState());
   File file = new File(sdCard, "VisitedScreen.temp");
   //file.createNewFile();
   FileOutputStream f = new FileOutputStream(file);
    byte[] buf = "Hello".getBytes();
    f.write(buf);
    f.close();
} catch (FileNotFoundException e) {
   e.printStackTrace();
} catch (IOException e) {
        e.printStackTrace();
      }
    }

          @Override
          protected void tearDown() throws Exception{
          solo.finishOpenedActivities();
          }



}
```

**/

OUTPUT



Finished after 5.51 seconds

Runs: 3/3 Errors: 0 Failures: 0

samsung-gt_i9300-3230510f47712139 [Runner: JUnit 3] (5.685 s)
com.reflectmobile.test.PhotoActivityTest (5.685 s)
testExternalSDCard (1.709 s)
testPhoto (2.201 s)
testPreconditions (1.775 s)

ISSUES WHILE SETTING & WORKING WITH ROBOTIUM

When working with Robotium one may encounter issues that are due to the SDK, Eclipse or how Robotium works.

1. NullPointerException when creating a test project for the first time
This problem comes, when they for the first time create a new workspace and start working with Robotium.
If we create a new workspace I, when creating the Test Project, Eclipse will show error

'java.lang.NullPointerException', because it is a new workspace and eclipse is not able to get selected Android resources for first test project.

We can solve this issue by following any of the suggestions below:

1. Create another TestProject & eclipse will automatically get selected resources(for the new one) & not
show error for second created Test Project(we can use the second one for our work)
OR
2. First create an Android Project (we can also use Android Sample application bundled with Android SDK), run it, then create Test Project will not show error

2. java.lang.NoClassDefFoundError:com.robotium.solo.Solo
In the latest Android SDK versions (17 or above) java.lang.NoClassDefFoundError: com.robotium.solo.Solo error is shown if the Robotium jar is not exported. To fix the issue, after adding the
Robotium jar go to the "Order & Export" tab and click the checkbox
besides the Robotium Jar and then
click "OK". Please see the screenshots below.

3. **Robotium does not work with all the activities**

Important to know that Robotium runs in the same process as the application under test. Due to this Robotium only works with the activities and views within the defined package (see below).
In the AndroidManifest.xml we use the following code to describe which application to test:
<instrumentation
android:targetPackage="com.app.ReflectMobileTest"
android:name="android.test.InstrumentationTestReflectMobile" />
“com.app.reflectmobiletest” is the package name of the application under test. Robotium is locked to this package.

4. **Unstable test cases**

Its important to continuously use the waitFor methods, especially if new windows open or loading screens are shown. The waitFor methods tell Robotium to wait for a condition to happen before the execution continues.

Example - On the below listed wait methods:

waitForActivity(String name)
Waits for the given Activity.
waitForActivity(String name, int timeout)

Waits for the given Activity.
waitForDialogToClose(long timeout)

5. Hanging/Freezing test cases

During test execution we open multiple activities, which stay alive depending upon the free memory available.Therefore use solo.finishOpenedActivities() in your tearDown method, it will close all the opened activities and free resources for the following test executions.

```
@AfterClass
public void tearDown() throws Exception
{
solo.finishOpenedActivities();
}
```