Agile Record

The Magazine for Agile Developers and Agile Testers

www.agilerecord.com free digital version made in Germany ISSN 2191-1320

July 2011

issue 7

Robotium @ XING. Automated regression tests on mobile Android devices

by Daniel Knott

Automated tests may reduce the amount of effort required in Agile software development teams, provided they are used in the right way. As mentioned in a previous XING blog entry, the Quality Assurance department at XING has a strong focus on test automation [XING09]. Every code change or new functionality could affect our existing features and their behavior. As there is never usually enough time for manual regression testing, which is also highly inefficient, we have always used Selenium, TestNG and Java to perform automated regression tests, which are in turn limited to XING's web platform.

According to a new forecast from BITKOM, more than 10 million smartphones will be sold in 2011 in Germany alone [BITK10]. This increase in devices goes hand in hand with a rise in mobile internet usage. So it is really important for XING to have high-quality apps in the respective app stores so as to provide the customers with access to the platform while out and about.

To fulfill these requirements, XING has a team dedicated to mobile applications such as the iPhone app, the Android app and the mobile web app touch.xing.com.

The high level of diversification of Android devices poses a special challenge for the Quality Assurance team, as the market is fragmented with a number of different vendors and their customized user interfaces. There are various Android software versions that can be installed on devices like smartphones or tablets, and the XING app needs to cover all of them.

As things stand, the XING Android app is used by customers running Android versions 2.1, 2.2 and 2.3, which are mostly installed on the devices HTC Desire/ HD, Samsung Galaxy S, Galaxy Tab, HTC Legend, HTC Wildfire, Motorola Milestone and LG Optimus.

Another requirement that affects quality assurance is the hardware performance of the devices. The mere fact that the app works well on current devices like the HTC Desire HD does not mean that the app will also work well on older devices with lower performance such as the HTC Legend.

Besides the problems with software and hardware, another parameter that makes app testing a lot more complex is language. The app could be used in different languages, and all text resources must be correctly translated into these languages. Text resources also have to fit on devices with differing screen sizes.

These three factors, i.e. software, hardware and language, make it impossible to manually test every code change on the devices with different Android software versions and language settings, as the amount of work involved is simply too high.

To solve these problems, automated regression tests are necessary to deliver an app with good quality that runs on every device with various settings. Up to now, the XING Android app was tested using automated unit and manual functional tests.

Robotium

The Robotium framework is used to develop a regression test suite for the XING Android app [ROBO11]. Robotium is a "Black Box" testing tool that is able to simulate and automate user interaction such as touching, clicking, text entry and any other gesture that is possible on a touch device. The tests could either be executed on the Android simulator (AVD – Android Virtual Device) or on a real device. Executing such tests on real devices has the major advantage that the app is running on real hardware within a real environment, so potential performance problems can be identified at an early stage with this technique.

Robotium is built on the Java programming language and the JUnit test framework. As mentioned before, Robotium is a "Black Box" testing tool, so you don't need any further information about the Android app's structure or implemented classes. All you need is the name of the main class and the path that links to it.

To develop stable and reliable tests, Robotium offers many methods that react to different graphical elements within an Android app, such as:

- clickOnText("Secure Login");
- clickOnButton("Save");
- searchText("Logout");
- goBack();
- getButton();
- isRadioButtonChecked();
- ...

With these simple methods, robust automated tests can be implemented really quickly. If you combine them with JUnit, you have additional ways of checking values or user interactions on the device for the correct response, which in turn makes these tests even more powerful.

Getting started. What's required?

The following software is required to implement automated tests for Android apps:

- Eclipse-IDE
- JAVA SDK (Software Development Kit)
- ADT (Android Development Tools)
- Robotium
- XING Android App

After installing all the components, a new *Android Test Project* can be created within Eclipse (see figure 1).

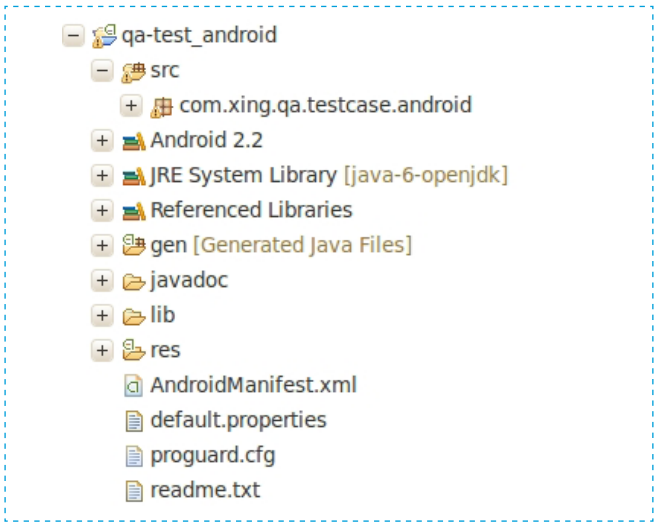

Figure 1

The next step is to adapt the AndroidManifest.xml file (see code listing 1). In this file, the XING Android app package name has to be entered after which Robotium is able to communicate with the app.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/
apk/res/android"
 package="com.xing.qa.testcase.android"
 android:versionCode="1"
 android:versionName="1.0">
 <uses-sdk android:minSdkVersion="8" />
 <instrumentation android:targetPackage="com.xing.
android" android:name="android.test.InstrumentationT-
estRunner" />
 <application android:icon="@drawable/icon"
android:label="@string/app_name">

 <uses-library android:name="android.test.runner" />
 </application>
</manifest>
```

Code listing 1: The AndroidManifest xml file

To start the first Robotium test class, a Java class that extends the Robotium Framework class `ActivityInstrumentationTestCase2` has to be created. This class then provides methods and activities to interact with the app. The core of the automated tests is the Robotium object `Solo`, which provides access to the entire Robotium framework along with all of the provided methods. The first step before using the object is to initialize it in the `setUp()` method (see code listing 2). In this method the object is initialized with central Android activity.

```
@Override
   protected void setUp() throws Exception {
 solo = new Solo(getInstrumentation(), getActivity());
   }
```

Code listing 2: The setUp() method

The really simple example in code listing 3 aims to demonstrate the functionality of Robotium and the `Solo` object. The example shows a test involving the login process. The `testLogin()` method is created to test the login process. Within this method, the following lines of Java code were written:

```
public void testLogin()throws Exception{
   solo.enterText(0, "Testusername");
solo.enterText(1, "secret");
solo.clickOnButton("Secure Login");
solo.waitForActivity("com.xing.android.activities.
SpinnerLoginActivity", 3000);
solo.assertCurrentActivity("Assertion failed, wrong
Activity", "DashboardActivity");
assertTrue(solo.searchText("News"));
}
```

Code listing 3: Simple test method for the login process

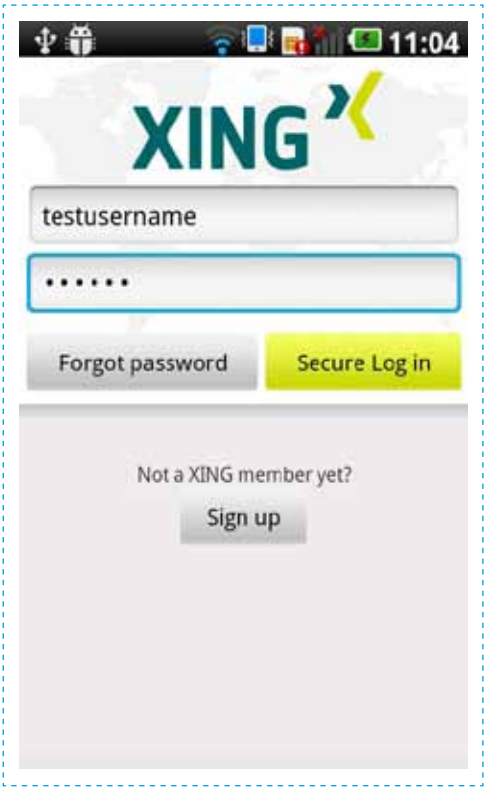

Figure 2

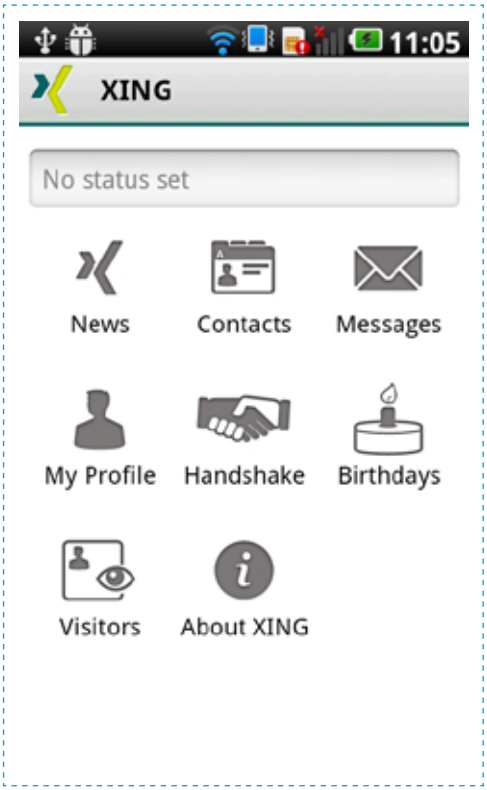

Figure 3

The `solo` object provides access to the `enterText` method, which requires two parameters that allow some text to be entered into a text input field within the app. The first value of the parameters represents the IDs of the input text field on the login screen. The second parameter is the string that should be entered. The `clickOnButton()` method "clicks" the button to login the user (see figure 2).

The `waitForActivity()` method waits for the login activity of the app until the user is logged in. After a successful login, Robotium uses the `assertCurrentActivity()` method to verify if the dashboard activity is shown (see figure 3). At the end of the test method, a JUnit `assertTrue` verification is performed to check whether or not the dashboard and /or "News" button is/are visible.

At the end of a test run, the `tearDown()` method is called to close all activities and to clean up the `solo` object (code listing 4).

```
@Override
public void tearDown() throws Exception {
   try {
         solo.finalize();

   } catch (Throwable e) {
         e.printStackTrace();
   }
   getActivity().finish();
   super.tearDown();
}
```

Code listing 4: The TearDown() method

Once the automated tests have been developed, they can then be run on Android devices. To execute them, start the Robotium test project as Android JUnit test. During the test run, JUnit generates a report, and displays error messages if any problems occur (see figure 4).

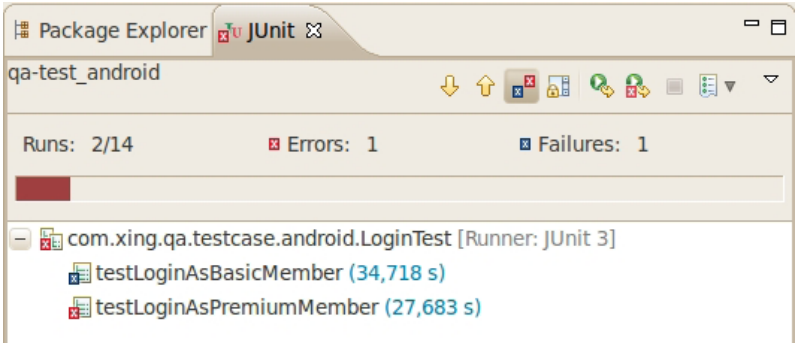

Figure 4

Robotium offers a number of benefits in terms of creating an automated regression test suite for Android devices. Developing stable and powerful automated tests is really easy and can save a lot of time by following a few programming rules. One of the biggest advantages of Robotium is that automated tests can be executed on real devices.

Until now, the core functions of the XING Android app, i.e. the login process, messages, news, visitors, personal profiles and search features, were automated using Robotium. Automated test development is an ongoing process during which every new feature of the app will be automated. Tests will be executed every day to give the Android team the guarantee that existing functions are still working, even after a code change. Changes that lead to errors are now found much earlier in the development process by our agile team, which in turn means faster and more effective development work.

The latest XING Android App is available for download here https://market.android.com/details?id=com.xing.android

References:

- [XING09] Tobias Geyer, *Making sure it still works: Regression Testing at XING*, http://blog.xing.com/2009/12/making-sure-it-still-works-regression-testing-at-xing/
- [BITK10] German Forecast, *Smartphone-Absatz 2011 über der 10-Millionen-Marke*, http://www.bitkom.org/de/presse/66442_65897.aspx
- [ROBO11] Robotium Homepage, http://code.google.com/p/robotium/

> About the author

Daniel Knott

has a technical background with different programming languages and quality assurance tools. After his vocational education at IBM Deutschland GmbH, he studied Computer Science with a focus on quality assurance. Since 2010 Daniel has worked as a Junior Quality Assurance Manager at XING. In his first project he was responsible for the test management, test automation and test execution in a search and recommendation team at XING. Currently, he works in the mobile team, where he is involved in the test management and test automation on Android and iPhone devices. Daniel likes to work in agile software development teams and to automate test cases using technologies such as Robotium, Selenium and Java. His XING profile: https://www.xing.com/profile/Daniel_Knott