Testing Android Applications using Appium and JUnit in Java

Overview

In this tutorial, we will create a JUnit Test Case in Java, using the Appium library for automation of Android applications.

Description

The following automated test is taken from the tutorials for Android provided by Appium.

> **Please note**
>
> This example is found in the public Github repository in https://github.com/appium/tutorial/tree/master/projects/java_android. It also provides examples for other languages.

Requirements

- The Android emulator should be started with a compatible virtual device.

```
emulator @Nexus_5_API_25
```

- Appium must be running in the machine with Android SDK.

```
appium
```

We will make a simple update to the pom.xml file in order to generate a JUnit xml report.

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- based on
https://github.com/appium/appium/blob/2fe7ea7b098ba2145e3c7b4cc31276a3e26921ec/sample-code/examples/java/junit
/pom.xml
-->
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>appium</groupId>
  <artifactId>tutorial_android</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>appium_tutorial_android</name>
  <description>JUnit Android examples</description>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
  </properties>
  <dependencies>
    <dependency>
      <groupId>io.appium</groupId>
      <artifactId>java-client</artifactId>
      <version>2.0.0</version>
    </dependency>
```

```
  <dependency>
    <!-- Must use 1.0.15 or better otherwise upload file will have cookie warnings
    https://github.com/saucelabs/saucerest-java/commit/99bce5b108354ad086ac31e06c1e3ab092000490
    -->
    <groupId>com.saucelabs</groupId>
    <artifactId>saucerest</artifactId>
    <version>1.0.16</version>
  </dependency>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.11</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>com.googlecode.json-simple</groupId>
    <artifactId>json-simple</artifactId>
    <version>1.1</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>commons-lang</groupId>
    <artifactId>commons-lang</artifactId>
    <version>2.6</version>
    <scope>test</scope>
  </dependency>
  <!--
   Sauce JUnit is contained in saucelabs/sauce-java
   https://github.com/saucelabs/sauce-java/tree/master/junit
  -->
  <dependency>
    <groupId>com.saucelabs</groupId>
    <artifactId>sauce_junit</artifactId>
    <version>2.1.3</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.2.4</version>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>2.17</version>
    </plugin>
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <source>1.5</source>
        <target>1.5</target>
      </configuration>
      <version>3.1</version>
    </plugin>

  </plugins>
</build>

<repositories>
  <repository>
    <id>saucelabs-repository</id>
    <url>https://repository-saucelabs.forge.cloudbees.com/release</url>
    <releases>
      <enabled>true</enabled>
    </releases>
    <snapshots>
      <enabled>true</enabled>
```

```
      </snapshots>
    </repository>
  </repositories>

    <reporting>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-report-plugin</artifactId>
            </plugin>
        </plugins>
    </reporting>

</project>
```

The class implementing the automated tests needs to be updated in order to properly set up the IP of the Appium server, along with the required Android version.

AppiumTest.java

```
package appium.tutorial.android.util;

import appium.tutorial.android.page.HomePage;
import com.saucelabs.common.SauceOnDemandAuthentication;
import com.saucelabs.common.SauceOnDemandSessionIdProvider;
import com.saucelabs.junit.SauceOnDemandTestWatcher;
import com.saucelabs.saucerest.SauceREST;
import io.appium.java_client.android.AndroidDriver;
import org.apache.commons.logging.LogFactory;
import org.junit.After;
import org.junit.Before;
import org.junit.Rule;
import org.junit.rules.TestRule;
import org.junit.rules.TestWatcher;
import org.junit.runner.Description;
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.remote.DesiredCapabilities;

import java.io.File;
import java.net.URL;
import java.nio.file.Paths;
import java.util.Date;
import java.util.concurrent.TimeUnit;

import static appium.tutorial.android.util.Helpers.driver;

public class AppiumTest implements SauceOnDemandSessionIdProvider {

    static {
        // Disable annoying cookie warnings.
        // WARNING: Invalid cookie header
        LogFactory.getFactory().setAttribute("org.apache.commons.logging.Log", "org.apache.commons.logging.impl.
NoOpLog");
    }

    /** Page object references. Allows using 'home' instead of 'HomePage' **/
    protected HomePage home;

    /** wait wraps Helpers.wait **/
    public static WebElement wait(By locator) {
        return Helpers.wait(locator);
    }

    private boolean runOnSauce = System.getProperty("sauce") != null;
```

```
/** Authenticate to Sauce with environment variables SAUCE_USER_NAME and SAUCE_API_KEY **/
private SauceOnDemandAuthentication auth = new SauceOnDemandAuthentication();

/** Report pass/fail to Sauce Labs **/
// false to silence Sauce connect messages.
public @Rule
SauceOnDemandTestWatcher reportToSauce = new SauceOnDemandTestWatcher(this, auth, false);

@Rule
public TestRule printTests = new TestWatcher() {
    protected void starting(Description description) {
        System.out.print("  test: " + description.getMethodName());
    }

    protected void finished(Description description) {
        final String session = getSessionId();

        if (session != null) {
            System.out.println(" " + "https://saucelabs.com/tests/" + session);
        } else {
            System.out.println();
        }
    }
};

private String sessionId;

/** Keep the same date prefix to identify job sets. **/
private static Date date = new Date();

/** Run before each test **/
@Before
public void setUp() throws Exception {
    DesiredCapabilities capabilities = new DesiredCapabilities();
    capabilities.setCapability("appium-version", "1.1.0");
    capabilities.setCapability("platformName", "Android");
    capabilities.setCapability("deviceName", "Android");
    capabilities.setCapability("platformVersion", "7.1");

    // Set job name on Sauce Labs
    capabilities.setCapability("name", "Java Android tutorial " + date);
    String userDir = System.getProperty("user.dir");

    URL serverAddress;
    String localApp = "api.apk";
    if (runOnSauce) {
        String user = auth.getUsername();
        String key = auth.getAccessKey();

        // Upload app to Sauce Labs
        SauceREST rest = new SauceREST(user, key);

        rest.uploadFile(new File(userDir, localApp), localApp);

        capabilities.setCapability("app", "sauce-storage:" + localApp);
        serverAddress = new URL("http://" + user + ":" + key + "@ondemand.saucelabs.com:80/wd/hub");
        driver = new AndroidDriver(serverAddress, capabilities);
    } else {
        String appPath = Paths.get(userDir, localApp).toAbsolutePath().toString();
        capabilities.setCapability("app", appPath);
        serverAddress = new URL("http://127.0.0.1:4723/wd/hub");
        driver = new AndroidDriver(serverAddress, capabilities);
    }

    sessionId = driver.getSessionId().toString();

    driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
    Helpers.init(driver, serverAddress);
}

/** Run after each test **/
```

```
    @After
    public void tearDown() throws Exception {
        if (driver != null) driver.quit();
    }

    /** If we're not on Sauce then return null otherwise SauceOnDemandTestWatcher will error. **/
    public String getSessionId() {
        return runOnSauce ? sessionId : null;
    }
}
```

The sample project contains two classes with automated Tests. Below is one of them:

AutomatingASimpleActionTest.java

```
package appium.tutorial.android;

import appium.tutorial.android.util.AppiumTest;
import org.openqa.selenium.WebElement;

import java.util.ArrayList;
import java.util.List;

import static appium.tutorial.android.util.Helpers.*;

public class AutomatingASimpleActionTest extends AppiumTest {

    @org.junit.Test
    public void one() throws Exception {
        text("Accessibility").click();
        text_exact("Accessibility Node Provider");
    }

    @org.junit.Test
    public void two() throws Exception {
        wait(for_text("Accessibility")).click();
        wait(for_text_exact("Accessibility Node Provider"));
    }

    @org.junit.Test
    public void three() throws Exception {
        wait(for_text(2)).click();
        find("Custom Evaluator");
    }

  @org.junit.Test
    public void four() throws Exception {
        setWait(0);

        List<String> cell_names = new ArrayList<String>();

        for (WebElement cell : tags("android.widget.TextView")) {
            cell_names.add(cell.getAttribute("name"));
        }

        // delete title cell
        cell_names.remove(0);

        for (String cell_name : cell_names) {
            scroll_to_exact(cell_name).click();
            waitInvisible(for_text_exact(cell_name));
            back();
            wait(for_find("Accessibility"));
            wait(for_find("Animation"));
        }

        setWait(30); // restore old implicit wait
    }
}
```

Tests can be run using Maven.

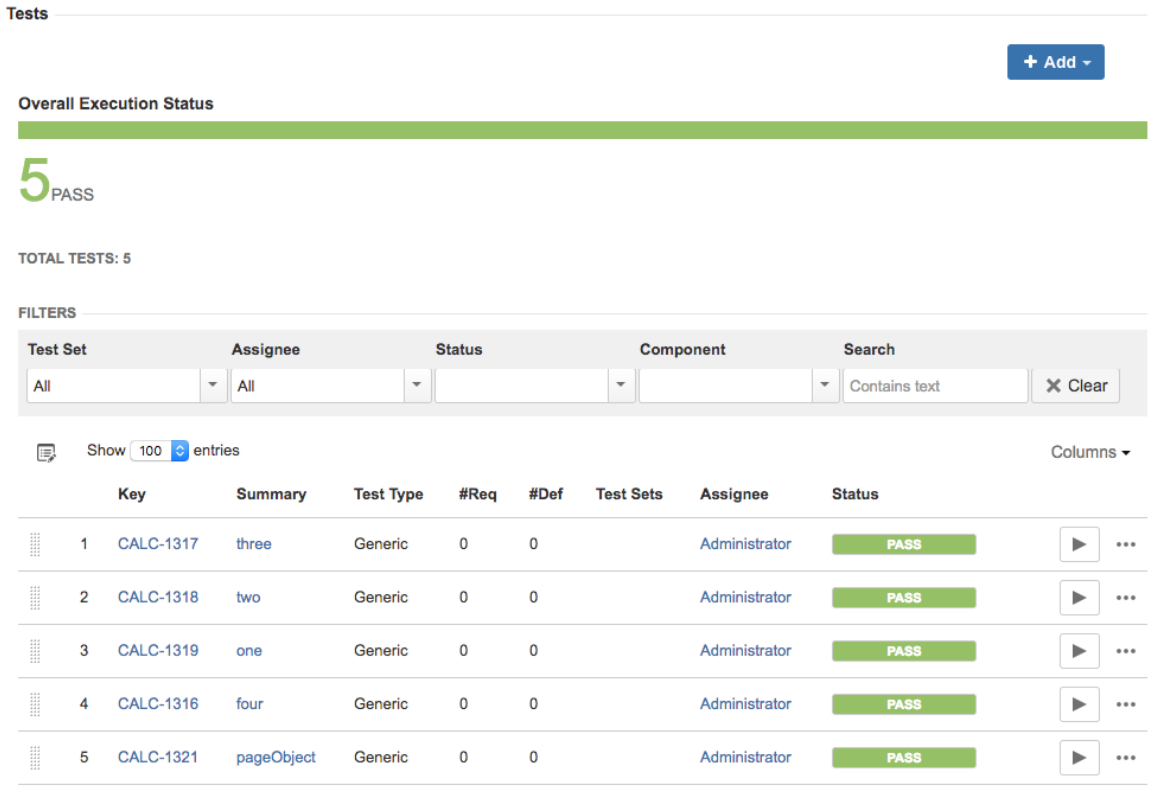
```
mvn clean test
```

Since the previous command generates multiple JUnit XML files, we may need to merge them into a single XML file so it can be submitted into a Test Execution more easily. That can be achieved by using the junit-merge utility.

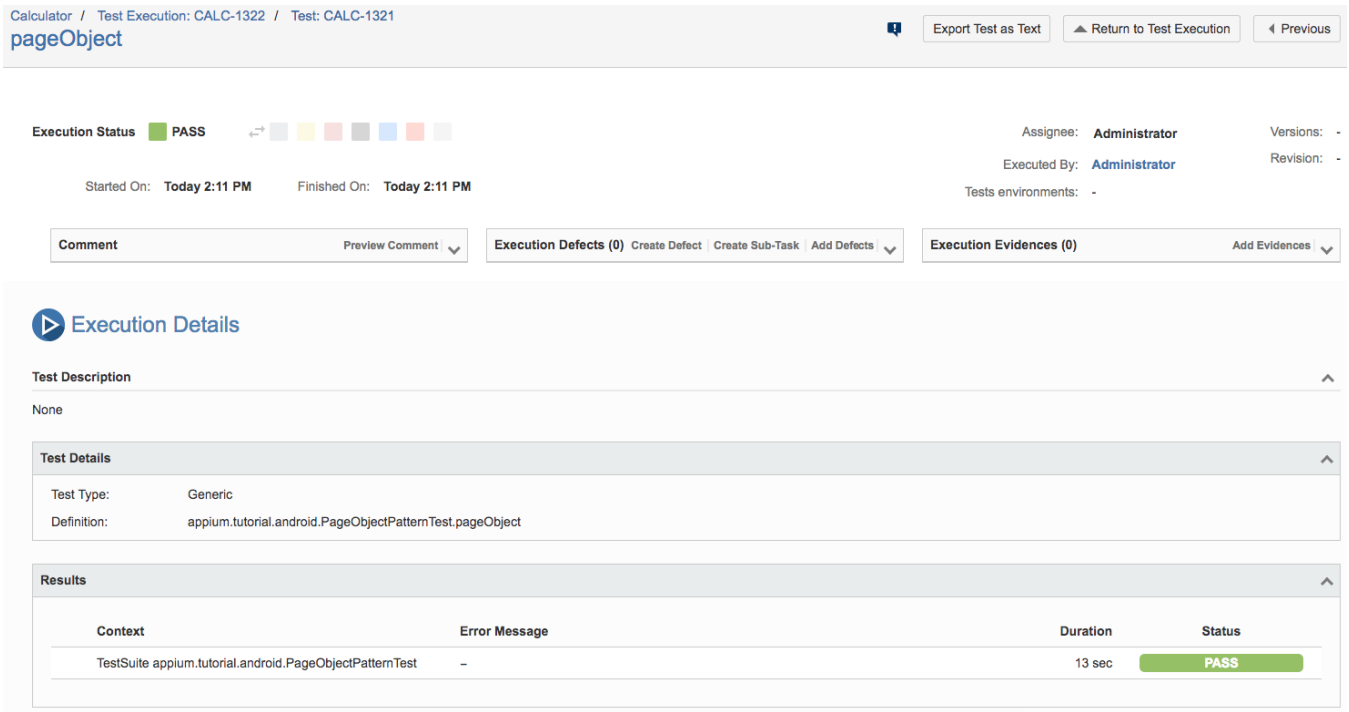
```
junit-merge  -o results.xml -d target/surefire-reports/
```

After successfully running the Test cases and generating the aggregated JUnit XML report (e.g., results.xml), it can be imported to Xray (either by the REST API or through the **Import Execution Results** action within the Test Execution).

Each JUnit's Test Case is mapped to a Generic Test in Jira, and the **Generic Test Definition** field contains the name of the package, the class and the method name that implements the Test Case. The summary of each Test issue is filled out from the name of the method corresponding to the JUnit Test.



The Execution Details of the Generic Test contains information about the Test Suite, which in this case corresponds to the Test Case class, including its namespace.



References

- https://github.com/appium/tutorial/tree/master/projects/java_android
- http://appium.io
- https://www.npmjs.com/package/junit-merge