Java Example Scripts for Android Mobile Application Tests

These examples employ the page object model to run tests on emulators and simulators. Feel free to clone these scripts directly from GitHub, and follow the instructions in the README file.

Page Objects

This object represents a single view/page in the sample application. Click below to see the script:

appium-example/src/test/java/example/android/Pages/GuineaPigPage.java (source from Sauce Labs Training Java)

```
package example.android.Pages;

import org.openqa.selenium.NoSuchElementException;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;

public class GuineaPigPage {

    @FindBy(id = "Heading1_1")
    private WebElement h1Text;

    @FindBy(id = "i_am_a_link")
    private WebElement theActiveLink;

    @FindBy(id = "your_comments")
    private WebElement yourComments;

    @FindBy(id = "comments")
    private WebElement commentsTextInput;

    @FindBy(id = "submit")
    private WebElement submitButton;

    public WebDriver driver;

    public GuineaPigPage(WebDriver driver) {
        this.driver = driver;
        PageFactory.initElements(driver, this);
    }

    public void submitComment(String text) {
        this.commentsTextInput.click();
        this.commentsTextInput.sendKeys(text);
        hideKeyboard();
        this.submitButton.click();
    }

    public String getSubmittedCommentText() {
        return this.yourComments.getText();
    }

    public void followLink() {
        this.theActiveLink.click();
    }

    public boolean isOnPage() {
        try {
            getSubmittedCommentText();
            return true;
        } catch (NoSuchElementException ex) {
            return false;
        }
    }

    /**
     * This method only work for this page and assumes the app supports keyboard hide on click-away.
     */
    public void hideKeyboard() {
        this.h1Text.click();
    }
}
```

Test Objects

These object represent the individual tests, as well as the prerequisite and postrequisite test tasks (TestBase.java). Click on any of the text below to see the scripts:

appium-example/src/test/java/example/android/Tests/TestBase.java (source from Sauce Labs Training Java)

```
package example.android.Tests;

// import Sauce TestNG helper libraries

import com.saucelabs.common.SauceOnDemandAuthentication;
import com.saucelabs.common.SauceOnDemandSessionIdProvider;
import com.saucelabs.testng.SauceOnDemandAuthenticationProvider;
import com.saucelabs.testng.SauceOnDemandTestListener;
import io.appium.java_client.android.AndroidDriver;
import org.openqa.selenium.MutableCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Listeners;

import java.lang.reflect.Method;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.UnexpectedException;

// import testng annotations
// import java libraries

/**
 * Simple TestNG test which demonstrates being instantiated via a DataProvider in order to supply multiple
browser combinations.
 *
 * @author Neil Manvar
 */
@Listeners({SauceOnDemandTestListener.class})
public class TestBase implements SauceOnDemandSessionIdProvider, SauceOnDemandAuthenticationProvider {

    public String seleniumURI = "@ondemand.saucelabs.com:443";
    public String buildTag = System.getenv("BUILD_TAG");
    public String username = System.getenv("SAUCE_USERNAME");
    public String accesskey = System.getenv("SAUCE_ACCESS_KEY");
    public String app = "https://github.com/saucelabs-training/demo-java/blob/master/appium-example/resources
/android/GuineaPigApp-debug.apk?raw=true";
    //public String app = "../../../../../../resources/android/GuineaPigApp-debug.apk";

    /**
     * Constructs a {@link SauceOnDemandAuthentication} instance using the supplied user name/access key.
To use the authentication
     * supplied by environment variables or from an external file, use the no-arg {@link
SauceOnDemandAuthentication} constructor.
     */
    public SauceOnDemandAuthentication authentication = new SauceOnDemandAuthentication(username, accesskey);

    /**
     * ThreadLocal variable which contains the  {@link AndroidDriver} instance which is used to perform
browser interactions with.
     */
    private ThreadLocal<AndroidDriver> androidDriver = new ThreadLocal<AndroidDriver>();

    /**
     * ThreadLocal variable which contains the Sauce Job Id.
     */
    private ThreadLocal<String> sessionId = new ThreadLocal<String>();

    /**
     * DataProvider that explicitly sets the browser combinations to be used.
     *
     * @param testMethod
     * @return Two dimensional array of objects with browser, version, and platform information
```

```
     */
    @DataProvider(name = "hardCodedBrowsers", parallel = true)
    public static Object[][] sauceBrowserDataProvider(Method testMethod) {
        return new Object[][]{
                new Object[]{"Android", "Samsung Galaxy Tab S3 GoogleAPI Emulator", "8.1", "1.9.1",
"portrait"},
                new Object[]{"Android", "Samsung Galaxy S9 Plus FHD GoogleAPI Emulator", "8.1", "1.9.1",
"portrait"}
        };
    }

    /**
     * @return the {@link AndroidDriver} for the current thread
     */
    public AndroidDriver getAndroidDriver() {
        return androidDriver.get();
    }

    /**
     * @return the Sauce Job id for the current thread
     */
    public String getSessionId() {
        return sessionId.get();
    }

    /**
     * @return the {@link SauceOnDemandAuthentication} instance containing the Sauce username/access key
     */
    @Override
    public SauceOnDemandAuthentication getAuthentication() {
        return authentication;
    }

    /**
     * Constructs a new {@link AndroidDriver} instance which is configured to use the capabilities defined
by the browser,
     * version and os parameters, and which is configured to run against ondemand.saucelabs.com, using
     * the username and access key populated by the {@link #authentication} instance.
     *
     * @param platformName      name of the platformName. (Android, iOS, etc.)
     * @param deviceName        name of the device
     * @param platformVersion   Os version of the device
     * @param appiumVersion     appium version
     * @param deviceOrientation device orientation
     * @return
     * @throws MalformedURLException if an error occurs parsing the url
     */
    protected void createDriver(
            String platformName,
            String deviceName,
            String platformVersion,
            String appiumVersion,
            String deviceOrientation,
            String methodName)
            throws MalformedURLException, UnexpectedException {
        MutableCapabilities capabilities = new MutableCapabilities();
        capabilities.setCapability("platformName", platformName);
        capabilities.setCapability("platformVersion", platformVersion);
        capabilities.setCapability("deviceName", deviceName);
        capabilities.setCapability("browserName", "");
        capabilities.setCapability("deviceOrientation", deviceOrientation);
        capabilities.setCapability("appiumVersion", appiumVersion);
        capabilities.setCapability("name", methodName);
        capabilities.setCapability("app", app);
        capabilities.setCapability("build", "Java-TestNG-Appium-Android");

        if (buildTag != null) {
            capabilities.setCapability("build", buildTag);
        }

        // Launch remote browser and set it as the current thread
```

```
        androidDriver.set(new AndroidDriver(
                new URL("https://" + authentication.getUsername() + ":" + authentication.getAccessKey() +
seleniumURI + "/wd/hub"),
                capabilities));

        String id = ((RemoteWebDriver) getAndroidDriver()).getSessionId().toString();
        sessionId.set(id);
    }

    /**
     * Method that gets invoked after test.
     * Dumps browser log and
     * Closes the browser
     */
    @AfterMethod
    public void tearDown() throws Exception {

        //Gets browser logs if available.
        androidDriver.get().quit();
    }
}
```

appium-example/src/test/java/example/android/Tests/FollowLinkTest.java (source from Sauce Labs Training Java)

```
package example.android.Tests;

import example.android.Pages.GuineaPigPage;
import org.openqa.selenium.InvalidElementStateException;
import org.openqa.selenium.WebDriver;
import org.testng.Assert;
import org.testng.annotations.Test;

import java.lang.reflect.Method;
import java.net.MalformedURLException;
import java.rmi.UnexpectedException;

/**
 * Created by mehmetgerceker on 12/7/15.
 * Updated by spider@saucelabs.com on 10/8/19.
 */

public class FollowLinkTest extends TestBase {

    /**
     * Runs a simple test verifying link can be followed.
     *
     * @throws InvalidElementStateException
     */
    @Test(dataProvider = "hardCodedBrowsers")
    public void verifyLinkTest(String platformName,
                               String deviceName,
                               String platformVersion,
                               String appiumVersion,
                               String deviceOrientation,
                               Method method)
            throws MalformedURLException, InvalidElementStateException, UnexpectedException {

        //create webdriver session
        this.createDriver(platformName, deviceName, platformVersion, appiumVersion, deviceOrientation,
method.getName());
        WebDriver driver = this.getAndroidDriver();

        GuineaPigPage page = new GuineaPigPage(driver);

        page.followLink();

        Assert.assertFalse(page.isOnPage());
    }

}
```

appium-example/src/test/java/example/android/Tests/TextInputTest.java (source from Sauce Labs Training Java)

```
package example.android.Tests;

import example.android.Pages.GuineaPigPage;
import org.openqa.selenium.InvalidElementStateException;
import org.openqa.selenium.WebDriver;
import org.testng.Assert;

import java.lang.reflect.Method;
import java.net.MalformedURLException;
import java.rmi.UnexpectedException;
import java.util.UUID;


/**
 * Created by mehmetgerceker on 12/7/15.
 */

public class TextInputTest extends TestBase {

    /**
     * Runs a simple test verifying if the comment input is functional.
     * @throws InvalidElementStateException
     */
    @org.testng.annotations.Test(dataProvider = "hardCodedBrowsers")
    public void verifyCommentInputTest(String platformName,
                                       String deviceName,
                                       String platformVersion,
                                       String appiumVersion,
                                       String deviceOrientation,
                                       Method method)
            throws MalformedURLException, InvalidElementStateException, UnexpectedException {

        this.createDriver(platformName, deviceName, platformVersion, appiumVersion, deviceOrientation,
method.getName());
        WebDriver driver = this.getAndroidDriver();

        String commentInputText = UUID.randomUUID().toString();

        GuineaPigPage page = new GuineaPigPage(driver);

        page.submitComment(commentInputText);

        Assert.assertTrue(page.getSubmittedCommentText().contains(commentInputText));
    }

}
```