

实验四—分析与测试实验

实验目的

本次实验主要是为了掌握基本的程序分析方法和测试方法。从静态分析、白盒测试、黑盒测试三个方面出发，将实验设置为以下三个部分：

- 代码静态分析
- 单元测试
- 基于GUI的系统级测试

实验所需工具

1. Android Studio
2. Appium

实验评分标准

1. 完成实验所需工具的安装使用，可得基础分60分。
2. 完成静态分析，并报告当前代码存在问题，可加10分。如果对所报告问题进行改进可适当加分。
3. 针对自己添加的模块编写测试用例（所测试方法不能重复），至少5个，可加10分。
4. 针对自己添加的模块编写GUI测试测试脚本（所测试功能不能重复），至少5个，可加10分。
5. 完成实验报告，可加10分。

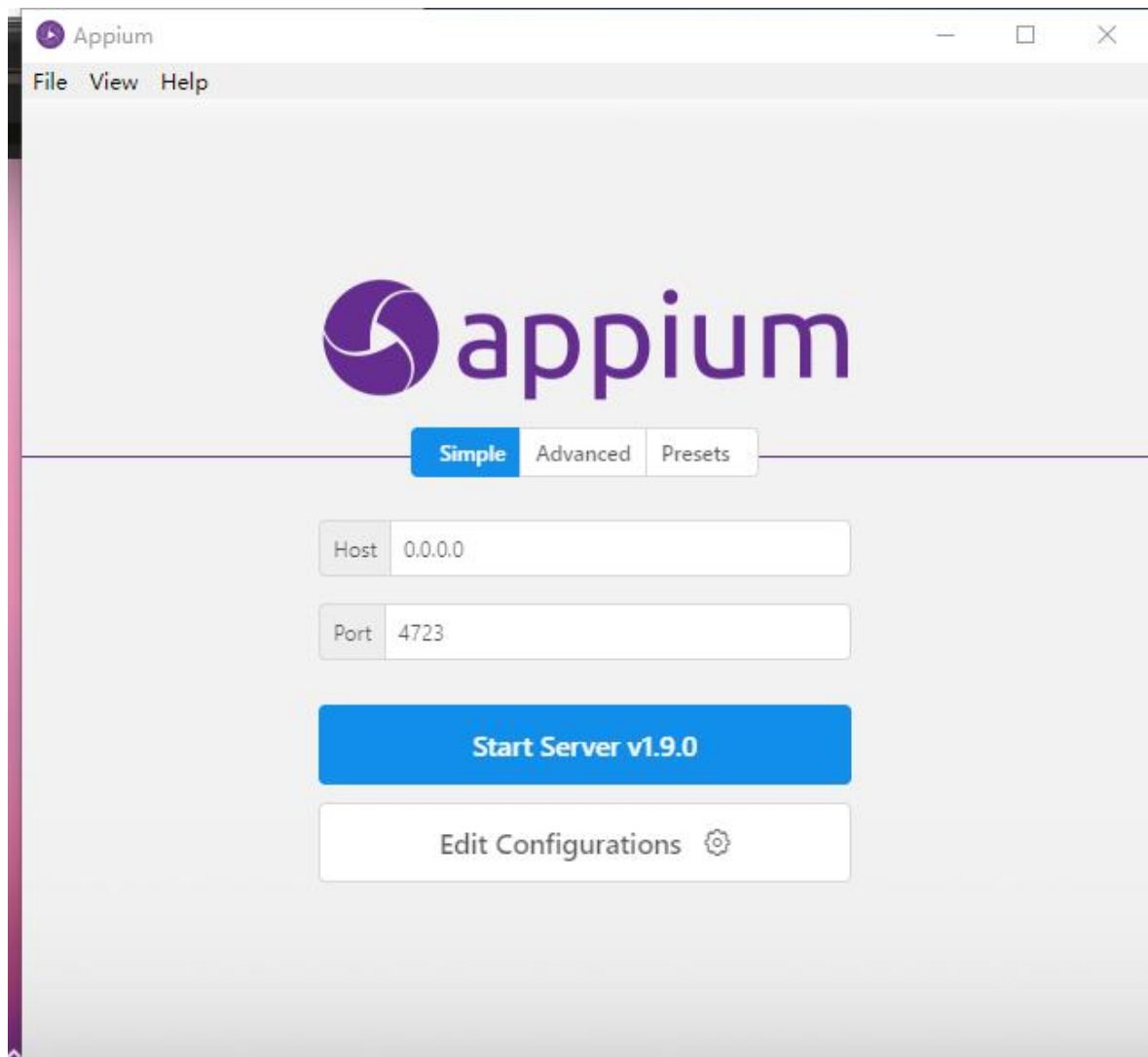
实验工具的安装说明

- **Appium的安装（推荐安装Appium客户端）**

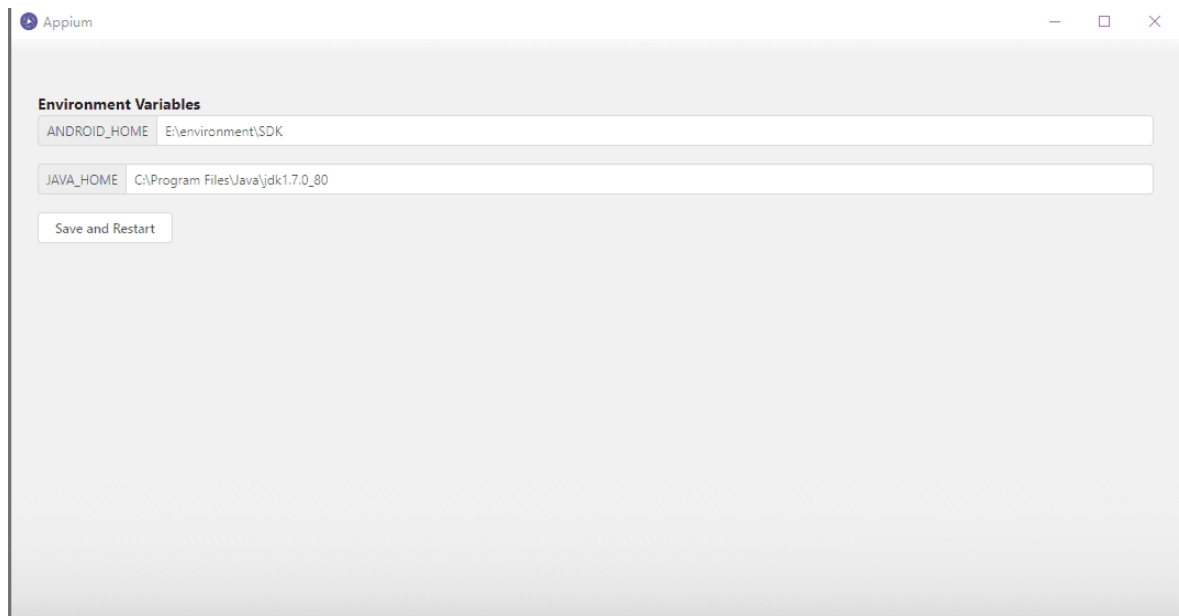
在<https://github.com/appium/appium-desktop/releases/tag/1.7.0>下载相应安装包即可。

下面以Windows为例讲解Appium的配置及使用方法。

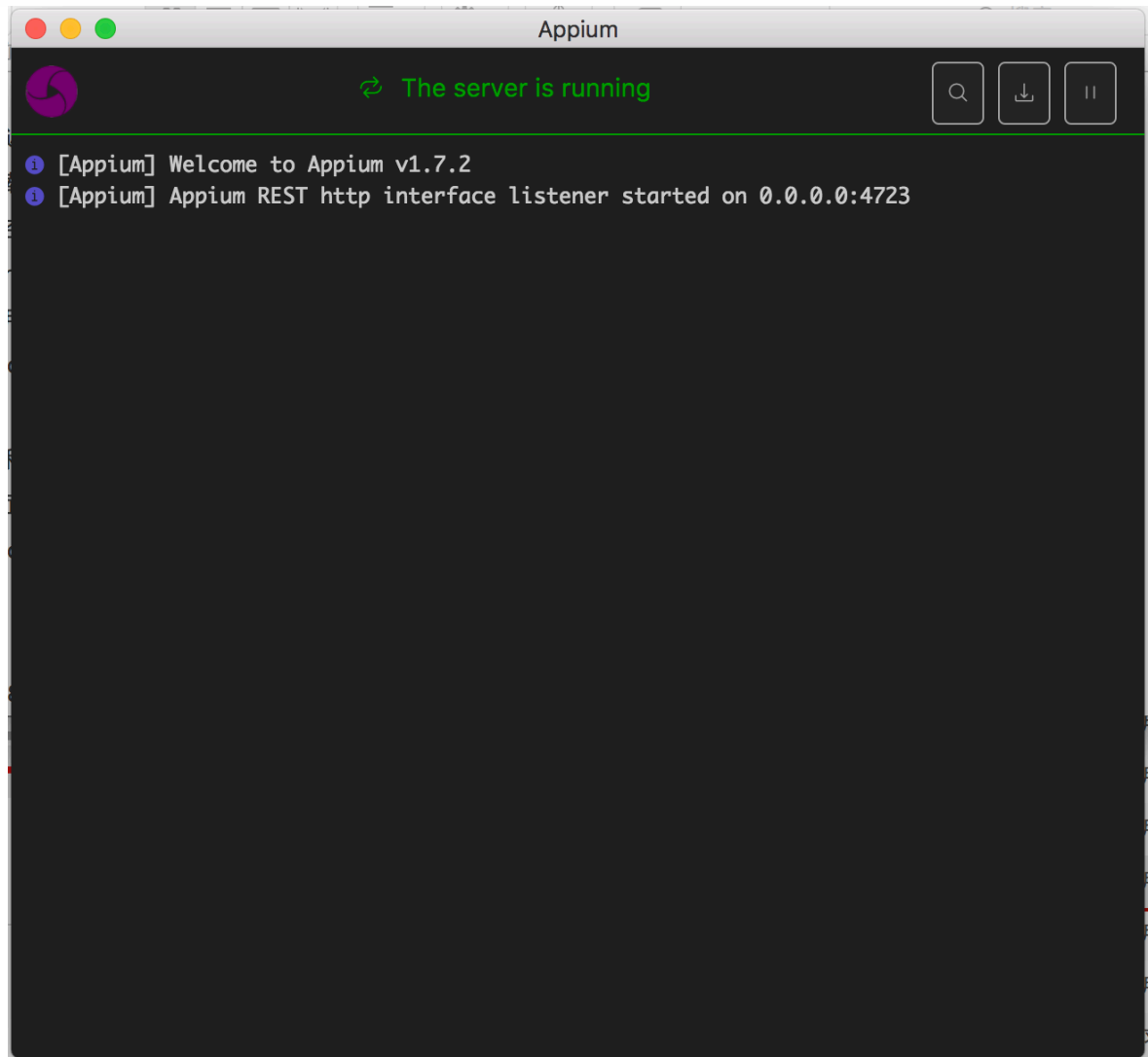
下载完成后双击运行Appium，启动页面如下：



点击Edit Configurations按钮进行相应环境变量配置，主要是Android SDK路径及Java路径。



完成路径配置后，点击save and restart按钮重启appium。再次在重启页面，点击start server即可启动appium。



可以看到，当前appium的监听端口为0.0.0.0:4723。此时appium已经成功启动。

- 本次使用GUI测试脚本使用Python3编写，Python环境配置过程如下：

Python下载地址：<https://www.python.org/downloads/>。请勿下载python2.7版本！下载完成后安装后配置相应环境变量即可。

对于Mac用户，推荐使用brew进行安装可避免与系统已有python版本冲突。

完成python环境配置后，使用pip命令进行appium依赖添加，pip命令如下：

```
pip install Appium-Python-Client
```

如果发现无法使用pip，请先进行pip的安装，安装方法如下：

参考网址：<https://pip.pypa.io/en/stable/installing/>

首先下载get-pip.py文件，下载方式如下：

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
```

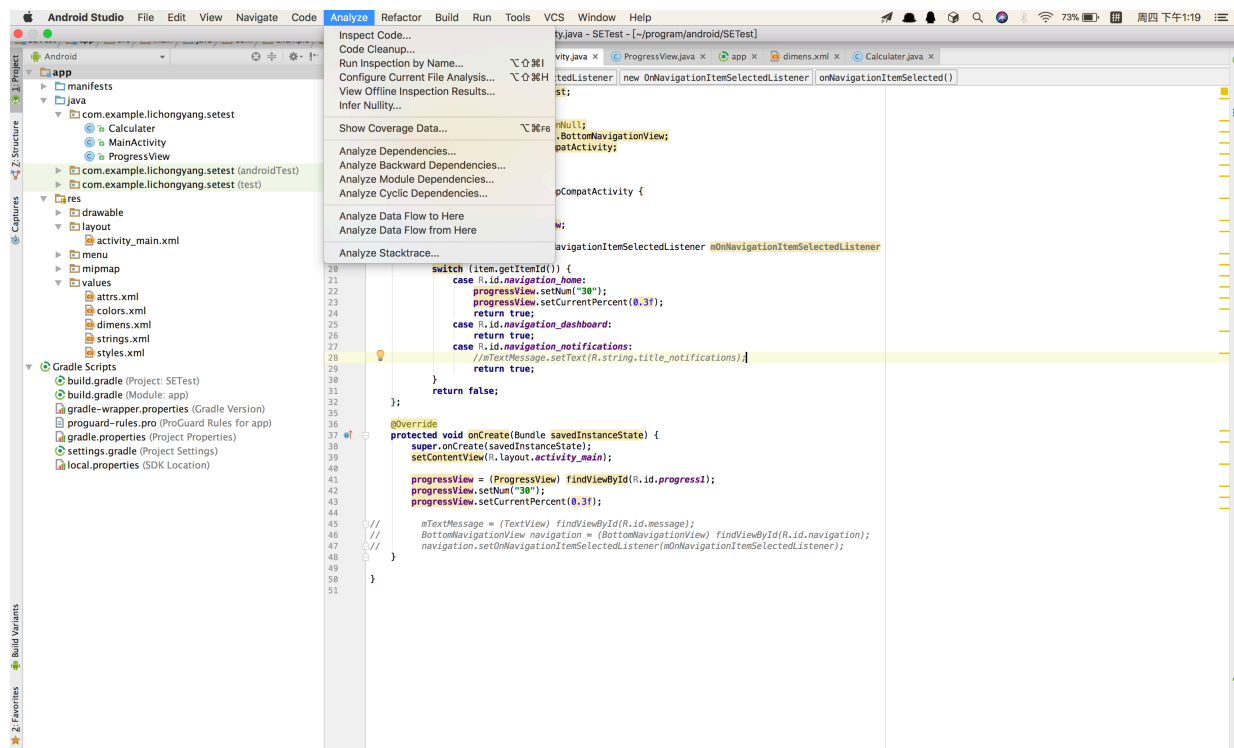
下载完成后，在get-pip.py文件所在目录下，运行如下命令安装pip：

```
python get-pip.py
```

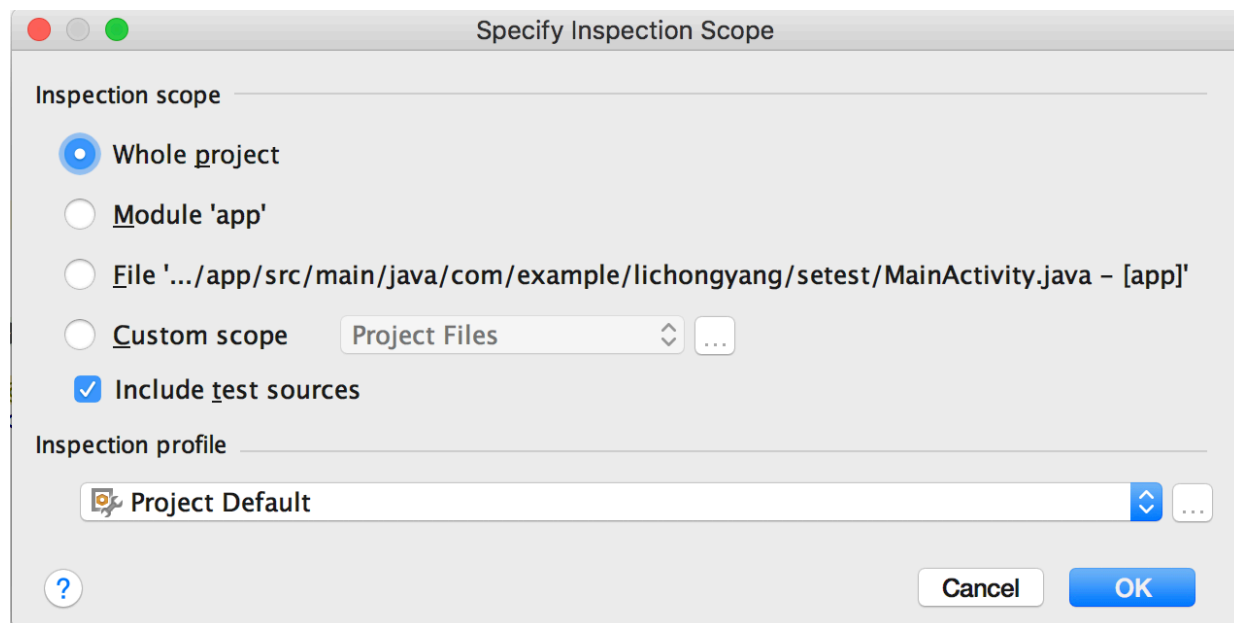
完成pip安装后重新执行appium依赖安装命令即可。

静态分析

Android Studio已经集成静态分析工具Lint。只需点击Analyze->Inspect Code即可。



在配置页面，你可以根据自己的需求更改配置，选择分析整个文件还是某一模块。当然你也可以直接点击ok进行代码分析。



运行完成后，分析结果报告如下：


```

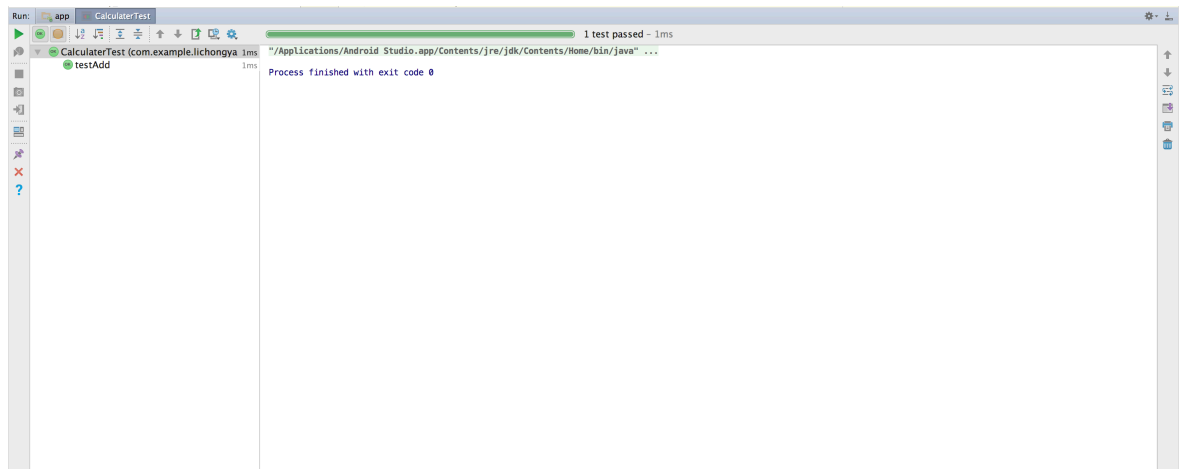
public class CalculatorTest {
    Calculator calculator = new Calculator();

    @org.junit.Test
    // 运行测试时，所有被@org.junit.Test注解的方法均会被执行
    public void testAdd(){
        int a = 1, b = 2;
        int result = calculator.add(a, b);

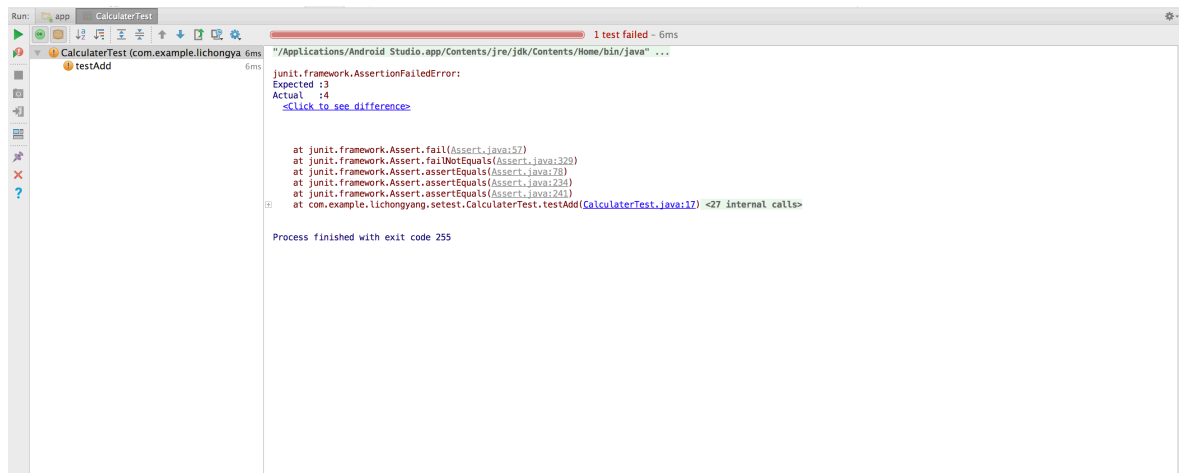
        Assert.assertEquals(result, 3);
    }
}

```

运行测试，在Android Studio中可以看到测试通过：



如果将上述代码改为 `Assert.assertEquals(result, 4);`，则测试失败：

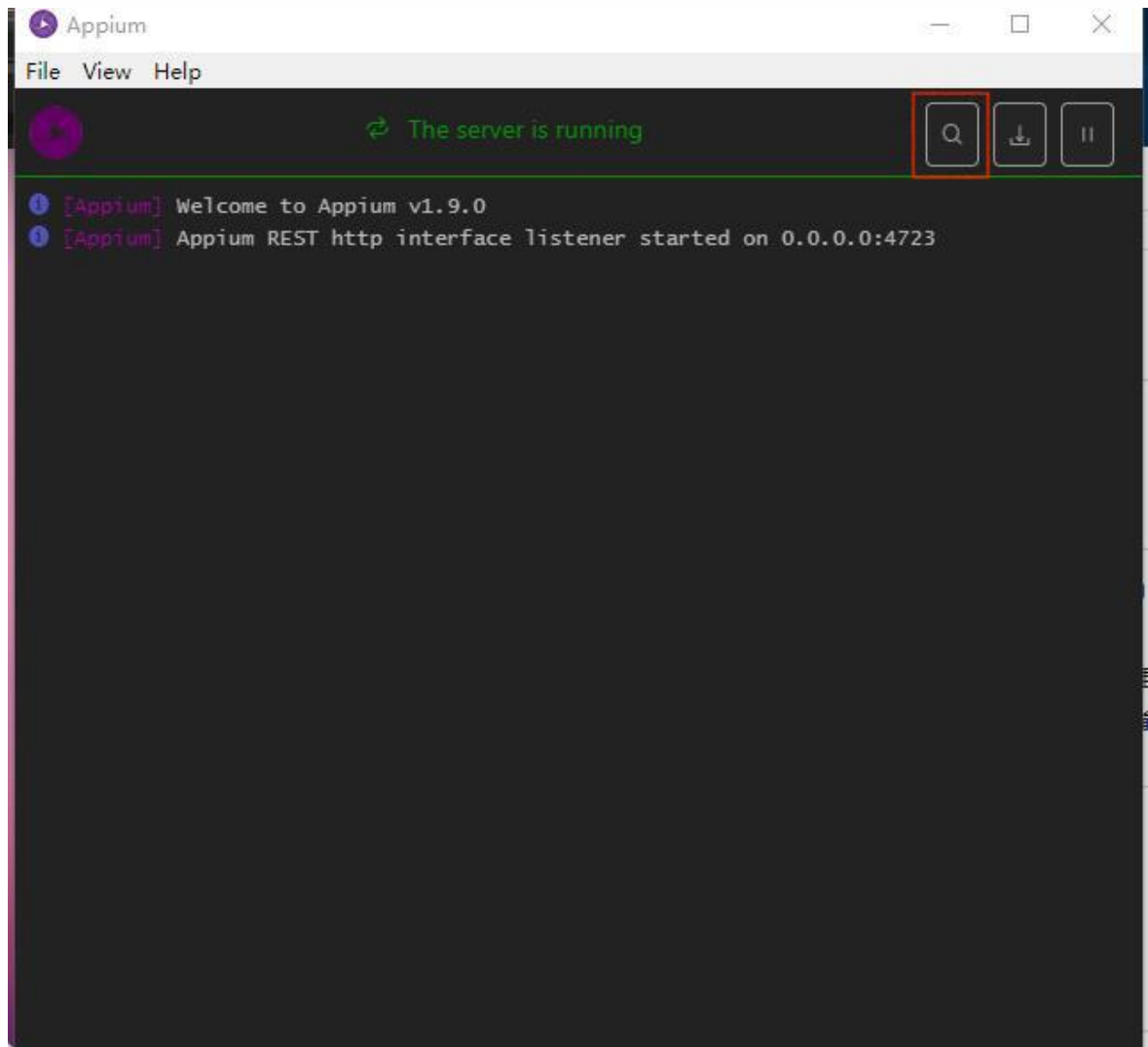


GUI测试

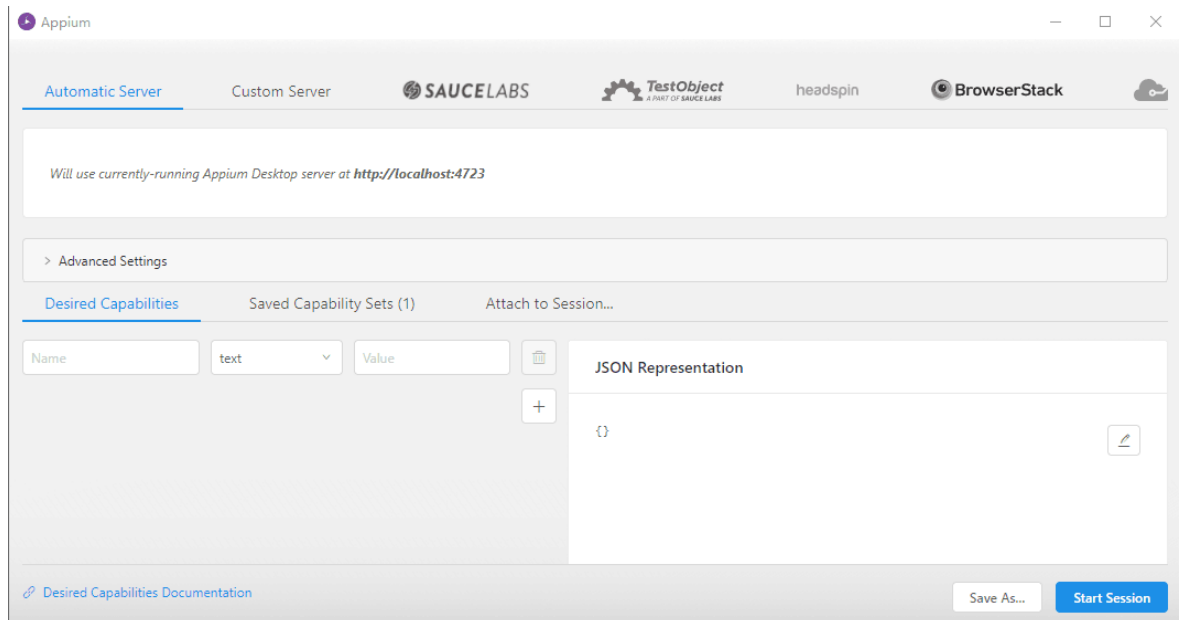
- 界面控件信息获取方法

Appium提供了获取当前页面中控件信息的功能。

首先按上述Appium启动步骤，启动Appium。然后点击下图红框中按钮。



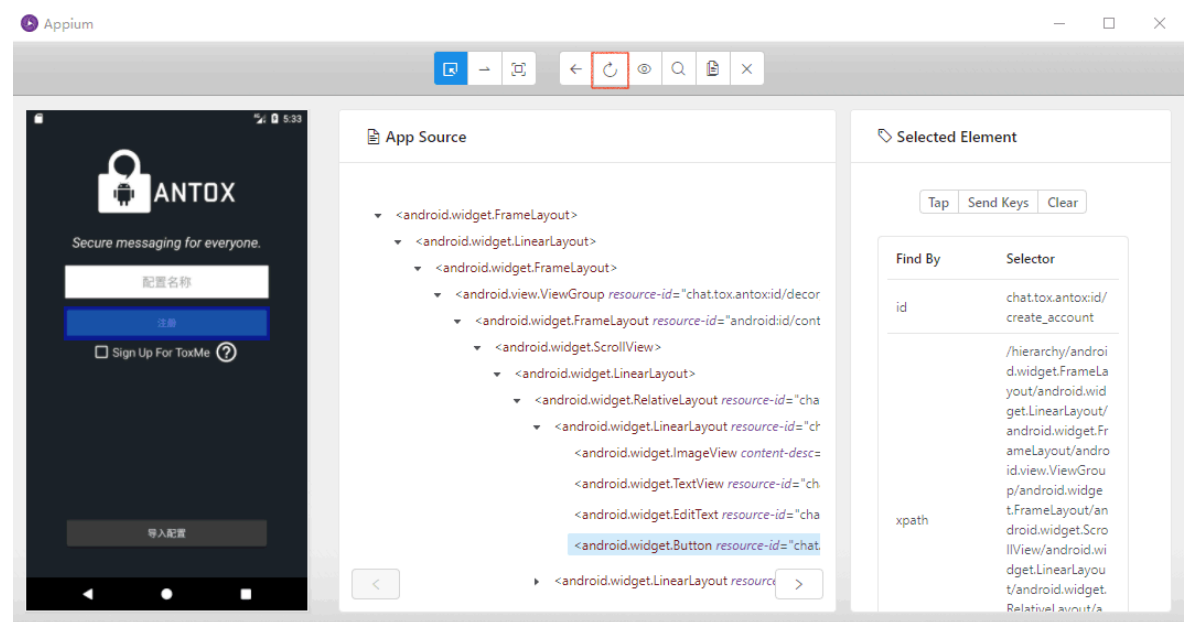
在配置界面，填写配置信息，即填写Desired Capabilites。



基本Desired Capabilites配置如下：

```
{
    "platformName": "Android", // 设备类型, Andorid平台默认为Android
    "platformVersion": "7.1.1", // 设备Android版本信息
    "deviceName": "emulator-5554", // 设备名称
    "appPackage": "chat.tox.antox", // 待测试应用包名
    "appActivity": "chat.tox.antox.SplashActivity" // 待测试应用
    MainActivity名称
}
// 其中设备信息可在设备的设置中获取
// 应用包名、MainActivity信息可在项目AndroidManifest.xml中获得
```

填写好相应配置后，点击start session即可，查看应用当前页面控件信息。



在左侧点击相应控件即可在右侧看到该控件的相应信息。

如果进入新的页面，只需点击上图红框中按钮即可刷新页面进而获得新的页面的控件信息。

- 测试脚本编写

以下给出一个简单的测试脚本：

```
# -*- coding:utf8 -*-
import time
from appium import webdriver
from appium.webdriver.common.touch_action import TouchAction
# appium 配置信息
desired_caps = {}
desired_caps['platformName'] = 'Android'
desired_caps['platformVersion'] = '7.1.1'
desired_caps['deviceName'] = 'emulator-5554'
desired_caps['appPackage'] = 'chat.tox.antox'
desired_caps['appActivity'] = 'chat.tox.antox.SplashActivity'
desired_caps['noReset'] = True
desired_caps["unicodeKeyboard"] = "True"
desired_caps["resetKeyboard"] = "True"
```

```

# test case3:
# 启动webdriver执行测试脚本
# http://0.0.0.0:4723/wd/hub为默认端口，可修改为启动appium时你所设置的端口，例如：http://127.0.0.1:2233/wd/hub
driver = webdriver.Remote('http://0.0.0.0:4723/wd/hub', desired_caps)
driver.implicitly_wait(20)

time.sleep(2)

# 根据控件的class_name来获取控件
el = driver.find_elements_by_class_name("android.widget.ImageButton")
[0]
el.click() # 点击该控件

# 根据控件的ID来获取控件
el = driver.find_elements_by_id("android:id/title")[0]
el.click()

el = driver.find_elements_by_id("a2dp.Vol:id/checkbox1")[0]
el.click()

el = driver.find_elements_by_class_name("android.widget.ImageButton")
[0]
el.click()

el = driver.find_elements_by_id("android:id/title")[0]
el.click()

driver.back()

driver.quit()

```

完成测试脚本的编写后，启动appium，再执行测试脚本即可对应用进行测试。

编写时可参考：

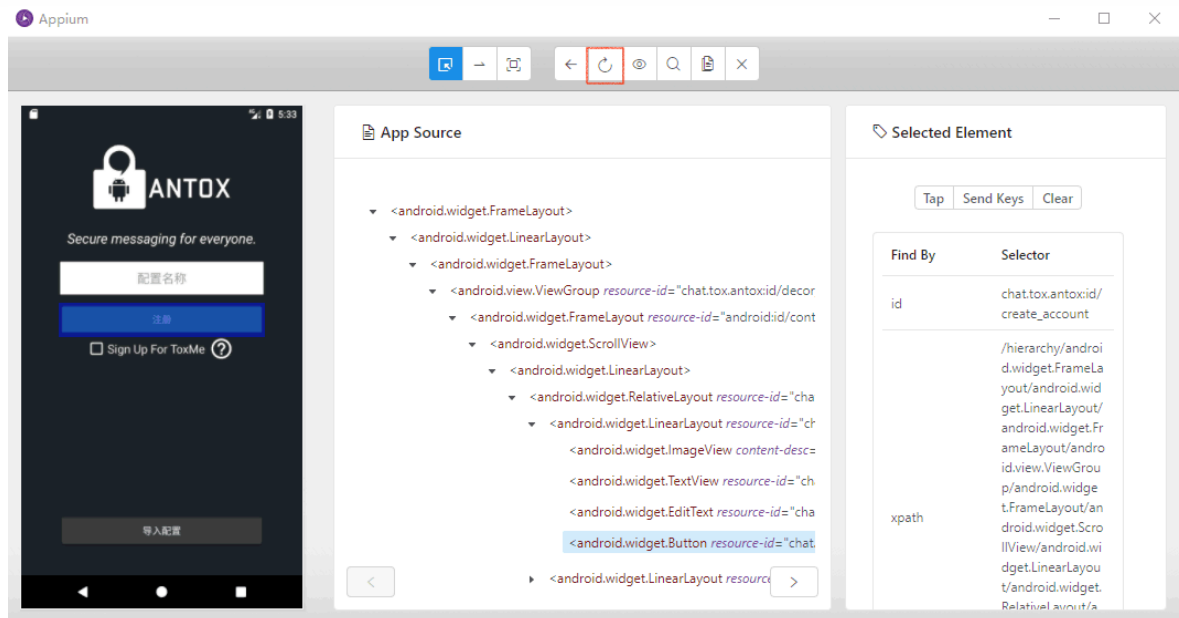
<https://www.cnblogs.com/xiaoxi-3-/p/7941253.html>

<https://github.com/appium-boneyard/sample-code/tree/master/sample-code/examples/python>

实验报告提交说明

- 任务一、安装、熟悉并使用实验所需工具

报告中请提交至少一张使用Appium获取控件信息功能获取应用页面控件的使用截图。图片类似下图：



注意：

- 截图中应用必须是你自己所分配的APP
- 任务二、静态分析
 - 以表格形式提交静态分析结果，表格格式类似下表：

类别	数目
Correctness	2
bug	2

- 如果你对静态分析结果进行了改进，请列出你所改进前后代码。如果更改代码较多，列出部分即可（3条以上）。
- 任务三、单元测试
 - 以表格形式列出你所编写单元测试基本信息，表格格式类似下表：

所测试方法	所测试方法 功能描述	测试用 例数目	Pass测试 用例数目	Fail测试 用例数目
MainActivity.onCreate()	加法	3	2	1

- 至少提交一张Pass测试用例的运行结果截图及一张Fail测试用例的运行结果截图。
- 任务四、GUI测试
 - 以表格形式列出你所编写测试脚本基本信息，表格格式类似下表：

测试场景	测试该场景的测试用例数目	测试该场景的测试用例代码行数
应用转账功能	2	30

- 将所有测试脚本（即.py）文件压缩为zip文件提交。