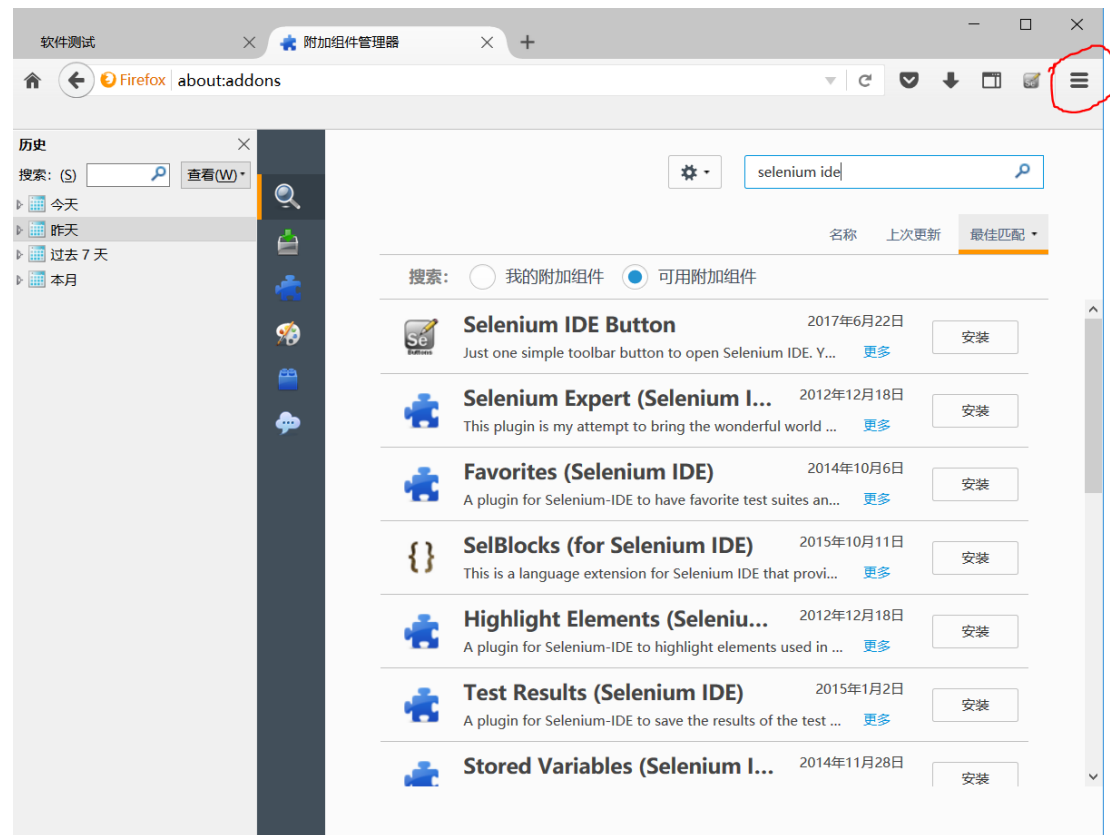


软件测试第二次实验——Selenium 安装与使用

1.SeleniumIDE 安装

安装 Firefox42.0 版本, 打开 Firefox 浏览器, 点击右上角菜单栏中附加组件, 搜索 Selenium IDE。



找到 Selenium IDE 后, 官方提示当前浏览器版本较低, 因此查看历史版本, 寻找兼容版本的 Selenium IDE 后加入到浏览器中, 根据提示下载安装后重启浏览器完成安装。



版本 3.0.1.0

发布 四月 11, 2018 · 3.3 MiB
兼容于 Firefox 56.0 及更新的版本

源代码在 [自定义许可协议](#) 下发布 · [这是什么?](#)

+ 添加到 Firefox

此附加组件不兼容您所使用的 Firefox 版本。
[继续下载](#)

版本 3.0.0.0

发布 四月 11, 2018 · 3.3 MiB
兼容于 Firefox 56.0 及更新的版本

源代码在 [自定义许可协议](#) 下发布 · [这是什么?](#)

+ 添加到 Firefox

此附加组件不兼容您所使用的 Firefox 版本。
[继续下载](#)

版本 2.9.1.1-signed

发布 三月 15, 2015 · 796.5 KiB
兼容于 Firefox 17.0 - 56.*

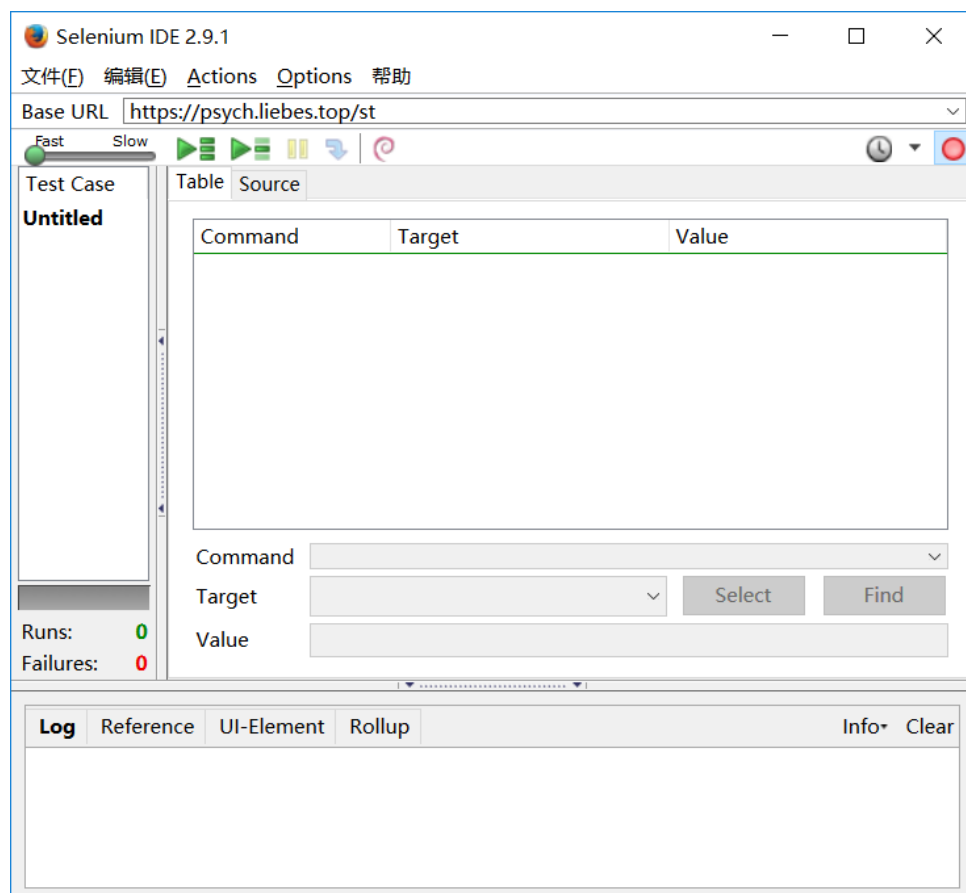
Selenium IDE v2.9.1
Merged all the official language exporters into the main addon and eliminated having a multi-xpi.

源代码在 [自定义许可协议](#) 下发布 · [这是什么?](#)

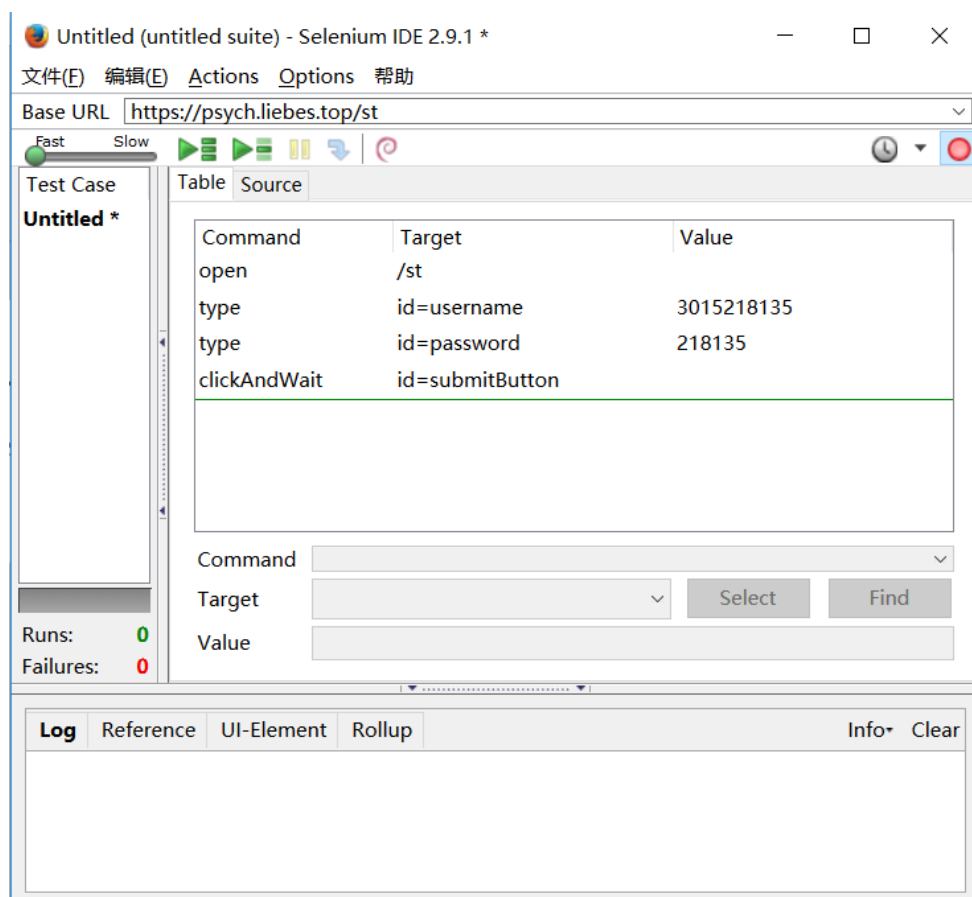
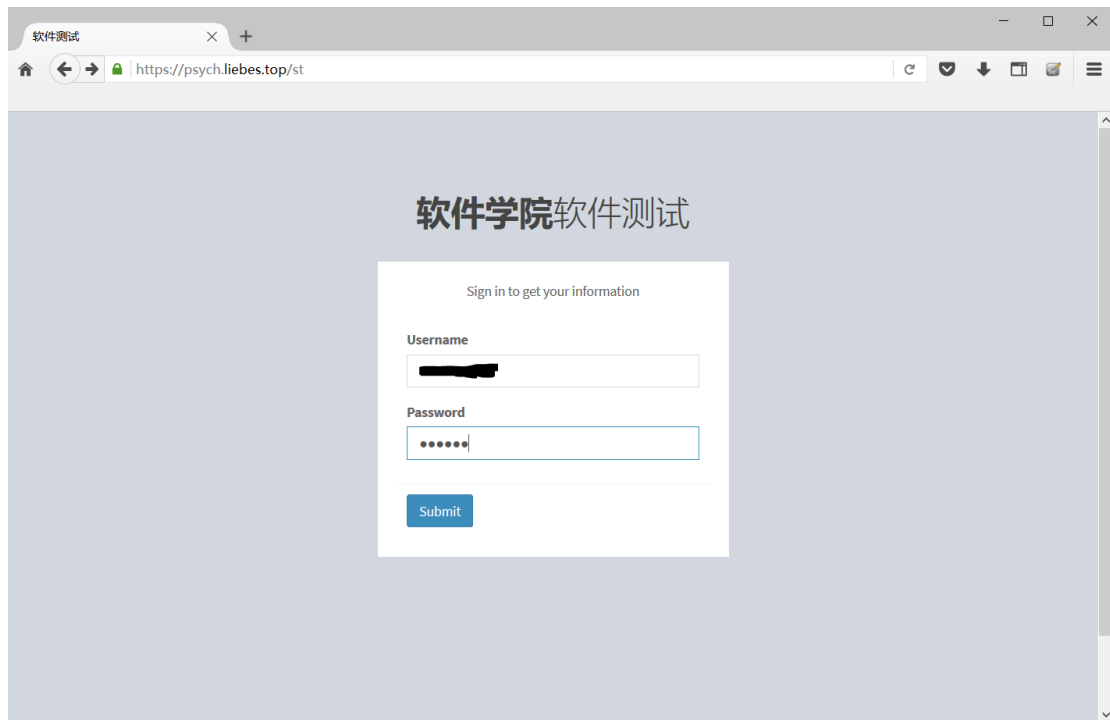
+ 添加到 Firefox

2.使用 Selenium IDE 录制脚本和导出脚本

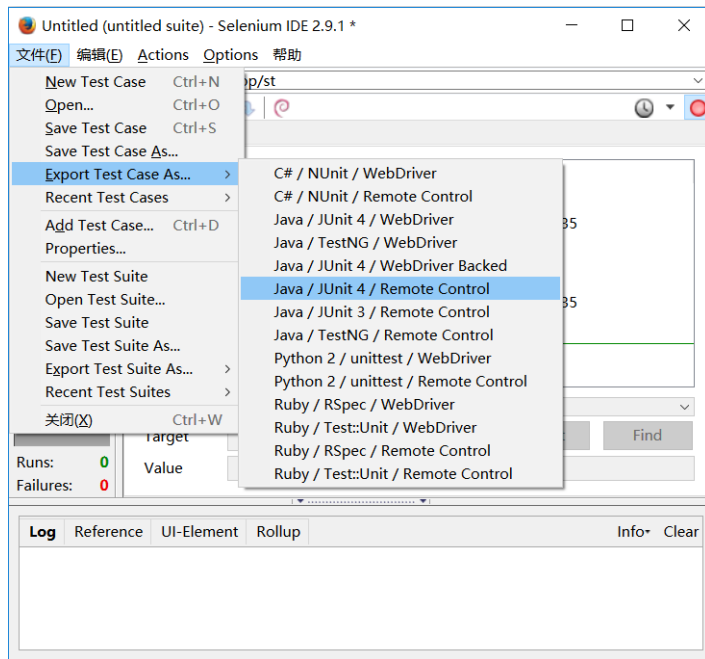
重启浏览器后，点击地址框右侧出现新图标(Se)，点击打开 Selenium IDE



此时已为代码录制状态，在浏览器中输入地址 <https://psych.liebes.top/st>，输入自己的学号和密码后，生成脚本

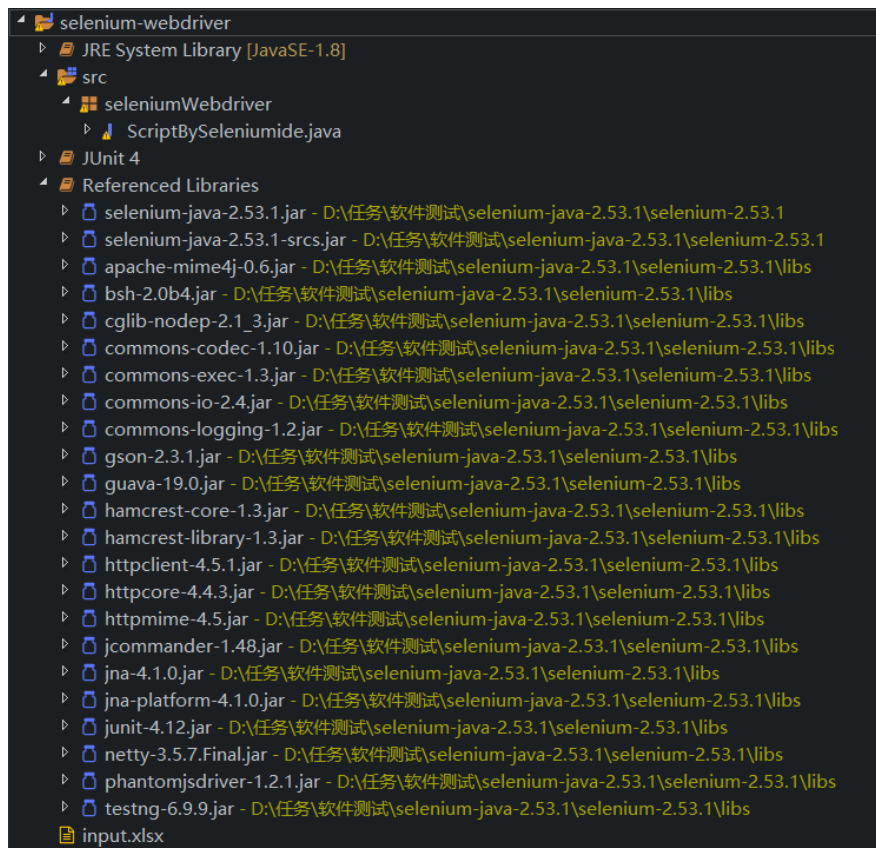


点击工具栏->文件->export test case as->java junit4 webdriver 导出脚本并保存



3.运行导出的测试脚本

在 eclipse 中创建一个 java 项目，将 selenium-java-2.53.1 目录中的 selenium-java.2.53.1.jar，selenium-java-2.53.1-srcs.jar 以及 libs 目录下的所有 jar 包导入项目中，导入 junit4 测试包，将之前导出的 java 脚本放入项目 src 目录中，文件层级如下：



导出的 java 脚本在 eclipse 中查看部分如下：

```

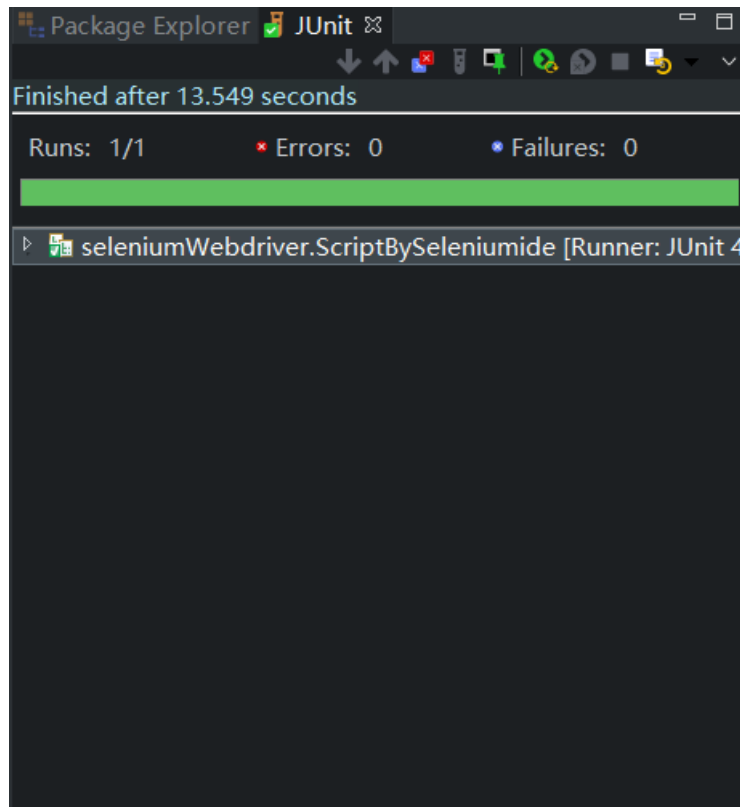
public class ScriptBySeleniumide {
    private WebDriver driver;
    private String baseUrl;
    private boolean acceptNextAlert = true;
    private StringBuffer verificationErrors = new StringBuffer();

    @Before
    public void setUp() throws Exception {
        driver = new FirefoxDriver();
        baseUrl = "https://psych.liebes.top/st";
        driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
    }

    @Test
    public void testScriptBySeleniumide() throws Exception {
        driver.get(baseUrl);
        driver.findElement(By.id("username")).clear();
        driver.findElement(By.id("username")).sendKeys("3015218135");
        driver.findElement(By.id("password")).clear();
        driver.findElement(By.id("password")).sendKeys("218135");
        driver.findElement(By.id("submitButton")).click();
    }
}

```

运行脚本结果如下：



4.编写 Selenium java webdriver 程序，测试测试 input.xlsx 表格中的学号和 git 地址的对应关系是否正确

在 3.所建项目 scr 中添加新类 TestScript.java，在该类中编写自己的 selenium java webdriver 测试程序。

- 将 input.xlsx 文件中的学号和对应 git 地址读出
在项目中添加处理.xlsx 的 jar 包

```

> dom4j-1.6.1.jar - D:\任务\软件测试\读取xls、xlsx文件
> poi-3.8-20120326.jar - D:\任务\软件测试\读取xls、xls
> poi-ooxml-3.8-20120326.jar - D:\任务\软件测试\读取
> poi-ooxml-schemas-3.8-20120326.jar - D:\任务\软件
> xmlbeans-2.3.0.jar - D:\任务\软件测试\读取xls、xlsx文

```

编写处理.xlsx 文件数据的方法, 用一个 Map 数据对存储所有的学号以及对应的 git 地址
部分代码如下:

```
public Map<String, String> getElementsFromXlsx() {
    XSSFSheet xssfSheet = xssfWorkbook.getSheetAt(0);

    if (xssfSheet == null) {
        System.out.println("no values to return");
        return null;
    }

    Map<String, String> idAndUrl = new HashMap<>();

    for (int rowIndex = 0; rowIndex < 97; rowIndex++) {
        XSSFRow curRow = xssfSheet.getRow(rowIndex);
        if (curRow != null) {
            //System.out.println(String.valueOf(curRow.getCell(0).getStringCellValue()));
            XSSFCell xCellId = curRow.getCell(0);
            XSSFCell xCellUrl = curRow.getCell(1);

            if (xCellId.getCellType() == xCellId.CELL_TYPE_NUMERIC) {

                // format the number
                //DecimalFormat df = new DecimalFormat("#, ##0.00");
                Long numericId = (Long)xCellId.getNumericCellValue();
                idAndUrl.put(String.valueOf(numericId),
                    String.valueOf(xCellUrl.getStringCellValue()));
            } else {
                idAndUrl.put(String.valueOf(xCellId.getStringCellValue()),
                    String.valueOf(xCellUrl.getStringCellValue()));
            }
        }
    }
}
```

- 创建一个 webdriver 实例, 并编写自动操作浏览器匹配方法:

```
public boolean isMatched(String id, String url) {
    // config firefox webdriver
    WebDriver driver = new FirefoxDriver();
    driver.get(BASEURL);
    driver.findElement(By.id("username")).sendKeys(id);
    driver.findElement(By.id("password")).sendKeys(id.substring(4));
    System.out.println(id + "\t" + id.substring(4));
    driver.findElement(By.id("submitButton")).click();

    /*
    try {
        Thread.sleep(100);
    } catch (Exception e) {
        // TODO: handle exception
        e.printStackTrace();
    }*/
    System.out.println(driver.findElement(By.xpath("//div[@class='login-box-body']/a/p[@class='login-box-msg']")).getText());
    if (driver.findElement(By.xpath("//div[@class='login-box-body']/a/p[@class='login-box-msg']")).getText().equals(url)) {
        driver.close();
        return true;
    } else {
        driver.close();
        return false;
    }
}
```

- 遍历 Map, 对每一个学号进行登录, 并对 git 地址进行匹配, 匹配失败的学号将会存入一个顺序表中。

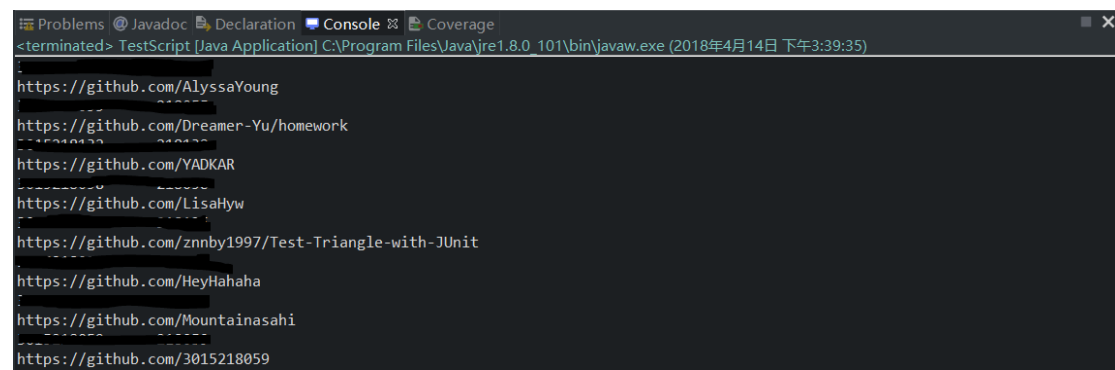
```
public List<String> getWrongIds() {
    List<String> wrongIds = new ArrayList<>();

    for (Map.Entry<String, String> entry: getElementsFromXlsx().entrySet()) {
        if (!isMatched(entry.getKey(), entry.getValue())) {
            wrongIds.add(entry.getKey());
        }
    }

    return wrongIds;
}
```

运行程序后, 系统自动弹出 Firefox 浏览器并进行指定地址访问和登录操作, 操作结束后关

闭界面，并对下一个 map 对进行相同操作。运行时部分匹配结果如下：



```
Problems Javadoc Declaration Console Coverage
<terminated> TestScript [Java Application] C:\Program Files\Java\jre1.8.0_101\bin\javaw.exe (2018年4月14日 下午3:39:35)

https://github.com/AlyssaYoung
https://github.com/Dreamer-Yu/homework
https://github.com/YADKAR
https://github.com/LisaHyw
https://github.com/znnby1997/Test-Triangle-with-JUnit
https://github.com/HeyHahaha
https://github.com/Mountainasahi
https://github.com/3015218059
```

5.实验总结

<1>.成功实现了 Firefox 浏览器自动访问指定地址界面并用给定用户信息进行登录的操作

<2>.存在一些不足等待改进:

单线程操作大量数据访问并登录，使得程序运行效率大大降低（导致最后我也没有测试完所有的学号和 git 地址）。解决设想：每次浏览器操作开设新的线程，应该可以很大程度提高效率，但大量线程将会存在同步问题。

每次浏览器操作完成后不会自动关闭该浏览器，导致有大量无用的浏览器打开。（查找更多通过 webdriver 操作界面的方法）。