SJTU-SE

测试报告

第 6 组

李　珊 516030910175

王梦瑶 516030910177

陈　诺 516030910199

胡雨奇 516030910257

目录

1. 测试目的

1. 练习和掌握 web GUI 测试的一般步骤
2. 掌握使用 Intellij 和 Selenium 进行测试的方法
3. 通过测试验证目标网站的正确性

2. 测试方法

使用 Intellij 和 Selenium 来进行 web GUI 测试。

通过针对网站代码编写测试代码，来验证网站基础功能的正确性。

3. 测试环境

1. 系统：Windows10-64bit
2. IDE： Intellij
3. 库：Selenium 3.141.59 Maven

4. 测试前准备

1. Selenium 使用

主要知识参考课程 PPT 与网络教程（见本文参考）

为了能在本项目中使用 Selenium，需要在 pom.xml 添加 Selenium 依赖：

```
<dependencies>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>3.141.59</version>
    </dependency>
</dependencies>
```

2. 测试代码架构

各个测试代码的入口文件是：./src/main/java/test/OpenBrowsers.java

在 OpenBrowsers.java 的 main 函数中，首先进行了基础设置（cookie 等），做好登陆工作。

然后依次实例化各个测试类并运行。

```
public static void main(String args[]) throws Exception{

    System.setProperty("webdriver.chrome.driver","path-to\\chromedriver.exe");

    ChromeDriver driver = new ChromeDriver();

    // log in
    LoginUtil lu=new LoginUtil();
    lu.addCookie(driver);
    lu.getCookie(driver);

    // 实例化测试类
    NavigateTest nt = new NavigateTest(driver);
    // 运行测试类
    nt.run();

    // 其他测试类

    Thread.sleep(1000);
    driver.quit();
}
```

在各个测试类的 run()函数中，再分别调用各个测试函数。每个测试函数在 return 之前会向 console 输出测试成功或者失败的信息；

3. 登录逻辑

首次登录时使用 addCookie：跳转入登录页面自动填写用户名密码登录，登录完成后，使用 driver.manage().getCookies()获取 cookie 并写入本地文件 douban.cookie.txt 中。

再次登录时使用 getCookie：读取本地文件 douban.cookie.txt 中的字段生成 cookie，使用 driver.manage().addCookie(cookie)使 cookie 生效完成自动登录。

```java
public class LoginUtil {
    void addCookie(ChromeDriver driver);
    void getCookie(ChromeDriver driver);
}
```

5. 测试内容

5.1 页面跳转

测试代码文件： .\src\main\java\test\NavigateTest.java

5.1.1 顶部导航栏

5.1.1.1 预期功能

点击导航栏中的某一项，如“读书”，弹出一个新的标签页（Tab），Url 是对应的内容。

5.1.1.2 测试逻辑

1）找到顶部导航栏

2）记录下现有所有标签页（Tab）的句柄

3）选择导航栏中的一个按钮，调用 click()事件

4）记录新的所有标签页（Tab）的句柄，找出多的项

5）切换到上述句柄对应的标签页（switchTo），核对 url 是否为预期 url

6）关闭这个句柄对应的标签页（Tab）

7）对导航栏中所有 8 个链接做上述测试。

以下代码以“豆瓣读书”为例：

```
WebElement topbar =  driver.findElementById("anony-nav");
WebElement links = topbar.findElement(By.className("anony-nav-links"));
assert links.getText().equals("豆瓣读书 豆瓣电影 豆瓣音乐 豆瓣小组 豆瓣同城 豆瓣FM 豆瓣时间 豆瓣豆品");

/* assert link number */
List<WebElement> link_list = links.findElements(By.tagName("li"));
assert (link_list.size() == 8);

/* assert Link url */
assertSubTabLink(link_list.get(0), "https://book.douban.com/");
```

5.1.1.3 测试结果

所有 8 个标签页的跳转逻辑正常，测试通过。

5.1.2 “Album”模块的跳转

5.1.2.1 预期功能

点击“Album”模块中的一个版块，页面从当前页面切换到该相册对应的页面，不打开新的标签页（Tab）

5.1.2.2 测试逻辑

1）找到“Album”模块

2）选择一个链接调用 click()方法

3）核对 url 中是否包含“https://www.douban.com/photos/album/”

4）返回 www.douban.com 主页面

```java
/* find album links */
WebElement grid_element = driver.findElementById("anony-sns");
WebElement album = grid_element.findElement(By.className("albums"));
List<WebElement> links = album.findElements(By.tagName("a"));

/* assert url jump */
links.get(0).click();
String current_url = driver.getCurrentUrl();
assert current_url.contains("https://www.douban.com/photos/album/");
System.out.println("AssertOK: " + "album check");

/* return main page */
driver.navigate().to(url);
```

5.1.2.3 测试结果

标签页跳转到正确的 url，测试通过

5.1.3 “Note”模块的跳转

5.1.3.1 预期功能

点击“Note”模块中的一个版块，页面从当前页面切换到该相册对应的页面，不打开新的标签页（Tab）

5.1.3.2 测试逻辑

1）找到“Note”模块

2）选择一个链接调用 click()方法

3）核对 url 中是否包含“https://www.douban.com/note/”

4）返回 www.douban.com 主页面

```
/* find note links */
WebElement grid_element = driver.findElementById("anony-sns");
WebElement note = grid_element.findElement(By.className("notes"));
List<WebElement> links = note.findElements(By.tagName("a"));

/* assert url jump */
links.get(0).click();
String current_url = driver.getCurrentUrl();
assert current_url.contains("https://www.douban.com/note/");
System.out.println("AssertOK: " + "note check");

/* return main page */
driver.navigate().to(url);
```

5.1.3.3 测试结果

标签页跳转到正确的 url，测试通过

5.1.4 “Topic”模块的跳转

5.1.4.1 预期功能

点击“Topic”模块中的一个版块，页面从当前页面切换到该相册对应的页面，不打开新的标签页（Tab）

5.1.4.2 测试逻辑

1）找到“Topic”模块

2）选择一个链接调用 click()方法

3）核对 url 中是否包含“https://www.douban.com/gallery/topic/”

4）返回 www.douban.com 主页面

```java
/* find topic links */
WebElement grid_element = driver.findElementById("anony-sns");
List<WebElement> topic_link =
grid_element.findElements(By.className("rec_topics_name"));

/* assert url jump */
topic_link.get(0).click();
String current_url = driver.getCurrentUrl();
assert current_url.contains("https://www.douban.com/gallery/topic/");
System.out.println("AssertOK: " + "topic check");

/* return main page */
driver.navigate().to(url);
```

5.1.4.3 测试结果

标签页跳转到正确的 `url`，测试通过。

5.2 按钮

测试文件：`.\src\main\java\test\ButtonTest.java`

5.2.1 [正在热映]组件

`testSlideButton1()` 测试翻页按钮，判断页数以及窗口中影片内容是否改变

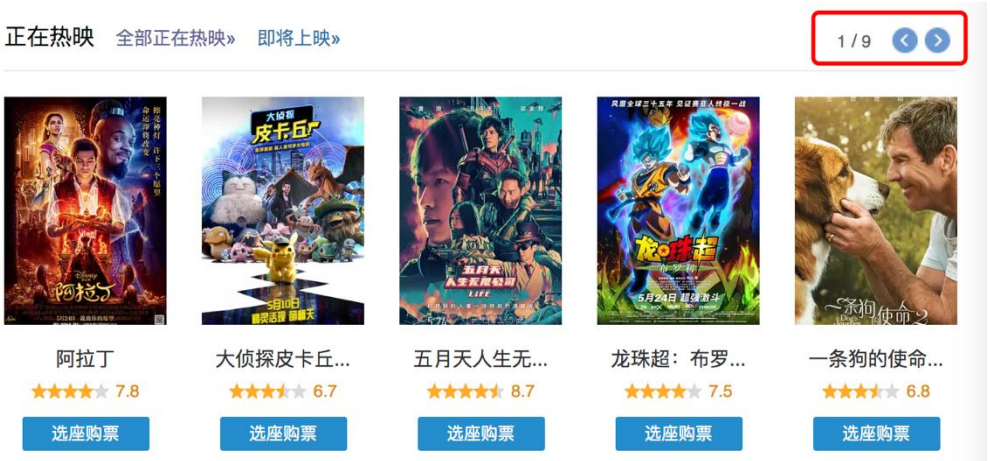

图 1. 正在热映组件

5.2.2 [最近热门电影]组件

`testSlideButton2("movie")` 测试翻页按钮，判断 idx 以及窗口中影片内容是否改变。

`testTagButton("movie")` 测试标签按钮，判断窗口中影片内容是否改变。

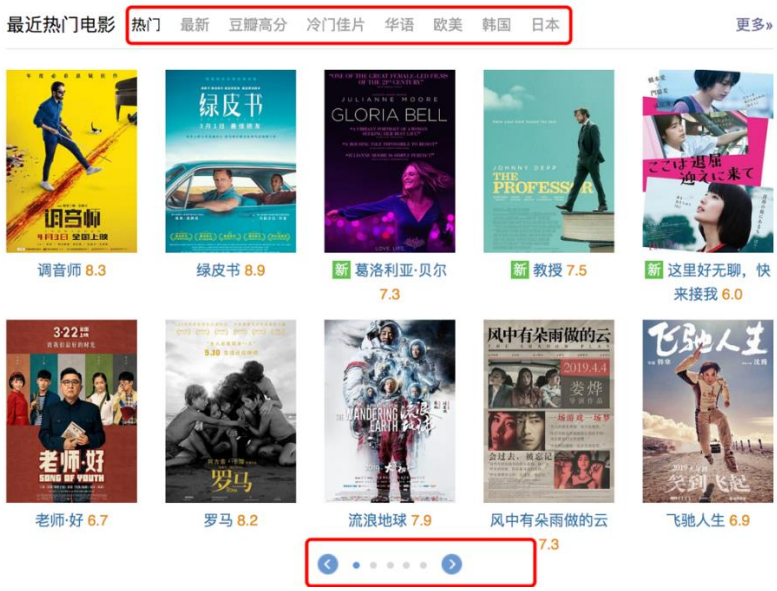

图 2. 最近热门电影组件

5.2.3 [最近热门电视剧]组件（组件结构同热门电影）

```
testSlideButton2("tv")
testTagButton("tv")
```

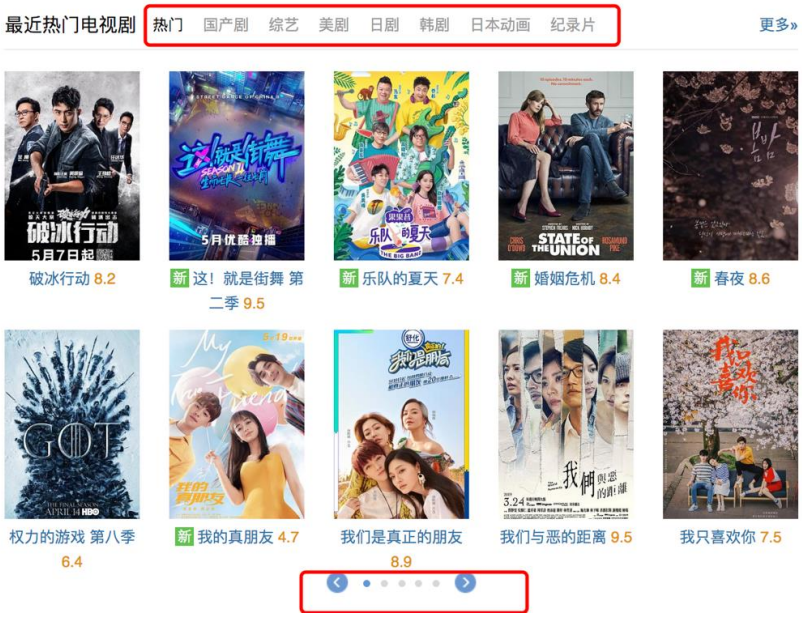

图 3. 最近热门电视剧组件

5.2.4 [热门推荐]组件

`testSlideButton3()` 测试定时翻页及测试翻页按钮，判断页数以及窗口中影片内容是否改变。

图 4. 热门推荐

5.3 输入框

5.3.1 搜索组件

`testSearch()` 填写输入框并点击搜索按钮，检查搜索结果并测试翻页按钮。

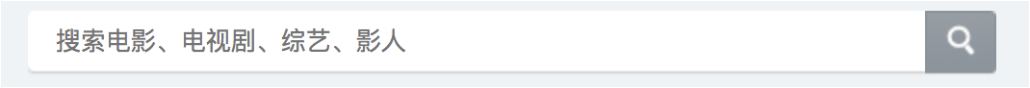

图 5. 搜索组件

图 6. 搜索结果

5.3.2 评论组件

testComment("movie")
测试浮窗并点击按钮[想看]

```
Actions action = new Actions(driver);
action.moveToElement(item).perform();

WebElement btn=driver.findElementByXPath(String.format("//div[@class='gaia gaia-lite gaia-%s slide-mode']/div[1]/div[1]/div[1]/p[3]/a[1]",topic));
btn.click();
```

图 7．最近热门电影

填写评论，并点击仅自己可见后按保存提交。

```
WebElement inputBox=driver.findElementByXPath("//textarea[@id='comment']");
inputBox.sendKeys("233");

WebElement checkBox=driver.findElementByXPath("//input[@id='inp-private']");
checkBox.click();

WebElement submitBtn=driver.findElementByXPath("//div[@id='submits']/span/input");
submitBtn.click();
```

图 8．添加收藏

为了能够重复测试，删除该条评论：点击进入该电影的详情页，点击删除，并操作 alert 窗口确认删除。

```
driver.navigate().to(movieLink);
WebElement cancelBtn=
driver.findElementByXPath("//div[@id='interest_sect_level']/div/span/span[2]/form/input");
cancelBtn.click();

Alert alert = driver.switchTo().alert();
alert.accept();
```

图 9．删除收藏

5.4 多选框

测试文件：.\src\main\java\test\CheckBoxTest.java

5.4.1 [分类]组件

testCheckBox1() 测试可播放和我没看过的多选框，看该组件状态是否可以被选中

找到多选框的列表：

```
List<WebElement> checkBoxInput = driver.findElementsByCssSelector(".check
label.checkbox span");

WebElement checkbox1 = checkBoxInput.get(0);
```

图 10. 选影视多选框

点击查看 css 变化

```
assert (checkbox1.getAttribute("class").equals("checkbox__input"));

checkbox1.click();
assert (checkbox1.getAttribute("class").equals("checkbox__input is-checked is-
focus"));

checkbox1.click();
assert (checkbox1.getAttribute("class").equals("checkbox__input is-checked"));
```

图 11. 选影视多选框

5.4.2 [选电影]组件

testCheckBox2() 测试可播放和我没看过的多选框，看该组件状态是否被选中,看该多选框是否可以同时被多选
点击一个：

```
WebElement e2=driver.findElementByName("watched");
assert !e2.isSelected();
e2.click();
assert e2.isSelected();
e2.click();
assert !e2.isSelected();
```

点击两个：

```
e2.click();e3.click();
assert (e2.isSelected()&&e3.isSelected());
```

图 12．选电影多选框

两个同时不选：

```
e2.click();e3.click();
assert ((!e2.isSelected())&&(!e3.isSelected()));
```

5.2.3 [电视剧]组件

testCheckBox3()测试前面的单选框，可播放和我没看过的多选框，看前三个单选框是否每次只能选中一个，看多选框是否可以同时被多选

单选框测试：

```
WebElement select1=driver.findElementByXPath("//div[@class='sort']/label[1]/input");
WebElement select2=driver.findElementByXPath("//div[@class='sort']/label[2]/input");
WebElement select3=driver.findElementByXPath("//div[@class='sort']/label[3]/input");
select1.click();
assert (select1.isSelected()&&(!select2.isSelected())&&(!select3.isSelected()));
select2.click();
assert (!select1.isSelected()&&(select2.isSelected())&&(!select3.isSelected()));
select3.click();
assert (!select1.isSelected()&&(!select2.isSelected())&&(select3.isSelected()));
```

检查是否选中一个会自动取消其他两个的选中

图 13. 热门电视剧多选框

5.5 下拉框

测试文件：.\src\main\java\test\DropBoxTest.java

5.5.1 下拉框组件

testDropBox() 点击下拉框并点击评分按钮，检查点击结果

图 14. 选影视下拉框组件-1

在没有点击这个按钮的时候，检查这个组件的属性，并查找下拉框的组件。

```
assert (d1.getAttribute("class").equals("angle-down"));
assert !doesElementExist("//div[@class='range-dropdown']");
```

是不存在的，点击之后是状态改变并且下拉框组件出现。

图 15．选影视下拉框组-2

拖动评分区间，值会改变

```
WebElement r3=driver.findElementByXPath("//div[@class='slider-wrap']/div[3]");
r3.click();
assert (d2.getAttribute("value").equals("2,10"));
```

图 16. 选影视下拉框-3

5.6 文件上传

测试文件：.\src\main\java\test\UploadFileTest.java

5.6.1 上传图片

网页位置：https://accounts.douban.com/passport/setting

测试内容：上传头像

5.6.2 测试逻辑

进入网页 https://accounts.douban.com/passport/setting 后，选择上传头像，并给弹出的文件夹输入目标图片位置，在浮动窗口中选择确定完成文件上传。

```
driver.get("https://accounts.douban.com/passport/setting");
WebElement upldAtrBtn=driver.findElementByXPath("//div[@class='account-
wrap']/div[1]/div[2]/div[1]/div[1]/div[1]/input");

File file = new File("C:\\Users\\31600\\Pictures\\me.jpg");
upldAtrBtn.sendKeys(file.getPath());

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

WebElement submitBtn = driver.findElementByXPath("//div[@class='ui-dialog-
mountnode']/div[1]/div[1]/div[1]/div[2]/div[1]/div[1]/div[3]/button[1]");
submitBtn.click();

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

System.out.println("Success: uploadPic");
```

5.6.3 测试结果

通过在 Intellij 中以 debug 模式运行并打断点查看测试结果。

1. 进入个人设置页

图 17. 个人设置页

2. 给弹出的文件夹输入图片路径

3. 在浮窗中确认图片

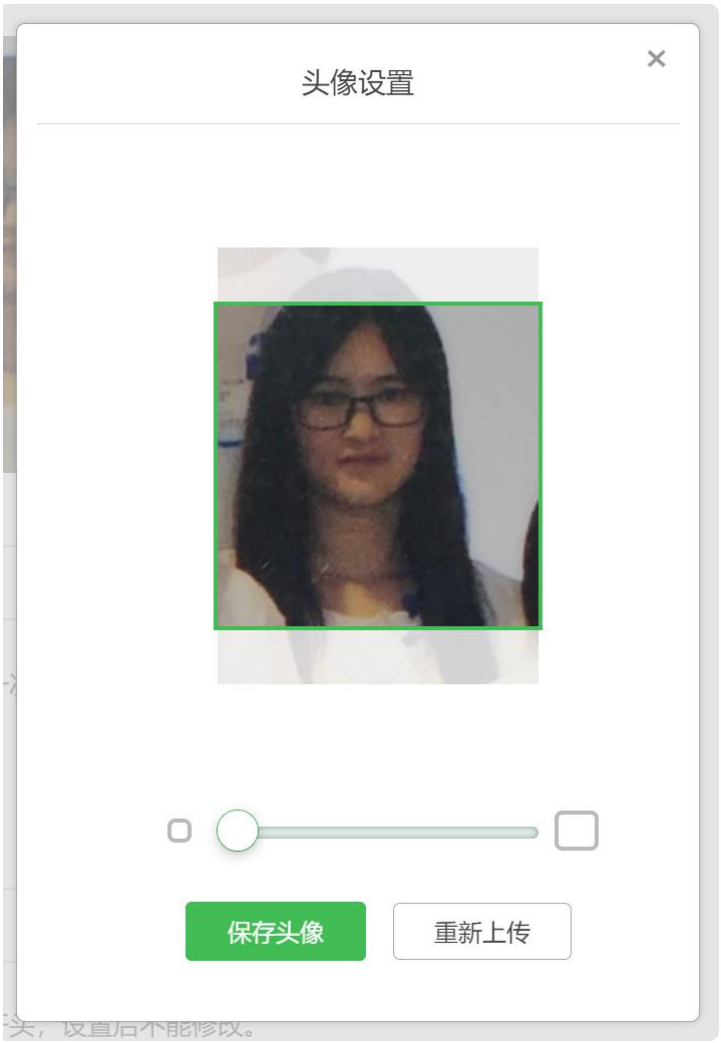

图 18. 确认图片

4. 测试结束

5.7 模态框

测试代码：`.\src\main\java\test\UploadFileTest.java`

5.7.1 离开当前页面

网页位置：https://www.douban.com/note/create
测试内容：未保存输入框文字，点击顶部导航栏按键试图回到首页。

5.7.2 测试逻辑

将 driver 重定向到首页，显式等待合适时间后通过点击按钮进图写日记界面，设置好输入框的文字。

图 19. 书写笔记页

点击顶部导航栏的图标尝试回到首页。这时由于输入框中的文字尚未保存，网站会弹出 alart() 模态框，这时判断网页是否弹出模态框。

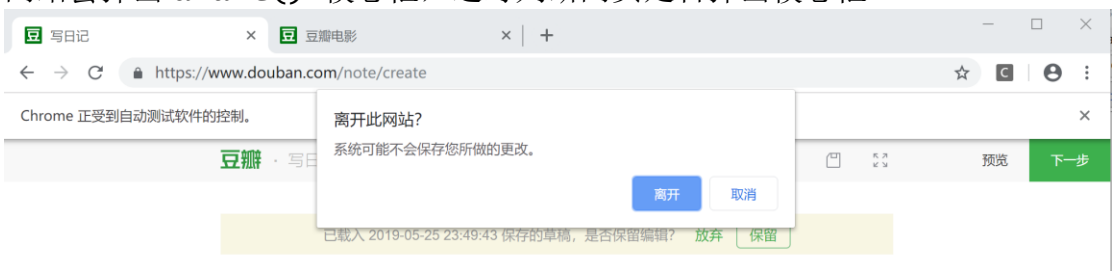

图 20. 弹出模态框

通过对模态框的选项做出不同选择，来完成当前页面的跳转或者停留在本页面。通过对当前页面的 url 使用 assert 来判断模态框是否正确执行。

这里需要注意的是，Selenium 不能直接对模态框做出操作，这是因为模态对话框会将父页面的 JS 挂起，直至对话框处理完毕才会继续执行父页面 JS。因为 Selenium 的底层实现是基于 JS 的，所以模态对话框会同时将 selenium 挂起，selenium 无法选中模态对话框，直至超时。

不得不使用 Selenium 时，可以选择使用 hack 方法，在第一次点击某个会触发模态框的页面元素时，使用 showModalDialog 来覆盖，改打开模态对话框为 window.open 打开网页，并将 Selenium 选中该弹出网页：

```
public void clickAndSelectModalDialog(String locator){
    clickForModalDialog(locator);
     selenium.selectWindow("name=modal");
}

private void  clickForModalDialog(String locator){
        String overrideShowModalDialogJs =
"if(selenium.browserbot.getCurrentWindow().showModalDialog){";

    overrideShowModalDialogJs +=
"selenium.browserbot.getCurrentWindow().showModalDialog = function( sURL, vArguments,
sFeatures)";

    overrideShowModalDialogJs +="selenium.browserbot.getCurrentWindow().open(sURL,
'modal', sFeatures);";

    overrideShowModalDialogJs += "};}";
    //showModalDialog方法进行覆盖
    selenium.getEval(overrideShowModalDialogJs);
    selenium.click(locator);
    selenium.openWindow("","modal");
    selenium.waitForPopUp("modal","15000");
}
```

在成功将模态框截获到新的页面之后，就可以在新的页面对模态框做出操作，获取需要返回给父页面的值。

在这里，为了尽量贴近使用场景，我选用了 driver 来操作，通过 driver.switchTo().alert() 来获得 alert 模态框。通过 accept() 和 dismiss() 来接受或者放弃 alert 操作。

```
driver.switchTo().alert().dismiss();
assert(driver.getCurrentUrl().equals("https://www.douban.com/note/create"));

driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);

driver.switchTo().alert().accept();
assert (driver.getCurrentUrl().equals("https://www.douban.com"));
```

5.7.3 测试结果

模态框的接受或放弃都够向父页面返回正确的值，在上面的测试情景下，能够正确停留在当前页面，也能够正确的返回到首页。测试通过。

6. 测试结果及总结

所有的测试都成功通过，证明豆瓣网的 web GUI 逻辑基本都是正确的。同时，组员们也都基本了解了 web GUI 测试的方法和操作。

```
D:\softwares\jdk-1.8\bin\java.exe ...
Starting ChromeDriver 74.0.3729.6 (255758eccf3d244491b8a1317aa76e1ce10d57e9-refs/branch-heads/3729@{#29}) on port 44222
Only local connections are allowed.
Please protect ports used by ChromeDriver and related test frameworks to prevent access by malicious code.
五月 26, 2019 8:39:56 下午 org.openqa.selenium.remote.ProtocolHandshake createSession
信息: Detected dialect: OSS
AssertOK: https://book.douban.com/
AssertOK: https://movie.douban.com/
AssertOK: https://music.douban.com/
AssertOK: https://www.douban.com/group/explore
AssertOK: https://shanghai.douban.com/
AssertOK: https://douban.fm
AssertOK: https://m.douban.com/time/
AssertOK: https://market.douban.com/
AssertOK: album check
AssertOK: note check
AssertOK: topic check
https://www.douban.com/
Success: testDropBox1
Success: testCheckBox1
Success: testCheckBox2
Success: testCheckBox3
Success: testSlideButton(正在热映)
Success: testSlideButton(最近热门movie)
Success: testSlideButton(最近热门tv)
Success: testSlideButton(热门推荐)
Success: testTagButton(最近热门movie)
Success: testTagButton(最近热门tv)
Success: testSearch
Success: testComment
Success: uploadPic
Success: modal box [dismiss]
Success: modal box [accept]

Process finished with exit code 0
```

图 21. 通过所有测试

豆瓣对于未登录用户和已登录用户的 GUI 界面不同，在本次测试中，有部分测试是基于未登录用户做的，其余部分是根据已登录用户界面做的。注意到豆瓣似乎有反爬虫机制，当测试过于密集时，会降低相应速度，因此我们在代码里显式的加入了暂停动作的代码。另外，在测试过于密集时，GUI 界面的代码也会有改变，导致一些组件不能被 Selenium 得到。

7. 参考

1. 主题：使用 selenium 测试 showModalDialog 模态对话框：
https://blog.csdn.net/aerchi/article/details/8102104

2. webdriver 对于模态窗口的处理：
https://blog.csdn.net/xie_0723/article/details/50617100

3. 自动化测试(8) | Selenium Java WebDriver 基础：
https://www.jianshu.com/p/3c05e8c9ee81