摘要

近年来，互联网信息科技的蓬勃发展，给软件科技行业带来了无限的机遇和挑战。由于 Web 软件系统具有平台无关的特性，而且还有并发性、异构性和分布性这些性质，Web 软件系统的测试相比于普通软件程序的测试要复杂很多，一个 Web 项目生命周期不只是开发的工作，测试在其中也有质量保证的作用，为了确保 Web 系统产品的竞争优势，自动化测试技术的地位与作用慢慢地展现出来。

本课题主要内容是以 Selenium 自动化测试技术研究与应用为主，通过 Jenkins 开源 Web 系统测试为实例，对 Jenkins 的部分功能进行需求分析、测试计划设计和测试用例设计与执行。使用 Selenium 能够帮助测试工程师模拟用户在浏览器中操作，从而达到测试的目的，而且，Selenium 兼容于多个浏览器平台并且能够使用不同的测试脚本语言进行编写，方便测试工程师掌握与应用。

关键词： 自动化测试 Selenium 开源 Web 系统 测试脚本

Abstract

In recent years, the rapid development of Internet information technology has brought unlimited opportunities and challenges to the software technology industry. Because the Web software system is platform-independent, and has the properties of concurrency, heterogeneity and distribution, the testing of the Web software system is much more complicated than the testing of the common software program, a Web project life cycle is not only the work of development, but also the role of testing in quality assurance. In order to ensure the competitive advantage of Web system products, the status and role of automated testing technology is gradually revealed.

The main content of this topic is the research and application of Selenium automation testing technology. Through Jenkins Open source Web system testing as an example, Jenkins part of the functional requirements analysis, Test Plan Design and test case design and implementation. Using Selenium can help test engineers simulate user actions in browsers for testing purposes, and it is compatible with multiple browser platforms and can be written in different test scripting languages, easy for test engineer to master and apply.

Key words: AutomatedTesting Selenium OpenSourceWebSystem TestScripts

目录

图表目录

1. 绪论

1.1 课题研究背景

现如今中国互联网信息科技技术的飞跃式发展以及中国互联网基础措施建设的不断完善，伴随着越来越多国内外先进技术被快速地运用和推广，众多企业走上了信息化的高速公路，发展的企业业务慢慢地快速化和国际化。在这个“百花齐开，竞相争艳”的信息科技时代，确保 web 系统保持稳定和性能达到要求的任务变得任重而道远，鉴于开发周期非常有限和公司财政支出等问题，测试人员时常会因为时间不足和工期赶而无视了一些本来需要注意的测试点或者使漏测了某些功能，这就是其中一个项目上线后产生问题的原因。没有充足的测试时间进行测试的情况下，线上产生了问题，项目经理首先第一个想到的是项目上线前最后把关的测试人员没测好，这就是其中一个测试人员为项目中产生的问题反馈“背锅”的原因。相对于敏捷开发给测试带来的问题，测试自动化就是团队工作中做好敏捷的一个必要实践。

2004 年 Jason 创造了 Selenium，富有多年测试经验的他，敏锐地注意到每次手工测试重复的工作太浪费时间了，于是他利用自己扎实的编程基础，开发了一个 JavaScript 库，达到与页面进行交互的功能效果，从而实现跨浏览器的环境下运行测试，最后，这个库就变成 Selenium RC 和 selenium IDE 的底层核心——Selenium Core。Selenium 作为一个工具不可能没有缺点，因为它一开始是基于 JavaScript 开发的，浏览器对于 JS 引擎的支持和安全方面的限制，使得有些情况下，它没法工作，更糟糕的是，随着浏览器不断完善，Web 应用也变得越来越复杂，这些因素更加阻碍了 Selenium 的工作运行。一个在谷歌工作的工程师西蒙·斯图尔特在 2006 年的时候开始了一个新的项目——WebDriver。Google 使用 Selenium 已经有很长的一段时间了，但是工程师却一直受制于产品的某些功能，而 WebDriver 的存在就是为了去解决 Selenium 所遇到的问题。到了 2008 年的时候，Selenium 和 WebDriver 开始整合，这是多么令人振奋的一件事，两方优势互补成为崭新的测试自动化利器。

1.2 课题研究意义

研究 Selenium 的 UI 自动化有什么意义呢？这个课题就是想要知道这个问题的答案。每个项目的产生都是因为需求的产生，测试人员就像以前机械工业还未发展的制衣工厂里面的裁剪人员，所有工序都需要手工进行操作，通过个人经验保证产品质量，逐渐地，当工厂的规模和速度呈上升趋势，这样的纯手工的弊端就会一一显现出来，因为生产规模的扩大而需要更多的人手，致使成本上升；因为生产要求

速度和工作量超过工作人员的极限而产生效率瓶颈；因为持续工作时间的增加和为了保证项目工期，致使工作人员疲劳，有时还会出现惯性思维，认为之前做的都没问题，现在做的也没问题，甚至会出现偷工减料的情况，这些人为错误造成产品质量的下降。而自动化测试就像工厂引入机械代替部分人力，它的产生就是为了解决纯手工测试所带来的弊端。当工作量增加，机械能够帮助工作人员完成部分工序，就厂商而言，避免了人力成本的增加，还能够合理安排工作人员完成一些机械无法完成的事情和维护机械并且利用个人经验保证产品质量。我从中提炼出自动化测试研究的意义是把人从重复度高、比较轻易犯人为错误的业务功能中解放出来，把这些工作都交给机器自动化运行，测试人员能够有更多时间专注于复杂度高的测试工作。而且自动化测试不单单是为了保证工作效率，还需要保证了产品质量，不应该为了自动化而自动化，也不是因为其流行而进行自动化，而是将自动化思维融合到整个项目流程，实现其该有的意义。

1.3 国内外研究现状

从 Selenium 的发展历史可以看出，这个自动化测试工具时至现在 2020 年了依旧在不停地完善和增加功能。2004 年只是一个 JavaScript 的库用来和页面进行交互，后来完善成 Selenium Core 作为整个软件的重要组成部分，Selenium RC 因为兼容多种语言，使我们变成的限制减少了。2006 年谷歌的工作人员西蒙，开发了 WebDriver 解决 Selenium 所遇到的问题，最后到 2008 年把 Selenium 和 WebDriver 整合成新的 Web 测试的软件。Selenium IDE、Selenium Grid 和 Selenium RC 这些组件造就了 Selenium 1.0，而 Selenium 2.0 使用简化的公式来表达：Selenium 2.0 = Selenium 1.0 + WebDriver， Selenium 2.0 中重点是 WebDriver 这一个驱动，它能够胜任为 Selenium RC 的升级版。Selenium 团队在 2013 年的时候曾经在官方博客预告圣诞节发布 Selenium 3.0，但真正发布 Selenium 3.0 第一个 beta 版已经是 2016 年了，而 Selenium 3.0 用公式表示：Selenium 3.0 = Selenium 2.0 － Selenium RC(Remote Control)。在 2018 年，Selenium 的开发者西蒙在一次会议上宣布了 Selenium4.0 什么时候发布和更新的核心内容。2004 年到如今 2020 年，十六年的时间，国外的工程师在使用 Selenium 的同时，一直不断地完善这个工具，我们能够从 GitHub 上看到 Selenium4.0 的开发进度。

而在国内，懂得使用 Selenium 是如今行业进行自动化的一种尺度，想到自动化第一想到的工具也是它，能够运用 Selenium 协助测试人员实现数量多且反复的工作。国内企业存在两种情况，一种是企业大量倚靠 Selenium，QA 工作人员团队所操作没有技术含量的工具开发的基础亦是如此，例如 RobotFramework，它可以让不懂代码的同学也可以使用其来进行 Web 自动化测试。另外一种清楚了自动化的关键性，

却无法完成自动化测试的质量保证专家，基于 Selenium 开发的软件能够让所有拥有 Web 基础知识和对浏览器有一定认识了解其原理的工作人员完成布置环境的工作，况且对 Web 的性能质量监测和循环测试也不一定需要清楚 Selenium 的源代码和原理，只需要去进行运用。由此可以比较出，国内的 Web 自动化基本是依赖于国外进行研究的，而且通过翻阅参考文献，我了解到国内只是对自动化的思想进行深入研究，而且大部份是基于 Selenium1.0 和 Selenium2.0 的测试框架，说明 Selenium3.0 虽然在国外已经发布三年多，国内依旧还处于摸索时期，由此看来，此次课题是基于 Selenium3.0 的 UI 自动化学习就有意义了。

1.4 课题研究的主要内容

本课题主要学习 Selenium 并进行 UI 测试自动化实战，测试人员不需要亲自动手就能够模拟真实的用户在 web 系统进行界面和功能的测试。而我本次课题研究的主要内容是研究 Selenium3.0 的各方面内容，明确自动化测试的意义与目的，从中清楚 Selenium 这么多年为什么能够在众多 Web 自动化测试工具中屹立不倒。此次毕业设计作品中，主要是基于 Java 编写 Selenium 的测试脚本，通过各种脚本运行驱动使浏览器进行 UI 自动化，最终通过分析数据对比功能测试和自动化测试的异同，从而深入了解 UI 自动化测试，但这些的前提都是有扎实的编程知识为基础。

1.5 本章小结

自动化测试的执行依靠的是机器，但自动化思想执行的却是人。通过了解 Selenium 的发展史和国内外研究的现状，明确自动化测试的意义和目的，清楚什么时候在什么地方对什么目标自动化，才能提高工作的效率，通过扩展自动化思维，把界面普通且重复固定输入和固定操作的功能和流程进行自动化，让测试人员能够顾及到其它复杂的功能和流程。接下来需要做的是研究好 Selenium3.0 知识基础，通过 Selenium 自动化实战掌握这种技术，适当地进行相应的自动化扩展，了解其历史背景和研究现状就好比测试人员在测试前了解清晰功能的需求和业务的流程操作一样，才能在后续的自动化实现过程中知道哪些需要编写自动化测试用例脚本，操作起来才不会迷茫为什么进行自动化测试。

2. Selenium 自动化测试技术研究

2.1 软件测试自动化研究

自动化就是通过手写一些机器能够看懂解析并运行的语言，或者使机器根据用户需要循环完成一系列操作，其中相应 UI 界面数据的校验和功能的实现与应用，页面跳转是否实现，都是自动化测试所要关注的重点。自动化测试包括三个层面：

- 单元测试自动化（数据处理层）针对单元控件进行排查和检验其代码，如 Java 的 Junit；
- 接口测试自动化（业务逻辑层）通过发送相应的数据获取返回值对比其是否符合接口文档要求，如 Postman；
- UI 测试自动化（GUI 界面层）UI 层是用户使用产品的入口，所有功能通过这一层提供给用户可视化图形界面进行操作，测试工作大多集中在这一方层 Error! Reference source not found.，如 Selenium；

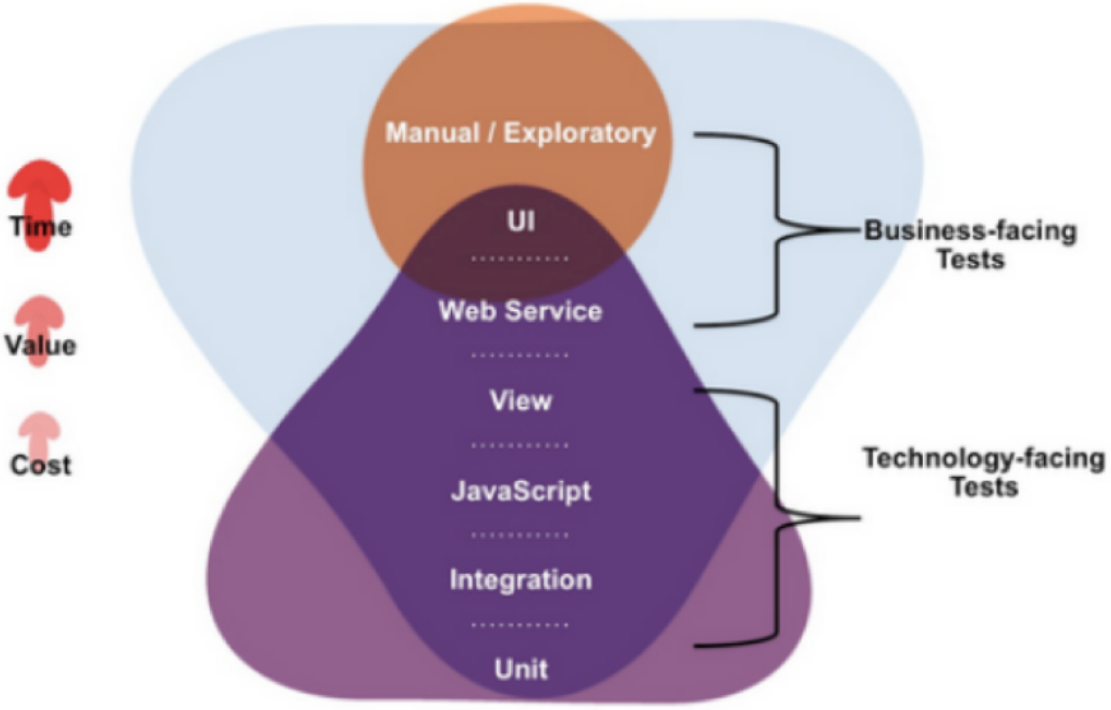

图 1 测试金字塔

众所周知，测试流程对应着开发流程，两者之间是互补互助的关系，缺一不可，测试更是占整个业务流程的一大部分。图片的越向下端走，意味着测试的作用更加的高，测试的效果提现得更加明显，底层是保障软件质量的基础，是测试的根基。就 Google 产品来说，大部分都是进行单元测试，期中集成、接口测试

占小部分的三分子二，而 UI 层自动化测试只占小部分的三分之一 Error! Reference source not found.。然而这不代表着 UI 测试并不受测试人员所关注，就像自动化趋势虽然这么多年来一直加强，手工测试也不可能完全丢弃，因为测试一定要站在用户的角度，站在业务层的角度，越往金字塔的上层，在测试的时候越能体会到用户感受。

做自动化测试的目的是使人们不沉寂在重复冗余的测试工作中，让测试人员能够尽可能多地对一些复杂化的测试功能点完整地按照思路做测试，主打回归测试来拔高软件的质量，完成一些之前版本已经完成的功能的反复查漏补缺，减少回归时所花费的时间，让测试工作更加流畅而且高效。举个例子：测试一个 web，分摊下来每人每天可能需要完成 500 个左右的测试功能点用例，所以 20000 个功能测试点点就应该一个半月左右才能结束，但是如果把这 20000 个测试用例在回归测试准备阶段转化成自动化测试脚本，到执行测试完成可能只需要 1 天的时间，然后再加上 1 天左右的测试结果分析，就把 20000 个测试用例完全跑完了，压缩出来的测试时间可以做复杂功能和新功能的测试，覆盖到每个测试点的同时还能确保测试效率。

俗话说“金无足赤，人无完人”，自动化测试就像人一样，有优点的同时，肯定也有不足。其缺点就是运行脚本需要不断的优化，反复的执行，大量的数据，最后才能得到用户想要的结果，这就意味着前期需要准备的工作量是非常大，才有后面收益的显著效果。说明自动化只适应长时间的迭代工作，并不契合赶项目的工作，并且如果数据和操作不是大量的结果也是比较难发现问题所在，一些隐藏的问题也需要时间进行挖掘。还有就是 UI 自动化并不是所有需求都转化成自动化，有一些还是需要人手工走一遍流程并了解其中测试要点才能够相对应进行测试覆盖，方便持续回归。最后，UI 自动化脚本执行不是特别稳定，会出现不可抗力的因素，比如说，有时候不是缺陷的地方也会反馈错误，执行到一半机器坏了需要人为干预。

结合以上所说，明白了手工测试并不可能被自动化所取代，机器终究是机器，就好比机器人再智能也不能取代人类的存在，机器运行的时候稳定性还是一般，有些判断过于主观造成判断是否为 bug 的时候，容易造成判断错误，而且自动化测试最终只是为了回归软件中的功能，在确保测试效率的同时保证软件中简单功能的质量，而不单单是为了发现更多的缺陷。

2.2 自动化测试技术的研究与比较

这么多年来，各式各样的 UI 自动化测试框架和工具，有的久经不衰，功能越来越多，测试所能达到的程度越来越广，有的却已经退出了历史的舞台。自动化测试技术应用主要分为三类，一类是通过录制并重复执行自动化，录制重复执行

的方法大多数是通过记录测试人员的操作行为以及记录被操作的屏幕坐标来开发用例，该方法似乎很容易，但如果应用发生一点小修改，对象定位的属性发生改变，录制好的脚本将无法使用；第二类是对文档对象模型（DOM）对象进行解析，这个是主流，但受特定平台限制，而且前端代码变化，脚本一般都要重写；最后一类是通过类似图像识别原理进行自动化操作，测试不易识别或无法定位的对象，不受前端代码修改影响，但如果屏幕分辨率、浏览器缩放等导致图片区域的大小发生变化就会失败 Error! Reference source not found.。

UI 自动化测试技术比较如下表 1 所示。

表 1 UI 自动化技术比较

比较项	QTP	Selenium	RobotFramework	Airtest
UI 元素的支持与管理	支持录制，兼容对象识别模式与图像识别模式	代码实现	关键字驱动框架	图像识别
元素定位准确性	中	高	高	取决于图片清晰度、分辨率、背景颜色等
浏览器支持	只支持 IE/Firefox	支持多种主流浏览器 IE/Firefox/ Chrome 等	支持多种主流浏览器 IE/Firefox/Chrome 等	支持多种主流浏览器 IE/Firefox/Chrome 等
开发语言	只支持 VBScript	Python,Ruby，Java 和 C#	Python,Java,C	Python
是否开源/免费	按照安装机器台数收费	开源	开源	否
代码移植性	可移植	可移植	可移植	本机（图片的识别会因分辨率、背景颜色、图片大小的变化而变化）

我之所以选择 Selenium 进行研究，就是因为通过 UI 自动化技术的比较，发现 Selenium 比较有研究的价值，首先就学生而言，测试工具是开源免费的肯定是首选，所以排除了 QTP 和 Airtest，其次，Selenium 和 RobotFramework 虽然都有元素定位准确性高，代码可移植，支持多种主流浏览器的特点，但 RobotFramework 是基于关键字的测试框架，维护的时候相比于 Selenium 多了一层伪代码，而且伪代码导致调试困难，致使工作量也加大了。

2.3 基于 Selenium 的 UI 自动化

从 Selenium 的发展史可以知道，Selenium 已经更新了 3 个版本，这不意味着 Selenium 是一个简单的工具，其中的内部结构是有多个应用所组成，每一个组件都有其特征和所涉及的操作场景。

- Selenium IDE（集成开发环境）是一个创建测试脚本的原型工具，它是一个 Firefox 插件，实现简单的浏览器操作的录制与回放，提供创建自动化测试的建议接口。Error! Reference source not found.
- Selenium RC 是 Selenium 的核心工具，支持多种编程语言设计测试代码，通过 Selenium RC 的服务器作为代理服务器去访问应用从而达到测试目的，Client Libraries 和 Selenium Server 是其中的核心，前者主要用于编写测试脚本对 Selenium Server 的库进行控制，后者负责控制浏览器行为。Error! Reference source not found.
- Selenium Grid 的作用是在我们测试脚本数量比较的时候，抑或是系统平台之间交互的时候，能够在不同的机器或浏览器进程同时工作，提升并行速率。
- Selenium WebDriver 是 Selenium2.0 主要的驱动，相当于 Selenium RC 的升级版，由于应用的升级版本都应该向下版本保持兼容，Selenium3.0 中，WebDriver 真正取代了 Selenium RC，所以 Selenium RC 被移除。

而我本次课题研究主要针对 Selenium3.0，事实上，Selenium3.0 相比较 Selenium2.0 改动的位置就是移除了 Selenium RC，主推的还是 WebDriver，所以我的课题研究也是围绕 WebDriver 实现的。Selenium3.0 工作原理如下图 2 所示。

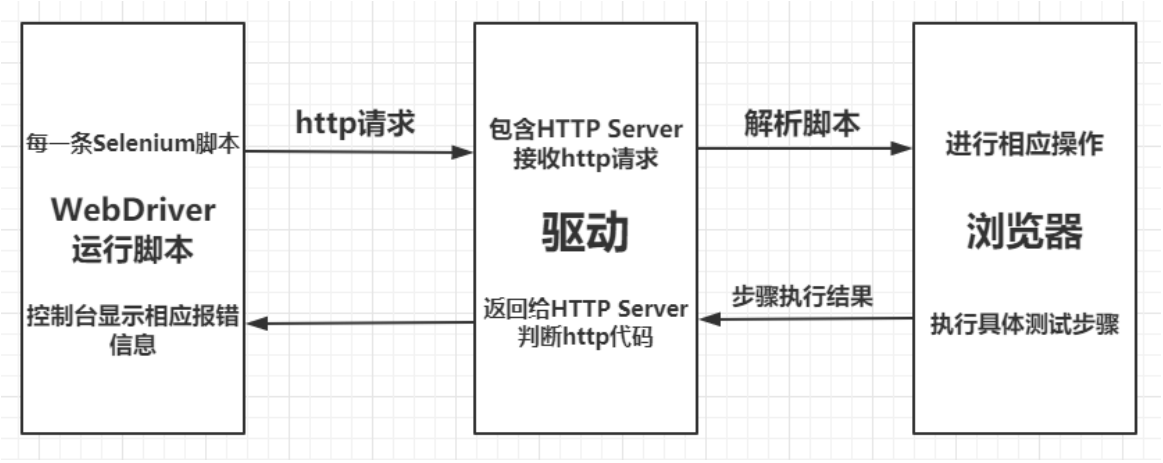

图 2 Selenium3.0 工作原理

之所以使用 HTTP 协议，是因为其作为标准协议的地位处于网络中，差不多所有的语言都拥有多姿多彩的 http 函数库， Client 和 Server 的请求响应都能够获取和处理这些返回数值，通过 Selenium3.0 工作原理图更能直观看到数据之间的传输与应用，知道哪个属于哪个，哪一部分接收哪一部分的数据。

2.4 本章小结

本章是为了让我明确做 UI 自动化的意义是站在用户角度去进行测试，尽可能满足用户需求，通过分析来解释本课题为什么选择用 Selenium 做 UI 自动化，俗话说得好“工欲善其事，必先利其器”，了解到 Selenium3.0 的工作原理，用起来才会得

心应手。打个比方，WebDriver 在控制台显示的报错信息，如果不知道它是对应 http

响应状态码，就会把整段报错信息复制花时间去找问题，事先了解到 WebDriver 为了给用户更明确的反馈信息，提供了更细化的 http 响应状态码的关键词 7 对应的 NoSuchElement、11 对应的 ElementNotVisible 和 200 对应 EverythingOK 等。

3. 搭建测试环境

3.1 Java 环境配置

由于我是使用 Java 实现 Selenium 的脚本代码，需要 Java 开发环境，所以到 Oracle 甲骨文网站下载较稳定且主流的 1.8 版本的 JDK 安装包进行安装，然后配置电脑中的环境变量并验证环境是否配置成功的步骤如下：

1、 打开环境变量配置窗口；

2、 新建两个新的系统变量，分别为 jdk 路径和 classpath 相应值“.;%Java_Home%\bin;%Java_Home%\lib\dt.jar;%Java_Home%\lib\tools.jar”，然后点击“确定”；

3、 在“系统变量”下找到“Path”并在其中添加“%Java_Home%\bin”和“%Java_Home%\jre\bin”；

4、 打开 cmd 命令窗口分别输入“java”、“javac”、“java -version”验证；

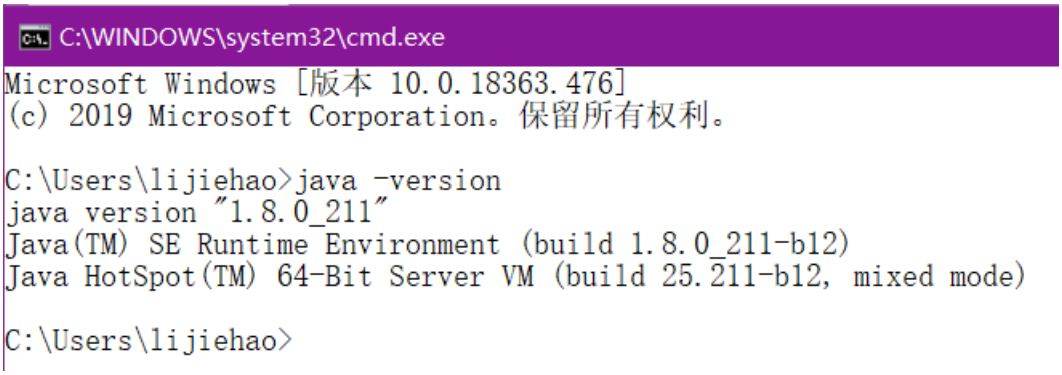

图 3 验证 Java 环境配置成功

3.2 Maven 环境配置

据我了解，传统的 Java 项目在创建的同时会新建一个目录命名为 libs，相关的库和 jar 包依赖导入这个目录，通过 subversion 上传到相应的位置，项目的工作人员功能够把这些数据都拉到本地存储，使得项目的依赖和库统一不会出现兼容问题。这种依赖管理的弊端是随着项目的增多，模块也增多，多个模块引用相同的依赖会产生冗余，还有就是项目合并时会因为依赖版本不一致造成问题。为了防止出现这些问题，我找到了一种方法能够像坚持党的领导一样对项目的依赖进行管理，每个项目只需要编辑 POM 文件就能够完成版本的一致，这种项目叫做 Maven，方便对项目的构建、信息和依赖合理的集成。就好比简单一个简单的 Java 项目想要使用 Selenium3.0 就需要下载相应的 jar 包并且进行 buildpath 操作，用了 Maven

构建项目直接在 pom. xml 中写入依赖语句就能直接调用 Selenium，简化了项目构建的过程从而提高工作效率。

Maven 的环境搭建步骤如下：

1、首先到 Maven 官方网站下载 Maven 的二进制解压包 apache-maven-3.6.3-bin.zip 到本地路径进行解压；

2、然后新建电脑环境系统变量，填写 Maven 解压后的路径；

3、在系统变量“Path”中添加变量值“%MAVEN_HOME%\bin”；

4、进入命令行输入“mvn -version”验证是否配置成功；

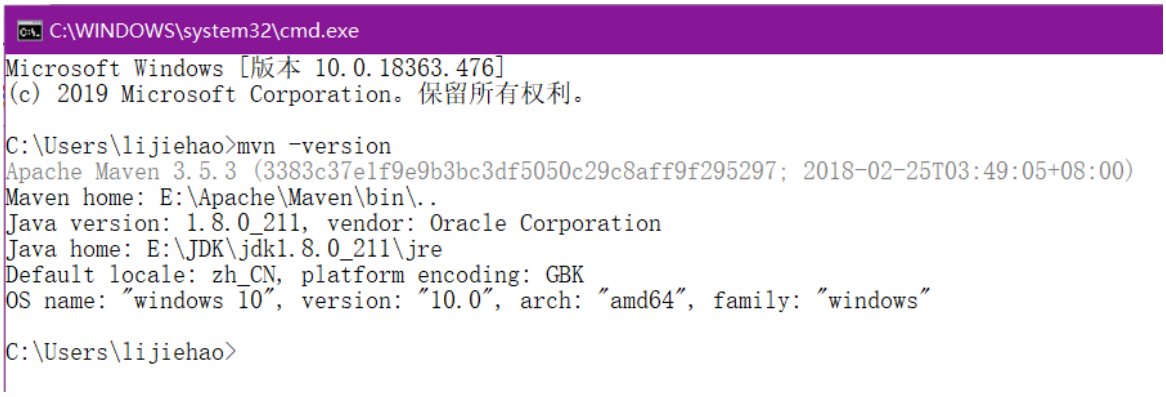

图 4 验证 Maven 环境配置成功

3.3 Jenkins 网站搭建

之所以使用 Jenkins，是因为其开源免费又是基于 Web 界面的持续集成平台，而且 Jenkins 的界面拥有常见的元素，方便研究 UI 自动化的元素定位的同时，后续也能使用 Jenkins 学习持续集成。

Jenkins 环境配置步骤如下：

1）到 Jenkins 的官方网站下载 Jenkins 的 war 包；

2）运行 war 包有两种方式：一种是基于 Tomcat、JDK 启动，一种是基于 Docker 启动，在本次研究使用 Tomcat 运行；

3）把 war 包放到 Tomcat 的 webapps 目录下；

4）直接使用浏览器访问 localhost:9000/jenkins（此处端口号 9000 是因为我 Tomcat 修改了原来 8080 的端口号）；

5）首次打开 Jenkins 会初始化配置，到 C:\Users\Administrator\.jenkins\secrets\initialAdminPassword 的路径下找到密码解锁 Jenkins；

6）首次初始化登录可能会耗费比较长的一段时间，插件可以后续管理运用，新建一个管理员用户就能够自动登录到 Jenkins 界面完成搭建了；

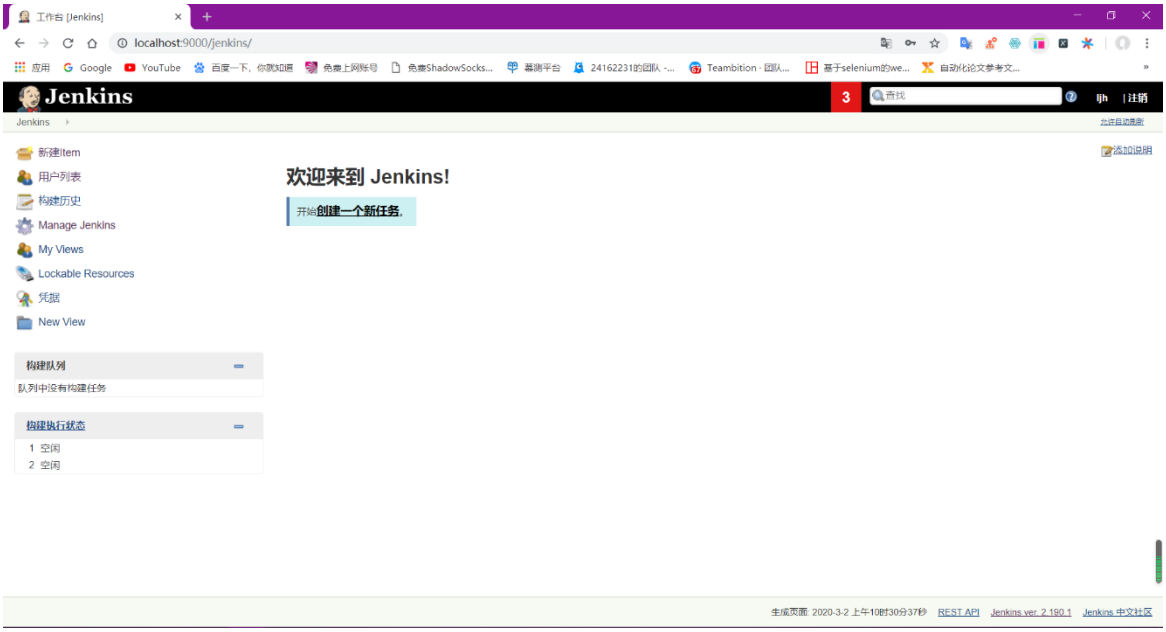

图 5 Jenkins 网站搭建完成

1）为了使其有注册流程，需要完成如下图 6 的选项勾选；

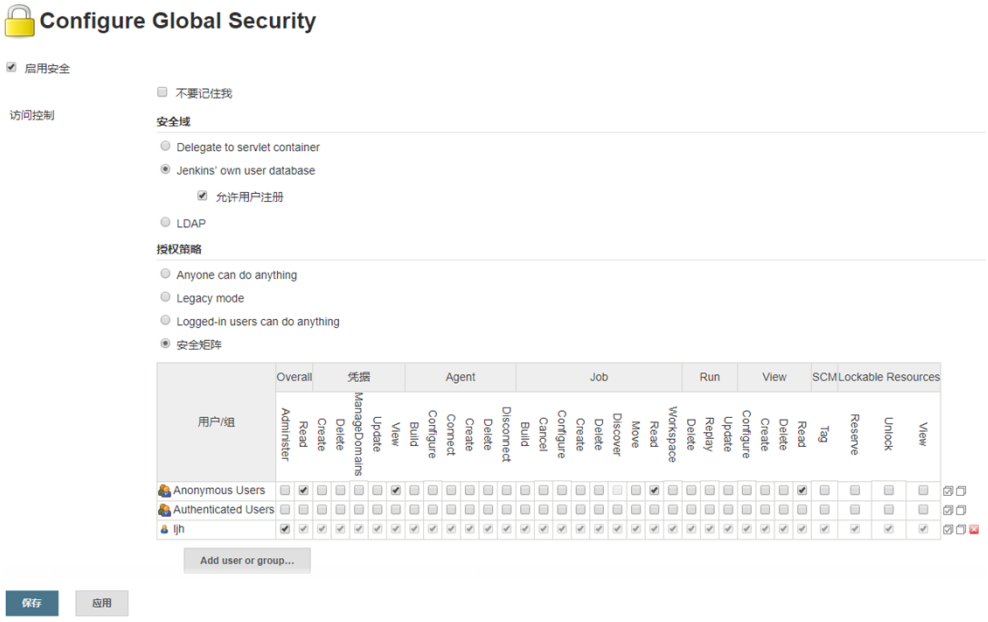

图 6 Jenkins 的全局安全配置

3.4 本章小结

完成测试环境的搭建，接下来就是真正开始进行测试设计与执行的阶段了，本章主要是需要了解为什么使用 Maven，从而提高工作的效率，自动化测试其实不单单是机器根据测试脚本进行自动化，我们能够把自动化的思维扩散到整个项目，能够有效率地进行项目构建与管理也是自动化的另一种表现。通过搭建测试网站来了解到其结构，从而方便下一步对网站进行测试设计，也清楚了为何使用其作为自动化测试的目标，掌握 UI 自动化测试

技术的同时，为未来更加深入了解持续集成测试而进行铺垫。

4. Web 自动化设计与分析

4.1 测试需求分析

4.1.1 业务需求分析

业务需求就我的理解中即是描述为什么开发一个系统，希望系统达到什么目的，属于业务层面，而 Jenkins 最主要的业务需求是提供一个监控持续重复工作的环境，但其中业务众多，因此不可能完全进行需求分析，此处只对登录与注册进行需求分析，注册业务最终输出的是一个用户账号能够进行登录操作，而登录最终目标是在系统输入账户和密码能够跳转到 Jenkins 的用户操作界面，所以，这个系统对于开发而言的业务需求就是开发一个可视化界面使其能够有注册登录的功能。

4.1.2 用户需求分析

用户需求是从用户角度出发，告诉用户应该做什么，相应的界面有什么功能，适当地指引用户完成操作，属于用户层面，此处的用户是进行持续集成的人，他们想要进入到工作可视化界面就需要有一个有权限 的账号，用户通过注册获得这个账号，使用注册的账号就能成功进行登录操作。

4.1.3 功能需求分析

业务需求针对的对象是业务的建设方，用户需求对应的是使用系统的用户，而功能需求对应的对象是产品应该具备的功能，属于产品层面，业务需求、用户需求和功能需求之间的联系是通过需求分析把前两个需求转变集成为第三个需求，形成一个完整的需求分析。

想要做好需求分析就需要了解业务流程，清楚每一个场景的每一个细节，广度和深度地解析功能。下面以 Jenkins 的注册和登录两部分业务为例。

注册阶段的业务需求是必须填写用户名、全称、邮箱和密码，用户需求是能简单填写就简单填写，有账号就停止注册操作，变成功能需求就是在输入框输入数据，并把数据上传到数据库，由于数据库或者前端的约束就需要对用户填写的信息进行判断，判断所有信息输入框是否为空，判断用户名输入框是否已经创建，对邮箱和密码输入框的格式进行判断，判断密码的强弱程度，除了判断外，还需要交互友好，若用户已经有账号能够通过点击“请登录”

直接跳转回登录界面，密码输入框能够通过勾选“显示”操作展示已输入的密码，能够有友好的密码输入提示，注册成功直接登录到 Jenkins 的工作台界面。如下图 7 和图 8 分别为注册 UI 界面图和注册业务流程图：

图 7 注册 UI 界面图

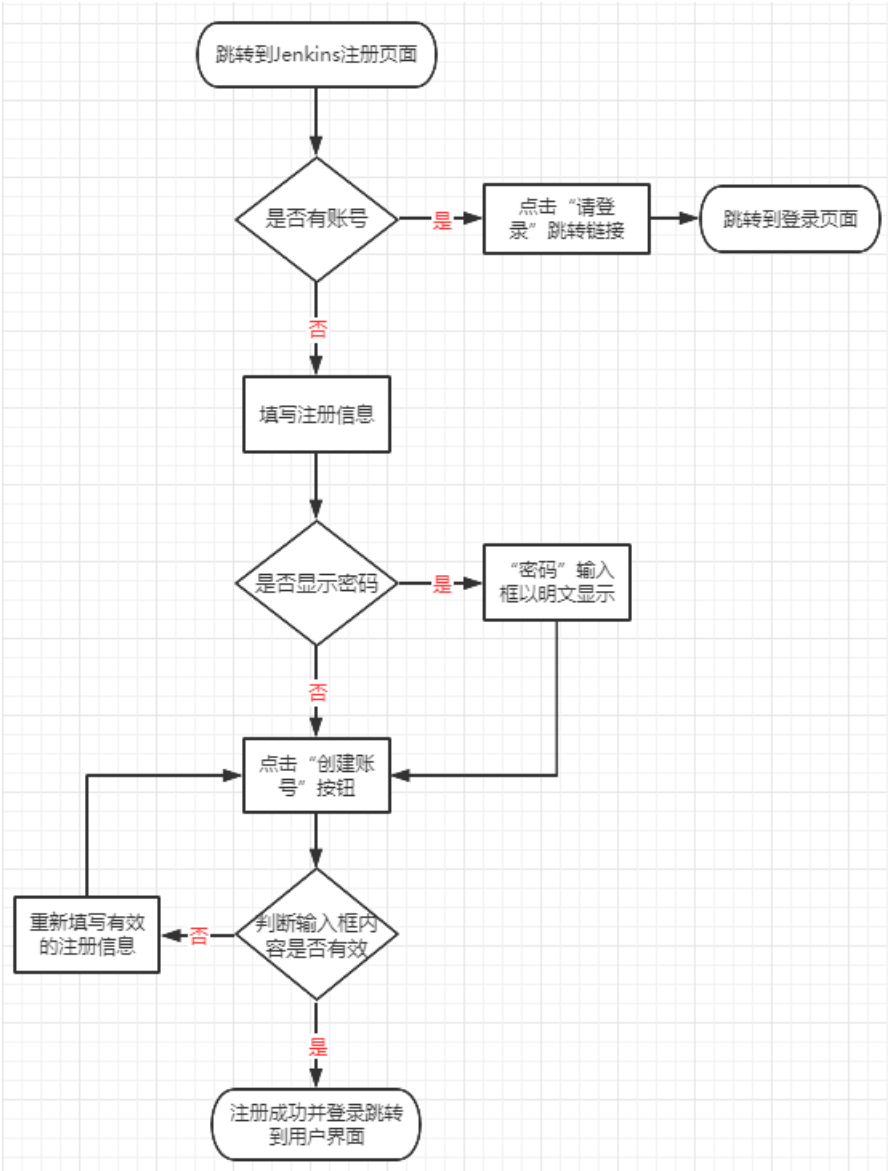

图 8 注册业务流程图

登录阶段的业务需求是必须填写用户名和密码进行登录操作，用户需求是使用已注册的用户账号是否能够进行登录，转化为功能需求就是有用户名输入框和密码输入框，用户没账号能够通过点击“创建一个用户账号”跳转到注册界面，判断输入框输入的信息是否为空和是否正确，若用户名和密码输入错误的数据是否有友好的交互提示“用户名或密码错误”，选中登录界面的“保持登录状态”，不注销的情况下关闭浏览器，重新打开并跳转 Jenkins 页面是否会保持登录状态。如下图 9 和图 10 分别为登录 UI 界面图和登录业务流程图：

欢迎来到 Jenkins!

创建一个用户账号 如果你没有注册用户.

用户名

密码

登录

保持登录状态

图 9 登录 UI 界面图

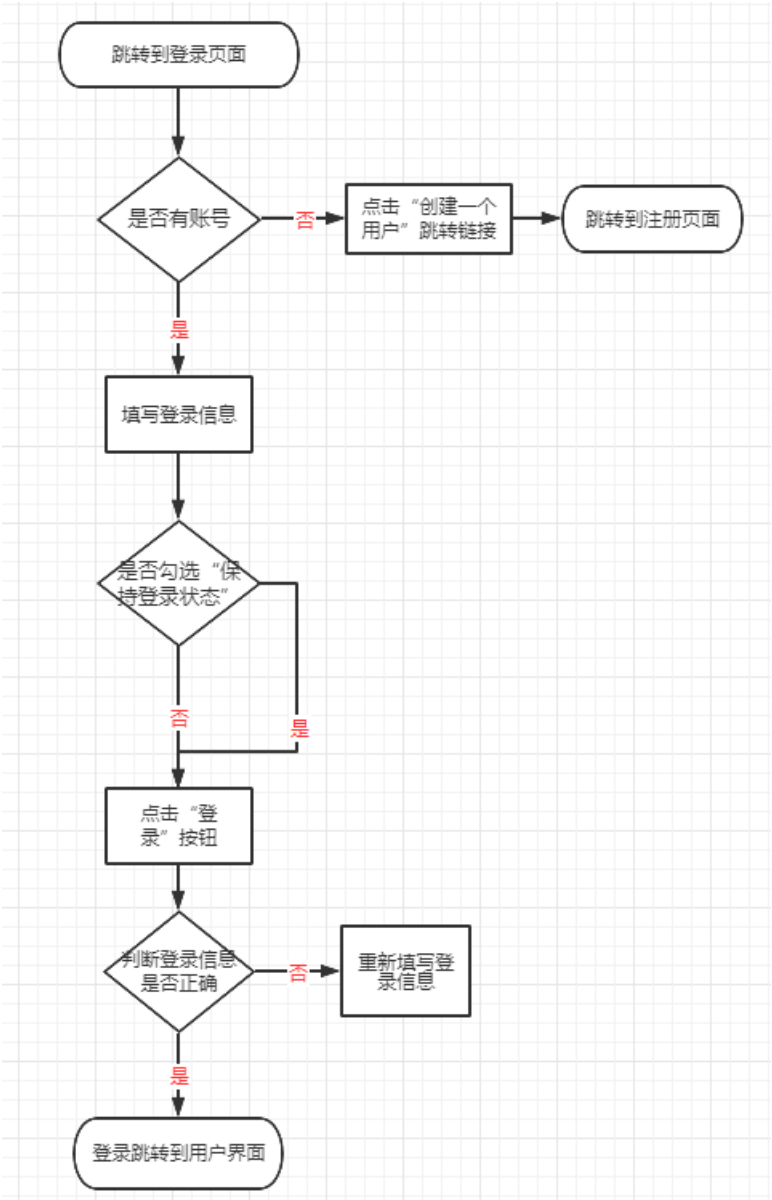

图 10 登录业务流程图

4.2 测试计划

在我们日常工作和生活中，经常需要做计划，它就像大海上指引船只的方向图，使船能够安全准确地到达目的地，让工作人员能够看到整个项目的进度安排和进展。当然，本课题主要研究 UI 自动化测试脚本，而 UI 自动化测试只是整个项目测试计划中的一部分，所以一般自动化测试都不会单一拿出来进行需求分析和测试计划的，而是基于手工功能测试基础下，选择部分合适的测试用例进行自动化测试的设计与回归。

测试计划主要的构造有引言、测试包含的模块、测试策略、测试需要准备的物资、过程需要的步骤、发布标准和风险说明。引言包括的内容是为了说明为什么这样编写，会读到这份测试计划的都有谁，借鉴了什么图书资源等

。测试范围的确定来自于需求文档，由于此次测试研究没有规范的需求文档，所以按照此次需求的目标确定测试范围是 Jenkins 中成功创建管理项目，涉及注册、登录和创建并管理项目这三个模块。理论上，自动化测试不是独立存在的，测试策略是根据项目测试类型的不同考虑不同的测试方法，而此处的测试类型主要是功能测试，测试方法是编写测试脚本的测试用例进行 UI 自动化测试，覆盖注册、登录和新建管理三个模块，编写测试脚本驱动不同的浏览器实现其兼容问题的反馈。测试资源主要是测试人员和测试环境，测试人员是本人，而测试环境上文已描述，所以此处不多加描述。测试进度安排如下表 2 所示。

表 2 测试工作进度安排

步骤	动作	负责人	进度	工作量（人日）
1	编写需求分析	李杰豪	2020 年 3 月 2 日	2d
2	编写测试计划	李杰豪	2020 年 3 月 3 日	1d
3	设计测试用例	李杰豪	2020 年 3 月 6 日	3d
4	编写测试脚本	李杰豪	2020 年 3 月 11 日	5d
5	执行测试脚本	李杰豪	2020 年 3 月 14 日	3d
6	优化测试脚本	李杰豪	2020 年 3 月 17 日	3d
7	生成测试报告	李杰豪	2020 年 3 月 18 日	1d
8	分析测试结果并多次回归	李杰豪	2020 年 3 月 21 日	3d

由于主要是研究学习自动化测试，所以发布标准可以说是不存在或者发布标准是 UI 自动化测试能够运行起来，而存在的风险是编写自动化测试脚本的时候遇到各种问题，比如技术上的问题，解决方法主要是依靠网络渠道。

4.3 测试用例设计

测试用例设计的时候涉及到功能、用户界面、考虑不同环境是否出问题、安全性、web 系统是否稳定和容量等，由于此课题主要研究方向为 UI 自动化，自动化的测试用例一般是从功能测试用例选择合适的进行脚本的编写，所以主要还是对功能测试用例、用户界面测试用例和兼容性测试用例进行设计编写。

编号	测试项	操作步骤	预期结果	数据	实际结果	结果比较说明
1	从注册界面跳转到登录界面	1、 点击“请登录”的字样	1、 点击后能够跳转到登录界面		1、 点击后能够跳转到登录界面	1、 a 标签实现跳转

图 11 注册流程其中一个测试项

如图 11 为注册流程

其中一个测试项，具体注册、登录和项目管理的功能测试用例参考附录一至三。我们可以通过参考功能测试用例，挑选并转化成合适的自动化测试用例，如下表 3 所示。

表 3 自动化测试用例 001

用例 id	Register001	注册中登录跳转链接	
步骤	**操作**	**数据**	**验证点**
1	打开注册页面	http://localhost:9000/jenkins/signup	
2	点击“请登录”链接	linkText 为“请登录”	
3	跳转到登录页面		匹配“欢迎来到 Jenkins！”的字样

Jenkins 系统的“项目管理”模块部分业务流程比较复杂，并不全适合进行 UI 自动化，因此我在“项目管理”模块只选择新增、重命名和删除这些简单又重复的测试操作进行功能测试用例的设计，其中，新建项目的时候默认选择“Freestyle project”直接点击“保存”，不需要配置项目增加自动化测试的复杂程度，配置项目的测试过程主要还是需要手工进行测试。

自动化测试用例其实就是为了方便实现自动化测试脚本的设计思维，通过网络资料，我总结了自动化测试用例的设计原理如下：

1）一个流程场景之间最好就不产生依赖，脚本之间尽量无关联，也就是说用例之间是相对独立的。

2）一个脚本只验证一个功能点。

3）不应该考虑过多逆向逻辑验证，尽可能只进行正向的逻辑验证，毕竟逆向逻辑的情况很多，验证起来会增加脚本繁琐程度，此外定位和操作的代码不一定能够找到相应的元素，导致运行时的不流畅，如果过多反向校验会容易出人为错误。

通过参考功能测试用例的过程，我们能够判断出哪些部分需要进行自动化，哪些部分需要进行手工测试，自动化测试脚本运行的同时能够把复杂流程进行手工测试，节省了时间和人力即为提高工作效率。

UI 即为用户界面的意思，上述的功能测试用例的设计就是为了验证 UI 功能的可行性，接下来我们也能够通过断言判断这个系统 UI 是否显示正确，不同的浏览器，用户界面和功能是否兼容，也可以通过自动化驱动不同的浏览器进行测试，相对的，编写脚本的时候我们也能够参考相应的测试用例，具体用户界面测试用例和兼容性测试用例参考附录四和附录五。

4.4 本章小结

测试用例是整个自动化测试的灵魂，有了测试用例就相当于有了测试思路，清楚测试的步骤和架构，并非任何的人为驱动测试用例百分百变成自动化的，需要测试人员进行筛选和思考。选型时，需要考虑成本，复杂流程尽量使用手工测试，选择的用例可以是像登录和注册这些重复执行的部分，能够帮助测试人员提高工作效率。转型时，测试人员应该了解脚本是怎么代替人工执行用例，当你写脚本的时候尽量简单、通俗、清晰，每一步骤都衔接好，自动化测试脚本运行起来后，测试人员不用花太多心思去维护，专心去对复杂的部分进行手工测试。事实上，不管是需求的过程，还是计划的设计与执行，抑或是用例的研究与转变，机器化和人为的测试并没什么太大的区别，都是为了了解并熟悉系统功能和流程做准备，方便测试人员清楚哪部分需要进行自动化。

5. 测试实施与优化

5.1 测试脚本编写

5.1.1 元素定位

元素定位失败是众多自动化疑难杂症之一，众所周知，Web 前端组成是 html 相当于一个大的框架，其中的样式和交互分别 css 和 js。Selenium 想要对页面进行操作，就需要有一个目标，这个目标就是 Html 中的元素。一个好的产品并非单单一个开发人员规范的开发习惯，如给页面元素加上唯一的 id，name 等，同时也需要测试人员编写脚本进行精准灵活的编写脚本定位。

比较基础的定位有 id 、name、className 等，这些都页面的基础标签属性，还有通过链接进行准确或者模糊定位，而 cssSelector 是根据 css 样式选择定位元素，使用“.”表示 class 样式元素，元素中的空格也要用“.”代替，“#”表示 ID 元素，“>”表示 html 的下一层级，还有就是通过 xpath 元素定位分为绝对定位和相对定位，绝对定位是“/”开头，从根目录开始一级一级找，若页面结构发生变化，则该路径失效；相对定位例如：//*[@name=’ phone’]形式，“//”表示匹配指定节点，不考虑位置，“*”表示通配符，匹配任意元素节点，“@”选取属性，“[]”属性判断条件表达式，优点是灵活方便，耦合性低。

```
//定位到密码
driver.findElement(By.id("password1")).sendKeys("123456");
//定位到创建账号按钮
driver.findElement(By.name("Submit")).click();
Thread.sleep(2*1000);
//注册成功之后，定位注销按钮
driver.findElement(By.partialLinkText("注销")).click();
```

图 12 Jenkins 在注册 UI 界面所使用到的定位脚本

如上图 12 看出我在 Jenkins 的注册 UI 定位时使用了 id、name 和 partialLinkText 进行定位，这些定位方法可以不唯一，比方说密码输入框，我们能够通过浏览器自带的开发者工具查看 html 的格式，通过选中结构体查看密码输入框所拥有的属性进行定位语句的编写，如下图 13 为密码输入框 html 的 input 标签能使用来定位的属性有 tabindex、name、id、type 和 class：

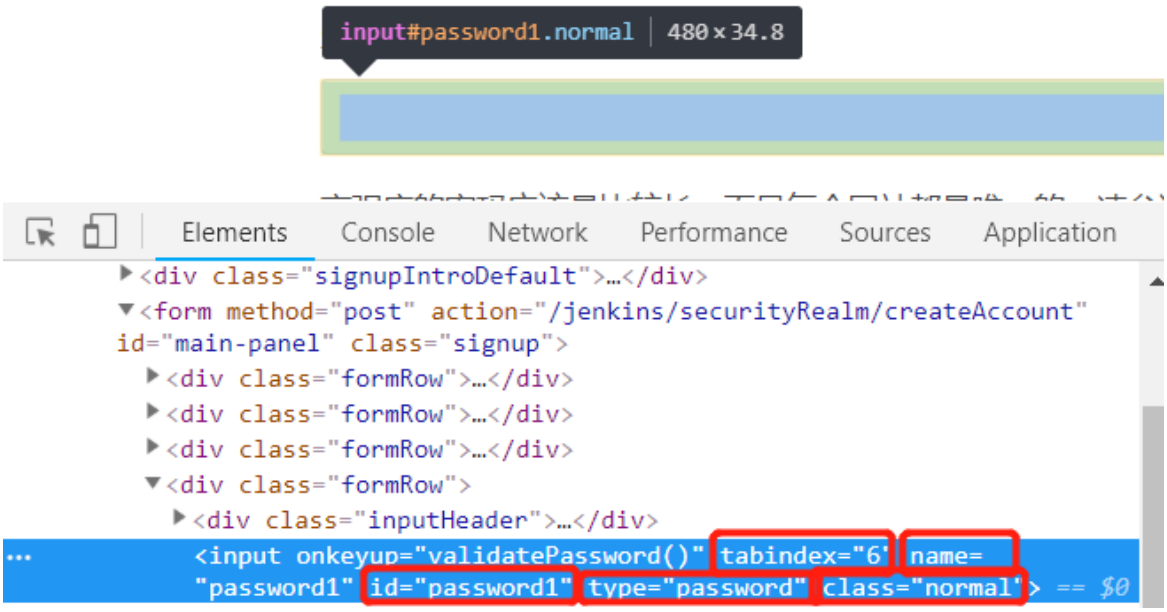

图 13 密码输入框的标签属性

当你编写的定位语句不能准确定位到元素，就能变通使用其它属性进行定位，优化测试脚本保证自动化测试能正常运行。

通过了解各种元素定位方法，使我们能够灵活熟练使用元素定位的同时，也能让我们冷静应对元素定位失败的问题，毕竟方法总比问题多，掌握多种元素定位的方法，总有一种能够精准定位元素，避免出现元素定位失败的问题，减少测试人员在维护上花太多时间。

5.1.2 元素常用 API

能够准确定位到页面元素后就是对元素进行操作，从而实现自动化操作，我们可以使用 click 方法对元素进行点击，使用 clear 和 sendKeys 能够清空输入框和进行输入操作，脚本只实现自动化操作并不算是自动化测试，还需要断言判断页面是否如预期结果般正确显示，当然这步骤也是交给机器进行判断，通过 getTagName 方法获取元素的标签名、getAttribute 方法根据想要的部分获取其中的内容、getText 方法获取元素文本值和 isDisplayed 方法查看元素是否显示，这四种元素操作 API 的方法获取页面显示结果进行断言。

如下图 14 为常用的点击 click()和输入 sendKeys()方法：

```
//定位用户名
driver.findElement(By.id("j_username")).sendKeys("lemon04");
//定位密码
driver.findElement(By.name("j_password")).sendKeys("123456");
//是否保持登录
driver.findElement(By.className("Checkbox-text")).click();
```

图 14 click()和 sendKeys()方法

5.1.3 driver 对象常用 API

Selenium 和 WebDriver 两个框架合并后，一个框架的缺陷就被另一个框架所弥补，比如 Selenium 必须操作真实浏览器，但 WebDriver 可以仿造人为运行的行为，在机器的内部实现一切的活动，更加轻便。又比如 WebDriver 无法同时进行操作多个浏览器和机器，Selenium 中的 Grid 组件协助实现用户想要的功能。而且 WebDriver 的存在替代了复杂的 Selenium RC，变得更加简洁易掌握，方便用户提高工作效率。

与 Webdriver 相关的常用 API 有如下十种：

1）get(String url)访问指定页面

2）getCurrentUrl()获取当前页面的 url 地址

3）getTitle()获取当前页面标题

4）getPageSource()获取当前页面源代码 Error! Reference source not found.

5）quit()关闭驱动对象以及所有相关窗口

6）close()把浏览器关掉

7）getWindowHandle()和其复数形式能够知道相应浏览器窗口的编号进行操作

8）manage()此方法可以获取 Options 浏览器菜单操作对象，语句是 driver. manage(). window()

9）navigate 对象可以用来访问指定 url 地址、刷新页面、回退或前进等操作

如下图 15 为 Webdriver 中的 quit()方法按照自己的需要进行二次封装，方便调用：

```
/**
 * web自动结束方法,休息5秒中后退出浏览器驱动
 * @param driver
 */
@Step("关闭浏览器驱动")
public static void finish(ChromeDriver driver) {
    try {
        Thread.sleep(5*1000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    driver.quit();
}
```

图 15 quit()方法二次封装

5.1.4 元素三大等待

在 UI 自动化测试中，必然会遇到环境不稳定，比如说页面元素处于加载状态，网速宽带慢等情况，若测试工作人员无动于衷地看着，这些报错的问题对分析根本

毫无意义。所以此处就像过马路等红绿灯一样需要等待，而 Selenium 等待分

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：
https://d.book118.com/227100152036006060