本科毕业设计（论文）

基于 selenium 的 web 应用自动化测试框架的设计与实现

Design and implementation of web application
automation testing framework
based on Selenium

学　　院：软件学院

专　　业：软件工程

学生姓名：XXX

学　　号：XXX

指导教师：XXX　副教授

北京交通大学

2016 年 5 月

学士论文版权使用授权书

本学士论文作者完全了解北京交通大学有关保留、使用学士论文的规定。特授权北京交通大学可以将学士论文的全部或部分内容编入有关数据库进行检索，提供阅览服务，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：　　指导教师签名：

签字日期：　　年　　月　　日　　签字日期：　　年　　月　　日

中文摘要

随着互联网技术与 web 技术的不断健壮与发展，web 应用已经成为了现今主流的信息中介与平台，各类基于 web 的应用以其方便，快速，易操作的特点成为了当前以及未来软件开发的主流。

笔者所在公司的 web 软件程序系统则大多是监控实际生产，或者记录生产资料类型的产品类软件。这类产品在投入生产后，在需求变更后重新开发后，需要在保证本公司产品软件的质量保证情况下，同时要缩短测试时间，在最大的程度上，提升生产效率。笔者通过了解现下主流的测试技术以及专业人员的指导下，选用 selenium 框架结合 testNG 框架来搭建一套是用于公司生产特点的自动化测试框架。并在此基础上编写脚本，完成测试，保证产品的质量同时使软件尽快投入生产过程中。本文工作如下：

（1）结合生产软件，按照自动化测试流程对其进行详细需求分析，并根据产品的需求得出自动化测试框架的需求分析

（2）结合自动化测试需求分析，对自动化整体架构进行总体框架构思，定义其数据层，业务层和页面接口层

（3）重点分析框架三层架构，根据低耦合的原则进行分离，详细设计各个主要模块。

（4）根据详细设计中对各模块的设计，实现了自动化测试框架的开发以及自动化测试脚本的编写。

（5）结合功能模块测试结果，同手工测试结果，录制-回放模式测试结果进行比较，建模计算，证明自动化测试框架解决问题的实用性，正确性。

关键词：自动化测试框架；web；selenium；

ABSTRACT

With the continuous development of Internet technology and the robust web technology, web applications have become the mainstream of today's information intermediary platform, all kinds of web-based applications with its convenient, fast, easy to operate features as the current and future software development mainstream.

Author of the company's web software program monitoring system is mostly actual production or types of products record production class software. After such products into production, after the re-development of demand after the change, it is necessary to ensure the quality of our products to ensure that the case of software, while reducing test time, to the maximum extent, improve production efficiency. By understanding the author under the guidance and testing technology holds many lessons for mainstream professionals, the choice selenium binding framework testNG framework to build an automated test framework for the production characteristics of the company. And write a script on this basis, to complete the test, to ensure product quality while making the software production process as soon as possible. This work is as follows:

1) combined with production software, automated test procedures in accordance with their detailed needs analysis and draw demand automated testing framework based on demand analysis

(2) combined with automated testing requirements analysis, the overall automation architecture overall framework concept, define its data layer, business layer and the interface layer pages

(3) focuses on the three-tier framework, based on the principle of low coupling separated, the detailed design of each main module.

(4) According to the detailed design of each module is designed to achieve the development and preparation of automated test scripts for automated testing framework.

(5) combined function module test results, test results with manual recording - playback mode test results were compared with model calculations demonstrate the practicality of automated testing framework to solve the problem, correct.

KEYWORDS：selenium； automation testing frame; web;

目　　录

1 绪论

时至今日，互联网与 web 技术的不断健壮与发展，不同于传统的软件，web 应用拥有着它独有的特点：web 应用会有大量的用户来点击访问，web 应用的开发周期比较短，Web 应用的界面较为稳定，仅仅是根据客户的需求不断地更改关于它的功能。基于 web 应用快速，便捷，易操作的特点，web 形式的软件越来越受到人们的欢迎和公司的亲睐。

1.1 研究背景

笔者所在公司其生产分部所使用的软件多是 web 应用，而本公司的软件使用上却有着它独特的特点，它是首先开发软件，测试完成后并投入生产，在生长过程中去如果发现了实际问题，就需要对软件进行不断的更改，版本不断的翻新。这样就造成了本公司所使用的 web 应用开发周期短，使用时间长，维护频繁，版本更新快的特点。这也就导致了初期测试简单，后期要进行大量的重复回归测试。几乎每次更改都需要对其相关的功能进行测试，以保证质量，防止在生产过程中造成损失。

而传统的手工测试通过手工测试人员设计测试用例，准备测试数据，然后通过人为操作按照测试用例一步步测试下去，一个项目一般 500-800 个测试用例（包括正向，反向），一个测试组 7 人，一般要一周的时间，这样就要消费 50 人天。然后一旦生产过程中有需求不符合时间生产流程，或者由于人员认知，技术的更新等原因造成需求变动，部门就需要对软件进行更新。当软件完成又需要分派两个测试人员，为期约一周的测试，以保证生产的正确性。这不仅消耗了大量的人力物力，更会因为 1 周的测试时间，而耽误了分部的生产。而且手工测试很大的程度上会受到人为主观的影响而出现误差。

这时候引入的自动化测试，在软件开发阶段的同时组织测试脚本的开发。通过简单录制，转换成代码，然后对代码进行修正系统管理。自动化测试能解决手工测试所遗留的问题，但是生硬的录制，僵化的脚本，然而这些脚本包含了数据，如果，产品有多个权限用户，就需要多次登录脚本，如果登录需要反向测试，又需要多次数据的操作，然后有不同的结果。这样的自动化测试写起来麻烦，维护起来更是麻烦，修改一个功能，需要将脚本一系列的脚本进行修改，甚至直接脚本报废，还不如重新录制转化修正。

为了解决掉传统自动化测试的录制回放继而执行脚本，以导致的脚本重用性及维护性大大降低，所以这个时候就需要开发一套适合于自己，适合于公司项目的自动化测试框架。

1.2 现状与发展

软件测试是 90 年代提出的，而 90 年代后期慢慢兴起的软件自动化测试发展至今，整整二十年，国内外的自动化测试技术日趋成熟，而主流的自动化测试工具也有很多：比如 selenium，QTP，Win Runner，Watir, 等，其中 selenium 是 thoughtworks 公司的一个免费的开源 web 自动化测试工具；QTP 是 mercury 公司的基于回归测试的，针对 web 应用的，适用于 BS 架构的程序测试，winrunner 也是该公司推广的产品，不过现今不再提供升级服务，所以在使用上并不是很推荐；而 Watir 是国外一款比较流行的测试工具，是由 watir craft 公司推出的。

总之，现在主流的测试工具种类越来越多，而测试工具也越来越功能强大，越来越精细。

一般而言，自动化测试的发展有过三段历程：

（1）第一代自动化测试。在自动化测试刚出来的时候，自动化测试是通过录制测试操作然后在回放的方式进行的，这种简单的方式还不是我们现在所说的录制回放思想，他只是通过录制下测试人员对网页操作的步骤，然后再根据步骤模仿用户操作就好了，然而这种方式太依赖录制的环境了，无法形成一种规模。

（2）第二代自动化测试。这段时期，由于录制回放已经不能满足自动化测试的需求，开始使用脚本的方式驱动自动化测试，可以通过各种脚本语言编写结构化的脚本，并且集合 API 和 CLI，在此基础上进行自动化测试，这段时期，模块化和库的自动化测试思想开始逐渐的形成了。

（3）第三代自动化测试。随着自动化的发展，自动化给人们带来的好处使自动化测试工具逐渐变得越来越多，满足各种自动化测试的功能，随着软件工具的变多，自动化测试需求也变高，继而数据测试，关键字驱动等思想也逐渐成熟，随着对象化的出现，自动化测试渐渐地开始形成一种规模，开始被各个行业所接纳并应用在软件开发的过程之中。自动化测试从此发展的趋势也越来越快。

而现今各个公司也开始开发属于他们独有的软件测试框架，下面将描述几种同种产品的介绍：

淘宝 Autorobot 框架：基于 selenium 使用关键字驱动思想的一套自动化测试框架，该框架提供了两种不同的脚本设计模式，一种是零代码模式，通过选择页面元素及操作来满足测试用例的设计，另一种是则是通过编写代码。它能通过 chrome 的插件的方式完成页面元素抓取，然后 selenium 基本方法查找对应的页面元素，自然语言描述测试用例。降低了测试人员的技术要求。

华为，思科等：采用的是 TCL 脚本，因为他的可维护性以及简单的特性，再加上 C/C++的易结合性。

其他小公司：小公司目前多使用基于关键字的思想开发，其所需要的自动化测试框架还是脚本集成的框架，是具体功能实现的一个框架，每个层次对应着不同的步骤与配置，以中期利益为主。

1.3 课题的介绍

1. 3. 1问题与解决途径

前文曾经提过，笔者公司的 web 应用多是应用于实际生产的流程应用，此类应用在使用过后很可能会因为实际工作中的技术更改或者因为需求不清晰等原因导致应用程序需要修改，然后再次测试，再投入生产，这样便会导致生产的延误。

搭建框架之前，我们需要对具体的问题进行分析：

（1）公司统一规范了硬件，但是没有规定软件浏览器的使用，所以浏览的兼容性是手工测试无法彻底解决的

（2）软件需求变动往往只需要修改某一部分功能，而在测试的时候，则需要对所有的测试用例重复测试

（3）公司产品操作功能多，但是简单，多为点击，输入数值等枯燥重复的操作，这会给手工测试带来主观上的影响。

这些都是影响测试时间的因素。我们通过引进自动化测试，可以将重复枯燥的手工测试转换成机器驱动，但是在使用了自动化测试，我们仍然会暴露出一些问题：

（1）脚本自动化测试，大量的测试脚本，与数据，用例绑定在一起，对于后期维护性要求较高。

（2）测试脚本与逻辑绑定，修改逻辑，可能导致修改脚本代码或者导致脚本彻底报废。同样会导致维护性差

（3）脚本代码冗余，繁杂，可复用性不高。

（4）自动化测试日志报告简单，不能准确定位出错地点。

为了解决这些问题，便引进了自动化测试框架的概念，通过对现有自动化测试工具的分析以及现行框架设计理念的研究，通过 selenium 框架搭建适用于公司的一套自动化测试框架，以解决上文所提到的一些问题：

（1）selenium 的核心技术可以支持各种浏览器。

（2）选用数据驱动的思想，可以将数据与脚本分离

（3）分层架构，将功能逻辑与脚本分离

（4）testNG 集成，可以通过场景集成脚本测试。

（5）Log4j 与 reportNG 的引用，丰富日志，将日志分为等级不同的类别。生成报告更加详细系统。

而本框架主要是在 selenium，testNG 的框架集成基础之上综合其他辅助工具，混合数据驱动以及关键字驱动的思想。对整个框架进行合理的分层建模，细致的构建，最终形成一套适用于公司项目且能解决实际问题的框架。

1.3.2 论文目标与意义

本课题主要是设计并实现一个基于 B/S 系统的功能齐全，稳定的自动化测试框架，能够利用框架编写出简单的测试，能够对脚本进行管理与组织。本文拟采用 selenium 框架为核心，以 JAVA 技术为脚本语言，结合 testNG 框架集成，配合以其他辅助工具如 reportNG，log4j 设计的框架。

框架搭建需要实现以下几个目标：

（1）易写，业务表达直白，

（2）自动化能力易拓展，可添加其他功能

（3）框架脚本可维护性强

（4）框架脚本可复用
（5）兼容多个浏览器
（6）能执行单个用例，也可执行多个用例
（7）有详细的报表，详细的日志描述。
（8）能与其他集成工具集成

通过搭建本框架，可以将测试人员从大量机械重复，数据输入等操作中解放出来，可以在需求改变小部分的情况下，快速组织其他模块自动化测试，以达到缩短测试时间，快速投入生产流程控制的使用中，将手工测试和自动化测试合理搭配起来，在保证软件质量的道路走向标准化，成熟化。也通过对本框架的搭建，将他推广到其他系统项目中，给未来自动化测试提供一个良好的技术基础。

1.4 论文安排

第一章，绪论。主要说明了笔者研究课题的背景，以及当前 web 应用的发展，软件自动化测试的现状与发展，剖析原因并根据公司现有问题提出解决方法以及课题的目标。

第二章，软件测试理论与技术支持。主要介绍了课题需要的理论支持，简述软件测试以及自动化测试的概念，软件测试的目的以及专属名词，自动化测试的过程与目的，介绍框架主要技术 selenium 以及 testNG 技术。

第三章，HYATF 框架分析与设计。 主要介绍了引入自动化框架的需求与原因，然后根据公司项目以及待测系统的功能需求，分析框架的功能需求，然后根据对框架的分析进行概要设计，并对分层进行架构建模，对最后各个模块详细设计。

第四章，HYATF 框架的实现。 主要介绍项目的部分实现，描述根据笔者参与设计的模块以及流程进行具体的实现过程。分别从环境搭建，项目搭建，测试用例模块编写与调用，testNG 执行与运行进行介绍。

第五章，HYATF 框架的应用与结果分析。通过对 HYATF 的示例演绎简述框架的应用过程以及通过对两个模块的测试数据进行建模分析。

第六章，总结与展望。总结了本文，并且对未来 HYATF 的发展方向以及自动化测试的发展方向以及前景作出展望

1.5 本章小结

本章主要是通过对课题研究的背景，自动化测试的发展与现状，然后找到问题，提出解决的方法，介绍要达到的目标，并简单说明关于论文的安排。

2 软件自动化测试理论与技术支持

在前一章中，通过研究当前自动化测试的现状，以及根据本公司应用程序的特点，根据发现的问题，然后我们提出了要开发一款适合我们公司的自动化测试框架，那么在本章将对后面所要使用的理论以及技术支持作简要介绍。

2.1 软件测试及自动化测试

2.1.1 软件测试

软件测试是软件生命周期中重要的一环，软件设计完成后，对软件进行严密的测试，以此发现整个软件产品在设计过程中的存在的问题进行改正，在测试过程中需要制定详细的做测试计划并且按照计划严格执行下去。

软件测试的目的是为了发现各种缺陷，他只能证明已有，并不能证明未有，只能尽量减少缺陷，不能彻底杜绝。成功的测试是以尽可能少的测试用例检查出尽可能多的缺陷。

回归测试是软件测试中的一种，软开开发完成后，如果因为测试出现问题，或者对需求变动等原因造成代码修改，这个时候为了验证代码修改是否会出现新的缺陷，同时修改代码没有产生错误以及对其他的原有代码造成影响而进行的测试叫做回归测试，自动回归测试是可以降低系统成本，维护升级阶段的成本的。

2.1.2 自动化测试

自动化测试(Automated test):是指利用自动化测试工具，将人工测试的过程转变为由机器，软件，程序自动驱动然后执行的测试。自动化测试可以解决功能稳定，数据量大，大量重复行为的测试。可以将大量的人力物力解放出来，而且自动化测试没有手工测试那样，会受到人为主观因素的干扰，所以对测试操作以及 BUG 的可重复性均比手工测试要强。

当然，要引入自动化测试，必须要了解自动化测试的适用范围，虽说自动化测试拥有者很多优势，但是自动化测试同样也不是适用于所有的软件，而一般比较适合用自动化测试来进行测试的软件基本上都有以上几个特点：软件系统周期时间长，软件产品功能稳定，软件产品需要大量的数据测试，其中包括负载测试，压力测试，性能测试以及配置测试。因为这些测试都需要进行大量的，重复的操作，人力是不可及的，这个时候，引入自动化测试是一条理想的解决办法。

而比较适合自动化测试的有以下几种情况：

1.回归测试

2.产品类软件，后期版本更新维护

3.大量重复枯燥的操作的测试

2.1.3 自动化测试过程及目的

自动化测试的过程其实可以类比软件开发的过程，然后首先对功能需求进行分析，然后对分析的结果设计出项目的概要设计，并对概要设计的模块进行详细的设计，然后就是逐步逐步的去实现。而自动化测试的过程便首先需要根据需求说明书进行测试计划的制定，然后分析需要进行自动化测试的需求，根据需求去设计能够进行自动化测试的测试用例，在此时也可以准备对自动化测试的搭建了，最后完成自动化测试脚本的编写，但脚本编写完成就可以开始自动化测试的执行了。最后记录结果并对结果进行分析。

◆ 需求分析

自动化测试成本非常高，所以一般情况下能够不需要使用自动化测试测试，就不要用自动化测试去测试，不能因为要自动化测试而去做自动化测试，因此获得一个项目时，首先要对项目的功能的去进行分析，明确分析出自动化测试检查点，知道哪些需要做，明晓需要做什么，明确测试的方式，时间，资源和环境。然后再根据需求对实现的方式进行选择。自动测试过程如下图所示。

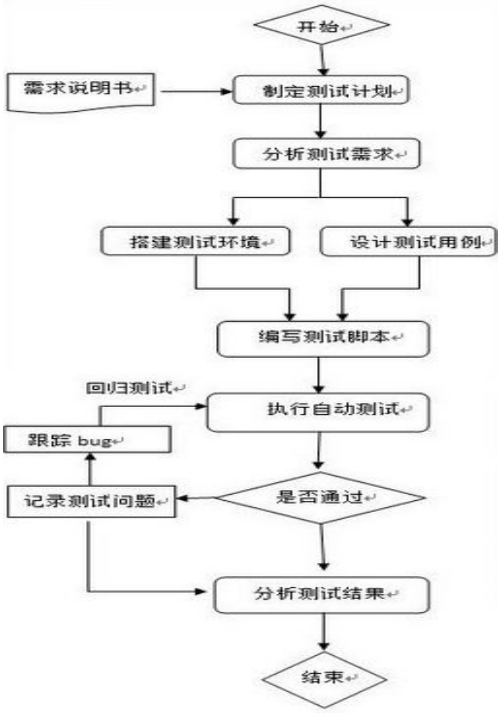

图 2.1.3 自动化测试过程

◆ 制定测试计划

自动化开始之前，首先需要做一个自动化测试的测试计划，在计划中对测试对象，目的等内容作详细的定义，同时也要对硬件，数据人力等方面资源做充足的准备。最后在测试计划完成过后，就可以将测试计划分发给相关的设计人员，使得相关人员能将测试所需要的用例设计出来

◆ 自动化测试需求

对自动化测试的需求是必须的，只有详细清晰的对项目自动化测试需求进行了解，测试用例设计的人员才能根据测试计划以及需求说明去设计测试用例，而本文中主要是

对回归测试中功能的一个测试，所以 web 应用的测试则必须要要覆盖链接，控件，功能，数据处理以及模块逻辑等 5 个方向的测试。

◆ 自动化测试用例设计

根据需求分析，设计出可以能够测试所有需求的测试用例，然后编写对应的测试用例文档，当然，手工测试用例是不能直接用来做自动化测试的，这个时候就需要把可以进行自动化测试的测试用例单独分离出来，并设计成自动化测试文档，同时可以将一些诸如登陆等固定模块分离出来。

◆ 自动化测试环境搭建

自动化测试人用例设计的同时，便可以进行自动化测试环境的搭建。在自动化测试框架环境的搭建包括页面元素，页面元素对象的添加，包括对软件部署，硬件调用等工作。

◆ 自动化测试脚本编写

脚本的编写是执行自动化测试的重中之重，通过不同难度对自动化测试用例获得页面元素控件，然后使用结构化的语言去组织执行，当然要辅助以验证性的检查点。在脚本的编写中，对脚本中数据与脚本的解耦可以使用参数化来操作，这一点后文将详细描述。脚本编写完成后，由于不同功能以及繁杂的大量代码，所以对脚本代码的统一管理也是很重要的。

◆ 自动化测试结果分析记录问题

对自动化测试的结果应该及时的进行分析，最好是能够每天都进行一次结果的分析，并且对软件的缺陷进行检查确认，确认这些上报的软件缺陷是否是真实的，如果使用开源软件，那么最好能够在进行二次测试，并能能够对软件缺陷进行缺陷跟踪。

◆ 自动化测试脚本维护

测试记录中的一些软件缺陷的记录，需要对应的缺陷管理系统，然后交由软件开发人员重新开发改正，使得最终的软件能解决缺陷，并且和客户能达到在功能需求上面的一致，在进行回归测试之前，对于脚本代码是需要修正和调试的。

2.2 环境与工具

2.2.1 selenium 介绍及组件

Selenium 是一款开源的用于 web 应用的自动化测试框架。Selenium 有很多底层函数库，通过模拟用户对实际 web 应用网页的操作，基于浏览器的驱动，并且验证是否正确完成操作来实现自动化测试操作的。selenium 支持各种主流浏览器。

Selenium 它的功能主要有两个方面：一、可以测试其与浏览器之间的兼容性问题，可以测试软件程序是否能够在不同的浏览器，不同的操作系统上良好的的运行。二、可以用来测试软件程序的功能模块是否正确，检验软件功能以及用户需求。同时 selenium ide 是还可以通过录制 web 操作然后回放，并且可以转换为各个版本脚本语言的脚本代码。支持 Net、Java、Perl 等各种脚本开发语言。

在学习 selenium 过程中，通常情况下都会了解以及使用以下几种 selenium 组件与工具，如下表所示

工具/组件	描述
Selenium IDE	Selenium ide 是一个基于 Firefox 浏览器开发的一个插件，可以录制测试人员操作浏览器页面，以记录他们的行为。
Selenium RC	Selenium 远程控制（RC）为旗舰测试框架，它允许多个简单的浏览器动作和线性执行。它使用的编程语言，如 Java，C#，PHP，Python 和 Ruby 和 Perl 的强大功能来创建更复杂的测试。
Selenium WebDriver	Selenium 的 webdriver 前身是 Selenium RC，直接发送命令给浏览器，并检索结果。
Selenium Grid	Selenium 网格用于运行在不同的机器，不同的浏览器同时以最小化执行时间的并行测试的工具。

表 2.2.1 selenium 各种工具

2.2.2 selenium 工作原理

Selenium 服务端 通过网络和 selenium 客户端进行通讯，然后接受 selenium 测试的指令。 selenium server 通过这些 JavaScript 函数把 Selenium core 嵌入到浏览器页面中，达到以程序对浏览器进行控制操作的效果，同时执行 selenium 客户端测试脚本中的用例，再通过网络把测试结果返回给 selenium 客户端，通过这种方式，从而实现 Web 应用自动化测试。

2.2.3 selenium 优势与不足

	优势	不足
1	开源的	Core 需要远程安装
2	简单，易于安装，易于工作	对现有控件兼容性差，比如文件上传以及日历控件无法用 RC 捕捉。
3	seleniumIDE 是 selenium 唯一可以再浏览器上记录用户行为的组件	不支持 http 连接方式
4	支持多种浏览器，支持多种操作系统，兼容性好	IEdriver 只能运行在 Windows 下
5	通过编写模仿用户行为，可以从用户的角度上测试页面元素	
6	能够录制回放生成脚本，支持各种语言转换	
7	测试用例是在浏览器上真实运行测试的	

2.2.4 Selenium 常用指令

操作（action）和断言（assertion）

操作（action）：selenium 模拟用户与 Web 应用程序的交互的具体操作。例如，单击按钮控件和填写表单输入文本框，这些常见的用户操作，可以用 Selenium 命令来自动化这些操作。

断言（assertion）：是用来验证一个命令的预期结果。而一些常见的断言就包括验证页面内容是否存在，当前位置是否正确。

Test Suite 和 Test Case：

Test Suite: 一组测试的集合，在实际中往往一个模块对应一个 Test Suite，在 Seleium 中显示在左上角。

Test Case：一个测试用例，包含多个 action 或者断言。

2.2.5 testNG

TestNG 是一个功能强大的测试框架，是 Junit 的一个增强版本，Junit 在使用多年之前，TestNG 才生效存在。NG 代表“下一代”。

testNG 好处：

- 注释可以帮助我们来组织使测试更容易。
- 灵活的测试配置。
- 测试例可以更容易地进行分组
- 可以使用 TestNG 实现测试并行
- 支持数据驱动 测试
- 内置的报告

2.2.6 testNG 的注解

通过注解，可以为测试用例赋值，这样可以将脚本和测试用例分离开来，同时通过注解的方式可以组织测试用例。如：

```
Public class 测试（）{
  @beforetest//测试前调用及准备
```

```
 Public dobeforetest()...
 @parameter （name1）
  @test
 Public dotest().....
  @aftertest//测试后操作
 Public dotest().....
}
<test name=’测试用例名’>
<parameter name=’name1’ value=’’></parameter>
<classes>
<class name=’包名.类名’/>
</classes>
</test>
```

这边是 test 注解的作用，通过修改 test 里面的值便可以改变脚本里面的值。以下还有 testNG 的一些基本注解：

注解	描述
@BeforeSuite	注解的方法将只运行一次，运行所有测试前此套件中。
@AfterSuite	注解的方法将只运行一次此套件中的所有测试都运行之后。
@BeforeClass	注解的方法将只运行一次先行先试在当前类中的方法调用。
@AfterClass	注解的方法将只运行一次后已经运行在当前类中的所有测试方法。
@BeforeTest	注解的方法将被运行之前的任何测试方法属于内部类的<test>标签的运行。
@AfterTest	注解的方法将被运行后，所有的测试方法，属于内部类的<test>标签的运行。
@BeforeGroups	组的列表，这种配置方法将之前运行。此方法是保证在运行属于任何这些组第一个测试方法，该方法被调用。
@AfterGroups	组的名单，这种配置方法后，将运行。此方法是保证运行后不久，最后的测试方法，该方法属于任何这些组被调用。
@BeforeMethod	注解的方法将每个测试方法之前运行。

图 2.2.6 常用注解

所以通过 testNG 驱动的使用能够很好的参数化测试。能够实现其他测试框架所不能实现的功能。使得测试方案最终的灵活多变。在这里可以实现驱动测试。

2.3 本章小结

本章通过对软件测试，回归测试，自动化测试的概念，过程，以及目的提供测试方面的理论支持；而通过介绍自动化测试主要核心技术 selenium 与 testNG 技术为后文提供技术上的支持。

3 HYATF 框架分析与设计

自动化测试框架的需求与分析，是离不开对待测系统的功能上的需求与分析的，也只有适用于系统产品测试要求的框架才是最好的自动化测试框架，才是最实用的自动化测试框架。所以框架的需求分析是要来源并依赖于待测产品或者产品类的功能需求分析。将产品需求以及框架需求结合起来，最终才能建立起一套适合于产品以及该类产品的自动化测试框架。

3.1 框架意向起源

华耀自动化测试框架（HuaYao Automated Testing Framework）（以下均简称 HYATF）是公司为华耀分部系列生产软件项目提起的自动化测试框架意向。华耀系列 web 应用包括华耀试样管理平台，试样仓库管理平台以及生产计划管理，棉纱计划管理等应用于华耀分部坯布生产过程的管理类平台，以试样管理平台为例，这类平台主要以添加信息，删改信息，删除信息，查询信息的功能为主。期间掺杂大量的重复枯燥的输入操作，一个坯布信息的添加，一个试样信息的添加，以及他大量对颜色染料等数据信息结合表示其技术点，这样就会造成测试过程中大量的数据输入，以及各种点击操作。该类软件产品的特点就是功能简单稳定，开发周期长，部分软件甚至需要边编写边投入使用，而且使用的时间很长甚至会一直用下去，这样就导致，软件会因为生产模式的改变，需求的改变，生产技术发展而功能对应改变，这也就导致了公司软件项目开发多以版本更新升级的方式开发产品。而同时每次的更新都只是少量的功能或者逻辑或者数据表的改变，而其他的功能不会改变。人为的测试可能会因为单调枯燥的数据填写而导致测试结果误差，甚至忽视继续测试，从而无法判断出版本更改是否改变了其他功能模块的功能，也正是由于这些特点，导致华耀系列产品急需自动化的测试流程，急需一套可以管理，部署，执行自动化测试功能的框架。

3.2 待测系统分析

华耀分部是公司的生产基地，公司信息中心软件部门为其开发了多款是用于生产的软件，这些软件的普遍功能是样式简单，数据繁杂，界面操作单一。主要多为输入框，按钮。用来记录并且统计生产数据，显示生产技术的功能。但是其功能不固定，随时会根据生产的改变而发生功能改变。而对于功能的修改必须要及时修改及时测试及时投入生产（如图 3.2 所示）。

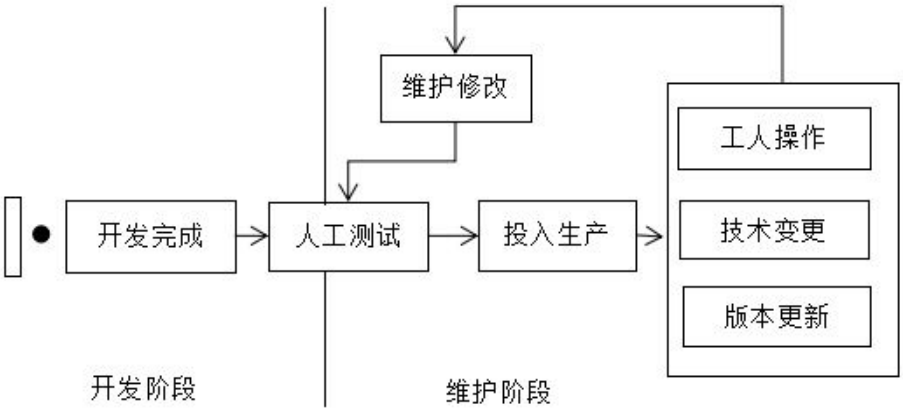

图 3.2 公司产品特点流程图

公司软件基本都是用于生产的产品软件，一款软件从他的生命周期开始一直到结束，其中在软件开发完成之后，产品软件便需要经过测试部的测试然后投入生产的使用过程中，在使用的过程中，由于实际工人操作费解等原因造成需求阶段隐藏问题，纱支，原料等技术的更改或者新型技术引进或者技术进一步提炼成熟等原因导致需求变更，重新开发。或者由于添加新功能新模块导致版本的大更新，这些时候不仅需要重新代码开发，而且还需要再次进入测试阶段，测试过程中不仅需要发现功能修改导致的新的错误，同时还要测试旧有功能模块是否有因再次开发导致其出现问题，所以需要重新验证性测试一次。

3.2.1 架构分析

公司生产过程使用的产品类管理软件多是使用 B/S 模型的 web 应用，故它也使用了经典三层结构开发的（如下图 3.2.1）。它有表现层（UI），业务逻辑层(BLL)；数据访问层（DAL）。

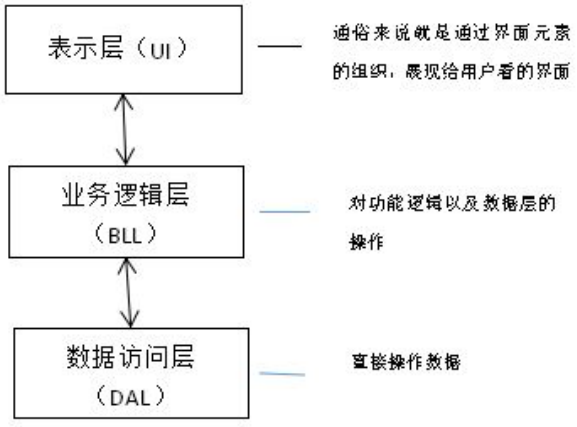

图 3.2.1 产品系统架构

3.2.2 功能分析

框架的开发是为了推广到华耀分部所有类型相似的产品的自动化测试过程中，此处以华耀试样管理平台为例，简单分析其主要功能特点：

（1）试样设计模块，试样的设计是通过内外踝口轮廓线，剪口深度，剪口密度等大量的数据添加来，这种表示以及管理的模式，到时他通过大量的文本输入框输入具体数据，通过这些数据达到添加试样的功能，以及修改，删除，查询等功能。

（2）相同的设计模块：菜单栏除了内地试样设计，还有包底围条试样设计，鞋面，鞋舌等多部分设计，这样也就导致相同功能模块。

（3）逻辑操作关联，系统功能上的功能相互依赖，前面功能的数据输入，后面的数据才能显示然后操作。这样也就是的第一个逻辑块输入输入多条数据，以供后面几个模块的测试。才能保证一条逻辑线上功能的完整测试。

3.2.3 其他分析

除了对测试的架构以及内容上的分析，还有其使用环境的介绍以及使用特点的介绍。

使用环境：统一硬件配置，某操作系统，内网，不同浏览器。

使用特点：不同语言的添加使用。

3.3 HYATF 测试框架分析

3.3.1 框架架构分析

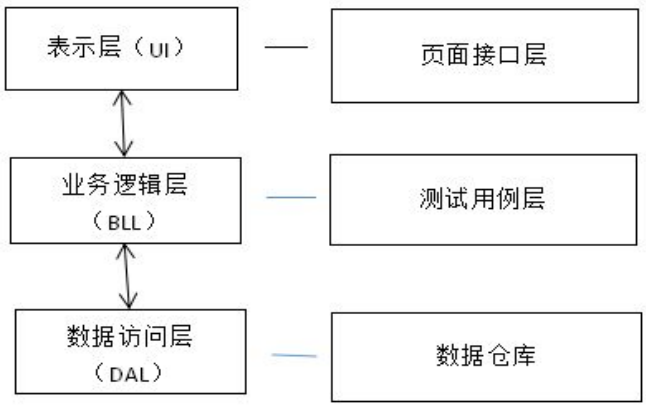

图 3.3.1 框架三层架构

在对之前软件系统的架构分析中，能知晓一般的系统应用使用了三层架构，这样在

分析框架的软件架构时，是不是可以参考一下软件系统的架构？

在系统架构中，通过表示层关于界面 UI 的设计开发，向用户展示功能模块以及显示。那么在框架中，需要创建页面接口层，页面接口层主要是对页面元素的对象化。

在系统的业务逻辑层主要用来表示系统功能的逻辑，所以框架的中间层便是用例层，通过对用例的操作展示框架中的脚本，逻辑。

在系统的数据访问层主要是数据库的操作，在框架的底层，由于大量的数据操作，可以使用数据仓库（层）对测试的数据进行存储，这也就体现了数据驱动的思想。

通过对系统的架构分析，就可以简要的将系统划分为页面接口层，测试用例层，以及数据仓库。

三层架构，本着“高内聚，低耦合”的原则，将数据与用例分离，用例与脚本的分离。

3.3.2 框架思想分析

根据第 2 章的内容，可以知道有很多的测试思想：

数据驱动比较方便设计与修改测试数据，但是不支持自动化快速开发，功能函数特别多。

而关键字驱动可以方便维护脚本与快速自动化开发。但是对于不断变化的测试数据支持却显得尤其乏力。

在系统分析过程中，我们的产品需要大量的数据输入，修改。所以通过分析这两种思想，模式，我们可以根据系统分析来选用数据驱动的思想来完成框架的思想选择。然而系统中的对象简单，如果通过封装对象，便可以节省那繁杂的函数了。两种思想均有其利处和弊端，那么有没有一种方法可以把他们整合出来呢，在这几年，这种方法渐渐在市面上流行起来。

混合驱动，也就是数据驱动+关键字驱动的框架架构思想。

可以对架构再次分析分层，分为底层支持函数层，脚本解析层，脚本层，用例层以及数据层。

底层支持函数提供一些对象的获取，脚本分析是对用例的解析生成 case，脚本层是各种执行 action，而用例层以及数据层分别是测试用例和数据的 xlsx 文件。

3.3.3 框架功能分析及工具选择

根据待测系统的分析，测试框架需要满足的基本需求应该包括以下几点：

- 产品应用与公司内网，电脑统一发配，系统统一，但是浏览器的使用会根据个人喜好，所以框架必须要支持不同的浏览器。
- 产品功能多，类型相似度高，但是功能修改频繁，升级频繁，所以好的脚本结构可以让测试人员更少的去因需求改动而修改测试代码，导致效率低下，所以框架需要有很好的复用性。脚本结构要易于维护。
- 有效的管理繁杂的测试用例，脚本。并且可以对测试用例重复大量的执行，以及数据的大量执行。

- 最好要有一篇定位详细的日志，可以通过日志来判断出错的具体位置。
- 有比较详细的测试报告，可以通过测试报告阅读测试用例的执行详细结果。

根据以上的功能，我们可以采用一些辅助工具来配合构建框架：

- Selenium Webdriver:Selenium 是可以直接运行在真实的浏览器上的，它模拟真正的用户去操作浏览器上的页面，就像是用户真正的在操作一样。同时 selenium 还支持 IE， Firefox、chorm 等多种浏览器。selenium 的主要功能有：一方面，可以测试与浏览器的兼容性问题，可以测试你的软件程序看是否能够在不同的浏览器以及不同的操作系统上好好的工作。另一方面，可以用来测试系统功能，检验软件功能和用户需求。同时 selenium 是可以支持录制回放和自动生成脚本代码。它适用于 Net、Java、Perl 等不同语言的开发脚本
- Maven：是基于项目对象模型，可以通过一段描述信息就可以管理项目的构建，报告和文档的软件项目管理工具。
- testNG：testNG 同样是一种测试框架，它集合了 Junit 和 Nunit 的功能，同时引入了一些新的功能，使其功能更加繁多，使用起来更加方便。同时，TestNG 还消除了很多的以前旧有的框架的一些限制，使得开发人员可以通过 testNG 编写灵活的编写和强大的测试。因为它在很大程度上借鉴了 Java 注解（JDK5.0 引入的）来定义的测试，它也可以告诉你如何使用这个新功能在真实的 Java 语言生产环境中
- reportNG：是一个配合 TestNG 运行 case 后自动帮你在 test-output 文件内生成一个相对较为美观的测试报告。

3.4 HYATF 的设计

3.4.1 框架设计准则

看待自动化测试，我们可以把它看做一个软件项目，而不仅仅只是一套录制回放的动作，它是需要良好的框架，然后再进行自动化测试。而且在自动化测试框架的设计的的时候，还需要参考自动化测试框架的一些设计准则。

什么是自动化测试框架，测试框架就是在自动化测试各个阶段，定义的一套准则规范与约束：需求分析阶段，脚本设计阶段，执行阶段，报告和维护阶段。它是对自动化测试内服复杂结构的包装，是对测试流程标准的强制执行。

1.选用合适的自动化测试类型。

2.不要过分的改造。

3.可重用性。

4.支持系统的不同版本

5.外部的可配置性

6.任何对象变动引起的变动应该是最小的

7.测试执行

自动化测试有时需要满足以下需求：

（1）执行单独的测试用例

（2）执行多个测试用例
（3）只执行失败的测试用例
（4）在一组测试用例的基础上执行下一个或一组测试用例
8.状态监测
9.报表
10.对工具的依赖尽可能的小
11.日志
12.易用性
13.灵活性
14.编码规范

3.4.2 HYATF 总体构思

基于上文对待测系统的需求分析，以及最后剖析出来的关于 HYATF 的功能需求分析，HYATF 框架作如下的概要设计：

1) 选用的是 selenium 基本框架搭建底层，模拟用户对页面进行点击，文本输入等操作，由于笔者熟悉 JAVA 开发，所以使用的是 JAVA 开发语言。
2) 使用网易 Dragger 框架对 selenium 进行封装，这一部分公司已经提供。
3) 通过 excel 和 XML 文件存储测试用例以及测试数据，测试数据以及测试用例完美的保存在 excel 表格中，然后可以通过 XML 来设计执行套，这样能更好的对测试脚本以及测试数据进行管理
4) 通过对日志的详细分析，准确定位，可以生成详细的测试报告并发送给相应的开发人员。使得测试报告更加详细，便捷。
5) 使用 Maven 构建工具实现自动化测试全过程的自动化，负责自动化测试环境的配置，测试脚本的编译，测试数据的调用，以及测试日志的分析，测试报告的发送等一系列活动的全自动化。

HYATF 主要适用于华耀系列的产品的自动化测试，可以通过程序编写来取消人工执行，可以将一次次版本更新，功能修改后的重复测试，大量数据执行化为自动化测试，对每次的测试点进行分析与定位，最终确保每次版本的更新后的回归测试正确，简洁的进行，能确保每次版本的更新升级，不影响原有的功能点，功能模块。

通过对框架的基本概要设计，能简单的勾画出 HYATF 框架的结构图（如图所示）：

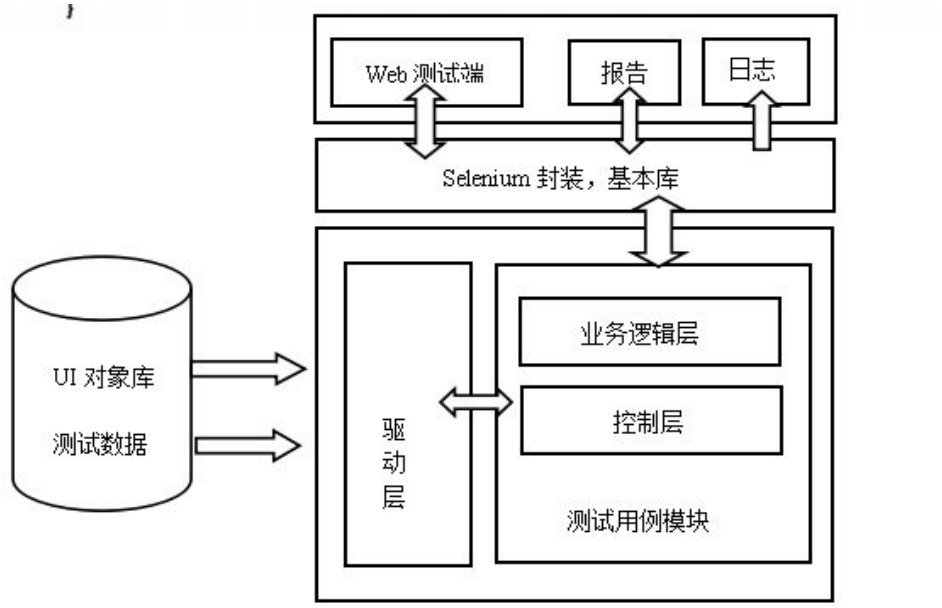

图 3.4.2（a） 框架分层结构图

驱动层是 testNG 的执行入口，包含大量的 action。测试用例模块包含了用例解析转换 testcase，同时可以调用数据仓库数据以及功能函数，然后通过 selenium 模拟用户操作对 web 系统操作，将测试结果通过报表的格式显示出来，测试的过程则记录在测试日志中。

同时可以将框架分成以下几个模块：

驱动模块，脚本组织模块，记录模块，处理模块，执行模块，控制模块，记录模块，测试用例模块，页面对象模块。

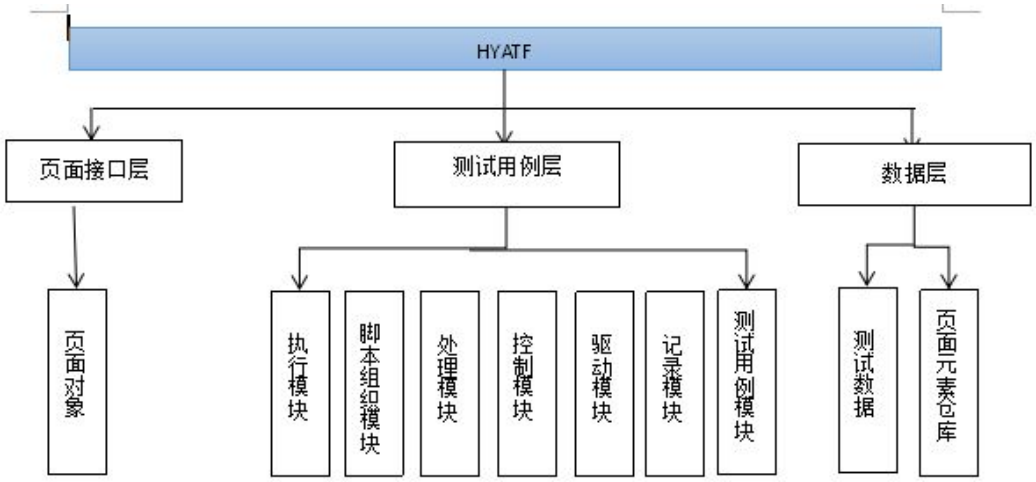

图 3.4.2（b）HYATF 模块图

3.4.3 HYATF 驱动模块

驱动模块，它的主要功能是能够实现进行业务脚本测试，也是整个框架的执行入口，由 testNG 框架驱动执行。通读全局配置，然后将需要加载的测试用例集合到测试套中，测试套，是指将需要执行的相关的测试场景相同的一系列细小测试用例的集合，通过 XML 文件加载在一起，然后调动测试数据，页面元素仓库。

3.4.4 HYATF 处理模块

Selenium 是模拟用户选择浏览器上的页面元素，然后对页面元素做点击或者输入数值等操作，然后判断是否得到预期的结果。而 selenium 在模拟用户操作的时候提供了一系列最底层的 selenium 的 API，而在使用这些 API 的时候，就会出现以下三个方面的不足：

- 函数没有做输入检查，如果直接调用很可能会因为参数不匹配而造成程序执行异常
- 函数冗余，使用起来特别麻烦，比如页面元素的获取就有 16 个方法，封装过后就不需要太多，只需要保留 find_element 以及 find_elements 就可以了，测试人员根本不需要考虑，具体的输入是什么。
- 函数稳定性差，click 等操作如果在页面加载没有完成或者等待时间过长而导致 click 函数的执行异常，而封装过后的的 click 函数增加了目标检测，异常处理，等待重试等操作，就能大大的提高程序执行的稳定性了。

而对于 selenium 的封装，我们一般封装成 function 库和 check 库。

1．funC 库

function 库是 selenium 封装模块的一个基本库。它很好的封装了 click，open,type,read，select 等操作类方法。

2．check 库

Check 库是 selenium 封装模块的另一个基本库。在 check 库中封装有 eq,isexit,等验证类。

3．自定义封装库

Functionde 库以及 assert 库的封装，也仅仅是 webdriver API 的封装，而在框架的搭建过程中以及自动化测试过程中，不仅仅满足于底层库的封装，有时候我们还需要对一些功能性的函数进行封装，形成可调用的功能库。

3.4.5 HYATF 数据模块

（1） 页面元素

最初的自动化测试，是通过录制回放，然后可以转换代码生成脚本，然后对脚本文件中的代码进行修改，形成一个个可以独立测试的小模块，然而这样的测试脚本并不能复用，也导致了各种僵化死板的代码。

Selenium 是通过模拟真实用户的操作，在浏览器上，按照测试用例的步骤通过操作对应的页面元素来测试软件。这样其中一个问题就是，如何准确，灵活的捕捉到这些元素。而这些元素的操作在 selenium 的基础库中定义特别复杂，这就要求测试开发人员需要很高的技术储备和专业素质，这样就会导致最终测试的成本大大的提高，而且最后的脚本代码也会冗余复杂，其他成员调用就需要再写一次，或者需要再次看懂学习一遍。这样会给未来的测试脚本开发带来不便，所以通过在一个项目的进行测试的开始，就可以获取所有页面上的页面元素，并将它保存在 XML 文件中。

也就是引入了对象库的概念，使得测试脚本的编写过程中将关注对象从页面元素过渡到对象库元素，这样便大大提高了测试效率。而且如果需要对页面进行功能改变，也只需要对对象库中的对象进行修改，而不需要再对脚本进行修改。

对象库的数据保存在 XML 文件中：

```
<?xml version="1.0" encoding="UTF-8"?>
<pageroot>
<page name='页面 1' ref="页面 1 url">
<element name ="用户名" disc="" type="text">
<xpath>user</xpath>
</element>
<element name ="密码" disc="" type="text">
<xpath>password</xpath>
</element>
</page>

<page name='页面 2' ref="页面 2 url">
<element name ="" disc="" type="">
<xpath></xpath>
</element>
</page>
</pageroot>
```

图 3.4.5 xml 文件示例

其中，pageroot 表示测试过程中可能涉及到的所有的页面集合。而一级节点 page 则表示一个具体的页面，其中 name 是页面的名称，ref 是只想页面的 url 地址，二节子节点 element 是页面中的一个具体的页面元素，name 表示元素的名称，type 表示元素的类别，而 xpath 则表示这个页面元素的 xpath 路径。

通过这个 XML 就能很清晰的把一个项目所有的页面元素转换成对象库对象。在测试项目之前，需要把对象全部准备好。这样在修改软件项目的时候，也只需要对应的变动修改对象库的对象，而不用再去脚本层去修改脚本。

页面元素抓取的工具，一般可以通过 selenium ide 录制，然后转化成脚本的过程中，

基本上都能将对象获取到。如果无法获取，这时候可以选用谷歌抓取插件来获得。这个的准备时间还是不多的。

（2）测试数据

基于低耦合的框架需求，我们可以将测试数据与脚本脱离开来，测试数据保存在 excel 表中。测试数据文件的命名应该和测试脚本的命名相同。

在运行的时候，调用测试脚本，自动调用数据加载模块，对预先准备好的测试数据文件进行读取并解析出数据。这样的操作，使得脚本与数据分离，使得脚本和数据均可以独立维护。

测试文件包括测试步骤名，测试数据，以及测试结果。

3.4.6 HYATF 脚本组织模块

（1）testcase 层

本层的主要作用就是根据测试用例的步骤，结合测试数据，两者结合，完成对应的功能块业务测试。本层的主要任务就是编写业务测试脚本，可以通过 testNG 中的注解调用脚本，执行脚本。Testcase 层的脚本可以调用外部文件中的数据进行操作，也可以读取上一测试页面返回结果进行测试，同时，testcase 可以使用 for 循环的方法循环多次加载测试数据。这在框架中是极其便捷的。也可以通过准备多次数据，顺换载入，参数化多次载入测试，这样可以得到多组测试结果。可以根据需要选择不同的加载方法。

（2） testsuit 层

Testsuit 层是对 testcase 层的一个补充，它主要的功能是对 testcase 的一个组合。比如某系统需要新建一个案例，然后后面需要做更改以及其他操作，这是就需要新建几个不同的数据。而循环新建数据之前的业务和之后的业务都是只需要操作一次，执行一次脚本。而前面的脚本以及后面的脚本都不会纳入循环。这时候就需要使用 testsuit 来组织 testcase 脚本了，testsuit 可以将 testcase 脚本按照不同的顺序加载集合在一起，并且按照规定的顺序运行（如下图所示）。

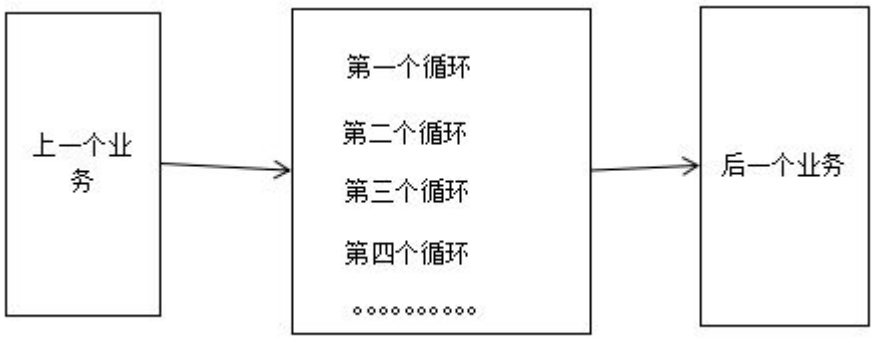

图 3.4.6 testsuit 循环表示图

3.4.7 HYATF 执行模块

本模块中将对 testcase 对象进行组织，调用，组合成一个个与功能业务相关的的功能模块脚本，通过这一层，可移植性脚本组织层关于 testcast 对象以及 testsuit 对象。

在这一层中，执行模块将调用数据，基础函数完成 web 端的操作。

3.4.8 HYATF 控制模块

HYATF 控制模块主要是对一些诸如截图，异常，监听等功能的实现，以及关于检查点的判断。

3.4.9 HYATF 测试用例模块

测试用例是决定一个测试的质量是否是高质量，并且有效的。而测试用例是与测试脚本相对应的，测试用例在框架中是可以放在一起，统一管理。

测试用例保存在 excel 文件中，每个 Excel 测试用例文件构成测试用例，为了保证这些测试用例的复用性，一般每个测试用例的长度都不会，也不允许太长，并且要相对保持稳定，这样就可以把这样的多个测试用例集合在一起，形成一个更加完善的测试集合，也就是我们前文驱动模块所介绍的测试套，然后形成一个套件文件，用.XML 表示的 testNG 文。

测试用例的结构设计一般是：testcase 名，步骤 ID，步骤名，步骤描述，操作类型，输入值，预期结果，实际结果。

Testcase 名：测试用例名，表示以及标注测试用例的信息；如登录信息

步骤 ID：对每一步操作生成唯一的 id 值。可以让驱动模块对测试用例调取是按照规定的循序来操作。

步骤名：步骤名是对每一步操作的命名，步骤名可以用日志打印出来，这样测试人员可以通过最后的日志来判断具体到哪一步出了问题。如输入用户名

步骤描述：是对步骤的补充说明，一般解释该步骤的作用等

操作类型：限定操作的类型，如文本输入，点击等

输入值：可以根据操作类型判定所要填写输入值。

预期结果：成功通过后的结果，预期结果要详细：比如系统反应是什么，结果数字是什么；用户跳转到什么页面，显示成功有什么信息，后台数据又怎样的修改结果。而不是简简单单的验证成功之类的抽象说法

实际结果：pass/fail。通过实际操作将实际结果与预期结果进行比对，如果通过，显示 pass，如果失败，则显示 fail。

3.4.10HYATF 记录模块

（1） 日志模块

没有使用 selenium 自带的日志，而是使用了 apache log4j。

因为传统的日志文件只包含简单的日志描述和截图，如果简单的测试或许日志还能胜任，但是一旦测试用例数量增加到一定数目，功能错误提示也有所相同，这样最后的日志就会出现大量的日志信息，测试人员很难从长篇的日志信息中获取到有效的信息，而且仅仅根据截图无法判断错误的具体位置，以及错误的出现情景。不能根据日志很好的定位到错误地点并推测出错误的发生原因。

所以通过 log4j 的配置，将日志分成标识为不同的日志类别，比如 debug，info，warn，error，fatal，然后就可以根据不同的配置文件将不同的信息存储到不同的位置，可以方便测试人员的查看。

（2） 测试报告模块

测试报告是用 reportNG 的工具生成的，以 HTML 或者 Excel 文件格式保存在 res 根目录下。可以通过直接打开 HTML 文件在浏览器上浏览测试的结果或者 Excel 格式直接观看。

测试报告主要是统计测试的结果的

testNG 有着其自带的的 report，虽然他的 report 还是比较全面得，但是不易阅读是他的一个不小的缺点。所以就需要用 reportNG 来替代 testNG 自带的 report。

而且 reportNG 还提供了通过一些简单的方式来查看详细的测试结果，可以对结果进行分析并着色上不同的颜色，通过模板修改定制化的内容，修改 CSS 样式替换输出样式。

（1）报告排序问题。默认报告是以字母排序的，而在测试过后，通常测试人员需要的是按照测试用例的前后顺序来排序的。

（2）希望报告中有详细描述，而且有截图，所以制定了模板文件和本地文件的方式定制化设计。

3.4.11模块依赖关系

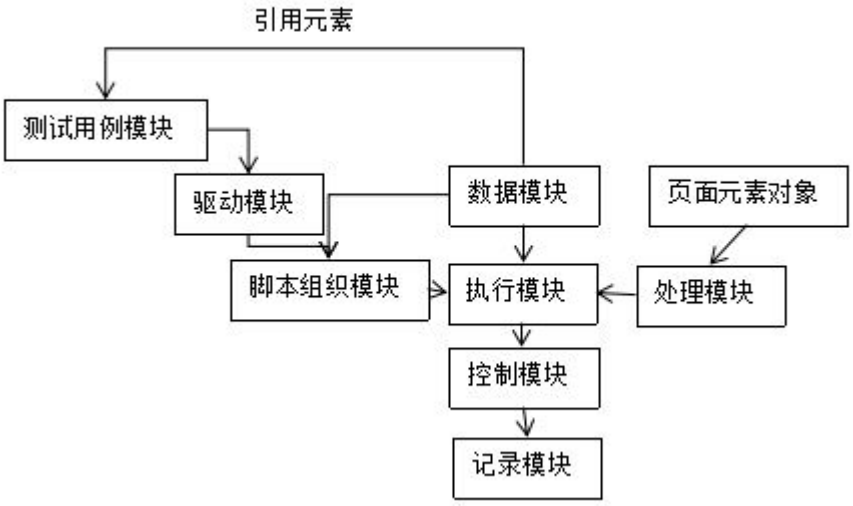

图 3.4.11 模块依赖图

（1）数据模块，是测试数据与页面元素仓库的集合，数据保存在 xlsx 文件中，页面元素保存在 xml 文件中；

（2）测试用例模块，测试用例，保存在.xlsx 文件中；

（3）驱动模块，读取 testNG 文件，读取出 excel 文件湖或者目录

（4）脚本组织模块，将 testcase.excel 表转化成 testcase 对象或者 testsuit 对象。

（5）执行模块，解析 testcase 对象，调用数据，处理函数完成对 web 端的操作。并生成报告。

（6）页面元素对象。将页面基本操作类型写成操作对象。操作对象只有 read 和 write 方法。以及常用功能组合如 login。

（7）处理模块。封装 selenium 的底层函数库，定义了常用的操作函数（包括动作类以及断言类），同时封装一些自定义函数。

（8）控制模块。截图，监听。

（9）记录模块。生成报表以及日志。

3.4.12主要结构设计

在 HYATF 框架中，最为重要的便是驱动模块以及处理模块中的两个构件。分别是执行入口 testexecutor()以及 selenium 封装类。驱动模块的构件是负责解析 testXML 文件的，而 selenium 封装的是 selenium 基本库的函数，是对浏览器的交互操作。

首先驱动模块构件会读取 test.xml 文件，判断是多个用例还是单个用例，并调用元素类型，以及数据。

而封装模块则是根据 selenium 的基本函数以及页面元素类型对浏览器进行操作。

最后返回结果，生成测试报告，由 reportNG 负责对测试报告进行组装(如下图所示)。

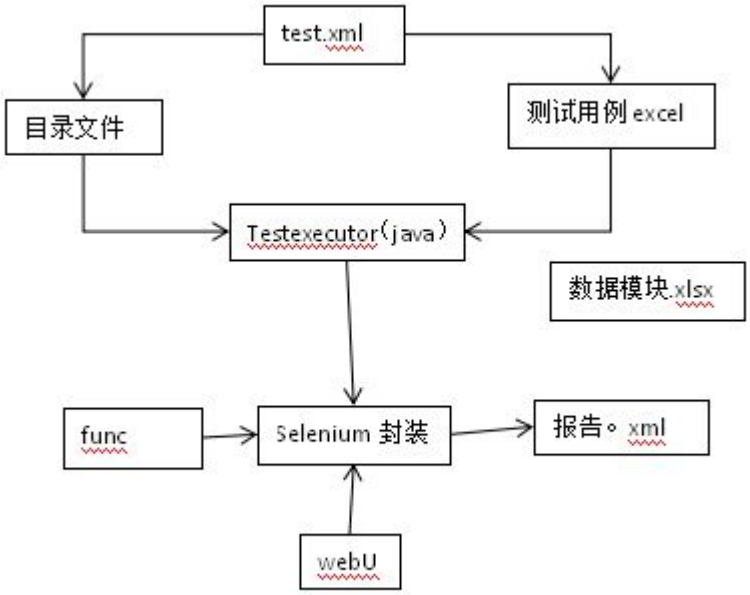

图 3.4.12 HYATF 框架结构图

3.4.13 拓展功能

（1）自定义功能拓展

可以自定义页面元素类型，页面对象及其实现方法。
定义页面对象只需继承抽象类，重写 read 以及 write 方法。
定义页面元素，使用 fn 关键字。

（2）插件拓展

（1）CI 工具集成

使用 CI 工具如 Jenkins 自动执行测试用例，首先需要将项目上传到 svn 或者 git 服务器，然后在源码管理处选择 Subversion 进行配置，为了能在 Jenkins 中直接查看测试报告，需要在构建后操作增加 Publish HTML reports 操作，具体配置详见下图：

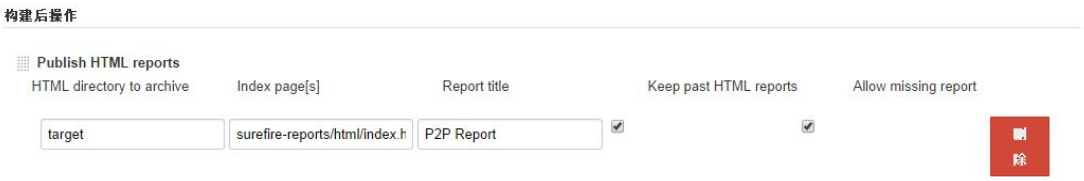

图 3.4.13（a）jenkins 配置

（2）Excel 插件

该插件可以让测试用例编写起来更加轻松、快速，且稳定性高。界面如下图所示：

图 3.4.13（a）excel 插件界面

4 HYATF 框架的实现

4.1 环境搭建

4.1.1 开发环境介绍

HYATF 框架的的搭建主要是采用了 java 语言开发，java 语言是面向对象的编程语言，具有良好的可复用性和可移植性。被广大的开发人员所喜爱和钟情，并应用于各种互联网系统以及移动 APP 的开发过程中。

HYATF 框架所开发时运行的系统是基于 windows7，Windows&是由微软公司在 2009 年推出的一款操作系统，具有非常良好的可视化的操作界面，而且系统的兼容性也特别的强强。

HYATF 框架工程是使用的 Maven 项目，Maven 是由 apache 公司退出了一个颇为成功的开源项目。Maven 是跨平台的一个关于项目管理的平台。主要服务于项目构建，依赖管理和项目信息管理。

HYATF 框架的开发工具我选用的是 eclipse，因为 eclipse 是一款功能强大的可以使用 java 来进行开发的集成开发环境。

4.1.2 java 开发环境搭建

首先下载 1.6 以上版本的 jdk（推荐下载 1.7 版本），然后安装 jdk。安装完成后进行环境变量配置，分别添加 JAVA_HOME，CLASSPATH 和 Path 变量，最后再 cmd 命令行输入 javac-version ，如果能显示版本信息则表示安装成功。

4.1.3 Maven 安装

首先，在官网上下载最新版本 maven。然后解压到指定目录下。

然后设置环境变量 M2_HOME 以及 Path 的值。打开 cmd 输入 mvn -v 命令，如果可以显示 maven 的版本信息则安装配置成功。

随后打开 eclipse，可以选用离线或者在线加载 maven 组件。打开窗口属性并设置 maven 本地库地址。

4.1.4 selenium 安装

Selenium 的安装比较简单，只需要下载 selenium 所需的安装包，然后直接建立的 Maven 项目中通过 build path 添加 jar 包就可以了。当然如果需要版本变动，可以直接通过 maven 进行配置，自动下载。

4.1.5 testNG 安装

在官网上下载插件。，然后放在 eclipse 的 plugins 文件夹下，启动 eclipse，然后点

击帮助->更新软件->安装软件，找到 testNGeclipse 插件，点击安装，重启 eclipse。最后打开窗口—>显示视图->其他，在 java 文件夹下，有 testNG 图标，双击便能看到 testNG 窗体了。testNG 到此安装完毕。（不推荐在线安装，下载速度很慢。）

4.2 项目创建

创建 maven 工程，并配置 pom.xml 文件，可参考以下范例。

```
    <project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
      <modelVersion>4.0.0</modelVersion>
      <groupId>com.huayao.atf</groupId>
      <artifactId>HYATF</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <description>Asset Information Management Platform</description>
    </project>
```

4.3 目录说明

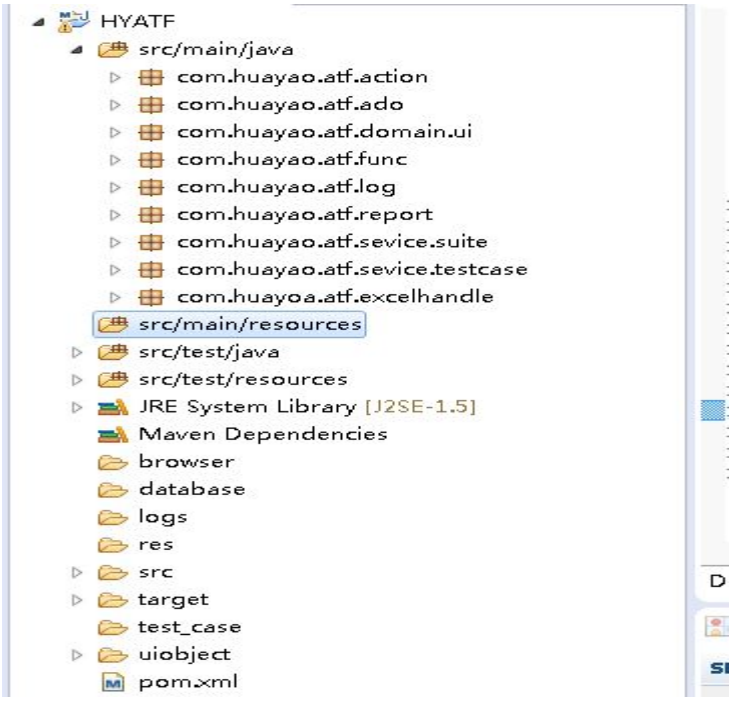

图 4.3 目录介绍

Com.huayao.atf.action 目录必须存在，框架会在该目录路径下寻找测试执行的入口。
Com.huayao.atf.ado 目录必须存在，外部测试数据加载包
Com.huayao.atf.domain.ui 目录必须存在，框架会寻找定义在本地的页面元素类型。
Com.huayoa.atf.func 该目录是日志模块
Com.huayao.atf.report 该目录下是报告模块
Com.huayao.atf.sevice.Testcase 业务层下 testcase 层
Com.huayao.atf.sevice.suite 业务层下 testsuit 层
其他目录不做要求，可任意命名。

Browser 放置浏览器信息配置。
Database 存放测试数据文件。
Logs 存放日志文件。
Uiobject 存放页面元素对象。

4.4 测试用例执行入口

创建 com.huayao.atf.action.DefaultTestEntry 类，此动作必须执行：

```
package com.huayao.atf.action.action;
public class DefaultTestEntry extends TestExecutor{}
```

4.5 基本页面元素类型

测试框架需要通过元素类型来判断如何操作页面对象，例如，如果页面元素类型是Button，框架就会调用相应的点击方法；如果是 Text，框架就会调用输入方法等等。

基本页面元素说明

概念	解释及用途
DefaultAlert	Write：根据传入参数，点击提示框的确定或取消按钮 Read：Abstracted
DefaultButton	Write：点击按钮 Read：获取按钮上的文本
DefaultCheckBox	Write：根据传入参数，确定单选框的勾选或者取消 Read：返回单选框是否选中状态
DefaultComboBox	Write：选择下拉列表项 Read：读取当前选择项
DefaultInput	Write：调用 javascript 在文本框中输入内容，延时 50 毫秒 Read：读取文本框中的文字内容
DefaultLabel	Write：不支持，报错 Read：读取 Label 对象的文本内容
DefaultRadio	不提供实现
DefaultTable	不提供实现
DefaultText	Write：调用 selenium 原生方法输入文本框 Read：读取文本框中的文字内容

4.6 编写测试用例

测试用例以 Excel 文件进行管理，框架会严格按照测试用例文件中定义的步骤执行测试，并记录结果。

4.6.1 测试用例目录结构

所有的测试用例文件统一存放在项目的 TestCase 目录下，该目录下可以在任意目录、层级下保存测试用例文件，在引用测试文件时，只需在测试集配置文件（配置详见 3.5 章节）中指定相对于 TestCase 的相对路径即可。

图 4.6.1 用例目录

4.6.2 测试用例文档结构

每一个测试用例文件格式都是一样的，包含多行，每一行根据内容不同，可以是一些注释信息，也可以是一条测试用例，控制测试框架执行一个动作。每行包含 9 列，每一列有不同的含义，详见下表。

测试用例文档结构说明

编号	列名称	解释及用途
1	Test Case ID	测试用例编号，每个文件中不唯一，一般用于标示包含多个测试步骤的组合用例，配合 Test Case Name 列的内容进行描述说明
2	Test Case Name	测试用例名称，与 Test Case ID 匹配出现，并会显示在测试报告中
3	Step ID【重要】	测试步骤的顺序编号 【注】：如果 Step ID 为空，则框架会自动跳过该行，因此如果不希望执行该行动作或者作为注释信息时，该行为空，其余情况必须填写
4	Step Name	测试步骤名称，与 Step ID 匹配使用，最终会体现在测试报告中
5	Action	测试关键字，框架将根据此关键字执行相应的测试动作
6	Element	操作元素对象，取值范围为元素仓库，与页面元素对象的 FullName 对应，测试框架将根据此列的值，在元素仓库中寻找相应的页面对象，再根据其 Locator 定位描述属性，在页面上寻找对应的元素进行操作
7	Value	一般与特定关键字配合使用，如与 type 关键

编号	列名称	解释及用途
		字配合，实现在文本框中输入内容，内容即为 Value 列的内容
8	Expect Type	测试检查点类型
9	Expect Value	测试检查点内容

测试样本截图：

testcase name	step id	step name	description	action	element	value	ExpectType	Expectvalue
登陆	1			assert				
	2	点击用户名		click	登陆，用户名文本框		text	用户名文本框
	3	输入用户名		type	登陆，用户名文本框	hefei		
	4	点击密码		click	登陆，密码文本框		text	密码文本框
	5	输入密码		type	登陆，密码文本框	123456		
	6	点击按钮		click	登陆，登录按钮			
	7			wait				

图 4.6.2 测试样本

4.7 测试用例配置

测试用例是<suite>套件的必须组成部分，每个<test>标签中可定义一个或多个将要被执行的测试用例 Excel 文件。只有 test-case 参数是必须的，测试框架需要以此来找到对应的测试用例。

1、执行单个测试用例文件，实现登陆测试

```
<test name="登录">
        <parameter name="test-case" value="登录/登录.xlsx"></parameter>
        <parameter name="screen-shot-switch" value="onFail"></parameter>

        <packages>
            <package name="com.huayao.atf.action"></package>
        </packages>
    </test>
```

图 4.7a 单例核心代码示例

2.指定目录下的多个测试用例文件，实现查看计划测试

```
<test name="查看计划">
        <parameter name="useflowauto" value="true"></parameter>
        <parameter name="test-case" value="查看计划"></parameter>
        <parameter name="screen-shot-switch" value="onFail"></parameter>
        <packages>
            <package name="com.huayao.atf.action"></package>
        </packages>
    </test>
```

图 4.7a 多文件核心代码示例

4.8 运行测试

4. 8. 1 单套件测试

通过 TestNG 调用，执行某一个测试套件的配置文件。

4.8.2 多套件测试

通过在 Maven 配置文件中集成测试套件，通过 maven-surefire-plugin 插件执行多个套件的测试。因此需要在整个项目的 pom.xml 配置文件中，增加以下配置段落，将需要运行的测试套件配置文件都添加到 suiteXmlFiles 部分。然后通过 Maven 命令配合不同的 Goal 即可调用执行测试，常用的执行方式包括 mvn test、mvn clean test。

```
<project>
 ..................
 <dependencies>
      ........................
 </dependencies>
 <build>
      <defaultGoal>test</defaultGoal>
      <plugins>
          <plugin>
              <groupId>org.apache.maven.plugins</groupId>
              <artifactId>maven-surefire-plugin</artifactId>
```

```
                <version>2.6</version>
                <configuration>
                    <!-- <argLine>-Dfile.encoding=UTF-8</argLine> -->
                    <properties>
                        .....................
                    </properties>
                    <workingDirectory>target/</workingDirectory>
                    <suiteXmlFiles>
                        <suiteXmlFile>res/异常操作.xml</suiteXmlFile>
                        <suiteXmlFile>res/登录功能检查.xml</suiteXmlFile>
                        <suiteXmlFile>res/静态页面检查.xml</suiteXmlFile>
                        <suiteXmlFile>res/计划查询检查.xml</suiteXmlFile>
                        <suiteXmlFile>res/我的计划检查.xml</suiteXmlFile>
                    </suiteXmlFiles>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

5 HYATF 框架的应用与结果分析

当测试框架搭建成功，笔者根据测试网站的登录模块，计划查看模块分别编写了对应的测试用例以及准备了测试数据。然后根据手工测试 126 例对这两个模块部分功能进行了测试。共计登录模块 12 例，计划模块 96 例。

然后分别通过手工测试，录制测试以及框架集成测试。

5.1 测试框架应用示例

框架初步搭建，目前只是对两个模块的页面元素以及数据，用例进行设计，然后在使用的过程中，必须按照框架的流程一步步操作，可以通过 testNG 单独运行一个模块的 suit，也可以通过在 maven 的配置文件中的配置，然后通过 maven 命令直接运行整个测试。

登陆测试模块示例：

（1）添加页面元素

由于登陆界面的元素有 3 个，分别是【登陆，用户名文本框】文本框，【登陆，密码文本框】文本框和【登陆，登录按钮】按钮。所以可以添加一个 XML 文件，将三个元素添加到配置中。

【登陆，用户名文本框】

```
<page name='login' ref="http://192.168.88.77:8080/login">
<element name ="登陆，用户名文本框" disc="" type="text">
<xpath>[@id=’username’]</xpath>
</element>
```

【登陆，密码文本框】

```
<element name ="登陆，密码文本框" disc="" type="text">
<xpath>[@id=’password’]</xpath>
</element>
```

【登陆，登录按钮】

```
<element name ="登陆，按钮" disc="" type="button">
<xpath>[@id=’loginbutton’]</xpath>
</element>
```

（2）

这是登录模块基于 firefox 正常登陆的测试用例：

testcase name	step id	step name	description	action	element	value	ExpectType	Expectvalue
登陆	1			assert				
	2	点击用户名		click	登陆，用户名文本框		text	用户名文本框
	3	输入用户名		type	登陆，用户名文本框	hefei		
	4	点击密码		click	登陆，密码文本框		text	密码文本框
	5	输入密码		type	登陆，密码文本框	123456		
	6	点击按钮		click	登陆，登录按钮			
	7			wait				

图 5.1 测试用例-登陆示例

(3)执行

步骤 1：使用 assert 关键字，判断是否在登陆页面

步骤 2：寻找到用户名文本框对象，使用 click 关键字，做点击用户名文本框操作，增加检验是否找到 text 对象

步骤 3：使用 type 关键字，在用户名文本框中输入用户名

步骤 4：寻找到密码文本框对象，使用 click 关键字，点击密码文本框，增加检验是否找到 text 对象

步骤 5：使用 type 关键字，在密码文本框中输入密码

步骤 6：寻找到登录按钮对象，执行关键字 click

步骤 7：使用关键字 wait，等到新页面加载。

（4）检查

设置检查点。判断是否登陆成功，成功填入日志，报告，失败启动截图函数。

5.2 测试框架结果分析

手工测试：

测试模块	用例数	可执行	准备时间	执行耗时	覆盖比率
登录模块	18	18	180min	15min	100%
计划模块	108	108	1200min	400min	100%
合计	126	126	1380min	415min	100%

录制-自动化测试：

测试模块	用例数	可执行	准备时间	执行耗时	覆盖比率
登录模块	18	18	480min	30min	100%
计划模块	108	87	3000min	240min	80.5%
合计	126	105	3480min	270min	84.1%

自动化框架测试数据：

测试模块	用 例	可 执	准备时间	耗时	覆 盖 比

	数	行			率
登录模块	18	18	16h	4min	100%
计划模块	108	96	100h	27min	88%
合计	126	114	116h	31min	90%

框架构建（雏形）：2 个月

注：用例数表示手工测试用例数，可执行表示相对可执行，能写出的测试用例，并不是所有的测试用例都能写出自动化测试用例

通过上面的数据分析：

手工测试一次需要准备 1380min,人工执行 415min；

录制测试一次需要准备 3480min，人工执行 270min；

框架测试一次需要准备 116h,人工执行 31min；

手工测试 X 次需要 Y=1380+415*X

录制测试 X 次需要 Y=3480+270*X

框架测试 X 次需要 Y=2*30*8*60+116*60+31*X

根据几个线性函数的分析如果没有框架或者仅仅是通过录制，还不如不适用自动化测试，因为自动化测试的成本反而比手工测试要高的多。而且灵活性不强，

一次自动化测试就相当于一次手工的操作而已。

而通过使用自动化测试框架，对比与手工测试。如果进行足够多的次数，手工测试和框架测试最终的花费是可以相等的。因为这里数据量比较小，所以需要 95.5 次。如果是一个项目的话有着大量的数据需要测试，这个时候，框架搭建的时间花费就会越来越显得低廉。

所以 HYATF 框架在成本上来说还是可行的。并且可以运用在实际项目过程中。

6 总结与展望

基于公司软件产品主要是用于生产，所以更多的是像点击，输入等操作类型，而样式之类的设计很少，所以构建一个比较简单，功能不需要太强大，但是可以将大量输入点击操作转换为机器测试的一款自动化测试框架，能很好地解决公司测试环节所出现的问题，并使之能应用与真正的测试环节。

6.1 工作总结

本文通过查阅如今现有的自动化测试技术发展的资料，了解并学习现今软件自动化测试框架的技术，通过分析公司华耀分部生产项目的功能需求，以低成本为目标，以解决操作过程中大量重复点击输入输出等操作的测试问题，以缩短维护阶段测试时间的需求，设计了一款基于华耀分部生产项目的自动化测试框架并完成搭建。使得框架能解决目前测试过程中所带来的巨大时间耽误以及人为失误。而在、自动化测试框架的设计与搭建过程中，本人所做的工作如下：

1) 通过了解学习目前主流自动化测试技术，了解，结合对公司现有情况下的生产系统做功能需求分析，并根据分析选取适用于开发框架的几框工具，并深入学习：了解 maven 管理工具以及 maven 的配置学习，使用的 JAVA 语言开发,使用 eclipse 开发工具，了解 selenium2.0 的工作原理以及常用 API 方法，以及 testNG 的注解应用和常用注解。
2) 对现行主流的框架分层架构做详细了解，并分析设计，分析公司所需最基本功能的分层，将其分为业务层，数据层，用例层。并针对三层架构详细设计了驱动模块，selenium 封装模块，业务管理模块，数据模块，测试用例模块以及记录模块。通过将业务逻辑，测试数据与脚本的分离。提高自动化测试框架脚本的复用性以及可维护性。
3) 根据设计搭建项目，并部分封装 selenium 的底层库，两个模块的页面元素类型，准备测试数据以及设计测试用例，并能调用 testNG，部分实现项目功能，并得出测试结果
4) 根据准备的测试数据以及测试用例调用测试框架，然后记录结果，并将其与前期手工测试以及录制测试进行比较，根据测试结果建模并分析计算，验证框架相比如手工测试，录制测试的优异点，证明框架在应用到实际生产中，是可以将人力物力资源解放出来，降低成本。证明本课题成功解决了公司实际项目生产过程中对系统应用程序测试过程中所表露出来的问题。能够解决手工测试延误生产的及时性的问题。

本次论文的重点不在于理论创新，不在于作出功能多么复杂的项目。而是根据学校所学知识并结合自主网上学习，查阅，以及在公司师傅与学校导师的指导下，能将实际公司工作过程中发现的问题解决了，对问题提出正确的解决方案。通过设计自动化框架来提高测试执行时间，降低成本。为公司的收益作出自己的那一份贡献。通过设计并实现这个框架提升自己的能力，展示自己的专业素养，创造出有价值，有意义的成果。这也是本次课题研究的目的以及初衷。

6.2 展望

本文所设计的框架已能够实现数据与脚本，业务逻辑的分离，已经实现通过 testsuit 层集合执行测试用例的功能，已经实现打印测试结果报表以及根据日志记录测试过程的功能。已经通过对 selenium 的底层库进行封转和简化。初步实现了框架的基本功能，能够简单的解决目前实际公司的测试要求并投入使用。但是由于时间仓促，以及自身掌握知识的欠缺。框架还是比较死板。对框架的使用也要有一定的技术要求。但是后续的时间，后续的工作，还是可以对框架进行补充。

1) 没有用户界面。没有创建一个可供管理测试项目，测试数据，测试报告，测试日志以及页面元素类型管理的管理平台。甚至没有一个可交互的 excel 表用来操作管理。在今后的工作中，会添加界面的设计，可以通过界面浏览元素对象，测试用例，测试集，封装 API 等，提供直观的操作流程，让测试人员更加舒心的使用。
2) 服务对象单一，封装库资源单一，只能符合本公司华耀生产的系统，只能对拥有大量点击，输入输出，样式简单的项目进行构建测试执行，后期会对 selenium 库里其他函数继续封装，完善，本次设计时间有限，只能部分封装，把主要操作的函数封装了。当然后面会对框架进行拓展以及健壮。
3) 测试框架拓展，框架目前只能简单集成测试，但是对于测试过程中出现的问题不能很好的定位。只能验证这次功能修改没有影响其他功能模块导致错误并能投入生产，而不是去发现修改的功能部分的错误。当然后面会提供一系列对日志的监控输出，以及对测试报表的分析，发送。
4) 要求测试人员技术比较高高。在脚本编写的前中需要了解框架的机制，要按照框架的功能才能够设计脚本并导入数据进行执行。后期会更加简单的封装一些自定义库方法。让测试人员能够通过界面平台搜索，添加，删除页面对象，测试脚本与数据。使得测试人员不需要掌握相关技术也能通过简单平台说明使用自然语句编写脚本。测试软件系统。

以上 4 点是由于该次雏形框架搭建过程中遗留下来的问题，将在后期的研究过程中进行补充修改。使得 HYATF 框架向更加健壮，更加成熟的方向发展，能很好地推广到公司其他分部其他项目以及其他产品，而不是单单解决生产部门软件系统的重复操作的自动化测试。让本框架在未来走的更高，更远。

参考文献

[1] 唐雨薇.分布式自动化测试框架研究.科技信息.2011 6-6

[2] 赵森.基于关键字的自动化软件测试框架设计.中国高新技术企业.2014 10-11

[3] thomas geiz.自动化改进生产流程.流程工业.2014 6-6

[4] 张德申.web 自动化测试工具 selenium.电子技术与软件工程.2013 11-13

[5] 姚杰.基于数据驱动自动化测试框架的研究与实现.工业控制计算机.2013 19-19

[6] 边耐政. 张琳. 一种基于 selenium 的 web 自动化测试低耦合框架. 计算机应用与软件. 2009. 12-29 1717

[7] 黄侨. 葛世伦. 开源 web 自动化测试框架的改进研究. 科学与技术工. 2012. 12-29

[8] 王莉殷锋. 软件自动化测试脚本设计研究 . 西南名族大学学报（自然科学版）. 2003. 12-18

[9] 白洛. 基于 selenium2 的自动化测试——从入门到精通. 机械工业出版社. 2014. 12-15

[10] tarun lalwani. 自动化测试框架参考准则. 2009. 16-20

[11] 张慧琳.李威.基于 selenium 和 testNG 的集成自动化测试平台设计.实验技术与管理.2015.21-23

[12] 赵静文.基于 selenium 的 web 自动化测试系统的研究与实现.东南大学.2014.2-23

[13] 史济民.顾春华.郑红.软件工程-原理，方法与应用.高等教育出版社.2009.2-29

[14] 常征.功能测试中自动化测试框架的分析与应用.北京林业大学.2007.2-29

[15] 吴莹 基于 selenium 的 web 自动化测试框架.科技传播.2011.2-29

[16] Rishab Jain,R Kaluri.《Design of automation scripts execution application for selenium webdriver and testng framework》. researchgate. 2015

[17] S Gojare R Joshi D Gaigaware.Analysis and Design of Selenium WebDriver Automation Testing Framework.Procedia Computer Science.2015

[18] P Bindal，S Gupta.Test Automation Selenium WebDriver using TestNG.Applied Sciences.2014

[19] 赵卓.Selenium 自动化测试指南.人民邮电出版社.2015

[20] peng gong. 自动化测试框架设计指南. TAS Team. 2013

[21]翰华自动化测试框架设计说明书. 2016

致　　谢

随着五月的到来，意味着我大学四年学习生涯马上就要结束，也意味着我马上就要毕业了，而我的毕业论文也在此刻接近了尾声。我借此机会真诚的感谢一路上支持我的人，因为有了你们，我求学的道路上，课题研究的成果里，都有着你们深深的情谊。

首先，我必须要感谢一下我的导师 XXX 教授，她宽阔的业界，思维的深度，以及学术理论的深厚以及对莘莘学子发自内心的呵护，给我毕业设计提供了帮助，帮我树立了明确的研究方向，树立了明确的科研目标。

其次，我要感谢我公司的指导师傅 XXX 主任，在我的毕业设计期间，给我了技术上的支持以及鼓励，在毕设过程中也给与了我的理解与宽容。

同时，我还要感谢来源于全国各地的好友，是你们不断地为我讲解，推荐，指导，给与我在毕设相关技术上的实现。

最后，我要感谢一下我的父母以及家人，在我实习期间以及毕设期间，给我提供的呵护与支持，感谢你们不辞劳苦的陪伴与鼓励。

附　录

附录 A 外文文献原文与翻译

原文：

（节选自 Automated Testing of Java Web Applications）

The frontend of Java web applications is usually implemented by using Java servlets. There is a standard for Java servlets and this means that a servlet that follows this standard works on every web server that conforms to this standard. It is quite easy to write a servlet. The only thing that we should do for following the standard is to implement the interface javax.servlet.Servlet. And even easier way is to extend the javax.servlet.http.HttpServlet which already provides some functionality and default implementations for all the methods needed. Besides extending HttpServlet we should override one of its methods for adding our own behavior. If the servlet is deployed to the web server then the request sent to the web server executes this method and provides the query as a parameter. The web server also provides an output stream for writing the response for the request. Our servlet can then interpret the request and write an appropriate response to it. Using servlets in this way is inconvenient on most cases. This approach works only with the very trivial applications because the API is too low level. Most web applications use some framework that simplifies the development. Usually such framework provides its own implementation for the Servlet interface and this implementation intercepts all the requests to the web application. In this case we don’t write the implementation for the Servlet interface but extend some framework specific class. In essence it is quite the same thing but this way it is possible to use frameworks functionality. This functionality usually consists of some sort of template system, easier request dispatch, automatic form object population from the Http request and more.

6.3 Testing Framework

6.3.1 Functionality of the Framework

Testing Framework allows executing test code without deploying the application 28 The framework enables instantiating Spring controllers so that they have all necessary dependencies. This way it is possible to just call controller's methods outside web server. This makes running individual tests much faster and makes Test Driven Development (TDD) more feasible. It also provides developers with the possibility to get debugging information much quicker. The framework provides also special methods for querying the intermediated state of the application under test. Also, just inserting "print" or "log" statements into AUT code works much better this way.

6.3.2 Special marker elements in Html code

To alleviate the problem of fragile tests the framework prescribes that developers use special marker elements in the Html that the AUT produces as output. If the developer follows this advice then the framework can take advantage of these special markers and produce tests that do not depend so much on the structure of the Html document. These special markers add additional structure to the Html pages. For example, let's say that there is a page with n tables and we would like to evaluate that one of them contains m links. If the page does not contain any additional markers we would have to rely on the order of the tables and to the fact that they are tables (rendered with the Html "table" tag). But if we surround the links we are looking for with additional tag, which has certain identifier, then it is possible to find those links without relying on the structure of the Html document. In this case the layout could change radically but if we still surround the links with this additional tag, then the tests don't break. These special markers could be rendered by macros which would allow turning of the rendering of the markers in a production environment where it is not needed.

6.3.3 Abstractions

The problem with a general purpose language like XPath is that it is meant for querying all possible XML documents. But the elements that we search for from the web applications' output are quite specific. It is logical that a general purpose tool is suboptimal for

the very
specific tasks. What we usually would like to check from the web applications output is that it
contains form fields with certain values, it contains certain messages or that certain links are
valid. But if we are interested in only these things it is possible to abstract away the structure
of the Html almost completely and use evaluation methods that are much more human
friendly. For example instead of using XPath query //input[@name='amount'] to find an
element and then evaluation the elements value attribute we could have a method like
fieldValueIs(“amount”, 7) that would accomplice the same thing. This way the tests are
much easier to write and are readable as well. 29

6.3.4 Usage

The intended usage of TF is very similar to the usage of HtmlUnit. Short example of using
HtmlUnit is presented in Listing 6-1.

```
01. @Test
02. public void testTitle() throws Exception {
03.     WebClient webClient = new WebClient();
04.     HtmlPage page = webClient.getPage("http://localhost/test");
05.     assertEquals("HtmlUnit ", page.getTitleText());
06. }
```

Listing 6-1

- Line 3 creates a browser object.
- Line 4 requests a Url from web server and saves the result.
- Line 5 evaluates that the resulting page has a title “HtmlUnit”.

In both cases test code is organized as JUnit test cases but TF uses ControllerTester instead of
WebClient and Result instead of HtmlPage. Both frameworks provide interface for executing
application functionality and enable inspection of the result through special object. The
difference is that TF executes application code outside of web container and thus does not
require that the application is deployed before the execution of test code. Also TF-s result
object provides much more feedback for evaluation than HtmlUnit’s HtmlPage. Listing 6-2
provides simple example of TF usage. A simple tutorial of the TF is presented in chapter

12

```
01. @Test
02. public void presentForm() {
03.    ControllerTester ct = new ControllerTester("sample1.xml");
04.    Result r = ct.execute("/login.html");
05.    assertThat(r.getViewName(), is("loginForm"));
06.    r.div("loginForm").contains("login.label.username");
07.    assertThat(r.getFormFieldValue("username"), is("john"));
08.    assertThat(r.getHtmlFieldValue("username"), is("john"));
09.    assertThat(r.getModelField("accounts['2']"), is("Account 2"));
10. }
```

Listing 6-2

• Line 3 creates ControllerTester object. “sample1.xml” is Spring configuration file that
contains applications configuration. All spring managed objects are described in this file.

• Line 4 executes the code of a controller that corresponds to the identifier “/login.html”
in Spring configuration.

• Line 5 checks that the controller code selects the right view to be rendered.

• Line 6 checks that a certain part of the output Html contains a certain label.

• Line 7 checks that the form bean’s field “username“ has a value “john”.

• Line 8 checks that the Html contains form field named “username“ and this has a value “john”.

• Line 9 makes an evaluation about model objects structure.

6.3.5 Architecture and Design

The central classes of the framework are ControllerTester and Result. Those classes bind
together different classes from Spring framework and HtmlUnit and add some custom
functionality. Together these classes make up the interface for the TF. ControllerTester
enables to compose and execute a query against a web application and Result provides
evaluation methods for inspecting the outcome.

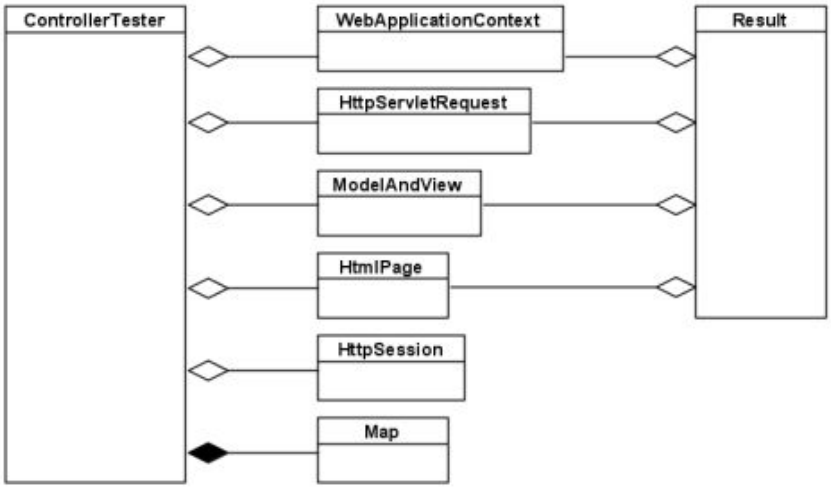

Figure 6-6

WebApplicationContext is a Spring class. This is the main point through which the Spring framework is accessed. When we need a Spring managed object we ask the WebApplicationContext to create it for us. WebApplicationContext consults its 31 configuration and then creates all the dependencies, creates the object requested and wires the dependencies to object requested. TF uses the WebApplicationContext to create controller objects. WebApplicationContext instantiates a controller class and it populates it with data access object, form bean object and other objects.

HttpServletRequest is actually populated with MockHttpServletRequest object which comes with spring distribution. This object is sent to controller as a parameter when the controller code is called. Also, this object can be inspected after the controller code has been executed. Among other things it is possible to get HttpSession object through HttpServletRequest.

ModelAndView is also Spring specific object. When controller code is called it returns ModelAndView object. This object contains information about which view to select and also all the dynamic data that controller wants view to render.

HtmlPage is HtmlUnit specific class. This class is not used in the execution of the test. Its purpose is to simplify html parsing code. HtmlPage is an abstraction of an Html document and it provides many methods for querying the document that it represents. Among others it

provides the possibility to use XPath to fetch elements from Html document.

HttpSession is created when it is when the client code calls ControllerTester objects startSession() method. After this call ControllerTester object creates Session object and
populates every HttpServletRequest that it sends with this session object. This enables testing functionality which depends on the possibility to store data in session. This could be
used for example to test multipage entry forms where all the pages except the last collect data
into the session and all the data is submitted to the permanent store when the whole multipage
form is finally submitted.

Map is ordinary Java Map interface implementation. This is used for collecting the parameters
that must be passed to controller. As ControllerTester takes care of the creation of HttpServletRequest object it must also populate this object with the right parameters. So,
ControllerTester accumulates the parameters for some query into this map and populates HttpServletRequest with them if there is a time to call the controller code which requires
HttpServletRequest as a parameter.

ControllerTester.execute() returns a Result object which provides access to request, session and model objects as well as html output. Most objects described here are shared between ControllerTester and Request as seen from the class diagram in the Figure 6-6. For
example HttpServletRequest object is shared between ControllerTester and the Result. 32
This is because ControllerTester creates the HttpServletRequest and passes this to controller code. Controller code could add some attributes to HttpServletRequest object (HttpSession is also accessible through HttpServletRequest object) and after the controller
code is finished we would like to see that the controller actually did the required modifications. So, it is possible to inspect the HttpServletRequest object through the Result
object. Result provides several evaluation methods which simplify the inspection of Html
output and the intermediate state of the execution.

译文：

java web 应用的前端通常是用 java servlet 实现。有 一个标准的 java servlet，这意味着一个 servlet，遵循这一标准的作品 符合本标准的每一个网络服务器。是很容易写一个 servlet。唯一的 我们应该做以下的标准是实现该接口的实例。 servlet.servlet。甚至更简单的方法是将 javax.servlet.http.httpservlet 这已经为所有的方法提供了一些功能和默认实现 需要。除了延长一个我们应该重写它的方法加入我们 自己的行为。如果 servlet 部署到 Web 服务器并发送到 Web 请求 服务器执行此方法，并提供查询为参数。网络服务器也 提供用于编写请求的响应的输出流。我们的 servlet 就可以 解释该请求并写下适当的回应。使用 servlet 这样 多数情况下不方便。这种方法只适用与非常琐碎的应用程序 因为原料药太低了。大多数网络应用程序使用一些简化的框架发展。通常这样的框架为 Servlet 提供自己的实现 接口和实现拦截所有的请求到 Web 应用程序。在这 如果我们不写实现 Servlet 接口可扩展框架 具体的类。从本质上来说，它是一个相当的东西，但这种方式，它是可以使用 框架的功能。此功能通常包括一些模板系统， 简单的请求调度，从 HTTP 请求并自动形成目标人群。

6.3 测试框架

6.3.1 功能框架

测试框架允许在不部署应用程序的 28 个应用程序中执行测试代码这个框架允许实例化弹簧控制器使他们拥有所有必要的依赖。这样就可以调用控制器的方法在网络服务器之外。这使运行单个测试速度快、测试驱动开发（TDD）更使可行。它还为开发人员提供了获取调试信息的可能性快。该框架提供了查询的中间状态的特殊方法测试中的应用。另外，只要将“打印”或“日志”语句自动编码这样的作品多好。

6.3.2 特殊标记元素中的 HTML 代码

为了缓解脆弱的测试框架规定，开发商利用特殊问题在 AUT 输出 HTML 标记元素。如果开发者如下建议的框架可以利用这些特殊的标记和生产测试不要太依赖于 HTML 文档的结构。这些特殊标记添加附加结构的 HTML 页面。例如，让我们说有一页有氮表我们想评估其中一个包含我的链接。如果页面不包含任何额外的标记，我们将不得不依赖于表的顺序和事实，即他们是表（渲染 HTML“表”标签）。但如果我们围绕着我们的链接寻找具有特定标识符的附加标记，然后可以找到这些标记不依赖于 HTML 文档的链接结构。在这种情况下，布局可以改变从根本上，但如果我们仍然围绕着这个附加标签的链接，然后测试不打破。这些特殊的标记可以由宏来呈现，这将允许在一个不需要的生产环境中的标记的绘制。

6.3.3 抽象

一个通用的语言，如 XPath 的问题在于它是用来查询所有可能的 XML 文件。但是，我们从网络应用程序搜索的元素输出是非常具体的。这是逻辑的一般用途的工具是不理想的非常具体的任务。我们通常想从网络应用程序输出的是，它包含某些值的表单域，它包含某些消息或某些链接有效。但如果我们对这些东西感兴趣，就可能把结构抽象出来的 HTML 几乎完全使用，更加人性化的评价方法友好的。例如，而不是使用 XPath 查询/输入【@名称=’amount]找到一个元素，然后评价元素的值属性，我们可以有一个类似的方法 fieldvalueis（“量”，7）将共犯一样。这样的测试是更容易写和可读性好。

6.3.4 使用

TF 的用途是 HtmlUnit 用法非常相似。使用实例 HtmlUnit 是清单 6-1 介绍。

```
01. @Test
02. public void presentForm() {
03.    ControllerTester ct = new ControllerTester("sample1.xml");
04.    Result r = ct.execute("/login.html");
05.    assertThat(r.getViewName(), is("loginForm"));
06.    r.div("loginForm").contains("login.label.username");
07.    assertThat(r.getFormFieldValue("username"), is("john"));
08.    assertThat(r.getHtmlFieldValue("username"), is("john"));
09.    assertThat(r.getModelField("accounts['2']"), is("Account 2"));
10. }
```

Listing 6-2

- 3 行创建一个浏览器对象。
- 4 行请求从网络服务器的网址，并保存结果。
- 5 行评估结果页面有一个标题“HtmlUnit”。

在这两种情况下的测试代码是有组织的 JUnit 测试用例，但 TF 使用 controllertester 代替 WebClient 和结果而不是 HTML。这两个框架提供接口，用于执行应用功能，并通过特殊对象检查结果。这个不同的是，在网络容器外部执行应用程序代码，因此不要求在测试代码执行前部署应用程序。也 tf-s 结果对象提供了比 HTML 更为 HtmlUnit 的评价反馈。清单 6-2 提供简单的实例使用方法。一个简单的教程的转铁蛋白在第 12 章中介绍

```
01. @Test
02. public void presentForm() {
03.    ControllerTester ct = new ControllerTester("sample1.xml");
04.    Result r = ct.execute("/login.html");
05.    assertThat(r.getViewName(), is("loginForm"));
06.    r.div("loginForm").contains("login.label.username");
07.    assertThat(r.getFormFieldValue("username"), is("john"));
08.    assertThat(r.getHtmlFieldValue("username"), is("john"));
09.    assertThat(r.getModelField("accounts['2']"), is("Account 2"));
10. }
```

Listing 6-2

- 3 行创建了 controllertester 对象。“1。XML 配置文件”是 spring 包含应用程序配置。所有春天管理的对象都在这 文件。
- 4 行执行控制器对应的标识符/登录密码。html” 在 spring 的配置。

- 5 行检查控制器代码选择正确的视图来呈现。
- 6 行检查输出的 HTML 的一部分包含一个特定的标签。
- 7 行检查，该表格的字段“用户名”有一个值“John”。
- 8 行检查 HTML 包含表单字段命名为“用户名”这一 值“JOHN”。
- 9 行对模型对象结构进行了评估。

6.3.5 架构与设计

该框架的核心类 controllertester 和结果。这些类绑定在不同于 Spring 框架和 HtmlUnit 类和添加一些自定义 功能。这些类构成了该接口的一种接口。controllertester 可以组成和执行针对网络应用程序和结果的查询 检查结果的评价方法。

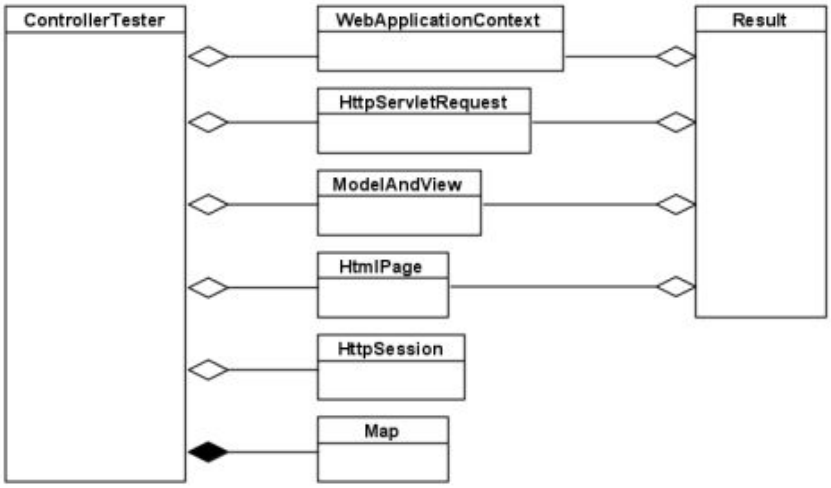

Figure 6-6

WebApplicationContext 是 spring 类。这是春天的最主要的一点访问框架。当我们需要一个管理对象时，我们问为我们创造的 WebApplicationContext。

WebApplicationContext 咨询 31 配置，然后创建所有的依赖关系，创建对象的要求和电线对象请求的依赖关系。TF 采用 WebApplicationContext 创建控制器对象。一个控制器类和它的 WebApplicationContext 实例化它填充数据访问对象，形成 bean 对象和其他对象。其实是有 mockhttpservletrequest HttpServletRequest 对象 spring 分布。该对象被发送到控制器作为参数时控制器代码被调用。此外，该对象可以检查后，控制器代码已执行。在其他事情上很有可能获得通过 HttpSession 对象 HttpServletRequest。

ModelAndView 也 spring 特定对象。当控制器代码被称为返回 ModelAndView 对象。该对象包含了视图选择的信息，并且还包含控制器希望视图呈现的所有动态数据。

HTML 是 HtmlUnit 特定类。这个类是不用于执行测试的。它的目的是简化 HTML 解析代码。HTML 是一个 HTML 文档的抽象它提供了许多方法来查询它所代表的文档。其中它提供了可能性，使用 XPath 获取 HTML 文档元素。

HttpSession 时创建的是当客户端代码调用 controllertester 对象 startsession()方法。此调用后 controllertester 对象创建会话对象填充每个消息发送该会话对象。这使测试功能，这取决于

在会话中存储数据的可能性。这可能是例如用来测试页的报名表，除了最后一个数据收集的所有网页在会议上，所有的数据提交到永久存储在整个页面表格最后提交。地图是普通的 java 地图接口的实现。这是用于收集参数必须传递给控制器。作为 controllertester 照顾创作.

HttpServletRequest 对象还必须用正确的参数填充该对象。所以，controllertester 积累参数的一些查询到这张地图和填充

HttpServletRequest 他们如果有时间打电话给控制器代码需要消息作为参数。controllertester.execute()返回一个结果对象提供了访问请求，会话和模型对象以及 HTML 输出。这里描述的大多数对象共享 controllertester 和请求之间从图 6-6 类图中看到。对于例子是 controllertester HttpServletRequest 对象和结果之间的共享。三十二这是因为 controllertester 创建消息和通过这控制器代码。控制器的代码可以添加一些属性 HttpServletRequest 对象（HttpSession 也可通过 HttpServletRequest 对象）和之后的控制器代码完成后，我们希望看到，该控制器实际上是所需的修改。因此，可以通过结果检查 HttpServletRequest 对象对象。结果提供了几种评价方法，简化了 HTML 的检查输出和中间状态的执行。