****

**网站UI自动化测试**

**技术方案**

|  |  |  |  |
| --- | --- | --- | --- |
| 文档状态 | [√] 初稿 | 当前版本 |  |
| [ ] 评审通过 |
| [ ] 修改 |
| [ ] 发布 |
| [ ] 作废 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **日期** | **版本** | **修订内容描述** | **作者** | **审核人** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**修 订 记 录**

**目 录**

[一、 概述 4](#_Toc521572827)

[1.1. 定义 5](#_Toc521572828)

[1.2. 意义 5](#_Toc521572829)

[1.3. 前提 5](#_Toc521572830)

[二、 框架 6](#_Toc521572831)

[2.1. 目标 6](#_Toc521572833)

[2.2. 技术 6](#_Toc521572834)

[2.2.1. Selenium 6](#_Toc521572835)

[2.2.2. TestNG 7](#_Toc521572836)

[2.2.3. Maven 7](#_Toc521572837)

[2.2.4. AutoIt 8](#_Toc521572838)

[2.2.5. Jenkins 8](#_Toc521572839)

[2.3. 技术框架 9](#_Toc521572840)

[三、 实现 9](#_Toc521572841)

[3.1. 设计思路 9](#_Toc521572843)

[3.2. 代码结构 10](#_Toc521572844)

[3.3. 代码规则 10](#_Toc521572845)

[3.3.1. 用例代码规则 10](#_Toc521572846)

[3.3.2. Java代码规则 11](#_Toc521572847)

[3.4. 代码实现 11](#_Toc521572848)

[3.4.1. 测试框架 11](#_Toc521572849)

[3.4.2. 测试用例结构 12](#_Toc521572850)

[四、 持续集成 13](#_Toc521572851)

[4.1. 设计思想 13](#_Toc521572853)

[4.2. Job执行架构 13](#_Toc521572854)

[4.3. 执行结果 13](#_Toc521572855)

[五、 测试效果 14](#_Toc521572856)

[六、 近期目标 14](#_Toc521572857)

[七、 远期目标 15](#_Toc521572858)

# 概述

有别于传统桌面应用开发，Web应用因为其迭代周期短，需求变化快的特性，越来越多的Web开发团队在拥抱敏捷。同时随着前端技术ajax、JavaScript、Html5等不断被推出和完善，Web应用也变得越来越复杂和绚丽。这都给我们的测试带来了挑战。

* 持续变化的需求
* 复杂的UI（各种JavaScript插件，Ajax等）
* 往往一个测试团队需要应付多个项目的测试工作
* 非常有限的资源（人力，成本，等等…）
* 跨平台（Linux，Mac，windows），跨浏览器（Firefox，Chrome，IE）

如何在有限的资源下，有效地解决目前我们测试团队所面临的挑战呢？最好最有效的方式即是通过工具及自动化手段实现。当前web端功能测试使用的自动化工具主要是QTP、Selenium等。

* QTP：优点在于技术要求较低，可通过录制调试实现大部分自动化功能；缺点也很明显，软件使用时间、用户数都需要购买。
* Selenium：优点在于免费、友好、开源、网上使用比较多；缺点也是很明显，需要比较专业的代码知识，较强的学习能力。

综合考虑，虽然Selenium开发的网站自动化测试框架起始阶段较为困难（主要是框架设计、环境搭建、代码逻辑、可移植性），但一旦前面的工作完成，后续工具都可比较顺利的开展。一人带头、多人推进，实现项目的持续、高效、高质的输出。

## 定义

UI即User Interface，用户界面。

UI自动化测试是指对软件的人机交互、操作逻辑、功能设计等方面通过工具/脚本进行测试。

## 意义

* 减少重复工作
* 用例回归
* 主流程冒烟

## 前提

* UI界面比较稳定
* 业务逻辑及需求变更较少
* 平台开发较为规范
* 软件周期较长

# 框架



## 目标

* 测试框架可移植
* 测试脚本易读、易修改
* 脚本可配置
* 提高测试用例执行效率
* 减少重复劳动
* 持续集成

## 技术

* 开发语言：Java
* 开发IDE：Eclipse
* 测试框架：Selenium + TestNG
* 项目管理工具：Maven
* 持续集成：Jenkins
* 辅助工具：autoIt

### Selenium

Selenium是一种跨平台、跨浏览器端到端Web自动化解决方案。Selenium主要包括：Selenium IDE、Selenium WebDriver Selenium Grid。网站自动化测试主要使用WebDriver。

* Selenium IDE：Firefox 插件，可录制回放并导出多种脚本语言（Java、C#等）
* Selenium WebDriver：Selenium API，包括浏览器控制、页面元素处理等。
* Selenium Grid：在多机情况下，实现多浏览器并发执行

### TestNG

TestNG是一个测试框架，其灵感来自JUnit和NUnit，但引入了一些新的功能，使其功能更强大，使用更方便。其特点为：

* 注解性
* 参数化
* 数据驱动
* 灵活的运行时配置
* 支持依赖测试
* 灵活的插件API
* 支持多线程测试

### Maven

Maven是一个项目管理工具。[Maven](http://www.yiibai.com/maven)提供了开发人员构建一个完整的生命周期框架。开发团队可以自动完成项目的基础工具建设，Maven使用标准的目录结构和默认构建生命周期。

在多个开发团队环境时，Maven可以设置按标准在非常短的时间里完成配置工作。由于大部分项目的设置都很简单，并且可重复使用，Maven让开发人员的工作更轻松，同时创建报表，检查、构建和测试自动化设置。

概括地说，Maven简化和标准化项目建设过程，增加可重用性并负责建立相关的任务。

### AutoIt

AutoIt设计用于Windows GUI([图形用户界面](https://baike.baidu.com/item/%E5%9B%BE%E5%BD%A2%E7%94%A8%E6%88%B7%E7%95%8C%E9%9D%A2))中进行自动化操作。模拟键盘输入、鼠标移动/点击、窗口/控件处理等组合实现自动化任务。

* 运行Windows和Dos程序
* 模拟键击动作(支持大多数键盘布局)
* 模拟鼠标移动和点击动作
* 对窗口进行移动,调整大小和其它操作
* 直接与窗口/控件“交互(设置/获取文本,移动,关闭等等)
* 配合剪贴板进行剪切/粘贴文本操作
* 对注册表进行操作

### Jenkins

持续集成(CI)是现代软件开发领域的基石，它改变了团队对于整个开发过程的理解。CI架构能够使得从开发到部署顺序进行，更快地发现和修复bug，最终给客户带来更多的价值。Jenkins即为这样一款，使用java语言编写的持续集成工具，其支持各种语言开发实现的项目，比如.net、Java、ruby、PHP等。其特点：

* 开源
* 易于使用，界面简单、直观
* 扩展性
* 分布式

## 技术框架

Maven

Java

TestNG

XML

Auto It

SQL DB

Test Result Report

Selenium WebDriver

Jenkins

1. 概要设计方案
2. 详细代码说明
3. Jenkins部署

# 实现



## 设计思路

使用Java作为开发语言，Selenium + TestNG 作为测试框架，Maven作为项目管理工具，Jenkins作为持续集成手段，来实现整个项目的开发测试。

框架有以下的特性：

* 脚本与框架分离（test、main）
* 对象库分离（page、script）
* 脚本框架组织、管理方便（maven）
* 脚本可重用、可配置（testng）

Initial

Page object

Test script

Lib

Test data

Framework

Microseer common

Tool utils

Config

Webdriver API

testng

Assert

Main

Microseer-Automation

Test

Test module

…

Initial

Page object

Test script

## 代码结构

## 代码规则

### 用例代码规则

* 常使用的方法，写入公共方法
* 自动化用例简单化，逻辑清晰
* 元素及动作放在page中，用例验证放在script中
* script中用例总数保持在30条左右，便于调试
* script加入相应模块注释
* case加入group及timeout信息
* case名称体现case验证信息
* 添加case的用例名称及作者
* case异常使用框架中的异常处理
* case执行完，若有alert弹窗，处理掉

### Java代码规则

* 尽量少使用全局变量
* 自定义变量使用有意义的字符串
* 类名以大写字母开头
* 方法以小写字母开头
* 方法前添加注释及author
* 关键逻辑添加注释信息

## 代码实现

### 测试框架

* 测试框架类（Framework）
  + WebDriver API封装类，包括元素处理、WebDriver初始化、WebDriver 控制、等待、多浏览器及并发等
  + 常用方法集Utils，包括string处理、xml处理、char处理、excel处理、csv处理、文件处理、数据库处理、zip包处理等
  + TestNG类，包括测试用例初始化，测试用例执行前准备beforeSuite、beforeTest、beforeClass、beforeGroup、beforeMethod；测试用例执行后的操作afterMethod、afterGroup、afterClass、afterTest、afterSuite等
* 公共方法类（MicroseerCommon）
  + 项目基本方法类
  + 模块特殊方法类
* 测试用例断言类（assert）
* 测试用例异常处理类
* 日志及报告配置类

### 测试用例结构

* Initial

（继承于TestNGCaseBaseUI）Case初始化脚本

* Page

（继承于MicroseerCommon），其中包括元素、验证方法，以及一些其他的有关数据操作等的方法

* Script

（继承于TestNGCaseBaseUI）中添加一条测试用例。包括@Test标签、组信息、限时，case断言、失败截图及异常

* Xml

testng.Xml、pom.xml，在class中引用script

# 持续集成



## 设计思想

定时执行、多节点分布式执行、模块化、代码自动部署

## Job执行架构

Jenkins（Master）

Client2

Client1

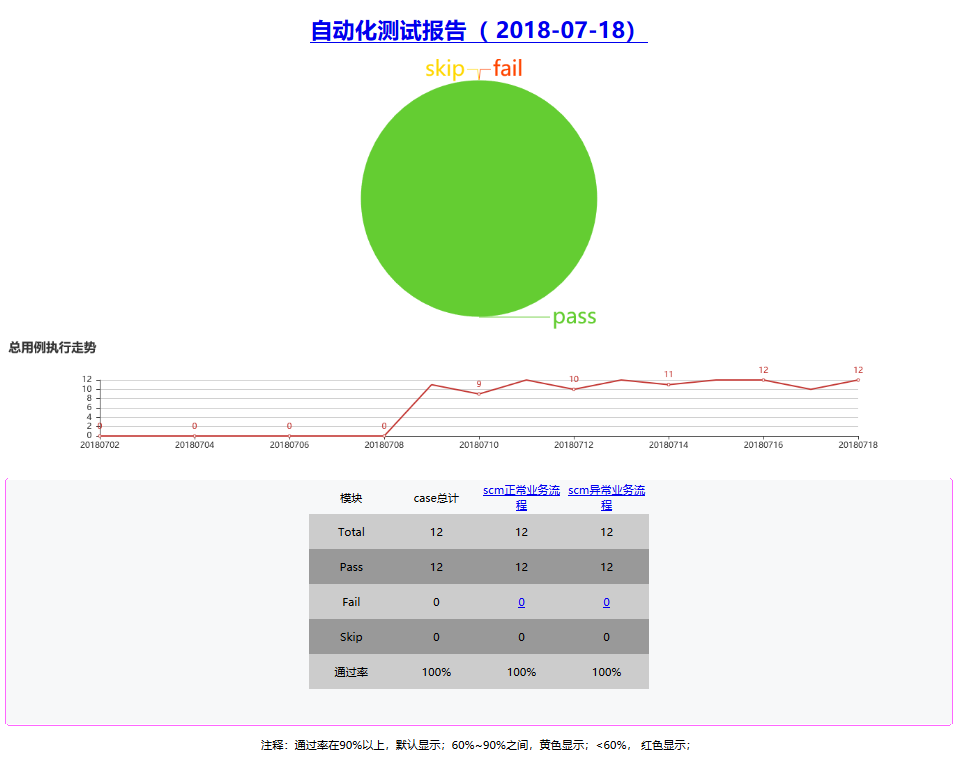
（client可添加）……

Job随机

Job随机

## 执行结果

* + - 概括化，即实现Jenkins结果的总结、统计
    - 可视化，即通过输出报告，查看测试失败、通过、跳出情况
    - 模块化，即通过输出报告，查看各模块测试失败、通过、跳出情况



# 测试效果

* 通过Jenkins，实现了用例持续执行
* 验证一些逻辑较复杂、繁琐的case，提高了测试效率
* 通过对相应模块xml文件的操作，即模块下某一组测试开关的打开关闭，实现回归测试
* 通过调整Jenkins配置，实现测试某单一模块或几个模块的集合

# 近期目标

* UI测试框架搭建及主流程（1级用例）测试用例开发
* Jenkins CI完成搭建
* Jenkins报告页面输出

# 远期目标

* 自动化用例覆盖60%以上
* 自动化测试用例平台化