中山大學
SUN YAT-SEN UNIVERSITY

本 科 生 毕 业 论 文

题　　目：web 前端测试工具开发

院　　系：软件学院

专　　业：计算机应用软件

学生姓名：陈学家

学　　号：10389054

指导教师：王青（讲师）

二〇一 四 年 三 月

摘　要

随着 web 技术的发展， web 前端作为其重要组成部分， 其技术越来越复杂，前端工程的代码量急剧增加， 以此带来了前端工程代码质量降低， 可维护性降低等问题。 前端测试驱动开发成为解决这一问题的必经途径。 然而因为前端工程的特殊性， 前端代码可测试低， 测试成本高， 测试驱动开发模式很难应用到前端工程当中。 针对这一现状，一些以简化前端测试流程的工具应运而生。

本项目就是针对前端开发的特殊问题，设计测试工具来降低前端测试驱动开发的成本， 提高前端代码的可测试性。 本项目的主体分为：1. 项目开发工程背景介绍；2. 项目的技术原理；3. 框架设计；

项目开发： 用 Node-Webkit 来作为工具的运行环境，selenium 来作为测试的基本方案。 应用系统分析设计的基本原理， 讲解了本项目开发过程中的：需求分析，用例分析， 领域模型，活动图，状态图，类图等。

技术原理：针对本项目的开发， 讲解了项目中所用到的技术，包括：Node-Webkit， selenium， socket.io, jasmine, backbone 等， 以及阐述了项目中遇到的主要问题和对应的解决方案。

框架设计：讲解了本系统基于 MVC 设计模式和可扩展的大型 javascript 项目原理结合应用到本项目的框架结构设计当中。

关键词： 测试驱动开发；测试工具开发；Node-Webkit；selenium；软件工程

ABSTRACT

With the development of web technology. The technology of front-end become more and more complex. The code size of front-end project increased dramatically. As a result, the quality and the maintainability of front-end code is reduced. Test driven development may be the key to solve the problem. However, font-end development is special. It has lower testability and higher test cost. Test driven development can hard be apply to front-end project. Aimed at this, some front-end test tools came out in order to simplify the process of front-end testing.

This project is aimed at the same problem of front-end testing. And design a testing tool called testx to simplify the process of front-end testing, reduce the cost of test-driven development in front-end. The subject of this paper is divided into three parts: 1. Project Development Background; 2. Technical Principles; 3. Architecture Design;

Project Development Background: Using Node-Webkit as a runtime environment , using selenium as the browser driver. Apply the basic design principles in Systems Analysis, and explain the whole development process: requirements analysis, use case analysis, the domain model, activity diagrams, state diagrams, class diagrams.

Technical Principles: Explain the technology used in this project including: Node-Webkit, selenium, socket.io, jasmine, backbone. And describes the main problems and the corresponding solutions encountered in project .

Architecture Design: Explain the architecture design of this project with the MVC design pattern and large scale javascript project design pattern.

Keywords: Test-driven development; test tool development; Node-Webkit; selenium; Software Engineering

目　录

第一章 前言

1.1 项目背景和意义

从 web1.0 时代到 web2.0 时代再到现在的 web app 时代， web 技术的发展让人瞠目结舌。 而 Web 前端作为 web app 的重要组成部分， 其相应的技术得到了前所未有的提升。 web 前端开发的技术变得越来越复杂：前端模块化， MV*[1]模式，前端脚本预处理，脚手架工具，web component[2]。各种新技术的出现意味着前端开发的难度和复杂度的提升。 前端代码不再是以前的 javascript 代码片段， 而是变得有组织， 模块化， 结构化的工程性的庞大代码仓库。 然而庞大的代码带来了如下问题：

1. **代码的可维护性变低；**

2. **代码质量变低；**

3. **开发缓慢， 迭代速度变慢；**

而从其他语言的项目开发中借鉴经验， 解决这些问题的一个有效方法是使用-**测试驱动开发**。 这种方法已经被应用到大多数项目开发当中， 是敏捷开发遵从的实践方法。 在各大语言当中都有相应的 UnitX 实现， 如 java， python， php 对应的 UnitX 框架有 junit，PyUnit， PHPUnit。对于 javascript，有如下框架：jasmine，mocha，chai 等等。 其实这些框架已经被应用于很多的大型前端应用开发当中。但事实上国内如阿里，腾讯甚至国外 facebook 等大型互联网公司并没有在前端业务开发中应用测试驱动开发。 那为什么大型公式不愿意利用测试驱动开发模式来优化前端工程项目的开发呢？ 前端开发和其他项目的开发有一下几个不同点：

1. **可测试性低。**前端的代码大多是交互和逻辑在一起，为了做页面的加载优化，很多数据需要 ajax 异步请求， 或者使用 bigpipe[3]模式， 这导致前端测试与传统的测试很大不同；

2. **测试开发成本更高**。 为了完成测试，需要做很多额外的工作，并在开发过程中不断的在测试环境和线上环境做代码切换。 这些问题使得测试驱动开发很难应用到前端业务代码当中；

3. **迭代速度快**。 对于大型网站来说，前端代码的更新速度相比其他项目更快，需要做快速的代码变更和迭代， 这导致前端开发人员根本没有时间顾及测试代码的编写；

刚才提到的 js 测试框架如 chai， jasmine 大多应用在核心逻辑代码， 框架代码或者

---

[1] MV*指的是前端开发中的 MVC， MVP， MVVM 模式的概称。

[2] Web component 一直让前端的元素更加组件化的范式， 目前已经纳入到下一代 web 的草案中。

[3] Bigpipe 模式是一种优化前端加载的方法， 在一个 http 链接下将页面分块加载进来， 目前国内的大型网站如淘宝， 新浪， 腾讯都是利用这种方法来实现页面加载优化。

nodejs 项目中。如果不涉及到交互和大量异步，这些框架已经提供了很好的前端测试基础。在此基础上为了解决上边提到的前端开发中的三个问题， 为了解决上面所说的问题， 最简单的方法就是在这些测试框架基础上提供工具来简化前端测试的步骤， 实现前端测试的最简化。 本文就是旨在设计一个这样的前端开发工具，做前端测试过程的管理和简化，以及解决前端开发过程中的交互异步问题。

1.2 研究开发现状分析

在前端测试工具领域， 开源界已经做了很多的尝试， 并且已经有了很多不错的成果。如 testem，karma，totora 。 前两个工具室国外的开源项目， 其中 karma 是目前 AngularJs[4] 团队所有的测试工具，所以如果项目中使用 AngularJs ，那么一定会用到这个测试工具。 第三个项目 totora 是国内淘宝的开源项目。 就使用而言，这些工具已经极大的简化了前端测试过程中的测试复杂问题。 总结一下它们有一下几个共同点：

1. **Node.js 实现。** 都是通过命令行来做测试的启动以及测试的报告输出， testem 甚至直接利用命令行来做交互。

2. **在真实浏览器当中运行。** 不是通过一个模拟程序来做测试， 都是通过在真实的浏览器环境当中运行， 这些浏览器包括 ie6-10 系列， chrome， firefox 等等。

3. **在浏览器中启动一个 runner 页面来运行测试样例**。为了让测试脚本能够在浏览器当中运行，需要一个代理页面来运行脚本， 而这个页面就是 runner 页面。Testem 是自动生成的页面，karma 和 totora 都是可以自定义 runner 页面。

4. **通过 adapter 支持多种测试框架。**这几个工具都能支持 jasmine， mocha， qunit 等测试框架

尽管目前这些工具都已经做到极大简化了测试的步骤， 并且可以让用户零成本的切换到工具的使用（无需改变原来的测试代码）。 但相对来说仍然存在如下问题：

1. **runner 页面无法完全模拟真实的运行环境。** runner 页面与需要测试的 web app 所在的应用地址无关， 这就意味着在做测试之前需要配置或者加入应用的代码才能执行测试脚本。 而这一过程会提高测试的成本。 除此之外 runner 页面另外一个缺点是无法简单的配合 web app 中的 dom 结构做测试，并且测试过程中由于 runner 在不同的域，会存在真实环境中不存在的跨域问题， 这对于 ajax 异步加载情况很难处理。

[4] AngularJs 是 google 打造的一个 MVVM 的前端框架。

2. **命令行模式很难做到好的交互体验。** 虽然基于命令行的测试工具对程序员来说简单直接，但当测试报告变多， 很难在命令行中查找并做错误分析， 并且命令行模式下很难完成交互动作。而本文所做工具利用 node-webkit 实现，基于桌面应用提供更好的数据展现效果。

1.3 项目目标和范围

本项目的目标就是基于 node-webkit 和 selenium webdriver 开发一个前端测试工具-TestX(后文都将以 TestX 来命名本项目所开发的工具)。 通过这个工具来简化前端测试过程的复杂度，让用户只需要关注编写测试脚本。 这个工具有以下几个功能:

1. 使用现有的测试框架来编写测试 specs[5]；
2. 界面化和可配置的测试过程管理， 通过 node-webkit 来实现桌面应用程序， 并通过这个应用来管理测试过程；
3. 可以选择在 firefox， chrome， ie 或者 phantomJs[6]中运行；
4. 提供交互异步解决方案， 为了满足前端的交互异步需求， 提供工具类实现前端交互动作的模拟；

1.4 论文结构介绍

这篇论文分为七个章节：

第一章：前言

说明项目的背景和开发意义，简洁现有的前端测试工具并分析了其优势劣势，概述本项目的目标和范围。

第二章：技术与原理

介绍该项目中所用到的技术以及如何应用这些技术组合实现本项目中的测试工具。

第三章：需求建模

说明本项目的需求，以及细化的讲解本项目所实现的功能，功能的可用性以及受到的

[5] 测试规格说明， BDD 模式将测试分为 test suit， test spec， test expect。
[6] phantomJs 是基于 webkit 的一个无头浏览器，除了没有界面以外， 和在真实的 webkit 浏览器中运行并无差别， 通常利用 phantomJs 来实现网页的截图以及数据抓取或者做交互测试。

限制。

第四章：架构设计

说明本项目的代码工程结构以及所用到的设计模式和方法。

第五章：模块设计

细化的说明项目工程代码中的具体模块的功能和实现。

第六章：部署和应用

说明本工具的安装部署流程，以及通过本工具实现一个 todo-list 实例。

第七章：总结

介绍本项目开发过程中， 个人的心得体会。 以及总结本项目的优缺点以及可能的优化方案

第二章　技术与原理

2.1 概述

这个章节主要描述本项目所用到的技术细节。 这些技术包括 node.js，node-webkit，selenium，wd.js，jasmine，socket.io，promise，backbone，bootstrap，Less。 以及利用这些技术实现测试的基本原理。

2.2 Node.js

Node.js 是一个基于 Chrome-JavaScript 运行时建立的一个平台，用来方便地搭建快速的，易于扩展的网络应用• Node.js 借助事件驱动，非阻塞 I/O 模型变得轻量和高效，非常适合 run across distributed devices 的 data-intensive 的实时应用。

Node.js 的简单易用性使其在短短的几年间发展迅速，甚至 paypal 这种大型公司的后端服务程序都改为 Node.js 所写。 Node.js 的易用性可以通过下面这段简单的 hello world 服务代码可以看出：

```
1  //--require a http module
2  var http = require('http');
3  //--create a server with a callback
4  //--listen to 1337 port
5  http.createServer(function (req, res) {
6    res.writeHead(200, {'Content-Type': 'text/plain'});
7    res.end('Hello World\n');
8  }).listen(1337, '127.0.0.1');
9  console.log('Server running at http://127.0.0.1:1337/');
```

图 2-1： Node.js 写的服务器程序

因为 Node.js 与前端使用统一的语言， Node.js 逐渐成为了前端工程师必备的基本技能。并且前端相关的很多工具也都是通过 Node 平台实现，比如前端的构建工具 grunt，gulp，前端的 mock 工具平台， js 静态分析工具等。 这也是在 1.2 节提到的现有前端测试工具都是通过 node 平台搭建的原因。

本项目中的服务程序都是通过 Node.js 所写， 并且感谢 npm[7]丰富的模块支持， 对本项目提供了很大帮助。

2.3 node-webkit

node-webkit 是一个基于 Chromium 和 Node.js 的 Web 运行环境，可让直接在 DOM

[7] Npm 是 nodejs 的包管理器， 包好丰富的 node 第三方模块。

中调用 Node.js 模块，并可使用任何现有的 Web 技术来编写本地应用， 目前 node-webkit 主要由 Intel 公司维护， 在 github 上开源。

node-webkit 将 chromium 和 node.js 的事件循环[8]绑定在一起，所以在应用代码中既可以使用 npm 中的模块也可以使用 chromium 中支持的 css3，html5 API。 这种优势极大的简化了桌面应用的开发流程， 下图很好的解释了这种工作方式。

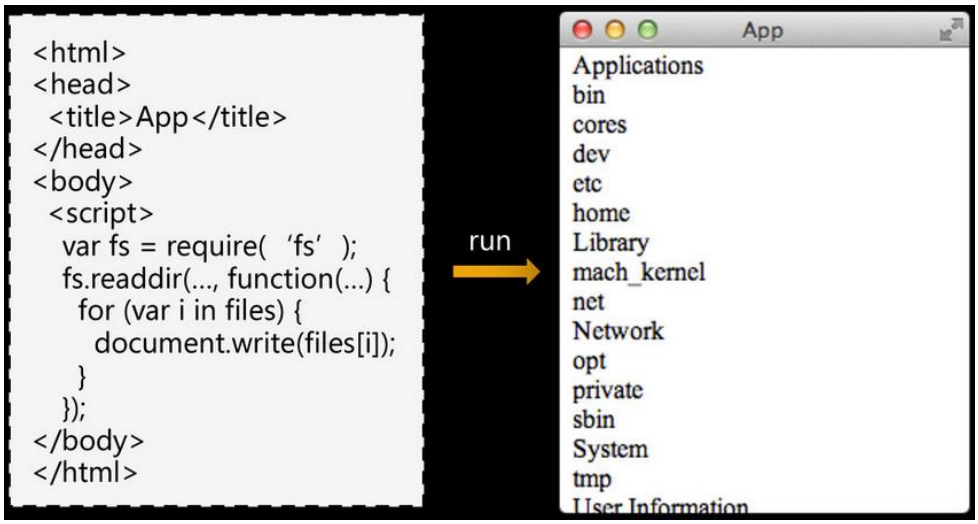

图 2-2：node-webkit 示例程序

目前国内已经有了很多项目是使用 node-webkit 构建，未来 node-webkit 更有可能发展到移动端的 Native App[9]开发， 而本项目所用到桌面应用开发的技术就是 node-webkit。

下图简单的描绘了 node-webkit 开发的基本目录结构， 通过 package.json 可以配置应用的窗口视图，程序的运行模式。 Node-webkit 所开发的项目大多是 SPA 应用， 所以基本上都只会看到一个 html 文件， 其他的都是脚本文件盒样式文件以及资源文件。

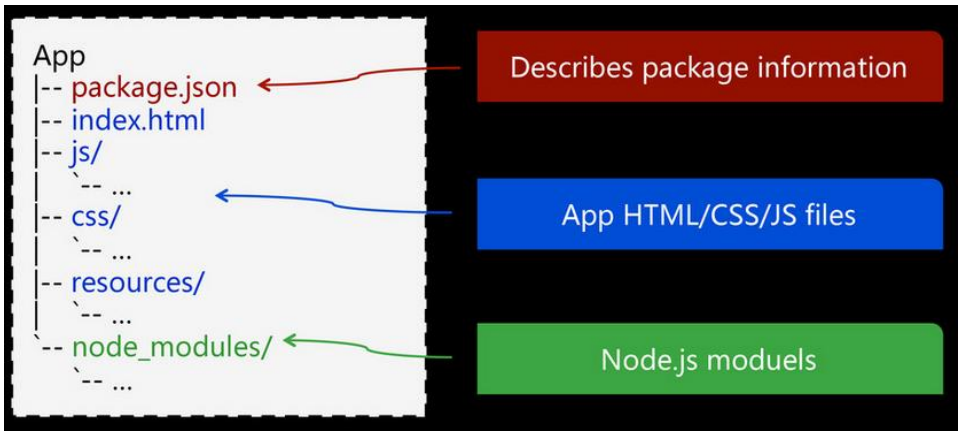

图 2-3：node-webkit 程序目录结构

[8] Js 的执行环境都是是单线程的， 在一个事件循环来完成所有的工作。
[9] Native App 是指区别于浏览器 web 应用的本地应用， 包括桌面应用以及移动端的原生应用。

2.4 Selenium 以及 wd.js

Selenium 是一个用于 Web 应用程序测试的工具。 测试直接运行在浏览器中，就像真正的用户在操作一样。 支持的浏览器包括 IE(7、8、9)、Mozilla Firefox、Mozilla Suite 等。selenium 是目前而言做 ATDD[10]测试的首要选择，下图为 selenium 的工作方式。

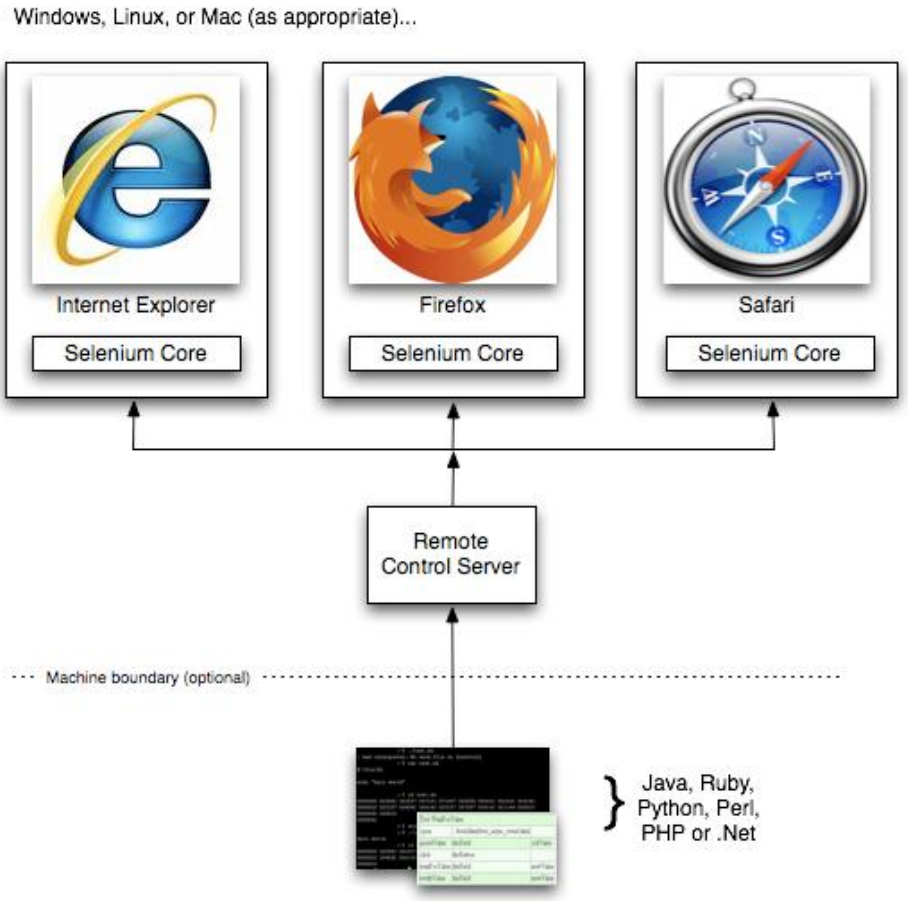

图 2-4：selenium 原理图

可以通过 selenium 控制各浏览器的运行，并且驱动 selenium 的程序可以用 java，ruby，python， node 等程序所写。 而本项目中的驱动程序就是用的 wd.js。 wd 是 webdriver 的简称。 Wd.js 是基于 node.js 的 selenium2 webdriver 开源模块。 通过 wd.js 可以很容易的实例化一个 broswer 对象。 并通过此对象来实现对浏览器的实际操作。 wd.js 的提供了基于 promise 模型的 syntactic sugar。 更是简化了 webdriver 的异步操作。

本项目中也用到了 selenium-standalone 这个 npm 开源模块。 这个模块帮我们管理了 selenium-server 的运行。当启动了 selenium-server 过后，我们再通过 wd.js 创建一个 remote 的 broswer 对象， 通过这样的协作方式来运行测试脚本。

[10] ATDD 是指验收测试驱动开发

2.5 Jasmine

Jasmine 是一个前端 javascript BDD 测试框架。 不依赖与任何第三方库，也不依赖于 DOM。可以在 Nodejs 环境和浏览器环境中运行。 提过简洁易用的 API，让前端测试代码过程变得极其简单直观， 下图为 jasmine 的简单实用示例。

```
1  //--use describe to define a test suit
2  describe("A suite", function() {
3    //--use it to define a test spec
4    it("contains spec with an expectation", function() {
5      //--BDD style expect
6      expect(true).toBe(true);
7    });
8  });
```

图 2-5：jasmine 示例程序

目前本项目使用 Jasmine2.0 作为默认测试框架， 未来会提供 mocha，Qunit，chai 等测试框架的兼容。 为了做到测试框架的兼容， 会对每个框架实现一个 adapter。 通过这个 adapter 提供统一的 api。

2.6 socket.io

Socket.io 是 npm 中的一个旨在让各种浏览器与移动设备上实现实时 app 功能的模块。其优势兼容了各个浏览器， 模糊化各种传输机制， 对于低级浏览器实用 ajax long polling，jsonp， iframe 等兼容方案， 对于高级浏览器用 websocket 方案。 并且解决了跨域问题。目前基于 node.js 的实时应用如群聊， 多人在线游戏都是通过 socket.io 实现。 一个简单的 socket.io 使用示例：

```
//--server
var io = require('socket.io').listen(80);

io.sockets.on('connection', function (socket) {
  socket.emit('news', { hello: 'world' });
  socket.on('my other event', function (data) {
    console.log(data);
  });
});

//--client
<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io.connect('http://localhost');
  socket.on('news', function (data) {
    console.log(data);
    socket.emit('my other event', { my: 'data' });
  });
</script>
```

图 2-6：socket.io 示例程序

示例中可以看出 Socket.io 的使用简洁性， 基于 pub/sub 机制让复杂的同步变得简单易用。 而在本项目中， 使用 socket.io 实现测试浏览器和应用之间的数据同步和状态同步。

2.7 promise 模式

异步模式在 web 编程中变得越来越重要，对于 web 主流语言 Javscript 来说，这种模式实现起来不是很利索，通常的方式是通过 callback 回调函数。当回调函数变深过后，无论对于书写和阅读都会出现很多问题。 为此提出了一种称为 promise 的抽象（有时也称之为 deferred）。 通过这种模式可以简化异步代码的书写， promise 已经在 ECMAScript6 中得到原生支持。

下面看一下使用原来 callback 方式的示例：

```
Parse.User.logIn("user","pass", {
  success:function(user) {
    query.find({
      success:function(results) {
        results[0].save({ key: value }, {
          success:function(result) {
            // the object was saved.
          }
        });
      }
    });
  }
});
```

图 2-7：回调模型

使用 promise 过后的示例：

```
Parse.User
  .logIn("user","pass")
  .then(function(user) {
    returnquery.find();
  })
  .then(function(results) {
    returnresults[0].save({ key: value });
  })
  .done();
```

图 2-8：promise 模型

代码的可阅读性明显提高， 本项目中要解决的一个问题是提供交互的 simulate。 交互的 click， mouseover， select， keydown， keyup 这些操作都是异步的， 为了简化这些异步操作的书写， 本项目提供了基于 promise 的 simulate 实现。 并且在 promise 之上做了一个异步队列， 用户只需要往队列里边 push 操作就行， 尽管这些操作时异步的。

2.8 前端技术

由于本项目所开发的测试工具提供 UI 界面化操作。 为了实现这些 UI 界面， 使用了如下工具： Ejs， Bootstrap， Backbone， Less。 下面对这些技术做简要介绍。

2.8.1 Backbone

Backbone 是一个前端 MVC 框架，提供构建 web app 基本的架构。包括带有 key-value 以及自定义事件绑定的 model 类。带有丰富 API 的 collections 类。声明式 定义事件绑定的 view 类。Backbone 以及在很多大型 web 应用中得到应用。 本项目中的客户端 UI 开发是基于 backbone 的 MVC 模式开发。

2.8.2 Ejs

Ejs 是一个开源的 html 模板引擎。通常在 Node.js 中使用， 本项目中， 由于界面的设计使用了 MVC 模式。 而 Ejs 就是作为 Backbone View 的模板。

2.8.3 Bootstrap

Bootstrap 是 Twitter 推出的一个开源的用于前端开发的工具包。它由 Twitter 的设计师 Mark Otto 和 Jacob Thornton 合作开发，是一个 CSS/HTML 框架。Bootstrap 提供了优雅的 HTML 和 CSS 规范，它即是由动态 CSS 语言 Less 写成。Bootstrap 一经推出后颇受欢迎，一直是 GitHub 上的热门开源项目，包括 NASA 的 MSNBC（微软全国广播公司）的 Breaking News 都使用了该项目。

2.8.4 Less

Less 是动态的样式表语言，通过简洁明了的语法定义，使编写 CSS 的工作变得非常简单。由上面的 bootstrap 简介可知，bootstrap 的样式是使用 Less 所写，可见 Less 已经在前端开发中占有一定的地位。 而本项目中的界面样式也是通过 Less 所写。

2.9 基本原理

为了更清晰的了解本项目的实现原理， 下面将会概括性的简要讲解如何使用上面提到的技术来实现测试管理， 下图为本项目的实现原理图：

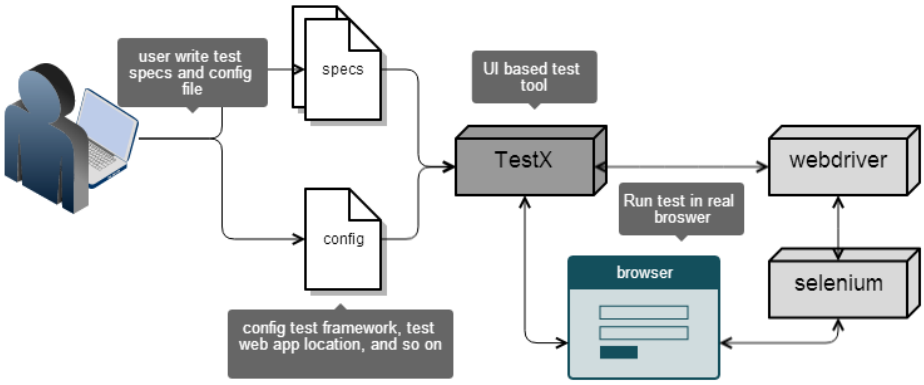

图 2-9：TestX 基本原理图

由上图可以看到， TestX 的测试过程大概分为以下几步：

1. 用户编写测试 specs， 以及 TestX 配置文件；

2. 在 TestX 中以项目的形式添加测试项目；

3. 在 TestX 中通过快捷键或按钮运行测试；

4. TestX 通过使用 webdriver 驱动 selenium 运行浏览器；

5. 在浏览器中运行测试， 并将测试结果返回到 TestX 中；

6. 用户可以查看经过格式化的测试报告， 以及测试过程中的 log 信息；

第三章　需求建模

3.1 概述

在软件开发过程中， 需求的定义是： 系统必须提供的能力和必须遵从的条件。 因此需求分析系统建设过程红不可或缺的一部分， 是软件建设的核心目标。 软件建设的过程是从需求分析到功能设计再到系统设计的过程。 需求建模是软件项目开发必经的过程， 将直接关系到项目开发的风险。 本章将分析本项目的需求建模过程。

3.2 需求分析

在第一章中已经介绍了 TestX 所要解决的几个问题， 概括下来本项目的核心目标就是简化前端测试的过程， 并提供交互异步的解决方案。 下面将围绕着这两个核心目标提出功能性需求。

3.2.1 在真实的浏览器中运行测试

为了能够测试应用代码在不同浏览器当中的兼容性， 已经因为前端应用的特殊性。测试脚本在真实浏览器中运行成为了一种隐性需求。 具体如下：

1. 工具能够将用户编写的测试脚本运行在真实的浏览器中；
2. 工具能够让用户选择在 chrome, firefox, ie , phantomJs 等平台下运行；
3. 工具能够让用户自定义需要测试的应用地址，并在这个应用加载成功后启动测试，在完全真实的环境中运行；

3.2.2 测试脚本与应用代码分离

为了简化前端测试过程的复杂度，用户不用在写测试代码的同时还需要关注应用代码。将应用代码与测试代码完全的分离编写。 具体如下：

1. 编写测试脚本的过程和应用代码独立，放在不同的目录下；
2. 测试代码中可以获取应用代码中的全局变量；

3.2.3 多测试框架选择

某些前端应用已经使用了自己的测试框架， 并且已经编写了大量的测试代码， 为了做到零成本切换， 工具需要提供各个测试框架的适配， 具体如下。

1. 工具能够支持 jasmine, mocha, Qunit, chai 来编写测试代码；

2. 测试的结果不因为框架的改变而改变；

3.2.4 测试过程的可视化管理

测试过程不应该局限于在 terminal 中查看测试报告。 测试结果应该是可交互的， 用户应该知道哪里错了， 并点击可以看到具体的错误代码。 具体如下：

1. 通过工具可以选择测试的项目；
2. 通过点击运行以及快捷键按钮就可以运行测试项目；
3. 提示运行过程进度的信息；
4. 通过列表展示测试结果；
5. 提示测试运行结果的成功或错误状态；
6. 输出用户在测试脚本中的 log 信息；
7. 展示错误信息， 错误位置， 对于错误代码行数可以点击查看具体的代码；

3.2.5 多个测试项目的管理

通常用户会拥有多个项目， 并每个项目都会有相应的测试文件， 那么本工具将提供项目机制来管理这些测试， 具体如下：

1. 用户可以随时添加或者删除一个项目；
2. 选择每一个项目的时候， 点击运行按钮会运行相应的项目测试脚本；
3. 当用户关闭工具再重新启动工具的时候，以前添加的项目不会改变；

3.2.6 可配置化

为了能够实现定制化的测试方案， 提供测试运行配置脚本 config.json。 具体如下：

1. 用户可以在配置脚本中配置运行的浏览器， 运行的测试应用地址， 测试框架， 以及其他一些配置项；
2. 如果没有 config.json ， 工具也同样能够正常运行；

3.3 用例分析

设计用例图， 并用一个列举想描述两个关键用例。

3.3.1 用例模型

针对上述的需求分析， 设计了如下用例图：

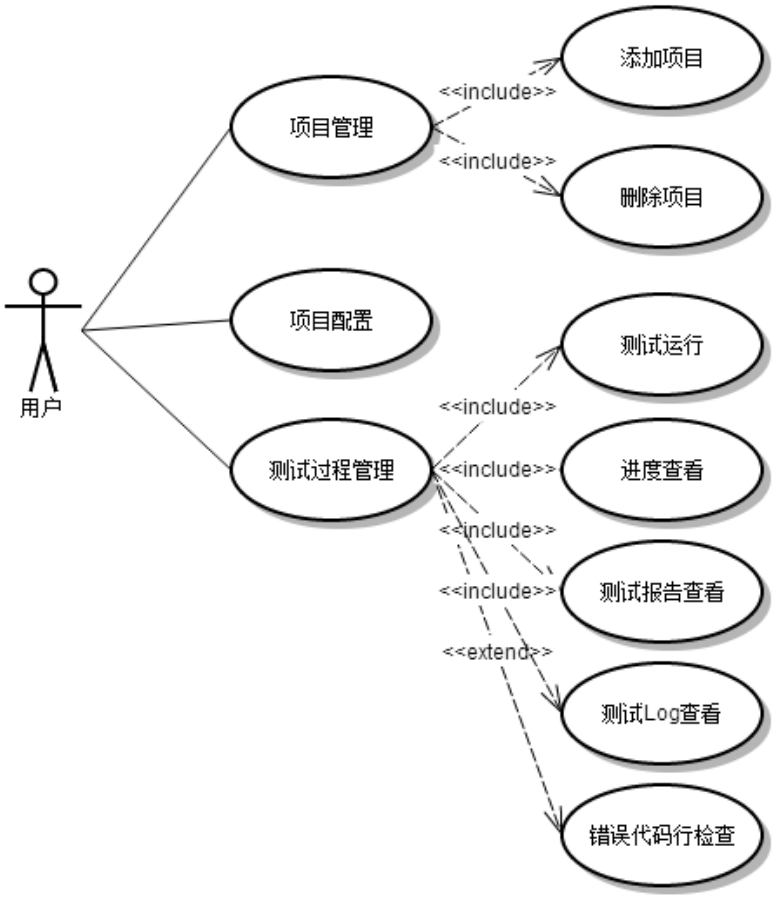

图 3-1：用例图

3.3.2 关键用例

用例 1：添加项目

范围：TestX 测试工具应用
级别：目标用户
主要参与者：用户
前置条件：用户安装了 Java，Node.js ，并安装了 TestX
最小保证：系统正常运行， 不会因为异常终止程序
成功保证： 项目目录中成功添加了用户所选项目

主成功场景：

1. 用户编写测试脚本代码
2. 用户打开 TestX，可以看到工具的基本界面，导航栏以及添加项目按钮， 用户点击添加项目按钮。
3. 用户选择测试脚本所在的目录， 点击确认选择；
4. 用户可以看到项目列表中添加了一个项目， 并且中间显示运行面板以及运行按钮盒项目相关信息；

扩展：

1a. 打开工具界面弹出错误提示框

1a1. 无法启动 selenium-server， server 端口被占用；

1a2. 无法启动 TestX-server， server 端口被占用；

3a. 弹出提示框显示列表中已经有了相同项目， 不能重复添加， 返回 1；

3b. 弹出提示框显示配置文件错误， config.json 配置文件语法错误；

3b1. 更改 config.json 返回 1；

用例 2：测试过程

范围：TestX 测试工具应用
级别：用户目标
主要参与者：用户
前置条件：安装了 Java，Node.js 并已经成功添加项目
最小保证：系统运行正常， 能够提示错误信息
成功保证：用户可以看到运行的过程信息输出， 运行结果输出

主成功场景：

1. 用户选择项目列表中的一个项目
2. 可以看到项目运行面板中的项目信息以及运行按钮
3. 用户点击运行按钮
4. 可以看到运行状态提示不断改变， log 面板不断输出运行状态信息
5. 看到运行测试结果， 以列表的方式展现在面板中。
6. 用户可以拖动滚动条， 查看完整的运行结果报告。

6.1 如果运行成功，运行面板显示为绿色表示成功， 返回 1

6.2 如果运行失败，运行面板显示为红色， 并提示错误信息

6.2.1 用户浏览错误信息， 查看错误栈

6.2.2 点击错误栈中的链接可以在右边 code view 面板中看到错误代码， 并滚动到相应的行

6.2.3 用户修改代码 ， 返回 1

扩展：

2a. 弹出错误信息

2a1. 提示项目已经不存在，并检查项目目录， 返回 1

2a2. 提示配置文件语法错误， 返回 1

3a. 弹出错误提示

3a1. 提示 server 端口被占用， 返回 1

3.4 领域模型

领域模型是 OO 分析中的重要和经典的模型。 它阐述了领域中的重要概念。 本小节结合用例图和用例模型， 设计出如下的领域模型图：

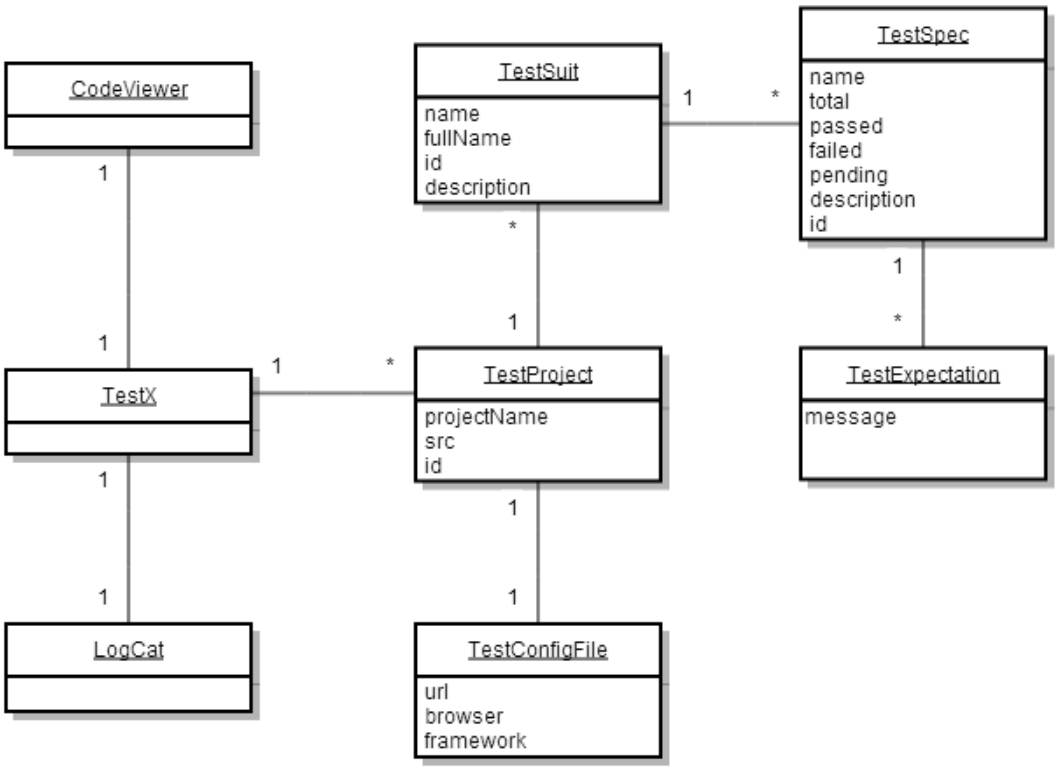

图 3-2：领域模型图

领域模型图分析了在 TestX 中的重要领域概念， 和测试相关的有 test project, test suit, test spec, test expectation, test config file 以及和工具相关的 code viewer， logcat。

第四章　架构设计

4.1 概述

架构设计是系统设计的重要祖层，系统架构设计的优劣将直接营销整个项目的扩展性，和可维护性。本章会主要讲解本项目的系统架构的组成以及各个部分之间的关系， 使用怎样的设计模式来优化系统的架构。

4.2 系统框架

【系统框架图】

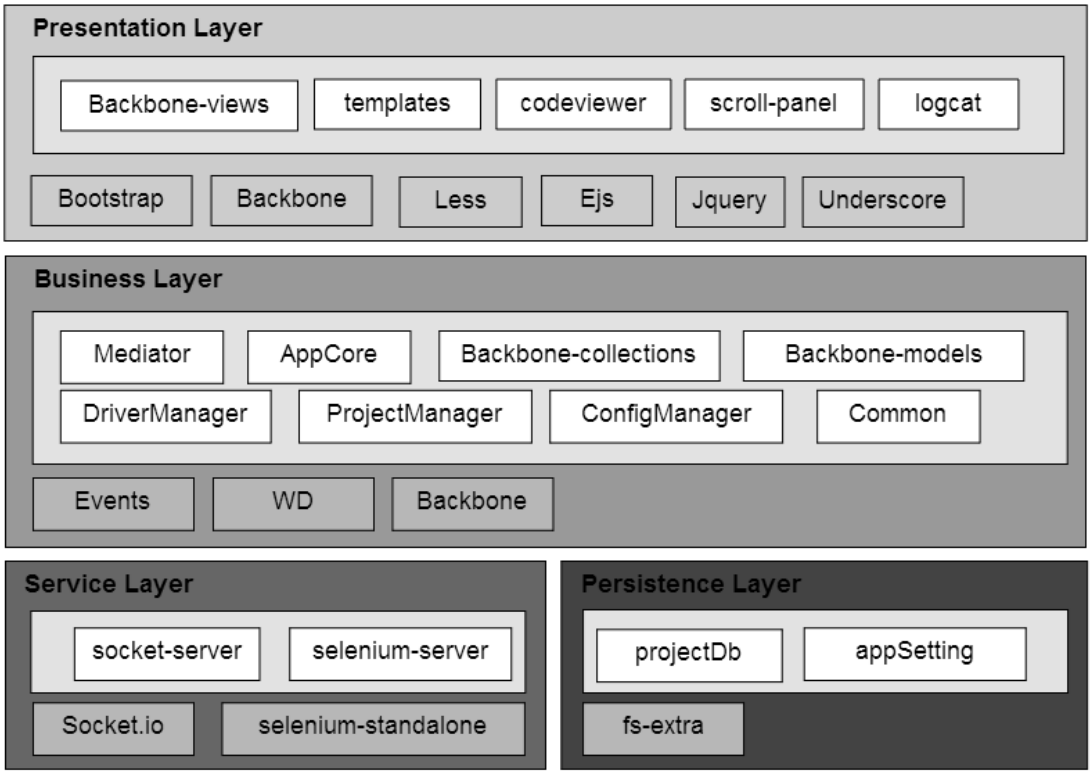

图 4-1：架构图

如图，是本系统的框架图， 系统的架构分为四个层次： 表现层， 业务层， 服务层，持久层。 设计本框架的过程中应用了如下的几点重要技术：中介者模式， MVC 设计模式，pub/sub 模式， http middleware；本小节将重点介绍这些技术在本项目中的应用。

4.2.1 Mediator 模式

在目前的大型前端应用中， 都是采用模块模式开发。 在正常的开发模式中， 各个模块之间会相互引用， 造成了各个模块的重度耦合， 为了解耦各个模块， 我们将采用

mediator 模式， 对于模块之间的消息传递， 将借由 mediator 来传递。 例如有两个模块 A, B。 A 需要将消息传递给 B， 那么 A 需要将消息传递给 mediator， 再由 mediator 将消息传递给 B， 如下图中所示：

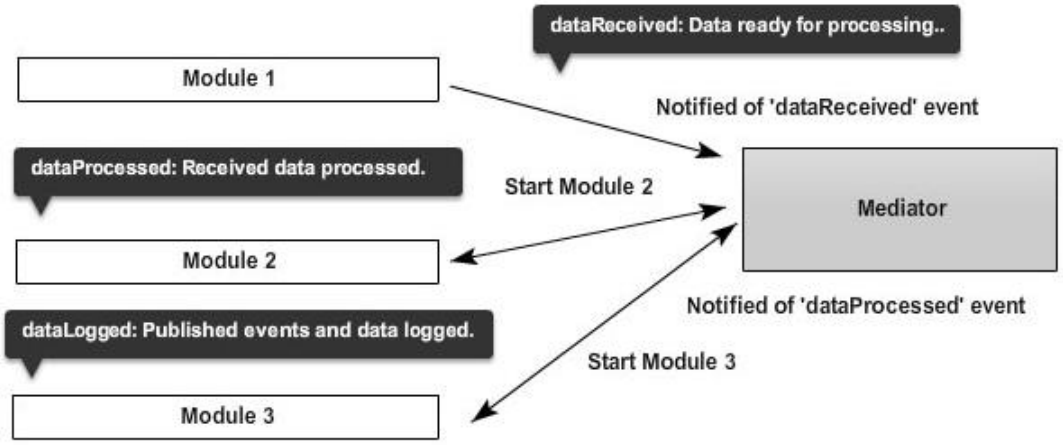

图 4-2：基于 mediator 模式的应用架构

通过 mediator 解耦过后的好处有如下：

1. **各个模块的可重用性提高**。 由于各个模块完全独立， 不依赖于其他模块， 模块可以很容易引入到其他项目中；

2. **容错性提高**；如果其中一个模块出错， 那么错误可以通过 mediator 控制， 不会影响到其他模块的正常运行；

3. **模块具有较高的测试性**；由于模块是独立的，很容易测试模块的功能；

在本项目中， 所有模块都是基于这种 mediator 模式进行开发的。 通过 mediator 将 UI 前端和后端模块分离， 以 main.js 作为应用的核心程序入口。 所有的消息传递都是通过 mediator 来实现。 通过这种机制很好的实现了模块和逻辑解耦。

4.2.2 MVC 设计模式

在本项目中， UI 层通过 Backbone 实现， 在第二章技术与原理中可知 backbone 是一个前端 MVC 框架。 其核心是通过分离逻辑视图以及数据为 Model, View, Collection 这三个类。

在设计过程中，数据相关的类都是通过继承自 Backbone.Model 以及通过使用 Backbone 集合 Collection 来管理数据集合。 项目中的 model 包括 TestProjectModel， TestSuitModel，TestSpecModel，相应的集合有 TestProjectCollection, TestSuitCollection。

视图的控制都是通过继承 Backbone.View, 视图中的渲染是通过 Ejs 模板引擎来实现。应用中的 View 关系和 DOM 的树状结构相似， 顶级 View 持有子 view， 而子 view 有可能持有其他 view。 在本项目中 AppView 是顶级 view。 主要的几个 view 包括 projectListView。PlaygroundView， ConsoleView， CodeViewerView。

4.2.3 Pub/Sub 模式

Pub/Sub 模式也即发布者订阅者模式， 其实本项目中的 mediator 就是一个典型的发布者和订阅者。 通过发布者和订阅者模式可以很好的解耦逻辑。 对于 Backbone 的 Model 和 View 之间的数据同步就是通过 Pub/Sub 模式实现。

如果 model 实例的数据有改变， Backbone 就会自动的发出数据改变的消息。 View 实例如果订阅了这个消息那么就会做相应的改变。 应用到本项目中，测试运行过程的消息状态以及消息结果的出现都是 view 订阅 model 的消息， 然后做通过更新函数做相应的处理。

4.2.4 HTTP Middleware

【引用 professional nodejs】在 Node.js 中, 当创建一个 Http 应用的时候， 通常需要做很多相同的事情， 比如解析 cookie ， 解析查询字符串， 提供静态文件服务， 日志记录。如果把这些都看成一个相互独立的功能， 并且当出现一个 Http 请求的时候， 这些功能会挨着一个一个的执行， 这些功能就叫做 http middleware。 更广泛的定义是：middleware 组件就是程序员可以控制的并且可以像栈一样叠加在一起的程序片段。 通过中间件的抽象，可以对 http 请求处理做很好的扩展。

应用到本项目中： 对于 server 部分， 通过 npm connect 模块提供中间件功能。 并通过中间件提过静态文件服务。 基于用中间件的模式是因为为了提高服务程序的可扩展性，当需要添加一个对 http 请求的处理时， 只需要提供一个 middleware 对象， 对原来的程序不会有任何影响。

4.3 状态图

系统开发过程中， 状态图让开发者能够更加清晰的知道系统中的具体通信情况， 状态图描述了模块间的通信情况， 以及每一个模块的职责和功能。 本系统主要涉及的主要两个状态图分别是：**项目管理状态图**， **测试运行状态图**。

4.3.1 项目管理状态图

项目管理原理：

1. 用户通过 UI 可以发出添加项目请求；

2. 根据用户选择的项目文件夹， 系统会查询其中的配置文件；

3. 解析配置文件并保持项目信息到 db.json 中；

4. 更新 TestprojectCollection；

5. ProjectView 订阅了 model-add 事件，并监听到这个事件，在视图面板中添加项目；

6. 用户通过点击项目的删除按钮可以删除项目；

7. 更新 TestProjectCollection

8. 订阅 model-delete 事件， 将 project 信息重 db.json 中移除；

实现过程中主要涉及到的模块有： **ProjectManager**， **TestProjectCollection**，**TestProjectModel**， **TestProjectView**，**TestProjectItem**。

图 4-3为添加项目的活动图：

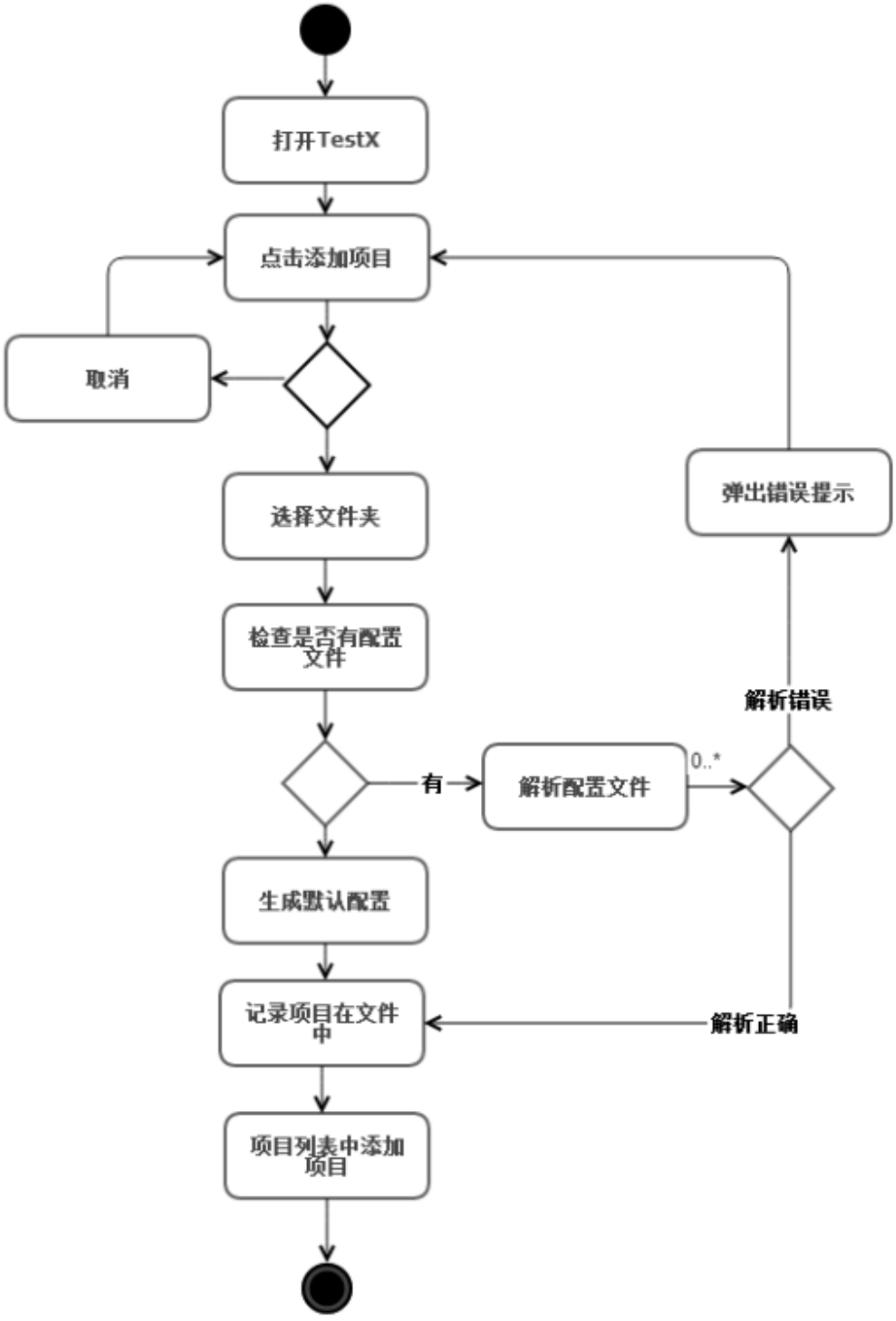

图 4-3：添加项目活动图

图 4-4为添加项目的系统顺序图：

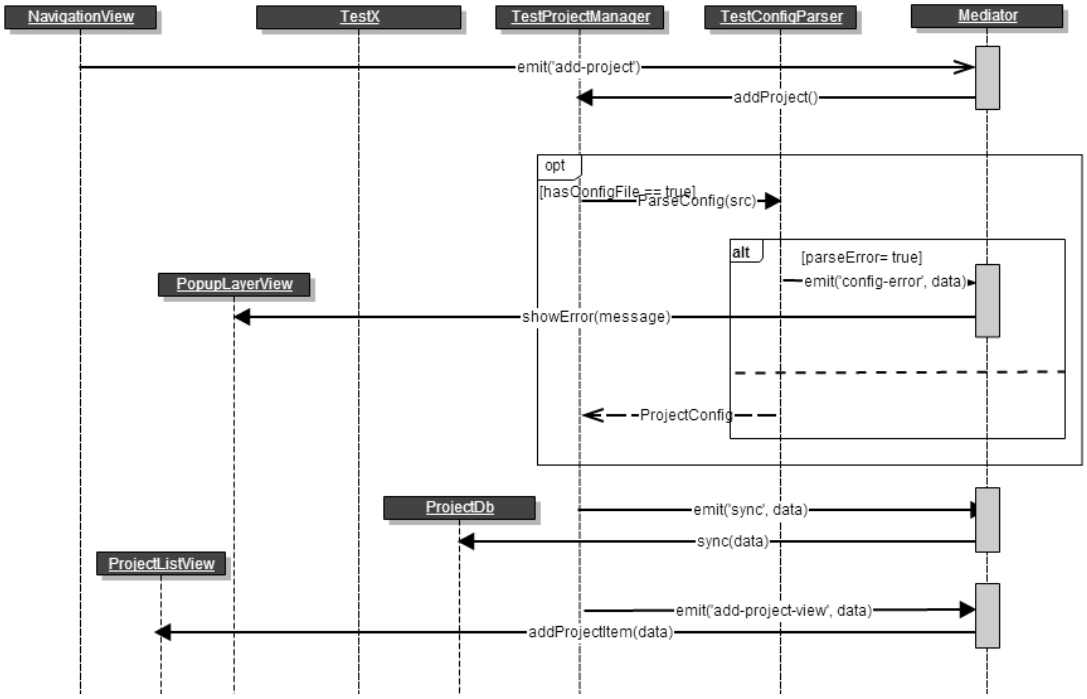

图 4-4：添加项目系统顺序图

4.3.2 测试运行状态图

测试运行过程的原理：

1. 用户选择项目，读取该项目的配置文件；

2. 用户触发运行项目事件， 提示测试开始；

3. 启动测试，通过 ProjectTask 整理运行过程中需要的前置步骤， 读取注入浏览器的 lib 文件， 读取测试文件， 读取测试框架文件， 将这些文件打包合并， 提示测试打包中；

4. webdriver 运行测试， 根据配置初始化浏览器， 打开浏览器， 加载完成后注入测试脚本， 提示测试开始运行；

5. 浏览器中运行测试， 通过 socket.io 同步测试过程和结果信息到 socket server。Socket server 将信息传递给 mediator；

6. Mediator 接受事件以及数据， 并执行信息提示， 以及运行结果的展示；

图 4-5为测试运行的活动图：

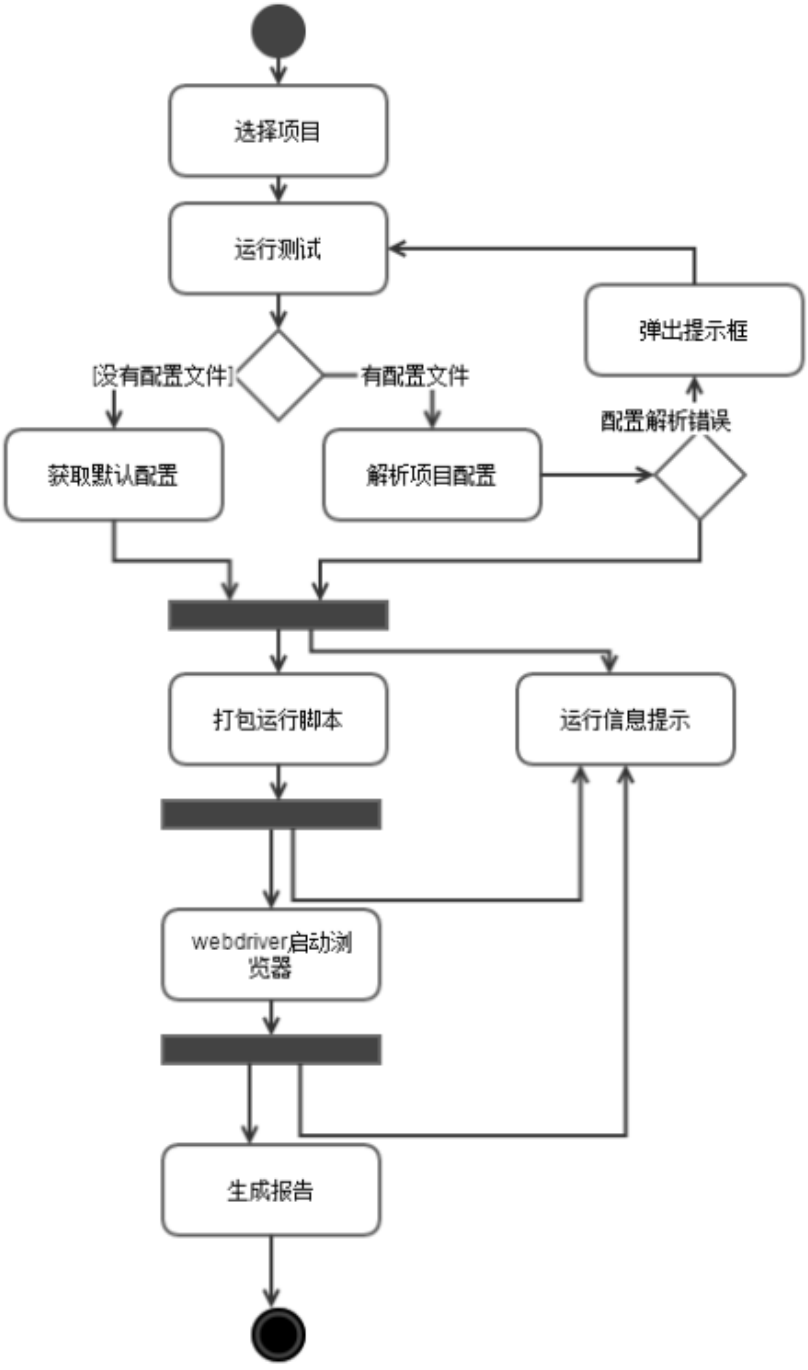

图 4-5：运行测试活动图

图 4-6为运行测试的系统顺序图：

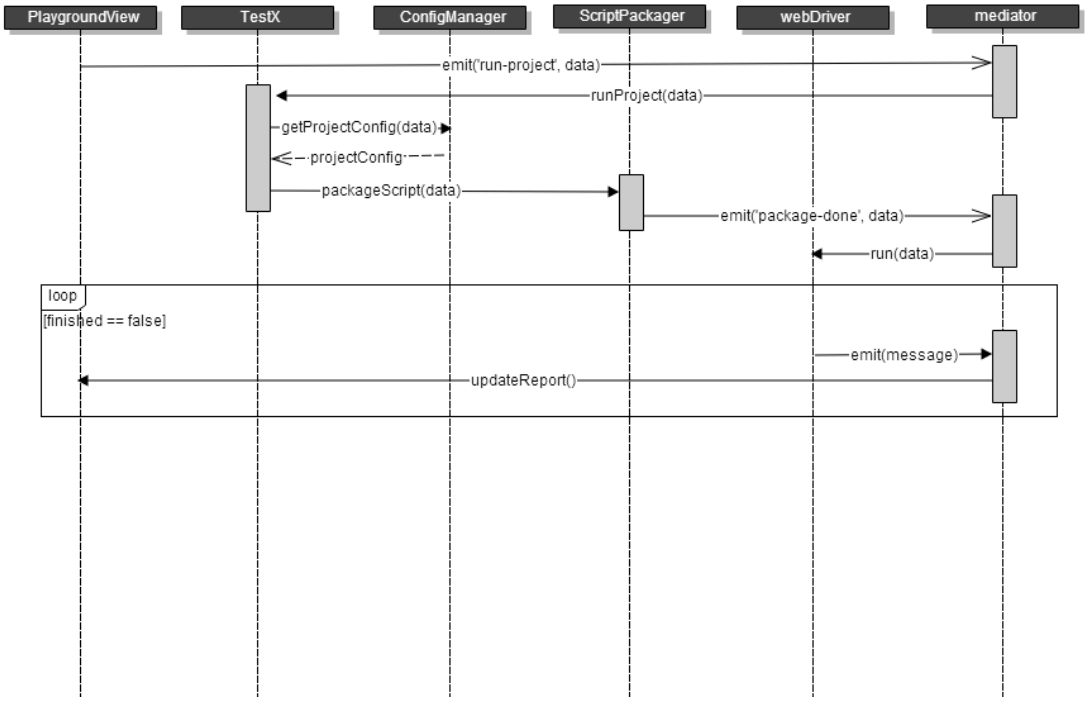

图 4-6：运行测试系统顺序图

4.4 类图

软件设计的一个重要环节就是通过用例图以及领域模型来绘制类图。 类图的直接描述了系统的各个模块之间的关系， 各个类的继承关系， 类中的具体方法和属性。 本小节主要是针对几个重要的类进行类图分析，通过类图以及设计类图使用的设计模式来简单介绍。

4.4.1 类图

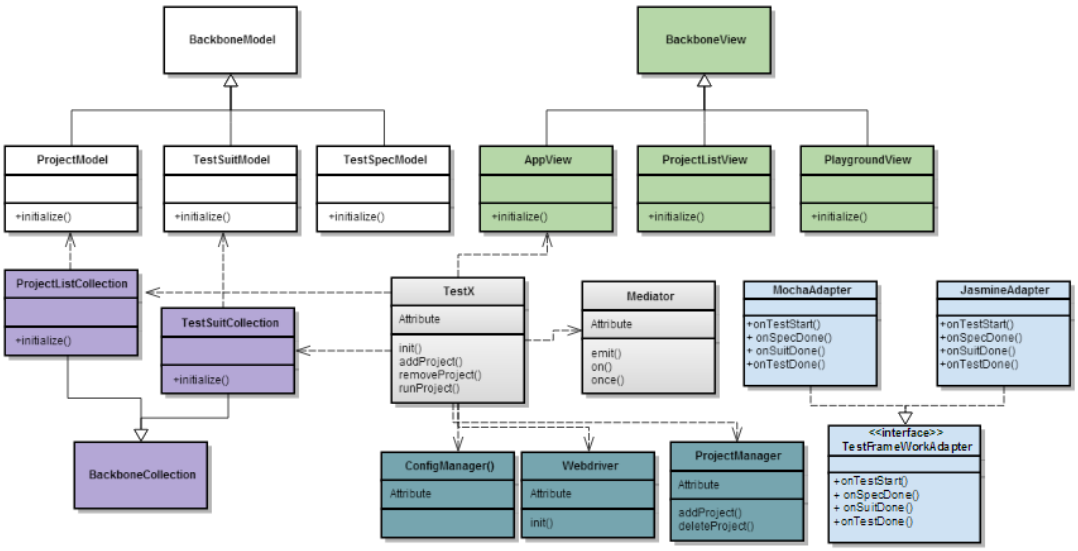

图 4-7：类图部分图

图 4-7显示了该系统的部分类图。 4.4.2 小节将详细的解析上图的类关系图。 本图显示了逻辑层和数据层以及视图层的类关系。

4.4.2 类图分析

在类图中， 画出了主要的视图类， 模型类， 以及核心逻辑控制的类。 其中 model 类全部直接继承自 backbone 的 model， 这些 model 包括了 ProjectModel， TestSuitModel，TestSpecModel。

Model 的持有者是 model 的集合 collection ， 继承自 backbone 的 collection 类， 包括 ProjectListCollection，TestSuitCollection。 对 model 的添加或者删除都通过 collection 来代理完成。 由于 backbone 的 model 和 collection 都继承了 pub/sub 模型。 所以可以在 collection 中监听 model 的数据改变事件， 并通过 collection 统一触发。 这样解耦了外部对 model 的依赖。

可以看到各个模块都相对的独立， 没有因为引用而导致耦合。 在设计中， 应用中介者模式， 模块间的消息传递通过 mediator 来完成。

对于测试框架使用了适配器模式， 由于各个框架所提供的接口不一致， 必须根据不同的框架使用由同一接口的 adapter 来兼容。

4.5 架构设计优缺点

本设计过程中， 通过应用经典的 MVC 模式实现了数据与视图的解耦， 应用大型

javascript 应用开发的 mediator 模式， 实现了模块间的解耦。 通过这两个模式的应用， 可以很容易的实现功能的扩展以及维护。

由于每次启动应用的时候都会启动两个后台服务， 服务会占用系统端口。 当启动两个实例的时候就会导致端口被占用问题， 无法完成测试， 后续的改进方案是提供多实例支持单一服务支持， 启动多个实例， 使用同一个服务。

第五章　模块设计

5.1 概述

本章主要阐述 TestX 的具体模块设计方案， 以及具体的模块设计使用的方法， api 接口设计， 具体会分别阐述： 核心模块， 界面模块， 项目管理模块， 服务模块。 以及会讲述界面的设计。

5.2 模块设计

5.2.1 核心模块

在第四章的架构设计中已经阐述了系统的主要部件以及功能， 核心模块将提供一个供其他模块相互交流的 mediator， 持有各个模块的引用。 具体代码实现在 main.js 中。 Main.js 也是整个应用的入口， 下面将讲解 main.js 的执行步骤。

1. **启动应用初始化函数**；

2. **共享 node 环境和 webkit 环境的公用变量；**包括 backbone，jquery，underscore 以及其他的必要对象； 由于 node-webkit 的特殊性， 在 node 环境如果要访问 webkit 的 dom 必须要传入 window 或者 document 的引用到 node 执行环境中。 此方法为 shareContext();

3. **初始化中介者**；

4. **错误控制**；在 node 环境中， 如果出现意外错误将导致应用崩溃， 解决方式是通过主动监听错误事件，并在代码中主动处理错误。

5. **启动后台 server**； 通过 spawn 的方式启动一个子进程服务， 而不是直接在应用中启用， 避免因为提供服务导致界面处于假死状态；

6. **初始化事件监听**；

7. **初始化 ui**；

5.2.2 ui 模块

在应用中将 ui 独立为单独的模块， 所有和界面相关的操作都将掌控在 ui 模块内， 外部对界面的控制所需要的只是引用 ui 模块。

ui 模块中主要是利用了 backbone 的 view 以及 ejs 模板， 系统的界面相关的结构都是放在 template 目录下面的 ejs 文件。 利用模板的好处除了对模板内提供赋值变量例外， ejs 模板可以通过 include 的方式引用子模板。 这样极大的提高了模板的重用性以及模板的解耦。 下面为这些 template 的目录结构：

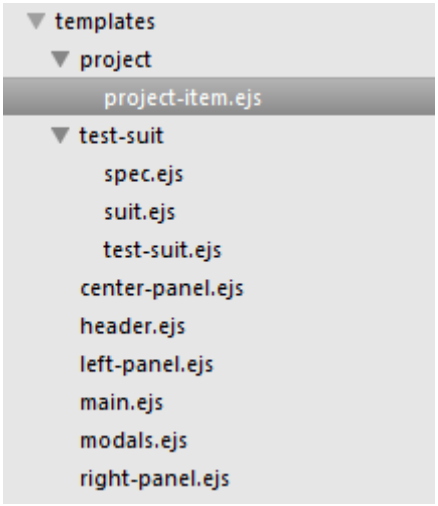

图 5-1：模板目录结构图

而使用 backbone 的 view 的好处就是讲界面看成由不同的模块相互组合而成的应用。每个模块有自己的独立事件处理。 在设计过程中整个应用是一个 AppView， AppView 细分为 NavBarView， ProjectListView, PlaygroundView。 而每个细分的子 View 又包含自己的子 View。

ui 部分的结构可以看下图：

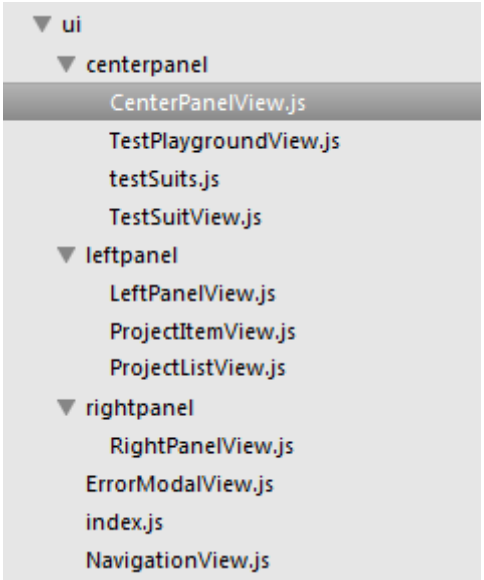

图 5-2：ui 代码目录结构图

5.2.3 项目管理模块

应用提供了以项目的方式运行测试， 所以为了更好的管理项目相关的逻辑处理。 将项目模块分离出来。

项目模块的目录结构为：

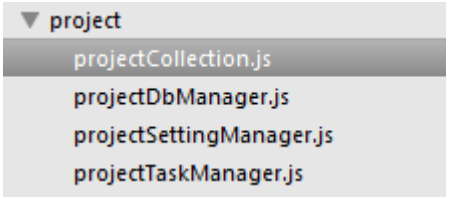

图 5-3：项目模块目录结构图

ProjectCollection 继承自 backbone 的 collection， 是 projectModel 的集合。 由于 project 需要存储在磁盘当中， 以 json 格式存储， 这里通过 projectDbManager 来实现对 project 数据的同步以及更新。 在 projectDbManager 中也只有两个方法：

```
 9  function save(data, handler, thisObj) {
17  }
18
19  function fetch(handler, thisObj) {
27  }
```

图 5-4：projectDbManager 代码

ProjectSettingManager 提供管理项目配置文件相关的逻辑处理， ProjectTaskManager 则提供项目运行过程中的运行任务逻辑处理。

5.2.4 服务模块

启动浏览器运行测试的过程中， 为了实时同步浏览器和系统的状态和信息， 需要利用 socket.io， 这时必须要在 node 上下文中启动一个 socket server。 而这里的服务模块就是这个 socket server 的实现模块。

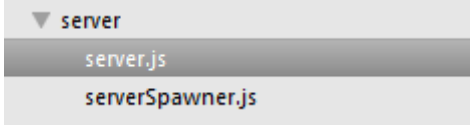

图 5-5：server 模块目录结构图

由于在 node-webkit 环境下， webkit 和 node 在同一个事件循环中， 这就意味着在处理逻辑的过程中不能渲染 ui， 在提供 socket 的服务的过程中会造成 ui 假死状态。 而解决方法是通过在 node 中 spawn 一个子进程来运行 server。 这里的 serverSpawner 就是为了完成

这一任务而设计的模块。

通过 spawn 方法来运行的子进程另一个好处就是主进程可以和子进程实现双向通信。当 server 接受到浏览器发来的 socket 信息过后， server 将数据发送给父进程。 这里 serverSpawner 会接受来自 server 的信息， 再将这些信息传递给 mediator。

5.3 界面设计

当第一次打开 TestX 的时候会看到如下界面：

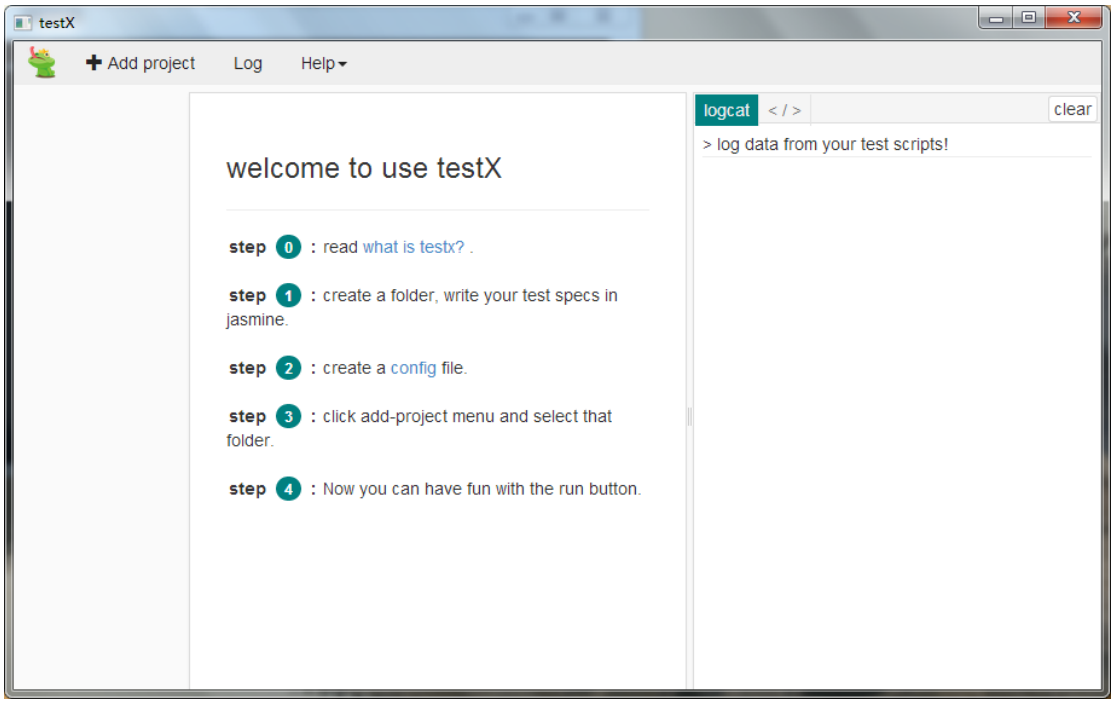

图 5-6：开始界面图

点击添加项目过后的结果：

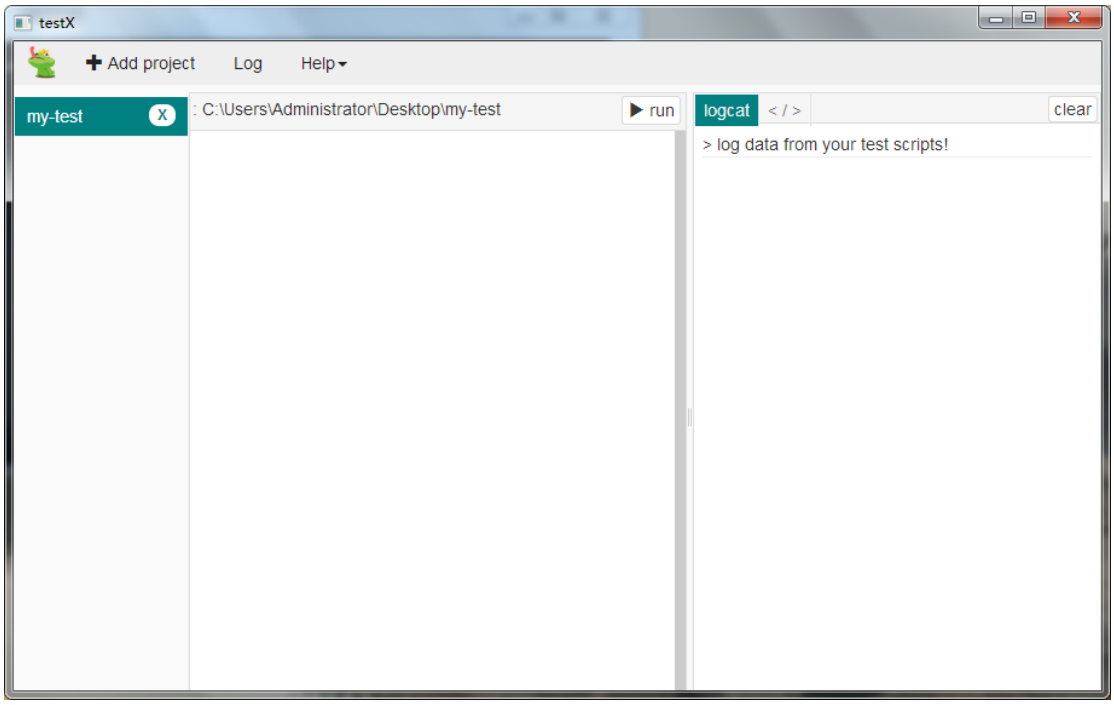

图 5-7：添加项目后

My-test 内编写的测试是 jasmine 的 sample 测试脚本， 运行测试过后的结果：

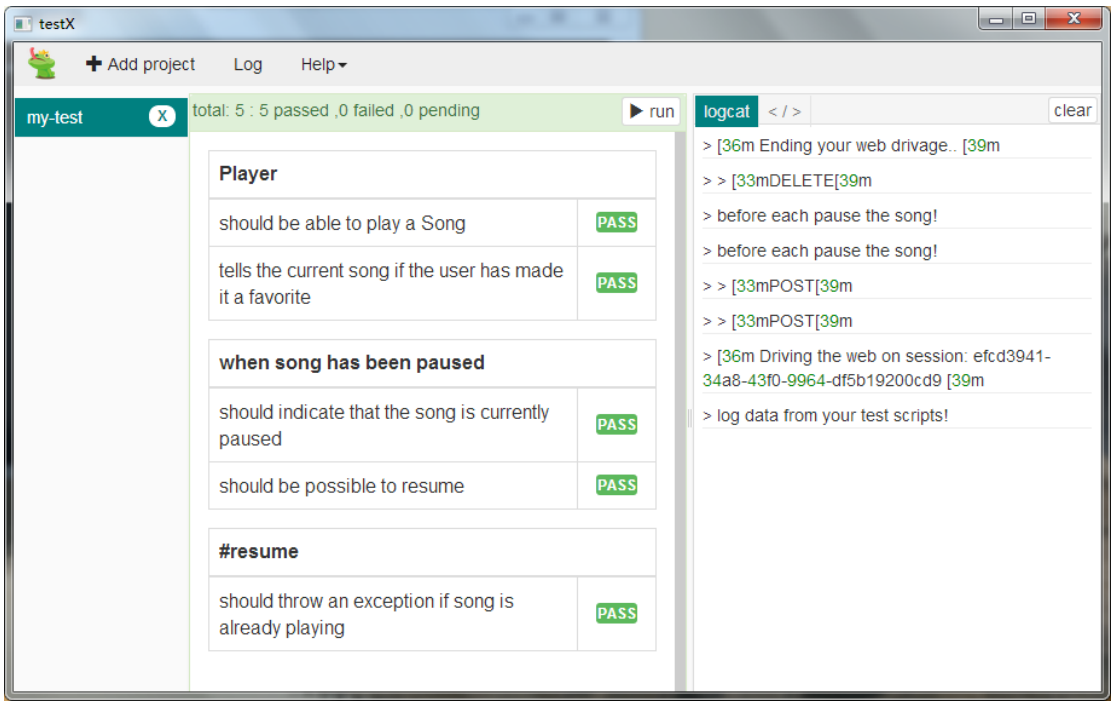

图 5-8：运行测试成功

修改脚本， 当出现错误的情况是如下：

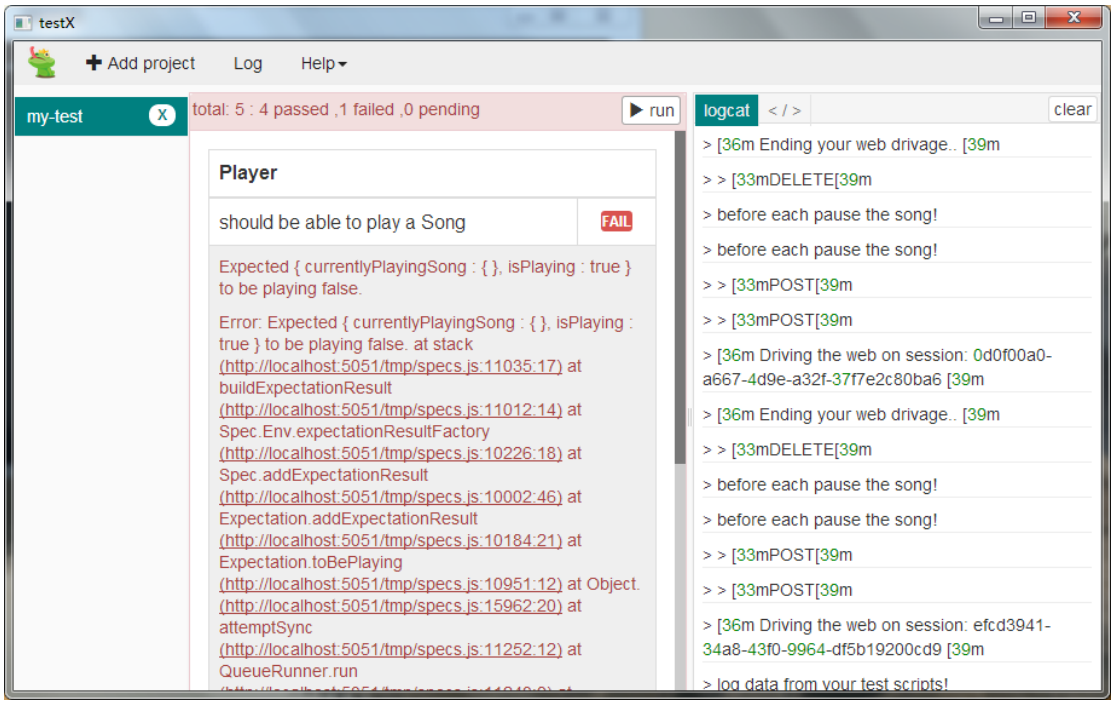

图 5-9：运行测试失败

查看具体的代码错误位置：

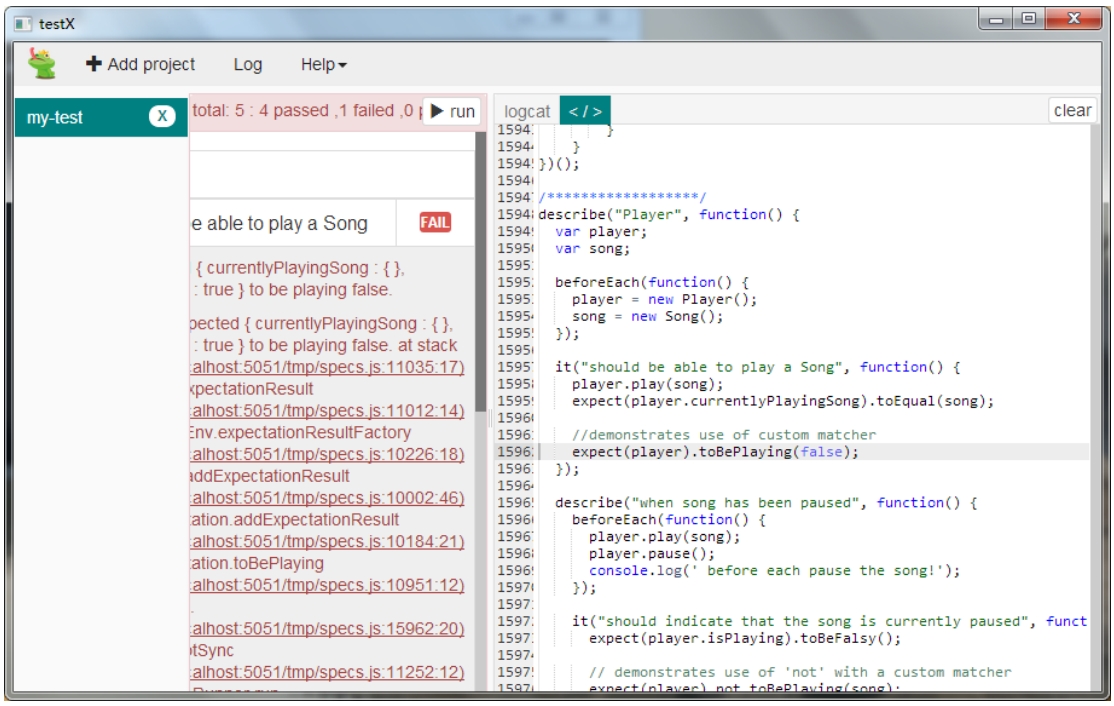

图 5-10：查看错误行代码

第六章　部署与应用

6.1 概述

本章主要论述 TestX 的配置安装方法， 以及运行流程， 以提供使用者部署方法。

6.2 部署环境

项目名称： web 前端测试工具开发
项目开发环境：node-webkit，nodejs
项目部署环境： java， nodejs， phaontomJs
项目部署方法： 直接运行 TestX 的可执行文件

6.3 系统运行流程

在环境已经部署好的情况下， 可以按如下步骤运行系统：

1. 建立一个 test 目录， 可以放在应用项目目录下， 也可以是任意其他位置。
2. 在 test 目录下添加配置脚本 config.json， 包括需要测试的应用链接， 使用的测试框架， 以及测试的浏览器等配置（可选）；
3. 添加测试脚本， 测试脚本的编写过程中可以直接引用应用中的全局变量；
4. 执行 TestX 可执行文件；
5. 第一次打开 TestX 会提示使用帮助，通过查看帮助可以了解更多关于工具的使用；
6. 点击添加项目按钮， 将刚才的 test 目录添加进来；
7. 点击运行按钮就可以启动测试了；
8. 查看测试结果， 并更具结果不断修改和添加测试脚本， 回到第三步骤；

第七章　总结

7.1 概述

在项目的完成过程中， 涉及到了相当广的知识面， 重前端知识到后台开发以及测试领域， 重分析研究到软件工程。 本论文的主要目的是需要解决前端测试复杂以及前端交互测试这两个问题。 在本章中， 将结合本文对这两个问题进行总结和分析。

7.2 研究成果

7.2.1 解决前端测试开发的复杂性

在第一章的背景中已经提到， 前端测试驱动开发面对的主要问题就是测试开发的复杂度。 本文针对这个问题作了如下解决方案：

1. 将测试代码与应用代码完全的分离， 通过 url 作为测试代码与应用代码的桥梁；
2. 通过提供界面化的工具管理测试项目以及简化测试运行过程；
3. 可以通过配置在 chrome， firefox， ie 或者 phantomJs 等环境中运行；

7.2.2 解决前端交互测试的问题

同样，前面已经讲述了前端开发过程中的另外一个难题，交互代码和逻辑代码的混合。前端测试更多是处于 ATDD 和 TDD 之间， 那么为了解决这个问题作了如下解决方案：

1. 提供前端事件模拟对象 elf， 通过这个对象可以完成 click， mousemove，mouseover， keydown， keyup 等事件的模拟；
2. 提过模拟方法的语法糖果。 模拟方法基于 promise 的异步队列来完成， 可以通过链式的方法调用， 也可以分开调用。
3. 通过结合 elf 和测试框架的异步功能，可以很容易的实现前端的异步动作测试。

7.3 开发中的不足

TestX 目前还只是处于原型阶段， 能够提供最简的可应用方案， 但是还有许多地方需要完善， 包括的问题有：

1. **目前还没有完成的 spy 功能；** 前端开发过程中为了更深入了解函数的执行情况，通过提供 spy 来监控函数的执行，这一功能不仅仅提供给测试框架，在任何 debug 的过程中都需要。 对于目前的测试框架提供的 spy 功能有很多不完善的地方， 本项目针对这一问题参考 weinre 以及 spyjs 做相应的解决方案。 目前只是完成一部分。

2. **测试框架支持不完善；** 为了将更多的精力放在测试功能的完善以及完整性上， 对于测试框架的支持目前还只是支持 jasmine， 未来会提供更多的框架支持；

3. **应用启动过程的错误捕获；**由于 webdriver 运行浏览器的过着中， 需要等待浏览器启动过后才能往浏览器中注入脚本。 这意味着如果之前的应用有错误代码那么测试将无法执行。

致谢

在本文撰写过程中， 感谢室友提供的莫大帮助， 以及感谢 github 上的各种开源项目和相应的贡献者，本文的很多思想都是借鉴于这些开源项目。

以及感谢中山大学软件学院四年的培养教育， 感谢天猫公司以及导师在实习过程中的耐心指导。

参考文献

[1] Lasse Koskela, 测试驱动开发的艺术（第 1 版），人民邮电出版社，2010

[2] Crag Larman，UML 和模式应用（第 3 版）， 机械工业出版社， 2009

[3] Pedro Teixeira, Professional Node.js, John Wiley & sons, Inc, 2013

[4] http://addyosmani.com/largescalejavascript/ , Addy Osmani

[5] https://github.com/rogerwang/node-webkit/wiki node-webkit wiki

[6] http://nodeapi.ucdok.com/ Node.js 官方文档

[7] http://docs.seleniumhq.org/ selenium 官方文档

[8] https://github.com/admc/wd wd.js wd.js 项目地址

[9] http://pm163.lofter.com/post/aa404_197d8e phantomJs 介绍

[10] http://backbonejs.org/ backbone 文档

附表一、毕业论文开题报告

论文（设计）题目：web 前端测试工具开发

目的：

在实习期间，了解现在企业对于 web 前端测试的极大需求，然而并没有合适的工具可以应对解决。其主要原因在于前端需要应对快速变化的需求以及传统的测试方法很难应对交互型的前端代码。 构建一个能够快速应对需求变化，以及解决交互异步的测试工具很有必要，本文旨在研究如何构建这样一个测试工具。

思路：

对于此文描述的测试工具， 其核心需求有两点， 第一是应对快速变化的需求， 第二是应对交互异步，那么该文的核心也是旨在如何解决这两个问题。对于第一个需求，大多数因为写测试的成本太高而放弃写测试脚本， 那么此文要阐述的找到简化写测试脚本成本的方法。 对于第二个需求，前端可以实现一个 robot 来代理完成交互。

方法：通过研究现有的测试工具， 尝试使用并发现其中的不足之处提出改进，以及按照上面提到的两个问题为核心提出相应的解决方案。

支持条件：

1. 本人在天猫实习岗位为前端工程师，对前端比较熟悉。
2. 因为天猫前端也亟需这样的一个工具，所以能够得到公司一定的支持。

进度安排：

2013 年 11 月下旬：	设计完成的项目实现方案
2013 年 12 月初：	学习相应的实现工具以及框架
2013 年 12 月下旬以后:	开始着手项目开发
2013 年 1 月：	结合完成测试工具原型
2013 年 2 月：	完成工具第一版本
2013 年 3 月：	完成论文初稿

学生签名：陈学家　　年　月　日

指导教师意见：

1、同意开题（ ） 2、修改后开题（ ） 3、重新开题（ ）

指导教师签名：　　年　月　日

附表二、毕业论文过程检查情况记录表

指导教师分阶段检查论文的进展情况（要求过程检查记录不少于 3 次）：

第 1 次检查

学生总结：

指导教师意见：

第 2 次检查

学生总结：

指导教师意见：

第 3 次检查

学生总结：

指导教师意见：

第 4 次检查

学生总结：

指导教师意见：

学生签名：　　年　　月　　日

指导教师签名：　　年　　月　　日

总体完成情况	指导教师意见： 1、按计划完成，完成情况优（ ） 2、按计划完成，完成情况良（ ） 3、基本按计划完成，完成情况合格（ ） 4、完成情况不合格（ ） 指导教师签名：　　年　　月　　日

附表三、毕业论文答辩情况

答辩人		专 业	
论文（设计）题目			
答辩小组成员			
答辩记录：			
记录人签名：		年 月 日	

学术诚信声明

本人所呈交的毕业论文，是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料均真实可靠。除文中已经注明引用的内容外，本论文不包含任何其他人或集体已经发表或撰写过的作品或成果。对本论文的研究作出重要贡献的个人和集体，均已在文中以明确的方式标明。本毕业论文的知识产权归属于培养单位。本人完全意识到本声明的法律结果由本人承担。

本人签名：　　日期：