Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Санкт-Петербургский государственный университет аэрокосмического приборостроения»

КАФЕДРА № 52

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доц., к.т.н. |  |  |  | Марковская Н.В. |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ |
| ТЕСТИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЙ |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 5321 |  |  |  | Аверьянов С. С. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2016

**Общая характеристика предприятия**

**ПАО "Интелтех" - одно из ведущих российских предприятий, специализирующееся на проведении научных исследований, разработке и производстве оборудования для автоматизированных систем управления и связи.**

**На протяжении десятилетий предприятие накапливало опыт создания телефонной, телеграфной, фототелеграфной техники, средств передачи данных и автоматизации управления гражданского и военного назначения.**

**В настоящее время ПАО "Интелтех" является поставщиком программно-технических комплексов для автоматизированных систем управления и обмена данными, цифровых сетей конфиденциальной телефонной связи и защищенных инфокоммуникационных систем для правительственных, силовых и гражданских ведомств России. Разработанное предприятием оборудование установлено в штабах и на объектах этих ведомств.**

**Комплексное использование мультисервисного оборудования ПАО "Интелтех" обеспечивает заказчикам возможность реализации гибкой стратегии развития ведомственных систем управления и связи на основе принципов конвергенции и поэтапного внедрения новых сетевых технологий и расширения спектра информационных услуг.**

**В соответствии с мировыми тенденциями развития информационных и связных технологий предприятие ориентируется на создание универсальной телекоммуникационной техники нового поколения, решающей задачи системной и сетевой интеграции и обеспечения безопасности информационного обмена.**

**Выпускаемое предприятием оборудование характеризуется использованием новейших аппаратных и программных платформ, высокой надежностью и безопасностью, компактностью и малым энергопотреблением, что соответствует современному уровню требований. Конструкторские решения позволяют использовать оборудование в необходимом заказчикам диапазоне внешних условий.**

**Постановка задачи**

**Веб-приложение — клиент-серверное приложение, в котором клиентом выступает браузер, а сервером — веб-сервер. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются кроссплатформенными сервисами.**

**В нашем случае веб-приложение создано на основе фреймворка AngularJS. Задачей является проверка правильности работы данной программы методами тестирования программного обеспечения.**

**AngularJS изначально спроектирован для создания одностраничных веб приложений. Таким образом браузер откроет только одну страницу и дополнительный контент будет загружаться по мере необходимости. При таком подходе экономится траффик и уменьшается нагрузка на сервер.**

**Тестирование программного обеспечения — процесс исследования, испытания программного продукта, имеющий две различные цели:**

* **продемонстрировать разработчикам и заказчикам, что программа соответствует требованиям;**
* **выявить ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации.**

**E2E (end to end) тесты** - это тесты пользовательского веб-уровня. Для их запуска необходимо наличие веб-сервер. Они работают для сравнения результатов при запуске браузера, и выполнении пользовательских действий, таких как, ввод в поле текста, нажатие на кнопки, изменение URL браузера и т.д. У них отсутствует доступ к коду приложения Angular. Данные тесты работают с уже сформированной страницей html и отличаются низкой скоростью.

**Unit (модульные) тесты** - это тесты уровня кода, в которых тестируется минимально возможный для тестирования компонент, например, отдельный класс или функция. С помощью них тестируются основные компоненты Angular приложений. Это изолированные от всего кода тесты, которые требуют установленного пакета karma и запускаются через консоль. Перед тестированием компонента необходимо внедрять зависимости, с которыми требуется работать. Эти тесты гораздо быстрее чем E2E тесты, т.к. для их тестирования не нужно запускать браузер и формировать страницу.

Unit тестирование

Для выполнения unit-тестов для Angular приложений можно использовать модуль NPM называемый karma (ранее testacular). Он позволяет запускать и проверять тесты прямо из консоли. Тесты пишутся с помощью Jasmine - BDD фреймворка (Behavior-Driven Development — Разработка на Основе Поведений) для тестирования JavaScript кода.

Основными ключевыми словами при работе с Jasmine являются:

* **describe** — определение набора тестов (наборы могут быть вложенными);
* **it** — определение теста внутри любого набора тестов;
* **expect** — определяет ожидания, которые проверяются в тесте.

Ключевые слова describe и it являются обычными вызовами функций, которым передаются два параметра. Первый — название группы или теста, второй — функция содержащая код.

Для того чтобы отключить выполнение набора тестов или конкретного теста, необходимо воспользоваться ключевыми словами xdescribe и xit соответственно.

Для того чтобы избежать повторения при создании и удалении объектов и загрузки фикстур, необходимых для выполнения тестов, используются функции beforeEach и afterEach. Они запускаются перед и после каждого теста в наборе соответственно.

Jasmine поддерживает тестирование асинхронных вызовов с помощью функций runs и waitsFor:

* **runs** — принимает асинхронную функцию для выполнения;
* **waitsFor** — принимает первым параметром функцию, которая должна вернуть true, если асинхронный вызов сделанный в runs был выполнен, второй параметр — сообщение об ошибке, третий — время ожидания в миллисекундах.

E2E тестирование

E2E тесты выполняются с помощью утилиты Protractor, которая запускает тесты в браузере, тем самым позволяя эмулировать взаимодействие пользователя с веб-приложением. Для написания тестов может быть использован Jasmine. Таким образом синтаксис e2e тестов аналогичен синтаксису unit-тестов.

Protractor добавляет несколько глобальных переменных, помогающих в написании тестов:

* protractor – namespace-оболочка протрактора, которая содержит статические вспомогательные переменный и классы;
* browser – оболочка вебдрайвера, используется для навигации и получение информации о странице;
* element – вспомогательная функция для нахождения и взаимодействия с элементами;
* by – коллекция стратегий поиска элементов (ccs selector, id, binding attribute).

Основные методы-помошники:

* browser.get(targetUrl) – переход на указанный URL;
* element(by.css(‘.error’)) – выбор элемента по css;
* element(by.model(‘modelName’)) – выбор элемента по модели;
* element(by.binding(‘variableName’)) – выбор элемента по биндингу (ng-bind или {{variableName}});
* element.all(by.repeater(‘item in items’)); – выбор списка элементов из ngRepeat;
* element(by.model(‘modelName’)).getText() – получение текстового значения;
* element(by.model(‘modelName’)).getAttribute(‘id’) – получение значение аттрибута;
* element(by.model(‘modelName’)).sendKeys(‘Some text’) – задание значения;
* element.all(by.repeater(‘item in items’)).count() – получение количества элементов в списке;
* element.all(by.repeater(‘item in items’)).get(1) – получение одного элемента из списка;
* element.all(by.repeater(‘item in items’)).first() – получение первого элемента из списка;
* element.all(by.repeater(‘item in items’)).last() – получение последнего элемента из списка;
* element.all(by.repeater(‘item in items’)).row(1).column(‘title’) – получение значение title из 2й строки;
* browser.isElementPresent(by.model(‘modelName’)) – проверка наличия элемента;
* $(‘.info’) – короткий алиас к element(by.css(‘.info’));
* $$(‘option’) – короткий алиас к element.all(by.css(‘option’));

Установка программного обеспечения

Для начала работы и написания тестов необходимо установить программное обеспечение, предназначенное для тестирования веб-приложений.

**Git** — распределённая система управления версиями. Актуальная версия доступна на официальном сайте по ссылке: <https://git-scm.com>.

**Node** или **Node.js** — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Требуется для установки программных пакетов, таких как karma. Актуальная версия доступна на официальном сайте по ссылке: <https://nodejs.org/en>.

**npm** — это пакетный менеджер node.js. С его помощью можно управлять модулями и зависимостями. Данный пакетный менеджер может устанавливать программные пакеты как глобально (пакет будет доступен для всех приложений), так и локально. По умолчанию npm будет устанавливать все пакеты в локальном каталоге, в которым вы сейчас работаете.

**Karma** – инструмент на node.js, позволяющий запускать JavaScript код в нескольких браузерах. Данный инструмент используется для проведения unit-тестирования. Установка происходит с помощью менеджера npm, для чего используется следующая команда:

npm install -g karma

**Jasmine** – фреймворк для тестирования JavaScript кода. Устанавливается также с помощью менеджера npm командой:

npm install karma-jasmine

**karma-chrome-launcher** – программный пакет, осуществляющий подключение karma к браузеру (в данном случае Chrome). Установка производится следующей командой:

npm install karma-chrome-launcher

Для запуска karma требуется файл конфигурации, в котором будут указаны все настройки запуска. Можно создать файл конфигурации с помощью мастера инициализации, воспользовавшись командой:

karma init

Основные моменты конфигурационного файла karma.config.js (в скобках указаны значения по умолчанию):

* basePath('') – путь к директории веб приложения
* files([]) - список файлов/шаблонов для загрузки
* exclude([]) - список исключений для предыдущего пункта
* reporters(['progress']) - вариант вывода прогресса
* port(8080) - порт веб-сервера(testacular)
* runnerPort(9100) - порт клиента
* colors(true) - включение/выключение цветов при выводе
* logLevel(LOG\_INFO) - LOG\_DISABLE|LOG\_ERROR|LOG\_WARN|LOG\_INFO|LOG\_DEBUG
* autoWatch(false) - выполнение тестов при изменении файлов
* browsers([]) – используемый браузер: Chrome, ChromeCanary, Firefox, Opera, Safari, PhantomJS
* captureTimeout(5000) - задание таймаута в миллисекундах
* singleRun(false) - для одноразового запуска
* preprocessors({}) - список обработчиков, которые будут применены к файлам, до загрузки в браузер, например - CoffeeScript

**Bower** — менеджер пакетов для клиентского javascript. Его отличие от npm заключается в том, что не навязывает пользователю свою систему сборки, а разработчику пакетов — метод подключения библиотеки (AMD, CommonJS и другие). Данный менеджер пакетов загружает файлы нужных библиотек и всё, что нужно для их работы в специальную папку. Установка может быть выполнена с помощью npm следующей командой:

npm install -g bower

Для работы тестов также потребуется библиотека **angular-mocks.js**, которая содержит mock-объекты для встроенных сервисов AngularJS. С их помощью, например, можно тестировать работу сервиса $http, не отправляя реальных запросов на сервер. Установка происходит с bower. Соответственно команда для установки данной библиотеки следующие:

bower install angular-mocks

Библиотека после выполнения данной команды будет расположена по адресу: bower\_components/angular-mocks/angular-mocks.js. Она должна быть включена в конфигурационном файле в список фалов для загрузки.

После того как все необходимые компоненты будут установлены, запуск утилиты karma осуществляется выполнением следующей команды:

karma start

После этого все тесты будут запускаться в соответствии с настройками, указанными в конфигурационном файле. Например, после внесения изменений в один из файлов karma выполнит все тесты, что позволит оперативно проверять правильность работы приложения.

Для удобства тестирования веб-приложения имеется возможность подключения karma к программе WebStorm.

**Protractor** – e2e тест-фреймворк, который запускает тесты в реальном браузере. Его установка выполняется глобально с помощью менеджера npm командой:

npm install -g protractor

После этого потребуется обновить **selenium server**, который будет выполнять e2e тесты:

webdriver-manager update

Для protractor также требуется файл конфигурации, для которого существуют следующие основные параметры:

* allScriptsTimeout – таймаут для выполнения всех сценариев;
* specs – пути к файлам сценариев тестов (относительно конфигурационного файла);
* exclude – исключения для предыдущего пункта;
* capabilities – выбор браузера с параметрами;
* multiCapabilities – опция для запуска тестов в нескольких браузерах;
* baseUrl – стартовая страница приложения;
* rootElement – элемент на котором инициализировано приложение;
* onPrepare – функция, которая будет выполнена, когда protractor готов к работе, но тесты еще не начали выполняться;
* params – параметры, которые будут внедрены в среду выполнения тестов;
* framework – фреймворк, который используется для тестирования (jasmine, cucumber, mocha);
* onCleanUp – функция, которая будет вызвана после выполнения всех тестов.

Запуск protractor осуществляется выполнением команды:

protractor conf.js

В данном примере conf.js является файлом конфигурации. Если файл конфигурации называется protractor.conf.js и лежит в текущей директории, то запуск может быть осуществлен командой:

protractor

После команды запуска будет открыт веб-браузер, в котором protractor осуществит выполнения всего набора e2e тестов, после чего браузер будет закрыт, а результаты отобразятся в окне консоли.

При установке программных пакетов таким образом требуется подключение к интернету. Существует способ их установки на компьютеры без подключения к интернету. Для этого требуется с помощью пакетного менеджера установить интересующий пакет на компьютер с интернетом, после чего будет сохранен кэш для установки этого пакетного модуля в папке

C:\Users\%username%\AppData\Roaming\npm-cache

Требуется скопировать содержимое данной папки на компьютер без подключения к интернету. Далее при установке всех пакетов требуется указать параметр ‑‑cache‑min 99999999. Например:

npm --cache-min 99999999 install -g karma

Таким образом установка будет произведена из кэша.

Пример unit теста

describe('app.components.resources module', function() {  
  
 describe('service', function(){  
 var resourceService;  
  
 *beforeEach*(module('app.components.resources'));  
  
 *beforeEach*(function() {  
 inject(function($injector) {  
 resourceService = $injector.get('resourceService');  
 });  
 });  
  
 *it*('should be defined', function() {  
 *expect*(resourceService).toBeDefined();  
 });  
  
 *it*('should translate', function() { *expect*(resourceService.translate('connecting')).not.toEqual('Не поддерживается');  
 *expect*(resourceService.translate('random string is not in the list')).toEqual('Не поддерживается');  
 });  
 });  
});

Данный тест проверяет наличие сервиса resourceService в приложении и тестирует работу его метода translate().

Пример e2e теста

describe('app', function() {  
  
 *beforeEach*(function() {  
 browser.ignoreSynchronization = true;  
 });  
  
 describe('index page', function() {  
 *beforeEach*(function() {  
 browser.get('/');  
 });  
   
 *it*('should render h3 on index page', function() {  
 browser.sleep(5000);  
 browser.waitForAngular();  
 *expect*(*element*.all(*by*.css('h3.text-center')).first().getText()).toMatch(/Мнемосхема/);  
 });  
   
 *it*('should go to elements page', function() {  
 browser.sleep(5000);  
 browser.waitForAngular();  
 *element*.all(*by*.css('.btn-group .btn.btn-default')).first().click();  
 *expect*(browser.getLocationAbsUrl()).toMatch("/elements");  
 });  
 });   
});

Данный тест проверяет загрузку главной страницы и тестирует переход на страницу /elements по нажатию на кнопку в браузере.

Список литературы:

<https://ru.wikipedia.org/wiki/Тестирование_программного_обеспечения> - Тестирование программного обеспечения

<http://shustov.su/blog/javascript/e2e-and-unit-tests-for-angular-js> - Пишем E2E и Unit тесты для Angular JS

<https://habrahabr.ru/post/167173/> - Введение в Jasmine

<https://habrahabr.ru/post/169699/> - Jasmine — дополнительные возможности

<https://habrahabr.ru/post/133363/> - Шпаргалка по пакетному менеджеру NPM

<https://karma-runner.github.io/latest/intro/installation.html> - Karma installation

<http://www.simplecoding.org/personal-maps-testirovanie-angularjs-servisa-s-pomoshhyu-jasmine-i-karma-chast-5.html> - Тестирование AngularJS сервиса с помощью Jasmine и Karma

<http://stepansuvorov.com/blog/2013/01/testacular-в-помощь-тестированию/> - Karma в помощь тестированию

<http://www.protractortest.org/#/> - Protractor installation

<http://nano.sapegin.ru/all/bower> - Bower: зачем фронтенду нужен менеджер пакетов

<http://stepansuvorov.com/blog/2014/02/angularjs-protractor/> - Тестируем AngularJS используя Protractor

<https://addyosmani.com/blog/using-npm-offline/> - Offline installation of npm packages