УДК 004

Выполнение тестов Jasmine с помощью Maven

Еровлева Регина Викторовна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Еровлев Павел Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье будут рассмотрены работы по выполнению тестов Jasmine с помощью Maven. Работа будет происходить в среде разработки IntelijIdea.
Ключевые слова: Jasmine, Spring, Maven

Executing Jasmine tests with Maven

Erovleva Regina Viktorovna
Sholom-Aleichem Priamursky State University
Student

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article will walk you through how to run Jasmine tests using Maven. The work will take place in the IntelijIdea development environment.
Keywords: Jasmine, Spring, Maven

Один из самых популярных фреймворков для тестирования JavaScript — это Jasmine. Он был интегрирован во многие платформы тестирования, такие как Karma, но есть также плагин Maven для запуска тестов Jasmine.

Р.И. Ибраимов, А.Б. Джемалетдинов, А.А. Шевченко в своей работе описывают информацию для веб-разработчиков, целью которых является улучшение написанного ими кода путём его проверки и тестирования [1]. Д. Ю. Балакишиев рассказывает в своей статье о написании приложения с использованием технологий Java 8, Spring Boot, Angular 2 [2]. Р.И. Ибраимов и др. описывают процесс разработки генеалогического дерева на языке Java с использованием фреймворка Spring и библиотеки gedcom4j [3]. В.И. Зарайский описывает разработку модуля автоматизации работы с конференциями в кафедральном приложении [4]. А.А. Голикова и В.А.

Строганов в своей работе описывают систему, разработанную для пользователей новостных порталов в сети Интернет [5].

Чтобы написать тест Jasmine, сначала объявим, что собираемся тестировать (рис.1).

```
describe('The application controller', function() {
  // ...
});
```

Рисунок 1 – Объявление теста

Затем можем начать создавать некоторые переменные, содержащие область контроллера, макеты и тестовые данные (рис.2).

```
var $scope, factory, saveCallback, queryCallback;

var ITEM1 = {
  id: 1,
  description: 'My first item',
  checked: false,
  $remove: function(callback) {
    callback();
  }
}, ITEM2 = {
  id: 2,
  description: 'My second item',
  checked: true,
  $remove: function(callback) {
    callback();
  }
}, DESCRIPTION = "A description";
```

Рисунок 2 – Объявление переменных

Последней подготовкой перед написанием тестов является настройка AngularJS с использованием фреймворка (рис.3).

```
beforeEach(module('myApp.controllers'));
beforeEach(inject(function($controller, $rootScope) {
  factory = function() { };
  factory.prototype.$save = function(cb) {
    saveCallback = cb;
  };
  factory.query = function(cb) {
    queryCallback = cb;
  };

  $scope = $rootScope.$new();
  $controller('AppController', {
    '$scope': $scope,
    'Item': factory
  });
}));
```

Рисунок 3 – Настройка AngularJS

После внедрения макета внутрь контроллера и подготовления всего, что можно было, протестируем контроллер с помощью «$scope» переменной (рис.4).

```
it('should clear the textfield when adding a new item', function() {

  $scope.newItem = DESCRIPTION;
  $scope.addItem(DESCRIPTION);

  expect($scope.newItem).toBe("");
});
```

Рисунок 4 – Тест контроллера

Здесь происходит то, что идет проверка, где если добавить элемент в список, то можно очистить модель, используемую для заполнения текстового поля.

Другие тесты очень похожи, иногда нужно имитировать какое-то действие, например, запрос ресурса должен возвращать элементы (рис.5).

```
it('should have correct items', function() {

  queryCallback([ITEM1, ITEM2]);

  expect($scope.items).toContain(ITEM1);
  expect($scope.items).toContain(ITEM2);
  expect($scope.items.length).toBe(2);
});

it('should save the item when adding a new item', function() {

  $scope.items = [];
  $scope.addItem(DESCRIPTION);
  saveCallback(ITEM1);

  expect($scope.items).toContain(ITEM1);
  expect($scope.items.length).toBe(1);
});

it('should remove the item from the list when it's deleted', function() {

  $scope.items = [ITEM1, ITEM2];
  $scope.deleteItem(ITEM1);

  expect($scope.items.length).toBe(1);
  expect($scope.items).toContain(ITEM2);
  expect($scope.items).not.toContain(ITEM1);
});
```

Рисунок 5 - Тестирование

Также можно использовать Jasmine для проверки того, чтобы определенные методы были вызваны в макете (рис.6).

```
it('should update the item when changing it', function() {

  var changedItem = jasmine.createSpyObj('Item', ['$update']);
  $scope.updateItem(changedItem);

  expect(changedItem.$update).toHaveBeenCalled();
});
```

Рисунок 6 – Тестирование с запросами

Итак, сначала настраиваем элемент, чтобы он имел $update() функцию, которая стала доступной для Jasmin. Позже можно проверить, была ли эта функция вызвана с помощью toHaveBeenCalled() функции.

Чтобы настроить Maven, будем использовать два дополнительных плагина, прежде всего понадобится плагин для запуска тестов Jasmine. Однако Jasmine требует, чтобы тесты запускали на платформе, поддерживающей JavaScript. Это может быть веб-браузер или платформа вроде PhantomJS.

Также добавим некоторые дополнительные свойства для включения/отключения тестов Jasmine чтобы не приходилось запускать их каждый раз. Другое свойство будет полезно для конфигурации Jasmine. Придется обращаться к библиотекам, поэтому будет намного проще, если сохранить путь к библиотекам в свойстве (рис.7).

```
<js-tests.skip>false</js-tests.skip>
<bower-components>${basedir}/src/main/resources/static/bower_components</bower-components>
```

Рисунок 7 – Настройка maven

Конфигурация плагина немного сложнее, потому что нужно настроить все исходники, PhantomJS и файлы тестирования (рис.8).

```
<plugin>
  <groupId>com.github.searls</groupId>
  <artifactId>jasmine-maven-plugin</artifactId>
  <version>1.3.1.5</version>
  <executions>
    <execution>
      <goals>
        <goal>test</goal>
      </goals>
    </execution>
  </executions>
  <configuration>
    <skipJasmineTests>${js-tests.skip}</skipJasmineTests>
    <webDriverClassName>org.openqa.selenium.phantomjs.PhantomJSDriver</webDriverClassName>
    <webDriverCapabilities>
      <capability>
        <name>phantomjs.binary.path</name>
        <value>${phantomjs.binary}</value>
      </capability>
    </webDriverCapabilities>
    <jsSrcDir>${basedir}/src/main/resources/static/app</jsSrcDir>
    <jsTestSrcDir>${basedir}/src/test/javascript</jsTestSrcDir>
    <preloadSources>
      <source>${bower-components}/angular/angular.js</source>
      <source>${bower-components}/angular-resource/angular-resource.js</source>
      <source>${bower-components}/angular-mocks/angular-mocks.js</source>
      <source>${basedir}/src/main/resources/static/app/app.js</source>
    </preloadSources>
  </configuration>
</plugin>
```

Рисунок 8 – Настройка плагина

Настройка этого плагина немного сложнее. Прежде всего сообщаем ему, какой веб-драйвер собираемся использовать. Плагин PhantomJS установит свойство с именем ${phantomjs.binary}, поэтому можно ссылаться на него в этом плагине. Затем сообщаем ему, где находятся исходные файлы и тестовые исходные файлы, и, наконец, сообщаем ему, где находятся дополнительные исходные файлы. В основном это библиотеки и тестовые библиотеки, но также добавим в этот список app.js , чтобы само приложение всегда загружалось для тестов.

При следующей сборке с использованием Maven будет выведено, что тесты Jasmine выполнены (рис.9).

Рисунок 9 – Проверка работоспособности

В данной статье были проверены тесты с использованием Jasmine. Работа выполнялась в среде разработке IntelijIdea с помощью Spring и Maven.

Библиографический список

1. Ибраимов Р. И., Джемалетдинов А. Б., Шевченко А. А. Spring boot: создание тестов для spring mvc контроллеров // Информационно-компьютерные технологии в экономике, образовании и социальной сфере. 2017. №. 4. С. 104-111.
2. Балакишиев Д. Ю. Использование java 8, spring boot и angular 2+ для разработки веб-приложения «справочник услуг» // Образовательные технологии и общество. 2019. №3. С. 145-154.
3. Ибраимов Р. И., Зайчик А. Р., Минзатов Н. С. Разработка генеалогического дерева средствами фреймвока spring boot //Информационно-компьютерные технологии в экономике, образовании и социальной сфере. 2017. №. 4. С. 18-23.
4. Зарайский В. И. Разработка модуля автоматизации работы с конференциями в кафедральном приложении //Вестник Ульяновского государственного технического университета. 2019. №. 3. С. 74-82.
5. Голикова А. А., Строганов В. А. Система обработки и расширения данных новостной ленты //Мир компьютерных технологий. 2020. С. 155-157.