

Angular JS Unit test framework – Karma jasmine configuration steps

VERSION CONTROL

Prepared by:	S.Kalaiyarasu
Date:	19/11/2015
Reviewed and Accepted by:	Pattar. Monesh
Date:	26/11/2015
Approved by:	Baskaran.Varadarajan, D.A.Soundararajan
Date:	29/12/2015

VERSION HISTORY

Date	Version	Author	Description
19/11/2015	0.1	S.Kalaiyarasu	Initial draft version
26/11/2015	1.0	Pattar. Monesh	Final Reviewed
28/12/2015	1.1	Baskaran.Varadarajan	Final Approved

Contents

1. INTRODUCTION TO UNIT TEST CASE FRAMEWORKS.....	4
2. INTRODUCTION TO KARMA JASMINE FRAMEWORK USING YEOMAN	5
3. FOLDER STRUCTURE.....	6
4. KARMA JASMINE CONFIGURATION FILES	7
5. ADDING DEPENDENCY FILES AND BOWER COMPONENTS.....	8
6. WRITING TEST FILE USING JASMINE.....	10
7. RUNNING KARMA JASMINE TEST FILE USING KARMA	12

1. INTRODUCTION TO UNIT TEST CASE FRAMEWORKS

Unit testing involves **breaking your program into pieces**, and subjecting each piece to a series of tests.

Unit testing simply verifies that individual units of code (mostly functions) work as expected. Usually you write the test cases yourself

Some of unit test case frameworks:

1. Junit
2. Mocha
3. UnitTesting
4. JSpec
5. Jasmine.

Jasmine is one of the framework which will be used to write test cases for javascript based files. Once done writing the test case using jasmine framework, that you can run using the karma test runner.

2. INTRODUCTION TO KARMA JASMINE FRAMEWORK USING YEOMAN

Karma:

Karma is a test runner for JavaScript that runs on Node.js. It is very well suited to testing AngularJS or any other JavaScript projects. Using Karma to run tests using one of many popular JavaScript testing suites (Jasmine, Mocha, QUnit, etc.) and have those tests executed not only in the browsers of your choice, but also on the platform of your choice (desktop, phone, tablet.) Karma is highly configurable, integrates with popular continuous integration packages (Jenkins, Travis, and Semaphore) and has excellent plugin support.

Jasmine:

Jasmine is an open source testing framework for JavaScript. It aims to run on any JavaScript-enabled platform, to not intrude on the application nor the IDE, and to have easy-to-read syntax.

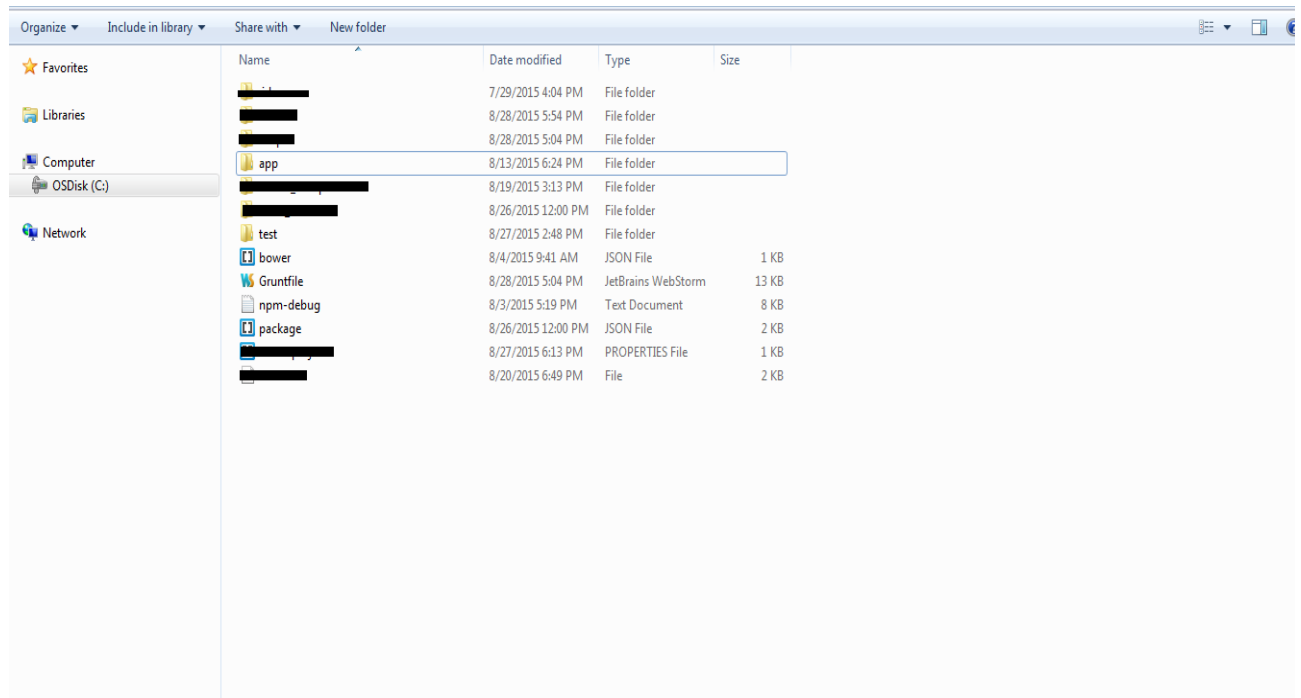
Yeoman:

Yeoman helps you to kick start new projects, prescribing best practices and tools to help you stay productive.

To do so, we provide a generator ecosystem. A generator is basically a plugin that can be run with the `yo` command to scaffold complete projects or useful parts.

3. FOLDER STRUCTURE

Yeoman installation will provide us the default MVC folder structure for our project. Which includes karma and jasmine by default in our configuration. Once configured the yeoman, the folder structure will be looks like as shown below.



As presents in screen shot above, **app**, **test**, **bower.json**, **gruntfile.js** and **package.json** are the important files that yeoman will create while configure it.

Test folder will contains all the unit test case file for respective source file present under app folder.

Note:

Running the **'YO'** command in your command prompt under specific path, will generate the folder structure as shown above.

4. KARMA JASMINE CONFIGURATION FILES

Karma.conf.js:

- The one important file required for testing jasmine and karma is karma.conf.js

This file contains all the dependencies that are belong to our angular js application module. While running grunt test command, this will loads all the dependencies files presents in karma.conf.js file and cross check with index.html.



```

1 // Karma configuration
2 // http://karma-runner.github.io/0.12/config/configuration-file.html
3 // Generated on 2015-07-24 using
4 // generator-karma 1.0.0
5
6 module.exports = function(config) {
7   'use strict';
8
9   config.set({
10     // enable / disable watching file and executing tests whenever any file changes
11     autoWatch: true,
12
13     // base path, that will be used to resolve files and exclude
14     basePath: './',
15
16     // testing framework to use (jasmine/mocha/qunit/...)
17     // as well as any additional frameworks (requirejs/chai/sinon/...)
18     frameworks: [
19       "jasmine"
20     ],
21
22     // list of files / patterns to load in the browser
23     files: [
24
25       'bower_components/angular-sanitize/angular-sanitize.js',
26       'bower_components/angular-touch/angular-touch.js',
27       'bower_components/jquery-ui/jquery-ui.js',
28       'bower_components/angular-ui-sortable/sortable.js',
29       'bower_components/angular-local-storage/dist/angular-local-storage.js',
30       'bower_components/angular-mocks/angular-mocks.js',
31       'bower_components/ocLazyLoad/dist/ocLazyLoad.min.js',
32       'bower_components/angular-ui-router/release/angular-ui-router.min.js',
33       'bower_components/angular-loading-bar/src/loading-bar.js',
34       'bower_components/angular-animate/angular-animate.js',
35       'bower_components/angular-cookies/angular-cookies.js',
36       'bower_components/angular-resource/angular-resource.js',
37       'bower_components/angular-route/angular-route.js',
38       'bower_components/angular-sanitize/angular-sanitize.js',
39       'bower_components/angular-touch/angular-touch.js',
40
41       "app/scripts/**/*.js",
42       "test/mock/**/*.js",
43       "test/spec/controllers/TestController.js"
44     ]
45   });
46 }
  
```

Sample Karma.conf.js file

In the above screen shot, last three lines are js files mapped from app and test folder. While running grunt test command, these test files will be tested and the respective source file will be mapped for controller as well as module verification.

And also there are dependency files, which are mapped before we mentioned app and test folder. These dependencies are presents in our angular application modules.

5. ADDING DEPENDENCY FILES AND BOWER COMPONENTS

Dependency files and bower components are configured in karma.conf.js file. These dependency and bower files are two types. One is project dependency file and second one is bower components that we used in app modules as sub modules.

Dependency file sample (Type 1):

```

25 'bower_components/jquery/dist/jquery.js',
26 'bower_components/angular/angular.js',
27 'bower_components/bootstrap/dist/js/bootstrap.js',
28 'bower_components/angular-bootstrap/ui-bootstrap.min.js',
29 'bower_components/angular-animate/angular-animate.js',
30 'bower_components/angular-cookies/angular-cookies.js',
31 'bower_components/angular-resource/angular-resource.js',
32 'bower_components/angular-route/angular-route.js',
33 'bower_components/angular-sanitize/angular-sanitize.js',
34 'bower_components/angular-touch/angular-touch.js',
35 'bower_components/jquery-ui/jquery-ui.js',
36 'bower_components/angular-ui-sortable/sortable.js',
37 'bower_components/angular-local-storage/dist/angular-local-storage.js',
38 'bower_components/angular-mocks/angular-mocks.js',
39 'bower_components/oclazyload/dist/oclazyload.min.js',
40 'bower_components/angular-ui-router/release/angular-ui-router.min.js',
41 'bower_components/angular-loading-bar/src/loading-bar.js',
42 'bower_components/angular-animate/angular-animate.js',
43 'bower_components/angular-cookies/angular-cookies.js',
44 'bower_components/angular-resource/angular-resource.js',
45 'bower_components/angular-route/angular-route.js',
46 'bower_components/angular-sanitize/angular-sanitize.js',
47 'bower_components/angular-touch/angular-touch.js',
48 'bower_components/angular-ui-grid/ui-grid.min.js',
49 'bower_components/angular-translate/angular-translate.js',
50 'bower_components/angular-translate/angular-translate.min.js',
51 'bower_components/angular-translate-loader-url/angular-translate-loader-url.min.js',
52 'bower_components/angular-translate-loader-url/angular-translate-loader-url.js',
53 'bower_components/angular-translate-loader-static-files/angular-translate-loader-static-files.min.js',
54 'bower_components/angular-translate-loader-static-files/angular-translate-loader-static-files.js',
55 // endbower
56 "app/scripts/**/*.js",
57 "test/mock/**/*.js",
58 "test/spec/controllers/TestController.js"
59
60
61 // list of files / patterns to exclude
62 exclude: [
63 ],
64
65 // web server port
66 port: 8080,
67
68 // Start these browsers, currently available:
69 // - Chrome

```

In the above screen shot, you can see that we have mapped our source and test files in karma.conf.js file.

Dependency file sample (Type 2):

```

25 'bower_components/jquery/dist/jquery.js',
26 'bower_components/angular/angular.js',
27 'bower_components/bootstrap/dist/js/bootstrap.js',
28 'bower_components/angular-bootstrap/ui-bootstrap.min.js',
29 'bower_components/angular-animate/angular-animate.js',
30 'bower_components/angular-cookies/angular-cookies.js',
31 'bower_components/angular-resource/angular-resource.js',
32 'bower_components/angular-route/angular-route.js',
33 'bower_components/angular-sanitize/angular-sanitize.js',
34 'bower_components/angular-touch/angular-touch.js',
35 'bower_components/jquery-ui/jquery-ui.js',
36 'bower_components/angular-ui-sortable/sortable.js',
37 'bower_components/angular-local-storage/dist/angular-local-storage.js',
38 'bower_components/angular-mocks/angular-mocks.js',
39 'bower_components/oclazyload/dist/oclazyload.min.js',
40 'bower_components/angular-ui-router/release/angular-ui-router.min.js',
41 'bower_components/angular-loading-bar/src/loading-bar.js',
42 'bower_components/angular-animate/angular-animate.js',
43 'bower_components/angular-cookies/angular-cookies.js',
44 'bower_components/angular-resource/angular-resource.js',
45 'bower_components/angular-route/angular-route.js',
46 'bower_components/angular-sanitize/angular-sanitize.js',
47 'bower_components/angular-touch/angular-touch.js',
48 'bower_components/angular-ui-grid/ui-grid.min.js',
49 'bower_components/angular-translate/angular-translate.js',
50 'bower_components/angular-translate/angular-translate.min.js',
51 'bower_components/angular-translate-loader-url/angular-translate-loader-url.min.js',
52 'bower_components/angular-translate-loader-url/angular-translate-loader-url.js',
53 'bower_components/angular-translate-loader-static-files/angular-translate-loader-static-files.min.js',
54 'bower_components/angular-translate-loader-static-files/angular-translate-loader-static-files.js',
55 // endbower
56 "app/scripts/**/*.js",
57 "test/mock/**/*.js",
58 "test/spec/controllers/TestController.js"
59
60
61 // list of files / patterns to exclude
62 exclude: [
63 ],
64
65 // web server port
66 port: 8080,
67
68 // Start these browsers, currently available:
69 // - Chrome

```


In above screen shot, highlighted bower dependencies that we are using in our angular JS module as shown below.

6. WRITING TEST FILE USING JASMINE

Look at the screen shots below for the sample source file and test case file.

```

1 'use strict';
2 /**
3  * @ngdoc function
4  * @name R2FPOC.controller:TestController
5  * @description
6  * # TestController
7  * Controller of the R2FPOC
8  */
9 angular.module('R2FPOC')
10 .controller('SimpleController', function($scope) {
11     $scope.Title = "Search Finance Engine";
12     $scope.formName = "SearchFinanceForm";
13     $scope.selectedValue = "";
14     $scope.data = [];
15     $scope.url = '';
16     $scope.name = "";
17     $scope.dataJSON = {};
18     $scope.refData = {};
19     $scope.refData.data = {};
20     $scope.getGridData = function(){
21         $scope.url = '/COB/warehouse';
22     };
23 });
24

```

Sample source file

```

1 describe('SimpleController', function () {
2     beforeEach(module('R2FPOC'));
3
4     var controller, $controller, $scope = {};
5
6     beforeEach(inject(function (_$controller_) {
7
8         $controller = _$controller_;
9         controller = $controller('SimpleController', {$scope: $scope});
10        $scope.Title = "Search Finance Engine";
11        $scope.formName = "SearchFinanceForm";
12        $scope.selectedValue = "";
13        $scope.data = [];
14        $scope.url = '';
15        $scope.name = "";
16        $scope.dataJSON = {};
17        $scope.refData = {};
18        $scope.refData.data = {};
19        $scope.getGridData();
20    }));
21
22    it('Scope title equal to Search Finance Engine: Success scenario', function () {
23        expect($scope.Title).toEqual('Search Finance Engine');
24    });
25
26    it('Verify the URL value from scope', function () {
27        expect($scope.url).toEqual('/COB/warehouse');
28    });
29
30    /*
31    it('Scope title not equal to Search Finance Engine: Failure scenario', function () {
32        expect($scope.Title).toEqual('Fail');
33    });*/
34
35    /*
36    it('Verify the URL value from scope', function () {
37        expect($scope.url).toEqual('/COB/warehouse');
38    });*/
39
40 });

```

Sample test file

Key concepts in jasmine framework:**Describe:**

```
describe('SimpleController', function () { });
```

Describe is the function that will combine all the unit test case into one section with the name of Source Controller name. One test case file may contain more than one describe, each will contain their own test cases writing inside it.

BeforeEach:

```
beforeEach(module('PROJECT'));
```

```
beforeEach(inject(function (_$controller_) {  
    $controller = _$controller_;  
    controller = $controller('SimpleController', {$scope: $scope});  
    $scope.Title = "Search Finance Engine";  
}));
```

Before each function will load the content inside the function before running each test case. So that the required data and module will be created before running the test case and the populated data will be used by test case block.

IT:

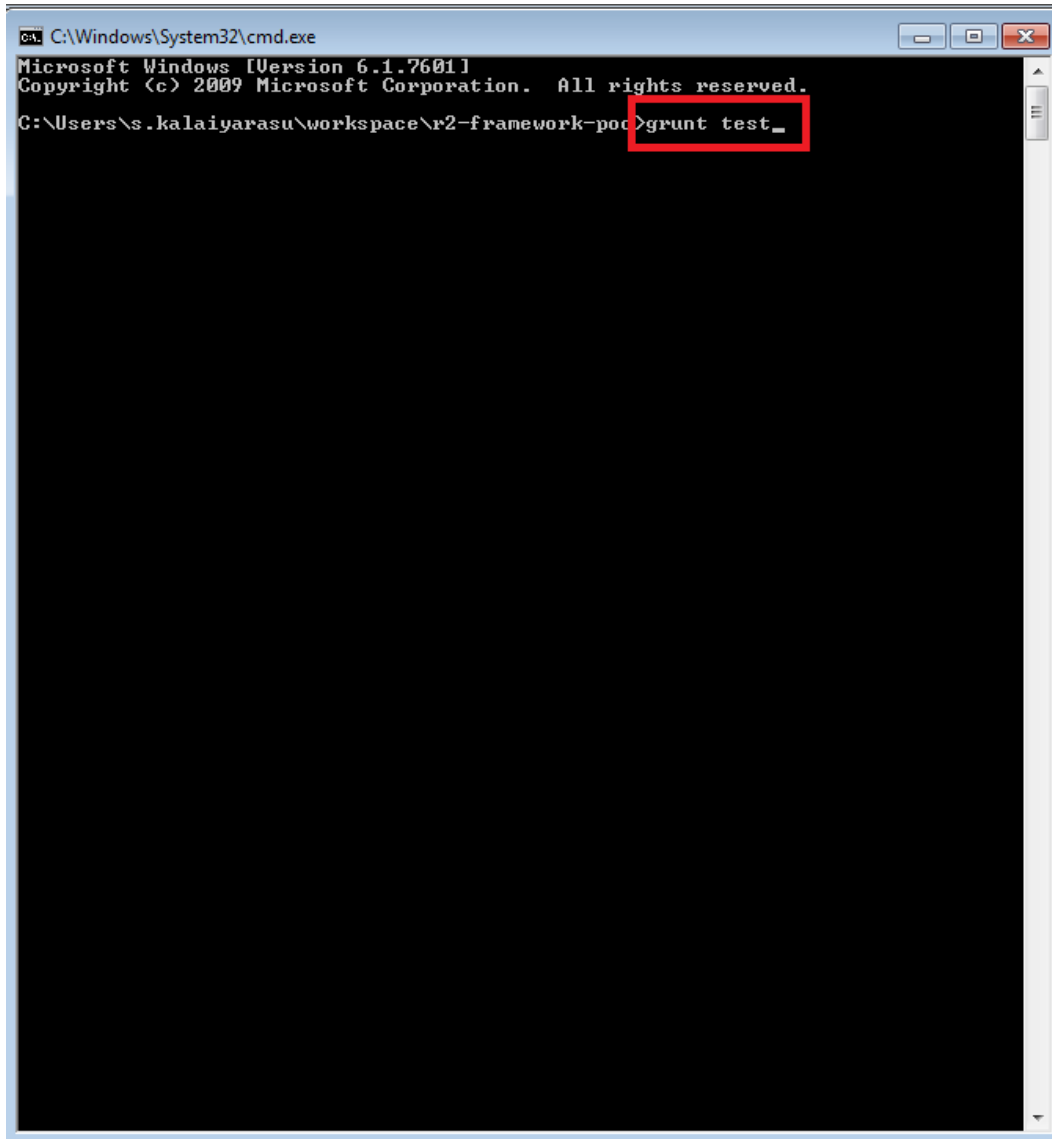
```
it('Scope title equal to Search Finance Engine: Success scenario', function () {  
    expect($scope.Title).toEqual('Search Finance Engine');  
});
```

'it' function will act as a test case. Each it function is a separate test case that will be created under describe function. Inside it function we can have some test code as shown above.

Have a look into sample source file and test case file screen shot for better understanding.

7. RUNNING KARMA JASMINE TEST FILE USING KARMA

grunt test is the command to run the test file. This command you have to run from your client path command prompt.



Once the command executed test case successfully, it will show the success and failure count of test case.

Make sure you are running the test command from your client path.

```
C:\Windows\System32\cmd.exe
Total 11.7s

C:\Users\s.kalaiyarasu\workspace\r2-framework-poc>grunt test --force
>> Local Npm module "jshint" not found. Is it installed?

Running "clean:server" (clean) task
Cleaning .tmp...OK

Running "concurrent:test" (concurrent) task
>> Local Npm module "jshint" not found. Is it installed?

Running "copy:styles" (copy) task
Copied 3 files

Done, without errors.

Running "autoprefixer:dist" (autoprefixer) task
File .tmp/styles/main.css created.
File .tmp/styles/r2-framework-poc.css created.
File .tmp/styles/timeline.css created.

Running "connect:test" (connect) task
Started connect web server on http://localhost:1010

Running "karma:unit" (karma) task
WARN [watcher]: Pattern "C:/Users/s.kalaiyarasu/workspace/r2-framework-poc/test/mock/**/*.js" does not match any file.
INFO [karma]: Karma v0.12.37 server started at http://localhost:8080/
INFO [launcher]: Starting browser PhantomJS
INFO [PhantomJS 1.9.8 (Windows 7 0.0.0)]: Connected on socket 4uFvFAQsqy5hIjrtnNMF with id 76155603
PhantomJS 1.9.8 (Windows 7 0.0.0): Executed 1 of 3 SUCCESS (0 secs / 0.075 secs)
PhantomJS 1.9.8 (Windows 7 0.0.0): Executed 2 of 3 SUCCESS (0 secs / 0.083 secs)
PhantomJS 1.9.8 (Windows 7 0.0.0) TestController Scope title not equal to Search Finance Engine: Failure scenario FAILED
Expected 'Search Finance Engine' to equal 'Fail'.
    at C:/Users/s.kalaiyarasu/workspace/r2-framework-poc/test/spec/controllers/TestController.js:31
PhantomJS 1.9.8 (Windows 7 0.0.0): Executed 3 of 3 (1 FAILED) (0 secs / 0.089 se
PhantomJS 1.9.8 (Windows 7 0.0.0): Executed 3 of 3 (1 FAILED) (0 secs / 0.089 se
cs)
Warning: Task "karma:unit" failed. Used --force, continuing.

Done, but with warnings.

Execution Time <2015-09-07 07:02:10 UTC>
concurrent:test      8.2s ████████████████████████████████ 59%
autoprefixer:dist   251ms █ 2%
connect:test        487ms ██ 3%
karma:unit          4.9s ████████████████████████████████ 35%
Total 14s

C:\Users\s.kalaiyarasu\workspace\r2-framework-poc>.idea_
```

In the above test case result, out of three test cases, two are getting passed and one got failure due to expect function not matching with actual result. The failure error will throw with test case description name which is showing in red.

```
C:\Windows\System32\cmd.exe
```

```
INFO: -----  
C:\Users\s.kalaiyarasu\workspace\r2-framework-poc>grunt test  
>> Local Npm module "jshint" not found. Is it installed?  
  
Running "clean:server" <clean> task  
Cleaning .tmp...OK  
  
Running "concurrent:test" <concurrent> task  
  
>> Local Npm module "jshint" not found. Is it installed?  
  
Running "copy:styles" <copy> task  
Copied 3 files  
  
Done, without errors.  
  
Running "autoprefixer:dist" <autoprefixer> task  
File .tmp/styles/main.css created.  
File .tmp/styles/r2-framework-poc.css created.  
File .tmp/styles/timeline.css created.  
  
Running "connect:test" <connect> task  
Started connect web server on http://localhost:1010  
  
Running "karma:unit" <karma> task  
WARN [watcher]: Pattern "C:/Users/s.kalaiyarasu/workspace/r2-framework-poc/test/  
mock/**/*.js" does not match any file.  
INFO [karma]: Karma v0.12.37 server started at http://localhost:8080/  
INFO [launcher]: Starting browser PhantomJS  
INFO [PhantomJS 1.9.8 (Windows ? 0.0.0)]: Connected on socket -at35u3e7MUTKlhceq  
2G with id 20702300  
PhantomJS 1.9.8 (Windows ? 0.0.0): Executed 1 of 2 SUCCESS <0 secs / 0.082 secs>  
PhantomJS 1.9.8 (Windows ? 0.0.0): Executed 2 of 2 SUCCESS <0.016 secs / 0.09 se  
cs>  
  
Done, without errors.  
  
Execution Time <2015-09-08 10:22:31 UTC>  
concurrent:test    11.3s   ██████████ 56%  
autoprefixer:dist  234ms   █ 1%  
connect:test       671ms   ██ 3%  
karma:unit         7.8s    ██████████ 39%  
Total 20.3s  
  
C:\Users\s.kalaiyarasu\workspace\r2-framework-poc>
```

In the above screen shot, executed all the test cases successfully.