Workshop Web tools - Part 2

Ahmed Bannany

Table of content

1. Testing

Javascript is a standardized programming language. The last version of javascript is 'ECMAScript 2016'. There are a lot of frameworks that can be used to test the javascript code. There are two kind of test frameworks. Unit test and e2e (end to end) test frameworks. With the first one a piece of code can be tested. With e2e testing a test can be performed from the users point of view. In the Javascript world there are a lot of test frameworks. The following is a snapshot of this test frameworks:

- QUnit
- MochaJs
- Jasmine

In this section we shall write a unit test with Jasmine. Normally we use Karma to runt the tests. But in this section just Jasmine is used.

1.1. Exercise 1

In this exercise Jasmine as unit test framework shall be used. Jasmine is BDD test framework. Jasmine was inspired by Rspec and Jspec. Jasmine is the most used test framework in Javascript world. The angular team used also this framework.

The objectives of this exercise is:

- Writing a simple Jasmine test
- To run the test in a browser

1.1.1. Steps

1. Perform the following commands

```
# Go to exercises/simple-jasmine-test
cd exercises/simple-jasmine-test

# Create a folder named spec
mkdir spec
cd spec

# Create a file named CalculatorSpec.js
touch CalculatorSpec.js
```

2. Add the following code snippet in the created file CalculatorSpec.js

```
describe("Calculator", function () {  ①
  describe("when addition is performed", function () {

    var calculator;

    beforeEach(function () { ②
      calculator = new Calculator();
    });

    it("should return 5 when 3 is added to 2", function () { ③
      expect(calculator.add(3, 2)).toEqual(5);
    });

    it("should return 20 when 10 is added to 10", function () { ③
      expect(calculator.add(10, 10)).toEqual(20);
    });
  });
});
```

```
<1> Here we define the test suite. The name of the suite is 'Calculator'.
<2> Fixture setup. This step is performed for every test
<3> The test specs to test the scenario's
```

3. Open the file SpecRunner.html of the 'simple-jasmine-test' folder with your favorite text editor.
    - The following shall be visible

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Jasmine Spec Runner v2.4.1</title>

  <link rel="shortcut icon" type="image/png" href="lib/jasmine-
2.4.1/jasmine_favicon.png">
  <link rel="stylesheet" href="lib/jasmine-2.4.1/jasmine.css"> ①

  <script src="lib/jasmine-2.4.1/jasmine.js"></script> ②
  <script src="lib/jasmine-2.4.1/jasmine-html.js"></script>
  <script src="lib/jasmine-2.4.1/boot.js"></script>

  <!-- include source files here... -->
  <script src="src/Calculator.js"></script> ③

  <!-- include spec files here... -->
  <script src="spec/CalculatorSpec.js"></script> ④

</head>

<body>
</body>
</html>
```

```
<1> Loading the Jasmine style sheet
<2> The Jasmine library is loaded
<3> Object under test is loaded
<4> Test specs are loaded
```

4. Open the file SpecRunner.html in the browser.
    - When the file is opened the a overview of the jasmine test is depicted

1.2. Exercise 2

In this exercise we shall use Karma. Karma is a test runner that runs tests on several browsers.

Without configuration for Karma, Karma shall do nothing. We must tel Karma:

- which tests and libraries to load.
- on which browser to run the tests.
- which tasks to perform before the tests are performed.
- to run the tests several times.

All of this must be included in a Karma configuration file.

The objectives for this exercise is to: * learn how to install Karma. * create a simple Karma configuration. * use Gulp to run Karma.

1.2.1. Steps

1. Go to folder 'Exercises/Karma-gulp'.
2. Execute the following commands

```
npm install -g Karma
```

3. Create a file named 'Karma.conf.js' and add the following code snippet to this file.

```
module.exports = function (config) {
    config.set({
        basePath: '',
        frameworks: ['jasmine', 'angular-filesort'], ①
        files: [ ②
            'bower_components/angular/angular.js',
            'bower_components/angular-mocks/angular-mocks.js',
            'src/**/*.js',
            'tests/*Spec.js',
            'tests/**/*Spec.js'
        ],
        angularFilesort: { ③
            whitelist: [
                'src/**/*.js'
            ]
        },

        exclude: [],

        reporters: ['progress'],
        port: 9876,
        colors: true,
        preprocessors: {},
        logLevel: config.LOG_INFO,
        autoWatch: true,

        browsers: ['Chrome_without_security'], ④

        customLaunchers: { ⑤
            Chrome_without_security: {
                base: 'Chrome',
                flags: ['--disable-web-security']
            }
        },

        singleRun: true,
        captureTimeout: 60000
    });

    ⑥
};
```

```
<1> Configure which test framework to use. Here we use Jasmine. The 'angular-filesort'
is used to load the
the files in the correct order.
<2> Loading the libraries and tests to use
<3> Configure which file to sort by the 'angular-filesort' module
<4> Which browser to use to execute the tests
<5> Configuration of the browser part
<6> See Karma site for further explanation (http://Karma-
runner.github.io/1.0/config/configuration-file.html)
```

4. Create a file with the name 'gulpfile.js' and add the following code snippet to this file.

```
var gulp = require('gulp');
var Server = require('Karma').Server;

gulp.task('test', function (done) { ①
    var KarmaConfig = {
        configFile: __dirname + '/Karma.conf.js' ②
    };
    new Server(KarmaConfig, done).start();
});

gulp.task('test-debug', function (done) { ③
    var KarmaConfig = {
        configFile: __dirname + '/Karma.conf.js',
        singleRun: false ④
    };
    new Server(KarmaConfig, done).start();
});

gulp.task('default', ['test-once']);
```

```
<1> Create a task named 'test' to run Karma and close the browser.
<2> With configFile the location of the Karma configuration can be set.
<3> Create the task 'test-debug' to run Karma and let the browser open.
<4> The property singleRun is set to false. This is done prevent Karma to close the
browser.
```

There is no plugin loaded in the gulp file to run Karma. The was one named 'gulp-Karma' but this plugin has been deprecated.

2. Angular 2

Angular 2 is not the same as Angular 1. Jus name Angular is the same. Migration from angular 1 to Angular 2 is very hard. Sometimes is a rewrite to Angular 2 better.

In this section you shall write an angular 2 application.

2.1. Exercise 3

In this exercise the Angular 2 Cli shall be used to create an Angular 2 project.

The objectives of this exercise are:

- Creating an angular 2
- Adding bootstrap 3

2.1.1. Steps

1. Go the the exercise folder and perform the following commands

```
npm install -g @angular/cli
ng new my-app ①
cd my-app
ng serve ②
```

```
<1> Creating a new application.
<2> 'ng serve' is used to start our application
```

The created application is a template application that contains:

- Typescript configuration
- Karma configuration
- .angular-cli.json is part of the webpack configuration

3. Perform the following commands to add the bootstrap libraries

```
npm install ng2-bootstrap bootstrap --save
```

4. Add the following in the .angular-cli.json:

```
      ...
      "styles": [
        "styles.css",
        "../node_modules/bootstrap/dist/css/bootstrap.min.css"
      ],
      ...
```