**Protractor for AngularJS**

## Table of contents

1. [Why is testing so important?](http://ramonvictor.github.io/protractor/slides/#/2)
2. [Whats the idea behind E2E testing?](http://ramonvictor.github.io/protractor/slides/#/3)
3. [The dark side of E2E testing](http://ramonvictor.github.io/protractor/slides/#/4)
4. [Protractor is built on top of WebDriverJS](http://ramonvictor.github.io/protractor/slides/#/6)
5. [Install](http://ramonvictor.github.io/protractor/slides/#/8)
6. [Setup a conf.js file](http://ramonvictor.github.io/protractor/slides/#/9)
7. [Write your tests using Jasmine and WebdriverJS](http://ramonvictor.github.io/protractor/slides/#/10)
8. [Protractor global variables](http://ramonvictor.github.io/protractor/slides/#/11)
9. [Let's run it?](http://ramonvictor.github.io/protractor/slides/#/13)
10. [element() vs element.all()](http://ramonvictor.github.io/protractor/slides/#/15)
11. [Executing events](http://ramonvictor.github.io/protractor/slides/#/22)
12. [Promises and the Control Flow](http://ramonvictor.github.io/protractor/slides/#/25)
13. [Debugging using elementexplorer](http://ramonvictor.github.io/protractor/slides/#/28)
14. [Maintanable Tests](http://ramonvictor.github.io/protractor/slides/#/29)
15. [Tests directory structure](http://ramonvictor.github.io/protractor/slides/#/31)
16. [Page Objects](http://ramonvictor.github.io/protractor/slides/#/32)
17. [Node.JS exports and require](http://ramonvictor.github.io/protractor/slides/#/33)
18. [Separate your tests in various test suites](http://ramonvictor.github.io/protractor/slides/#/35)
19. [Enable multiCapabilities](http://ramonvictor.github.io/protractor/slides/#/36)
20. [Using onPrepare](http://ramonvictor.github.io/protractor/slides/#/37)
21. [Using params](http://ramonvictor.github.io/protractor/slides/#/40)
22. [Using jasmineNodeOpts](http://ramonvictor.github.io/protractor/slides/#/42)
23. ["But, I want to use Protractor in a non-AngularJS app"](http://ramonvictor.github.io/protractor/slides/#/43)
24. [gulp-protractor-qa](http://ramonvictor.github.io/protractor/slides/#/49)
25. [Final thoughts](http://ramonvictor.github.io/protractor/slides/#/51)
26. [Learning Resources](http://ramonvictor.github.io/protractor/slides/#/52)

## Why is testing so important?

*Testing is about gaining confidence that your code does what you think it should do*

## Whats the idea behind E2E testing?

* How would the users see my application?
* Is my backend communicating with my frontend?
* Can I release this code?
* It does **NOT** replace Unit Testing!

## The dark side of E2E testing

* It needs a specific running environment
* It's hard to write
* It's difficult to debug
* It's hard to keep the tests up-to-date

## Protractor is built on top of[WebDriverJS](https://code.google.com/p/selenium/wiki/WebDriverJs)

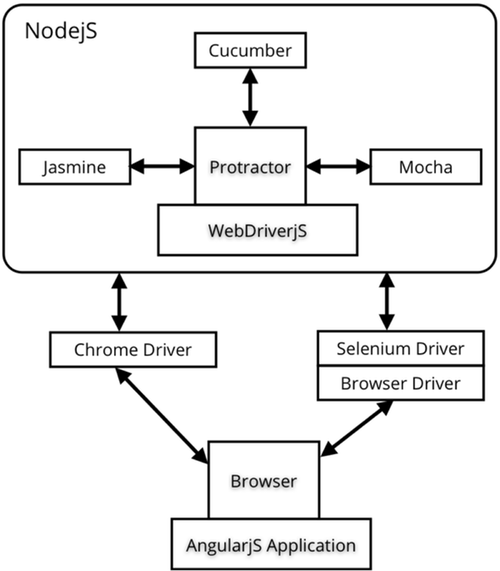
**Testing system** (NodeJS, Java, etc)

|

**Webdriver** (a.k.a. Selenium)

|

**Your AngularJS App**



## Install

1. Download [Node.JS](http://nodejs.org/download/)
2. sudo npm install protractor -g
3. sudo webdriver-manager update

## Setup a conf.js file

exports.config = {

seleniumAddress: 'http://localhost:4444/wd/hub',

capabilities: {

'browserName': 'chrome'

},

specs: ['example-spec.js'],

jasmineNodeOpts: {

showColors: true

}

};

## Write your tests using[Jasmine](http://jasmine.github.io/2.0/introduction.html) and WebdriverJS

describe('by model', function() {

it('should find an element by text input model', function() {

var username = element(by.model('username'));

var name = element(by.binding('username'));

username.clear();

expect(name.getText()).toEqual('');

username.sendKeys('Jane Doe');

expect(name.getText()).toEqual('Jane Doe');

});

});

## Protractor global variables

* browser: browser.get()
* element and by: element(by.model('yourName'))
* protractor: protractor.Key

## Basic example

// example-spec.js

describe('angularjs homepage', function() {

it('should greet the named user', function() {

browser.get('http://www.angularjs.org');

element(by.model('yourName')).sendKeys('Julie');

var greeting = element(by.binding('yourName'));

expect(greeting.getText()).toEqual('Hello Julie!');

});

});

## Let's run it?

First things first, open the terminal and start the webdriver server:

webdriver-manager start

After that, you can run Protractor in another terminal by typing:

protractor test/e2e/config.js // this is the relative path to your config.js file

## Searching for elements on the page

## element() vs element.all()

Single element

element( by.binding('appName') );

Collection of elements

// clicks the 3rd element

element.all( by.css('[ng-click="openPage()"]') ).get(2).click();

## by.binding

### In your test

element( by.binding('myModel') );

### In your application

<span ng-bind="myModel"></span>

<!-- or -->

<span>{{myModel}}</span>

## by.model

### In your test

element( by.model('myModel') );

### In your application

<input ng-model="myModel" />

## by.repeater

### In your test

element( by.repeater('user in users').row(0).column('name') );

### In your application

<ul>

<li ng-repeat="user in users">

<span>{{user.name}}</span>

</li>

</ul>

## by.css

### In your test

element( by.css('[ng-click="sendMail()"]') );

### In your application

<button ng-click="sendMail()">Send mail!</button>

## Find out more in Protractor API

* by.select()
* by.partialButtonText()
* elementArrayFinder.each()
* [Full list of available methods](http://angular.github.io/protractor/#/api)

## Searching elements best practices

* Take advantage of AngularJS attributes using by.model,by.repeater, etc
* Avoid using potential CSS attributes, mainly IDs and Classes.

## Executing events

## .click()

### In your test

element( by.css('[ng-click="submit()"]') ).click();

### In your application

<button ng-click="submit()"><button>

## On Enter Press

### In your test

element( by.model('commentText') ).sendKeys("Hi!", protractor.Key.ENTER);

### In your application

<textarea ng-model="commentText"><textarea>

## Promises and the Control Flow

## Promises based

All Protractor methods are asynchronous and return [promises](https://www.promisejs.org/).

// Example of getText() promise

element( by.model('zipcode') ).getText()

.then(function(val) {

var num = val.substring(0, 4);

var isNum = !isNaN(num);

expect( isNum ).toBeTruthy();

});

## Control Flow

WebDriverJS maintains a queue of pending promises, called the[control flow](https://github.com/angular/protractor/blob/master/docs/control-flow.md), to keep execution organized.

it('should find an element by text input model', function() {

browser.get('#/home'); // (1) method browser.get

// (2) method by.binding

var login = element(by.binding('login'));

// (3) method getText

expect(login.getText()).toEqual('User');

});

In the example above, the control flow would execute the queue following the sequence we see in the comments. Basically method by.binding would only run once browser.get promise is resolved, and so on.

## Debugging using elementexplorer

1. Start your webdriver server:  
   webdriver-manager start
2. Run:  
   /usr/local/lib/node\_modules/protractor/bin/**elementexplorer.js** http://angularjs.org
3. Press 'tab' and play with any element locator.

## Maintanable Tests

### Best practices to organize your tests

## The big picture

* **Page Objects** - These are the js files where you map the elements and write the functions to perform actions;
* **Exports and Require** - This is how you connect your Page Objects to your Test Specs;
* **Test specs** - These are the js files where you write your tests using jasmine syntax.

## Tests directory structure

projectfolder/

|-- css/

|-- js/

|-- img/

|-- tests/

|-- unit/

|-- e2e/

| |-- homepage/

| | |-- homepage.po.js

| | |-- \*.spec.js

| |-- profile/

| | |-- profile.po.js

| | |-- \*.spec.js

| |-- config.js

var AngularHomepage = function() {

this.nameInput = element(by.model('yourName'));

this.greeting = element(by.binding('yourName'));

this.get = function() {

browser.get('http://www.angularjs.org');

};

this.setName = function(name) {

this.nameInput.sendKeys(name);

};

};

## Node.JS exports and require

**Your Page Object file**

var AngularHomepage = function() {

this.nameInput = element(by.model('yourName'));

this.greeting = element(by.binding('yourName'));

// ...

};

module.exports = AngularHomepage;

**Your Test file**

var AngularHomepage = require('./homepage.po.js');

describe('HomePage Tests', function() {

var angularHomepage = new AngularHomepage();

angularHomepage.nameInput.sendKeys('Rafael');

//...

});

## Node.JS exports and require

**Your Page Object file**

var AngularHomepage = function() {

this.nameInput = element(by.model('yourName'));

this.greeting = element(by.binding('yourName'));

// ...

};

module.exports = AngularHomepage;

**Your Test file**

var AngularHomepage = require('./homepage.po.js');

describe('HomePage Tests', function() {

var angularHomepage = new AngularHomepage();

angularHomepage.nameInput.sendKeys('Rafael');

//...

});

## Separate your tests in various test suites

exports.config = {

seleniumAddress: 'http://localhost:4444/wd/hub',

capabilities: { 'browserName': 'chrome' },

suites: {

homepage: 'tests/e2e/homepage/\*\*/\*Spec.js',

search: ['tests/e2e/contact\_search/\*\*/\*Spec.js']

},

jasmineNodeOpts: { showColors: true }

};

**Running specific suite of tests**

protractor protractor.conf.js --suite homepage

## Enable multiCapabilities

exports.config = {

seleniumAddress: 'http://localhost:4444/wd/hub',

multiCapabilities: [

{

'browserName' : 'chrome'

},

{

'browserName' : 'firefox'

}

],

specs: ['example-spec.js'],

jasmineNodeOpts: {

showColors: true

}

};

## Using onPrepare

**Set window size before starting the tests**

exports.config = {

seleniumAddress: 'http://localhost:4444/wd/hub',

capabilities: {

'browserName': 'chrome'

},

onPrepare: function() {

browser.driver.manage().window().setSize(1600, 800);

},

jasmineNodeOpts: {

showColors: true

}

};

## Using onPrepare

**Export xml results of your Automated Suites**

First, install jasmine-reporters:

npm install jasmine-reporters

And to keep xml results in Timestamp directories, install mkdirp package:

npm install mkdirp

## Using onPrepare

**Export xml results of your Automated Suites**

## Using onPrepare

**Export xml results of your Automated Suites**

//config.js

exports.config = {

onPrepare: function() {

var folderName = (new Date()).toString().split(' ').splice(1, 4).join(' ');

var mkdirp = require('mkdirp');

var newFolder = "./reports/" + folderName;

require('jasmine-reporters');

mkdirp(newFolder, function(err) {

if (err) {

console.error(err);

} else {

jasmine.getEnv().addReporter(new jasmine.JUnitXmlReporter(newFolder, true, true));

}

});

},

};

## Using params

Your config.js

exports.config = {

seleniumAddress: 'http://localhost:4444/wd/hub',

capabilities: { 'browserName': 'chrome' },

// This can be changed via the command line as:

// --params.login.user 'ngrocks'

params: {

login: {

user: 'protractor-br',

password: '#ng123#'

}

},

jasmineNodeOpts: { showColors: true }

};

## Using params

Your test

describe('login page', function() {

var params = browser.params;

it('should login successfully', function() {

element( by.model('username') ).sendKeys( params.login.user );

element( by.model('password') ).sendKeys( params.login.password );

element( by.css('[ng-click="login()"]') ).click();

expect( element(by.binding('username') ).getText() ).toEqual( params.login.user );

});

});

## Using jasmineNodeOpts

exports.config = {

seleniumAddress: 'http://localhost:4444/wd/hub',

capabilities: { 'browserName': 'chrome' },

jasmineNodeOpts: {

showColors: true,

defaultTimeoutInterval: 30000,

isVerbose: true,

includeStackTrace: true

}

};

## "But, I want to use Protractor in a non-AngularJS app"

Sorry, you can't! :(

## ust kidding, of course you can! :)

You only need to access the webdriver instance by using browser.driver:

browser.driver.findElement(by.css('[data-ptor="submit-btn"]'));

## It can be even more elegant

### In your config.js

onPrepare: function() {

global.dvr = browser.driver;

}

### In your test

dvr.findElement(by.css('[data-ptor="submit-btn"]'));

## Protractor waits for Angular to finish its work

Though you can tell it not to be that smart about your non-Angular app:

beforeEach(function() {

return browser.ignoreSynchronization = true;

});

## Let's make it more semantic?

### In your config.js

onPrepare: function() {

global.isAngularSite = function(flag) {

browser.ignoreSynchronization = !flag;

};

}

### In your test

beforeEach(function() {

isAngularSite(false); // isAngularSite(true), if it's an Angular app!

});