Angular JS:

•Creating Jasmine test cases to test the Angular JS code.

•By using Protractor framework performed End to End UI automation of Angular JS Web application.

•Used Angular JS and Node JS to structure JavaScript code in an MVC (Model, View, and Controller).

Java Script:

•Experience in using the JavaScript frameworks Angular.js, React.JS & Node.js

•Experience in using test-driven development in JavaScript

•By using Protractor, created the class files for each view using JavaScript.

•Tested many features in an agile environment using JavaScript

•Enhanced Test cases in Selenium using Java Programming features

**Salient features of the Protractor Automation tool:**

Built on the top of WebdriverJS and Selenium server.

Introduced new simple syntax to write tests.

Allows running tests targeting remote addresses.

Can take advantage of Selenium grid to run multiple browsers at once.

Can use Jasmine or Mocha to write test suites.

Protractor is a wrapper (built on the top) around Selenium WebDriver, so it contains every feature that is available in the Selenium WebDriver. Additionally, Protractor provides some new locator strategies and functions which are very helpful to automate the AngularJS application. Examples include things like: waitForAngular, By.binding, By.repeater, By.textarea, By.model, WebElement.all, WebElement.evaluate, etc.

It's a combination of Selenium WebDriver, NodeJS, Jasmine, and Cucumber.

Protractor is an end-to-end test framework for Angular and AngularJS applications. Protractor runs tests against your application running in a real browser, interacting with it as a user would.

These days, most applications use JavaScript, which increases the amount of difficult to test applications. So basically when we use the selenium its difficult to us to identify the web element. Protractor is a NodeJS program that is written in JavaScript, so it can easily identify web elements in an Angular JS application. The Angular JS application extends the HTML attribute and some extra attributes like (ng-model, ng-repeater, etc.), which is not included in Selenium WebDriver locators. So, Selenium is not able to identify these locators. That's why we use the Protractor for AngularJS applications.

So, for the Automation of angular applications, we need Selenium WebDriver and NodeJS.

Test Like a User

Protractor is built on top of WebDriverJS, which uses native events and browser-specific drivers to interact with your application as a user would.

For Angular Apps

Protractor supports Angular-specific locator strategies, which allows you to test Angular-specific elements without any setup effort on your part.

Automatic Waiting

You no longer need to add waits and sleeps to your test. Protractor can automatically execute the next step in your test the moment the webpage finishes pending tasks, so you don’t have to worry about waiting for your test and webpage to sync.

 The configuration file is used to define some parameters which will be passed to Protractor to execute our spec files. Please note:

1. Selenium server details (required)
2. Location of our spec files (required)
3. Browser capabilities for spec files (required)
4. Jasmine node configuration options (required)
5. Jasmine reporter configuration options (optional)

WRITE YOUR TEST CASES:

Create a new folder and name it “test cases” or anything as per your convenience.

·         Create a test case spec file. For example: “grid\_spec.js” file below, you can create multiple spec files as per your automation scope.

**grid\_spec.js**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | describe('HealthSense POC Grid', function() {           beforeEach(function() {             browser.get('http://Your Application URL');              ptor = protractor.getInstance();             });            it('should click on the grid link', function() {                 element(by.xpath('//a[contains(text(),"Grid")]')).click();             expect(element(by.xpath('//h1')).getText()).toEqual('Master Grid');             });            it('should enter ID in filter', function() {                     ptor.actions().sendKeys(protractor.Key.HOME).perform();            element(by.model('Model.ID')).sendKeys('10');            results = element.all(by.repeater('value in testValues'));                 expect(results.count()).toEqual(5);             element(by.model('Model.ID')).clear();            ptor.actions().sendKeys(protractor.Key.SPACE).perform();          });            it('should change the number of records per page to 10', function(){             element(by.xpath('//select')).click();              element(by.css('option[value="10"]')).click();              results1 = element.all(by.repeater('value in testValues'));            expect(results1.count()).toEqual(10);          });  }); |

·         Create a configuration: The configuration file is used to define some parameters which will be passed to Protractor to execute our spec files. Please note:

1. Selenium server details (required)
2. Location of our spec files (required)
3. Browser capabilities for spec files (required)
4. Jasmine node configuration options (required)
5. Jasmine reporter configuration options (optional)

Below is a sample configuration file:

So, for running the Angular JS application, we need two files. One is the configuration file, which describes which file will be run and the Selenium server address. The other file is the main file, where we describe locators which will interact with the application. Protractor, by default, uses Chrome driver for automation.

So, we will make two files: **conf.js**and **filename.js**.

**conf.js**:

exports.config = {

seleniumAddress: ‘http: //localhost:4444/wd/hub’,

specs: [‘filename.js’]

};

**filename.js**:

describe(‘angularjs homepage todo list’, function() {

it(‘should add a angular’, function() {

browser.get(‘https: //angularjs.org’);

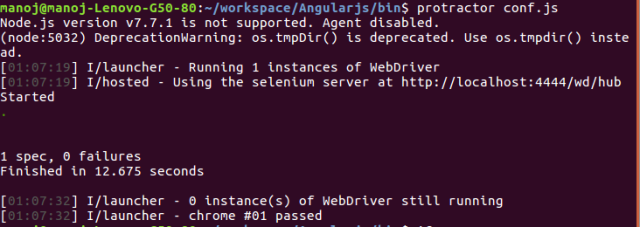
element(by.model(‘todoList.todoText’)).sendKeys(‘please fill you first test’); element(by.css(‘[value = ”add”]’)).click(); expect(browser.getTitle()).toContain(‘AngularJS’);

});

});

Now, we run the application with **$ protractor conf.js**.

It will open the Chrome browser, run the application, and show the results in terminal.



**Conf.js**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | exports.config = {      //The address of a running selenium server.    seleniumAddress: 'http://localhost:4444/wd/hub',      //Capabilities to be passed to the webdriver instance.    capabilities: {      'browserName': 'chrome'    },      //Specify the name of the specs files.    specs: ['grid\_spec.js'],      //Options to be passed to Jasmine-node.    jasmineNodeOpts: {        onComplete: null,        isVerbose: false,        showColors: true,        includeStackTrace: true    }  }; |

EXECUTE YOUR TESTS:

Once you are ready with your test cases and configuration file, the next step is to execute the test and see the results. Please follow these steps to execute the tests:

**Run selenium server**

* Make sure the Selenium server (which is specified in the configuration file) is running.

**Run tests**

Finally, you are ready to run the tests. Type the following command in the command prompt:

|  |  |
| --- | --- |
| 1 | protractor Conf.js  The test output should be 1 test, 3 assertions, 0 failures |

* You should be able to see the test execution progress and its status on the command line and also generate the test reports in an XML format. Please see the next section for details.

GENERATE REPORTS:

Protractor supports integration with Jasmine reporters. Jasmine reporters have a collection of JavaScript Jasmine reporter classes. We have used the JUnitXMLReporter which provided the test execution results in XML format. Please follow the following steps to configure the reporter.

**Install Jasmine reporter**

Install Jasmine reporter by typing the following command on command promptnpm install -g jasmine-reporters

**Update Configuration file**

After successful installation, create a report output directory inside your project folder and specify the report output directory name in the configuration file. Add the following code into your configuration

|  |  |
| --- | --- |
| 1  2 | file.require('jasmine-reporters');  jasmine.getEnv().addReporter(new jasmine.JUnitXmlReporter('outputdir/', true, true)); |
| 1  2  3  4 | <strong style="font-family: Arial, Verdana, sans-serif;">  XML output report  </strong><span style="font-family: Arial, Verdana, sans-serif;">  After execution of your tests again, you can find the test execution report XML in the  Output directory | |

how to extract data from Excel from protractor testing (give me some example codes using protractor)

Follow the below mentioned steps to set it up :

Install xlsjs  
npm install xlsjs  
Define a JavaScript utility function as :  
cellFromXLS = function (cellId) {  
‘use strict’;  
//Define sheetNumber  
var sheetNumber = 0;  
//Define file Path name  
var fileNamePath = path.join(dirPath, ‘data1.xls’);  
//NodeJs read file  
var XLS;  
if (typeof require !== ‘undefined’) {  
XLS = require(‘xlsjs’);  
}  
//Working with workbook  
var workbook = XLS.readFile(fileNamePath);  
var sheetNamelist = workbook.SheetNames;  
var value = workbook.Sheets[sheetNamelist[sheetNumber]][cellId].v;  
return value;  
};  
Call function as:  
var email = cellFromXLS(‘B1’);

# Why Use Protractor for End-to-End Testing of AngularJS Applications?

EmailTwitterFacebookLinkedInGoogle+Automation of testing frameworks helps you improve the quality, speed, and accuracy of the testing processes. The strategy to adopt [test automation](http://www.tothenew.com/testing/automated-independent-manual-testing) should clearly define when to opt for automation, its scope and selection of the right kind of tools for execution.

AngularJS framework is rapidly gaining momentum as it can quickly launch applications by extending the functionality of HTML. In this blog, we will discuss the key features and advantages of using Protractor as the end-to-end testing framework of [AngularJS applications](http://www.tothenew.com/front-end-angularjs-development" \o "Angular development). Protractor is a Node.js program that supports test frameworks like Jasmine, Mocha, and Cucumber.

**Key features of Protractor**

The following are some of the features of Protractor that make it worthy for [testing AngularJS applications](http://www.tothenew.com/blog/building-intuitive-frontend-interfaces-with-angularjs/).

* Protractor is a wrapper around WebDriverJS and supports behavior-driven development frameworks like Jasmine, Mocha, Cucumber etc.
* It offers some new locator strategies and functions that automate the testing of AngularJS applications. Protractor works in conjunction with Selenium to offer an automated test infrastructure.
* It makes use of Selenium grid to run multiple browsers at once.

**Advantages of Protractor**

* No need to add waits and sleeps: As the webpage finishes pending tasks, Protractor executes the next steps of the testing process by automatically connecting with the AngularJS application.
* Page Object: It is easy to set up page objects. Protractor does not perform WebDriver commands till an action is needed, i.e., set up page objects so that tests can operate page elements without moving the HTML.
* Angular-specific locator support: Protractor supports Angular-specific locator strategies including binding, model, repeater as well as native WebDriver locator strategies.

**Conclusion**: Protractor acts as an end-to-end testing framework for both Angular and non-Angular applications that does not use CSS attributes. Automating web applications built on AngularJS using Protractor is a major step forward for enterprises in their efforts to automate the business process applications.