# Worth 3% Course Bonus Mark - Submit this by March 1st, 6PM

Submit this to the inclass 4 drop box.

Late submissions are not permitted.

# E2E Testing

As applications grow, manual testing can become very difficult. End to end tests can help test the interface.

## Protractor

Protractor is a Node.js program that the AngularJS team built to perform end to end tests. Protractor uses WebDriver to run browsers and simulate user actions. Protractor also uses Jasmine syntax. Similar to unit testing, a test file is comprised of one or more *it* blocks that describe the requirements of your application. *it* blocks are made of commands and expectations.

* Commands tell Protractor to do something with the application such as navigate to a page or click on a button.
* Expectations tell Protractor to assert something.

### Document Locator Methods

To find objects in a document, these locator methods (in addition to many others) can be applied to locate items within the document:

by.binding('{{status}}')

by.className('redBtn')

by.css('.redBtn')

by.id('loginButton')

by.input("user")

by.linkText('Go Home')

by.partialLinktext('Home')

by.model('message')

by.name('email')

by.repeater("cat in pets")

by.repeater("cat in pets").row(1).column("{{cat.name}}"))

by.select("user")

by.selectedOption("red")

by.tagName('h2')

by.xpath('')

## Web Element Functions

To assist with testing, several additional functions can be used to extract data from elements and to perform actions on elements.

|  |  |
| --- | --- |
| **clear()** | If this element is a text entry element, this will clear the value. |
| **click()** | Click this element. |
| **getAttribute(name)** | Get the value of a given attribute of the element. |
| **getCssValue(propertyName)** | Get the value of a given CSS property. |
| **getLocation()** | Where on the page is the top left-hand corner of the rendered element? |
| **getSize()** | What is the width and height of the rendered element? |
| **getTagName()** | Get the tag name of this element. |
| **getText()** | Get the visible (i.e. not hidden by CSS) innerText of this element, including sub-elements, without any leading or trailing whitespace.  This does not work with the input element. Instead use the following:  var firstNameElement = element(by.model('firstName'));  var inputContents = firstNameElement.getAttribute('value'); |
| **isDisplayed()** | Is this element displayed or not? This method avoids the problem of having to parse an element's "style" attribute. |
| **isEnabled()** | Is the element currently enabled or not? This will generally return true for everything but disabled input elements. |
| **isSelected()** | Determine whether or not this element is selected or not. |
| **sendKeys(keysToSend)** | Use this method to simulate typing into an element, which may set its value. |

## Browser Functions

To run a test in a browser, the following functions can be used to direct a browser to a site or get information about the page title.

|  |  |
| --- | --- |
| **get()** - Gets url. | browser.get('http://ssdprogram.ca/protractor/test.html'); |
| **getTitle()** - Gets page title. | browser.getTitle() |

### Setting Up Protractor

Using GitBash as administrator at the node js directory. So on Windows navigate to:

**C:\Program Files\nodejs**

Then on Windows run the command:

**npm install –g protractor**

Followed by:

**npm install protractor**

Then, using GitBash, navigate to the bin folder where protractor exists:

**cd "C:\Program Files\nodejs\node\_modules\protractor\bin"**

Then run the following command to update webdriver-manager:

**webdriver-manager update**

Example 1: Testing the Protractor Install

🞑 Now, let’s test our Protractor install with a simple AngularJS application. To test with Protractor, you need to store your HTML file on a web server like Apache or you can use my AngularJS application which is hosted at http://ssdprogam.ca/protractor/test.html. Here is the code for it.

**test.html**

|  |
| --- |
| <!doctype html>  <html>  <head>  <title>Starting Angular</title>  </head>  <!-- The ng-app directive -->  <body ng-app='' ng-init="message='Hello world!'">  Message: <input type="text" ng-model="message">  <p ng-bind="message"></p>  <p> **{{**message**}}**</p>  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>  </body>  </html> |

After, create a test file and reference the html file on the web server. I have added in my address for testing. You will need to adjust this link if you are hosting your application locally.

**js/test.js**

|  |
| --- |
| describe('angularjs homepage todo list', function () {  it('should add a todo', function () {  // Add in your test url.  browser.get('http://ssdprogram.ca/protractor/test.html');  expect(browser.getTitle()).toEqual("Starting Angular");  var inputElement = element(by.model('message'));  // getText() does not work with input elements.  expect(inputElement.getAttribute('value')).toEqual('Hello world!');  expect(element(by.binding('message')).getText())  .toEqual('Hello world!');  });  }); |

Next, create a **conf.js** file inside the protractor bin directory. This configuration file needs to be placed in the folder at “C:\Program Files (x86)\nodejs\node\_modules\protractor\bin”.

Here is a copy of my conf.js file. This file references the default WebDriver port and server location plus the file path location of the tests. The default selenium address and port of 4444 are also listed. If you are ever uncertain what the address and port of WebDriver are though you will see it appear in the output when starting webdriver-manager as described in the next step.

|  |
| --- |
| exports.config = {  seleniumAddress: 'http://localhost:4444/wd/hub',  specs: ['C:/angTest/WebApplication1/WebApplication1/js/test.js']  }; |

Then, with GitBash, navigate to the bin folder where protractor exists:

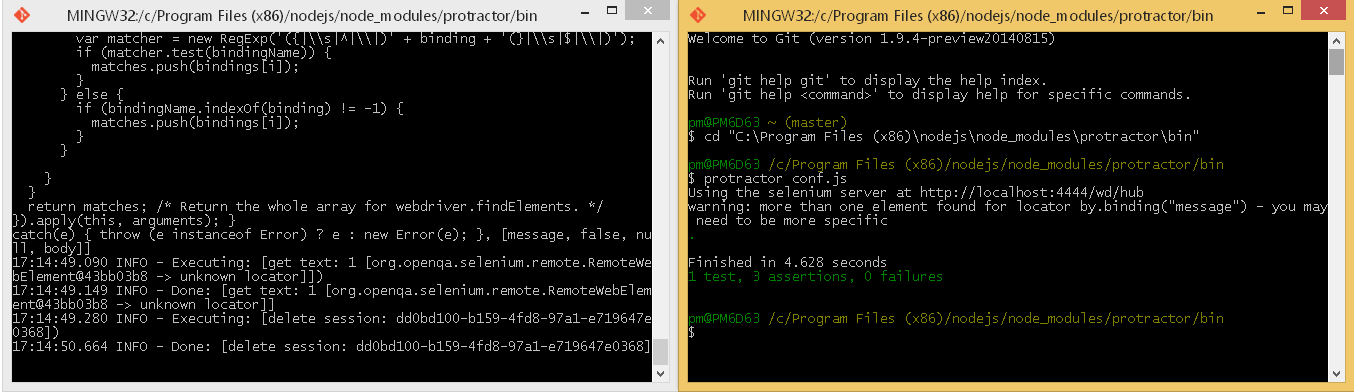
**cd "C:\Program Files\nodejs\node\_modules\protractor\bin"**

Then enter the command:

**webdriver-manager start**

In open another GitBash window and navigate to "C:\ProgramFiles\nodejs\node\_modules\protractor\bin". Enter the following command to launch your test:

**protractor conf.js**



When your test launches a browser will open and your test will be executed. If successful, a message should appear to indicate that the assertions have been successful.

Exercise 1

🖍 What AngularJS specific references are made to locate items in the test.html file during the end to end test in Example 1?

|  |
| --- |
| by.model  by.binding |

Exercise 2

🖍 Run the following code from a local web server.

|  |
| --- |
| <!doctype html>  <html>  <head>  <title>Check boxes</title>  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>  <script>  var formSample = angular.module('formSample', []);  formSample.controller('ExampleController', ['$scope', function ($scope) {  $scope.chk1 = true;  $scope.chk2 = 'Hello';  $scope.update = function () {  alert('hi');  }  }]);  </script>  </head>  <body ng-app="formSample">  <form name="myForm" ng-controller="ExampleController">  Value1: <input type="checkbox" ng-model="chk1" ng-change="update()"> <br />  Value2: <input type="checkbox" ng-model="chk2"  ng-true-value="'Hello'" ng-false-value="'Good-bye'"> <br />  <p>chk1 = <label ng-bind="chk1"></label> </p>  <p>chk2 = <label ng-bind="chk2"></label></p>  </form>  </body>  </html> |

Using protractor with jasmine syntax, write a test that:

* verifies the page title.
* verifies that the first checkbox is checked.
* verifies that the label which is bound to chk2 has a value of ‘Hello’.

In your test, remember to modify the location of the server to your local server where the page is hosted.

Show your test here:

|  |
| --- |
| describe('angularjs homepage todo list', **function** () {  it('should add a todo', **function** () {  browser.get('http://localhost:63342/ex1\_protractorIntroduction/index.html');  expect(browser.getTitle()).toEqual("Check boxes");  **var** inputElement = element(by.model('chk1'));  expect(inputElement.getAttribute('value')).toBeTruthy();  expect(element(by.binding('chk2')).getText()).toEqual('Hello');  }); }); |

Show a screenshot of your second Gitbash window which shows the test results:

|  |
| --- |
|  |

Example 2: Testing Sending Text, Clearing Inputs, and Button Clicks

🞑 Here is a more advanced example that shows how to test other controls on a web page. Note this test references a test2.html file which I have hosted on my website but you may wish to test it on a local server.

js/test.js

|  |
| --- |
| describe('Protractor tutorial.', function () {  // Added this helper function. getText() does not work with <input/> element.  var myGetText = function (element) {  var inputText = element.getAttribute('value');  return inputText;  }  it('Checking page items.', function () {  var firstNameElement = element(by.model('firstName'));  var nameInputText;  // Must get address from live server.  browser.get('http://ssdprogram.ca/protractor/test2.html');  // Check Title  expect(browser.getTitle()).toEqual('Starting Angular');  // Check input box contents.  nameInputText = myGetText(firstNameElement);  expect(nameInputText).toEqual('Josie');  // Check input box.  element(by.model('firstName')).sendKeys(' Louis');  nameInputText = myGetText(firstNameElement);  expect(nameInputText).toEqual('Josie Louis');  // Check submitted input in paragraph tag.  var btnName = element(by.id('btnName'));  btnName.click();  var elemGreeting = element(by.binding('submittedName'));  expect(elemGreeting.getText()).toEqual('Greetings: Josie Louis');  // Check cleared input box.  element(by.model('firstName')).clear();  nameInputText = myGetText(firstNameElement);  expect(nameInputText).toEqual('');  });  }); |

js/app.js

|  |
| --- |
| // Declare module that references our controllers.  var myApp = angular.module('myApp', ['myControllers']);  var myControllers = (function () {  var myControllers = angular.module('myControllers', []);  myControllers.controller('AppCtrl', ['$scope', function ($scope) {  $scope.title = "Angular JS!";  $scope.firstName = "Josie";  $scope.error = false;  $scope.submittedName = '';  $scope.numbers = ["1", "2", "3", "4", "5", "6", "7", "8", "9"];  $scope.add = function (data) {  $scope.submittedName = 'Greetings: ' + data;  }  }]);  return myControllers;  }()); |

This file is hosted at http://ssdprogram.ca/protractor/test2.html. However, you may wish to host it locally.

**test2.html**

|  |
| --- |
| <!doctype html>  <html>  <head>  <title>Starting Angular</title>  </head>  <!-- Notice here that ng-app references our 'myApp' module. -->  <body ng-app='myApp' ng-controller="AppCtrl">  <h1 ng-bind="title"></h1>  <!-- Name input. -->  <input type="text" id="nameInput" ng-model="firstName" />  <p id="greeting" ng-bind="submittedName"></p>  <button id="btnName" class="nameInput" ng-click="add(firstName)">Upddate</button>  <!-- Reference the application level controller for the title model. -->  <ul ng-repeat="number in numbers">  <li ng-bind="number"></li>  </ul>  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>  <script src="./js/app.js"></script>  </body>  </html> |

Exercise 3

🖍 See if you can verify if the number count is 9. You can do this by testing the repeater. Use syntax like the following ;

var history = element.all(by.repeater('result in memory'));

Show your revised test script here:

|  |
| --- |
| describe('Protractor tutorial.', **function** () {  // Added this helper function. getText() does not work with <input/> element.  **var** myGetText = **function** (element) {  **var** inputText = element.getAttribute('value');  **return** inputText;  }   it('Checking page items.', **function** () {  **var** firstNameElement = element(by.model('firstName'));  **var** nameInputText;   // Must get address from live server.  browser.get('http://localhost:63342/ex2\_protractorContinued/index.html');   // Check Title  expect(browser.getTitle()).toEqual('Starting Angular');   // Check input box contents.   nameInputText = myGetText(firstNameElement);  expect(nameInputText).toEqual('Josie');   // Check input box.  element(by.model('firstName')).sendKeys(' Louis');  nameInputText = myGetText(firstNameElement);  expect(nameInputText).toEqual('Josie Louis');   // Check submitted input in paragraph tag.  **var** btnName = element(by.id('btnName'));  btnName.click();  **var** elemGreeting = element(by.binding('submittedName'));  expect(elemGreeting.getText()).toEqual('Greetings: Josie Louis');   // Check cleared input box.  element(by.model('firstName')).clear();  nameInputText = myGetText(firstNameElement);  expect(nameInputText).toEqual('');   // Check number count.  **var** history = element.all(by.repeater('number in numbers'));  expect(history.count()).toEqual(9);  }); }); |