The information on [Protractor’s site](http://www.protractortest.org) is rather complete. The site not only displays how to setup, configure, to write and execute protractor tests it also provides a [style guide](http://www.protractortest.org/#/style-guide). Protractor is a [Node.js](https://nodejs.org/en/) program. To execute protractor, you will need to have Node.js installed. Protractor is built on top [WebDriverJS](https://github.com/SeleniumHQ/selenium/wiki/WebDriverJs). Java Development Kit (JDK) needs to be installed to run a local Selenium Server. Selenium WebDriver supports several browser implementations or drivers. The default behavior driven development (BDD) test framework in the Protractor install is [Jasmine](http://jasmine.github.io/). Below is the version of Protractor and supporting software installed on my personal computer.

|  |  |
| --- | --- |
| Description | Command |
| Version of Node | $ node --version  v4.4.5 |
| Version of npm | $ npm --version  2.15.5 |
| Version of Protractor (also the version of Jasmine and Selenium WebDriver) | $ protractor --version  Version 3.3.0  $ npm ls -depth=0  protractor@3.3.0 /usr/local/lib/node\_modules/protractor  ├── adm-zip@0.4.7  ├── chalk@1.1.3  ├── glob@6.0.4  ├── jasmine@2.4.1  ├── jasminewd2@0.0.9  ├── optimist@0.6.1  ├── q@1.4.1  ├── request@2.67.0  ├── saucelabs@1.0.1  ├── selenium-webdriver@2.52.0  └── source-map-support@0.4.0 |
| Version of Java | $ java -version  java version "1.8.0\_45"  Java(TM) SE Runtime Environment (build 1.8.0\_45-b14)  Java HotSpot(TM) 64-Bit Server VM (build 25.45-b02, mixed mode) |

Protractor needs a test file and a configuration file to run. This [reference.js](https://github.com/angular/protractor/blob/master/docs/referenceConf.js) shows all of the configuration options that may be passed to Protractor. The configuration file tells Protractor where the test file(s) is located and where to talk to the Selenium Server. The test files use the syntax of your test framework, for example Jasmine, and the [Protractor API](http://www.protractortest.org/#/api). Below demonstrates how to run a Protractor test.

|  |  |
| --- | --- |
| Description | Command |
| 1. Start Selenium Server by either: |  |
| a. Start Selenium WebDriver | webdriver-manager start |
| b. or update the configuration file to reference the selenium standalone server jar | Reference the selenium standalone server jar.  seleniumServerJar: './selenium\_server\_standalone/selenium-server-standalone-2.53.1.jar', |
| 2. Run Protractor Test | protractor conf.js |

Below is the Protractor configuration and test files used in the [ProtractorTutorial](https://github.com/gdombchik/ProtractorTutorial) GitHub repository.

|  |  |
| --- | --- |
| File | Description |
| conf.js | An example Protractor configuration file.  Protractor configuration options used:   * seleniumAddress: “To connect to a Selenium Server which is already running.” * capabilities:{‘browser’:}: “Protractor will launch specified browser. In our example we are launching a headless browser PhantomJS.” * framework: “Framework to use. Jasmine is the default test framework.” * specs: “The Protractor test files.” * jasmineNodeOpts: “Options to be passed to Jasmine.” |
| zoo\_spec\_2.js | A Protractor test file.  Use Protractor [element finder](http://www.protractortest.org/#/api?view=ElementFinder).  Use Protractor [browser.get](http://www.protractortest.org/#/api?view=Protractor.prototype.get) to navigate to a site.  Use Protractor Locator [model](http://www.protractortest.org/#/api?view=ProtractorBy.prototype.model) to find an element by ng-model.  Use Protractor Locator [bind](http://www.protractortest.org/#/api?view=ProtractorBy.prototype.binding) to find an element by ng-bind. |
| zoo\_spec\_3.js | Build on zoo\_spec\_2.js Protractor test file.  Use Protractor Locator [buttonText](http://www.protractortest.org/#/api?view=ProtractorBy.prototype.buttonText) to find a button by text.  Use Protractor Locator [click](http://www.protractortest.org/#/api?view=webdriver.WebElement.prototype.click) to schedule a command to click on an element.  Use Protractor Locator [$](http://www.protractortest.org/#/api?view=ElementFinder.prototype.$) to find elements within a parent.  Use Protractor Locator [partialButtonText](http://www.protractortest.org/#/api?view=ProtractorBy.prototype.partialButtonText) to find a button by partial text. |
| zoo\_spec\_4.js | Build on zoo\_spec\_3.js Protractor test file.  Use Jasmine Expectations [expect](http://jasmine.github.io/2.0/introduction.html) statements to test the state of the code. |
| zoo\_spec\_5.js | Build on zoo\_spec\_4.js Protractor test file.  Use Jasmine [beforeEach](http://jasmine.github.io/2.0/introduction.html) function to be called once before each spec.  Use Jasmine [afterEach](http://jasmine.github.io/2.0/introduction.html) function to be called once after each spec. |
| zoo\_spec\_6.js | A Protractor test file.  User Protractor Locator [css](http://www.protractortest.org/#/api?view=webdriver.By.css) to locate elements using a css selector. |
| zoo\_spec\_7.js | A Protractor test file.  Use the [Page Objects](http://www.protractortest.org/#/style-guide) design pattern.  Reasons to use the Page Objects design pattern as specified by the Protractor style guide:   * Encapsulate information about the elements on the page under test * They can be reused across multiple tests * Decouple the test logic from implementation details |
| protractorsite/conf\_protractor-jasmine2-screenshot-reporter.js | An example Protractor configuration file.  Uses the [protractor-jasmine2-screenshot-reporter](https://www.npmjs.com/package/protractor-jasmine2-screenshot-reporter) node package. Reporter for Protractor that will capture a screenshot after each executed test case and store the results in a HTML report. (supports jasmine2) |

In reviewing how to run the Protractor tests using a continuous integration (ci) tool such as Jenkins, an option is to use a shell script or a JavaScript task runner such as Grunt in the ci to start the selenium server, run the tests using a headless environment using tools such as PhantomJS, capture test results, and stopping the selenium server.

|  |  |
| --- | --- |
| Description | Command |
| Installation of nodejs on AWS E2 Ubuntu server | $ curl -sL https://deb.nodesource.com/setup\_4.x | sudo -E bash -  $ sudo apt-get install -y nodejs  $ npm –version  2.15.8 |
| Global Installation of Protractor and webdriver-manager | $ sudo npm install -g protractor  $ protractor --version  Version 4.0.2  $ sudo webdriver-manager update |
| Global Installation of PhantomJS | $ sudo npm install -g phantomjs-prebuilt  $ phantomjs --version  2.1.1 |
| Update the configuration file to reference PhantomJS | Update conf.js and change the browserName to phantomjs.  capabilities: {  'browserName': 'phantomjs' }, |
| Update the configuration file to reference the selenium standalone server | Reference selenium server standalone jar.  seleniumServerJar: './selenium\_server\_standalone/selenium-server-standalone-2.53.1.jar', |
| Add shell script that executes the protractor test | a. Contents of /selenium\_server\_standalone  /execute\_protractor\_tests.sh  #!/bin/bash  protractor ../conf.js  b. Execute execute\_protractor\_tests.sh  $ ./ execute\_protractor\_tests.sh |

In summation, a front-end test framework can be built by installing Protractor and supporting software, generate Page Objects to represent the web application, build test or specs referencing the generated Page Objects in the Protractor and Jasmine test framework, and then porting the solution into a continuous integration environment.