Protractor

* Protractor is a Node JS program
* **end to end testing framework for Angular JS based applications.**
* Combines powerful technologies like Selenium WebDriver and Jasmine.
* Jasmine is a behaviour-driven testing framework for JavaScript Programming language.
* It is intended not only to test AngularJS application but also for writing automating regression tests for normal Web Applications as well.
* written in JavaScript.
* Runs with Node to identify the web elements in AngularJS applications
* It also uses WebDriver to control the browser with user actions.
* Protractor supports Angular-specific locator strategies, which allows you to test Angular-specific elements without any setup effort on your part.
* 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.

Need to use Protractor Framework

* JavaScript is used in only almost all web applications. As the applications grows, JavaScript also increases in size and complexity. In such case, it becomes difficult task for Testers to test the web application for various scenarios.
* Sometimes it is difficult to capture the web elements in AngularJS applications using JUnit or Selenium WebDriver.

AngularJS application

* AngularJS applications are Web Applications which uses **extended HTML's syntax** to express web application components. Angular JS applications have some extra HTML attributes like ng-repeater, ng-controller, ng-model .., etc. which are not included in Selenium locators.
* It is mainly used for dynamic web applications.
* These applications use less and flexible code compared with normal Web Applications.
* Selenium is not able to identify those web elements using Selenium code. So, Protractor on the top of Selenium can handle and controls those attributes in Web Applications.
* To start building your Node.js applications, the first step is the installation of the node.js framework. The Node.js framework is available for a variety of operating systems right from Windows to Ubuntu and OS X. Once the Node.js framework is installed you can start building your first Node.js applications.

Node.js and Angular JS

**Node**.**js** is a runtime environment useful in building server-side applications while **AngularJS** is a JavaScript framework mainly useful in building client-side part of applications which run inside a web browser.

* Protractor is a **Node JS program.**
* The Protractor is an end to end testing framework for **Angular JS based applications**

https://docs.angularjs.org/api/ng

Installing protractor

Before we start Protractor, we need to install the following:

* Selenium WebDriver
* NPM (Node.js)

<http://www.guru99.com/download-install-node-js.html>

* After installing Selenium WebDriver and NPM, install protractor

http://www.guru99.com/protractor-testing.html

Open command prompt and type **"npm install –g protractor"** and hit Enter**.**

Check the installation and version using **"Protractor –version”**

**Update the web driver manager “webdriver-manager update”**

**Start the web driver manager “webdriver-manager start”**

spec file and configuration file

C:\Users\beena\AppData\Roaming\npm\node\_modules\protractor

Protractor needs 2 files to run, a **spec** file and **configuration** file.

* **Configuration file**: This File helps protractor to where the test files are placed (specs.js) and to talk with Selenium server (Selenium Address). Chrome is the default browser for Protractor.
* **Spec file:** This File containsthe logic andlocatorsto interact with the application**.**

Execution of Code

* Open the command prompt
* Make sure selenium web driver manager is up and running. For that give command as "**webdriver-manager start**" and hit Enter**.**
* **Open a new command prompt** and give the command as **"protractor conf.js"** to run the configuration file.

**C:\Users\beena\AppData\Roaming\npm\node\_modules\protractor\example**

**example contains conf.js and spec.js**

Protractor uses [Jasmine](http://jasmine.github.io/1.3/introduction.html) for its test syntax. As in 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 about the application's state, such as the value of a field or the current URL.

If any expectation within an it block fails, the runner marks the it as "failed" and continues on to the next block.

Test files may also have beforeEach and afterEach blocks, which will be run before or after each it block regardless of whether the block passes or fails.

<http://www.protractortest.org/#/api?view=ProtractorBy>

The Protractor Locators. These provide ways of finding elements in Angular applications by binding, model, etc.

| **Function** | **Description** |
| --- | --- |
| [addLocator](http://www.protractortest.org/#/api?view=ProtractorBy.prototype.addLocator) | Add a locator to this instance of ProtractorBy. |
| [binding](http://www.protractortest.org/#/api?view=ProtractorBy.prototype.binding) | Find an element by text binding. |
| [exactBinding](http://www.protractortest.org/#/api?view=ProtractorBy.prototype.exactBinding) | Find an element by exact binding. |
| [model](http://www.protractortest.org/#/api?view=ProtractorBy.prototype.model) | Find an element by ng-model expression. |
| [buttonText](http://www.protractortest.org/#/api?view=ProtractorBy.prototype.buttonText) | Find a button by text. |
| [partialButtonText](http://www.protractortest.org/#/api?view=ProtractorBy.prototype.partialButtonText) | Find a button by partial text. |
| [repeater](http://www.protractortest.org/#/api?view=ProtractorBy.prototype.repeater) | Find elements inside an ng-repeat. |
| [exactRepeater](http://www.protractortest.org/#/api?view=ProtractorBy.prototype.exactRepeater) | Find an element by exact repeater. |
| [cssContainingText](http://www.protractortest.org/#/api?view=ProtractorBy.prototype.cssContainingText) | Find elements by CSS which contain a certain string. |
| [options](http://www.protractortest.org/#/api?view=ProtractorBy.prototype.options) | Find an element by ng-options expression. |
| [deepCss](http://www.protractortest.org/#/api?view=ProtractorBy.prototype.deepCss) | Find an element by css selector within the Shadow DOM. |

#### **Extends webdriver.By**

| **Function** | **Description** |
| --- | --- |
| [className](http://www.protractortest.org/#/api?view=webdriver.By.className) | Locates elements that have a specific class name. |
| [css](http://www.protractortest.org/#/api?view=webdriver.By.css) | Locates elements using a CSS selector. |
| [id](http://www.protractortest.org/#/api?view=webdriver.By.id) | Locates an element by its ID. |
| [linkText](http://www.protractortest.org/#/api?view=webdriver.By.linkText) | Locates link elements whose [visible text](http://www.protractortest.org/#/api?view=webdriver.WebElement.prototype.getText) matches the given string. |
| [js](http://www.protractortest.org/#/api?view=webdriver.By.js) | Locates an elements by evaluating a JavaScript expression, which may be either a function or a string. |
| [name](http://www.protractortest.org/#/api?view=webdriver.By.name) | Locates elements whose name attribute has the given value. |
| [partialLinkText](http://www.protractortest.org/#/api?view=webdriver.By.partialLinkText) | Locates link elements whose [visible text](http://www.protractortest.org/#/api?view=webdriver.WebElement.prototype.getText) contains the given substring. |
| [tagName](http://www.protractortest.org/#/api?view=webdriver.By.tagName) | Locates elements with a given tag name. |
| [xpath](http://www.protractortest.org/#/api?view=webdriver.By.xpath) | Locates elements matching a XPath selector. |