Introduction to Protractor – Pluralsight

# Quick Start: Start Up Processes

1. > **cd** c:\Bob\dev\pluralsight\intro-protractor\intro-to-protractor\protractor
2. **> nvm use 6.5.0** *(sets node version)*
3. > **Grunt** debug
4. > **webdriver**-manager start
5. >**mongod** **--dbpath** C:\Bob\dev\pluralsight\intro-to-protractor\intro-to-protractor\db
6. >**protractor** conf.js

# E2E vs Unit Test

## Unit Testing

Interaction with a sing, Isolated unit

Focuses on the functionality of the code

## E2E

Interact with the entire app

Focused on user experience

Runs in browser

# Installation

Verify Java is installed in cmd prompt:

* Java
* %JAVA\_HOME%

Or download at <https://java.com/en/download/manual.jsp>

## Use npm to install

**npm install -g protractor**

\*\*\* the npm install was looking for ***nodejs 4.2.x***

\*\*\* I installed 4.2 via NVM 🡪 **nvm install 4.2**

## Verify installation

* protractor –version

Version 4.0.4

## Update Webdrive Manager

* webdriver-manager update

***output***:

[16:49:24] I/file\_manager - creating folder C:\Users\mazzo\AppData\Roaming\nvm\v

0.12.0\node\_modules\protractor\node\_modules\webdriver-manager\selenium

[16:49:24] I/downloader - selenium standalone: downloading **version 2.53.1**

[16:49:24] I/downloader - curl -o C:\Users\mazzo\AppData\Roaming\nvm\v0.12.0\nod

e\_modules\protractor\node\_modules\webdriver-manager\selenium/selenium-server-sta

ndalone-2.53.1.jar https://selenium-release.storage.googleapis.com/2.53/selenium

-***server-standalone-2.53.1.jar***

[16:49:24] I/downloader - chromedriver: downloading version 2.22

[16:49:24] I/downloader - curl -o C:\Users\mazzo\AppData\Roaming\nvm\v0.12.0\nod

e\_modules\protractor\node\_modules\webdriver-manager\selenium/chromedriver\_2.22wi

n32.zip https://chromedriver.storage.googleapis.com/2.22/chromedriver\_win32.zip

[16:49:26] I/update - chromedriver: unzipping C:\Users\mazzo\AppData\Roaming\nvm

\v0.12.0\node\_modules\protractor\node\_modules\webdriver-manager\selenium\chromed

river\_2.22win32.zip

## Start Protractor and launch browser

* webdriver-manager start

Browers: <http://localhost:4444/wd/hub/static/resource/hub.html>

## Installing Sample App

## Clone git repo (from Nate Taylor)

Clone git from: <https://github.com/taylonr/intro-to-protractor>

* git clone <https://github.com/taylonr/intro-to-protractor.git>
* npm install
* bower install

\*\*\* ***Make sure MongoDB is running in a separate cmd prompt***:

* C:\Bob\dev\pluralsight\intro-protractor\intro-to-protractor>**mongod** -**dbpath** C:\Bo

b\dev\pluralsight\intro-protractor\intro-to-protractor\db

## Installing the MEAN.JS app separately (not necessary for this course)

### Clone the MEAN.JS boilerplate template for testing

* git clone https://github.com/meanjs/mean.git meanjs
* npm install (this installs package dep., see package.json)

### Using the MEAN generator

* as per <http://meanjs.org/generator.html>
* npm install -g generator-meanjs

## Install nodejs dependencies

* npm install

## MEAN Stack

## Mongo

1. Created new ‘db’ dir
2. Launch Mongo
   1. >**mongod** **--dbpath** C:\Bob\dev\pluralsight\intro-to-protractor\db

# Module: Writing our first test

## Testing that the page is loaded

Create a new “***protractor***” folder (..\***intro-protractor\intro-to-protractor\protractor\***), and a “***ratings***” folder below that.

Create new file ***event.list.item.spec.js*** below ratings folder.

We write our first testing here against views:

..\public\modules\ratings\views\**event.list.html**

..\public\modules\ratings\views\**event.details.html**

\*\*\* see ..\public\modules\ratings\config\ratings.routes.js 🡪 “***events***” **route**

## Configure selenium port in conf.js

See “Testing that the page is loaded” (**at 6min**) under “Writing Our first test” module

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

## Run Protractor test, from protractor folder:

..\intro-to-protractor\protractor>**protractor conf.js**

# Types of Tests

## Unit Test, Integrations Test, User Tests

Protractor falls btwn Integr and User Tests

Similarity with Integration Tests

Nothing is mocked

Not limited to a unit test

Differences with Integration Tests

Integration tests are usually aware of the rest of the code, but with Protractor we are not accessing data layers.

Similarity with User Tests

Interacts with UI

Simulates a user

We are testing workflows (i.e. adding a new event)

Differences with User Tests

Protractor enters the same data format each time, whereas user may not be as consistent.

# Module: Locators & Test Suites

## Specific Locators (by.\*)

by.binding(‘name’) // gets the element

by.model(‘name’) // looks for the ng-model

by.css(‘.primary.header’);

by.buttonText(‘Save’)

by.repeater(‘u in users’)

by.options(‘r for r inroles’) //i.e. <select ng-options=’r for r in roles’>

by.id

by.linkText // gets the link based on the text of the link

by.name() // gets by element name prop

by.tagName // pass in an html tag name

by.xPath // first element, etc.

by.addLocator // custom locator

## Creating and Executing Test Suites

### Add a suites object to conf.js

The idea here is to setup some brief “smoke” tests that will run quickly. These smoke tests can be run quickly as a sanity check, then longer tests can be run afterwards.

* Suite example

suites:{  
 **smoke**: **'./smoke/\*.spec.js'**,  
 **longRunning**: **'./ratings/a.bunch.of.long.tests.spec.js'**,  
 **full**: **'./\*\*/\*.spec.js'**}

* Running a suite (specify either one or multiple suite objects) :

>**protractor** conf.js --suite=smoke

>**protractor** conf.js --suite=smoke,longRunning

# Module: Page Objects

## Common problems with UI Testing

* The UI changes
  + i.e. buttons get moved to a different area of screen
* Lots of repetition in test scripts
  + Element(by.model(‘..’)
* Test maintenance headaches
  + Model name changes; we must search tests to now replace the name
* Broken Tests
  + We must deal with assertion failures; i.e. a model name changes

## Page Objects

An easier way to access all the page elements by exporting page objects from a page file.

Instead of going into the spec files, we can edit the page file which represents our model.

* We create a new “pages” folder
  + Create a new event.page.js file
  + Export using “module.exports”
* In the spec.js test file, we access the page file
  + **var** CreatePage = **require**(**'./pages/create.event.page.js'**);