An Introduction to AngularJS End to End Testing using Protractor

What is Protractor

- An AngularJS E2E Testing Framework
- Introduced during AngularJS 1.2 and Beyond presentation
- A new replacement of the existing E2E Testing framework

What's Different

- Built on WebdriverJS and Selenium Server
- New syntax when writing tests
- Allows running tests targeting remote addresses, No longer need to have test files on the server being tested
- Can take advantage of Selenium Grid to run multiple browsers at once; ie Sauce Labs
- Has it's own runner, no need for Karma
- Can use Jasmine or Mocha to write test suites

Installing Protractor

- Installed via Node Package Manager

```
npm install protractor
```

- Install Selenium Standalone Server

```
node_modules/protractor/bin/install_selenium_standalone
```

Note: The Standalone Server is not needed if you are going to run against an existing Selenium Grid or Installation

Note: Windows users need to execute second command from node

The Configuration File

There are five things you need to provide so Protractor can run your specs:

1. Selenium Server Configuration
2. Where your spec files are located
3. Browser capabilities required by spec files
4. The Base Url for your spec files
5. Jasmine Node Configuration

Selenium Configuration

There are three ways to specify what Selenium Server to use:

1. seleniumServerJar and seleniumPort - to start Selenium Standalone locally.
2. seleniumAddress - to connect to a Selenium server which is already running.
3. sauceUser/sauceKey - to use remote Selenium servers via SauceLabs.

Selenium Configuration (cont.)

- **seleniumArgs** - You can also pass command line arguments to the Selenium Server by using the parameter.

- **chormeDriver** - Provides a path to the web driver for chrome to the Standalone Selenium Server, if not provided PATH is used.

- **capabilities** - List of Browser Capabilities you require for the test. See https://code.google.com/p/selenium/wiki/DesiredCapabilities

Other Configuration

- **specs** - an array of file patterns that point to your spec files. Patterns are relative to the current working directory when Protractor is started up.

```
specs: [‘test/scenarios/*-scenarios.js’]
```

- **baseUrl** - A base URL for your application under test. Calls to protractor.get() with relative paths will be prepended with this.

```
baseUrl: 'http://localhost:9000'
```

- **jasmineNodeOpts** - an array of options for jasmine node output.

```
onComplete: function to call before the driver quits.
isVerbose: provide verbose output during spec runs.
showColors: provide colored output during spec runs.
includeStackTrace: include a stack trace on errors.
```

Basic Test Structure

```
var util = require('util');

describe('Adjunct List', function () {
    var ptor;

    beforeEach(function () {
        ptor = protractor.getInstance();
        ptor.get('#/');
    });

    it('should do something', function () {
        ptor = protractor.getInstance();
        ptor.findElement(protractor.By.className('brand')).click();
        expect(ptor.getCurrentUrl()).toContain('#/');
    }, 10000);
});
```

Finding Elements in a Test

Use Selenium WebdriverJS Syntax:

```
ptor.findElement(protractor.By.x('...'));

or

ptor.findElements(protractor.By.x('...'));
```

findElement returns a single element, findElements returns an array of elements.

Both will throw an exception if the locator cannot find the element on the page

Protractor Locators

protractor.By.className('redBtn')
protractor.By.css('.redBtn')
protractor.By.id('loginButton')
protractor.By.linkText('Go Home')
protractor.By.partialLinktext('Home')
protractor.By.name('email')
protractor.By.tagName('h2')
protractor.By.xpath('')

Protractor Locators (cont)

protractor.By.binding('{{status}}')
protractor.By.select("user")
protractor.By.selectedOption("red")
protractor.By.input("user")
protractor.By.repeater("cat in pets")
protractor.By.repeater("cat in pets").row(1).column("{{cat.name}}"))

WebElement Methods

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 the 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.

WebElement Methods (cont)

getText()	Get the visible (i.e. not hidden by CSS) innerText of this element, including sub-elements, without any leading or trailing whitespace.
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.

Migrating Existing Test Scripts

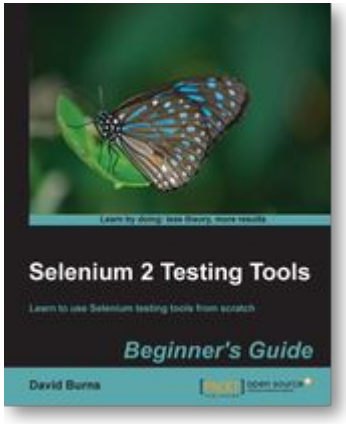
browser().navigateTo(url)	ptor.get(url)
browser().location().url()	ptor.getCurrentUrl()
binding(name)	ptor.findElement(protractor.By.binding('{{status}}')).getText()
input(name).enter(value)	ptor.findElement(protractor.By.input("user")).sendKeys(value)
input(name).check()	ptor.findElement(protractor.By.input("user")).click();
input(name).select(value)	See Select below.
input(name).val()	ptor.findElement(protractor.By.input("user")).getText()

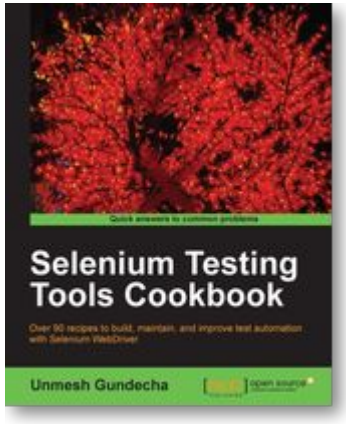
Migrating Existing Test Scripts(cont)

repeater(selector, label).count()	ptor.findElements(protractor.By.repeater("cat in pets")).length()
repeater(selector, label).row(index)	ptor.findElements(protractor.By.repeater("cat in pets")).row(index)
repeater(selector, label).column(binding)	ptor.findElements(protractor.By.repeater("cat in pets")).row(index)..column(binding)
select(name).option(value)	ptor.findElement(protractor.By.id(‘selectId’).click(); ptor.findElement(protractor.By.css(‘option [value=”0”]’’).click();
select(name).options(value1, value2...)	ptor.findElement(protractor.By.id(‘selectId’).click(); ptor.findElement(protractor.By.css(‘option [value=”0”]’’).click(); ptor.findElement(protractor.By.css(‘option [value=”2”]’’).click(); ptor.findElement(protractor.By.css(‘option [value=”4”]’’).click();
element(selector, label)	See WebElement Methods Mentioned Earlier

Demo

Resources





Protractor Github Page - https://github.com/angular/protractor

WebdriverJS User Guide - https://code.google.com/p/selenium/wiki/WebDriverJs

Stackoverflow WebdriverJS Content - http://stackoverflow.com/search?q=%5Bselenium%5D+webdriverjs