

# Using Protractor to Build Automated UI “Smoke Tests”

---

DALE SPOONEMORE

[DALE@SEEDTOSPOON.NET](mailto:DALE@SEEDTOSPOON.NET)

@DALESPOONEMORE

[WWW.SEEDTOSPOON.NET](http://WWW.SEEDTOSPOON.NET)

A solid blue horizontal bar at the bottom of the slide.

# Why Have Automated Tests?

---

- Smoke tests
  - Know when apps are down or having performance issues
- Functional UI tests
  - Tests that are tedious and time consuming
  - Testing with lots of data

# But I Have Unit Tests...

---

- Unit Tests only test that part of the code
- UI tests launch browser and interacts with the browser as a user
- Tests database, UI, server-side code, server/network hardware

# The Tools

---

- Webdriver

<http://www.seleniumhq.org/projects/webdriver/>

- Selenium

<http://www.seleniumhq.org/>

- Protractor

<http://www.protractortest.org/#/>

# Setup

---

Install Node.JS

<https://nodejs.org>

## **Command Line Statements**

```
npm install protractor -g  
webdriver-manager update  
webdriver-manager start
```

# Protractor Config

---

```
//conf.js
```

```
exports.config = {  
  framework: 'jasmine',  
  seleniumAddress: 'http://localhost:4444/wd/hub',  
  specs: ['spec.js'],  
  getPageTimeout: 30000,  
  allScriptsTimeout: 30000,  
  onPrepare: function() {  
    browser.driver.manage().window().maximize();  
    browser.ignoreSynchronization=true; //for non-angular apps  
  },  
};
```

Full list of options

<https://github.com/angular/protractor/blob/master/lib/config.ts>

# Test script

---

```
// spec.js
```

```
describe('From Seed to Spoon Smoke Test', function() {  
  it('should verify the main page loads', function() {  
    browser.get('http://www.seedtospoon.net/');  
    expect(element(by.css('h2.maintitle')).getText()).toEqual('Grow Your Own Food');  
  });  
});
```

Full list of capabilities

<http://www.protractortest.org/#/api>

# Controlling the Browser

---

```
browser.get('http://www.seedto spoon.net');  
browser.navigate().back();  
browser.navigate().forward();  
browser.sleep(10000);  
browser.getLocationAbsUrl() // get the current address  
browser.ignoreSynchronization = true; // non-angular
```



# Using Locators to Find Elements

---

```
element(by.id('user_name'))
element(by.css('div.blog-post .entry-title'))
element (by.input( 'username' ));
element(by.buttonText('Save'));
element(by.partialButtonText('Save'));
element(by.linkText('Save'));
element(by.partialLinkText('Save'));
```

Angular-specific

```
element(by.model('person.name')) // refers to ng-model directive
element(by.binding('person.concatName')); // refers to ng-bind directive
```

# Collections of Elements

---

```
var list = element.all(by.css('.items'));  
expect(list.count()).toBe(3);  
expect(list.get(0).getText()).toBe('First')  
expect(list.get(1).getText()).toBe('Second')  
expect(list.first().getText()).toBe('First')  
expect(list.last().getText()).toBe('Last')
```

# Elementor

---

Elementor

<https://github.com/andresdominguez/elementor>

# Actions

---

## **Click**

```
element (by.buttonText ('Submit')) .click();
```

## **Use Keyboard**

```
element (by.id ('user_name')) .sendKeys ("user1");  
sendKeys (protractor.Key.ENTER);  
sendKeys (protractor.Key.TAB);
```

## **Clear a field**

```
element (by.id ('user_name')) .clear()
```

# Checking Data/Visibility

---

```
element(by.id('create')).isDisplayed() //Is element currently visible/displayed?  
expect(element(by.css('h2.maintitle')).getText()).toEqual('Grow Your Own Food');
```

# Demo

---

# Enhancements

---

- **Multiple Browsers**

```
multiCapabilities: [{  
  'browserName': 'firefox'  
}, {  
  'browserName': 'chrome'  
}]
```

- **Screenshots**

protractor-screenshoter-plugin

<https://www.npmjs.com/package/protractor-screenshoter-plugin>

- **Generate test data files**

jasmine-trx-reporter

<https://www.npmjs.com/package/jasmine-trx-reporter>

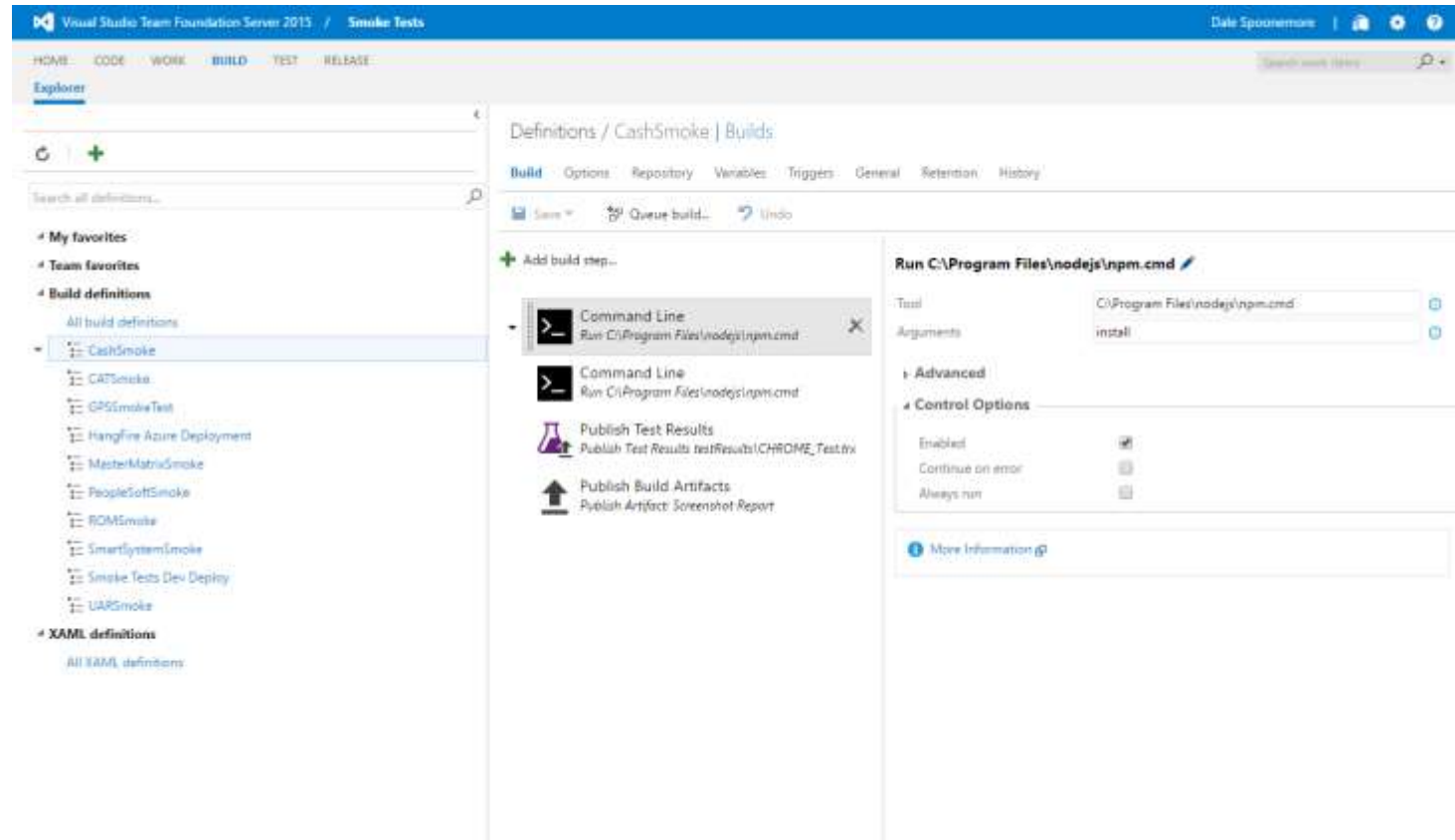
# Automating test runs

---

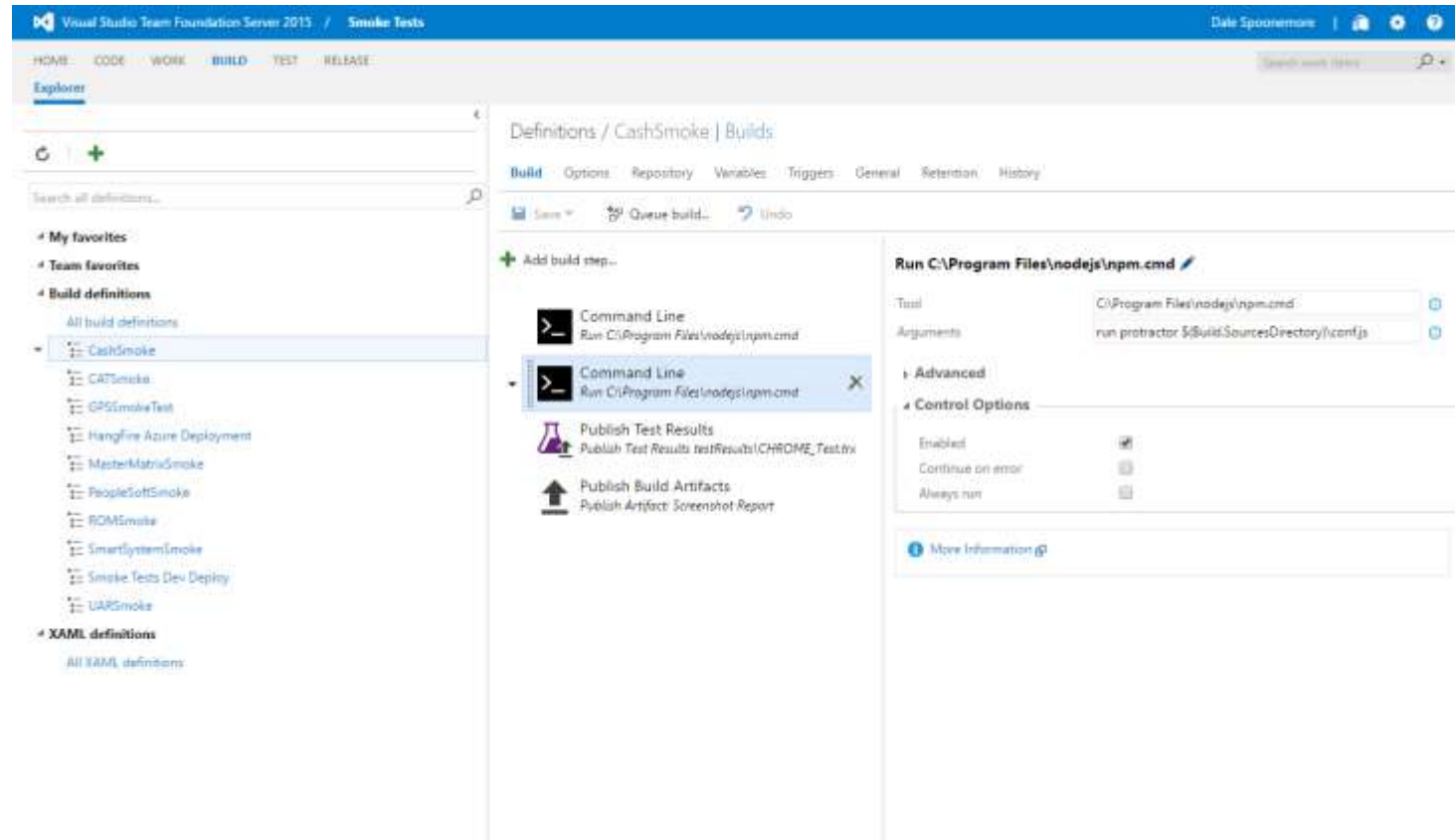
- TFS
- Jenkins / Grunt / Gulp
  - <https://www.linkedin.com/pulse/automating-angular-apps-using-protractor-integrating-continuous>



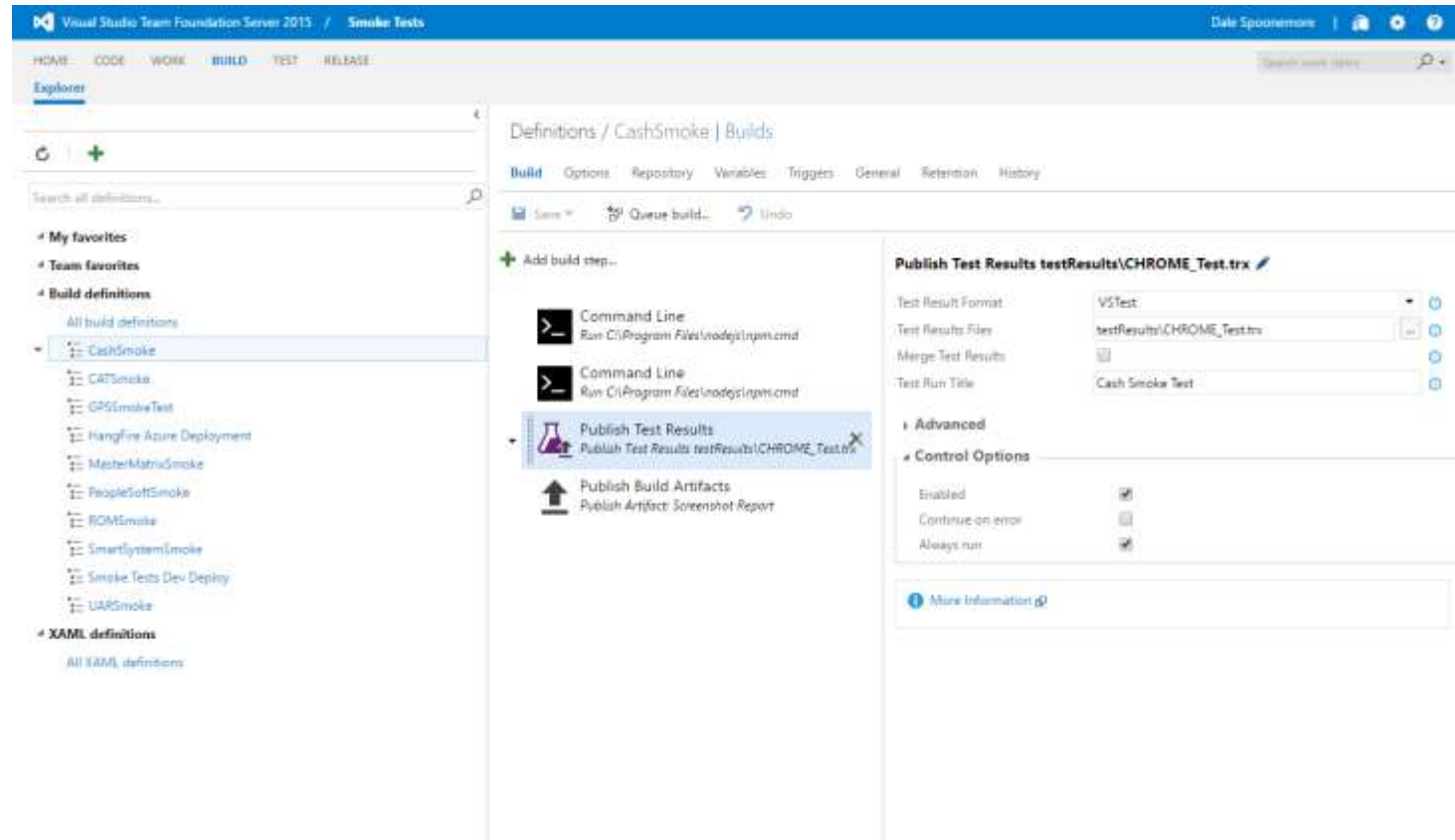
# TFS build steps – Run install



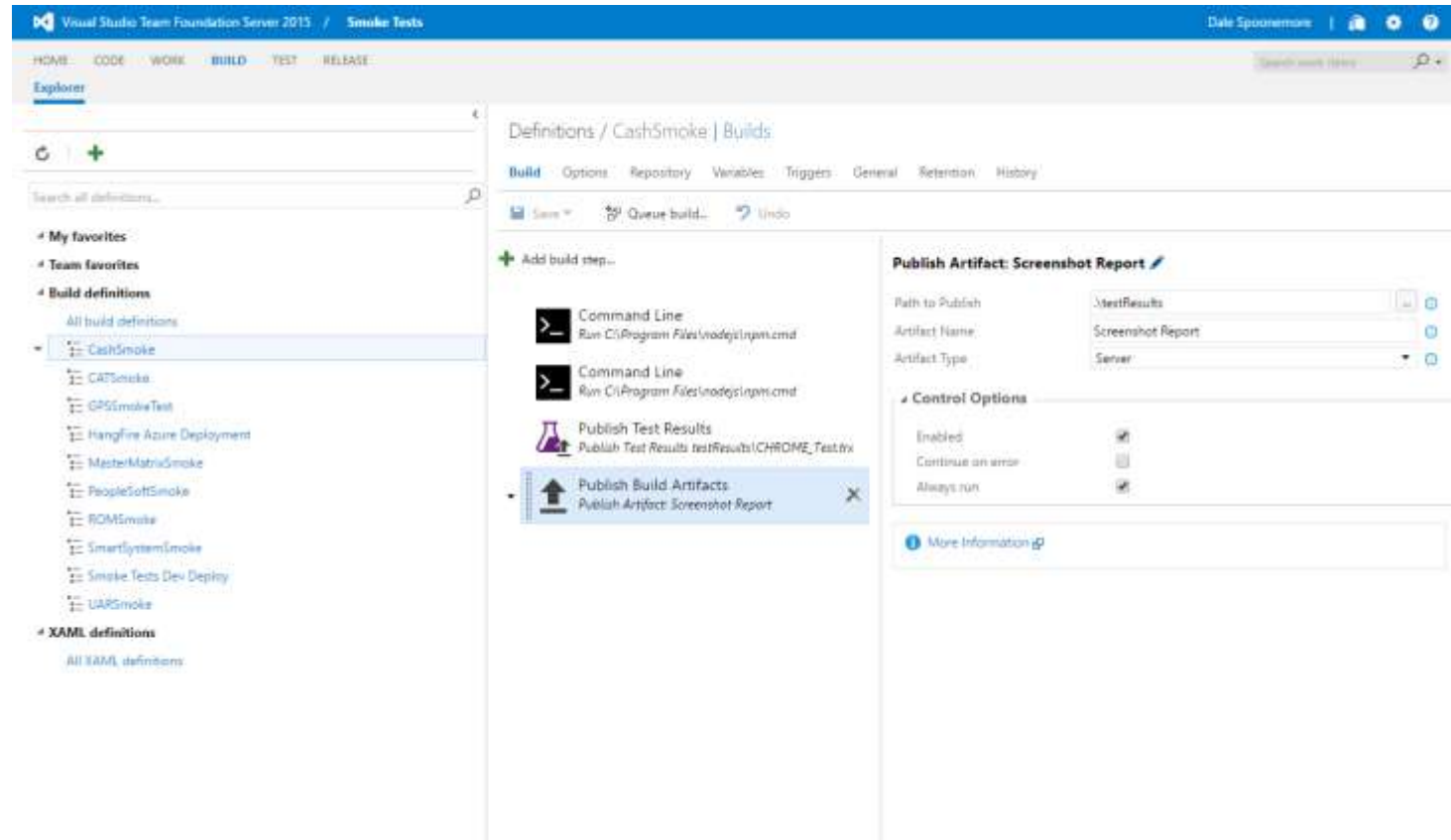
# TFS build steps – Launch test



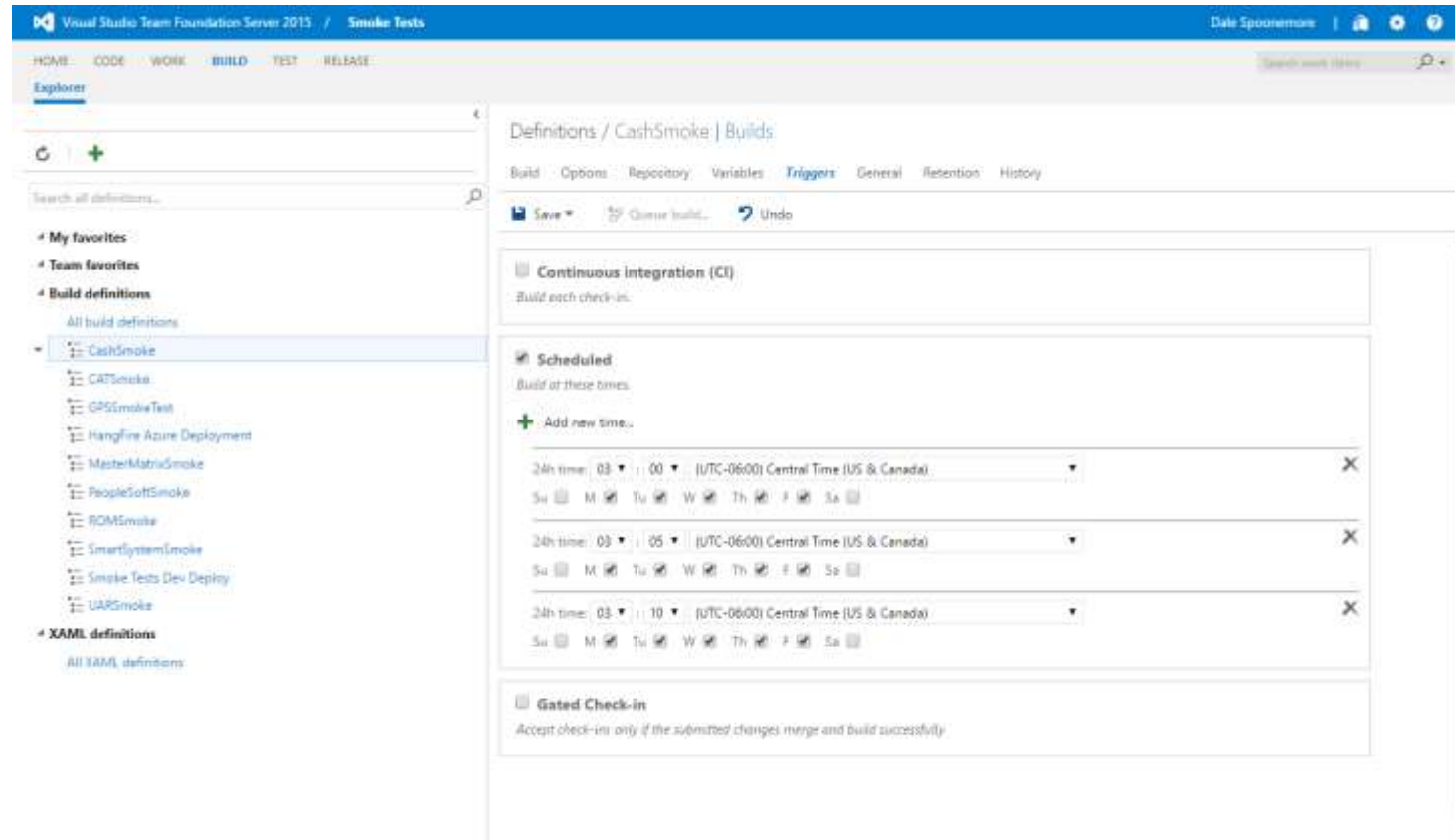
# TFS build steps – Publish Test Results



# TFS build steps – Publish Build Artifacts



# TFS build steps – Scheduling



# Automating Builds

---

Wil Isaacs  
@wvisaacs

Hangfire

<http://hangfire.io/>

**Demo**

<http://hangfiresmoketestpres.azurewebsites.net/>

# Next Steps

---

- UI for business users to subscribe to and manage email notifications
- More advanced CI UI tests

# Credits

---

Protractor

<http://www.protractortest.org/#/>

Protractor Cheat Sheet

<https://gist.github.com/javierarques/0c4c817d6c77b0877fda>

Introduction to Protractor PluralSight Course

<https://www.pluralsight.com/courses/protractor-introduction>

Elementor

<https://github.com/andresdominguez/elementor>



# Using Protractor to Build Automated UI “Smoke Tests”

---

DALE SPOONEMORE

[DALE@SEEDTOSPOON.NET](mailto:DALE@SEEDTOSPOON.NET)

@DALESPOONEMORE

WWW.SEEDTOSPOON.NET

A solid blue horizontal bar at the bottom of the slide.