



PROTRACTOR

© 2017, ACTIBYTI PROJECT SLU, Barcelona
Autor: Ricardo Ahumada

ÍNDICE DE CONTENIDOS

1. Introducción
2. Instalación
3. Configuración de una prueba
4. Ejecución de la prueba
5. Generación de informes de pruebas
6. Protractor y Angular 2

1

Introducción a Protractor.

Introducción a Protractor.

- Protractor (<http://www.protractortest.org/#/>), juega un papel importante en la prueba de aplicaciones Angular y funciona como un integrador de soluciones que combina varias tecnologías de gran alcance como Selenium y Jasmine, entre otros.
- Es un framework de tests end-to-end (simula usuario). Se **centra en probar la funcionalidad real** de una aplicación y no en pruebas unitarias.
- Se usa no solo para realizar pruebas a las aplicaciones Angular, sino también para la escritura de pruebas automatizadas de regresión para las aplicaciones web normales.
 - › JavaScript se usa en casi todas las aplicaciones web y a medida que las aplicaciones crecen, también aumenta en tamaño y complejidad del código javascript.
 - › Esto hace que sea una tarea complicada probar la aplicación web en diversos escenarios.
- Podemos consultar la documentación en:
 - › <http://www.protractortest.org/#/toc>

2

Instalación Protractor



Instalación de Protactor

- Antes de usar Protactor, debemos tener instalado:
 - Selenium:
 - Guía de instalación: <https://www.guru99.com/installing-selenium-webdriver.html>
 - Node.js

- Para instalar Protractor de manera global:

```
npm install -g protractor
```

- Comprueba la instalación y la versión con el comando:

```
Protractor --version
```

Instalación de Protractor



- Lo siguiente es actualizar el gestor de controladores web, que se utiliza para ejecutar las pruebas de las aplicaciones web angular en un navegador específico.
- Para ello se usa el comando:

```
webdriver-manager update
```

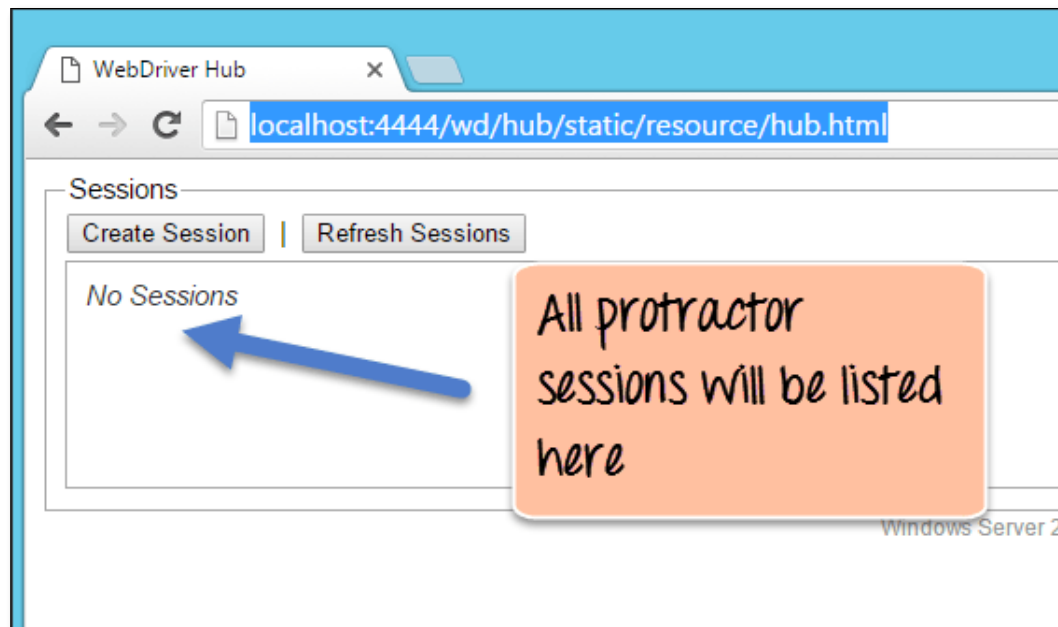
- Luego, podemos iniciar el gestor de controladores web con el siguiente comando:

```
webdriver-manager start
```

Instalación Protactor



- Lo siguiente será dirigirnos a la siguiente URL en el navegador:
 - `http://localhost:4444/wd/hub/static/resource/hub.html`



- Aquí se mostrarán las sesiones de ejecución de Protractor
- El controlador web se ejecutará en segundo plano.

3

Configuración de una prueba

Archivos de configuración

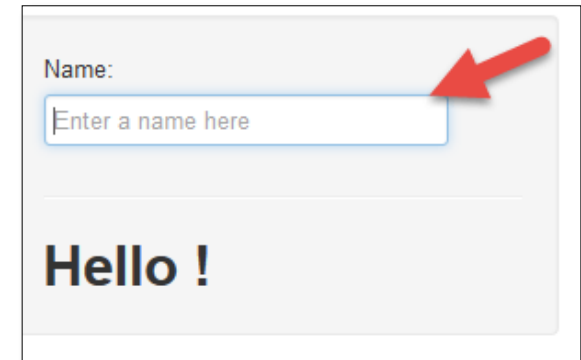
Protractor necesita dos archivos para ser ejecutado:

- **Fichero de configuración (config.js),**
 - Este archivo indica a Protractor la localización de las pruebas (specs.js) y a hablar con el servidor Selenium.
 - Chrome es el navegador por defecto para Protractor.

- **Archivo de especificaciones (specs.js)**
 - Contienen la lógica y los localizadores que interactúan con la aplicación.

Ejemplo de pruebas de aplicación Angular

- Supongamos que, debemos iniciar sesión en una app web (<https://angularjs.org/>) e introducir el texto "GURU99" en el campo "Enter a name here".
- En el texto de salida se espera que se muestre lo siguiente:
- Necesitaremos capturar el texto de la página web luego de introducir el nombre, ya que debemos verificar el texto esperado.



Archivos de configuración

- Tenemos que preparar el archivo de configuración **conf.js** y el archivo de especificaciones **spec.js**, de la siguiente forma:
- Archivo **spec.js**

```
describe('Enter GURU99 Name', function() {  
  it('should add a Name as GURU99', function() {  
    browser.get('https://angularjs.org');  
    element(by.model('yourName')).sendKeys('GURU99');  
  
    var guru= element(by.xpath('html/body/div[2]/div[1]/div[2]/div[2]/div/h3'));  
    expect(guru.getText()).toEqual('Hello GURU99!');  
  
  });  
});
```

specs.js

En el código anterior, podemos visualizar lo siguiente:

- **describe**('Enter GURU99 Name', function()
 - Sintaxis Jasmine que define los componentes de una aplicación, que puede ser una clase o función, etc.
 - "Enter GURU99" es sólo un string y no un código.
- **it** ('should add a Name as GURU99', function() ...
- **browser.get**('https://angularjs.org'), se abrirá una nueva instancia del navegador con la url mencionada.
- **element**(by.model('yourName')).**sendKeys**('GURU99')
 - Encontramos el elemento web utilizando el nombre del Modelo "yourName" (el valor de ng-model en la página)
- **var guru =**
element(by.xpath('html/body/div[2]/div[1]/div[2]/div[2]/div/h1')),
 - encontramos el elemento web utilizando XPath y almacenar su valor en una variable "guru".
- **expect**(guru.getText()).**toEqual**('Hello GURU99!'),
 - finalmente validamos el texto que hemos conseguido desde la página web con el texto esperado.

Localizadores

- El core de las pruebas de extremo a extremo es encontrar elementos DOM, interactuar con ellos y obtener información sobre el estado actual de su aplicación.
- Protractor facilita para ello **localizadores**
 - › <http://www.protractortest.org/#/locators>
- Los localizadores se acceden mediante la sintaxis
 - › **by.<tipo>()**

```
by.css('.myclass') // Find an element using a css selector.
```

```
by.id('myid') // Find an element with the given id.
```

```
// Find an element with a certain ng-model.
```

```
// Note that at the moment, this is only supported for AngularJS apps.
```

```
by.model('name')
```

```
// Find an element bound to the given variable.
```

```
// Note that at the moment, this is only supported for AngularJS apps.
```

```
by.binding('bindingname')
```

Conf.js

➤ Archivo **conf.js**:

```
exports.config = {  
  seleniumAddress: 'http://localhost:4444/wd/hub',  
  specs: ['spec.js']  
};
```

- **seleniumAddress**: El archivo de configuración le indica a Protractor la ubicación de Selenium Address para hablar con Selenium WebDriver.
- **specs:['spec.js']**: indica la ubicación del archivo de prueba spec.js.

4

Ejecución de la prueba

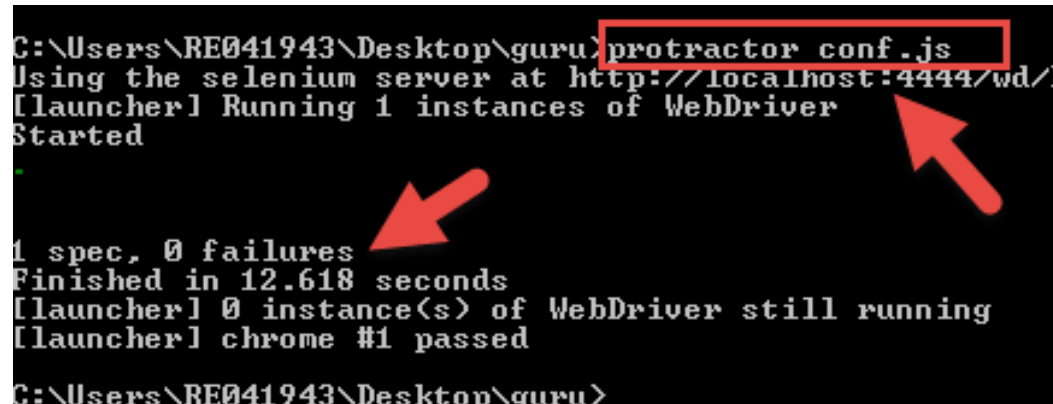
Ejecución del código.

- Abriremos un terminal de consola y nos posicionamos donde se encuentran los **confi.js** y **spec.js**.
- No aseguramos que el administrador del controlador de web de Selenium está en funcionamiento.
 - webdriver-manager start

```
C:\Users\RE041943>webdriver-manager start
seleniumProcess.pid: 3168
18:13:36.624 INFO - Launching a standalone Selenium Server
Setting system property webdriver.chrome.driver to C:\Users\RE041943\AppData\Roaming\npm\node_
18:13:37.531 INFO - Java: Oracle Corporation 25.51-b03
18:13:37.534 INFO - OS: Windows 7 6.1 amd64
18:13:37.659 INFO - v2.51.0, with Core v2.51.0. Built from revision 1af067d
18:13:37.903 INFO - Driver class not found: com.opera.core.systems.OperaDriver
18:13:37.904 INFO - Driver provider com.opera.core.systems.OperaDriver is not registered
18:13:37.929 INFO - Driver provider org.openqa.selenium.safari.SafariDriver registration is sk
registration capabilities Capabilities [{browserName=safari, version=, platform=MAC}] does not
18:13:39.631 INFO - RemoteWebDriver instances should connect to: http://127.0.0.1:4444/wd/hub
18:13:39.635 INFO - Selenium Server is up and running
```

Ejecución del código.

- Abre una nuevo terminal de consola y lanzar el siguiente comando (para correr el archivo de configuración):
 - **protractor conf.js**



```
C:\Users\RE041943\Desktop\guru>protractor conf.js
Using the selenium server at http://localhost:4444/wd/
[launcher] Running 1 instances of WebDriver
Started
.

1 spec, 0 failures
Finished in 12.618 seconds
[launcher] 0 instance(s) of WebDriver still running
[launcher] chrome #1 passed
C:\Users\RE041943\Desktop\guru>
```

- En este caso, protractor ejecutará el archivo de configuración con el spec definido.
- Podemos visualizar el servidor de Selenium funcionando en la url (definida en conf.js)
 - <http://localhost:4444/wd/hub>
- Además, se puede ver el resultado de cuantas pruebas correctas e incorrectas tenemos.

Generando un fallo

- Para simular un fallo modificaremos el resultado esperado en **spec.js** a algo incorrecto:

```
describe('Enter GURU99 Name', function() {  
  it('should add a Name as GURU99', function() {  
    browser.get('https://angularjs.org');  
    element(by.model('yourName')).sendKeys('GURU99');  
  
    var guru= element(by.xpath('html/body/div[2]/div[1]/div[2]/div[2]/div/h3'));  
    expect(guru.getText()).toEqual('Hello change GURU99!');  
  });  
});
```

- Guardamos el archivo **spec.js** y repetir los pasos anteriores.

Generando un fallo

- Obtendremos un resultado con error

```
C:\Users\RE041943\Desktop\guru>protractor conf.js
Using the selenium server at http://localhost:4444/wd/hub
[launcher] Running 1 instances of WebDriver
Started
F
Failures:
1) Enter GURU99 Name should add a Name as GURU99
   Message:
     Expected 'Hello GURU99!' to equal 'Hello change GURU99!'.
   Stack:
     Error: Failed expectation
       at Object.<anonymous> (C:\Users\RE041943\Desktop\guru\spec.js:10:24)
       at C:\Users\RE041943\AppData\Roaming\npm\node_modules\protractor\node_modules\jasmine
       at new wrappedCtr (C:\Users\RE041943\AppData\Roaming\npm\node_modules\protractor\node
       at controlFlowExecute (C:\Users\RE041943\AppData\Roaming\npm\node_modules\protractor\
       at goog.async.run.processWorkQueue (C:\Users\RE041943\AppData\Roaming\npm\node_module
       at process._tickCallback (node.js:369:9)
1 spec, 1 failure
Finished in 11.021 seconds
[launcher] 0 instance(s) of WebDriver still running
[launcher] chrome #1 failed 1 test(s)
[launcher] overall: 1 failed spec(s)
[launcher] Process exited with error code 1

C:\Users\RE041943\Desktop\guru>
```

5

Generación de informes de pruebas

Instalando Jasmine Reporter

- Protractor es compatible con Jasmine Reporter para generar informes de pruebas, en este caso necesitamos JunitXMLReporter para generar informes de ejecución de pruebas de forma automática en XML.
- Primero, debemos instalar Jasmine Reporter de manera local

```
npm -install --save-dev jasmine-reporters@^2.0.0
```

```
C:\Users\RE041943>cd Desktop
C:\Users\RE041943\Desktop>cd guru
C:\Users\RE041943\Desktop\guru>npm install --save-dev jasmine-reporters@^2.0.0
jasmine-reporters@2.0.0 node_modules\jasmine-reporters
└─ mkdirp@0.3.5
C:\Users\RE041943\Desktop\guru>
```

Configurando las specs para Jasmine Reporter

- Comprobamos la instalación en el directorio **node_modules**.

result	3/15/2016 6:24 PM	File folder	
node_modules	3/16/2016 9:39 PM	File folder	
E Ranjith kumar .txt	3/16/2016 9:51 PM	Text Document	1 KB

- Añadimos el siguiente código al archivo **conf.js** para generar el archivo **"JUnitXmlReporter"** e indicar una ruta donde almacenar el informe.

```
exports.config = {
  seleniumAddress: 'http://localhost:4444/wd/hub',
  capabilities: {
    'browserName': 'firefox'
  },
  specs: ['spec.js'],
  framework: 'jasmine2',
  onPrepare: function() {
    var jasmineReporters = require('C:/Users/RE041943/Desktop/guru/node_modules/jasmine-reporters');
    jasmine.getEnv().addReporter(new jasmineReporters.JUnitXmlReporter(null, true, true)
    );
  }
};
```

Generación de informes de pruebas.

- Abriremos el símbolo del sistema y ejecutamos el comando.

protractor conf.js

```
C:\Users\RE041943\Desktop\guru>protractor conf.js
Using the selenium server at http://localhost:4444/wd/hub
[launcher] Running 1 instances of WebDriver
Started
F
Failures:
1) Enter GURU99 Name should add a Name as GURU99
  Message:
    Expected 'Hello GURU99!' to equal 'Hello change GURU99!'.
  Stack:
    Error: Failed expectation
      at Object.<anonymous> (C:\Users\RE041943\Desktop\guru\spec.js:10:24)
      at C:\Users\RE041943\AppData\Roaming\npm\node_modules\protractor\node_
      at new wrappedCtr (C:\Users\RE041943\AppData\Roaming\npm\node_modules\
      at controlFlowExecute (C:\Users\RE041943\AppData\Roaming\npm\node_mod
      at goog.async.run.processWorkQueue (C:\Users\RE041943\AppData\Roaming\
      at process._tickCallback (node.js:369:9)

1 spec, 1 failure
Finished in 17.696 seconds
```

- Cuando ejecutamos el código anterior, el archivo **JUnitResults.xml** se generará en la ruta definida.

JUnitResults.xml

- Abrimos el XML y verificamos el resultado.
- El mensaje de error se muestra en el archivo de resultados, ya que nuestro caso de prueba es un error: el resultado esperado de "spec.js" no coincide con el resultado real de la página Web

```
<?xml version="1.0" encoding="UTF-8"?>
<testsuites>
  - <testsuite failures="1" skipped="0" tests="1" errors="0" time="17.631" hostname="localhost" timestamp="2016-03-16T22:02:41" name="Enter GURU99 Name">
    - <testcase time="17.628" name="should add a Name as GURU99" classname="Enter GURU99 Name">
      - <failure message="Expected &apos;Hello GURU99!&apos; to equal &apos;Hello change GURU99!&apos;." type="toEqual">
        <![CDATA[Error: Failed expectation of Object.<anonymous> (C:\Users\RE041943\Desktop\guru(spec.js:10:24) at C:\Users\RE041943\AppData\Roaming\npm\node_modules\protractor\node_modules\jasminewd2\index.js:23 at [object Object].promise.Promise.goog.defineClass.constructor (C:\Users\RE041943\AppData\Roaming\npm\node_modules\protractor\node_modules\selenium-webdriver\lib\goog\../webdriver/promise.js:1056:7) at new wrappedCtr (C:\Users\RE041943\AppData\Roaming\npm\node_modules\protractor\node_modules\jasminewd2\index.js:82:18) at [object Object].promise.ControlFlow.goog.defineClass.goog.defineClass.abort_.error.executeNext_.execute_ (C:\Users\RE041943\AppData\Roaming\npm\node_modules\protractor\node_modules\selenium-webdriver\lib\goog\../webdriver/promise.js:2776:14) at [object Object].promise.ControlFlow.goog.defineClass.goog.defineClass.abort_.error.executeNext_ (C:\Users\RE041943\AppData\Roaming\npm\node_modules\protractor\node_modules\selenium-webdriver\lib\goog\../webdriver/promise.js:758:21) at goog.async.run.processWorkQueue (C:\Users\RE041943\AppData\Roaming\npm\node_modules\protractor\node_modules\selenium-webdriver\lib\goog\async\run.js:124:15) at process (C:\Users\RE041943\AppData\Roaming\npm\node_modules\protractor\node_modules\selenium-webdriver\lib\goog\async\run.js:369:9)]]
      </failure>
    </testcase>
  </testsuite>
</testsuites>
```

Actual Result from Web page

Expected Result from spec.js file



Pongámoslo en práctica

- Sigue el tutorial de la página oficial de Protractor
<http://www.protractortest.org/#/tutorial>
- En ella se prueba la siguiente aplicación
 - › <http://juliemr.github.io/protractor-demo/>
- Además añade la generación de reportes

5

Protractor y Angular 2

Instalando Jasmine Reporter

- Puede probar aplicaciones Angular 2 con Protractor (a partir de Protractor 2.5.0).
- Para Protractor 5.0.0+, no se tiene que hacer nada específico, Protractor detectará automáticamente la versión Angular utilizada en la aplicación bajo prueba.
- Para Protractor $> 2.5.0$ y $\leq 4.0.14$, sólo tendría que agregar a su configuración.
 - `useAllAngular2AppRoots: true`

conf.js para Protractor >= 2.5.0 y <= 4.0.14

```
var env = require('./environment.js');

exports.config = {
  seleniumAddress: env.seleniumAddress,

  framework: 'jasmine',

  specs: [
    'ng2/async_spec.js'
  ],

  capabilities: env.capabilities,

  baseUrl: 'http://localhost:8000',

  // Special option for Angular2, to test against all Angular2 applications
  // on the page. This means that Protractor will wait for every app to be
  // stable before each action, and search within all apps when finding
  // elements.
  useAllAngular2AppRoots: true

  // Alternatively, you could specify one root element application, to test
  // against only that one:
  // rootElement: 'async-app'
};
```

CLI angular y transportador

- El CLI de Angular nos da toda la funcionalidad que necesitamos para la estructuración, construcción y pruebas de nuestras aplicaciones angulares.
- Incluye pruebas unitarias y la configuración necesaria para nuestras pruebas E2E.
- Las pruebas E2E usan Protractor y el código de prueba en sí está escrito usando Jasmine.
- En nuestro proyecto Angular CLI tenemos una carpeta llamada e2e. Aquí es donde se mantienen nuestras pruebas e2e.
- La configuración de protractor viene en el archivo
 - **protractor.conf.js**
 - Este se referencia desde el archivo de configuración del cli:
 - .angular-cli.json

protractor.conf.js

```
// Protractor configuration file, see link for more information
// https://github.com/angular/protractor/blob/master/lib/config.ts

const { SpecReporter } = require('jasmine-spec-reporter');

exports.config = {
  allScriptsTimeout: 11000,
  specs: [
    './e2e/**/*.e2e-spec.ts'
  ],
  capabilities: {
    'browserName': 'chrome'
  },
  directConnect: true,
  baseUrl: 'http://localhost:4200/',
  framework: 'jasmine',
  jasmineNodeOpts: {
    showColors: true,
    defaultTimeoutInterval: 30000,
    print: function() {}
  },
  onPrepare() {
    require('ts-node').register({
      project: 'e2e/tsconfig.e2e.json'
    });
    jasmine.getEnv().addReporter(new SpecReporter({ spec: { displayStacktrace: true } }));
  }
};
```

- Se indica la localización de las especificaciones
 - ./e2e/**/*.e2e-spec.ts
- Asimismo se indica la localización del archivo de configuración para Typescript
 - e2e/tsconfig.e2e.json

e2e/tsconfig.e2e.json

```
{  
  "extends": "../tsconfig.json",  
  "compilerOptions": {  
    "outDir": "../out-tsc/e2e",  
    "baseUrl": ".",  
    "module": "commonjs",  
    "target": "es5",  
    "types": [  
      "jasmine",  
      "jasminewd2",  
      "node"  
    ]  
  }  
}
```


Clase de objeto de página

```
// app.po.ts
import { browser, by, element } from 'protractor';

export class MyAppNg2Page {
  navigateTo() {
    return browser.get('/');
  }

  getParagraphText() {
    return element(by.css('app-root h1')).getText();
  }
}
```

- Nuestra clase de objeto de página es una clase que describe una vista de página de alto nivel.
- Con `navigateTo` navegamos a la página de inicio.
- Con `getParagraphText` buscamos un texto encerrado en `H1`.
- Utilizamos la función `Protractor by.css ()` para seleccionar elementos en la página por `css`.

Archivo de especificación de prueba

```
// app.e2e-spec.ts
import { MyAppNg2Page } from './app.po';

describe('my-app-ng2 App', () => {
  let page: MyAppNg2Page;

  beforeEach(() => {
    page = new MyAppNg2Page();
  });

  it('should display welcome message', () => {
    page.navigateTo();
    expect(page.getParagraphText()).toEqual('Welcome to app!');
  });
});
```

- En el archivo de configuración usamos Jasmine para llamar a los métodos del archivo de configuración y generar las aserciones de los métodos.



Pongámoslo en práctica

- Clona es el siguiente repositorio (por **Cory Rylan**)
 - <https://github.com/coryrylan/ng-pokedex.git>
- Revisa la configuración de pruebas e2e
- Lanza las pruebas usando
 - npm run e2e
- O simplemente con
 - protractor



 **netmind**

WeKnowIT

Barcelona

C. Almogàvers, 123
08018 Barcelona
Tel. 93 304.17.20
Fax. 93 304.17.22

Madrid

Plaza Carlos Trías Bertrán, 7
28020 Madrid
Tel. 91 442.77.03
Fax. 91 442.77.07

www.netmind.es



MINISTERIO
DE ENERGÍA, TURISMO
Y AGENDA DIGITAL

red.es



UNIÓN EUROPEA

Fondo Social Europeo
"El FSE invierte en tu futuro"