End-to-End Testing

Guide

v1.0

23/07/2020

Introducción

El Testing End-To-End es una prueba de extremo a extremo y su objetivo es verificar que la aplicación funciona como un todo incluyendo conexión a datos.

En otras palabras las Pruebas Unitarias se centran en testear unidades de código de forma aislada (Requerimiento funcional) , mientras que el Testing End-To-End se centra en la Historia de Usuario (Usos del usuario en metodologías Ágiles) o Escenarios de Uso (Usos del usuario en metodologías Tradicionales) proporcionando pruebas de alto nivel en el flujo general de datos.

Lo que se puede detectar con el Testing E2E son problemas con la arquitectura general de la aplicación.

La mayoría de las pruebas E2E operan automatizando las interacciones comunes del usuario con la aplicación examinando el DOM de para determinar resultados de esas interacciones.

Términos, Abreviaturas y Acrónimos

- E2E: End-To-End
- Testing: Testeo

Test Structure

Al generar una aplicación Ionic Angular se genera automáticamente una carpeta predeterminada e2e que actúa como una aplicación.

La aplicación e2e utiliza Protractor para controlar la navegación e interacción con modelos DOM y Jasmine revisado en el recurso **3.1.1 Testing Ionic Angular** para estructurar y ejecutar las pruebas.

La estructura cuenta con 4 archivos:

1. **protractor.conf.js** con la configuración de Protractor
2. **tsconfig.e2e.json** con la configuración TypeScript para la aplicación de las pruebas 3. **src/app.po.ts** un Page que contiene métodos que navegan por la aplicación y consulta y manipulan elementos de la página
4. **src/app.e2e-spec.ts** el script a testear.

Protractor

Es una herramienta que permitirá emular el comportamiento de un usuario ejecutando pruebas automáticamente mediante un navegador como

1. Ir al índice de la página

2. Luego hacer click en el menú
3. Hacer click en Productos

Antes de Comenzar

Antes de comenzar se recomienda instalar Chromium en su última versión disponible para evitar el error **session not created: This version of ChromeDriver only supports Chrome version XX**

Comencemos

Para efectos de iniciar en las Pruebas E2E es que crearemos un proyecto nuevo de tabs, pero ustedes lo podrán aplicar a sus proyectos ya existentes.

Paso 0: Crear un proyecto prueba

```
ionic start e2e-example-tabs tabs --type=angular
```

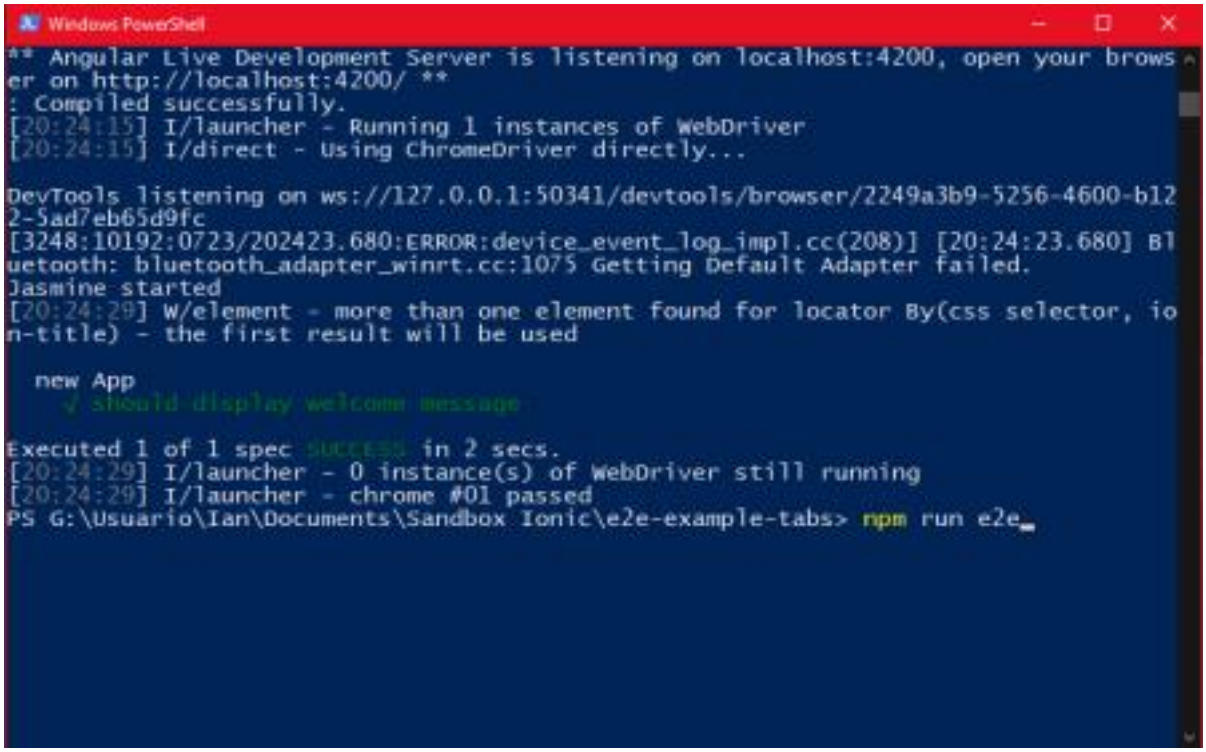
Se crea un proyecto de prueba para iniciar en las pruebas E2E llamado e2e-example-tabs con una plantilla tabs de tipo Angular. Cuando finalice recuerda trabajar en el directorio del proyecto.

```
cd .\e2e-example-tabs\
```


Paso 1: Ejecutar prueba E2E

Luego se ejecuta la aplicación creada de prueba e2e generada al crear el proyecto. `npm run e2e`

Al ejecutarlo deberíamos tener una salida como esta:



Como se puede observar se ejecutó 1 de 1 prueba de forma exitosa en 2 seg. Esto por atrás ejecuto el archivo **src/app.e2e-spec.ts** que contiene el siguiente código:

```
import { AppPage } from './app.po';

describe('new App', () => {
 let page: AppPage;

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

 it('should display welcome message', () => {
 page.navigateTo();
 expect(page.getPageTitle()).toContain('Tab 1');
 });
});
```


Donde se Importa un Page desde la misma carpeta src, esté page llamada AppPage tiene una función en el cual navegamos a la raíz de la aplicación y una función que obtiene el texto de un elemento mediante el selector CSS.

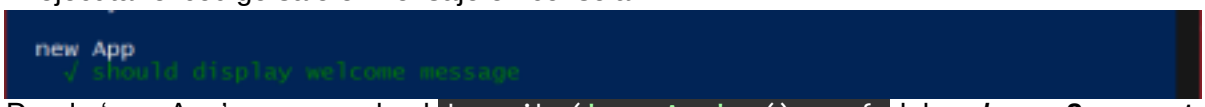
```
import { browser, by, element } from 'protractor';
```

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

 getPageTitle() {
 return element(by.css('ion-title')).getText();
 }
}
```

Luego de importar el Page, se desarrolla una unidad de código mediante el describe entregando un nombre descriptivo, donde se crea una variable page de tipo **AppPage**, posteriormente en la misma unidad de código se ejecuta un código de configuración donde se inicializa de forma manual el Page importado para finalmente ejecutar la prueba unitaria mediante la función **it** en el cual se usa la función **navegateTo** del Page importado para ir a la ruta principal y luego validar mediante el **expect** que el título del page contenga cierta cadena, en este caso 'Tab 1' mediante el **toContain.**

Al ejecutar el código sale el mensaje en consola:



Donde 'new App' corresponde al `describe('new App', () => {` del **src/app.e2e-spec.ts** y 'should display welcome message' `it('should display welcome message', () => {`

Con este paso, usted aprendió cómo se estructura, funciona y ejecuta una prueba E2E. Ahora crearemos nuestro propio archivo de test.

Paso 2: Generar archivo de Testing

Primero debemos generar un archivo de test de forma manual en la carpeta e2e/src y lo llamaremos **tabs.e2e-spec.ts** donde importaremos browser, element y by desde protractor

```
import { browser, element, by } from "protractor";
```

Luego crearemos una unidad de código, donde lo describiremos cómo 'tabs' y ejecutaremos una configuración del test donde navegaremos a la ruta raíz.


```
describe("tabs", () => {
 beforeEach(() => {
 browser.get("/");
 });

});
```

Luego de la configuración, ejecutaremos dos pruebas, la primera verificaremos que el texto del

elemento **label** del **tab seleccionado** contenga el texto ‘Tab 1’

```
it("La pestaña Tab One se muestra por defecto", () => {
 expect(element(by.css(".tab-selected ion
label")).getText()).toContain("Tab 1");
});
```

La segunda prueba se probará la navegación al tab2 de nuestra aplicación donde se simulará un click sobre el tab2 de forma asíncrona (se espera a que se ejecute la navegación), se espera medio minuto y se verificará que el texto del elemento **label** del **tab seleccionado** contenga el texto ‘Tab 2’

```
it("El usuario puede navegar a la pestaña Tab Two", async () =>
{  await element(by.css("[tab=tab2]")).click();
 browser.driver.sleep(500);
 expect(element(by.css(".tab-selected ion
label")).getText()).toContain("Tab 2");
});
```


La unidad de código quedaría de la siguiente forma

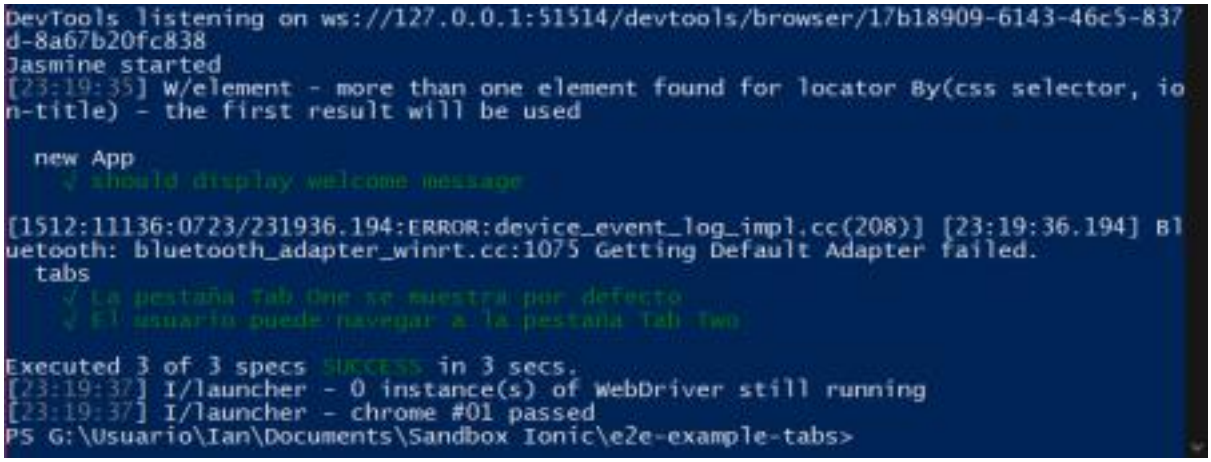
```
import { browser, element, by } from "protractor";

describe("tabs", () => {
 beforeEach(() => {
 browser.get("/");
 });

 it("La pestaña Tab One se muestra por defecto", () => {
 expect(element(by.css(".tab-selected ion
label")).getText()).toContain("Tab 1");
 });

 it("El usuario puede navegar a la pestaña Tab Two", async () => {
await element(by.css("[tab=tab2]")).click();
 browser.driver.sleep(500);
 expect(element(by.css(".tab-selected ion
label")).getText()).toContain("Tab 2");
 });
});
```

Al ejecutarlo obtendremos lo siguiente



Referencias

E2E (End-to-End) Testing in Ionic: An Introduction. (2019, 18 septiembre). joshmorony - Learn Ionic & Build Mobile Apps with Web Tech. https://www.joshmorony.com/e2e-end-to-end-testing in-ionic-2-an-introduction/

BRANDYSCARNEY, & PERRYGOVIER. (2020c, abril 2). Testing - Ionic Documentation. Ionic Docs. https://ionicframework.com/docs/angular/testing