Proctartor : Qu'est-ce que c'est ?

Protractor est un framework de tests e2e (End-to-end) développé par l'équipe d'AngularJS. Il permet à la fois de tester les applications AngularJS et Angular mais aussi des sites web non-Angular. Protractor utilise Jasmine comme framework de tests et d'assertion et le "webdriver" Selenium pour communiquer avec les "browsers".

Protractor : https://www.protractortest.org/#/

Jasmine : https://jasmine.github.io/

Angular : https://angular.io/

Angular JS : https://angularjs.org/

Selenium : https://www.selenium.dev/

Structure d'un test E2E

Pour créer un test e2e il faut plusieurs types de fichiers :

- Un fichier e2e.ts qui va nous permettre de fournir les instructions qu'on souhaite faire à notre test.

```
import { LoginPO } from "../../pages/login.po";
import { SashaHomePO } from "../../pages/sasha/sasha-home.po";

describe("Test to see if the bookings appear in the last files created section", suite);

function suite(): void {

  const loginPage: LoginPO = new LoginPO();
  const sashaHome: SashaHomePO = new SashaHomePO();

  beforeAll(async () => {
    await loginPage.navigateAndLogin();
  });

  it('Opens Sasha and Test to see if the bookings appear in the last files created section', async ()=> {
```

```
    await sashaHome.navigateToSasha();
    //loading the booking list
    // retrieve the booking code
    const firstBookingCode: string = await sashaHome.getFirstBookingCode();
    expect(firstBookingCode)
      .toBeTruthy("Ce test a échoué car impossible de trouver un booking");
    await sashaHome.clickFirstBooking();
  });
}
```

- Un fichier PO (PageObject) qui permet la réutilisation de code dans plusieurs tests.

```
import { browser, by, element, ElementFinder, ExpectedConditions as EC, protractor } from
"protractor";

export class SashaHomePO {
  public async navigateTo(): Promise<unknown> {
    return browser.get(`${browser.baseUrl}/sasha/home`);
  }

  public async getTitleText(): Promise<string> {
    return element(by.css('app-root .content span')).getText();
  }

  public async waitForDisplay(): Promise<void> {
    const el_7 = element(by.css("booking-saved-searched-summary"));
    await browser.wait(EC.presenceOf(el_7));
  }
}
```

Fichier de configuration

Protractor utilise un fichier de configuration afin de fonctionner.

Permet de paramétrer les timeouts, le serveur que l'on souhaite tester, les rapports générés, etc.

Pour plus d'informations : https://github.com/angular/protractor/blob/master/lib/config.ts

Comment Le programme retrouve les éléments auxquels on souhaite interagir ?

Pour qu'un test e2e puisse retrouver un élément précis d'une page web, on peut ajouter des classes CSS si nécessaires dans le code des pages HTML. (Si pas besoin on n'en ajoute pas).

Ces classes doivent absolument commencer par e2e, car plus facile pour s'y retrouver.

Exemple :

la fonction getFirstFlightListByRatingYucangoo va interroger le fichier PO,

Ensuite, la fonction indique ensuite que cette interaction doit utiliser la classe ".e2e-best-rating-yucangoo".

Le programme va ensuite chercher dans la page en question l'élément correspondant.

Ici le but était de pouvoir récupérer la note Yucangoo

On peut également utiliser l'outil de développement de Chrome pour mettre au point les sélecteurs CSS.

Exécuter un test E2E

Tout d'abord, pour lancer le serveur il faut exécuter la commande :

npm run serve-sasha

Ensuite: (cette commande exécute le programme).

npx protractor e2e/protractor.conf.js --verbose --specs
e2e/src/specs/nom_du_dossier/nom_du_fichier.ts

ATTENTION, il faut remplacer les 2 derniers paramètres de la commande, en modifiant avec le bon emplacement de votre test.

Les évènements Protractor

Protractor, propose plusieurs évènements afin qu'on puisse réaliser notre test E2E. En effet, il y a l'évènement :

- Await : permet de dire au programme d'attendre que la première action soit terminé avant que la suite s'exécute.

```
await sashaHome.navigateToSasha();

await sashaHome.sendForChooseBooking("ZCRSTT");

await sashaHome.clickForSearchBooking();

await sashaHome.waitForSearchBooking();
```

- Wait : permet d'attendre que la page se charge complètement avant de continuer.

```
browser.wait(EC.presenceOf(quoteResults));
```

- sendKeys : permet de saisir du texte dans un input.

```
.sendKeys(input);
```

- Element : permet de sélectionner une classe CSS afin d'interagir avec un élément de la page web.

```
element(by.css(".e2e-menu-item-duration")).click();
```

- getText : permet de récupérer une valeur présente dans la page web.

```
.getText();
```

- click : permet de cliquer sur un élément de la page web.

```
.click();
```

Pour plus d'informations : https://www.protractortest.org/#/api?view=ProtractorBrowser

On nomme également les variables en fonction de leur action. Quand l'action de la variable et de cliquer alors le début du nom de la variable sera click.

Exemple:

Voir le résultat après exécution de son test E2E

Protractor met à disposition une interface web nous permettant de voir comment notre test s'est exécuté.

La flèche rouge montre quand un test a rencontré une erreur et nous explique le problème.

La flèche noire montre quand un test a échoué.

La flèche bleue montre quand un test a réussi.

Le rond en vert montre le temps d'exécution du test.

Le rond en violet montre la possibilité de regarder le Screen au moment de la fin du test.

Exemple :

Les résultats de chaque test est stocké localement dans un dossier qui s'appelle "reports".