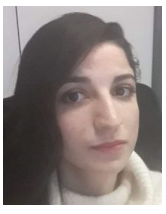
All4Test

Test E2E avec Protractor, cucumber en utilisant TypeScript!

Rédacteurs

Zied HANNACHI
Consultant - ALL4TEST



Insaf KSSOURI
SDET - ALL4TEST

Table des matières

Introduction

La plupart du développement Web utilise la bibliothèque angularJS pour sa légèreté et sa facilité d'utilisation, Protractor est devenu populaire pour faire des tests de bout en bout (Test E2E) pour les applications angularJS.

Dans cet article nous allons parler de notre solution d'automatisation de test dans lequel nous avons utilisé le framework Protractor, Cucumber et le langage de scripting TypeScript.

Framework

Protractor

Protractor est un Framework de test E2E développé pour Angular et AngularJS. Il utilise les fonctionnalités de Selenium WebDriver en interne et fonctionne comme un intégrateur de solution combinant des technologies puissantes comme Selenium, Jasmine, Web Driver, etc.

Pourquoi le test E2E

L'avantage des tests E2E est de pouvoir tester son application comme elle apparaîtra aux yeux de l'utilisateur.
Protractor permet d'exécuter ces tests E2E directement dans le ou les navigateurs physiques de votre choix. Il interagit avec le navigateur naturellement comme tout utilisateur le ferait.

Pourquoi utiliser Protractor

- Installation et configuration faciles.
- Protractor inclut tous les avantages de Selenium WebDriver.
- Synchronisation automatique - pas besoin d'attendre automatiquement les actions de l'utilisateur, nous n'avons donc pas besoin d'ajouter explicitement des attentes à notre test.
- Prend en charge les tests parallèles via plusieurs navigateurs
- Protractor utilise le framework Jasmine ou Mocha ou Cucumber pour la rédaction des tests.
- Test des applications à base angulaire et non angulaire.

Cucumber

Behavior-driven development (BDD)

Cucumber est un outil de test qui prend en charge le cadre de développement axé sur le comportement (BDD). Il définit le comportement de l'application à l'aide d'un texte anglais simple, défini par une langue appelée Gherkin.

Gherkin

Gherkin utilise un ensemble de mots clés spéciaux pour donner une structure et un sens aux spécifications exécutables. Gherkin permet d'écrire des scénarios de test compréhensibles par des individus non techniques. Cette approche sert deux objectifs : documenter les fonctionnalités à développer d'une part, et permettre l'automatisation des tests d'autre part.

La plupart des lignes d'un document Gherkin commencent par l'un des mots clés:

- Le "Given" mot-clé précède le texte définissant le contexte; l'état connu du système (ou condition préalable).
- Le "When" mot-clé précède le texte définissant une action.
- Le "Then" mot-clé précède le texte définissant le résultat de l'action sur le contexte (ou résultat attendu).

TypeScript

TypeScript est un langage de programmation open source développé et maintenu par Microsoft . Il s'agit d'un superset syntaxique strict de JavaScript et ajoute une saisie statique facultative au langage.

Protractor a commencé officiellement à prendre en charge TypeScript avec sa version 4.0.0 en ajoutant de nombreuses fonctionnalités intéressantes comme la saisie automatique des instructions, les méthodes API et les suggestions de documentation, etc.

Fonctionnement de TypeScript

Le fichier “hello.ts” est un fichier ts, lors de la compilation du fichier ts avec le compilateur à l'aide de la commande tsc, le compilateur génère automatiquement un fichier javascript avec le même nom que hello.js.

Pourquoi TypeScript?

Problème avec Javascript

Nous n'avons pas besoin de suivre des typages stricts comme définir des variables et suivre des syntaxes strictes lors de l'écriture d'un script en utilisant du Javascript simple. Cela rendra le code difficilement lisible.

Utilisations de TypeScript

Typescript permettre aux développeurs d'utiliser des outils et des pratiques de développement hautement productifs comme la vérification statique et la refactorisation de code lors du développement d'applications JavaScript.

Typescript aura également la capacité de fournir intellisense à travers des éditeurs tels que 'Atom', 'Sublime', 'VisualStudio', 'Vim' et 'Webstorm' etc.

Pourquoi Protractor avec TypeScript ?

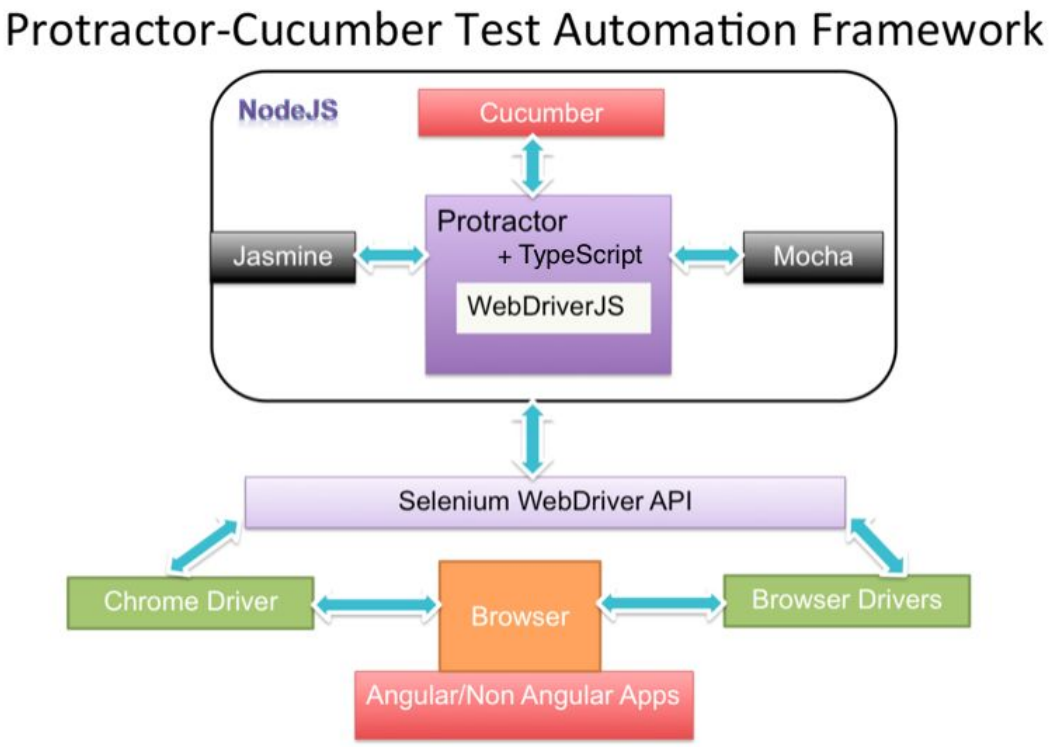
Le plus gros problème avec les frameworks Protractor est de les écrire en JavaScript. C'est un langage très puissant, mais pas assez efficace pour écrire du code rapidement et sans erreurs. La meilleure solution pour cela serait TypeScript. Il s'agit d'un “superset” de JavaScript.

La syntaxe de Protractor avec TypeScript est proche de la syntaxe JavaScript.
Protractor avec TypeScript fournit des propriétés liées à l'espace de noms global via une simple importation. À partir de cette propriété importée, nous pouvons obtenir la saisie semi-automatique, la signature de méthode et le JSDoc.

Architecture et Fonctionnement

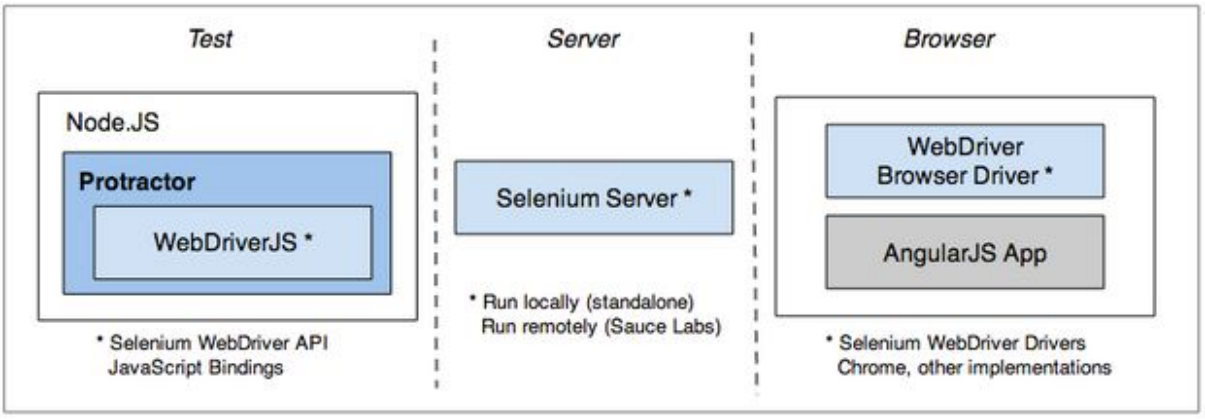
Architecture Protractor-Cucumber

L'image ci-dessous montre le squelette du Protractor-Cucumber qui intègre des outils et technologies puissants tels que NodeJS, Selenium, WebDriverJS, Jasmine, Cucumber et Mocha qui nous offrent un cadre de test évolutif de bout en bout pour nos applications Web.



Comment cela fonctionne ?

Protractor fonctionne en collaboration avec Selenium pour fournir une infrastructure de test automatisée qui peut simuler l'interaction d'un utilisateur avec une application angulaire exécutée dans un navigateur ou un appareil mobile.



Lorsque vous travaillez avec Protractor, il est important de garder à l'esprit les points suivants:

- Protractor est un wrapper autour de WebDriverJS, les liaisons JavaScript de l'API Selenium WebDriver.
- Les commandes WebDriver sont asynchrones. Ils sont planifiés sur un flux de contrôle et des promesses de retour, et non des valeurs primitives.
- Vos scripts de test envoient des commandes au serveur Selenium, qui à son tour communique avec le pilote du navigateur. Lire la suite pour plus de détails.

Configuration du Protractor

Protractor est un programme node.js, pour commencer à l'utiliser vous devez installer Node.js en premier lieu qui est une plateforme logicielle libre en JavaScript basé sur le moteur JavaScript V8 de Chrome.
Après on installe Angular CLI qui est un outil permettant de créer, construire, générer et tester des applications et librairies Angular et est basé sur typescript.
L'installation d'angular CLI est simple à travers du npm.

npm install -g @angular/cli

Puis on créé un projet angular

ng new Protractor_TypeScript_Cucumber_Test

Maintenant on peut installer l'outil protractor dans notre projet angular à traver la commande npm

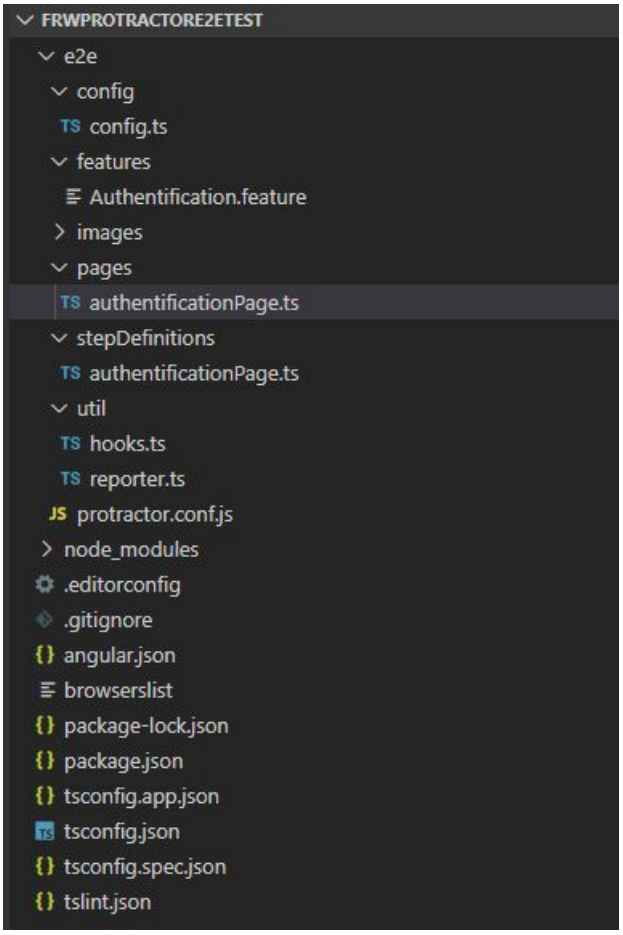
npm install -g protractor

Le fichier Config.js assure la configuration du protractor lors de son lancement. Le fichier doit avoir les paramètres obligatoires suivants:

- Adresse du serveur Selenium.
- Emplacement du fichier de spécifications (specs).
- Capacités du navigateur.
- Options du Node.
- Les options de Cucumber (cucumberOpts).

Solution Protractor-Cucumber avec TypeScript

L'image ci-dessous montre l'architecture de notre solution d'automatisation de test



Écriture de fonctionnalités

Nous écrivons des fichiers de fonctionnalités au format Given, When & Then décrivant le contexte du test.

```
Feature: Authentification
    Scenario: Authentification
        Given I go to the application
```

```
        When I enter "Admin" in the username field
        And I enter "admin123" in the password field
        And I press the login button
        Then The dashboard page is displayed
```

Rédaction de StepDefinitions

Nous écrivons le code typescript équivalent pour chacune des définitions d'étape qui nous aident à tester les scénarios de fonctionnalité.

```
import { authentificationPageObjects } from
"../pages/authentificationPage";
const { Given, When, Then } = require("cucumber");
import {expect} from 'chai';

const authentification: authentificationPageObjects = new
authentificationPageObjects();

Given(/^I go to the application$/, async () => {
    await
authentification.OpenBrowser("https://opensource-demo.orangehrmlive.com
/index.php/auth/login");
});

When(/^I enter "(.*?)" in the username field$/, async (username:
string) => {
    await authentification.UsernameSendKeys(username);
});

When(/^I enter "(.*?)" in the password field$/, async (password:
string) => {
    await authentification.PasswordSendKeys(password);
});

When(/^I press the login button$/, async () => {
    await authentification.LoginButtonClick();
});

Then(/^The dashboard page is displayed$/, async () => {
    const dashbordText = await authentification.dashboardLinkText();
```

```
    expect(dashbordText).to.equal("Dashboard");
});
```

Écriture des objets de page

Les objets de page nous aident à écrire des tests plus propres en encapsulant des informations sur les éléments de notre page d'application.

```
import {browser, element, by, protractor, $$, $} from 'protractor';
export class authentificationPageObjects {

    username = element(by.id("txtUsername"))
    password = element(by.id("txtPassword"))
    loginButton = element(by.id("btnLogin"))
    dashboardLink = element(by.id("menu_dashboard_index"))

    OpenBrowser(url: String){
        browser.get("url");
    }
    UsernameSendKeys(username: String){
         this.username.sendKeys("username");
    }
    PasswordSendKeys(password: String){
        this.username.sendKeys("password");
    }
    LoginButtonClick(){
        this.loginButton.click();
    }
    dashboardLinkText(){
        return this.dashboardLink.getText();
    }
}
```

Balises CucumberOpts

les options de ligne de commande Cucumber ont un certain nombre de fonctionnalités et de balises de support qui nous aident à exécuter des scénarios et des fonctionnalités spécifiques.

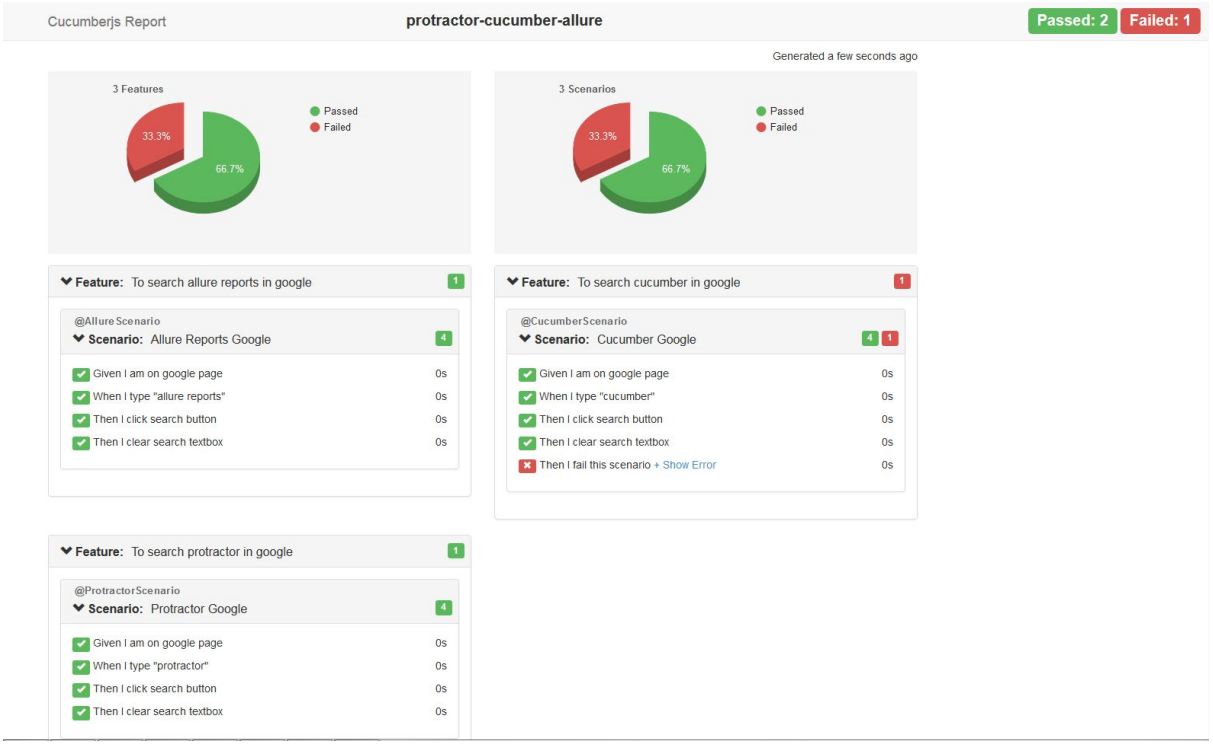
```
cucumberOpts: {
        compiler: "ts:ts-node/register",
        format: "json:./reports/json/cucumber_report.json",
        require: ["../../e2e/stepDefinitions/*.ts",
                  "../../e2e/util/*.ts"],
        strict: true,
        tags: "@CucumberScenario",
    },
```

Résultats de test

La solution a été intégrée avec cucumber-html-reporter , qui est généré dans le répertoire Rapport de test lorsque vous exécutez la commande de lancement de test protractor. Ils peuvent être personnalisés en fonction des besoins spécifiques de l'utilisateur.
L'installation du cucumber-html-reporter est à travers la commande npm.

npm install cucumber-html-reporter --save-dev

L'image ci-dessous montre un exemple du rapport cucumber-html-reporter



Bonnes pratiques

Voici une liste des bonnes pratiques pour construire un framework d'automatisation efficace avec Protractor.

- Utilisez TypeScript au lieu de JavaScript
- L'objet de page est définitivement un modèle de conception incontournable
- Un fichier TypeScript, une classe d'objets page
- Ayez toujours de bonnes conventions de codage pour les objets de page
- Supprimer l'importation circulaire TypeScript / JavaScript
- Gérer le comportement asynchrone de JavaScript en utilisant asynchrone et attendre
- Envelopper l'élément Protractor et les objets originaux du navigateur
- La méthode de l'objet page doit attendre que la page soit entièrement chargée avant de quitter
- ProtractorBy est le plus rapide mais XPath est le localisateur le plus stable
- Le cas de test ne doit appeler que les méthodes de l'objet de page
- Ne placez pas d'assertions dans les classes d'objets de page
- Avoir une structure de projet simple mais efficace
- Poussez uniquement les éléments nécessaires vers le référentiel de code

Un bon cas de test avec Protractor

Écrire un cas de test n'est pas un gros problème, mais le plus important c'est d'écrire un bon cas de test. Vous devez savoir quelle méthode de Protractor utiliser. Voici des meilleures pratiques à suivre lors de la rédaction d'un scénario de test.

browser.sleep () VS browser.wait ()

Il est préférable d'utiliser browser.wait () au lieu de browser.sleep ()

toBe () vs toEqual ()

```
expect(A).toEqual(B);   // returns true
expect(A).toBe(B);        // returns false
```

toBe (true) vs toBeTruthy () vs toBeTrue ()

Utiliser toBe (true) est la meilleure option

isPresent () vs isDisplayed ()

isPresent () vérifie si l'élément existe dans DOM et peut être visible ou masqué. D'un autre côté, isDisplayed () vérifie si l'élément existe dans DOM et est également visible.

EC.presence () vs EC.visibilityOf ()

presence () vérifie un élément présent dans le DOM et peut être visible ou non.
visibilityOf() vérifie un élément présent dans le DOM et également visible c'est à dire à la hauteur et la largeur.

Intégration projet Protractor avec d'autres outils

Il existe plusieurs outils disponibles pour améliorer encore plus vos cas de test telques Git ,Jenkins, BrowserStack… ces outils ajoutent une valeur significative à vos scripts de test Protractor.

Git

Git est un outil de contrôle de version très puissant. Il est toujours recommandé de conserver votre code dans Git si plusieurs développeurs sont impliqués.

Jenkins

Jenkins est un outil d'intégration continue avec lequel vous pouvez planifier vos cas de test et l'exécuter selon vos besoins. Les scripts de Protractor peuvent également être configurés avec Jenkins. La meilleure utilisation de l'exécution de vos cas de test sur Jenkins est qu'elle est très rapide et vous pouvez également exécuter plusieurs cas de test à la fois.

BrowserStack

BrowserStack est un outil de test multi-navigateur qui peut également être utilisé pour tester vos applications sur différents navigateurs. Il peut également être intégré à Protractor en ajoutant les informations d'identification browserStack dans votre fichier de configuration.

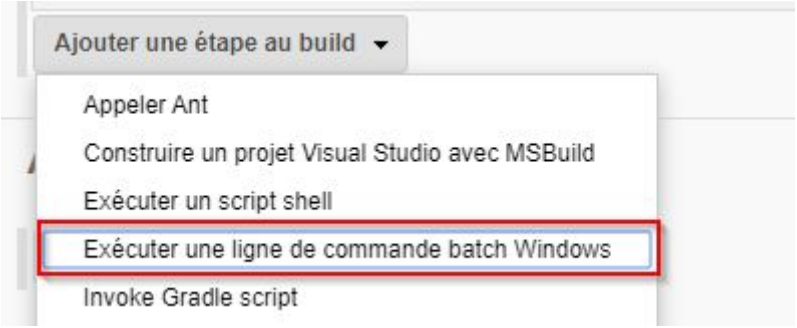
Exécution du Protractor avec Jenkins

Pour le lancement et le suivi des résultats de test nous avons utilisé l’outil Jenkins

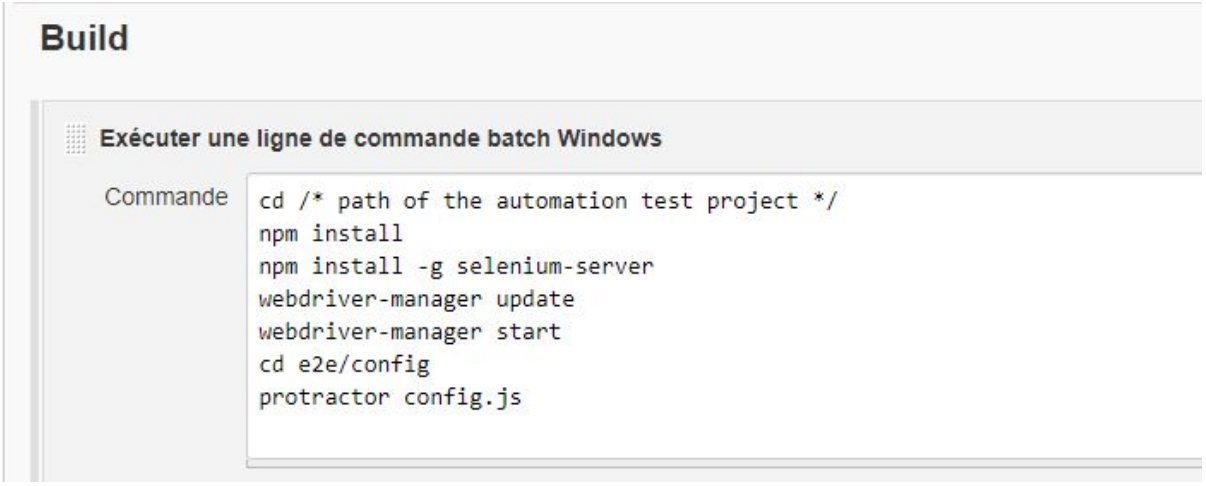
Jenkins

Tout développeur de scripts d'automatisation voudrait exécuter ses scripts régulièrement et garder une trace des résultats et Il est très probable qu'il ajoute continuellement plus de scripts au projet et souhaite qu'ils soient également suivis. Jenkins agit comme un outil de gestion de build qui gère chaque build de projet logiciel de manière continue. Facile à tester et à suivre les résultats de vos tests en un seul endroit.

Une solution d'automatisation protractor peut être intégré avec Jenkins de plusieurs manières simples. dans notre projet nous avons utilisé la commande batch windows pour l'exécution de notre solution et le lancement des build.

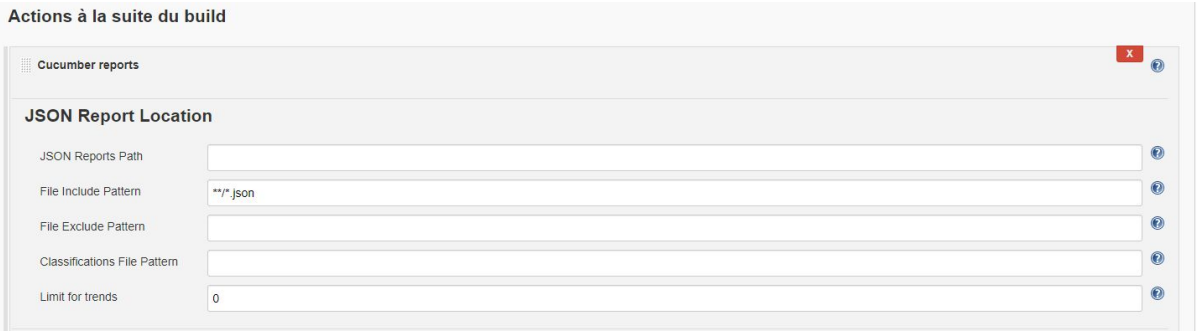


L'image ci-dessous montre les commandes utilisées pour configurer la solution d'automatisation avec jenkins:



Cucumber Report Jenkins

Pour la partie rapport d'exécution des tests nous avons utilisé le plugin cucumber report et nous avons ajouté la configuration nécessaire à notre projet créé.



Conclusion

Au cours des dernières années, nous avons assisté à une croissance massive dans le domaine de l'automatisation des tests et cette expérience nous a permis d’ouvrir un peu les yeux sur les nouvelle tendance en terme des technologies et nous a appris comment utiliser un outil puissant pour tester les scénarios de test de bout en bout d'une application Web comme protractor en intégrant l’approche BDD pour le préparation des cas de test et en utilisant le langage Typescript pour la rédaction des scripts de test.

Dans cet article, nous vous avons montré comment vous pouvez facilement mettre en place une solution d’automatisation de test avec les frameworks Protractor, Cucumber et le langage de scripting TypeScript.