2021

PUPPETEER WITH CUCUMER FRAMEWORK DOCUMEMTATION

V1.0

M. JUNAID AKHTER AND HAIDER ALI KHAN

Contents

PUPPETEER WITH CUCUMBER FRAMEWORK DOCUMENTATION

1. Framework Introduction

This framework is designed for testing purposes so we can test our NodeJs based web application easily. This framework consists mainly of puppeteer, chai and cucumber. Brief introduction of each is given below

2. Puppeteer Introduction

Puppeteer is a Node library which provides a high-level API to control Chrome or Chromium over the NodeJs web applications. Puppeteer runs headless by default, but can be configured to run full (non-headless) Chrome or Chromium.

Language use for Puppeteer testing is JavaScript. Puppeteer enables us to do most of the things that we can do manually in the browser like generate screenshots and PDFs of pages, Crawl a SPA (Single-Page Application) and generate pre-rendered content, Capture a timeline trace of our sites to help diagnose performance issues, Test Chrome Extensions.

2.1. Why Puppeteer

- **Headless Browser** - In Headless browsers state app processes without any user interface elements. Headless browsers are simply faster, consume less memory, are more flexible, and are steadier under automation stress.

- **Screenshot** - Puppeteer-screenshot-tester is a library in the puppeteer that allows screenshots to be generated while testing.

- **Performance Testing** - Puppeteer has a high-level API control over Chrome Developers Tools Protocol that enables recording of the Performance Timeline that when automated, reveals the problems associated with performance.

3. Cucumber Introduction

Cucumber is a testing framework that supports Behavior Driven Development (BDD). Cucumber offers a way to write tests in simple English language (called Gherkin) that anybody can understand, regardless of their technical knowledge. We can write multiple test scenarios in a single feature file.

3.1. What is Gherkin syntax

Gherkin uses a set of special keywords to give structure and meaning to executable specifications. Following keywords we can use for Gherkin Language.

- **Feature** - The first primary keyword of a scenario must always be Feature. Actually a short text that describes the feature.
- **Rule** - Optional Rule keyword has been part of Gherkin since v6. The purpose of the Rule keyword is to represent one *business rule* that should be implemented. It provides additional information for a feature.
- **Given** - Given is used to describe the initial context of the system - the *scene* of the scenario. It is typically something that happened in the *past*.
- **When** - When is used to describe an event, or an *action*. This can be a person interacting with the system, or it can be an event triggered by another system.
- **Then** - Then is used to describe an *expected* outcome, or result.
- **And, But** - We can write multiple Given, When and Then Or can make the example more fluidly structured by replacing the multiple Given, When and Then with And and But.
- **Background** - You may have repeating steps in each scenario, so can group them under a Background statement.
- **Scenario Outline** -Scenario Outline is used to run the same scenario multiple times, with different combinations of values.

3.2. What are Step Definitions?

Step definitions connect Gherkin steps to programming code. A step definition carries out the action that should be performed by the step.

4. Chai Introduction

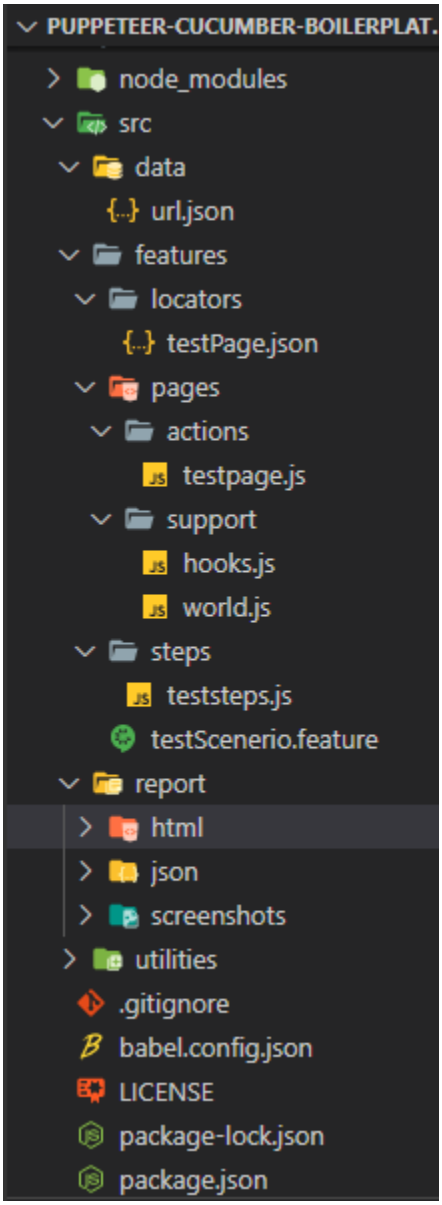
Chai is an BDD/TDD assertion library. It provides functions and methods that help you compare the output of a certain test with its expected value. Chai provides clean syntax that almost looks like English.

5. Prerequisites of the project

- Node (version : v14.17.3 or above) must be installed
- npm (version : 7.20.0 or above) must be installed
- Any IDE but preferably webstorm should be used
- Open project in cmd
- Enter npm install and press Enter Key
- Above command will add node modules folder and will install all dependencies
- Open project in webstorm and start working

6. Project Structure

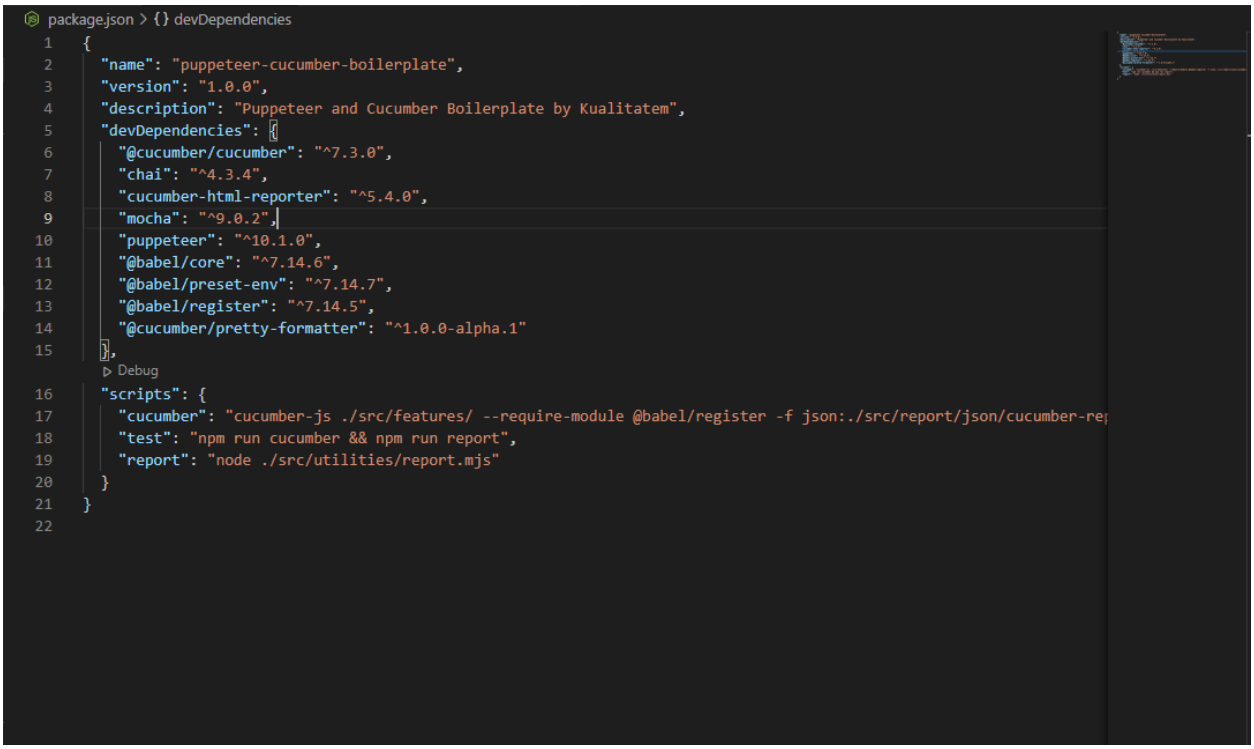
Our project structure looks like:



- **node_modules** - A node_modules folder present at top will be created when you run “npm install” command and will contain a list of all installed packages, dependencies and libraries etc.
- **src** - src folder is like a main folder which will contain subfolders like data, features, report and utilities.
- **data** - A subfolder under the src folder which will contain all test data files.
- **features** - A subfolder under the src folder which will contain subfolders like locators, pages, stepsand all feature files.
- **locators** - A subfolder under the feature folder which will contain json files of all possible locators (Actually locator is an instruction that points out the GUI Element).
- **pages** - A subfolder under the feature folder which will contain subfolders like actions and support.
- **actions** - A subfolder under the pages folder which will contain JS files of all possible actions (The files in which we perform our click, enter data actions).
- **support** - A subfolder under the pages folder which will contain hooks and world.js files.
- **hooks** - The files in which we perform our before and after actions.
- **world** - A file in which we can pass cucumber parameters and refer to it in our hooks and steps.
- **steps** - A subfolder under the feature folder which will contain JS files of all step definitions.
- **feature files** - Under the folder you can add different feature files where you can implement multiple scenarios in GHerkin language.
- **report** - A subfolder under the src folder which will contain three sub folders. Purpose of each folder is defined below.
- **html** - html folder will contain a result report generated by cucumber in html format.
- **json** - json folder will contain a result report generated by cucumber in json format.
- **screenshot** - screenshot folder will contain screenshots of failed scenarios.
- **utilities** - A subfolder under the src folder which will contain a subfolder and utility files. (Utility files basically contain all commonly used functions like click, input data, wait etc)
- **helper** - A subfolder under the utilities folder which will contain JS files of specific actions for specific events.

6.1. Package.json

Package.json will contain all libraries and tools etc. We just need to enter the command “npm install” in cmd and all dependencies will be installed and a node_modules folder will be created. Package.json also contains the command to run the scenario and generate the report Package.json will look like.



- scripts will contain the following :
    - cucumber will execute test cases.
    - test will run two commands.
        - “npm run cucumber” will execute all test cases.
        - “npm run report” will generate a report in html format while using a json file generated by the “npm run cucumber” command.
- devDependencies will contain the following :
- **@cucumber/cucumber** - A simple BDD framework that is already explained in the above section of the document.
- **chai** - An assertion library which is explained in the above section of the document.
- **cucumber-html-reporter** - It will help to generate html reports after execution.
- **Puppeteer** - A library which we are using to automate our test cases. Puppeteer is already explained in the above section of the document.