5. Using Page Object Model (POM) with Cucumber

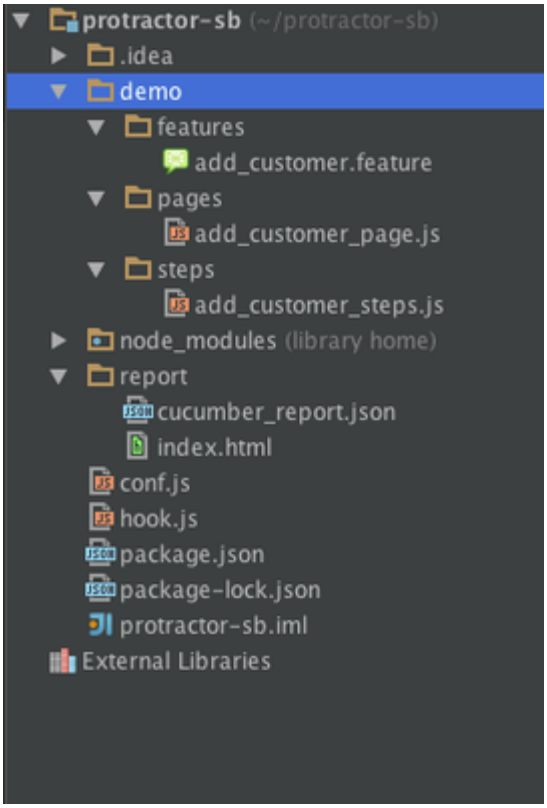
- Structure of project
- Create a Node project
- Create 2 files json to get the dependences (package.json , package-lock.json)
- Create config file
- Create feature file
- Create definition file
- Create Page file
- Create Cucumber Report file

A Page Object Model (POM) is a type of design pattern. This pattern is very popular within the Selenium ecosystem, so you may have come across it before.

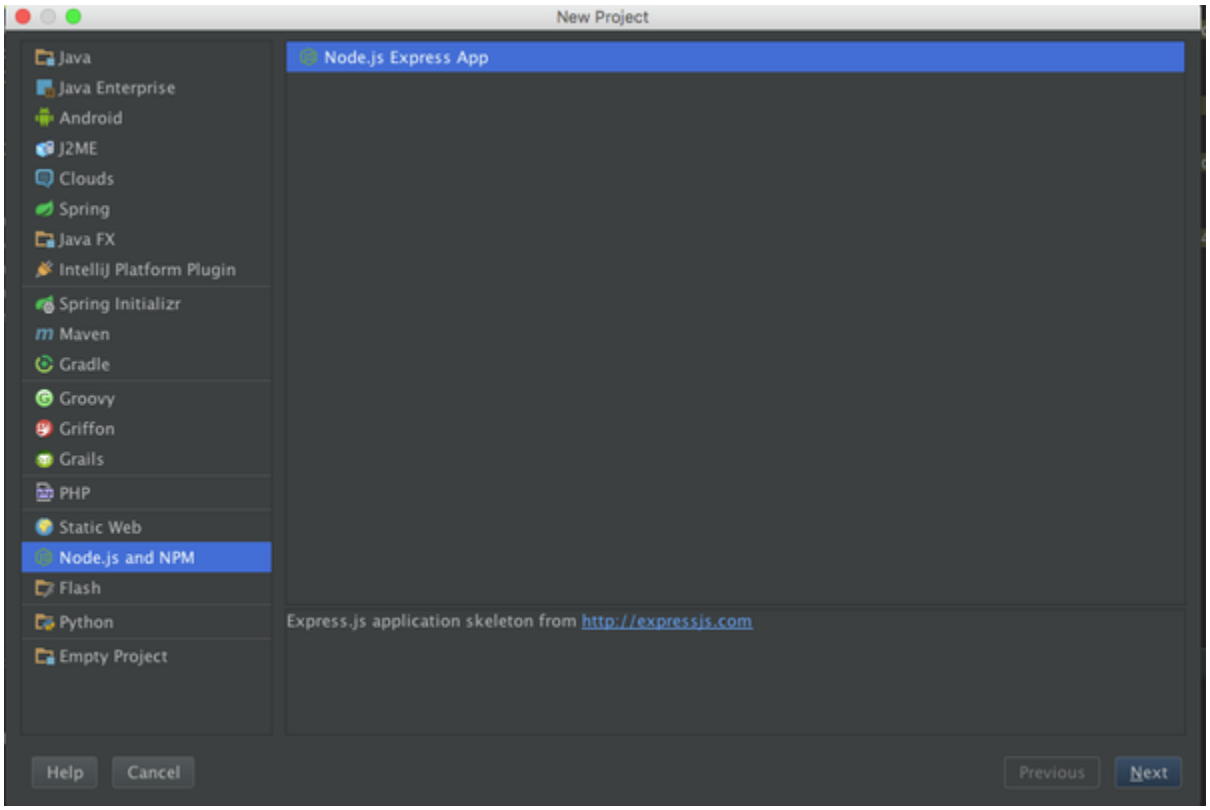
A page object is a class that simply stores your page elements. Elements and methods are housed in the page object file. This pattern allows testers to write clean code, avoid duplication, and better maintain their suites.

*chai list http://chaijs.com/api/bdd/

- **Structure of project**



- **Create a Node project**



- **Create 2 files json to get the dependences (package.json , package-lock.json)**

- **package.json**

```
{
  "name": "basic-cucumber-sb",
  "description": "Example of using Protractor with Cucumber and Page
Objects",
  "repository": "",
  "license": "Apache-2.0",
  "devDependencies": {
    "chai": "~3.5.0",
    "chai-as-promised": "~6.0.0",
    "cucumber": "~1.3.2",
    "protractor": "^5.1.1",
    "protractor-cucumber-framework": "^3.0.0"
  },
  "scripts": {
    "prestart-webdriver": "webdriver-manager update",
    "start-webdriver": "webdriver-manager start",
    "protractor": "protractor conf.js"
  }
}
```

- **package-lock.json**

- **cd project folder**
- ***npm install*** *to download the dependencies*

- **Create config file**

conf.js file

```
var jsonReportFile = 'protractor-report/cucumber_report.json';
exports.config = {
    seleniumAddress: 'http://localhost:4444/wd/hub',

    framework: 'custom',
    frameworkPath:
require.resolve('protractor-cucumber-framework'),

    specs: [
        'demo/features/*.feature'
    ],
    cucumberOpts: {
        require: ['hook.js','demo/steps/*_steps.js'],
        format: 'pretty'
    },
}
```

- **Create feature file**

AddCustomer.feature

```
Feature: Create Customer Feature
  As a user of XYZ Bank,
  I want to open an account
  So that I have to create an account firstly
  Scenario: Create the customer successfully
    Given I open the website
http://www.globalsqa.com/angularJs-protractor/BankingProject/#/manager/a
ddCust
    Then I verify that add customer screen appears
    When I type first name as Thuy , last name as Nguyen , postcode as
05550
    When I click submit button
    Then I check the customer first name as Thuy , last name as Nguyen ,
postcode as 05550 has been existed in the customer list

  Scenario: Create the customer successfully
    Given I open the website
http://www.globalsqa.com/angularJs-protractor/BankingProject/#/manager/a
ddCust
    Then I verify that add customer screen appears
    When I type first name as Hong , last name as Ho , postcode as 05147
    When I click submit button
    Then I check the customer first name as Hong , last name as Ho ,
postcode as 05147 has been existed in the customer list
```

- **Create definition file**

add_customer_steps.js

```
/**
 * Created by nguyenthihongthuy on 30/08/17.
 */

var chai = require('chai');
var chaiAsPromised = require('chai-as-promised');



chai.use(chaiAsPromised);
var expect = chai.expect;

var AddCustomerSteps = function () {

    var AddCustomerPage = require("../pages/add_customer_page.js");
```

```
    this.World = function MyWorld() {
        this.page = new AddCustomerPage();
    };

    this.Given('I open the website $website', function (website,
callback) {
        this.page.getURL(website);
        browser.sleep(1000)
        callback();
    });

    this.Then('I verify that add customer screen appears', function
(callback) {
        expect(element(by.model('fName'))).to.be.present;
        callback();
    });

    this.When('I type first name as $fname , last name as $lname ,
postcode as $code', function (fname, lname, code, callback) {
        this.page.setFirstName(fname);
        this.page.setLastName(lname);
        this.page.setPostCode(code);

        callback();
    });


    this.When('I click submit button', function (callback) {
        this.page.clickAddCustomer();
        callback();
    });

    this.Then('I check the customer first name as $fname , last name as
$lname , postcode as $code has been existed in the customer list',
function (fname, lname, code,callback) {

        callback();
    });
```

```
};

module.exports = AddCustomerSteps;
```

- **Create Page file**

add_customer_page.js

```
/**
 * Created by nguyenthihongthuy on 30/08/17.
 */

var AddCustomerPage = function() {

    this.getURL = function(value) {
        browser.get(value);
        browser.sleep(5000);
    };

    this.setFirstName = function(value) {
        element(by.model('fName')).sendKeys(value);
        browser.sleep(1000);
    };

    this.setLastName = function(value) {
        element(by.model('lName')).sendKeys(value);
        browser.sleep(1000);
    };

    this.setPostCode = function(value) {
        element(by.model('postCd')).sendKeys(value);
        browser.sleep(1000);
    };


    this.clickAddCustomer = function() {
        element.all(by.buttonText('Add Customer')).get(1).click();
        browser.sleep(1000);
        browser.switchTo().alert().accept();

    }

    this.test = function() {
        browser.sleep(5000);
        element(by.model('lName')).isDisplayed();
    }

};

module.exports = AddCustomerPage;
```

- **Create Cucumber Report file**

  hook.js

```
var Cucumber = require("cucumber");
var fs = require("fs");
module.exports = function() {
    var outputDir = './report';
    this.After(function (scenario, callback) {
        if (scenario.isFailed()) {
            browser.driver.takeScreenshot().then(function (base64png) {
                var decodedImage = new Buffer(base64png,
'base64').toString('binary');
                scenario.attach(decodedImage, 'image/png');
                callback();
            }, function (err) {
                callback(err);
            });
        } else {
            callback();
        }
    });

    var createHtmlReport = function (sourceJson) {
        var CucumberHtmlReport = require('cucumber-html-report');
        var report = CucumberHtmlReport.create({
            source: sourceJson, // source json
            dest: outputDir // target directory (will create if not
exists)
        });
    };

    var JsonFormatter = Cucumber.Listener.JsonFormatter();
    JsonFormatter.log = function (string) {
        if (!fs.existsSync(outputDir)) {
            fs.mkdirSync(outputDir);
        }

        var targetJson = outputDir + '/cucumber_report.json';
        fs.writeFile(targetJson, string, function (err) {
            if (err) {
                console.log('Failed to save cucumber test results to
json file.');
                console.log(err);
            } else {
                createHtmlReport(targetJson);
            }
        });
    };

    this.registerListener(JsonFormatter);
}
```