Hellofresh-Web-Test (Including Framework)

Prerequisite Setup Instructions:

1 Clone testNg reporting project and build local machine:

- https://github.com/shiwanthalakmal/testng-reporter-Maven-Plugin.git
- Build this reporting project using `mvn clean install` command

2 Download chrome driver latest version (Tested with `chromedriver-2.43`) Here and set driver location `config.properties` file using given format.

3 Log4J report will generate under local machine `C: drive -> log` dir, if C: not available in local machine please go and update `log4j.properties` file under test->resources folder

- log4j.appender.file.File=C:\log\shopping-app-project.log

Setup HelloFresh-Web-Test Project

1 clone hellofresh-web-test project and follow below steps

- https://github.com/shiwanthalakmal/hellofresh-web-test.git
- Execute `mvn idea:clean idea:idea` command to clean project and convert into .idea type project to open using InteliJIdea.
- Open **hellofresh-web-test** project using InteliJIdea tool and resolve all required external dependencies.
- Run individual test level, class level and suite level(using testng.xml) with InterliJIdea.

`**Note: Loggers and Reporter files only generate through the command line execution**`

Execute HelloFresh-Web-Test Project (Command line)

1 clone and hellofresh-web-test project and follow below steps (*above prerequisite needed)

- git clone hellofresh-web-test
- Run `mvn clean install -Pdemo` to execute configure demo test suite with maven profile
- Generate real time log under `c:->log` dir as well as `console` out put.
- End of the execution comprehensive report will generate under project target dir `taeget`

`**Note: Possible to manage test suites using maven profile easily**`

Feature of the framework (framework also comes with test project)

- ☑ Advance comprehensive reporting with executed steps(including root cause analysis with error categorization)
- ☑ Console logger and Reporter logger support for root cause analysis.
- ☑ Simplified test level (page-object & test-case level) using proper page-object & framework design
- ☑ External Test data maintain test case basis (as master data and test specific data)
- ☑ Tester can write test cases without knowing selenium or java
- ☑ Introduce page-object + fluent design pattern to write chaining based test cases to improve testcase readability
- ☑ Support pararal execution using maven profile and testNG
- ☑ Command line execution support and jenkins CI/CD pipeline integration support
- ☑ TestPlan management using maven profile with testNG.xml
- ☑ Execution listener (TestNG Listener) monitor and track test execution states and information
- ☑ Random Test data creation dynamically by using `Faker` Api while during the execution
- ☑ Introduces "page" and "panel" concepts to reduce simplify the page-object level and reduce duplications
- ☑ Singleton Driver initialization and maintain same driver through the all pages using abstract "BasePage"
- ☑ Introduce 'WorkingMemory' concept to track page-object input values while during the execution and verify that values under different page-object
- ☑ Fully keyboard utility support to perform execution smoothly
- ☑ Generate scree-shot for fail test case with testcase name
- ☑ Cross browser testing support
- ☑ WebDriver provide using factory design patten
- ☑ Introduce advance generic web-element smart wait concept and its managed framework level implicitly (test level no need to worry about explicit waits)
- ☑ Exception categorization minimize test maintenance effort and advance root cause analysis mechanism
- ☑ Using Decorator dsign pattern simplify web element action complexity and reduce page-object complexity and enhance scripting speed and readability