****

**TÀI LIỆU MÔ TẢ HỆ THỐNG AUTOMATION**

**--- ---**

GHI NHẬN THAY ĐỔI

**A**: Thêm mới; **E**: Chỉnh sửa; **D**: Xóa

|  |  |  |  |
| --- | --- | --- | --- |
| **Ngày** | **\*A/E/D** | **Diễn giải** | **Phiên bản** |
| 07/12/2023 | A | Tạo mới tài liệu | 1.0 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

DUYỆT TÀI LIỆU

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên** | **Chữ ký** | **Vị trí** | **Ngày** |
|  |  |  |  |
|  |  |  |  |

**MỤC LỤC CHỨC NĂNG**

[1 công nghệ sử dụng 5](#_Toc152928283)

[2 Cấu trúc source code 5](#_Toc152928284)

[2.1 Tổ chức cây thư mục 5](#_Toc152928285)

[2.2 Design pattern 6](#_Toc152928286)

[2.3 Mô hình hoạt động của file test: 6](#_Toc152928287)

[3 Cấu hình hệ thống 6](#_Toc152928288)

[3.1 Sử dụng Maven để tạo thư viện 6](#_Toc152928289)

[3.2 Cấu trúc file testcase 7](#_Toc152928290)

[3.3 Triển khai code 7](#_Toc152928291)

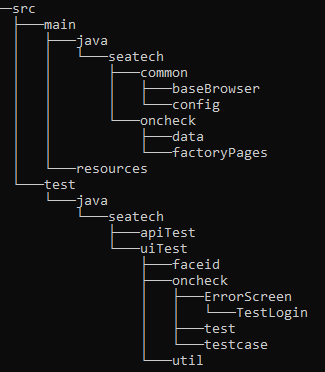
[3.4 Chạy testcase 8](#_Toc152928292)

# công nghệ sử dụng

* Intellij
* Java ( jdk 11)
* Maven
* Selenium (version 4.15.0)
* Testng (version 7.4.0)

# Cấu trúc source code

## Tổ chức cây thư mục



* Folder main.java.seatech: chứa thiết lập chung cho hệ thống, thiết lập data truyền vào, và page test dành cho từng hệ thống
* Folder test.java.seatech:
* chứa thiết lập test loại test gì ( ui test hay api test )
* test.java.seatech.system\_name: cấu hình test cho từng hệ thống
  + - * test.java.seatech.system\_name.ErrorScreen: folder chứa ảnh chụp màn hình các trường hợp test sai
      * test.java.seatech.system\_name.test: nơi viết code test theo testcase
      * test.java.seatech.system\_name.testcase: chứa test case

## Design pattern

* Page Object Model (POM):
  + Hỗ trợ khởi tạo PageObject tương tác với UI web
  + Tái sử dụng các thao tác với UI web ở nhiều test case khác nhau
  + dễ dàng nâng câp hoặc sửa lỗi
* Factory:
  + Khởi tạo PageObject dựa trên từng màn hình

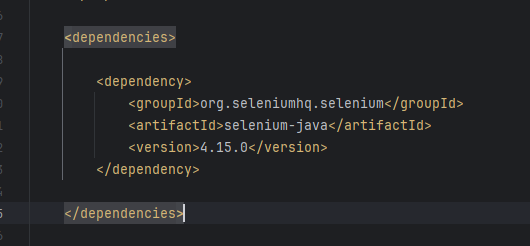
## Mô hình hoạt động của file test:

* Map dữ liệu từ file csv -> khởi tạo browser + tạo page object -> chạy test với dữ liệu đã được map từ file csv -> kiểm tra đúng sai
  + nếu đúng -> pass test -> tắt browser -> chạy tiếp những testcase khác
  + nếu sai -> chụp ảnh màn hình test sai -> tắt brower -> chạy tiếp những testcase khác

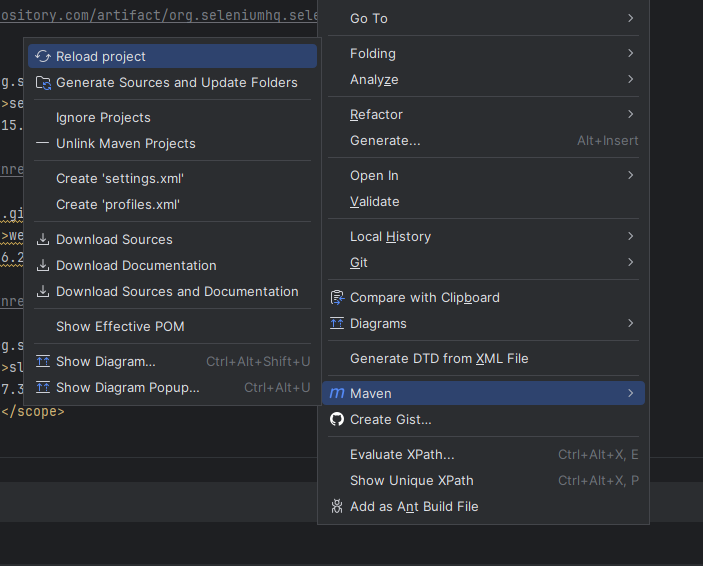
# Cấu hình hệ thống

## Sử dụng Maven để tạo thư viện

* Khai báo dependency trong file pom.xml, các dependency phải được đặt bên trong thẻ dependencies



* Khi khai báo xong cần reload lại Maven để thử viện được thêm vào project: chuột phải màn hình -> Maven -> Reload project



* Trang tìm dependency để khai báo: <https://mvnrepository.com/>

## Cấu trúc file testcase

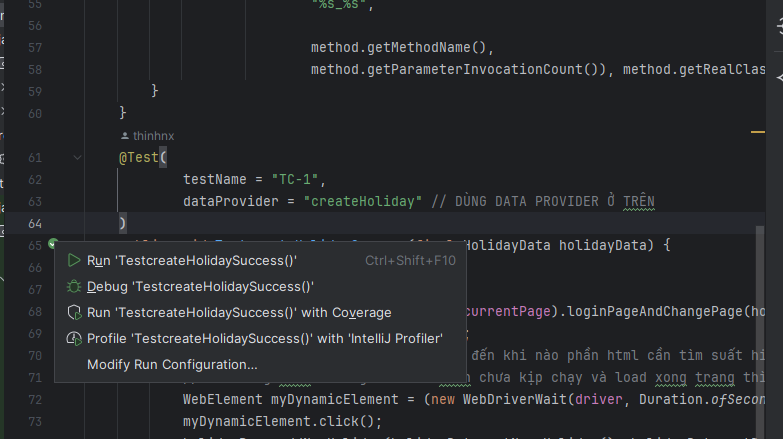
* Testcase sẽ được viết trong file csv
* Các cột bắt buộc phải có trong file csv: Test Case ID, Test Case Description, Condition
* Ngoài các cột trên có thể thêm các cột khác để sử dụng cho việc đọc dữ liệu và test

## Triển khai code

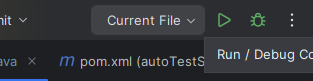
* **Config.properties**: thiết lập tham số, các thông tin mặc định sẽ sử dụng trong hệ thống
* **Main.java.seatech.common.browser**:
  + Khai báo các browser Driver sẽ sử dụng khi chạy ứng dụng, mỗi browser sẽ có 1 driver riêng và Options riêng để cấu hình
  + Sử dụng browser nào khi chạy code sẽ được thiết lập ở file **config.properties**
* **Main.java.seatech.common.config**:
  + đọc thông tin từ file **config.properties**
* **Main.java.seatech.system\_name.data**:
  + Map dữ liệu từ file csv
  + BaseData.java: Map những cột bắt buộc phải có trong csv (Test Case ID, Test Case Description, Condition)
  + Những file java còn lại sẽ kế thừa từ BaseData và map những cột còn lại ( những cột này sẽ tùy thuộc vào từng trang web cần những field nào )
* **Main.java.seatech.system\_name.factoryPages**:
  + BasePage: Khởi tạo page Object
  + PageFactory: Hỗ trợ tạo pageObject dựa theo màn hình muốn tạo
  + Các Page khác sẽ kế thừa từ BasePage và thiết lập các sự kiện ứng với từng UIweb
  + Chú thích decorator: @FindBy() : tìm kiếm attribute trên giao diện web hoặc tìm kiếm theo css, xpath
* **Test.java.uiTest.testname**: file viết testcase cần cấu hình các phương thức mặc định sau:
  + Data provider: dung để lấy dữ liệu từ file csv và map vào class đã tạo
  + Setup: khởi tạo browser và UI web
  + captureScreenshotOnFaiure: chụp ảnh màn hình nếu test case đó bị sai
  + teardown: đóng brower mỗi khi thực hiện xong 1 testcase
  + chú thích decorater:
    - @DataProvider( name = “”): phân biệt testcase sẽ sử dụng data nào để test
    - @BeforeMethod: luôn chạy trước khi mỗi testcase chạy
    - @Aftermethod: luôn chạy sau khi testcase chạy xong
    - @Test(testname= “”, dataProvider = “”): hàm chạy test case với testname là tên Test Case ID được liệt kê trong file csv, dataProvider: tên data provider được thiết lâp ở trên
* **Test.java.seatech.uiTest.util.runall.xml**: Cấu hình chạy test case, cho phép chạy song song, chạy tất cả testcase …

## Chạy testcase

* Chạy từng testcase: muốn chạy riêng test case nào thì vào file test -> chọn hàm cần test -> bấm run



* Chạy toàn bộ test case của 1 class: giống như chạy từng test case hoặc chọn vào clas muốn chạy và bấm run trên thanh công cụ ( file chạy chọn là current file)



* Chạy toàn bộ test case: sử dụng file runall.xml và chọn run

