**Selenium Page Factory  
Page Generator Manager**

***DAO MINH DAM  
Email:***[*daominhdam@gmail.com*](mailto:daominhdam@gmail.com)

***Skype:****daominhdam (Đào Minh Đảm)*

[***https://automationfc.vn***](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://automationfc.vn%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628670500556000%26amp;usg%3DAOvVaw2vYmQ2PJs_Q_e4V8yaHucn&sa=D&source=editors&ust=1628670500633367&usg=AOvVaw0x2-IS2Hun6bCGbGWbvenP)*-*[***https://automationfc.com***](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://automationfc.com%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628670500557000%26amp;usg%3DAOvVaw2WPHnvkkifKFeTLKz6f4sh&sa=D&source=editors&ust=1628670500633483&usg=AOvVaw0wDzonsVe0hQ5b1IPF1HfP)

***Tài liệu này chỉ sử dụng trong khóa học - vui lòng không chia sẻ công khai***  
-------------------------------------------------------------------------------------------------------------------------------

**Page Factory (Selenium)**

* **FindBy/ FindBys/ FindAll/ PageFactory/ CacheLookup**

|  |
| --- |
| import org.openqa.selenium.support.FindBy; import org.openqa.selenium.support.FindBys; import org.openqa.selenium.support.FindAll; import org.openqa.selenium.support.PageFactory;  import org.openqa.selenium.support.CacheLookup; |

* **FindBy -**Sử dụng để locate 1 hoặc nhiều element thỏa mãn 1 điều kiện

|  |
| --- |
| @FindBy(how = How.XPATH, using = "//form[@name='frmLogin']") private WebElement loginForm;  @FindBy(how = How.NAME, using = "uid") // @FindBy(name = "uid") private WebElement userIDTextbox;          @FindBy(how = How.CSS, using = "input[name='btnLogin']") // @FindBy(css = "input[name='btnLogin']") private List <WebElement> loginButton; |

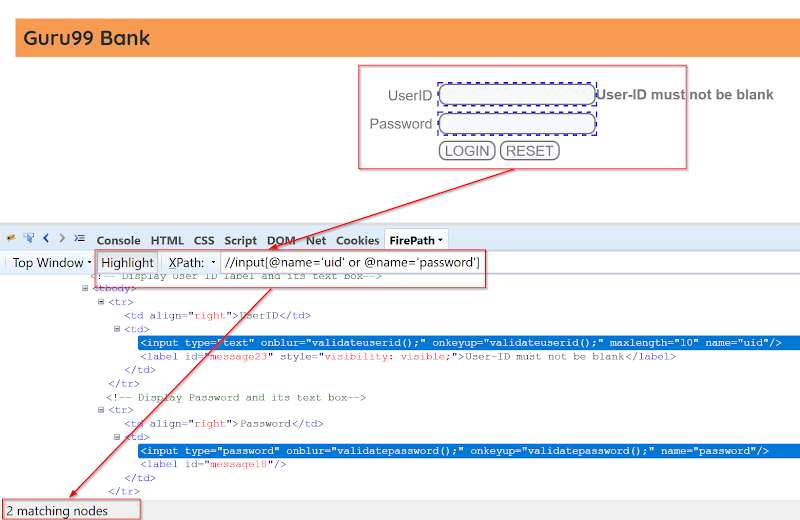
* **FindBys -**Thỏa mãn nhiều điều kiện (AND) - 1 trong 2 sai là ko tìm thấy

|  |
| --- |
| @FindBys({         @FindBy(how = How.NAME, using = "uid"),         @FindBy(how = How.NAME, using = "password") }) private List <WebElement> bothCriteria; |

* **FindAll -**Thỏa mãn nhiều điều kiện (OR) - 1 trong 2 đúng là tìm thấy

|  |
| --- |
| @FindAll({             @FindBy(how = How.NAME, using = "uid"),             @FindBy(how = How.NAME, using = "password")         }) private List<WebElement> eitherCriterion; |

|  |
| --- |
| System.out.println("Using @FindBy, we found " + loginButton.size() + " element(s)"); System.out.println("Using @FindBys, we found " + bothCriteria.size() + " element(s)"); System.out.println("Using @FindAll, we found " + eitherCriterion.size() + " element(s)");  Using @FindBy, we found 1 element(s) Using @FindBys, we found 0 element(s) Using @FindAll, we found 2 element(s) |



* **PageFactory.initElements(driver, this);**
* Khi khởi tạo - các WebElement sẽ chưa findElement. Khởi tạo chỉ để tạo ra LIÊN KẾT giữa WebElement và Locator. Khi nào WebElement đó được sử dụng (gọi - call) thì mới được tìm kiếm dựa trên các loại WebElement đã được khai báo ở trên
* Nếu WebElement đó được sử dụng nhiều lần thì nó được tìm kiếm nhiều lần. Nếu muốn chỉ tìm 1 lần rồi sử dụng lại - sử dụng annotation @CacheLookup (tăng perfomance)
* **CacheLookup**
* Cho phép element chỉ được lookup 1 lần duy nhất (Được cache lại - chỉ sử dụng trong trường hợp element luôn cố định trên page - ko nên sử dụng khi DOM change/ page refresh/ element refresh)
* **How**
* CLASS\_NAME
* CSS
* ID
* ID\_OR\_NAME
* LINK\_TEXT
* NAME
* PARTIAL\_LINK\_TEXT
* TAG\_NAME
* XPATH
* **PageFactory (Viết lại testcase register/ login theo dạng PageFactory của Selenium)**
* LoginPageFactory

|  |
| --- |
| package bankguru.pageFactory;  import java.util.List; import org.openqa.selenium.WebDriver; import org.openqa.selenium.WebElement; import org.openqa.selenium.support.FindBy; import org.openqa.selenium.support.How; import org.openqa.selenium.support.PageFactory; import commons.AbstractPage;  public class LoginPageFactory extends AbstractPage {         private WebDriver driver;          @FindBy(how = How.XPATH, using = "//form[@name='frmLogin']")         private WebElement loginForm;          @FindBy(how = How.NAME, using = "uid")         private WebElement userIDTextbox;          @FindBy(how = How.NAME, using = "password")         private WebElement passwordTextbox;          @FindBy(css = "input[name='btnLogin']")         private List<WebElement> loginButton;          @FindBy(how = How.XPATH, using = "//a[text()='here']")         private WebElement hereLink;          @FindBy(how = How.NAME, using = "emailid")         private WebElement emailIDTextbox;          public LoginPageFactory(WebDriver mappingDriver) {                 driver = mappingDriver;                 PageFactory.initElements(driver, this);         }          public boolean isLoginFormDisplayed() {                 return loginForm.isDisplayed();         }          public String getLoginPageUrl() {                 return getCurrentPageUrl(driver);         }          public void clickToHereLink() {                 hereLink.click();         }          public void inputToUserIDTextbox(String userIdInfor) {                 userIDTextbox.sendKeys(userIdInfor);         }          public void inputToPasswordTextbox(String passwordInfor) {                 passwordTextbox.sendKeys(passwordInfor);         }          public void clickToLoginButton() {                 loginButton.get(0).click();         }          } |

* RegisterPageFactory

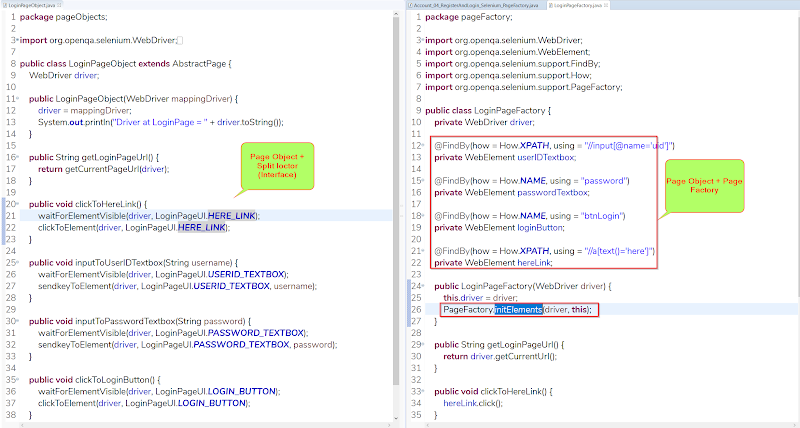
|  |
| --- |
| package bankguru.pageFactory;  import org.openqa.selenium.WebDriver; import org.openqa.selenium.WebElement; import org.openqa.selenium.support.FindBy; import org.openqa.selenium.support.How; import org.openqa.selenium.support.PageFactory; import commons.AbstractPage;  public class RegisterPageFactory extends AbstractPage {         private WebDriver driver;          @FindBy(how = How.XPATH, using = "//input[@name='emailid']")         private WebElement emailIDTextbox;          @FindBy(how = How.NAME, using = "btnLogin")         private WebElement submitButton;                  @FindBy(how = How.XPATH, using = "//td[text()='User ID :']/following-sibling::td")         private WebElement userIDText;                  @FindBy(how = How.XPATH, using = "//td[text()='Password :']/following-sibling::td")         private WebElement passwordText;          public RegisterPageFactory(WebDriver mappingDriver) {                 driver = mappingDriver;                 PageFactory.initElements(driver, this);         }                  public boolean isRegisterPageDisplayed() {                 return emailIDTextbox.isDisplayed();         }                  public void inputToEmailIDTextbox(String email) {                 emailIDTextbox.sendKeys(email);         }                  public void clickToLoginButton() {                 submitButton.click();         }                  public void openLoginPage(String loginPageUrl) {                 openAnyUrl(driver, loginPageUrl);         }                  public String getUserIDInfor() {                 return userIDText.getText();         }          public String getPasswordInfor() {                 return passwordText.getText();         } } |

* HomePageFactory

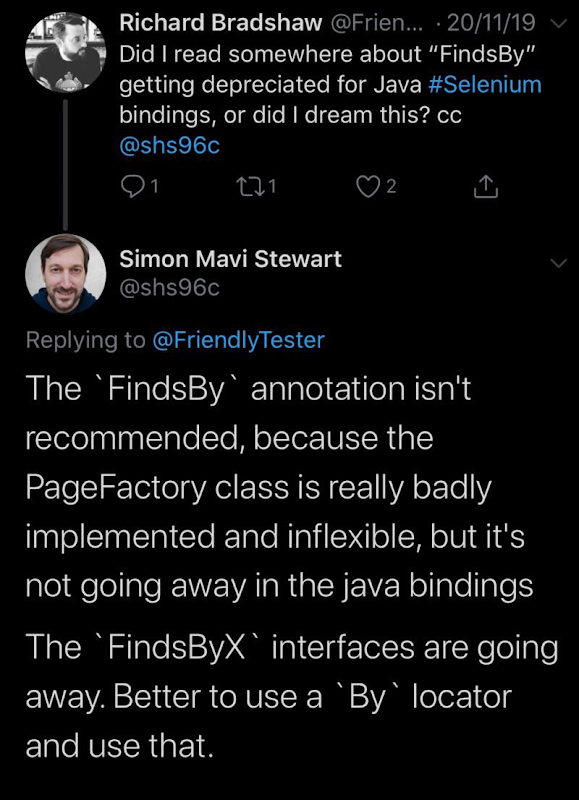
|  |
| --- |
| package bankguru.pageFactory;  import org.openqa.selenium.WebDriver; import org.openqa.selenium.WebElement; import org.openqa.selenium.support.FindBy; import org.openqa.selenium.support.How; import org.openqa.selenium.support.PageFactory; import commons.AbstractPage;  public class HomePageFactory extends AbstractPage {         private WebDriver driver;          @FindBy(how = How.XPATH, using = "//marquee[text()=\"Welcome To Manager's Page of Guru99 Bank\"]")         private WebElement welcomeMessage;                  public HomePageFactory(WebDriver mappingDriver) {                 driver = mappingDriver;                 PageFactory.initElements(driver, this);         }                  public boolean isWelcomeMessageDisplayed() {                 return welcomeMessage.isDisplayed();         } } |

* Account\_Level\_04\_PageFactory\_Selenium

|  |
| --- |
| package com.bankguru.account;  import java.util.Random; import java.util.concurrent.TimeUnit; import org.openqa.selenium.WebDriver; import org.openqa.selenium.firefox.FirefoxDriver; import org.testng.Assert; import org.testng.annotations.AfterClass; import org.testng.annotations.BeforeClass; import org.testng.annotations.Test; import bankguru.pageFactory.HomePageFactory; import bankguru.pageFactory.LoginPageFactory; import bankguru.pageFactory.RegisterPageFactory;  public class Account\_Level\_04\_PageFactory\_Selenium {         private WebDriver driver;         String loginPageUrl, userIdInfor, passwordInfor, email;         LoginPageFactory loginPage;         RegisterPageFactory registerPage;         HomePageFactory homePage;          @BeforeClass         public void beforeClass() {                 driver = new FirefoxDriver();                 System.out.println(driver.toString());                 driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);                 driver.manage().window().maximize();                 driver.get("http://demo.guru99.com/v4/");                 email = "selenium09" + randomNumber() + "@gmail.com";         }          @Test         public void TC\_01\_RegisterToSystem() {                 loginPage = new LoginPageFactory(driver);                 Assert.assertTrue(loginPage.isLoginFormDisplayed());                                  loginPageUrl = loginPage.getLoginPageUrl();                 loginPage.clickToHereLink();                                  registerPage = new RegisterPageFactory(driver);                 Assert.assertTrue(registerPage.isRegisterPageDisplayed());                                  registerPage.inputToEmailIDTextbox(email);                 registerPage.clickToLoginButton();                 userIdInfor = registerPage.getUserIDInfor();                 passwordInfor = registerPage.getPasswordInfor();         }          @Test         public void TC\_02\_LoginToSystem() {                 registerPage.openLoginPage(loginPageUrl);                                  loginPage = new LoginPageFactory(driver);                 Assert.assertTrue(loginPage.isLoginFormDisplayed());                                  loginPage.inputToUserIDTextbox(userIdInfor);                 loginPage.inputToPasswordTextbox(passwordInfor);                 loginPage.clickToLoginButton();                                  homePage = new HomePageFactory(driver);                 Assert.assertTrue(homePage.isWelcomeMessageDisplayed());         }          @AfterClass         public void afterClass() {                 driver.quit();         }          public int randomNumber() {                 Random random = new Random();                 return random.nextInt(999999);         } } |



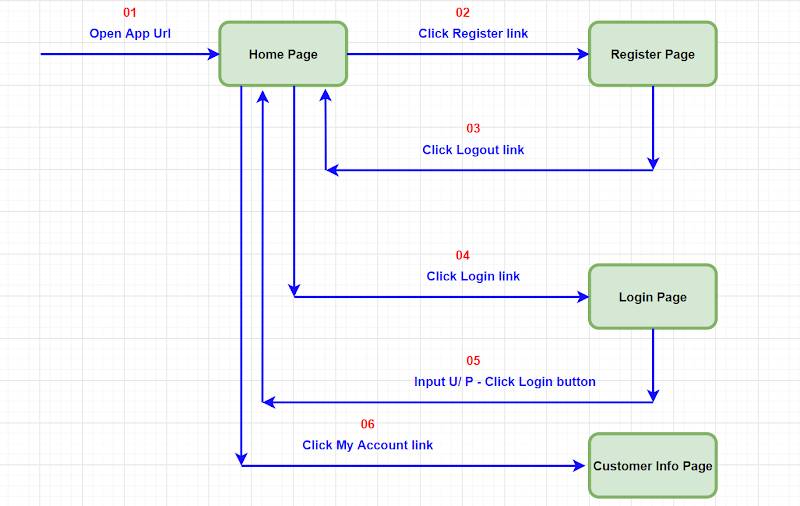
* **Advantaged**
* Tăng performance lên khi không cần tìm kiếm lại element nhiều lần trên cùng 1 page (annotation @CacheLookup)
* Che giấu được việc đi tìm element (findElement) - chỉ cần dùng 1 method PageFactory.**initElements**
* Class**AbstractPage** viết sẽ gọn hơn/ ít step hơn/ ko có việc findElement trong class này
* Cách design khá gọn nhẹ và dễ hiểu/ ít tầng (layer)/ bỏ được tầng pageUIs (interfaces) - đưa luôn việc định nghĩa/ tìm kiếm element vào trong 1 class
* **Disadvantaged**
* Gom cả việc **định nghĩa element** và**các sự kiện của page** vào chung 1 class dẫn đến việc khó quản lí khi page đó phình to ra (số lượng testcase/ function nhiều lên)
* Định nghĩa element (UI): 1 element = 3 lines of code
* Thao tác vs element (Action): 1 element = 5 lines of code
* 1 class page object: 100 action/ element = 800 lines of code
* Những model web/ mobile app hiện nay sử dụng các công nghệ như: Ajax/ Angular/ React/ VueJS/ NuxtJS/.. (Single Page Application) -> DOM dễ bị change (Page refresh/ element bị remove/ add thêm thuộc tính) -> Failed testcase: due stale element ([Link](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://www.seleniumhq.org/exceptions/stale_element_reference.jsp%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628670500580000%26amp;usg%3DAOvVaw1HM74NnQmV-6R_0-Qsuh8u&sa=D&source=editors&ust=1628670500639972&usg=AOvVaw3QZdicXQO8QydCDtQcx9x8))
* **Khó sử dụng dynamic locator (text) -> Page Factory sử dụng @FindBy (WebElement)**
* Không tiết kiệm thời gian khi scripting
* Không phù hợp với những dự án lớn/ chức năng nhiều và phức tạp
* Explicit Wait chỉ hỗ trợ những hàm với tham số là WebElement cho PageFactory - vì các hàm như By thì PageFactory ko apply được
* Take time để chờ cho element báo không tìm thấy (Invisible) - chờ hết timeout của implicit => Đang dùng các hàm wait của explicit với tham số là WebElement (Bị ảnh hưởng bởi implicit wait) => Solution là ko dùng implicitWait/ bỏ luôn/ chỉ dùng explicit wait
* [https://medium.com/@dheerajgambhir/findby-limitation-to-accept-only-static-string-24739b83faad](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://medium.com/@dheerajgambhir/findby-limitation-to-accept-only-static-string-24739b83faad%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628670500581000%26amp;usg%3DAOvVaw2xFaQ2ZIsSnU2mz5Jf-URo&sa=D&source=editors&ust=1628670500640300&usg=AOvVaw3DItA46q4cEDbge2mAegoV)



* **Reference**
* [https://github.com/SeleniumHQ/selenium/wiki/PageFactory](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://github.com/SeleniumHQ/selenium/wiki/PageFactory%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628670500582000%26amp;usg%3DAOvVaw1q9m4Yd9K4kIR9cdd2NXoS&sa=D&source=editors&ust=1628670500640484&usg=AOvVaw3QmQFAscwH4OWiERe0sCMv)
* [https://seleniumhq.github.io/selenium/docs/api/java/org/openqa/selenium/support/PageFactory.html](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://seleniumhq.github.io/selenium/docs/api/java/org/openqa/selenium/support/PageFactory.html%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628670500582000%26amp;usg%3DAOvVaw3twvDG1hlUIThbQ50WZtZk&sa=D&source=editors&ust=1628670500640572&usg=AOvVaw0SRjZE2-DRnkZWT410MHN8)
* [https://www.ontestautomation.com/findby-strategies-for-selenium-explained](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://www.ontestautomation.com/findby-strategies-for-selenium-explained/%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628670500583000%26amp;usg%3DAOvVaw1gznEpGdT994Dr9BZj5pMj&sa=D&source=editors&ust=1628670500640660&usg=AOvVaw0UwUYRtaXaxuVdgqzzZwJv)
* [https://www.seleniumeasy.com/selenium-tutorials/page-factory-pattern-in-selenium-webdriver](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://www.seleniumeasy.com/selenium-tutorials/page-factory-pattern-in-selenium-webdriver%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628670500583000%26amp;usg%3DAOvVaw2TWKVPyqSRyY1AGErxfpnM&sa=D&source=editors&ust=1628670500640750&usg=AOvVaw0EyhUaE47sjYe2UklKLa71)
* [https://blog.testproject.io/2016/09/06/test-automation-selenium-page-object-model-page-factory/](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://blog.testproject.io/2016/09/06/test-automation-selenium-page-object-model-page-factory/%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628670500584000%26amp;usg%3DAOvVaw3hlnK53ADXsXCFKPt0S4Py&sa=D&source=editors&ust=1628670500640837&usg=AOvVaw33QMi_xeMRsitY83rN6-Jl)
* [https://www.edureka.co/blog/page-object-model-in-selenium/](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://www.edureka.co/blog/page-object-model-in-selenium/%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628670500584000%26amp;usg%3DAOvVaw05F-qQeJbPXRxe6WO_JZ6s&sa=D&source=editors&ust=1628670500640926&usg=AOvVaw37ZB5dRu7s6m-Tloc1hL98)
* [https://www.swtestacademy.com/page-factory-selenium-webdriver/](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://www.swtestacademy.com/page-factory-selenium-webdriver/%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628670500585000%26amp;usg%3DAOvVaw3Mi7WihPi2vctzVV-PGV3u&sa=D&source=editors&ust=1628670500641043&usg=AOvVaw16nabi8d1ctVhZYexgUcZb)
* [https://www.toolsqa.com/selenium-webdriver/page-object-pattern-model-page-factory/](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://www.toolsqa.com/selenium-webdriver/page-object-pattern-model-page-factory/%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628670500585000%26amp;usg%3DAOvVaw05i_NAg1NiZvTENFIWBSaA&sa=D&source=editors&ust=1628670500641127&usg=AOvVaw01kq3WcQpiAOAzVaO5-z4s)
* [https://www.janbasktraining.com/blog/pom-with-page-factory-in-selenium-webdriver/](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://www.janbasktraining.com/blog/pom-with-page-factory-in-selenium-webdriver/%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628670500586000%26amp;usg%3DAOvVaw0m2hTMujTEePHmAheoMsAw&sa=D&source=editors&ust=1628670500641213&usg=AOvVaw1f0eBzYXI0fnM25UW7zYlH)
* [Selenium: State of the Union - Keynote by Simon Stewart @ Selenium Conf 2014](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://www.youtube.com/watch?v%253DFgG4SE-hB8g%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628670500586000%26amp;usg%3DAOvVaw3V_uNETA6GGPFgXUo_gTM8&sa=D&source=editors&ust=1628670500641296&usg=AOvVaw1W5V8cMFXhGqW5abEfM5tn)
* [https://github.com/bhavikbpatel/SeleniumPageFactoryFramework](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://github.com/bhavikbpatel/SeleniumPageFactoryFramework%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628670500587000%26amp;usg%3DAOvVaw0Eevxai0st0jDOl7PYCBbZ&sa=D&source=editors&ust=1628670500641378&usg=AOvVaw2aDmEGbVzsqK42GAdpf_sS)
* [https://github.com/osandadeshan/selenium-testng-page-factory-extended](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://github.com/osandadeshan/selenium-testng-page-factory-extended%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628670500587000%26amp;usg%3DAOvVaw2IUohjKzZa5yfX1p8z77E9&sa=D&source=editors&ust=1628670500641499&usg=AOvVaw067CElFdVXMEq1ouwmEuFi)
* [https://github.com/swtestacademy/PageFactory](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://github.com/swtestacademy/PageFactory%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628670500588000%26amp;usg%3DAOvVaw0rl4tWhP65PQf1zfBBy41x&sa=D&source=editors&ust=1628670500641596&usg=AOvVaw2HM0kwv3Y-CBXFJzm76CZy)

**Page Generator Manager**

* Về bản chất (knowledge) của Page Object Pattern là khi chuyển từ page object class này qua class kia thì phải khởi tạo page class kia lên và có sự chuyển page



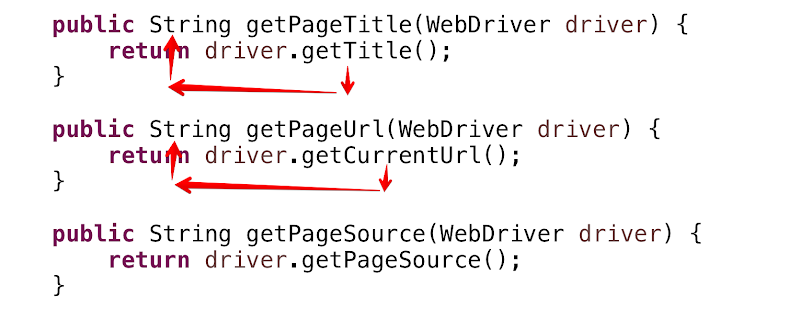
* **Quản lí việc khởi tạo đối tượng của page khi chuyển page**
* Page A -> Page B -> Page C -> Page A -> ...

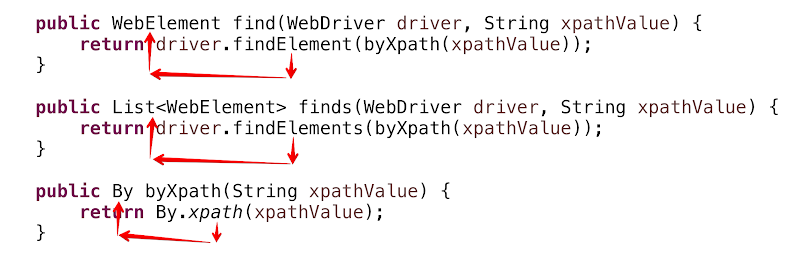


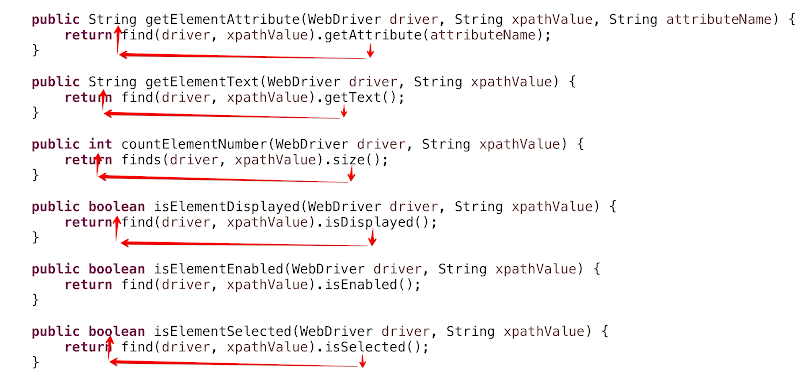
* **01 - Test Class**
* Khởi tạo page mới trong chính class test
* Sai nguyên tắc của OOP
* Che giấu đi việc khởi tạo của đối tượng (page object class) - nguyên tắc đóng gói [Encapsulation]
* Lặp lại việc khởi tạo page
* Ko có tính kết nối giữa 2 page với nhau - code ngắn gọn/ dễ đọc/ dễ hiểu hơn
* page mới = page cũ + action



* **02 - Page Object Class**
* Kiểu dữ liệu trả về cùng 1 kiểu với kiểu dữ liệu của hàm







* Ưu điểm:
* Che giấu sự khởi tạo của đối tượng ở tầng testcase
* Đưa việc khởi tạo vào page object class
* Có sự kết nối giữa các page vs nhau (manual step/ business) - ở trên tầng testcase
* Ko được mất kết nối giữa các step
* Ko được mất kết nối giữa 2 testcase (kết thúc testcase này sẽ là bắt đầu của testcase khác - 2 test đang phụ thuộc nhau - bố trí chạy theo flow)
* Nhược điểm:
* Khi đã apply/ build framework thì cái framework đó phải giải quyết triệt để được những vấn đề về trùng lặp code - phải tái sử dụng/ giảm sự trùng lặp
* Chưa xử lí được việc đối tượng được khởi tạo nhiều lần/ lặp lại
* Khởi tạo LoginPageObject 3 lần (hình dưới)



* **03 - PageGeneratorManager class**
* Gom việc khởi tạo của tất cả page object lại chung vào 1 class
* Dễ dàng cho việc bảo trì (update 1 page ảnh hưởng cho tất cả các chức năng đang sử dụng nó/ thêm page mới thì không ảnh hưởng đến các chức năng đang sử dụng/ tồn tại)
* Che giấu việc khởi tạo đối tượng ở cả testcase class và page object class

|  |
| --- |
| package commons;  import org.openqa.selenium.WebDriver; import bankguru.pageObjects.HomePageObject; import bankguru.pageObjects.LoginPageObject; import bankguru.pageObjects.RegisterPageObject;  public class PageGeneratorManager {         public static LoginPageObject getLoginPage(WebDriver driver) {                 return  new LoginPageObject(driver);         }          public static RegisterPageObject getRegisterPage(WebDriver driver) {                 return new RegisterPageObject(driver);         }          public static HomePageObject getHomePage(WebDriver driver) {                 return new HomePageObject(driver);         } } |



* Có sự kết nối giữa các page object vs nhau trên tầng testcase (giống như manual step)
* 1 - Login qua Register khi click vào here link
* 2 - Register về lại Login khi mở Url của LoginPage
* 3 - Login valid qa HomePage
* 4 - HomePage logout về lại Login



**Reference**

* [https://cppdeveloper.com/tutorial/tinh-dong-goi-encapsulation-trong-lap-trinh-huong-doi-tuong/](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://cppdeveloper.com/tutorial/tinh-dong-goi-encapsulation-trong-lap-trinh-huong-doi-tuong/%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628670500594000%26amp;usg%3DAOvVaw0BUcNlDEkN5RtgswTqLSEm&sa=D&source=editors&ust=1628670500643972&usg=AOvVaw3SGUqNagSJ_AAY3dQ6NDUi)
* [Lập trình hướng đối tượng – Wikipedia tiếng Việt](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://vi.wikipedia.org/wiki/L%2525E1%2525BA%2525ADp_tr%2525C3%2525ACnh_h%2525C6%2525B0%2525E1%2525BB%25259Bng_%2525C4%252591%2525E1%2525BB%252591i_t%2525C6%2525B0%2525E1%2525BB%2525A3ng%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628670500595000%26amp;usg%3DAOvVaw0Gh0V1cxg2_S1dxYDuqC-j&sa=D&source=editors&ust=1628670500644149&usg=AOvVaw1IIoWNu9mIZ4t2J6dW4t8h)
* [https://www.amazon.com/Selenium-Design-Patterns-Best-Practices/dp/1783982705](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://www.amazon.com/Selenium-Design-Patterns-Best-Practices/dp/1783982705%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628670500595000%26amp;usg%3DAOvVaw0FQVm29-MUH1wJTxIachAa&sa=D&source=editors&ust=1628670500644258&usg=AOvVaw1RO6FbP4YmXgS-Mv9iQkix)
* [https://blog.testproject.io/2020/06/29/design-patterns-in-test-automation/](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://blog.testproject.io/2020/06/29/design-patterns-in-test-automation/%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628670500596000%26amp;usg%3DAOvVaw1gnReaT3R64OKk1OWAna8M&sa=D&source=editors&ust=1628670500644418&usg=AOvVaw2WIDCh0maMHzqrycCcxyHx)
* [https://kobiton.com/book/chapter-12-test-automation-design-patterns-you-should-know/](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://kobiton.com/book/chapter-12-test-automation-design-patterns-you-should-know/%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1628670500596000%26amp;usg%3DAOvVaw1L0Y-T_YNF3ZYzTQSb8opF&sa=D&source=editors&ust=1628670500644571&usg=AOvVaw3owHfer0jg_7HWDXTr7YYz)