Selenium POM Homework

1. The following homework will include creating new project and importing all dependencies. The project name will be Selenium-Homework-FirstName and Last Name.
2. We will need to create 3 tests using the POM (Page Object Model) as follows:
3. On the following link: https://demoqa.com/links under category **Following links will send an api calls** click all links and assert all messages that they display for each link.
4. On the following link: https://demoqa.com/login we will need a data provider with 5 arrays (meaning populate the fields username and password with different values) to insert username, password, click login and verify the message **Invalid Username or Password**.
5. On the same link we need another data provider with 5 arrays (meaning populate the fields with 5 different values) to click New User, populate firstname, lastname, username, password, click register and assert the message **Please Verify Recaptcha to register.**

**Note**: Upload the homework as a zip file (Exported Intellij project: File -> Export -> Project to zip file)

1. **Project Setup and Naming**:
   * Create a new project in IntelliJ IDEA with the name “Selenium-Homework-FirstNameLastName”.
   * Make sure to choose the appropriate language (Java, Kotlin, Groovy, or Python) and build system (Maven or Gradle) based on your preference.
2. **Page Object Model (POM)**:
   * The Page Object Model (POM) is a design pattern that helps organize your Selenium tests by creating separate classes for each web page.
   * We’ll create separate classes for the login page and the links page.
3. **BaseTest Class**:
   * Create a BaseTest class that initializes the WebDriver and sets up common methods (e.g., opening the browser, closing the browser).
   * This class will be extended by other test classes.
4. **DataProvider for Login Page**:
   * Create a data provider with 5 arrays containing different username-password combinations.
   * Use these combinations to log in and verify the “Invalid Username or Password” message on the login page.
5. **DataProvider for New User Registration**:
   * Create another data provider with 5 arrays containing different values for firstname, lastname, username, and password.
   * Use these values to register a new user and assert the “Please Verify Recaptcha to register” message.
6. **Assertions for Links Page**:
   * Navigate to the link: https://demoqa.com/links.
   * Click all links on the page and assert the messages displayed for each link.
7. import org.openqa.selenium.WebDriver;
8. import org.openqa.selenium.chrome.ChromeDriver;
9. public class BaseTest {
10. protected WebDriver driver;
11. public void setUp() {
12. System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");
13. driver = new ChromeDriver();
14. driver.manage().window().maximize();
15. }
16. public void tearDown() {
17. driver.quit();
18. }
19. }

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

public class LoginPage {

private WebDriver driver;

public LoginPage(WebDriver driver) {

this.driver = driver;

}

public void login(String username, String password) {

driver.findElement(By.id("username")).sendKeys(username);

driver.findElement(By.id("password")).sendKeys(password);

driver.findElement(By.id("login")).click();

}

public String getErrorMessage() {

return driver.findElement(By.id("error")).getText();

}

}

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import java.util.List;

public class LinksPage {

private WebDriver driver;

public LinksPage(WebDriver driver) {

this.driver = driver;

}

public void clickAllLinksAndAssertMessages() {

List<WebElement> links = driver.findElements(By.tagName("a"));

for (WebElement link : links) {

link.click();

String message = driver.findElement(By.id("message")).getText();

System.out.println("Link: " + link.getText() + " - Message: " + message);

}

}

}

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.Test;

public class LoginTest extends BaseTest {

private LoginPage loginPage;

@BeforeMethod

public void setUp() {

super.setUp();

loginPage = new LoginPage(driver);

}

@Test(dataProvider = "loginData", dataProviderClass = TestData.class)

public void testLogin(String username, String password) {

loginPage.login(username, password);

String errorMessage = loginPage.getErrorMessage();

// Assert the error message

}

}

// Similar test class for New User registration

import org.testng.annotations.DataProvider;

public class TestData {

@DataProvider(name = "loginData")

public static Object[][] loginData() {

return new Object[][]{

{"user1", "password1"},

{"user2", "password2"},

// Add more test data

};

}

// Similar data provider for New User registration

}