Page Object Model (POM)

Berikut adalah contoh penerapan POM secara praktis beserta struktur file dan skrip Selenium:

Struktur File:

1. Folder "pages": Berisi file-file kelas halaman yang mewakili halaman-halaman dalam aplikasi.
2. Folder "tests": Berisi file-file skrip pengujian yang menggunakan kelas-kelas halaman.

Contoh Struktur File:

```
- project
  - pages
    - LoginPage.java
    - HomePage.java
  - tests
    - LoginTest.java
    - HomeTest.java
  - utils
    - DriverUtils.java
```

Contoh File LoginPage.java (Kelas Halaman Login):

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;

public class LoginPage {
    private WebDriver driver;
    private By usernameField = By.id("username");
    private By passwordField = By.id("password");
    private By loginButton = By.cssSelector("button[type='submit']");

    public LoginPage(WebDriver driver) {
        this.driver = driver;
    }

    public void enterUsername(String username) {
        WebElement usernameElement = driver.findElement(usernameField);
        usernameElement.sendKeys(username);
    }

    public void enterPassword(String password) {
        WebElement passwordElement = driver.findElement(passwordField);
        passwordElement.sendKeys(password);
    }

    public void clickLoginButton() {
        WebElement loginButtonElement = driver.findElement(loginButton);
        loginButtonElement.click();
```

```
    }
}
```

Contoh File LoginTest.java (Skrip Pengujian Login):

```
import org.junit.Assert;
import org.junit.Test;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class LoginTest {
    @Test
    public void testLogin() {
        // Inisialisasi WebDriver
        System.setProperty("webdriver.chrome.driver",
"path/to/chromedriver");
        WebDriver driver = new ChromeDriver();

        // Buka halaman login
        driver.get("https://example.com/login");

        // Membuat objek kelas halaman login
        LoginPage loginPage = new LoginPage(driver);

        // Melakukan interaksi dengan elemen-elemen halaman login
        loginPage.enterUsername("username");
        loginPage.enterPassword("password");
        loginPage.clickLoginButton();

        // Verifikasi hasil login
        String actualTitle = driver.getTitle();
        String expectedTitle = "Dashboard";
        Assert.assertEquals(expectedTitle, actualTitle);

        // Menutup browser
        driver.quit();
    }
}
```

Dalam contoh di atas, terdapat struktur file yang terorganisir dengan folder "pages" untuk kelas-kelas halaman dan folder "tests" untuk skrip pengujian. Setiap kelas halaman (misalnya LoginPage.java) memiliki elemen-elemen UI yang direpresentasikan oleh objek By, dan fungsi-fungsi untuk melakukan interaksi dengan elemen-elemen tersebut.

Dalam skrip pengujian (misalnya LoginTest.java), WebDriver diinisialisasi, halaman login dibuka, dan objek kelas halaman Login dibuat. Kemudian, fungsi-fungsi dalam kelas halaman digunakan untuk melakukan interaksi dengan elemen-elemen UI. Setelah itu, hasil pengujian diverifikasi menggunakan assersi (Assert), dan browser ditutup.

Dengan menggunakan POM, pengujian UI dengan Selenium menjadi lebih terstruktur, mudah dipelihara, dan dapat digunakan kembali. Anda dapat membuat kelas halaman untuk setiap halaman dalam aplikasi, dan mengisolasi logika pengujian dari implementasi UI. Ini membantu meningkatkan keterbacaan skrip pengujian karena logika pengujian terpisah dari implementasi UI. Jika ada perubahan pada halaman, Anda hanya perlu memperbarui kelas halaman terkait, tidak perlu mengubah semua skrip pengujian.

Selain itu, dengan menggunakan POM, Anda dapat menghindari duplikasi kode. Misalnya, jika ada elemen yang sama digunakan di beberapa halaman, Anda dapat membuat kelas halaman yang terpisah untuk elemen tersebut dan menggunakannya secara bersamaan dalam skrip pengujian yang berbeda.

Selain kelas halaman, dalam contoh struktur file yang diberikan, terdapat juga folder "utils" yang berisi file "DriverUtils.java". File tersebut berisi fungsi-fungsi utilitas yang dapat digunakan di berbagai skrip pengujian. Misalnya, fungsi untuk menginisialisasi WebDriver atau fungsi untuk menutup browser setelah pengujian selesai. Ini membantu dalam memisahkan logika pengujian dan logika pengelolaan WebDriver, sehingga memudahkan pemeliharaan dan pengelolaan konfigurasi WebDriver secara terpusat.

Selain itu, Anda juga dapat menggunakan pendekatan lain seperti menggunakan kerangka kerja pengujian yang sudah memuat POM atau menggunakan dependensi manajemen proyek seperti Maven atau Gradle untuk mengatur dependensi dan struktur proyek secara lebih baik.

Dengan menerapkan POM, Anda dapat mengorganisir skrip pengujian UI/UX Anda dengan lebih terstruktur, meningkatkan keterbacaan, pemeliharaan, dan reusabilitas, serta mengurangi duplikasi kode. Hal ini akan mempermudah pekerjaan Anda sebagai UI/UX tester dengan Selenium dan meningkatkan efisiensi serta kualitas pengujian yang Anda lakukan.