JIKA (Jurnal Informatika) Universitas Muhammadiyah Tangerang *P ISSN : 2549-0710*
Tangerang, Januari 2024, Vol 8 No.1, pp 33- 42 *E ISSN : 2722-2713*

OPTIMALISASI PROSES PEMBAHARUAN LOCATOR PADA KODE OTOMATIS SELENIUM DENGAN MENGGUNAKAN PAGE OBJECT MODEL

Mohamad Dhicy Ramdhani[1)], Arief Setyanto[2)], Dhani Ariatmanto[3)]
[1,2,3] Magister Teknik Informatika, AMIKOM Yogyakarta
Jl. Ring Road, Condong Catur, Sleman Yogyakarta
Co Responden Email: m.dhicy.ramdhani@students.amikom.ac.id

Article history
Received 23 Sep 2023
Revised 14 Oct 2023
Accepted 03 Dec 2023
Available online 27 Jan 2024

Keywords
Automation testing,
Locator,
Page object modelling,
Object repository,
Selenium webdriver

Abstract

Application testing is an important process in software development. One way to speed up the testing process is to use automated testing techniques using Selenium webdriver. This technique uses automatic testing scripts that are organized in a test case. Test cases consist of some test steps which has locators. Locators are queries to access elements on the website that represent objects accessed by users such as buttons, text input and others. Same locator can be used in different test cases. In previous research, web-based automation testing was proven to decrease test time, which was 30 seconds faster than the manual testing process. However, the research didn't discuss how to update the locator effectively when there is a change in the structure of the application system. This research aims to optimize the process of updating the automated code by processing locator data using page object modeling (POM) based on login page. Each locator is put in one place called the repository so that the locator update process is done centrally on the repository without having to search and open test cases one by one. The results showed efficiency of 14.28% faster when compared to the update process without the POM method.

Riwayat
Diterima *23 Sep 2023*
Revisi *14 Okt 2023*
Disetujui *03 Des 2023*
Terbit *27 Jan 2024*

Kata Kunci
Pengujian otomatis,
Locator,
Page object modelling,
Obyek repositori,
Selenium webdriver

Abstrak

Pengujian aplikasi adalah proses penting pada pengembangan perangkat lunak. Salah satu cara mempercepat pengujian aplikasi dengan menggunakan teknik pengujian otomatis menggunakan selenium webdriver. Teknik ini menggunakan kode otomatis yang disusun di dalam sebuat test case. Pada test case terdapat langkah pengujian yang terdiri dari banyak locator. Locator merupakan query untuk mengakses element pada website yang merepresentasikan objek yang diakses oleh pengguna seperti tombol, input teks dan lainnya. Locator yang sama bisa digunakan pada test case yang berbeda Pada penelitian sebelumnya, automatisasi pengujian perangkat lunak berbasis website terbukti meningkatkan waktu uji yakni 30 detik lebih cepat dibandingkan proses pengujian secara manual. Namun pada penelitian tersebut tidak dibahas bagaimana proses pembaharuan locator yang efektif ketika terjadi perubahan struktur sistem aplikasi. Penelitian ini bertujuan untuk mengoptimalisasi proses pembaharuan kode automatisasi dengan mengolah data locator menggunakan teknik page object modelling (POM) pada halaman login website. Setiap locator ditempatkan pada satu tempat yang disebut dengan repositori sehingga proses pembaharuan locator dilakukan secara sentralisasi pada repositori tanpa harus mencari dan membuka test case satu per satu.Hasil penelitian menunjukkan efisiensi sebesar 14.28 % lebih cepat jika dibandingkan dengan proses pembaharuan tanpa metode POM**.**

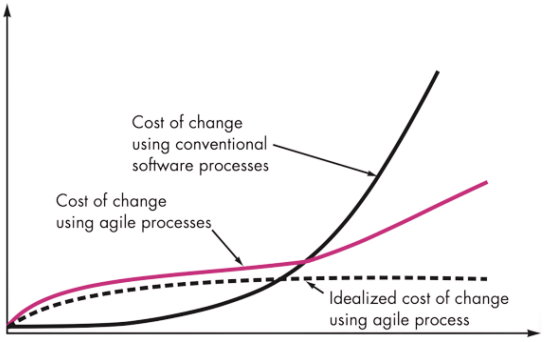
I. PENDAHULUAN

Proses testing merupakan salah satu proses pengujian aplikasi pada pengembangan perangkat lunak yang akan menimbulkan biaya dikarenakan oleh kesalahan sistem (Garousi et al., 2021). Selain itu proses testing dikatakan sebagai aktivitas eksperimental pada sebuah development untuk mencari error dalam sebuah software. Lebih lanjut Min et al., (2020) menyatakan bahwa proses testing digunakan untuk melakukan pengecekan

antara requirement dengan sistem yang diimplementasi. Proses ini juga digunakan untuk mengukur kualitas dari sebuah software sebelum di-release ke konsumen dan sebagai cara menemukan anomali dari software secara dini. Proses testing mengambil peranan yang sangat penting dalam sebuah proses development aplikasi dalam sebuah sistem dengan porsi 30% - 60% dari sebuah project. Kompleksitas software dan kecepatan pengembangan perangkat lunak membuat proses testing lebih lama, sedangkan di sisi lain seluruh tim pengembang ingin memastikan kualitas tetap terjaga namun mengedepankan efisiensi dari sisi pengujian dan perbaikan error (Pombo and Caio, 2021).

Software testing terdiri atas manual testing dan automation testing. Manual testing adalah pengujian dengan melakukan pengecekan terhadap dokumen spesifikasi project tanpa melakukan konversi ke dalam kode otomatisasi. Sedangkan automation testing adalah pengujian dengan menggunakan perangkat lunak yang bertujuan membuat proses testing lebih efisien dari sisi biaya dan waktu(Banjarnahor dan Lucia, 2022). Tim penguji aplikasi melakukan efisiensi proses testing dengan melakukan konversi pengujian manual ke otomatis dengan menggunakan perangkat lunak pengujian otomatis (Garousi et al., 2021). Automation testing bukan hanya berbicara tentang bagaimana sebuah kode otomatis dijalankan, namun lebih menitikberatkan kepada sebuah design yang dirancang seperti manajemen test case, locator dan objek repositori.

Testing otomatis biasa digunakan pada metode pengembangan perangkat lunak berbasis agile. Agile merupakan metode pengembangan software yang banyak digunakan saat ini dan bersifat fleksibel dalam menghadapi perubahan. Tim penguji aplikasi harus bisa mengimplementasikan sifat cepat tanggap untuk merespon perubahan yang terjadi ketika proses pengembangan software. Perubahan yang dimaksud antara lain perubahan proses *build* software, perubahan anggota tim, perubahan yang diakibatkan adanya teknologi baru dan perubahan lainnya. Banyaknya proses perubahan menyebabkan proses testing harus berjalan cepat dengan mengimplementasikan testing otomatis sehingga bisa mengikuti perubahan yang cepat.



Gambar 1. Development schedule process

yang lebih datar serta fleksibel. Sehingga mendukung perubahan secara tiba-tiba dalam proses *software development* tanpa adanya biaya yang melonjak sangat cepat dibanding metode konvensional dan waktu yang terdampak akibat adanya perubahan sangat kecil (Pressman, 2010). Hal ini terlihat pada Gambar 1 dengan kurva berwarna merah muda tebal. Sedangkan kurva datar putus-putus tebal menggambarkan biaya ideal yang dikeluarkan pada proses *agile development*. Salah satu cara untuk menekan biaya pengembangan karena perubahan dinamis adalah menggunakan testing otomatis dengan melakukan manajemen kode testing otomatis dengan tepat menggunakan *Page Object Model* sehingga proses testing tetap terjaga efektif meskipun terjadi perubahan dengan frekuensi yang tinggi pada locator.

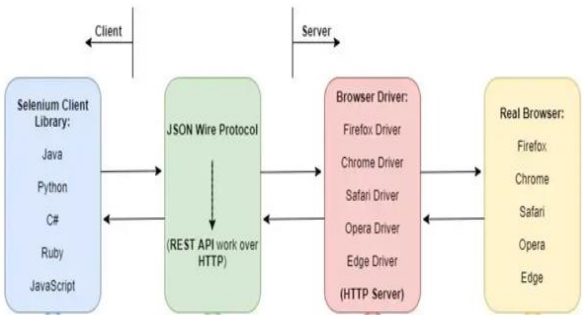
Selenium WebDriver merupakan open-source framework automatic testing yang digunakan untuk melakukan otomatisasi testing pada Web Apps yang digunakan sebagai sarana regression test pada sebuah aplikasi website di dalam proses testing (Marshall et al., 2019). Selain itu selenium juga merupakan library yang terdiri dari *class* dan *functions* untuk mendefinisikan sebuah test. Selenium API pada didesain untuk bekerja di dalam sebuah server sides. Pattrick Lightbody dan Paul Hammer (2004) mencari cara yang lebih mudah dalam melakukan kontrol terhadap test dan bagaimana sebuah library bisa digunakan disemua bahasa pemrograman. Kemudian terbentuklah selenium remote control dengan menggunakan bahasa pemrograman java sebagai client server yang akan membuat komunikasi melalui proxy server supaya terjadi komunikasi data antara

selenium dan browser. Berikut ini merupakan jenis - jenis selenium;

1. Selenium IDE
2. Selenium Remote Control (RC)
3. WebDriver
4. Selenium Grid

Selenium RC dan WebDriver telah disatukan menjadi satu kerangka kerja yakni selenium 2 pada update terakhir. Hal ini dilakukan karena pada versi selenium 2 telah mengimplementasikan WebDriver code sehingga Selenium RC menjadi lebih lengkap fungsinya. Sampai detik ini selenium masih dipakai diberbagai software automation sebagai library utama karena memiliki beberapa kelebihan antara lain;

1. Opensource
   Selenium merupakan tools pengujian bersifat open-source yang artinya bisa dikembangkan secara bebas tanpa menggunakan lisensi
2. Integrasi dengan banyak bahasa pemrograman
   Sifat selenium yang sangat fleksibel dan bekerja di sisi client server sehingga bisa digunakan dengan berbagai macam bahasa pemrograman.
3. Fleksibel
   Kelebihan selenium adalah kemampuan untuk mengelompokkan sebuah test case. Hal ini berguna untuk mempercepat proses jika terjadi perubahan kode, sehingga SQA (*Software Quality Assurance*) bisa melakukan pengujian denga lebih sederhana dan efisien
4. Multiplatform
   Selenium bisa digunakan pada banyak platform dan browser. Dengan kelebihan ini banyak developer atau tester bisa menuliskan kode dengan mudah tanpa harus memikirkan platform yang dipakai.



Gambar 2. Arsitektur selenium webdriver

Berikut ini merupakan penjelasan dari arsitektur selenium webdriver pada gambar 2 :

1. *Selenium Client Library*
   Selenium client library memiliki peran sebagai client yang memfasilitasi berbagai macam bahasa pemrograman. Client library melakukan komunikasi antar bahasa pemrograman sehingga perintah bisa dikirimkan dengan tepat kepana jalur protokol website.
2. *JSON Wire Protocol*
   JSON Wire Protocol berfungsi untuk melakukan komunikasi antara klient dan server agar bisa saling mengerti. Komunikasi yang dilakukan yakni melakukan interaksi antara element dengan kode sehingga menghasilkan kontrol . Prinsip ini mirip seperti yang digunakan pada REST API untuk melakukan komunikasi.
3. *Browser Driver*
   Browser driver berfungsi urnuk menerima request dari klien yang dikirimkan oleh JSON Wire Protocol
4. *Real Browser*
   Browser yang digunakan sebagai sarana automatic testing. Setiap instruksi yang dikirimkan oleh selenium webdriver ke dalam script untuk dijalankan pada proses testing automation

METODE PENELITIAN

Studi yang dilakukan untuk memperoleh model terbaik kerangka kerja dari sebuah kode otomatis tersusun atas beberapa tahapan.

1. ***Software Under Test (SUT)***

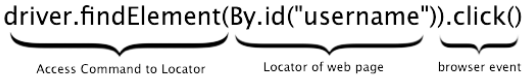
Software Under Test yang digunakan adalah simulasi aplikasi berbasis website yakni https://practicetestautomation.com/practice-test-login/ yang menggunakan objek yang sama pada test case berbeda sehingga implementasi struktur objek yang efisien akan disusun kedalam sebuah stuktur page object model. Halaman website SUT merupakan halaman dengan komponen standar seperti button dan text field. Halaman website yang digunakan menggunakan Angular.js sebagai framework pengembangan sebuah website berbasiskan java script. Namun, tidak tertutup kemungkinan untuk menggunakan jenis kerangka kerja pengembangan website lainnya pada SUT yang digunakan sebagai tempat melakukan penelitian.

2. Bahasa Pemrograman

Penyusunan kode otomatis dapat menggunakan bahasa pemrograman java script, python dan bahasa pemrograman lainnya. Berbagai bahasa pemrograman tersebut dapat mengimplementasikan sebuah struktur yang efisien dengan mengimplementasikan Page Object model dengan baik. POM disajikan dalam bentuk class dan modul dalam pembagian elemen halaman website.

3. Struktur Kode *Selenium Webdriver*

webdriver memiliki struktur kode yang terdiri atas perintah akses ke locator, property locator dan aksi locator. Berikut ini merupakan struktur susunan dasar dari locator webdriver dengan menggunakan selenium ;



Gambar 3. Struktur Kode Selenium

Pada gambar 3 memperlihatkan bahwa terdapat proses koneksi ke sebuah locator menggunakan library selenium webdriver dengan melakukan pencarian locator menggunakan perintah findElement. Locator diakses melalui atribut yang terdiri atas id,class, name dan atribut lainnya. Atribut tersebut berfungsi untuk mengidentifikasikan suatu elemen html sehingga bisa diakses oleh library selenium secara tepat. Setelah dilakukan pengenalan, selanjutnya dikirim sebuah aksi terhadap locator yakni aksi klik. Klik merupakan salah satu aksi yang biasa dilakukan pada sebuah locator website. Aksi lain yang bisa dilakukan antara lain setText() yang berfungsi untuk mengisi sebuah field atau selectByLabel() yang untuk memilih pilihan dari sebuah dropdown list.

4. Penyimpanan Locator

Locator disimpan ke dalam sebuah tempat yang disebut dengan Page object factory. Page object factory berfungsi sebagai penghubung antara halaman web dengan test case yang dibuat. Berdasarkan Gambar 3 terlihat bahwa page object menyambungkan antara halaman page dalam hal ini locator dengan test case yang direpresentasikan sebagai aksi klik pada suatu element

Proses mengambil atribut locator menggunakan teknik inspect element yang terdapat pada setiap browser. Penelitian ini menggunakan browser chrome sebagai perambah inti. Berikut ini merupakan cara untuk mengambil atribut locator dari sebuah SUT.

1) buka perambah google chrome,
2) masukkan alamat website,
3) klik kanan pada element yang tampil pada website (contoh : button sign In),
4) cari atribut locator (contoh :id).

Gambar 4. Atribut id Locator

Pada gambar 4 terlihat bahwa terdapat atribut id pada locator element input teks username. atribut tersebut digunakan untuk melakukan interaksi sebuah antara website dengan test case yang kemudian disimpan pada tempat yang disebut repositori.

5. Proses Pembentukan dan Pengujian

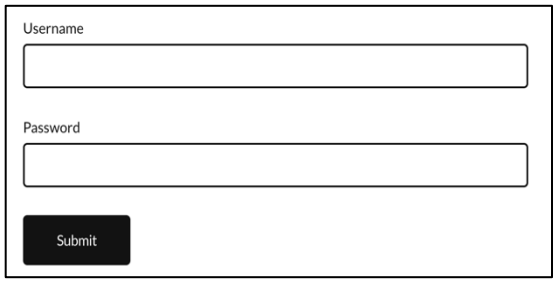
Pengujian dalam penelitian ini berfokus pada proses pengujian terintegrasi. Pengujian ini merupakan salah satu jenis pengujian setelah proses unit testing selesai dijalankan dengan tujuan untuk menguji sistem yang telah terintegrasi dari beberapa modul serta tampilan antar muka secara keseluruhan (Riasat et al., 2021).

Jenis TC (Test Case) eksperimen yang digunakan berjumlah 7 test case. Test case yang digunakan terdiri dari 1 TC positif dan 6 TC negatif. Test case tersebut dibentuk dari halaman login pada tampilan website https://practicetestautomation.com/practice-test-login/. Tabel 1 menjelaskan bagaimana test case terbentuk berdasarkan data bernilai benar yang divisualisasikan dengan logo centang (✓) dan data bernilai salah dengan logo kali (✕). Sedangkan logo minus (-) menunjukkan data tidak diisi. Status positif menjelaskan bahwa modul login berhasil melakukan autorisasi user, sedangkan status negative menunjukkan ada data yang tidak valid sehingga fungsi login mengalami kegagalan autorisasi.

Tabel 1 Kombinasi test case berdasarkan input data

Test Case	Username	Password	status
Tc_1	✓	✓	positif
Tc_2	✓	✗	negative
Tc_3	✗	✓	negative
Tc_4	✗	✗	negative
Tc_5	—	—	negative
Tc_6	✓	—	negative
Tc_7	—	✓	negative

Gambar 5 adalah tampilan dari modul login yang terdiri dari 3 komponen yakni 2 text field (username dan password) serta sebuah button (submit).



Gambar 5. Modul Login

Berikut merupakan test case dasar berdasarkan modul login pada gambar 5 sehingga menghasilkan kombinasi TC seperti pada tabel 1;

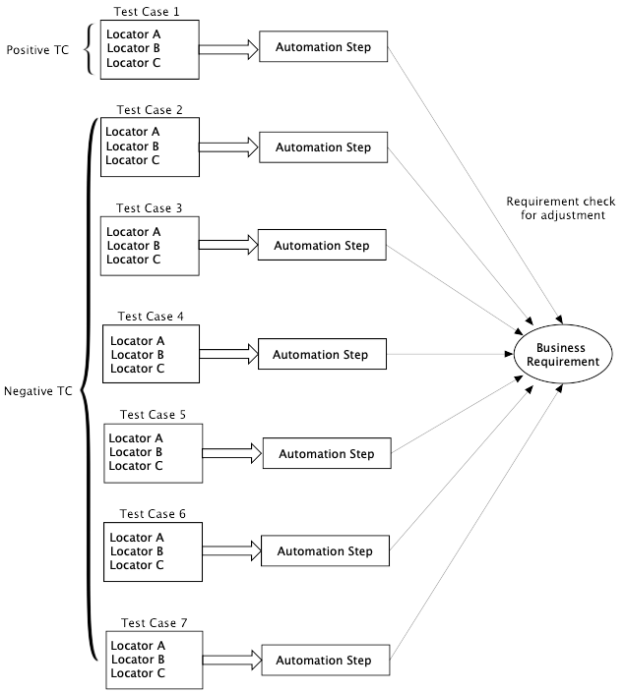
1. Buka halaman https://practicetestautomation.com/practice-test-login/
2. Masukkan data username pada **field username**
3. Masukkan data password pada **field password**
4. Klik **tombol submit**
5. Verifikasi message yang tampil (positif dan negative)

Terdapat tulisan tebal yakni field username, field password dan tombol submit. Tulisan tersebut menggambarkan Locator pada sebuah halaman website. Jika diimplementasikan ke dalam kode automatisasi maka akan menghasilkan baris kode seperti yang terlihat pada gambar 6 yang mewakili 1 test case. Angka di dalam lingkaran merepresentasikan nomor urut test case dasar. Kode automatisasi harus diduplikasi sebanyak 7 kali untuk menghasilkan 7 test case. Locator terdapat didalam deklarasi setelah findElement seperti `by.id("username")` yang terlihat pada gambar 6.

```
Webdriver driver = new ChromeDriver();
System.setProperty("webdriver.chrome.driver", "Path of the chrome driver");

driver.get("https://practicetestautomation.com/practice-test-login/");          1

WebElement username=driver.findElement(By.id("username"));                      2
username.sendKeys("student");

WebElement password=driver.findElement(By.id("password"));                      3
password.sendKeys("Password1234");

WebElement submit=driver.findElement(By.name("submit"));                        4
submit.click();

String messageText = "${message}";
    if ( driver.getPageSource().contains("${message}")){
      System.out.println("Text: " + messageText + " is present. ");             5
    } else {
      System.out.println("Text: " + messageText + " is not present. ");
    }
```

Gambar 6. Konversi kode automatisasi

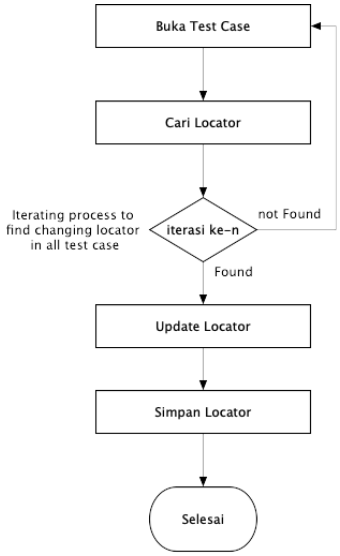
Gambar 7 menjelaskan implementasi antara TC 1 sampai TC 7 dimana terdapat nama locator beserta nilai/attribut. Locator tersebut digunakan sebagai penghubung antara halaman website dengan aksi yang akan dilakukan. Aksi pada gambar disebut dengan automation step. Automation step menjelaskan bagaimana langkah pengujian yang dilakukan terhadap sistem. Langkah tersebut disesuaikan dengan business requirement sehingga antara requirement dengan proses testing memiliki tujuan yang sama.



Gambar 7. Test Case Strategy

6. Proses pembaharuan

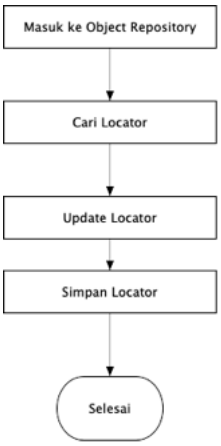
Proses pembaharuan locator menjadi salah satu proses yang sangat krusial ketika sebuah sistem mengalami peningkatan kompleksitas atau perubahan yang terjadi terjadi dengan frekuensi tinggi dalam waktu yang berdekatan. Gambar 8 dan gambar 9 memperlihatkan flowchart dari proses pembaharuan locator dengan menggunakan POM dan tanpa menggunakan POM.



Gambar 8. Updating Process Without POM

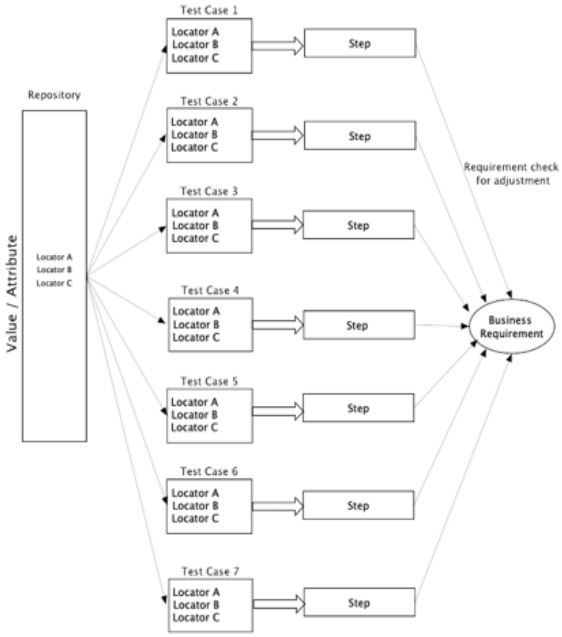
Proses pembaharuan locator tanpa POM terlihat pada gambar 8. Gambar tersebut menjelaskan proses perubahan locator yang dilakukan dengan mencari terlebih dahulu locator yang ingin diubah pada seluruh test case yang ada yakni berjumlah 7 TC. Pencarian locator dilakukan di semua test case untuk memastikan perubahan terjadi pada semua test case yang menggunakan locator terbaru.

Sedangkan pada gambar 9 terlihat bahwa susunan langkah mengimplementasi-an POM di dalam proses pembaharuan locator tidak perlu mencari ke dalam seluruh test case melainkan hanya mencari locator pada repositori. Locator yang telah dirubah nilainya di repositori akan secara otomatis ter-*update* pada 7 test case yakni locator username, password dan submit. Proses membuka satu per satu test case hanya untuk merubah locator tidak perlu dilakukan karena locator telah diubah di dalam repositori. Proses ini memiliki efisiensi yang baik dengan mengabaikan proses iterasi untuk membuka seluruh test case.



Gambar 9. Updating Process with POM

Visualisasi locator POM terlihat pada gambar 10, dimana terdapat sebuah tempat yang disebut dengan repositori. Repositori ini berfungsi untuk menyimpan nilai atau atribut locator kemudian dipanggil ke dalam test case sesuai dengan nilai atau attribut dari sebuah locator. Teknik penyimpanan ini akan menempatkan nilai locator yang unik sehingga tidak terjadi duplikasi. berbeda halnya dengan gambar 6, seluruh locator dan nilainya disimpan pada test case sehingga terjadi duplikasi locator yang sama pada test case yang berbeda. Duplikasi locator menyebabkan proses perubahan menjadi lebih rumit ketika test case mengalami perubahan atau penambahan.



Gambar 10. Test Case Strategy with POM

Modul login pada penelitian ini dapat dikonversi ke dalam kode otomatisasi POM dengan langkah sebagai berikut;

1. Buat class objek sebagai repositori locator dan tempat dari step automation.
2. Buat class utama yang berfungsi untuk menjalankan kode automatisasi dengan memanggil locator dan step dari object respository.
3. Jalankan class utama.
4. Ketika terjadi update maka ubah locator yang terdapat di dalam class repositori kemudian jalankan class utama.

Pembuatan class objek sebagai repositori berfungsi untuk sentralisasi tempat locator dan step sehingga ketika proses pembaharuan bisa dilakukan pada class objek. Gambar 11 menggambarkan class *loginActivity* yang terdiri dari locator dan step automation. Seluruh detail dari locator disimpan pada class *loginActivity*. Dalam konsep POM class ini disebut dengan repositori.

```
public class loginActivity {
    WebDriver driver;
    By username = By.id("username");
    By password = By.id("password");          ──► Locator
    By submit = By.id("submit");
    By message = By.id("error");

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

    //set username
    public void setUserName(String strUserName){
        driver.findElement(username).sendKeys(strUserName);
    }

    //set password
    public void setPassword(String strPassword){
        driver.findElement(password).sendKeys(strPassword);   ──► Step
    }

    //Click on login button
    public void clickLogin(){
            driver.findElement(submit).click();
    }
    //Verify Text
    public void checkText(String strMessage){
            if ( driver.getPageSource().contains(strMessage)){
        System.out.println("Text: " + strMessage + " is present. ");
      } else {
        System.out.println("Text: " + strMessage + " is not present. ");
      }
    }
}
```

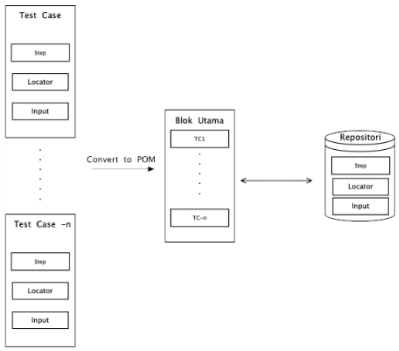
Gambar 11. Class repositori

Setelah membuat class repositori, kemudian class utama dibentuk. Kemudian class utama memanggil class repositori dan setting lainnya sebelum script automation dijalankan. Pada gambar 12 terlihat script automation disusun secara lebih terstruktur dengan urutan langkah yang lebih jelas. Perbedaan yang jelas terlihat adalah bagaimana class utama terlihat lebih *maintainable* ketika locator mengalami perubahan. Hal ini dikarenakan pada class utama seluruh test case berada di dalam satu file. Sedangkan pada kode dasar (gambar 6) hanya mewakili 1 test case untuk satu file.

```
public class mainActivityLogin {
String driverPath = "C:\\geckodriver.exe";
WebDriver driver;
loginActivity objLoginActivity;

@BeforeTest
public void setup() {
System.setProperty("webdriver.chrome.driver", "C:\\research\\chromedriver.exe");
driver = new ChromeDriver();
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
driver.get("https://practicetestautomation.com/practice-test-login/");
}
@Test(priority = 1)
public void TC1() {
objLoginActivity = new loginActivity(driver);
objLoginActivity.setUserName("Student");
objLoginActivity.setPassword("Password1234");
objLoginActivity.clickLogin();
objLoginActivity.checkText("Invalid")
}                                                    ──► Kode Utama
@Test(priority = 2)
public void TC2() {
objLoginActivity = new loginActivity(driver);
objLoginActivity.setUserName("wrongUsername");
objLoginActivity.setPassword("Password1234");
objLoginActivity.clickLogin();
objLoginActivity.checkText("Invalid")
}
@Test(priority = 3)
public void TC3() {
objLoginActivity = new loginActivity(driver);
objLoginActivity.setUserName("Student");
objLoginActivity.setPassword("wrongUsername");
objLoginActivity.clickLogin();
objLoginActivity.checkText("Invalid")
}
}
```

Gambar 12. Class utama hasil konversi POM

Perbandingan antara sebelum menerapkan POM dan setelah menerapkan POM bisa terlihat pada gambar 13. Class test case tanpa POM memiliki jumlah yang sesuai dengan jumlah kombinasi test case. Hal ini terjadi karena di dalam test case terdapat input data yang berbeda sehingga mengharuskan test case dibuat berulang untuk input data yang yang lainnya. Sedangkan metode POM hanya memiliki satu class utama test case. Class ini memiliki seluruh kombinasi test case. Perbedaan yang paling signifikan adalah pada akses locator dan step yang berada pada satu tempat sehingga ketika terjadi proses pembaharuan locator dan perubahan step menjadi lebih pendek dan ringkas.



Gambar 13 Simplifikasi test case ke dalam bentuk POM

HASIL DAN PEMBAHASAN

Penelitian ini menghasilkan beberapa parameter yang dijadikan acuan dalam melakukan efisiensi penyusunan locator dalam sebuah test case berdasarkan gambar 6 maka dapat dihasilkan seperti tabel 2 .

Tabel 2. Frekuensi Locator pada Test Case

Locator	TC	Frekuensi
A(username)	Tc_1 Tc_2,Tc_3 Tc_4 Tc_5,Tc_6 Tc_7	7
B(password)	Tc_1 Tc_2,Tc_3 Tc_4 Tc_5,Tc_6 Tc_7	7
C(submit)	Tc_1 Tc_2,Tc_3 Tc_4 Tc_5,Tc_6 Tc_7	7
Total frekuensi		21
Total locator		3

Tabel 2 menjelaskan bagaimana frekuensi sebuah locator terdapat di dalam sebuah test case. Total dari frekuensi locator menunjukkan jumlah proses yang harus dilakukan dalam melakukan pembaharuan locator ketika terjadi perubahan. Mengacu pada gambar 8, dengan asumsi semua locator mengalami perubahan, maka total proses yang harus dilakukan adalah sebanyak 21 kali proses membuka test case dan melakukan perubahan pada locator jika terdapat penyesuaian. Ketika POM diimplementasikan ke dalam test case, maka variabel yang hitung hanya total dari locator yang ada. Sehingga total proses yang dilakukan hanya sebanyak 3 kali proses perubahan dimana nilai 3 menunjukkan total dari locator yang ada. Hal ini berbeda dengan perhitungan tanpa menggunakan metode POM yang menggunakan variabel frekuensi kemunculan locator pada test case. Dari hasil analisis dengan menggunakan metode POM berdasarkan variabel locator dan frekuensi menghasilkan sebuah nilai efisiensi seperti yang ditunjukkan pada persamaan 1.

$$Efisiensi = \frac{\sum Locator}{\sum Frekuensi} \times 100\% \quad (1)$$

Keterangan :

Efisiensi = nilai efisiensi berupa peningkatan proses

$\sum Locator$ = jumlah locator yang digunakan

$\sum Frekuensi$ = total frekuensi locator pada test case

$$Efisiensi = \frac{3}{21} \times 100\% = 14.28\ \%$$

Berdasarkan persamaan 1 maka didapatkan nilai efisiensi sebesar 14.28 %. Hal ini menunjukkan bahwa penggunaan metode Page Object Model dalam melakukan proses pembaharuan memiliki jumlah langkah yang lebih sedikit dibandingkan proses pembaharuan tanpa metode POM. Pembaharuan locator pada repositori akan secara otomatis merubah seluruh locator yang digunakan di dalam test case tanpa membuka satu per satu test case. Sedangkan tanpa menggunakan metode POM, locator disimpan pada setiap test case sehingga untuk melakukan pembaharuan harus membuka test case untuk mengetahui apakah ada locator yang harus dirubah atau tidak.

Nilai efisiensi berbanding terbalik dengan jumlah locator. jika locator semakin banyak dan total frekuensi tetap, maka nilai efisiensi akan semakin meningkat. Hal ini menggambarkan bahwa jumlah proses perubahan terhadap locator akan semakin sedikit meskipun jumlah locator terus bertambah.

Pada penelitian sebelumnya (Min et al., 202) menjelaskan bagaimana proses automation testing lebih cepat dibandingkan dengan manual testing. Proses tersebut mengolah total test case dengan total test case yang berhasil dikonversi ke dalam bentuk automation script sehingga menghasilkan nilai percent automable atau tingkat persentasi testing otomatis. Sedangkan pada penelitian ini berfokus bagaimana manajemen locator yang digunakan di dalam test case berjalan dengan efektif dengan menggunakan metode POM sehingga mampu membuat proses pembaharuan locator berjalan lebih cepat . Data yang diolah adalah total dari locator dan total frekuensi locator di dalam test case. Hasil dari penelitian ini saling menguatkan dimana pada penelitian (Min et al., 2020) bertujuan mempercepat proses testing dengan melakukan konversi ke dalam bentuk automation testing sedangkan pada penelitian ini bertujuan untuk mempercepat proses pembaharuan ketika terjadi perubahan requirement sehingga proses automation testing tetap terjaga kecepatannya.

KESIMPULAN

Dari hasil penelitian dapat disimpulkan bahwa penggunaan metode POM meningkatkan efektifitas dalam proses pembaharuan locator sebesar 14.28 %. Nilai tersebut merepresentasikan pengurangan usaha dalam melakukan pembaharuan locator. Proses pembaharuan locator merupakan salah satu faktor bagaimana kode testing otomatis bisa tetap berjalan dengan efektif pada sistem yang mengalami perubahan pada locator. Proses pembaharuan yang lama dapat mengakibatkan proses testing menjadi tertunda. Penggunaan repositori sebagai tempat attribut locator dan step secara signifikan mempercepat proses pembaharuan dibandingkan ketika attribut locator didefinisikan langsung di dalam test case. Ketika terjadi perubahan dari locator, maka nilai attribut pada repositori dirubah sehingga nilai tersebut akan secara otomatis meng-*update* locator di seluruh test case hanya dalam 1 kali perubahan meskipun locator yang sama digunakan pada test case berbeda dalam jumlah yang besar.

Proses mapping antara nama locator dengan atribut locator yang disimpan di dalam repositori membuat locator menjadi fleksibel dalam merespon perubahan. Proses manajemen locator dimulai dengan melakukan definisi nilai locator menggunakan inspeksi element pada elemen website. Dari hasil inspeksi dapat diketahui nilai/attribut dari sebuah elemen. Data tersebut disimpan didalam repositori. Penggunaan repositori tidak terbatas hanya pada satu bahasa pemrograman saja. Namun secara umum dapat diimplementasikan pada semua bahasa pemrograman dengan menggunakan konsep POM yang biasanya terbagi ke dalam sebuah class atau modul. Penelitian selanjutnya diharapkan bisa menganalisis bagaimana jenis attribut yang dimiliki oleh elemen dari website berpengaruh terhadap kecepatan interaksi antara locator dengan website sehingga proses testing berjalan lebih cepat. Selain itu bagaimana merancang manajemen data masukan pada test case bisa diteliti lebih mendalam sehingga bisa menyempurnakan konsep POM serta membuat struktur input data ke dalam test case menjadi lebih baik.

REFERENSI

Garousi, V., et al. (2021). Model-based testing in practice: An experience report from web applications domain. *Elsevier Science Direct The Journal of System and Software 180 (2021) 111032*. doi: https://doi.org/10.1016/j.jss.2021.111032

Pombo N., and C. Martins. (2021). Test Driven Development in Action: Case Study of a Cross-Platform Web Application. *IEEE EUROCON 2021 - 19th International Conference on Smart Technologies, Lviv, Ukraine, 2021, pp. 352-356.*

Banjarnahor M.T., and L. S. Istiyowati. (2022). Smoke Automation and Regression Testing on a peer-to-peer lending Website with the Data-Driven Testing Method. *Jurnal Rekayasa Sistem dan Teknologi Informasi*. 6(4). 684-691.

Joe Lian Min, A. Istiqomah, dan A. Rahmani. (2020). Evaluasi Penggunaan Manual dan Automated Software Testing pada Pelaksanaan End-to-End Testing. *Jurnal Teknologi Terapan*. 6(1)

Riasat, H., Nazir, Z., and Zubair, S. (2021). Critical Analysis of software testing techniques and automation testing tools, *International Journal of Scientific & Engineering Research.*12 (2). ISSN : 2229-5518

Anand, R, and M. Arulprakash. (2018). Business Driven Automation Testing Framework. *International Journal of Engineering and Technology*. 7(2.8).

Malik, A., and A. Mehta. (2022). *Automation Testing - A Review*. *International Research Journal of Modernization in Engineering Technology and Science*. 4(6).

Ateşoğulları D., and A. Misra. (2020). Automation Testing Tools: A Comparative View. *International Journal of Computer Science and Information Technologies*. 4(12).

Ramler, R., and C. Klammer. (2019). Enhancing Acceptance Test-Driver Development with Model-Based Test Generation. *IEEE 19th International Conference on Software Quality, Realibility and Security Companion.*

Ouriques R., Krzysztof W., and Tony G. (2023). The Role of Knowledge-based resources in Agile Software Development Context.*The Journal of Systems & Software*. doi:https://doi.org/10.1016/j.jss.2022.111572

Imtiaz J., Zohaib I., and Uzair K. (2020). An Automated Model-based Approach to Repair Test Suite of Evolving Web Applications. *The Journal of System and Software*.doi:https://doi.org/10.1016/j.jss.2020.110841.

Fischbach J., Andreas V., Dominik S., and Andreas W. (2020). SPECMATE: Automated Creation of Test Cases from Acceptance Criteria. *IEEE 13th International Conference on Software Testing, Validation and verification (ICST)*.doi: 10.1109/ICST46399.2020.00040

Biagiola, M., et al. (2019). Web test dependency detection. in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundation of Software Engineering*. doi:https://doi.org/10.1145/3338906.3338948

Presler-Marshall, K., E. Horton, S. Heckman, and K.T. Stolee. (2019). Wait Wait No Tell Me Analyzing Selenium Configuration Effects on Test Flakiness. *2019 IEE/ACM 14th International Workshop on Automation of Software Test (AST).* doi: 10.1109/AST.2019.000-1.

Biagiola, M., et al. (2019). Diversity-Based Web Test Generation. *Proceeding of 27th ACM Joint European Sofware Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'19).* doi: 10.1145/3338906.3338970

Rosa, A. S. dan M Shalahuddin. (2013). *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek*. Bandung: Informatika.*dan Berorientasi Objek*. Bandung: Informatika.

Roger S. Pressman. (2010). *Software Engineering : A Practitioner's Approach (7th ed)*. New York: McGraw-Hill.

J. Sutherland. (2014). *SCRUM: The Art of Doing Twice the Work in Half the Time.* New York: Crown Business.

Cara Kerja, Pengertian, Konsep selenium webdriver. Retrieved September 19, 2023, from browserstack website: https://www.browserstack.com/guide/selenium-framework.