BAB III

Selenium Webdriver

Selenium Webdriver merupakan bagian dari Selenium 3. WebDriver menggerakkan browser secara asli, seperti yang akan dilakukan user, baik secara lokal atau pada mesin jarak jauh menggunakan server Selenium, menandai lompatan maju dalam hal otomatisasi browser. Pada bab ini akan dijelaskan fitur-fitur yang ada pada Selenium webdriver. Fitur yang akan dibahas pada bab ini adalah capabilities yang terdiri dari shared capabilities, chromium capabilities, internet explorer capabilities, dan firefox capabilities.

Fitur pada browser yang terdiri dari browser information, browser navigation, browser alert, browser cookies, browser frames, dan browser windows. Fitur pada WebEkements yang terdiri dari locator, interaction, information, dan select list. Fitur pada remote webdriver, wait dan action api yang terditi dari mouse actions dan keyboard actions

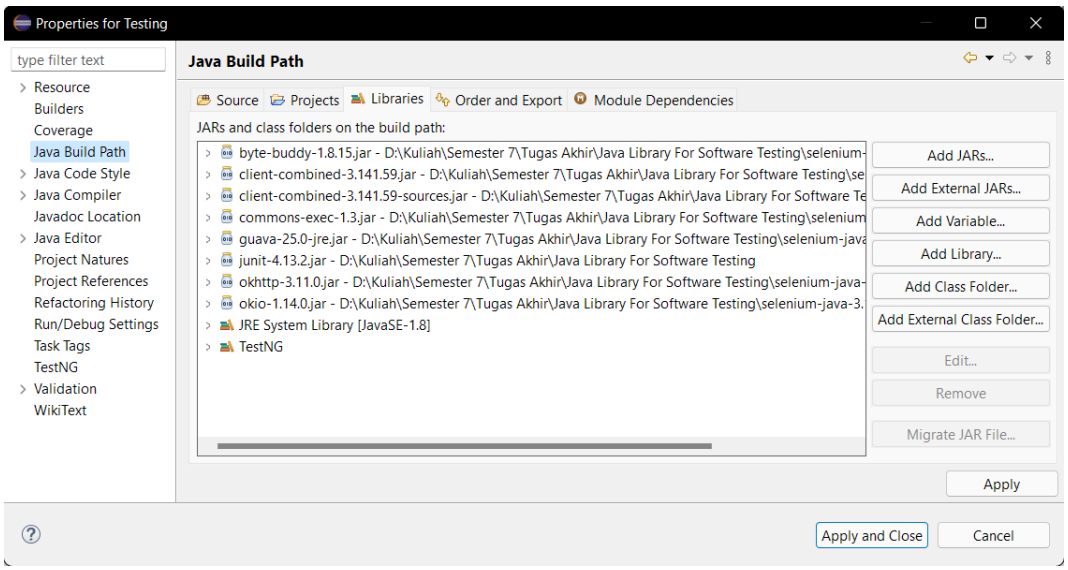
3. 1 Persiapan Pemakaian Selenium Webdriver

WebDriver adalah API dan protokol yang mendefinisikan antarmuka bahasa-netral untuk mengontrol perilaku browser web. Setiap browser didukung oleh implementasi WebDriver tertentu, yang disebut driver. Driver merupakan komponen yang bertanggung jawab untuk menghubungkan ke browser, dan menangani komunikasi menuju dan dari Selenium dan browser.

Pengaturan selenium sangat berbeda dari pengaturan alat komersial lainnya. Untuk menggunakan Selenium dalam proyek otomatisasi Anda, Anda perlu menginstal pustaka binding bahasa untuk bahasa pilihan Anda. Selain itu, Anda akan memerlukan binari WebDriver untuk browser yang ingin Anda otomatisasi dan jalankan pengujiannya. Langkah langkah untuk melakukan instalasi pada Selenium Webdriver akan dijelaskan sebagai berikut.

Pertama tester harus melakukan download JDK java dengan versi sesuai dari operating sistem yang dimiliki oleh tester. Setelah melakukan download jdk

tester melakukan download pada IDE yang digunakan untuk melakukan testing. Testing pada tugas akhir akan menggunakan IDE eclipse. Setelah melakukan download IDE, tester melakukan download Selenium Java Client driver. Kemudian user melakukan download browser driver. Ada beberapa driver yang dapat digunakan pada selenium Webdriver. Driver tersebut antara lain: chromedriver untuk google chrome, geckodriver untuk mozzila firefox, safari driver, dan opera driver. Langkah berikutnya adalah membuat new project pada eclipse. Untuk memasukkan package yang sudah di download, user melakukan klik kanan pada project kemudian memilih properties, dan memilih java build path. Gambar dari contoh package yang dimasukkan diberikan pada gambar 3.1.



Gambar 3.1
Gambar package untuk menggunakan Selenium Webdriver

Gambar 3.1 merupakan gambar contoh package yang diperlukan oleh selenium webdriver. Byte Buddy adalah library pembuatan dan manipulasi kode untuk membuat dan memodifikasi class yang ada pada Java selama runtime aplikasi Java dan tanpa bantuan kompiler. Client combined dan client combined merupakan Selenium Webdriver itu sendiri. Commons-exec, guava, oktthp, dan okio merupakan bagian dari Selenium Java. Junit merupakan package yang digunakan untuk melakukan automated testing pada Java. JRE merupakan Java client driver yang digunakan untuk melakukan automated testing, dan TestNG

adalah package yang digunakan untuk membuat report karena Selenium tidak memiliki fitur untuk membuat report

3. 2 Desired Capabilities

Desired Capabilities adalah seperangkat properti yang digunakan untuk mengkonfigurasi instance driver Selenium WebDriver. Desired Capabilities merupakan komponen dari paket org.openqa.selenium.remote. DesiredCapabilities. Ini membantu Selenium WebDriver untuk mengatur properti untuk instance browser. Jadi kita dapat mengatur properti browser dengan menggunakan kemampuan yang berbeda dari Kemampuan yang Diinginkan. Misalnya, versi browser, nama browser, jalur driver browser di sistem, dll. Kami menggunakan kemampuan yang diinginkan ini sebagai pasangan kunci atau nilai untuk menyetelnya untuk browser. Dengan menggunakan kemampuan yang diinginkan, user dapat mengonfigurasi instance driver seperti ChromeDriver, Firefox Driver, InternetExplorerDriver, SafariDriver.

Setiap kasus pengujian khusus perlu dijalankan di lingkungan yang berbeda (browser, seluler, atau sistem operasi) sesuai dengan tren pasar. Misalnya, tester mungkin perlu menguji aplikasi Web pada dua browser berbeda (Chrome, Safari) yang diinstal pada perangkat seluler yang berjalan pada sistem operasi berbeda (Android, iOS). Di sinilah Kemampuan yang Diinginkan memungkinkan tester untuk menginstruksikan Selenium WebDriver mengenai lingkungan yang akan digunakan saat melakukan pengujian. Menggunakan metode setCapabilities dari kelas DesiredCapabilities, tester dapat menjalankan pengujian paralel pada perangkat, browser, dan sistem operasi yang diinginkan yang tersedia di Cloud Selenium Grid. Ada 4 capabilities secara umum pada Selenium Webdriver yaitu Shared Capabilities, dan capabilities untuk masing-masing browser yaitu google chrome, mozzila firefox, edge dan opera.

3.2.1 Shared Capabilities

Capabilities ini merupakan capabilities yang di share pada semua browser yang ada. Capabilities ini dibutuhkan untuk melakukan testing ketika user

melakukan testing pada browser. Beberapa capabilities yang disupport oleh Selenium antara lain: browserName, browserVersion, pageLoadStrategy, platformName, acceptInsecureCerts, timeouts, unhandledPromptBehavior, setWindowRect, strictFileInteractibility, dan proxy. Shared Capabilities pertama adalah browserName. BrowserName digunakan untuk melakukan set pada session yang dipilih. Berikutnya adalah browserVersion. BrowserVersion ini merupakan capabilities yang bersifat optional karena bisa di set ataupun tidak. BrowserVersion merupakan capabilities yang digunakan untuk set versi browser pada testing session.

Capabilities berikutnya adalah pageLoadStrategy. Capabilities ini merupakan capability yang mengatur kapan testing dilakukan. Secara default, pageLoadStrategy bernilai normal dimana Selenium akan menunggu website untuk melakukan load semua data sebelum melakukan testing. Selain normal, pageLoadStrategy memiliki dua value lagi yang bisa menggantikan normal yaitu eager dan none. Eager berarti testing dilakukan ketika DOM saja yang sudah terload, sedangkan resource lainnya seperti image bisa saja belum terload tetapi testing sudah dijalankan. Value none berarti testing bisa dilakukan langsung walaupun DOM dan resource lainnya masih belum terload.

Capabilities berikutnya adalah platformName. Capabilities ini berguna untuk mendapatkan operating system apa yang digunakan saat testing dijalankan. Capabilities berikutnya adalah acceptInsecureCerts. Capabilities ini berfungsi untuk cek apakah sertifikat yang dimiliki web tersebut sudah valid atau belum. Sertifikat pada web ini disebut dengan TLS certificate. TLS merupakan singkatan dari Transport Layer Security yang berfungsi untuk mengamankan data. Capabilities berikutnya adalah timeouts. Timeouts sendiri dibagi menjadi 3 jenis yaitu Script Timeouts, Page Load Timeouts, dan Implicit Wait Timeouts.

Script Timeouts merupakan timeout yang digunakan untuk memberikan jeda saat melakukan eksekusi script pada browser. Nilai default dari script timeout ini adalah 30000 ms. Page load timeouts merupakan timeout yang digunakan untuk memberikan batas waktu halaman yang diuji terload. Jika page yang diload melebihi batas waktu maka script akan memberikan exception yaitu TimeOut

Exception. Nilai default dari Page load timeouts adalah 300000 ms. Timeout yang ketiga adalah Implicit Wait Timeout. Implicit wait timeout merupakan timeout yang digunakan untuk membatasi waktu yang diperlukan script mencari elemen yang ada. Implicit wait timeout secara default memiliki value 0.

Shared Capabilities berikutnya adalah Unhandled Prompt Behavior. Unhandled Prompt Behavior merupakan capabilities yang dimiliki oleh Selenium Webdriver untuk melakukan set User Prompt Handler. Ada 5 value untuk user prompt handler ini antara lain dismiss, accept, dismiss and notify, accept and notify, dan ignore. Nilai default untuk user prompt handler adalah dismiss and notify. Capabilities berikutnya adalah setWindowsrect. Set Windows Rect merupakan setter dari getWindowsRect. Terdapat 4 value yang bisa digunakan untuk set windows rect yaitu maximized, minimized, normal, dan fullscreen. Capabilities berikutnya adalah strictFileInteractibility. Capabilities ini digunakan pada input type file dan secara default bernilai false.

Capabilities berikutnya adalah proxy. Server proxy bertindak sebagai perantara permintaan antara klien dan server. Proxy server digunakan untuk menangkap traffic jaringan, Menirukan panggilan backend yang dilakukan oleh web dan mengakses situs web yang diperlukan di bawah topologi jaringan yang kompleks atau pembatasan/kebijakan perusahaan yang ketat

3.2.2 Chromium Capabilities

Capabilities chromium merupakan opsi yang dapat digunakan untuk menyesuaikan dan mengonfigurasi sesi EdgeDriver, dan ChromeDriver. Pada EdgeDriver capabilities yang dapat digunakan antara lain args, binary, debuggerAddress, detach, excludeSwitches, extensions, localState, minidumpPath, mobileEmulation, perfLoggingPrefs, prefs, wdpAddress, wdpPassword, wdpUsername, windowsApp, dan windowsType. Capabilities pertama adalah args. Args merupakan Daftar argumen baris perintah yang digunakan saat memulai Microsoft Edge. Capabilities berikutnya adalah binary. Binary merupakan capabilities yang digunakan untuk mendapatkan path Microsoft edge diinstall.

Capability berikutnya adalah debuggerAddress. Capability ini digunakan untuk mendapatkan hostname yang digunakan untuk mengakses website contohnya localhost:3000. Capability yang dimiliki Microsoft edge selanjutnya adalah detach. Detach merupakan capability yang mengatur perhentian webdriver service. Jika detach bernilai false maka Microsoft edge akan tertutup ketika webdriver service telah dishutdown. Jika detach bernilai true maka Microsoft Edge hanya berhenti jika ujung lokal WebDriver menutup sesi. Capabilities berikutnya adalah excludeSwitches. Exclude switches merupakan Daftar sakelar baris perintah Microsoft Edge untuk mengecualikan EdgeDriver itu secara default lewat saat memulai Microsoft Edge.

Capabilities berikutnya adalah extension. Extension merupakan daftar ekstensi yang akan dipasang saat startup. Setiap item dalam daftar harus berupa ekstensi paket yang disandikan dengan base-64. Capabilities berikutnya localState. Localstate merupakan dictionary dengan setiap entri terdiri dari nama preferensi dan nilainya. Preferensi diterapkan ke file Local State di folder data pengguna. Capabilities berikutnya adalah wdpAddress.

Wdp Address merupakan alamat server portal device windows yang disambungkan, dalam bentuk hostname/ip:port, misalnya 127.0.0.1:3000. Capabilities berikutnya adalah wdpPassword. Wdp password merupakan password yang bersifat optional yang digunakan untuk connect menuju WDP Server. Capabilities berikutnya adalah wdpUsername. Wdp username bersifat optional sama seperti wdpPassword. Wdp username digunakan untuk connect menuju WDP server jika wdp server memiliki authentication.

Capabilities berikutnya adalah windowsApp. Capabilities windowsApp merupakan ID model pengguna aplikasi dari paket aplikasi Microsoft Edge yang akan dijalankan. Capabilities ini digunakan saat menyambungkan ke perangkat atau emulator Windows 10X menggunakan Portal Perangkat Windows. Capabilities berikutnya adalah windowTypes. Capabilitiy windowTypes merupakan Daftar jenis window yang ditampilkan dalam daftar window handle. Window handles merupakan identifier unik yang menyimpan data pada windows yang dibuka.

3.2.3 Firefox Capabilities

Firefox capabilities merupakan capabilities yang secara spesifik dimiliki oleh browser firefox. Untuk menggunakan capabilities yang ada dapat menggunakan firefoxOption. FirefoxOptions adalah cara baru untuk menentukan kemampuan untuk browser Firefox dan umumnya harus digunakan dalam preferensi untuk DesiredCapabilities. Cara penggunaan firefox option akan diberikan pada segmen program 3.1.

Segmen Program 3.1 Mendefinisikan Capabilities dengan FirefoxOption

```
1.    FirefoxOptions options = new FirefoxOptions();
2.    options.addPreference("network.proxy.type", 0);
3.    driver = new RemoteWebDriver(options)
```

Segmen program 3.1 merupakan program untuk mendefinisikan capabilities dengan firefoxoption. Capabilities yang digunakan adalah proxy dengan value yaitu 0. Capabilities yang hanya ada pada firefox antara lain profile, log, prefs, dan env. Capability profile biasanya memiliki extension zip dimana capability ini digunakan untuk menginstall extension atau sertifikat custom. Contoh untuk menambahkan capabilities profile akan diberikan pada segmen program 3.2

Segmen Program 3.2 Cara menambah capability profile pada firefoxOption

```
1.    FirefoxProfile profile = new FirefoxProfile();
2.    FirefoxOptions options = new FirefoxOptions();
3.    profile.addExtension(
4.          new File
5.          (“./src/test/resource/extensions/xpath_finder.xpi”)
6.    ) ;
7.    profile.setPreference("browser.shell.checkDefaultBrowser",
8.    true);
9.    profile.setAssumeUntrustedCertificateIssuer(false);
10.   profile.setAcceptUntrustedCertificates(false);
11.   profile.setDeleteAfterUse(true)
12.   options.setProfile(profile);
13.   driver = new RemoteWebDriver(options);
```

Program 3.2 merupakan program untuk menambahkan capability profile pada firefoxOptions. Dapat dilihat pada program 3.2 ada 7 buah properties yang dapat digunakan pada FirefoxProfile. Properties pertama yang ada pada profile

adalah Assume Untrusted Certificate Issuer. Properties ini digunakan untuk menentukan apakah testing tetap dijalankan jika sertifikat ssl yang dimiliki tidak dapat dipercaya atau untrusted. Jika assumeUntrustedCertificateIssuer bernilai true maka walaupun sertifikat ssl bersifat untrusted testing tetap akan dijalankan. Jika bernilai false, maka user akan mendapatkan exception yaitu ssl_error_bad_cert_domain.

Assume untrusted certificate issuer melakukan cek jika sertifikat bersifat untrusted apakah user yang membuat sertifikat tersebut juga untrusted. Jika bersifat true maka dianggap user yang mengirim sertifikat juga bersifat untrusted dan jika bersifat false maka asumsi ini tidak akan dibuat. Properties berikutnya yang dimiliki oleh firefoxProfile adalah AcceptUntrustedCertificates. Jika bernilai false maka testing akan memberikan exception dan testing akan dihentikan. Properties berikutnya yang dimiliki oleh firefox profile adalah AlwaysLoadNoFocusLibrary. Properties ini berguna untuk menentukan apakah library selalu di load untuk menjalankan command tanpa window focus.

Properties always load no focus library memiliki nilai true dan false. Jika bernilai true maka library akan selalu di load tanpa window harus focus. Jika bernilai false maka library tidak akan di load jika window tidak focus. Properties berikutnya EnableNativeEvent. Properties ini digunaakn apakah native event akan dijalankan pada firefox atau tidak. Contoh native event akan diberikan pada segmen program 3.3

Segmen Program 3.3 Pengunaan Native Event pada firefox

```
1.    FirefoxProfile profile = new FirefoxProfile();
2.    FirefoxOptions options = new FirefoxOptions();
3.    profile.setEnableNativeEvents(true);
4.    options.setProfile(profile);
5.    driver = new RemoteWebDriver(options);
6.    WebElement username = driver.findElement(By.id("uname"));
7.    WebElement password = driver.findElement(By.id("password"));
8.    actions.moveToElement(username);
9.    actions.clickAndHold(password);
10.   actions.release().perform();
```

Segmen program 3.3 merupakan program yang diguanakan untuk handle native event pada firefox. Beberapa native event yang ada antara lain

moveToElement, clickAndHold dan release. Untuk dapat melakukan ini properties enable Native event harus bernilai true. Jika enable native event bernilai false maka native event tidak akan bisa dijalankan. Native event moveToElement berguna untuk memindahkan cursor pada elemen yang dituju. Event click and hold berarti melakukan click dan menahan elemen tersebut. Event release berarti melepaskan hold tadi. Properties berikutnya pada firefoxProfile adalah port. Port berguna untuk menentukan pada port berapa webdriver connect pada firefox profile. Keenam properties ini memiliki getter dan setter. Properties terakhir pada firefox profile adalah profileDirectory. Profile directory tidak memiliki setter dan hanya memiliki getter. Profile directory berguna untuk mendapatkan directory tempat firefoxProfile disimpan.

3.2.4 Internet Explorer Capabilities

Internet explorer capabilities merupakan capabilities yang secara spesifik dimiliki oleh browser internet explorer. Untuk menggunakan capabilities yang ada dapat menggunakan Internet Explorer options. Capabilities yang terdapat pada internet explorer antara lain fileUploadDialogTimeout, ensureCleanSession, ignoreZoomSetting, ignoreProtectedModeSettings, silent, Command-Line Options, dan forceCreateProcessApi. Segmen program file upload dialog timeout akan diberikan pada segmen program 3.3

Segmen Program 3.4 Pengunaan Capability upload dialog time out

```
1.    InternetExplorerOptions options =
2.    new InternetExplorerOptions();
3.    options.waitForUploadDialogUpTo(Duration.ofSeconds(2));
4.    WebDriver driver = new RemoteWebDriver(options);
```

Segmen program 3.4 merupakan program untuk menggunakan capability upload dialog time out. Capability ini berguna untuk mengatur berapa lama jeda file upload dialog sebelum terbuka. Secara default nilai dari upload dialog time out ini adalah 1000ms. Pada segmen program 3.3 nilai dari capability ini berupa detik menggunakan duration.ofseconds sebanyak 2 detik. Capability berikutnya adalah ensure clean session. Capability ensure clean session akan diberikan pada segmen program 3.5.

Segmen Program 3.5 Pengunaan Capability ensure clean session

```
1.    InternetExplorerOptions options =
2.    new InternetExplorerOptions();
3.    options.destructivelyEnsureCleanSession();
4.    WebDriver driver = new RemoteWebDriver(options);
```

Segmen program 3.5 merupakan contoh penggunaan capability ensure clean session. Capability ensure clean session merupakan capability yang berfungsi untuk mengatur apakah cache di internet explorer di clear terlebih dahulu atau tidak saat menjalankan Selenium Webdriver. Parameter dari capability ini merupakan boolean dan secara default ensure clean session bernilai false. Capability ini dapat mengurangi performa karena menghapus browsing data seperti cache dan cookies. Capability berikutnya yang dimiliki oleh internet explorer options adalah ignore zoom settings. Segmen program untuk capability ignore zoom settings akan diberikan pada segmen program 3.6.

Segmen Program 3.6 Pengunaan ignore zoom setting

```
1.    InternetExplorerOptions options =
2.    new InternetExplorerOptions();
3.    options.ignoreZoomSettings();
4.    WebDriver driver = new RemoteWebDriver(options);
```

Segmen program 3.4 merupakan program penggunaan capability ignore zoom setting. Capability ignore zoom setting digunakan untuk mengatur testing hanya berjalan jika zoom 100%. Value dari parameter pada capability ini adalah Boolean dan secara default bernilai true. Capability berikutnya yang dimiliki oleh internet explorer option adalah ignore protected mode setting. Segmen program contoh capability ignore protected mode setting akan diberikan pada segmen program 3.7.

Segmen Program 3.7 Pengunaan ignore protected mode

```
1.    InternetExplorerOptions options =
2.    new InternetExplorerOptions();
3.    options.ignoreZoomSettings();
4.    WebDriver driver = new RemoteWebDriver(options);
```

Segmen program 3.7 merupakan segmen program penggunaan ignore protected mode. Protected mode pada internet explorer merupakan mode pada internet explorer dimana web yang dicurigai memiliki konten berbahaya seperti virus, spyware, dan adware disegel didalam objek software yang bernama

AppContainer. Capability ignore protected mode merupakan capability yang mengatur apakah cek protected mode dijalankan atau tidak dijalankan. Secara default capability ini bernilai false. Capability berikutnya adalah silent. Capability silent akan diberikan contohnya pada segmen program 3.8.

Segmen Program 3.8 Penggunaan capability silent

```
1.    InternetExplorerOptions options =
2.    new InternetExplorerOptions();
3.    options.setCapability("silent", true);
4.    WebDriver driver = new RemoteWebDriver(options);
```

Segmen program 3.8 merupakan segmen program untuk menunjukkan penggunaan capability silent. Capability ini merupakan capability yang digunakan apakah output diagnosa dari IE driver server diberikan atau tidak. Capability ini menerima parameter boolean. Secara default capability ini bernilai false. Capability berikutnya yang akan dijelaskan adalah capability disable native event. Capability ini akan diberikan contoh pada segmen program 3.9.

Segmen Program 3.9 Penggunaan capability disable native events

```
1.    InternetExplorerOptions options =
2.    new InternetExplorerOptions();
3.    options.disableNativeEvents();
4.    WebDriver driver = new RemoteWebDriver(options);
```

Segmen program 3.9 merupakan program dengan contoh penggunaan capability disable native event. Capability ini berguna untuk menentukan apakah native event dapat dijalankan pada Internet explorer atau tidak. Secara default capability ini memiliki nilai false. Capability berikutnya yang akan dijelaskan adalah capability disable native events. Capability ini akan diberikan contoh pada segmen program 3.9.

Segmen Program 3.10 Penggunaan capability disable native events

```
1.    InternetExplorerOptions options =
2.    new InternetExplorerOptions();
3.    options.disableNativeEvents();
4.    WebDriver driver = new RemoteWebDriver(options);
```

Segmen program 3.10 merupakan segmen program untuk capability disable native event. Capability ini berguna untuk menentukan apakah native event bisa berjalan di internet explorer atau tidak. Capability ini memiliki value

boolean dan secara default bernilai false. Capability berikutnya yang akan dijelaskan adalah force create process api. Segmen program untuk contoh capability ini akan diberikan pada segmen program 3.11.

Segmen Program 3.11 Penggunaan capability force create process api

```
1.    InternetExplorerOptions options =
2.    new InternetExplorerOptions();
3.    options. useCreateProcessApiToLaunchIe();
4.    WebDriver driver = new RemoteWebDriver(options);
```

Segmen program 3.11 merupakan segmen program penggunaan capability force create process api. Api ini digunakan agar capability command line dapat dijalankan. Command line yang dapat digunakan pada command line option adalah -k, -private, dan -extofl. Penggunaan command line akan diberikan contoh pada segmen program 3.12.

Segmen Program 3.12 Penggunaan command line option

```
1.    InternetExplorerOptions options =
2.    new InternetExplorerOptions();
3.    options. useCreateProcessApiToLaunchIe();
4.    options.addCommandSwitches("-k");
5.    options.addCommandSwitches("-private");
6.    options.addCommandSwitches("-extofl");
7.    WebDriver driver = new RemoteWebDriver(options);
```

Segmen program 3.12 merupakan program untuk menggunakan capability command line option. Pada segmen program terdapat 3 buah command yang digunakan yaitu -k, -private, dan -extofl. Command line pertama adalah -k. Command ini berguna untuk membuka internet explorer dalam mode kiosk. Kiosk mode adalah mode fullscreen pada internet explorer dimana title bar, menu bar, tool bar, dan status bar tidak ditampilkan.

Command line berikutnya adalah private. Command line ini berguna untuk membuka internet explorer dengan mode private browsing. Pada mode private browsing, browsing data saat melakukan pengujian menggunakan Selenium Webdriver tidak akan disimpan. Command line berikutnya adalah -extofl. Command line ini digunakan untuk memulai internet explorer tanpa menggunakan add ons yang ada pada internet explorer.

3. 3 Browser

Browser merupakan cara dari Selenium Webdriver untuk berinteraksi dengan browser web yang ada. Package yang digunakan untuk fitur browser ini adalah org.openqa.Selenium. Pada browser Selenium Webdriver dibagi menjadi 5 kelompok yaitu browser information itu sendiri, navigation, alert, cookies, frames, dan windows. Pada 3.3.1 akan dibahas mengenai browser information serta segmen program sebagai contoh penggunaan dan output dari setiap method yang termasuk browser information ini.

3.3.1 Browser Information

Browser information merupakan beberapa method yang digunakan untuk mendapatkan info browser yang digunakan. Method yang termasuk pada browser information ini adalah getTitle, getCurrentUrl, getPageSource. Method pertama yang akan dibahas adalah getTitle. Method ini akan dibahas pada segmen program 3.10.

Segmen Program 3.10 Penggunaan Method getTitle

```
1.    WebDriver driver = new ChromeDriver();
2.    String title = driver.getTitle();
3.    System.out.println(title);
```

Segmen program 3.10 merupakan segmen program contoh penggunaan method getTitle. Method getTitle digunakan untuk mendapatkan judul dari halaman yang dibuka. Pada segmen program 3.10 program akan mendapatkan judul halaman awal dan akan disimpan dalam variabel. Setelah judul disimpan dalam variable, judul akan ditampilkan pada console. Method berikutnya yang akan dijelaskan adalah method getCurrentUrl. Method getCurrentUrl akan dijelaskan pada segmen program 3.11.

Segmen Program 3.11 Penggunaan Method getCurrentUrl

```
1.    WebDriver driver = new ChromeDriver();
2.    String url = driver.getCurrentUrl();
3.    System.out.println(url);
```

Segmen program 3.11 merupakan segmen program contoh penggunaan method getCurrentUrl. Method getCurrentUrl digunakan untuk mendapatkan

judul dari halaman yang dibuka. Pada segmen program 3.11 program akan mendapatkan url halaman awal dan akan disimpan dalam variabel. Setelah url disimpan dalam variable, url akan ditampilkan pada console. Method berikutnya yang akan dijelaskan adalah method getPageSource. Method getPageSource akan dijelaskan pada segmen program 3.12.

Segmen Program 3.12 Penggunaan Method getCurrentUrl

```
1.    WebDriver driver = new ChromeDriver();
2.    String source = driver.getPageSource();
3.    System.out.println(source);
```

Segmen program 3.12 merupakan segmen program contoh penggunaan method getPageSource. Method getPageSource digunakan untuk mendapatkan source code dari website yang sedang dibuka. Pada segmen program 3.12 program akan mendapatkan source code dari halaman awal dan akan disimpan dalam variabel. Setelah source code disimpan dalam variable, source code akan ditampilkan pada console.

3.3.2 Browser Navigation

Browser navigation merupakan kelompok dari beberapa method yang digunakan untuk melakukan navigasi antar halaman. Method yang termasuk pada browser navigation ini adalah get dan navigate. Method navigate sendiri dibagi lagi menjadi navigate.to, navigate.back, navigate.forward, dan navigate.refresh. Method pertama yang akan dibahas adalah method get. Method ini akan dibahas pada segmen program 3.13.

Segmen Program 3.13 Penggunaan Method get untuk navigasi

```
1.    WebDriver driver = new ChromeDriver();
2.    driver.get("https://selenium.dev");
```

Segmen program 3.13 merupakan segmen program contoh penggunaan method get. Method get digunakan untuk melakukan navigasi pada web yang dipilih. Pada segmen program 3.13 website akan berpindah halaman menuju https://selenium.dev. Selain method get terdapat juga method navigate.to yang

memiliki fungsi sama dengan method get. Method navigate.to akan diberikan contoh pada segmen program 3.14

Segmen Program 3.14 Penggunaan Method navigate

```
1.    WebDriver driver = new ChromeDriver();
2.    driver.navigate().to("https://selenium.dev");
3.    driver.navigate().back();
4.    driver.navigate().forward();
5.    driver.navigate().refresh();
```

Segmen program 3.14 merupakan segmen program contoh penggunaan method naviagate. Terdapat 4 navigate yang dapat digunakan yaitu navigate to, navigate back, navigate forward, dan navigate refresh. Method pertama yang akan dibahas adalah method navigate to. Method ini digunakan untuk melakukan navigasi pada web yang dipilih. Pada segmen program 3.14 website akan berpindah halaman menuju https://selenium.dev. Method berikutnya yang akan dibahas adalah method navigate back.

Method navigate back digunakan untuk kembali pada website yang dibuka sebelumnya dan memiliki fungsi yang sama ketika user menekan tombol back. Jika website tidak memiliki halaman sebelumnya maka tidak akan ada yang terjadi pada method ini. Pada segmen program 3.14 website akan berpindah halaman menuju halaman awal saat membuka browser. Method berikutnya yang akan dibahas adalah method navigate forward.

Method navigate forward digunakan untuk kembali pada website yang dibuka sebelum menekan tombol back atau menjalankan method navigate back. Jika user belum pernah menekan tombol back atau method dari navigate back tidak ada maka method ini tidak menyebabkan terjadinya perubahan. Pada segmen program 3.14 website akan berpindah menuju halaman https://selenium.dev karena halaman ini merupakan halaman yang dibuka sebelum method navigate back dijalankan. Method berikutnya yang akan dibahas adalah method navigate refresh. Method navigate refresh digunakan untuk melakukan refresh pada halaman yang dibuka sekarang. Pada segmen program 3.14 website akan melakukan refresh pada halaman https://selenium.dev.

3.3.3 Browser Alert

Browser alert merupakan kelompok dari beberapa element yang termasuk dalam alerts. Selenium WebDriver menyediakan API untuk bekerja dengan tiga jenis popup yang dimiliki oleh JavaScript. Jenis popup yang tersedia pada Selenium WebDriver adalah alerts, confirm, dan prompts. Popup yang akan dibahas pertama adalah alerts. Contoh penggunaan alert akan diberikan pada segmen program 3.15

Segmen Program 3.15 Penggunaan Alerts

```
1.    WebDriver driver = new ChromeDriver();
2.    driver.findElement(By.Id("submit")).click();
3.    Alert alert = wait.until(
4.          ExpectedConditions.alertIsPresent()
5.    );
6.    String text = alert.getText();
7.    alert.accept();
```

Segmen program 3.15 merupakan segmen program contoh penggunaan alerts. Alerts sendiri merupakan sebuah pop up yang hanya memiliki satu buah button yang bisa di klik. Pada segmen program 3.15 pertama webdriver mencari elemen dengan id submit kemudian diklik. Setelah diklik webdriver menunggu hingga alert muncul kemudian text yang ada alert diambil dengan method getText(). Setelah mendapat text pada alert button alert ditekan menggunakan method accept(). Pop up yang akan dibahas berikutnya adalah pop up confirm. Pop up confirm. Contoh program untuk pop up jenis confirm akan diberikan pada segmen program 3.16.

Segmen Program 3.16 Penggunaan Confirm

```
1.    WebDriver driver = new ChromeDriver();
2.    driver.findElement(By.Id("submit")).click();
3.    Alert alert = wait.until(
4.          ExpectedConditions.alertIsPresent()
5.    );
6.    alert.dismiss();
```

Segmen program 3.16 merupakan segmen program contoh penggunaan confirm. Confirm merupakan sebuah pop up yang memiliki dua buah button yang bisa di klik yaitu ok dan cancel. Terdapat 3 buah method yang bisa digunakan untuk confirm ini yaitu getText, ok, dan cancel. Untuk mengklik ok pada confirm

dapat menggunakan method yang sama dengan penggunaan alert yaitu accept. Untuk mengklik button dismiss dapat menggunakan method dismiss. Untuk mendapatkan text pada alert dapat menggunakan getText. Pada segmen program 3.16 pertama webdriver mencari elemen dengan id submit kemudian diklik. Setelah diklik webdriver menunggu hingga alert muncul kemudian text yang ada alert diambil dengan method getText(). Setelah mendapat text pada alert button alert ditekan menggunakan method dismiss(). Pop up yang akan dibahas berikutnya adalah pop up confirm. Pop up prompt. Contoh program untuk pop up jenis prompt akan diberikan pada segmen program 3.17.

Segmen Program 3.17 Penggunaan Confirm

```
1.    WebDriver driver = new ChromeDriver();
2.    driver.findElement(By.Id("submit")).click();
3.
4.    Alert alert = wait.until(
5.          ExpectedConditions.alertIsPresent()
6.    );
7.    alert.dismiss();
```

Segmen program 3.17 merupakan segmen program contoh penggunaan confirm. Confirm merupakan sebuah pop up yang memiliki dua buah button yaitu ok dan cancel. Terdapat 3 buah method yang bisa digunakan untuk confirm ini yaitu getText, ok, dan cancel. Method accept digunakan untuk melakukan klik pada button ok, sedangkan method dismiss digunakan untuk melakukan klik pada button cancel. Method getText digunakan untuk mendapatkan text pada pop up confirm. Pop up yang akan dibahas berikutnya adalah pop up confirm. Pop up prompt. Contoh program untuk pop up jenis prompt akan diberikan pada segmen program 3.18.

Segmen Program 3.18 Penggunaan prompt

```
1.    WebDriver driver = new ChromeDriver();
2.    driver.findElement(By.Id("submit")).click();
3.    Alert alert = wait.until(
4.          ExpectedConditions.alertIsPresent()
5.    );
6.    alert.sendKeys("Selenium");
7.    alert.accept();
```

Segmen program 3.18 merupakan segmen program contoh penggunaan prompt. Prompt merupakan sebuah pop up yang memiliki dua buah button dan sebuah text field. Terdapat 3 buah method yang dapat digunakan untuk prompt antara lain sendKeys, accept, dan dismiss. Method sendKeys digunakan untuk mengisi textfield. Method accept digunakan untuk melakukan klik pada button ok dan method dismiss digunakan untuk melakukan klik pada button dismiss.

3.3.4 Browser Cookies

Cookie merupakan kumpulan informasi yang berisi data aktivitas ketika user menelusuri sebuah website. Secara sederhana cookies adalah kumpulan data yang diterima komputer dari sebuah website. Dengan cookies, website bisa menyimpan aktivitas yang dilakukan pengunjung website. Selenium webdriver menyediakan beberapa method untuk berinteraksi dengan cookies antara lain: menambah cookie, menghapus cookie, dan mendapatkan data cookies. Method yang akan dijelaskan pertama adalah method untuk menambah cookie.

Segmen Program 3.19 Menambah Cookie

```
1.    WebDriver driver = new ChromeDriver();
2.    driver.get("https://www.google.com");
3.    driver.manage().addCookie(new Cookie
4.          ("user", "sam")
5.    );
```

Segmen program 3.19 merupakan segmen program untuk menambahkan cookie. Method add Cookie hanya menerima satu set objek JSON yang diserialisasi.

Untuk dapat menambah cookie user harus membuka website dan berada di halaman cookie yang valid. Pada segmen program 3.19 Selenium membuka halaman google.com dan menambahkan cookie dengan nama user dan value sam. Selain add cookie Selenium juga menyediakan method untuk mendapat cookie. Ada 2 method yang berkaitan dengan mendapatkan cookie yaitu getCookieNamed, dan getCookies. Contoh program untuk kedua method ini akan diberikan pada segmen program 3.20.

Segmen Program 3.20 Get cookies

```
1.    WebDriver driver = new ChromeDriver();
2.    driver.get("https://www.google.com");
3.    driver.manage().addCookie(new Cookie
4.          ("user", "sam")
5.    );
6.    Cookie cookie1 = driver.manage().getCookieNamed("user");
7.    Set <Cookie> cookies = driver.manage().getCookies();
8.    System.out.println(cookie1);
9.    System.out.println(cookies);
```

Segmen program 3.20 merupakan segmen program untuk mendapatkan cookie. Pada segmen program 3.20 terdapat dua cara untuk mendapatkan cookie yaitu getCookieNamed dan getCookies. Method getCookieNamed digunakan untuk mendapatkan satu buah cookie. Pada segmen program 3.20 getCookies digunakan untuk mendapatkan semua cookie yang ada pada halaman google.com. Method getCookieNamed digunakan untuk mendapatkan cookie yang dipilih. Fitur berikutnya untuk browser cookies adalah menghapus cookie. Contoh program untuk menghapus cookie ada pada program 3.21

Segmen Program 3.21 Delete cookies

```
1.    WebDriver driver = new ChromeDriver();
2.    driver.get("https://www.google.com");
3.    driver.manage().addCookie(new Cookie
4.          ("user", "sam")
5.    );
6.    Cookie cookie1 = new Cookie(“name”,”Sam”);
7.    Cookie cookie2 = new Cookie(“age”,”22”);
8.    driver.manage().addCookie(cookie1);
9.    driver.manage().deleteCookieNamed("user");
10.   driver.manage().deleteCookie(cookie1);
11.   driver.manage().deleteAllCookies();
```

Segmen program 3.21 merupakan contoh program untuk menghapus cookie yang ada. Ada 2 method untuk menghapus cookie yaitu deleteCookieNamed, deleteCookie, dan deleteAllCookies. Method deleteCookieNamed digunakan untuk menghapus cookie dengan parameter nama cookie. Pada segmen program 3.21 method deleteCookieNamed akan menghapus cookie dengan nama user. Method deleteCookie digunakan untuk menghapus cookie dengan parameter object cookie.

Pada segmen program 3.21 method delete cookie akan menghapus cookie dengan nama cookie1. Method deleteAllCookies adalah method untuk menghapus

semua cookie yang ada pada browser. Cookie yang dikirim menggunakan Selenium Webdriver memiliki dua jenis yaitu strict dan lax. Contoh program untuk dua jenis cookie ini akan diberikan pada segmen program 3.22

Segmen Program 3.22 Delete cookies

```
1.    WebDriver driver = new ChromeDriver();
2.    driver.get("https://www.google.com");
3.    driver.manage().addCookie(new Cookie
4.          ("user", "sam")
5.    );
6.    Cookie cookie1 = new Cookie(“name”,”Sam”);
7.    Cookie cookie2 = new Cookie(“age”,”22”);
8.    driver.manage().addCookie(cookie1);
9.    driver.manage().deleteCookieNamed("user");
10.   driver.manage().deleteCookie(cookie1);
11.   driver.manage().deleteAllCookies();
```

Segmen program 3.22 merupakan contoh program untuk melakukan delete cookies. Pada segmen program 3.22 terdapat 3 buah method berbeda yang digunakan untuk menghapus cookie yaitu deleteCookieNamed, deleteCookie, dan deleteAllCookies. Method deleteCookieNamed digunakan untuk mengahapus cookie dengan parameter string yang merupakan nama cookie yang ingin dihapus. Method deleteCookie akan menghapus cookie dengan parameter objek cookie. Method deleteAllCookies merupakan method yang digunakan untuk menghapus semua cookie yang ada.

3.3.5 Browser Frames

Iframe merukapan elemen di halaman web yang merupakan tempat dari berbagai jenis media seperti gambar, dokumen, dan video. Media tersebut dapat berupa gambar atau dokumen internal yang berhubungan dengan website atau dari website lain. Selenium WebDriver memungkinkan user untuk melakukan testing pada Iframe sama seperti elemen lainnya. Perbedaan dari Iframe dengan elemen lainnya adalah cara untuk mencarinya. Contoh dari penggunaan Iframe akan diberikan pada segmen program 3.23.

Segmen Program 3.23 Menekan button di dalam iframe

```
1.    driver.findElement(By.tagName("button")).click();
2.    driver.switchTo().frame("buttonframe");
```

```
3.    driver.findElement(By.tagName("button")).click();
```

Segmen program 3.23 merupakan contoh program untuk menekan button yang terletak didalam iframe. Iframe tidak bisa dicari langsung menggunakan method findElement. Untuk mendapatkan iframe dan isinya diperlukan method switchTo. Baris program pertama akan menghasilkan error no element found karena driver tidak bisa menemukan elemen button yang berada di dalam iframe. Error ini terjadi karena Selenium hanya mengetahui elemen-elemen dalam dokumen tingkat atas. Untuk dapat menekan button pertama driver harus berpindah pada frame kemudian baru dicari elemen tersebut. Method yang digunakan untuk berpindah frame adalah method switchTo. Contoh program cara mendapatkan iframe akan diberikan pada segmen program 3.24

Segmen Program 3.24 Mencari Elemen Iframe

```
1.    WebElement iframe = driver.findElement(
2.          By.cssSelector("#modal>iframe")
3.    );
4.    driver.switchTo().frame("buttonframe");
5.    driver.switchTo().frame(1);
6.    driver.switchTo().defaultContent();
```

Segmen program 3.24 merupakan segmen program cara untuk mencari elemen iframe. Ada 3 cara untuk mendapatkan iframe yaitu dengan webElement, id, class, dan dengan index. Baris 1 pada segmen program 3.24 merupakan contoh untuk mendapatkan iframe menggunakan webElement. Baris ke 4 pada segmen program 3.24 merupakan contoh untuk mendapatkan iframe dengan menggunakan id, class, ataupun name. Baris ke 5 merupakan cara untuk mendapatkan iframe dengan menggunakan index. Baris ke 6 merupakan cara untuk keluar dari frame yang dipilih dengan cara kembali ke konten awal.

3.3.6 Browser Windows

Selenium WebDriver tidak membuat perbedaan antara jendela dan tab. Saat user membuka tab atau windows baru, Selenium akan membiarkan user bekerja menggunakan window handles. Setiap window memiliki identifier unik

yang tidak akan berubah pada satu sesi. Contoh dari penggunaan window handle akan diberikan pada segmen program 3.25

Segmen Program 3.25 Berpindah browser dan tab

```
1. String originalWindow = driver.getWindowHandle();
2. assert driver.getWindowHandles().size() == 1;
3. driver.findElement(By.linkText("new window")).click();
4. wait.until(numberOfWindowsToBe(2));
```

Lanjutan Segmen Program 3.25 Berpindah browser dan tab

```
5. for (String windowHandle : driver.getWindowHandles()) {
6.     if(!originalWindow.contentEquals(windowHandle)) {
7.         driver.switchTo().window(windowHandle);
8.         break;
9.     }
10.   }
11.   wait.until(titleIs("Selenium documentation"));
```

Segmen program 3.25 merupakan cara yang diberikan Selenium untuk berpindah window dan tab pada browser. Pertama akan di cek apakah window handle memiliki Panjang lebih dari 1. Jika window handle memiliki panjang lebih dari 1 maka window handle tadi akan diloop dan akan dipindahkan ke tab yang lain. Untuk berpindah tab dapat menggunakan method switchTo dengan parameter string yang berisi identifier window handle. Contoh untuk membuka tab baru atau windows baru akan diberikan pada segmen program 3.26.

Segmen Program 3.26 Membuka new tab atau new Window

```
1.    driver.findElement(
2.          By.cssSelector("Body")
3.    ).sendKeys(Keys.CONTROL+”t”);
4.    driver.findElement(
5.          By.cssSelector("Body")
6.    ).sendKeys(Keys.CONTROL+”n”);
```

Segmen program 3.26 merupakan segmen program contoh membuka tab baru dan window baru pada Selenium. Pada Selenium Webdriver milik Selenium 3 untuk membuka tab baru dan windows baru dapat menggunakan shortcut yang disediakan oleh windows yaitu control dan t untuk membuka tab baru dan control dengan n untuk membuka windows baru. Tab dan windows yang dibuka akan langsung tampil sebagai window yang paling depan sehingga secara default semua testing yang dilakukan setelah membuka windows atau tab baru akan dijalankan pada tab atau windows baru tersebut. Berikutnya akan dijelaskan cara

untuk menutup browser yang dibuka dan bagaimana cara kembali ke tab atau windows yang dibuka sebelumnya. Contoh untuk cara menutup windows atau tab dan cara untuk kembali pada tab sebelumnya akan diberikan pada segmen program 3.27.

Segmen Program 3.27 Menutup tab dan kembali ke tab sebelumnya

```
1.    driver.close();
2.    driver.switchTo().window(originalWindow);
```

Segmen program 3.27 merupakan segmen program contoh untuk menutup tab yang dibuka dan kembali ke tab sebelumnya. Pada segmen program 3.27 driver.close() digunakan untuk menutup tab atau window yang sedang terbuka sekarang. Bagian 2 dari segmen program 3.27 digunakan untuk kembali ke tab yang disimpan. Pada segmen program 3.25 terdapat String originalWindow dimana data window disimpan.

Method pada baris 2 dari segmen program 3.27 diguanakan untuk kembali pada tab yang telah disimpan datanya pada string originalWindow. Jika user lupa untuk beralih kembali ke tab atau window lain setelah menutup window, Selenium Webdriver akan memunculkan error yaitu No such Window Exception. Error ini terjadi karena setelah tab atau window di close maka tidak ada Window atau tab yang dijalankan di Selenium Webdriver sehingga script dijalankan di window yang sudah di close. Berikutnya akan dijelaskan cara untuk menutup browser pada Selenium Webdriver. Contoh untuk menutup browser akan diberikan pada segmen program 3.28

Segmen Program 3.28 Menutup browser

```
1.    @AfterAll
2.    public static void tearDown() {
3.        driver.quit();
4.    }
```

Segmen program 3.28 merupakan segmen program untuk menutup browser driver yang dijalankan. Penutupan browser driver dilakukan ketika test yang dilakukan telah selesai dilakukan. Secara spesifik driver.quit akan menutup semua window yang berkaitan dengan browser driver tersebut. Selain itu method driver.quit() juga akan menutup semua browser dan driver yang aktif pada

process. Jika user tidak menggunakan test context @AfterAll dapat diganti menggunakan try dan finally. Berikutnya akan dijelaskan cara untuk mendapatkan ukuran window yang dibuka dan mengganti resolusi window. Contoh program untuk mendapatkan ukuran window yang dibuka dan mendapatkan resolusi akan diberikan pada segmen program 3.29.

Segmen Program 3.29 Mendapatkan ukuran window dan mengganti resolusi

```
1.    int width = driver.manage().window().getSize().getWidth();
2.    int height = driver.manage().window().getSize().getHeight();
3.    Dimension size = driver.manage().window().getSize();
4.    int width1 = size.getWidth();
5.    int height1 = size.getHeight();
6.    driver.manage().window().setSize(new Dimension(1024, 768));
```

Segmen program 3.29 merupakan contoh program untuk mendapatkan ukuran window. Ada dua cara untuk mendapatkan dimensi dari window yang dibuka. Cara pertama adalah dengan menggunakan variabel dengan tipe data int untuk menampung hasil dari method getWidth() dan getHeight(). Cara kedua dapat dilihat pada baris ketiga di segmen program 3.29 yaitu dengan menggunakan Dimension. Baris 6 pada segmen program 3.29 merupakan cara untuk mengubah resolusi pada window dari browser yang diguanakan. Berikutnya akan dijelaskan mengenai cara untuk mendapatkan posisi window dan cara memindahkan posisi window yang dibuka. Contoh program untuk mendapatkan posisi window dan cara memindahkan posisi window yang dibuka akan diberikan pada segmen program 30

Segmen Program 3.30 Mendapatkan dan mengubah posisi window

```
1.    int x = driver.manage().window().getPosition().getX();
2.    int y = driver.manage().window().getPosition().getY();
3.    Point position = driver.manage().window().getPosition();
4.    int x1 = position.getX();
5.    int y1 = position.getY();
6.    driver.manage().window().setPosition(new Point(0, 0));
```

Segmen program 3.30 merupakan segmen program untuk mendapatkan posisi window dan mengubah posisi window dari browser. Posisi dari window yang didapatkan berupa koordinat dengan tipe data integer. Untuk memindahkan posisi window dapat menggunakan method setPosition dengan parameter Point

yang berisi dua buah integer yaitu x dan y. Berikutnya akan dijelaskan cara untuk melakukan maximize, minimize, dan membuat window menjadi fullscreen. Segmen program contoh untuk melakukan maximize, minimize, dan membuat window menjadi fullscreen akan diberikan pada segmen program 3.31.

Segmen Program 3.31 Minimize, Maximize, dan Fullscreen Window

```
1.    driver.manage().window().maximize();
2.    driver.manage().window().minimize();
3.    driver.manage().window().fullscreen();
```

Segmen program 3.31 merupakan contoh program untuk melakukan minimize, maximize, dan fullscreen pada window. Baris pertama dari segmen program 3.31 merupakan script untuk melakukan maximize pada window. Baris kedua dari segmen program 3.31 merupakan script untuk melakukan minimize window. Baris ketiga dari segmen program 3.31 merupakan script untuk melakukan fullscren pada window. Berikutnya akan dibahas cara untuk melakukan screenshot. Contoh program untuk melakukan screenshot akan diberikan pada segmen program 3.32

Segmen Program 3.32 Screenshot Window

```
1.    WebDriver driver = new FirefoxDriver();
2.    driver.get("http://www.google.com/");
3.    File scrFile = (
4.          (TakesScreenshot)driver
5.    ).getScreenshotAs(OutputType.FILE);
6.    FileUtils.copyFile(scrFile,
7.          new File("c:\\tmp\\screenshot.png")
8.    );
```

Segmen program 3.32 merupakan segmen program untuk melakukan screenshot pada window. Hasil dari screenshot adalah file yang disimpan dalam bentuk encoding basis 64. Berikutnya akan dijelaskan mengenai print halaman dari web yang dibuka. Contoh segmen program untuk melakukan print halaman akan diberikan pada segmen program 3.33.

Segmen Program 3.33 Print Window

```
1.    import org.openqa.selenium.print.PrintOptions;
2.    driver.get("https://www.selenium.dev");
3.    printer = (PrintsPage) driver;
4.    PrintOptions printOptions = new PrintOptions();
```

```
5.     printOptions.setPageRanges("1-2");
6.     Pdf pdf = printer.print(printOptions);
7.     String content = pdf.getContent();
```

Segmen program 3.33 merupakan segmen program contoh cara melakukan print pada window yang dipilih. Halaman yang diprint akan dibentuk menjadi seperti file dengan format pdf. File hasil print juga bisa disetting berapa halaman file akan diprint. Syarat untuk menggunakan print window ini adalah chromium browser harus dijalankan dalam mode headless.

3. 4 Web Elements

Elemen adalah bagian penting dari sesuatu yang abstrak, dengan cara yang sama Sertifikasi Selenium merangkum elemen bentuk sederhana yang disebut WebElement yang mewakili elemen HTML. Jadi, dalam artikel ini, kita akan menggali lebih dalam untuk memahami peran utama yang dimainkan oleh WebElement di Selenium. Apa pun yang ada di halaman web adalah WebElement seperti kotak teks, tombol, dll. WebElement mewakili elemen HTML. Selenium WebDriver merangkum elemen bentuk sederhana sebagai objek WebElement. Ini pada dasarnya mewakili elemen DOM dan semua dokumen HTML dibuat oleh elemen HTML ini. Terdapat lima pengelompokkan untuk web element ini yaitu locators, finder, interactions, information, dan select list. Pada bagian 3.4.1 akan dibahas mengenai locator atau cara menemukan web element yang ada pada halaman web.

3.4.1 Locators

Locator merupakan cara Selenium untuk menemukan elemen di web page. Ada dua method yang bisa digunakan untuk menemukan element yaitu findElement() dan findElements(). Method findElement() mengembalikan objek WebElement berdasarkan kriteria pencarian yang ditentukan atau akhirnya melemparkan pengecualian jika tidak menemukan elemen yang cocok dengan kriteria pencarian. Method findElements() mengembalikan daftar WebElements yang cocok dengan kriteria pencarian. Jika tidak ada elemen yang ditemukan, ia mengembalikan daftar kosong. Locator yang terdapat Selenium adalah mencari

dengan class, id, name, DOM, link text, partial link text, dan HTML tag name. Contoh penggunaan locator akan diberikan pada segmen program 3.34

Segmen Program 3.34 Contoh Pengunaan Locator

```
1.    driver.findElement(By.id("IdValue"));
2.    driver.findElement(By.name("nameValue"));
3.    driver.findElement(By.className("classValue"));
4.    driver.findElement(By.linkText("textofLink"));
5.    driver.findElement(By.partialLinkText("PartialTextofLink"));
6.    driver.findElement(By.tagName("htmlTag");
7.    driver.findElement(By.cssSelector("cssValue")
8.    driver.findElement(By.xpath("xpathValue"))
```

Segmen program 3.34 merupakan contoh cara penggunaan locator. Ada du cara untuk menemukan element yaitu dengan menggunakan method findElement dan findElements. Method findElement digunakan untuk mendapatkan satu buah webElement. Method findElements digunakan untuk mendapatkan lebih dari satu webElement dan memiliki return value array. Pertama akan dibahas mengenai cara penggunaan id.

Locator menggunakan ID di Selenium merupakan metode tercepat dan teraman untuk menemukan elemen karena ID seharusnya hanya satu dan unik, tetapi tidak semua developer mengikuti aturan ini karena tidak ada error ketika developer menggunakan id yang sama untuk dua atau lebih elemen. Beberapa kasus yang umum adalah saat pembuatan elemen dilakukan secara dinamis. Selain dalam kasus itu locator ID memiliki tingkat kecepatan dan keakuratan yang tinggi untuk digunakan sebagai pencari elemen. Contoh dari cara menggunakan locator ID dapat dilihat pada segmen program 3.34 baris pertama.

Locator berikutnya adalah selector dengan menggunakan name dari suatu elemen. Selector name tidak seperti selector ID, yang unik untuk sebuah halaman, pencari Nama mungkin atau mungkin tidak memiliki nilai unik. Jika ada WebElements dengan nama yang sama, locator memilih elemen pertama dengan Nama tersebut di halaman. Jika tidak ada nama yang cocok dengan nilai atribut yang ditentukan, NoSuchElementException dimunculkan. Contoh penggunaan selector name ada pada segmen program 3.34 baris kedua.

Locator berikutnya yang akan dijelaskan adalah selector menggunakan className. Locator class name digunakan untuk menemukan WebElements yang

didefinisikan menggunakan atribut class. Class name tidak memiliki sifat unique seperti ID sehingga memungkinkan untuk menemukan lebih dari satu web element dengan class name yang sama. Contoh penggunaan selector class name ada pada segmen program 3.34 baris ketiga.

Locator berikutnya yang akan dijelaskan adalah selector link text. Selector link text merupakan cara untuk menemukan elemen berdasarkan text dari tag (a) atau elemen yang digunakan untuk membuat hyperlink. Ada dua jenis pencarian yang dilakukan menggunakan link text yaitu partial link text dan full link text. Contoh penggunaan link text ada pada segmen program 3.34 baris ketiga sedangkan partial link text pada segmen program 3.34 di baris ke lima.

Locator berikutnya yang akan dijelaskan adalah selector menggunakan html tag name. HTML tag name merupakan tag yang menentukan jenis apakah web element tersebut. Beberapa contoh dari HTML tag name adalah div, span, dan table. Locator ini akan mencari web element berdasarkan html tag tersebut. Contoh penggunaan locator html tag name ada pada segmen program 3.34 baris ke enam. Locator berikutnya yang akan dibahas adalah css selector. Ada beberapa cara untuk penggunaan css selector. Contoh dari penggunaan css selector akan diberikan pada segmen program 3.35

Segmen Program 3.35 Contoh Pengunaan css Selector

```
1.    WebElement firstName =
2.    driver.findElement(By.cssSelector(
3.          "input[name='firstname']"
4.    ));
5.    driver.findElement(By.cssSelector("input#firstname"));
6.    driver.findElement(By.cssSelector("input.myForm"));
7.    driver.findElement(By.cssSelector("div[class='ajax_enabled']
8.    [style='display:block']"));
```

Segmen program 3.35 merupakan contoh penggunaan css selector. Baris pertama dan kedua merupakan cara mencari elemen dengan tipe input dan memiliki name firstname. Baris ke lima merupakan cara untuk mencari elemen dengan tipe input dan memiliki id first name. Baris ke enam merupakan cara untuk mencari elemen dengan tipe input dan class yang bernama myForm. Baris ketujuh merupakan cara untuk menggunakan css Selector dengan kondisi lebih dari satu. Pada contoh baris ketujuh elemen yang dicari adalah elemen dengan div

yang memiliki class ajax_enabled, dan memiliki style display block. Berikutnya akan dijelaskan mengenai interaction.

3.4.2 Interaction

Interaction merupakan interaksi yang bisa dilakukan oleh user terhadap elemen. Terdapat 5 interaction dasar yang bisa dilakukan oleh user terhadap web element yaitu click, select, submit, send key, dan clear. Interaction yang akan dibahas adalah click. Untuk interaksi yang menggunakan Action akan dibahas pada bagian Action. Contoh penggunaan interaction click ada pada segmen program 3.36

Segmen Program 3.36 Contoh Pengunaan Interaction Click

```
1.    driver.get("https://www.google.com/")
2.    WebElement button = driver.findElement(By.cssSelector(
3.          "input[type=submit]"
4.    ));
5.    button.click()
```

Segmen program 3.36 merupakan contoh penggunaan interaction click. Pada segmen program 3.36 elemen yang diklik merupakan input dengan tipe submit. Interaction click dapat dilakukan pada semua elemen yang tidak dalam status hidden ataupun terdisable. Untuk melakukan klik pada element yang di disable dapat menggunakan javascript executor. Jika tester melakukan klik pada elemen yang tidak bisa di klik maka Selenium akan memunculkan error yaitu Element is not clickable at point. Ada beberapa penyebab dari exception ini selain melakukan klik pada element yang terdisabled. Penyebab pertama adalah Elemen yang bertumpuk.

Di sebagian besar aplikasi dinamis, ada elemen yang saling bertumpuk. Elemen bertumpuk yang dimaksud adalah modal dan pop up. Jika Selenium melakukan klik pada elemen yang terletak di bawah pop-up exception Element is not clickable akan ditampilkan. Penyebab berikutnya adalah elemen yang belum direnderer secara sempurna. Agar saat test dilakukan tidak terjadi error, maka perlu ditambahkan pengecekan apakah elemen yang dicari bisa diklik. Contoh pengecekan akan diberikan pada segmen program 3.37

Segmen Program 3.37 Pengecekan Apakah Elemen bisa diklik

```
1.    WebDriverWait wt = new WebDriverWait(driver,6);
2.    wt.until(ExpectedConditions.elementToBeClickable
3.    (By.className("button")));
```

Segmen program 3.37 merupakan contoh pengecekan apakah elemen bisa diklik. Jika elemen tidak bisa diklik maka Selenium akan memberikan error berupa timeout. Selain click terdapat interaction lain yaitu sendKeys. Send Keys merupakan method yang terdapat pada Selenium untuk mengetik konten secara otomatis ke dalam elemen yang bersifat editable. Send Keys tidak seperti click yang bisa dilakukan pada elemen pada apapun. Contoh program untuk send key ada pada segmen program 3.38

Segmen Program 3.38 Contoh Send Keys

```
1.    driver.get("https://google.com");
2.    driver.findElement(By.name(“q”)).sendKeys(“tes”);
```

Segmen program 3.38 merupakan contoh program untuk melakukan interaksi send keys. Pada segmen program 3.38 program akan membuka website google.com dan akan mencari elemen dengan nama q yang merupakan input untuk melakukan search. Send Keys akan mengisikan tulisan test pada search bar. Interaction berikutnya adalah clear. Clear digunakan untuk menghapus tulisan pada Elemen yang bersifat editable. Contoh penggunaan clear akan diberikan pada segmen program, 3.39

Segmen Program 3.39 Contoh Penggunaan Clear

```
1.    driver.get("https://google.com");
2.    WebElement search = driver.findElement(By.name("q"));
3.    search.sendKeys("tes");
4.    searchInput.clear();
```

Segmen program 3.39 merupakan contoh program untuk melakukan interaksi clear. Pada segmen program 3.39 program akan membuka website google.com dan akan mencari elemen dengan nama q yang merupakan input untuk melakukan search. Sama seperti segmen program 3.38, Selenium akan mengisikan text test. Setelah melakukan sendKeys method clear digunakan untuk menghapus tulisan pada elemen search. Interaction berikutnya adalah submit.

Interaction submit memiliki fungsi yang sama dengan interaction click. Ada beberapa perbedaan antara interaction submit dengan interaction click.

Perbedaan pertama interaction submit digunakan untuk mengirimkan data pada form sedangakan interaction click digunakna untuk menekan button yang ada pada form. Interaction submit tidak bisa digunakan jika pada form terdapat button dengan tipe submit. Perbedaan kedua adalah interaction click membutuhkan wait secara manual agar interaction dijalankan ketika website telah selesai di load. Interaction Submit secara otomatis akan dijalankan ketika halaman telah selesai di load. Contoh program untuk penggunaan interaction submit akan diberikan pada segmen program 3.40

Segmen Program 3.40 Contoh Penggunaan Submit

```
1.    driver.findElement(By.id("email")).sendKeys("a@gmail.com");
2.    driver.findElement(By.id("pass")).sendKeys("123456");
3.    driver.findElement(By.id("pass")).submit();
```

Segmen program 3.40 merupakan contoh penggunaan interaction submit pada sebuah form. Pada segmen 3.40 program melakukan submit form melalui elemen pass. Pada segmen program 3.40 dapat terlihat perbedaan penggunaan click dan penggunaan submit. Jika menggunakan click maka method click dijalankan pada elemen button. Jika menggunakan submit maka method submit dijalankan pada elemen lain selain button.

3.4.3 Information

Setiap elemen yang ada pada Selenium dapat didapatkan setiap informasi. Informasi yang bisa didapatkan adalah display, enabled, selected, tag name, size, position, css value, text attribute, DOM property, dan DOM attribute. Informasi pertama yang akan dibahas adalah display. Contoh program untuk display ada pada segmen program 3.41

Segmen Program 3.41 Mendapatkan status Display

```
1.    WebElement email = driver.findElement(By.id("email"))
2.    Boolean display = email.isDisplayed();
3.    System.out.println("Element displayed is :"+display);
```

Segmen program 3.41 merupakan cara untuk mendapatkan apakah elemen yang dipilih sedang ditampilkan atau di hidden. Status display dari suatu elemen dapat didapatkan dengan menggunakan method isDisplayed(). Jika elemen yang dicari ditampilkan, maka nilai yang dikembalikan adalah true. Jika elemen yang dicari hidden atau tidak ditampilkan, maka nilai yang dikembalikan false. Information berikutnya adalah enabled. Contoh program untuk mendapatkan status enabled dari suatu elemen diberikan pada segmen program 3.42

Segmen Program 3.42 Mendapatkan status Enabled

```
1.    WebElement email = driver.findElement(By.id("email"))
2.    Boolean enable = email.isEnabled();
3.    System.out.println("Element Enabled is :"+enable);
```

Segmen program 3.42 merupakan program untuk mendapatkan status enable pada suatu elemen. Status enabled dari suatu elemen dapat didapatkan dengan menggunakan method isEnabled(). Jika elemen yang dicari dalam status aktif, maka nilai yang dikembalikan adalah true. Jika elemen yang dicari tidak aktif maka nilai yang dikembalikan false. Information berikutnya adalah selected. Contoh program untuk mendapatkan status selected dari suatu elemen diberikan pada segmen program 3.43

Segmen Program 3.43 Mendapatkan status selected

```
1.    WebElement option= driver.findElement(By.id("option-1"))
2.    Boolean selected = option.isSelected();
3.    System.out.println("option 1 is :"+selected);
```

Segmen program 3.43 merupakan program untuk mendapatkan status selected dari suatu elemen. Status selected dari suatu elemen dapat didapatkan dengan menggunakan method isSelected(). Selected biasa digunakan pada radio button, checkbox, dan option. Jika elemen yang dicari dalam status terpilih, maka nilai yang dikembalikan adalah true. Jika elemen yang dicari tidak terpilih maka nilai yang dikembalikan false. Jika method isSelected() digunakan pada elemen lain, Selenium tidak akan memberikan exception tetapi nilai selected selalu bernilai false. Information yang akan dibahas berikutnya adalah tag name. Contoh program untuk mendapatkan tag name diberikan pada segmen program 3.44

Segmen Program 3.44 Mendapatkan Tag Name dari Suatu Elemen

```
1.    WebElement email = driver.findElement(By.id("email"))
2.    String tagName = email. getTagName();
3.    System.out.println("Email tag Name is :"+tagName);
```

Segmen program 3.44 merupakan program untuk mendapatkan tag Name pada suatu elemen. Tag Name dari suatu elemen dapat didapatkan dengan menggunakan method getTagName(). Pada program 3.44 tag name yang didapat adalah input karena elemen dengan id email merupakan sebuah input. Information yang akan dibahas berikutnya adalah informasi tentang ukuran dan posisi dari suatu elemen. Contoh program untuk mendapatkan ukuran dan posisi dari suatu elemen akan diberikan pada segmen program 3.45

Segmen Program 3.45 Mendapatkan Ukuran dan Posisi dari Suatu Elemen

```
1.    WebElement email = driver.findElement(By.id("email"))
2.    Rectangle res =  driver.findElement(
3.          By.cssSelector("h1")
4.    ).getRect();
5.    int height = res.getHeight();
6.    int width = res.getWidth();
7.    int x = res.getX();
8.    int y = res.getY();
```

Segmen program 3.45 merupakan program untuk mendapatkan ukuran dan posisi dari suatu elemen. Pada program 3.45 hasil posisi ditampung dalam class Rectangle. Class Rectangle memiliki method untuk mendapatkan Panjang, lebar, tinggi, koordinat x, dan y.Untuk mendapatkan tinggi dari elemen yang dipilih dapat menggunakan method getHeight(). Contoh penggunaan method getHeight() ada pada segmen program 3.45 baris kelima. Untuk mendapatkan panjang dari elemen yang dipilih dapat menggunakan method getWidth(). Contoh penggunaan method getWidth ada pada segmen program 3.45 baris keenam.

Selain tinggi dan lebar, class Rectangle juga bisa digunakan untuk mendapatkan koordinat x, dan koordinat y dari suatu elemen. Koordinat x dan y yang didapatkan bersifat relative terhadap ukuran dari browser window yang dibuka. Untuk mendapatkan koordinat x dapat menggunakan method getX() yang terdapat pada segmen program 3.45 baris ketujuh. Untuk mendapatkan koordinat y dapat menggunakan method getY() yang terdapat pada segmen program 3.45 baris kedelapan. Information berikutnya yang akan dibahas adalah mendapatkan

nilai dari suatu css. Contoh program untuk mendapatkan nilai dari suatu css pada elemen akan diberikan pada segmen program 3.46

Segmen Program 3.46 Mendapatkan Ukuran dan Posisi dari Suatu Elemen

```
1.    WebElement email = driver.findElement(By.id("email"))
2.    String cssValue = email.getCssValue(“color”);
```

Segmen program 3.46 merupakan contoh program untuk mendapatkan nilai css dari suatu elemen. Nilai css dari suatu elemen dapat didapatkan menggunakan method getCssValue(). Hasil dari method ini adalah string dan jika elemen yang dicari tidak memiliki css tersebut maka string yang dikembalikan adalah string kosong. Contoh penggunaan getCssValue ada pada segmen program 3.46 baris kedua. Information berikutnya yang akan dibahas adalah text attribute. Contoh penggunaan method untuk mendapatkan text attribute diberikan pada segmen program 3.47.

Segmen Program 3.47 Mendapatkan Text dari suatu Elemen

```
1.    WebElement h1 = driver.findElement(
2.          By.cssSelector("h1")
3.    );
4.    String text = h1.getText();
```

Segmen program 3.47 merupakan contoh program untuk mendapatkan text dari suatu elemen. Text dari suatu elemen dapat didapatkan menggunakan method getText(). Hasil dari method ini adalah string. Contoh penggunaan getText() ada pada segmen program 3.47 baris keempat. Berikutnya akan dibahas mengenai cara mendapatkan attribute dari suatu elemen. Contoh program untuk mendapatkan attribute dari suatu elemen akan diberikan pada segmen program 3.48

Segmen Program 3.48 Mendapatkan Attribute dari suatu Elemen

```
1.    WebElement img =driver.findElement(
2.          By.xpath("//img[@class='tp-logo']")
3.    );
4.    String src = img.getAttribute("src");
```

Segmen program 3.48 merupakan contoh program untuk mendapatkan attribute dari suatu elemen. Attribute dari suatu elemen dapat didapatkan menggunakan method getAttribute(). Method get attribute memiliki parameter

berupa string berisi nama attribute yang ingin dicari nilainya. Hasil dari method ini adalah string. Contoh penggunaan getAttribute() ada pada segmen program 3.48 baris keempat.

3.4.4 Select List

Select list berbeda dengan elemen lain. Select list memiliki fitur-fitur tersendiri sehingga select list tidak digabung dengan interaction lain dan memiliki class sendiri. Class untuk select list adalah org.openqa.selenium.support.ui.Select. Jika elemen lain hanya menggunakan WebElement kemudian menggunakan interaction umum. Select memiliki class sendiri yaitu Select. Class Select memiliki method sendiri yang tidak dimiliki oleh interaction lain. Contoh cara penggunaan class select diberikan pada segmen program 3.49

Segmen Program 3.49 Mendapatkan Attribute dari suatu Elemen

```
1.    WebElement selectElement = driver.findElement(
2.          By.id("selectElementID")
3.    );
4.    Select selectObject = new Select(selectElement);
```

Segmen program 3.49 merupakan program untuk menggunakan class select. Ada tiga cara untuk melakukan select pada option yang terdapat pada elemen select. Cara pertama adalah melakukan select dengan index. Cara kedua adalah melakukan select dengan value. Cara ketiga select dengan text yang terlihat pada select. Contoh untuk melakukan select diberikan pada segmen program 3.50

Segmen Program 3.50 Melakukan select pada suatu option

```
1.    selectObject.selectByIndex(1);
2.    selectObject.selectByValue("value1");
3.    selectObject.selectByVisibleText("Bread");
```

Segmen program 3.50 merupakan contoh program untuk melakukan select pada suatu option yang terdapat pada select. Segmen program 3.50 baris pertama merupakan cara untuk memilih option dengan index 1. Segmen program 3.50 baris kedua merupakan cara untuk memilih option berdasarkan value yang tidak terlihat oleh user. Segmen program 3.50 baris ketiga merupakan cara untuk

memilih option berdasarkan value yang terlihat di select. Berikutnya akan dijelaskan cara untuk cek option yang dipilih pada select box dan mendapatkan semua option yang ada pada suatu select. Contoh program untuk melakukan cek option yang dipilih dan mendapat semua option yang ada pada select box ada pada segmen program 3.51

Segmen Program 3.51 Cek Option yang Dipilih pada Select

```
1.    List<WebElement> all = selectObject.getAllSelectedOptions();
2.    WebElement first = selectObject.getFirstSelectedOption();
3.    List<WebElement> allOptions = selectObject.getOptions();
```

Segmen program 3.51 merupakan cara untuk cek option yang dipilih pada select. Pada segmen program 3.51 terdapat dua cara untuk mendapatkan option yang dipilih pada suatu select. Segmen program 3.51 baris pertama merupakan cara untuk mendapatkan semua option yang di select pada suatu select. Segmen program 3.51 baris kedua merupakan cara untuk mendapatkan option pertama yang di select pada suatu select. Segmen program 3.51 baris ketiga merupakan cara untuk mendapatkan semua option yang terdapat pada suatu select. Selain melakukan select Class Select juga menyediakan cara untuk melakukan deselect. Contoh program untuk melakukan deselect pada select akan diberikan pada segmen program 3.52

Segmen Program 3.52 Cek Option yang Dipilih pada Select

```
1.    selectObject.deselectByIndex(1);
2.    selectObject.deselectByValue("value1");
3.    selectObject.deselectByVisibleText("Bread");
4.    selectObject.deselectAll();
```

Segmen program 3.52 merupakan contoh program untuk melakukan select pada suatu option yang terdapat pada select. Segmen program 3.52 baris pertama merupakan cara untuk deselect option dengan index 1. Segmen program 3.52 baris kedua merupakan cara untuk deselect option berdasarkan value yang tidak terlihat oleh user. Segmen program 3.52 baris ketiga merupakan cara untuk deselect option berdasarkan value yang terlihat di select. Segmen program 3.52 baris keempat merupakan cara untuk melakukan deselect pada semua option yang dipilih.

Select memiliki property yang paling penting yaitu multi select. Tidak semua select bisa dilakukan multi select. Ada select yang hanya bisa melakukan select pada satu buah option. Pengecekan yang untuk mengecek apakah select menerima multi select atau single select dapat dilakukan dengan method isMultiple() pada class select. Jika nilai kembalian dari method bernilai true maka select tersebut bisa menerima multi option. Jika nilai kembalian dari method bernilai false maka select tersebut hanya bisa menerima satu buah pemilihan option.

3. 5 Remote Webdriver

Remote Webdriver mengimplementasikan antarmuka WebDriver untuk menjalankan skrip pengujian melalui server RemoteWebDriver pada mesin jarak jauh. Remote Webdriver digunakan saat user melakukan testing secara remote dan tidak secara local. Perbedaan dari Selenium Remote Webdriver dan Selenium Webdriver hanya terletak pada konfigurasinya. Penggunaan elemen, cara mendapatkan browser data memiliki cara yang sama dengan Selenium Webdriver yang dijalankan secara local. Sebuah perbedaan yang mencolok adalah perbedaan dari penggunaan class dimana Selenium Webdriver menggunakan class org.openqa.Selenium, sedangkan semua fitur yang digunakan pada Selenium Remote Webdriver akan menggunakan class org.openqa.Selenium.remote. Dalam Selenium Remote Webdriver terdiri dari dua bagian yaitu remote dan server.

Pada Selenium Remote Webdriver, hub adalah komputer yang merupakan titik pusat pengujian. Hub juga bertindak sebagai server karena hub bertindak sebagai titik pusat untuk mengontrol jaringan mesin Uji. Selenium Grid hanya memiliki satu hub dan merupakan master jaringan. Ketika tes dengan DesiredCapabilities diberikan ke Hub, Hub mencari node yang cocok dengan konfigurasi yang diberikan.

Untuk menjalankan Selenium Remote Webdriver, pertama-tama user harus terhubung ke RemoteWebDriver. Untuk dapat melakukan tes pertama Selenium Remote Webdriver harus di connect kepada server yang ada. Setelah melakukan connect baru user bisa melakukan test melakukan Selenium Remote

Webdriver. Contoh konfigurasi connect Selenium Remote Webdriver menuju server diberikan pada segmen program 3.53.

Segmen Program 3.53 Konfigurasi Selenium Remote Webdriver

```
1.    FirefoxOptions firefoxOptions = new FirefoxOptions();
2.    WebDriver driver = new RemoteWebDriver(
3.          new URL("http://www.example.com"),
4.          firefoxOptions
5.    );
6.    driver.get("http://www.google.com");
7.    driver.quit();
```

Segmen program 3.53 merupakan cara konfigurasi Selenium Webdriver. Pada segmen program 3.53 Selenium Remote Webdriver terhubung dengan server yang ada pada www.example.com. Setelah melakukan konfigurasi dengan Selenium Remote Webdriver penggunaan berikutnya untuk melakukan test sama dengan saat menggunakan Selenium Webdriver yang ada di local. Sebuah fitur yang hanya ada di Selenium Remote Webdriver adalah local file detector. Contoh penggunaan local file detector diberikan pada segmen program 3.54

Segmen Program 3.53 Konfigurasi Selenium Remote Webdriver

```
1.    driver.setFileDetector(new LocalFileDetector());
2.    driver.get(
3.          "http://sso.dev.saucelabs.com/test/guinea-file-upload"
4.    );
5.    WebElement upload = driver.findElement(By.id("myfile"));
6.    upload.sendKeys("/image/test.jpg");
```

Segmen program 3.53 merupakan contoh program untuk melakukan transfer file dari client ke server. Method yang digunakan untuk melakukan transfer adalah setFileDetector. Cara kerja dari detector file adalah saat tester melakukan upload file ke web, Remote Web Driver akan menstransfer file secara otomatis dari client ke server. Ini memungkinkan file diupload dari remote client.

3. 6 Wait

Perintah wait sangat penting dalam menjalankan tes pada Selenium Webdriver. Perintah wait membantu untuk mengamati dan memecahkan masalah yang mungkin terjadi karena perbedaan jeda waktu. Saat menjalankan tes Selenium, biasanya tester mendapatkan pesan “Element Not Visible Exception“.

Exception ini muncul ketika elemen web tertentu yang berinteraksi dengan Selenium WebDriver, belum selesai dimuat. Untuk mencegah exception ini, Perintah wait Selenium harus digunakan.

Dalam automated testing, wait memerintahkan test langsung untuk berhenti selama beberapa waktu sebelum melanjutkan ke langkah berikutnya. Ini memungkinkan WebDriver untuk memeriksa apakah satu atau lebih elemen web yang dicari bisa di interaksi. Ada tiga jenis wait yang terdapat pada Selenium Webdriver yaitu Implicit Wait, explicit Wait, dan Fluent Wait. Jenis Wait pertama yang akan dibahas adalah Implicit Wait.

Implicit wait merupakan wait dimana Selenium WebDriver mensurvei DOM selama durasi tertentu saat mencoba menemukan suatu elemen. Implicit wait berguna ketika elemen tertentu pada halaman web tidak segera tersedia dan perlu beberapa waktu untuk dimuat. Implicit wait secara default tidak aktif dan harus diaktifkan manual per sesi test. Implicit wait akan memberi tahu Selenium WebDriver untuk melakukan polling DOM selama jangka waktu tertentu saat mencoba menemukan elemen atau elemen jika tidak segera tersedia. Pengaturan default dari implicit wait adalah 0 yang berarti tidak aktif. Setelah digunakan implicit wait akan tetap jalan selama driver belum di tutup. Contoh penggunaan implicit wait akan diberikan pada segmen program 3.54

Segmen Program 3.54 Konfigurasi Implicit Wait

```
1.    WebDriver driver = new FirefoxDriver();
2.    driver.manage().timeouts().implicitlyWait(
3.          Duration.ofSeconds(10)
4.    );
5.    driver.get("http://somedomain/url_that_delays_loading");
6.    WebElement myDynamicElement = driver.findElement(
7.          By.id("myDynamicElement")
8.    );
```

Segmen program 3.54 merupakan konfigurasi untuk penggunaan implicit wait. Pada segmen program 3.54 wait yang digunakan adalah 10 detik. Saat membuka website dan mencari elemen maka Selenium Webdriver akan menunggu hingga 10 detik dan melakukan polling setiap detiknya untuk mengetahui apakah elemen yang dicari sudah terload atau belum. Jenis Wait berikutnya adalah explicit wait. Berbeda dengan implicit wait explicit wait akan

menghentikan jalannya program hingga kondisi terpenuhi. Cara penggunaan explicit wait akan diberikan pada segmen program 3.55

Segmen Program 3.55 Konfigurasi Explicit Wait

```
1.    WebDriver driver = new ChromeDriver();
2.    driver.get("https://google.com/ncr");
3.    driver.findElement(By.name("q")).sendKeys(
4.          "cheese" + Keys.ENTER
5.    );
6.    WebElement firstResult = new WebDriverWait(
7.          driver,
8.          Duration.ofSeconds(10)
9.    ).until(ExpectedConditions.elementToBeClickable
10.         (By.xpath("//a/h3")
11.   ));
12.   System.out.println(firstResult.getText());
```

Segmen program 3.55 merupakan segmen program untuk melakukan explicit wait pada suatu elemen. Dapat dilihat perbedaan pertama dari explicit wait dan implicit wait adalah explicit wait dijalankan setiap elemen bukan setiap sesi. Implicit wakt akan berjalan berulang kali hingga return value sesuai yang diharapkan oleh user. Nilai kembalian adalah segala sesuatu yang dievaluasi menjadi boolean true, seperti string, integer, boolean, objek (termasuk WebElement), atau list. Wait terakhir yang ada pada Selenium Webdriver adalah fluent wait. Contoh program untuk penggunaan fluent wait ada pada segmen program 3.56

Segmen Program 3.56 Konfigurasi Fluent Wait

```
1.    Wait<WebDriver> wait = new FluentWait<WebDriver>(driver)
2.      .withTimeout(Duration.ofSeconds(30))
3.      .pollingEvery(Duration.ofSeconds(5))
4.      .ignoring(NoSuchElementException.class);
5.    WebElement foo = wait.until(
6.          new Function<WebDriver, WebElement>() {
7.                public WebElement apply(WebDriver driver) {
```

Lanjutan Segmen Program 3.56 Konfigurasi Fluent Wait

```
8.    return driver.findElement(By.id("foo"));
9.                }
10.         }
11.     );
```

FluentWait menentukan jumlah waktu maksimum untuk menunggu suatu kondisi, serta berapa kali kondisi tersebut dijalankan. Saat menggunakan Fluent Wait, tester dapat melakukan konfigurasi pada periode polling default sesuai

kebutuhan. Pengguna dapat mengonfigurasi wait untuk mengabaikan exception apa saat periode polling. Exception yang diabaikan secara umum adalah exception NoSuchElementException. Fluent Wait berguna saat berinteraksi dengan web element yang memakan waktu lebih lama untuk di load. Fitur berikutnya yang dimiliki oleh Selenium adalah Action API. Action api merupakan bentuk lanjutan dari Interaction. Action API akan dijelaskan pada bagian 3.7

3. 7 Action API

Action merupakan interface tingkat rendah yang menyediakan input perangkat virtual ke browser web. Tidak seperti interaction pada web elemen Actions API menyediakan kontrol terperinci atas perangkat input. Ada dua jenis input yang terdapat pada action yaitu mouse input dan keyboard input. Action pertama yang akan dibahas adalah action yang dimiliki oleh mouse. Secara detail action yang dimiliki oleh mouse input akan diberikan pada bagian 3.7.1

3.7.1 Mouse Actions

Mouse action merupakan interaction yang mensimulasikan mouse input yang diberikan oleh Selenium Webdriver. Mouse action berbeda dengan interaction mouse yang ada pada Selenium Webdriver. Pada mouse interaction, interaction yang dimiliki hanya click. Pada Action API mouse click akan diberikan lebih detail dan tidak hanya melakukan click tetapi ada click and hold, context click, double click, drag and drop, dan drag and drop by. Action yang akan dibahas pertama adalah click and hold. Contoh program penggunaan click and hold akan diberikan pada segmen program 3.57.

Segmen Program 3.57 Click and Hold

```
1.    WebElement searchBtn = driver.findElement(By.Id("login"));
2.    Actions actionProvider = new Actions(driver);
3.    actionProvider.clickAndHold(searchBtn).build().perform();
```

Segmen program 3.57 merupakan segmen program untuk melakukan click and hold pada elemen. Method click and hold akan melakukan click pada button kemudian click tersebut akan ditahan. Pada segmen program 3.57 dapat dilihat

contoh penggunaan action api. Jika Interaction biasa langsung digunakan pada web element. Action dibuild pada Class Actions kemudian method dari action yang dipilih dijalankan bukan melalui web elemen tetapi dijalankan pada class action tersebut. Action berikutnya yang akan dibahas adalah context click. Contoh program untuk context click akan diberikan pada segmen program 3.58

Segmen Program 3.58 Context Click

```
1.    WebElement searchBtn = driver.findElement(By.Id("login"));
2.    Actions actionProvider = new Actions(driver);
3.    actionProvider.contextClick(searchBtn).build().perform();
```

Segmen program 3.58 merupakan contoh program untuk melakukan context click. Context click merupakan method yang digunakan untuk melakukan simulasi klik kanan pada suatu elemen. Action yang akan dibahas berikutnya adalah double click. Contoh program untuk melakukan double click akan diberikan pada segmen program 3.59.

Segmen Program 3.59 Double Click

```
1.    WebElement searchBtn = driver.findElement(By.Id("login"));
2.    Actions actionProvider = new Actions(driver);
3.    actionProvider.doubleClick(searchBtn).build().perform();
```

Segmen program 3.59 merupakan contoh program untuk melakukan double click pada elemen. Berikutnya akan dijelaskan mengenai moveTo. Move to merupakan action yang memindahkan mouse pada bagian tengah dari elemen yang dipilih. Contoh program untuk moveTo akan diberikan pada segmen program 3.60.

Segmen Program 3.60 Move To

```
1.    WebElement searchBtn = driver.findElement(By.Id("login"));
2.    Actions actionProvider = new Actions(driver);
3.    actionProvider. moveToElement(searchBtn).build().perform();
```

Segmen program 3.60 merupakan contoh program untuk melakukan action move to. Action move to merupakan action yang digunakan untuk memindahkan mouse berada pada bagian tengah elemen. Cara kerja dari action move to adalah Selenium akan mencari elemen kemudian Action akan memindahkan cursor ke tengah dari elemen yang telah dicari tadi. Berikutnya akan dijelaskan action move

by offset. Contoh program untuk move by offset akan diberikan pada segmen program 3.61

Segmen Program 3.61 Move By Offset

```
1.    WebElement gmailLink = driver.findElement(By.Id("input-1"));
2.    int xOffset = gmailLink.getRect().getX();
3.    int yOffset = gmailLink.getRect().getY();
4.    Actions actionProvider = new Actions(driver);
5.    actionProvider.moveByOffset(xOffset, yOffset)
6.    .build().perform();
```

Segmen program 3.61 merupakan contoh program untuk melakukan move by offset. Move by offset merupakan action yang digunakan untuk memindahkan cursor pada suatu titik koordinat. Ada dua buah parameter yang dibutuhkan oleh method moveByOffset yaitu x dan y. Nilai x dan y didapatkan dengan menggunakan method getRect() pada WebElement. Nilai x dan y yang didapatkan dijadikan parameter untuk method moveByOffset. Berikutnya akan dibahas mengenai action drag and drop. Contoh program untuk method drag and drop akan diberikan pada segmen program 3.62

Segmen Program 3.62 Drag And Drop

```
1.    WebElement start = driver.findElement(By.id("start"));
2.    WebElement target = driver.findElement(By.id("drop"));
3.    Actions actionProvider = new Actions(driver);
4.    actionProvider.dragAndDrop(start, target).build().perform();
```

Segmen program 3.62 merupakan contoh program untuk melakukan drag and drop. Untuk melakukan drag and drop, method click and hold tidak diperlukan karena method drag and drop sudah termasuk melakukan click and hold. Ada dua parameter pada method drag and drop yaitu start dan target. Start merupakan lokasi awal elemen yang akan di drag and drop. Target merupakan tempat elemen start yang di drag tadi di drop. Berikutnya akan dibahas mengenai drag and drop by. Contoh program untuk drag and drop by akan diberikan pada segmen program 3.63

Segmen Program 3.63 Drag And Drop by

```
1.    WebElement start = driver.findElement(By.id("start"));
2.    WebElement target = driver.findElement(By.id("drop"));
3.    int targetEleXOffset = target.getLocation().getX();
4.    int targetEleYOffset = target.getLocation().getY();
```

```
5.    Actions actionProvider = new Actions(driver);
6.    actionProvider.dragAndDropBy
7.    (start, targetEleXOffset, targetEleYOffset)
8.    .build().perform();
```

Segmen program 3.63 merupakan contoh program untuk melakukan drag and drop by. Untuk melakukan drag and drop by, method click and hold tidak diperlukan karena method drag and drop by sudah termasuk melakukan click and hold. Perbedaan method drag and drop by dengan method drag and drop adalah pada parameter yang ada di method. Ada tiga parameter pada method drag and drop by yaitu start, targetX, dan targetY. Start merupakan lokasi awal elemen yang akan di drag and drop. TargetX, dan target merupakan koordinat x, dan y dari elemen tempat elemen start akan di drop. Method terakhir yang dimiliki oleh mouse action adalah release. Contoh program untuk melakukan release akan diberikan pada segmen program 3.64

Segmen Program 3.64 Release

```
1.    WebElement start = driver.findElement(By.id("draggable"));
2.    WebElement target = driver.findElement(By.id("droppable"));
3.    Actions actionProvider = new Actions(driver);
4.    actionProvider.clickAndHold(start).moveToElement(target)
5.    .build().perform();
6.    actionProvider.release().build().perform();
```

Segmen program 3.64 merupakan contoh program untuk melakukan action release. Action release merupakan action yang digunakan untuk melepaskan klik kiri dari mouse jika ada. Pada segmen program 3.64 Action release akan melepaskan klik yang dilakukan oleh method click and hold. Berikutnya akan dijelaskan mengenai Action yang terdapat pada keyboard. Secara detail action pada keyboard akan dibahas pada subbab 3.7.2.

3.7.2 Keyboard Actions

Keyboard action merupakan interaction yang mensimulasikan input oleh keyboard yang diberikan kepada browser oleh Selenium Webdriver. Keyboard action berbeda dengan interaction keyboard yang ada pada Selenium Webdriver. Pada keyboard interaction, interaction yang dimiliki hanya sendKeys. Pada Action

API sendKeys akan dipecah menjadi dua yaitu key up dan key down. Action yang akan dibahas pertama adalah key down. Contoh program penggunaan key down akan diberikan pada segmen program 3.65.

Segmen Program 3.65 Key Down

```
1.    Actions actionProvider = new Actions(driver);
2.    Action keydown = actionProvider.keyDown
3.    (Keys.CONTROL).sendKeys("a").build();
4.    keydown.perform();
```

Segmen program 3.65 merupakan contoh program untuk melakukan key down. Key down merupakan action yang digunakan untuk melakukan hold pada key yang dipilih. Pada segmen program 3.65 program akan melakukan hold control dan menekan tombol a secara bersamaan. Berikutnya akan dijelaskan mengenai key Up. Contoh program untuk penggunaan key up akan diberikan pada segmen program 3.66

Segmen Program 3.66 Key Up

```
1.    Actions action = new Actions(driver);
2.    WebElement search = driver.findElement(By.name("q"));
3.    action.keyDown(Keys.SHIFT).sendKeys(search,"ini")
4.    .keyUp(Keys.SHIFT)
5.    .sendKeys(" coba").perform();
```

Segmen program 3.66 merupakan contoh program untuk melakukan key up. Key up merupakan kebalikan dari key down. Key up merupakan action yang digunakan untuk melepaskan hold dari key yang ditahan tadi. Pada segmen program 3.66 input search akan berisi dengan tulisan INI coba. Tulisan ini bisa berubah menjadi huruf kapital karena shift ditahan dan keys yang dikirim secara otomatis berubah menjadi huruf kapital. Text coba tidak lagi menjadi kapital karena sebelum key coba dikirim key shift yang tadi ditahan telah dilepaskan menggunakan method key Up.