

Selenium 4 New Feature How To Handle Windows



Table of Contents

New Window/New Tab.....	2
Concept of Opening A New Window/Tab.....	2
Demo.....	3
Open, Switch & Work In New Tab.....	6
Minimize Window.....	8
Contact Info.....	13

New Window/New Tab

Selenium 4 allows us to open a new window and open a new tab in the same session at the same time. After opening the window or tab, we have the ability to work in it without creating a new driver.

Concept of Opening A New Window/Tab

In this video, we are going to look at the concept of opening a new window and a new tab. With previous versions of Selenium, we could use the Robot Class to simulate the keyboard and open a new tab by pressing CTRL + T then switching focus to the tab after getting the window handle. Now, we can perform those same steps with Selenium 4 using 1 line of code: We can use it for a window or tab.

`driver.switchTo().newWindow(WindowType.WINDOW or TAB).get(url);`

New Window / New Tab Syntax

```
driver.switchTo().newWindow(WindowType.WINDOW).get(url);
```

```
driver.switchTo().newWindow(WindowType.TAB).get(url);
```

Notice, WINDOW and TAB are written in green.

- driver is used to control the browser
- switchTo() is a method that sends commands to a window. It switches focus between the windows
- newWindow() is a method that creates a new window or tab then automatically switches focus to that new window or tab
- WindowType is an enum. Enum is short for enumeration which is a list of constants that define a new data type. In this case, the constants are WINDOW and TAB that instructs our program to open a new window or new tab.
- WINDOW is the parameter for creating a new window
- TAB is also a parameter but it's used for creating a new tab
- Last is the get(url) method which loads a new page in our browser

New Window / New Tab Syntax

```
driver.switchTo().newWindow(WindowType.WINDOW).get(url);
```

```
driver.switchTo().newWindow(WindowType.TAB).get(url);
```

Argument	Description
driver	Controls the browser
switchTo()	Sends commands to a window
newWindow()	Create a new browser window or tab then automatically switch focus
WindowType	An enum containing a list of constants (WINDOW and TAB)
WINDOW	A type hint parameter that creates a new window
TAB	A type hint parameter that creates a new tab
get(url)	Loads a new page in the browser

Demo

For our Test Script, we have a set up method to load an Automation Practice website then get the title. I commented out the tear down method because I want the page to stay open after executing our Test Script.


```
@BeforeMethod
public void setUp () {
    WebDriverManager.chromedriver().setup();
    driver = new ChromeDriver();
    driver.manage().window().maximize();
    driver.get("http://automationpractice.com/index.php");
    System.out.println("Title: " + driver.getTitle());
}
```

```
@AfterMethod
public void tearDown () {
    //driver.quit();
}
```

Let's start the test by writing @Test / public void testNewWindowConcept () {}
driver.switchTo().newWindow(WindowType.TAB);

```
@Test
public void testNewWindowConcept () {
    driver.switchTo().newWindow(WindowType.TAB);
}
```

If I hover the newWindow() method, and the description states "Creates a new browser window and switches the focus for future commands of this driver to the new window". In shorter words, it will open a new window then switch focus to that window. We also see the parameters are type hint which is a type of new browser window to be created. The type hint will be *WINDOW* or *TAB*. Returns this driver focused on the given window. When it says window, it's referring to a window or a tab. That's why I was able to write WindowType.TAB. One more point, do you see how the Return Type is WebDriver?

 **WebDriver** org.openqa.selenium.WebDriver.TargetLocator.newWindow(WindowType typeHint)

Creates a new browser window and switches the focus for future commands of this driver to the new window.

See [W3C WebDriver specification](#) for more details.

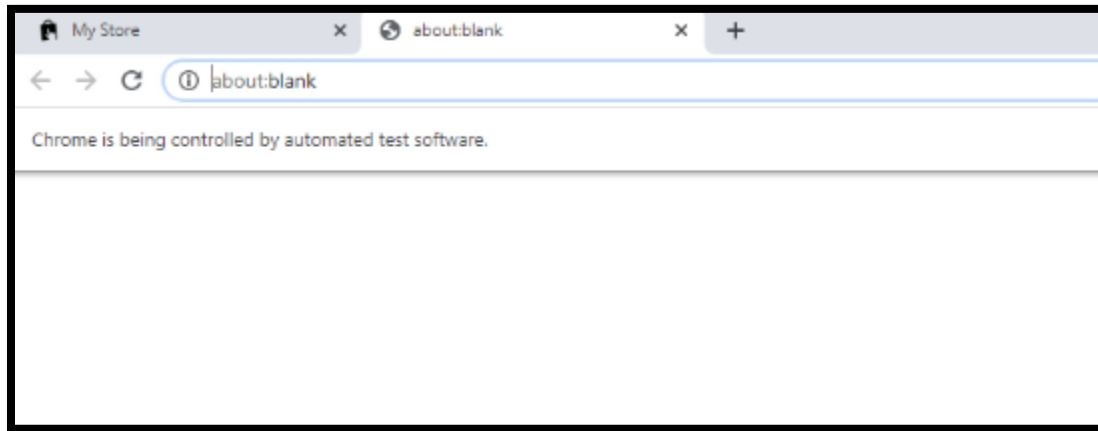
Parameters:

typeHint The type of new browser window to be created. The created window is not guaranteed to be of the requested type; if the driver does not support the requested type, a new browser window will be created of whatever type the driver does support.

Returns:

This driver focused on the given window

If we choose to, we can assign this statement to WebDriver with any name like newPage =. Let's Run.
We have 2 tabs and a blank page is opened with automatic focus in the new tab.



The same outcome happens with a new window. Change TAB to WINDOW then run. A blank page is opened in a new window. For our Test Script, after the blank page, we are going to open the Contact Us page. Let me grab this URL and load a new page by writing `newPage.get("Paste the URL")` and print the page title `sysout ("Title: " + driver.getTitle())`.

```
@Test
Run | Debug
public void testNewWindowConcept () {
    WebDriver newPage = driver.switchTo().newWindow(WindowType.WINDOW);
    newPage.get("http://automationpractice.com/index.php?controller=contact");
    System.out.println("Title: " + driver.getTitle());
}
```

Let's Run. Both titles are displayed in the Console: My Store and Contact Us – My Store.

```
INFO: Detected dialect: W3C
Title: My Store
Title: Contact us - My Store
PASSED: testNewWindowConcept
```

Next, I will show you how to open, switch, work in the new tab then switch back to the parent tab.

Open, Switch & Work In New Tab

Our steps for opening, switching, working, and switching back to the parent is to access this Home page of this Automation Practice site then open the Sign In page in the new tab. On the Sign In page, we will work in this tab by entering an email address then clicking the Create An Account button. We are not going to perform any actions to create an account but switch back to the parent tab and start shopping.

Let's start our test by writing `@Test / public void testOpenSwitchWorkInNewTab () {}` then some comments as our steps. `// Open & Switch To New Window-Tab`
`// Work In The New Window-Tab // Get The Window ID Handles // Switch Back To The Parent Window-Tab & Work In The 1st Window-Tab.`

```
@Test
Run | Debug
public void testOpenSwitchWorkInNewTab () {

    // Open & Switch To New Window-Tab

    // Work In The New Window-Tab

    // Get The Window ID Handles

    // Switch Back To The Parent Window-Tab & Work In The 1st Window-Tab
}
```

First Step: We are going to Open & Switch To The New Window-Tab, we write `driver.switchTo().newWindow(WindowType.TAB).get("http://automationpractice.com/index.php?controller=authentication&back=my-account");` Paste the URL. Since this URL is so long, we can place the get method on the next line my hitting enter after the dot. You can also end the previous line with a semi-colon and start the next line with `driver.get`. However, I'm going to remove the semi-colon and `driver`. It's still long. But let's print the title `sysout("Title: " + driver.getTitle());`

```
// Open & Switch To New Window-Tab
driver.switchTo().newWindow(WindowType.TAB).
get("http://automationpractice.com/index.php?controller=authentication&back=my-account");
System.out.println("Title: " + driver.getTitle());
```

Now, let's work in the new tab. Go back to the AUT and inspect both elements. Email address has `email_create` as the id value and the button has `SubmitCreate` as the id value. Capital S and Capital C. Okay, for the email we write

driver.findElement(By.id("email_create")).sendKeys("Automation34@Selenium4.com"); then driver.findElement(By.id("SubmitCreate")).click(); for the button.

```
// Work In The New Window-Tab
driver.findElement(By.id("email_create")).sendKeys("Automation34@Selenium4.com");
driver.findElement(By.id("SubmitCreate")).click();
```

We are finished working in the new tab. At this point, you can decide to close the new tab or leave it open. If you close the tab, make sure to use driver.close and not driver.quit. driver.close will close the new window but driver.quit will quit the driver and close both windows. [Video 18](#) that will show you the difference between close and quit.

Now, let's get the Window Handles of each tab. The Window Handle is an alphanumeric id assigned to each window or each tab. Set <String> allWindows = driver.getWindowHandles() and we 2 methods. The getWindowHandle method returns 1 window id handle. getWindowHandles method return a set of window handles that can be used to iterate over all open windows. Select this method. The next method is to return an Iterator for the collection of both Window ID's. <String> iterate = allWindows.iterator(); We need a way to iterate the collection of elements and next() is a method for returning an element. String parentFirstWindow = is the parent tab and first element in the iteration. iterate dot is the next() element which is String childSecondWindow = and that window is the new tab we are going to open. For this example, that's all we need to get the ID's and iterate the ID's. You can also loop through the windows.

```
// Get The Window ID Handles
Set <String> allWindows = driver.getWindowHandles();
Iterator<String> iterate = allWindows.iterator();
String parentFirstWindow = iterate.next();
String childSecondWindow = iterate.next();
```

The last step is to switch back to the parent. Let's write driver.switchTo().window. Let me explain this part. Do you see window and newWindow? Both methods involve switching focus to a window but the window() method deals with handling a window and the newWindow() method opens a window. The window() method receives a name or handle. We get the name or handle information after using the getWindowHandle method or getWindowHandles method. There's a difference between handling the window and opening the window. I created [video 71](#) that provide more details on how to handle the windows. In that video, I used all 3 methods: getWindowHandle, getWindowHandles, and driver.switchTo().window().

Right now, we are opening a new window then working in the window which is a benefit because it allows us to multi-task in a different tab. Before Selenium 4, we could not multi-task but had to perform a different task in the same tab. Select window then pass in parentFirstWindow because we are switching back to the parent. Let's go to the AUT and inspect the Search box. We see id is search_query_top and the search icon has no id value but a value of submit_search for name.

Let's enter some data by writing driver.findElement(By.id("search_query_top")).sendKeys("Faded Shirt"); and click the search icon by writing driver.findElement(By.name("submit_search")).click(); Also print the page title.

sysout("Title: " + driver.getTitle());

```
// Switch Back To The Parent Window-Tab & Work In The 1st Window-Tab
driver.switchTo().window(parentFirstWindow);
driver.findElement(By.id("search_query_top")).sendKeys("Faded Shirt");
driver.findElement(By.name("submit_search")).click();
System.out.println("Title: " + driver.getTitle());
```

Let's Run. The reason why I like to print the title because I want to make sure we can see the page title for each page in the console. We see both tabs had some kind of information. The parent tab has Faded Shirt and the new tab is ready to create an account with our Email Address already populated.

Let's see if the console shows each page title. The console shows all 3 titles: My Store, Login – My Store, and Search – My Store. Bingo – Thanks for watching.

Minimize Window

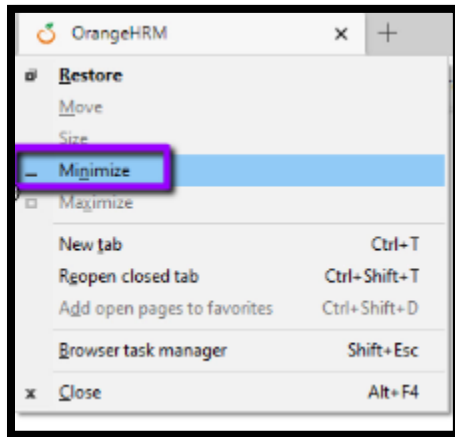
In this session, let's look at a new Selenium 4 feature to minimize the browser. We could always minimize the browser but this feature is built into Selenium. The other ways were to set the position of a window and to use the Robot Class. Personally, I like to watch the automation of our Test Scripts in a browser but if you have a requirement for minimizing the browser then this will help you out.

Also, if you are interested in more videos, you can subscribe to my YouTube channel and click the bell icon. Also, follow me on Twitter, connect with me on LinkedIn and Facebook.

We are going to use this Orange HRM site by entering the username, password, and clicking the button.

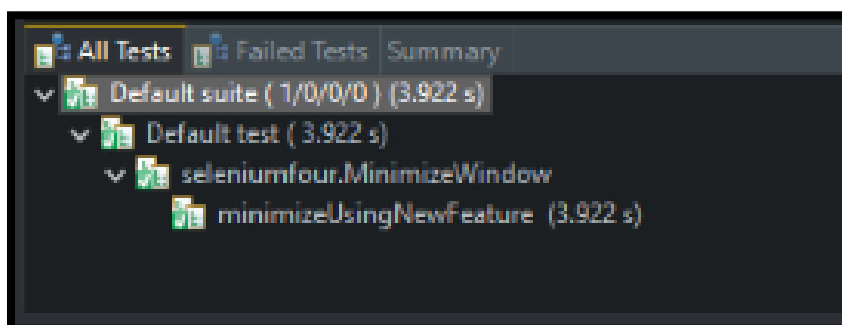


Username is Admin and Password is admin123. Inspect the Username and it shows txtUsername as the value for id, inspect the Password and it shows txtPassword as the value for id. Now, let's inspect the button and it shows btnLogin as the value for id. To use the Robot Class, we select ALT + SPACE + N and do you see Minimize? That's how we minimize using the Robot class.



Oh, by the way, this minimize feature became available in Selenium 4 Alpha 5. So far, this Test Script is set up for ChromeDriver and to load our AUT. Now, we write @Test
 public void minimizeUsingNewFeature () {}
 Import the @Test annotation from TestNG. Now, we write driver.manage().window(). Normally we select maximize but here's minimize(). Enter the username
 driver.findElement(By.id("txtUsername")).sendKeys("Admin");
 Enter the password driver.findElement(By.id("txtPassword")).sendKeys("admin123");
 Next is to click the button driver.findElement(By.id("btnLogin")).click(); Let's Run. It passed in 3.922 seconds.

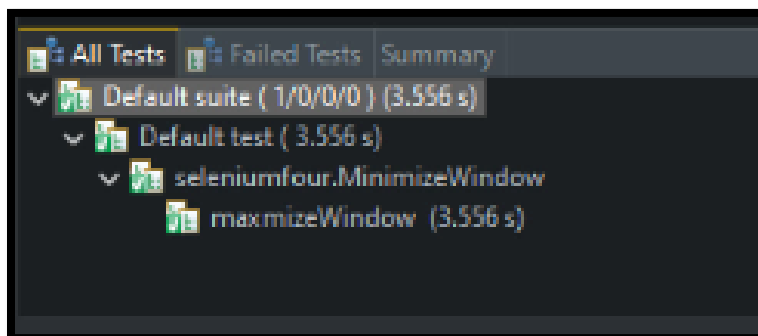
```
@Test
Run | Debug
public void minimizeUsingNewFeature () {
    driver.manage().window().minimize();
    driver.findElement(By.id("txtUsername")).sendKeys("Admin");
    driver.findElement(By.id("txtPassword")).sendKeys("admin123");
    driver.findElement(By.id("btnLogin")).click();
}
```



I've seen it runs faster with minimize and also seen it run faster with maximize. Let's run it with maximize and see what happens. @Test

public void maximizeWindow () {} Copy and Paste this code from minimize and change to minimize to maximize. Run. Maximize executed in 3.556 seconds.

```
@Test
Run | Debug
public void maximizeWindow () {
    driver.manage().window().maximize();
    driver.findElement(By.id("txtUsername")).sendKeys("Admin");
    driver.findElement(By.id("txtPassword")).sendKeys("admin123");
    driver.findElement(By.id("btnLogin")).click();
}
```



Now, let's minimize the browser by setting the position and using the Robot class

Starting with set position, we write @Test public void minimizeUsingSetPosition () {}
 driver.manage().window().setPosition(new Point(-2000, 0)); Import Point from the Selenium package.
 Point is a Selenium class that's a copy of Java's awt.Point package. It removed the dependency of awt.
 Copy and paste the same code. We're finished with the Set Position test.

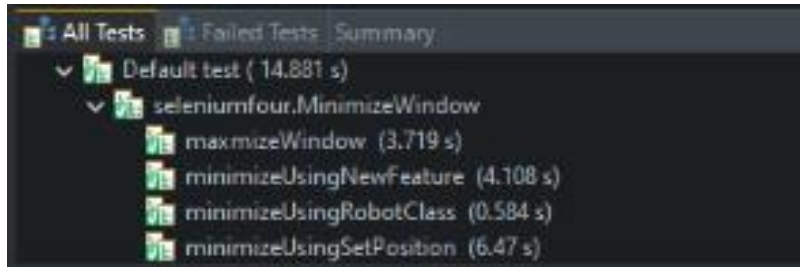
```
@Test
Run | Debug
public void minimizeUsingSetPosition () {
    driver.manage().window().setPosition(new Point(-2000, 0));
    driver.findElement(By.id("txtUsername")).sendKeys("Admin");
    driver.findElement(By.id("txtPassword")).sendKeys("admin123");
    driver.findElement(By.id("btnLogin")).click();
}
```

Next is the Robot class. @Test public void minimizeUsingRobotClass () {} Robot robot = new Robot (); Import Robot from java.awt then add throws declaration – select AWTException but we can also select Exception. First, we press the ALT Key by entering robot.keyPress(KeyEvent.VK_ALT); Now, let's press the SPACE key. robot.keyPress(KeyEvent.VK_SPACE); Next is the N key robot.keyPress(KeyEvent.VK_N); We have to also release the keys so I'm going to copy these 3 statements, paste them and change Press to Release. For the last time, change Press to Release.

```
@Test
Run | Debug
public void minimizeUsingRobotClass () throws AWTException {
    Robot robot = new Robot ();
    robot.keyPress(KeyEvent.VK_ALT);
    robot.keyPress(KeyEvent.VK_SPACE);
    robot.keyPress(KeyEvent.VK_N);

    robot.keyRelease(KeyEvent.VK_ALT);
    robot.keyRelease(KeyEvent.VK_SPACE);
    robot.keyRelease(KeyEvent.VK_N);
}
```

Now, let's run all 4 of the methods: minimizUsingRobotClass, minimizeUsingSetPosition, maximizeWindow, and minimizeUsingNewFeature. Run All. We see the time for each Test Script and it looks like Robot class was the fastest. This time it showed maximize 2nd, minimize 3rd, and minimize using set position as the last for as speed Thanks for watching and I'll see you in the next session.



Contact Info

- ✓ Email Rex.Jones@Test4Success.org
- ✓ YouTube <https://www.youtube.com/c/RexJonesII/videos>
- ✓ Facebook <http://facebook.com/JonesRexII>
- ✓ Twitter <https://twitter.com/RexJonesII>
- ✓ GitHub <https://github.com/RexJonesII/Free-Videos>
- ✓ LinkedIn <https://www.linkedin.com/in/rexjones34/>