

While automating any website or a web application, we must have witnessed a scenario where multiple windows open within an application when a button is clicked and the user has to perform some action on all the opened windows. Now user might not be able to work on all windows at the same time and hence need some mechanism through which he can take control over parent and child windows or if I may say from a QA's perspective, how can we handle window using Selenium?. In this tutorial, we will be understanding how this can be achieved with Selenium WebDriver. We will be focusing on the following key topics:

What is a window in Selenium?

How do we identify parent window and child windows?

Why do we need to handle multiple windows in Selenium?

What is a window handle in Selenium?

What are the different methods used for window handling in Selenium?

How do we handle child windows in Selenium?

How to handle multiple windows in Selenium?

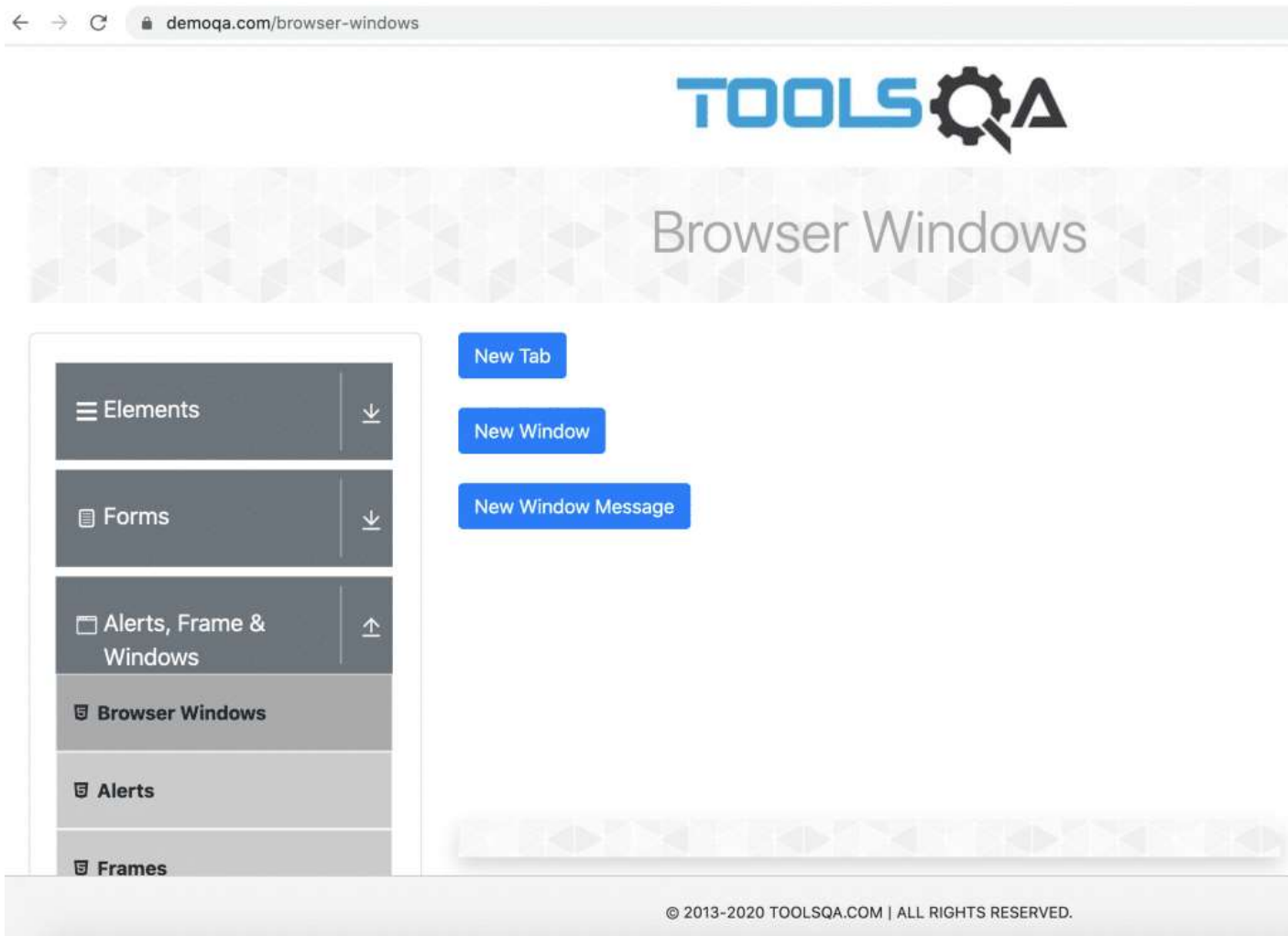
Also, how do we switch back to the parent window from the child windows in Selenium?

How to close all windows in Selenium?

What is a window in Selenium?

A window in any browser is the main webpage on which the user is landed after hitting a link/URL. Such a window in ***Selenium*** is referred to as the ***parent window*** also known as the ***main window*** which opens when the *Selenium WebDriver* session is created and has all the focus of the *WebDriver*.

To view an example of how the *main window* looks like you can visit the ***ToolsQA demo site***, and check. The same is shown in the screenshot below:

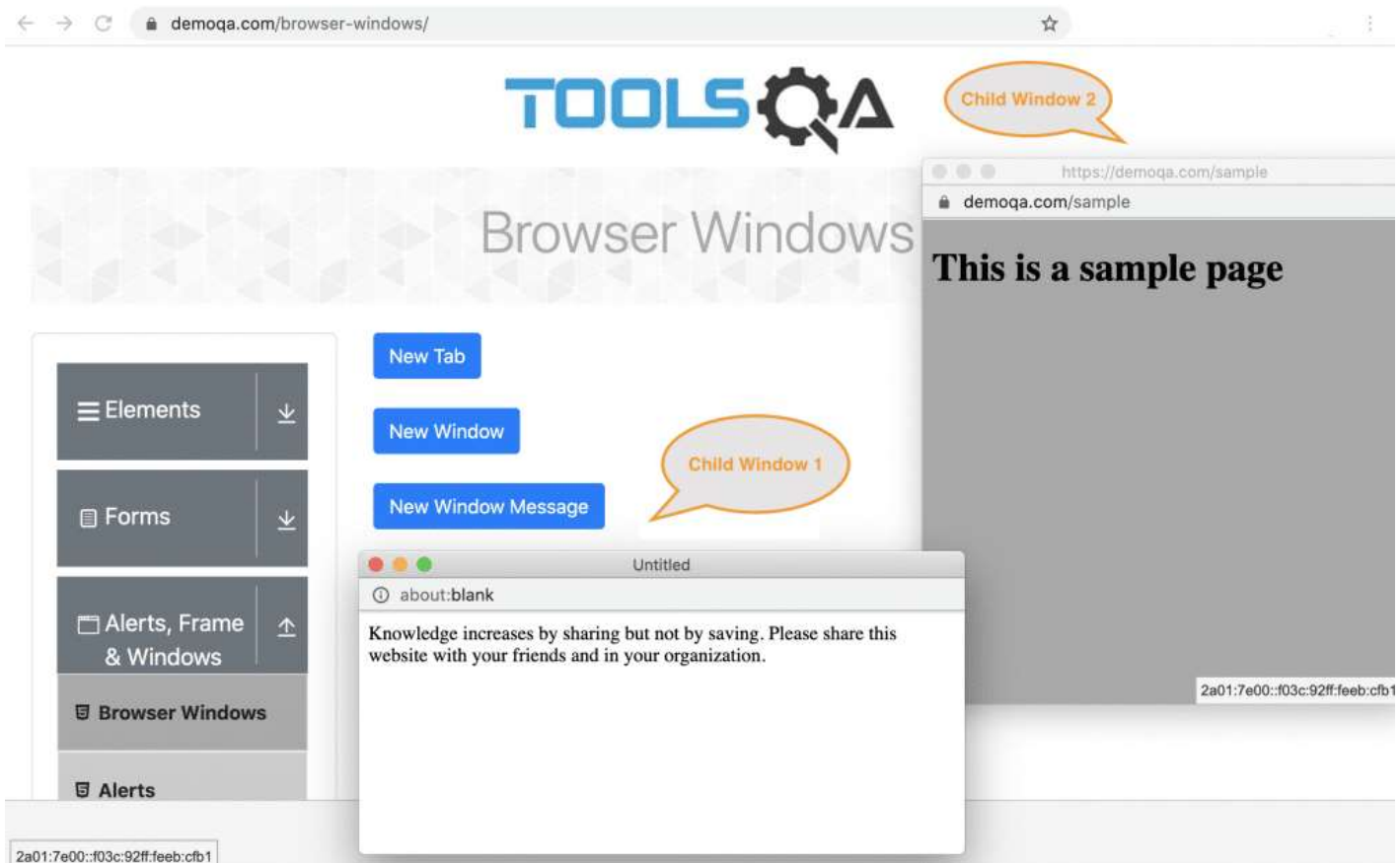


As you can see in the image above, even though the web page has multiple elements, but all of them are part of the main window. The *URL* navigated using the *Selenium WebDriver* will always have the context of the main window. But when we click on the buttons "**New Window**" and "**New Window Message**", it will open other windows, within the parent window. These windows are **Child Windows**. Let's understand how to locate the parent and child windows?

How do we identify parent window and child windows?

When a user hits a URL, a webpage opens. This main page is the **parent window** i.e the main window on which the user has currently landed and will perform any operation. This is the same webpage that will open when our *Selenium* automation script will execute. All the windows which will open inside your main window will be termed as **child windows**.

Taking an example of [ToolsQA demo site](#), the main page which has elements like the **new tab**, **new window**, etc is our parent window and the two sub-windows displayed are the **child windows** that will open when we click on "**New Window**" and "**New Window Message**" buttons, as shown below:



Note: There can be a single child window or multiple child windows in your parent window.

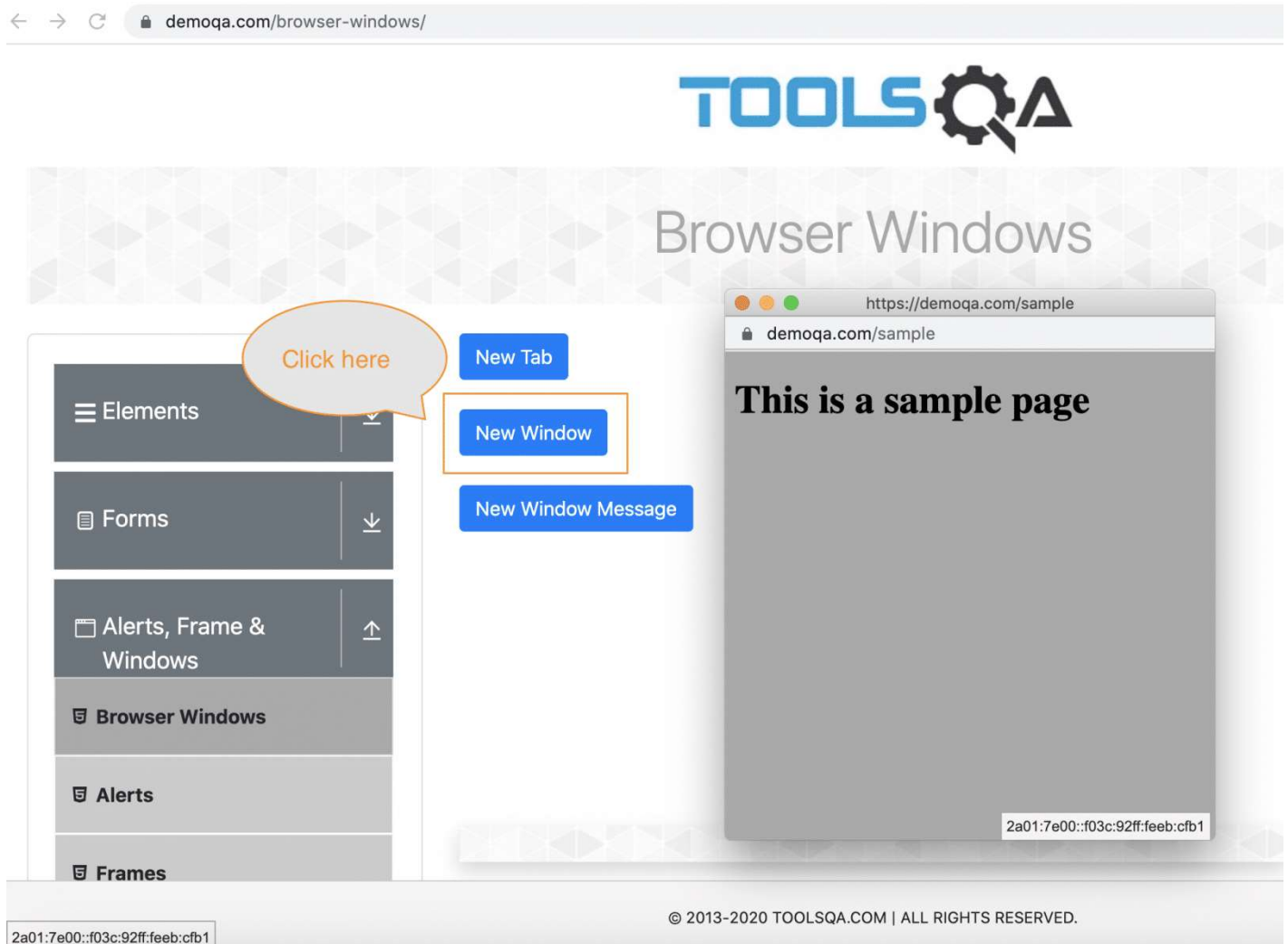
A child window may or may not have any URL. As shown above, *Child Window 1* doesn't have any explicit URL, whereas *Child Window 2* has an explicit URL.

So, when we are testing a web application manually, it is very easy to check the behavior of child windows, as they are easily visible in the context of the main window. But the same is not the case while automating using *Selenium*. Let's understand what is the need to handle the different window types when automating the application using *Selenium WebDriver*?

Why do we need to handle multiple windows in Selenium?

When a user is working on a web application, there might arise a scenario where a new window will open inside your main window. Consider a scenario of an e-commerce site selling garments that will have a size chart linked against each garment and upon clicking, a child window opens. Now, as we understand that *Selenium* works in a specific context only, and each of the child windows will have a separate context. So, for automating such scenarios using *Selenium WebDriver*, we have to direct the *WebDriver* accordingly, so as *Selenium* can get the context of the specific window and perform the needed actions in that window.

Let's try to understand the concept of the *different contexts for each window* with the help of automating the scenario on - "<https://demoqa.com/browser-windows>". After opening the *URL*, we will click on the "**New window**" button within the application, a new browser window opens. We will read the text from the newly opened window i.e. "**This is sample page**" and will print it.



The below code-snippet will click on the "**New Window**" button and will try to read the text from the new Window.

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
public class childWindow {
    public static void main(String[] args) throws InterruptedException {
        System.setProperty("webdriver.chrome.driver", "./src/resources/chromedriver");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://demoqa.com/browser-windows");

        // Open new window by clicking the button
        driver.findElement(By.id("windowButton")).click();
    }
}
```

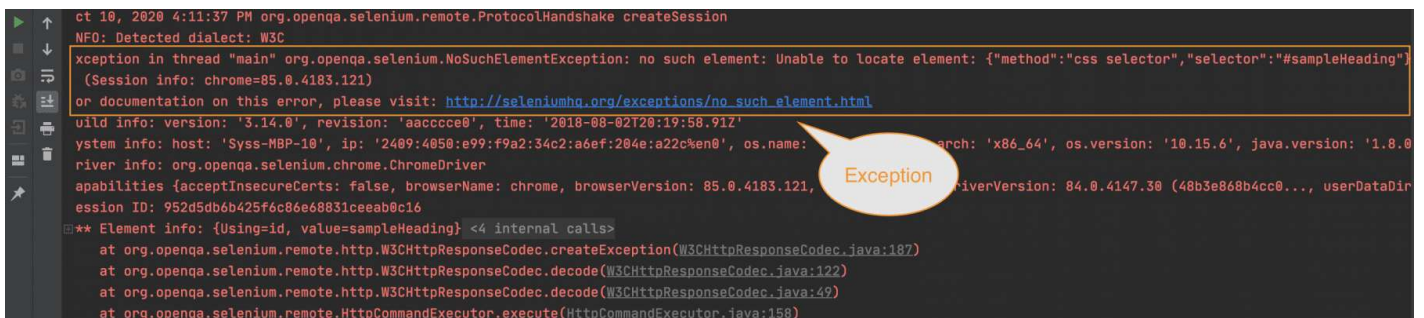
```

// Click on the new window element and read the text displayed on the window
WebElement text = driver.findElement(By.id("sampleHeading"));

// Fetching the text using method and printing it
System.out.println("Element found using text: " + text.getText());
driver.quit();
}
}

```

Output : Once we execute the code, an **"no such element: Unable to locate element "** error is encountered as the *WebDriver* is not able to locate the element and get the text which is displayed on the new window.



```

ct 10, 2020 4:11:37 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
Exception in thread "main" org.openqa.selenium.NoSuchElementException: no such element: Unable to locate element: {"method":"css selector","selector":"#sampleHeading"}
(Session info: chrome=85.0.4183.121)
or documentation on this error, please visit: http://seleniumhq.org/exceptions/no_such_element.html
Build info: version: '3.14.0', revision: 'aacc0ce0', time: '2018-08-02T20:19:58.91Z'
System info: host: 'Syss-MBP-10', ip: '2409:4050:e99:f9a2:34c2:a6ef:204e:a22c%en0', os.name: 'Mac OS X', arch: 'x86_64', os.version: '10.15.6', java.version: '1.8.0_212'
Driver info: org.openqa.selenium.chrome.ChromeDriver
Capabilities {acceptInsecureCerts: false, browserName: chrome, browserVersion: 85.0.4183.121, javascriptEnabled: true, platform: MAC, platformVersion: 10.15.6, driverVersion: 84.0.4147.30 (48b3e868b4cc0..., userDataDir: /tmp/.org.chromium.Chromium.952d5db6b425f6c86e68831ceeab0c16)
** Element info: {Using=id, value=sampleHeading} <4 internal calls>
at org.openqa.selenium.remote.http.W3CHttpResponseCodec.createException(W3CHttpResponseCodec.java:187)
at org.openqa.selenium.remote.http.W3CHttpResponseCodec.decode(W3CHttpResponseCodec.java:122)
at org.openqa.selenium.remote.http.W3CHttpResponseCodec.decode(W3CHttpResponseCodec.java:49)
at org.openqa.selenium.remote.HttpCommandExecutor.execute(HttpCommandExecutor.java:158)

```

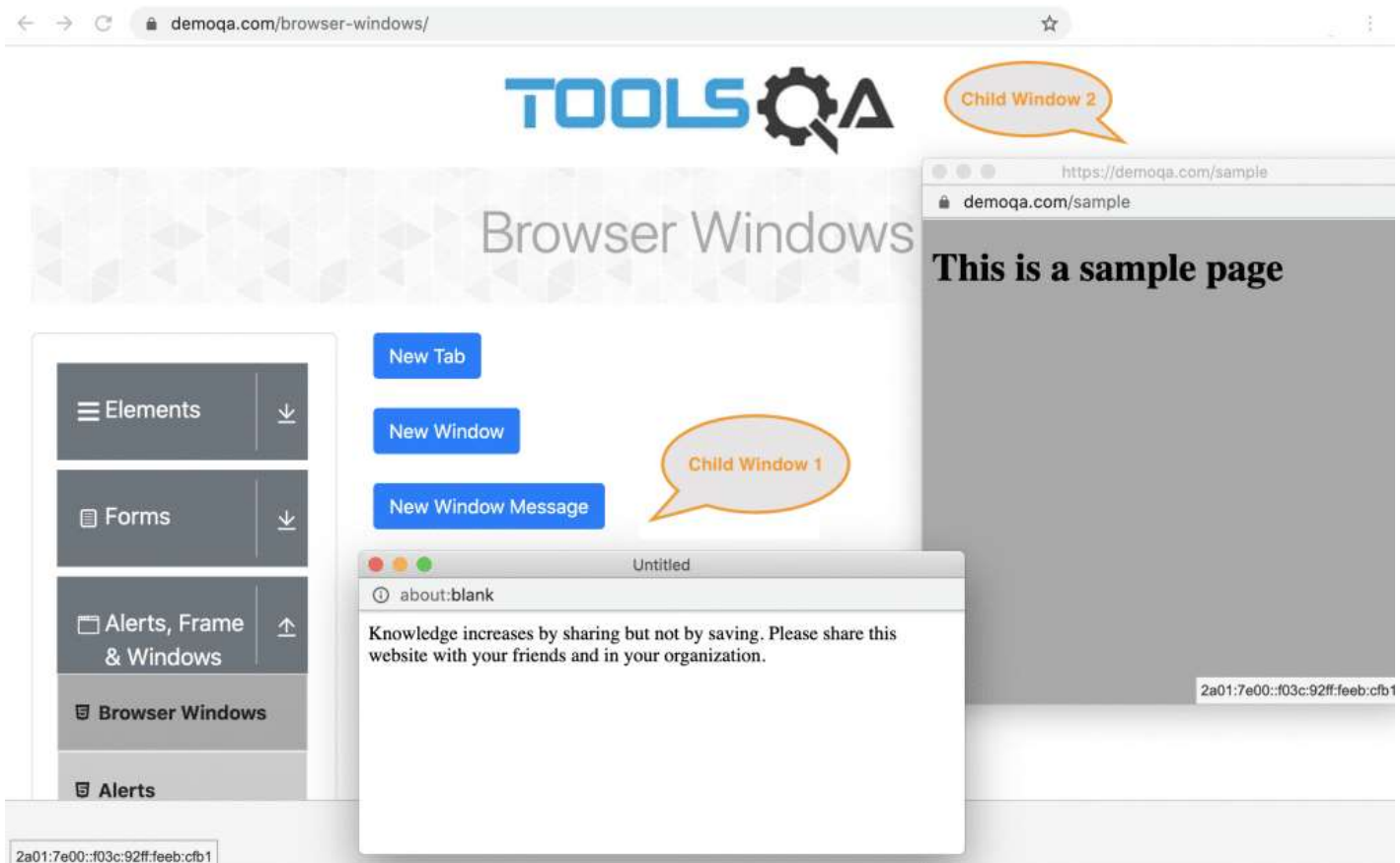
As we can see, even though the text was visible on the screen, but *Selenium WebDriver* was not able to locate it, as it was not in the same context in which *Selenium WebDriver* was executing.

For switching the context, *Selenium WebDriver* makes use of a specific ID of the window, known as the handle of the window. Let's understand what exactly is a window handle in the *Selenium context*?

What is a window handle in Selenium?

A window handle stores the unique address of the browser windows. It is just a pointer to a window, whose return type is alphanumeric. The window handle in *Selenium* helps in handling multiple windows and child windows. Each browser will have a unique window handle value with which we can uniquely identify it.

For example: When you open a website say " <https://demoqa.com/browser-windows> " and click on browser windows inside, each opened window will have an ID using which we will be able to switch the *Selenium WebDriver* context to that window and perform any operation within that window. In the image below you can see three windows - one parent window and two child windows. All three of them will have a unique window handle i.e an ID.



Each window here will have a unique *ID* that we can get using the methods provided by *Selenium WebDriver* and then use the same to switch the context to that specified window. Let's first understand what are the different methods provided by *Selenium WebDriver* for handling windows?

What are the different methods used for window handling in Selenium?

Selenium WebDriver provides various methods for handling of windows. Few of them are:

getWindowHandle() : When a website opens, we need to handle the main window i.e the parent window using `driver.getWindowHandle()` method. With this method, we get a unique ID of the current window which will identify it within this driver instance. This method will return the value of the **String** type.

getWindowHandles() : To handle all opened windows which are the **child windows** by web driver, we use `driver.getWindowHandles()` method. The windows store in a **Set** of String type and here we can see the transition from one window to another window in a web application. Its return type is **Set <String>**.

switchTo() : Using this method we perform switch operation within windows.

action: This method helps in performing certain actions on the windows.

How do we handle child windows in Selenium?

As seen in the above example if we have child windows in any web application then interaction with them without proper window handling will lead to an exception. For this we have different methods explained above, we will use them here with a practical example.

We will be using ***getWindowHandle()*** and ***getWindowHandles()*** method here along with ***switchTo()*** method. In the snapshot below, we have highlighted the two main methods for window handling in Selenium.

```
driver.get("https://demoqa.com/browser-windows");
driver.findElement(By.id("windowButton")).click();
String mainWindow = driver.getwind
Set<String> s1 = driver.getWindowHandle()
Iterator<String> i1 = s1.getWindowHandles()
while (i1.hasNext()) {
    String ChildWindow = i1.next();
```

Taking the same example of "***ToolsQA Demo Site***" above where exception was encountered, we will show how it will be executed successfully. After opening the URL, we will click on the "***New window***" button within the application, a new browser window opens. We will read the text from the newly opened window i.e "***This is sample page***" and will print it.

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import java.util.Iterator;
import java.util.Set;

public class ChildWindow {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "./src/main/resources/chromedriver");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://demoqa.com/browser-windows");

        // Open new child window within the main window
        driver.findElement(By.id("windowButton")).click();

        //Get handles of the windows
        String mainWindowHandle = driver.getWindowHandle();
        Set<String> allWindowHandles = driver.getWindowHandles();
        Iterator<String> iterator = allWindowHandles.iterator();

        // Here we will check if child window has other child windows and will fetch the
        while (iterator.hasNext()) {
            String ChildWindow = iterator.next();
            if (!mainWindowHandle.equalsIgnoreCase(ChildWindow)) {
                driver.switchTo().window(ChildWindow);
                WebElement text = driver.findElement(By.id("sampleHeading"));
            }
        }
    }
}
```

```

        System.out.println("Heading of child window is " + text.getText());
    }
}
}
}
}

```

Launch the website "<https://demoqa.com/browser-windows>" and click on the windows - "**windowbutton**".

String mainWindow = driver.getWindowHandle(); It stores parent window value in a unique identifier of string type.

Set<String> s = driver.getWindowHandles(); All child windows are stored in a set of strings.

Iterator<String> i = s.iterator(); Here we will iterate through all child windows.

if (!MainWindow.equalsIgnoreCase(ChildWindow)) : Now check them by comparing the main window with the child windows.

driver.switchTo().window (ChildWindow); Switch to the child window and read the heading.

WebElement text = driver.findElement(By.id("sampleHeading")); Find the element and store in a web element through which we will get the text of heading using **getText()** method.

The output of the code will print the text **"This is a sample page"** as shown below:

```

↑ ChromeDriver was started successfully.
↓ [1602402094.833][WARNING]:
⏏ Oct 11, 2020 1:11:34 PM org.openqa.selenium.remote.ProtocolHandshake createSession
⇅ INFO: Detected dialect: W3C
⇅ Heading of child window is This is a sample page
⇅
⇅ Process finished with exit code 0

```

So this way, we switched the context to the child window and then printed the text in the child windows.

How to handle multiple windows in Selenium?

In *Selenium*, when we have multiple windows in any web application, the approach may need to switch control among several windows i.e from one window to another to perform any action and we can achieve this by using **switchto();** method. Furthermore, we will be using window handle here to store the unique value of the windows and perform window handling using *Selenium*.

Note: If you have to switch between tabs then also you have to use the same approach.

Let's understand this with the help of the code below:

```

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import java.util.Iterator;
import java.util.Set;

```



```

import java.util.Set;

public class multipleChildWindows {
    public static void main(String[] args) throws InterruptedException {
        System.setProperty("webdriver.chrome.driver", "./src/main/resources/chromedriver");
        WebDriver driver = new ChromeDriver();

        driver.manage().window().maximize();
        driver.get("https://demoqa.com/browser-windows");

        // Opening all the child window
        driver.findElement(By.id("windowButton")).click();
        driver.findElement(By.id("messageWindowButton")).click();

        String MainWindow = driver.getWindowHandle();
        System.out.println("Main window handle is " + MainWindow);

        // To handle all new opened window
        Set<String> s1 = driver.getWindowHandles();
        System.out.println("Child window handle is" + s1);
        Iterator<String> i1 = s1.iterator();

        // Here we will check if child window has other child windows and when child window
        //is the main window it will come out of loop.
        while (i1.hasNext()) {
            String ChildWindow = i1.next();
            if (!MainWindow.equalsIgnoreCase(ChildWindow)) {
                driver.switchTo().window(ChildWindow);
                driver.close();
                System.out.println("Child window closed");
            }
        }
    }
}

```

Launch the website "<https://demoqa.com/browser-windows>" and click on the two popup windows - "**windowbutton**" and "**messagewindowbutton**".

String Mainwindow = driver.getWindowHandle(); It stores parent window value in a unique identifier of string type.

Set<String> s1 = driver.getWindowHandles(); All child windows are stored in a set of strings.

Iterator<String> i1 = s1.iterator() : Here we will iterate through all child windows.

if (!MainWindow.equalsIgnoreCase(ChildWindow)) : Now close them by comparing the main window with the child windows.

driver.switchTo().window(ChildWindow); It will also print when the child windows close.

driver.close(); When the main window is the only active window then it will come out of the loop and window will close.

```

INFO: Detected dialect: W3C
Main window handle is CDwindow-E596A0ED63E81B55F2F355F85EBA778D
Child window handle is[CDwindow-E596A0ED63E81B55F2F355F85EBA778D, CDwindow-AD0CB8FF337D1147A9B2170FCD315C5F, CDwindow-762CDBD447418CC43BDCF6760ED023A2]
Child window closed
Child window closed
Process finished with exit code 0

```

Once the *Selenium WebDriver* instantiates, allocation of a unique alphanumeric id happens to the window called *window handle* and it identifies the browser windows. In the above code, parent window and one of the child window has the same ID, the other two windows have a different ID.

This is because a parent window is the child of itself. But notice that only two closed due to the same reason. Since this id is distinctive, the *Selenium WebDriver* use such it to switch between different windows (*or tabs*). The id retains until the session closes.

How do we switch back to the parent window from the child windows in Selenium?

Once you have switched to the *child window*, the *Selenium WebDriver* will hold the current context of it, and you will not be able to identify elements present in the *parent window*. To access the elements of the parent window we have to shift the focus back on it. We can achieve this as shown in the below code snippet:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import java.util.Iterator;
import java.util.Set;

public class switchbackParentWinow {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "./src/main/resources/chromedriver");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://demoqa.com/browser-windows");

        driver.findElement(By.id("windowButton")).click();
        String mainwindow = driver.getWindowHandle();
        Set<String> s1 = driver.getWindowHandles();
        Iterator<String> i1 = s1.iterator();

        while (i1.hasNext()) {
            String ChildWindow = i1.next();
            if (!mainwindow.equalsIgnoreCase(ChildWindow)) {
                driver.switchTo().window(ChildWindow);
                WebElement text = driver.findElement(By.id("sampleHeading"));
                System.out.println("Heading of child window is " + text.getText());
                driver.close();
                System.out.println("Child window closed");
            }
        }

        // Switch back to the main window which is the parent window.
        driver.switchTo().window(mainwindow);
        driver.quit();
    }
}
```

The output of the above code is shown below:

```
Oct 11, 2020 1:59:05 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
Heading of child window is This is a sample page
Child window closed

Process finished with exit code 0
```

Practise Exercise : Try to implement the following scenario, where you can print text from the main window after switching back to the parent window.

- . Open the URL "<https://demoqa.com/browser-windows>".
- . Click on all the child windows
- . Print the text present on all the child windows in the console.
- . Print the heading of the parent window in the console.

How to close all windows in Selenium?

When we are working on multiple windows it is important to close windows simultaneously when we finish the action. For closing the window on which *WebDriver* has current focus we have **`driver.close()`**; method. We use this method majorly when we have multiple windows and we wish to close a selective window.

Another method that exists for the closing window is **`driver.quit()`** which will close all the windows opened in a particular session. It basically stops the driver instance and any further actions to *WebDriver* may result in an exception. It is generally the last statement of any code.

In the examples given above, we have used both methods.

Note: After closing a child window we have to explicitly switch back to the parent window before using any *WebDriver* command to avoid "**`nosuchwindow`**" exception.

Key Takeaways

A window is a webpage that will open when a user hits a URL. There are two types of windows in the Selenium - parent window and its child windows.

The window handle is a unique identifier that stores the values of windows opened on a webpage and helps in window handling in Selenium.

`getWindowHandles()` and **`getWindowHandles()`** handle windows in Selenium.

*The user has to switch from the parent window to the child window to work on them using **`switchTo()`** method.*

To close windows two methods exist **`driver.close()`** and **`driver.quit()`**.