**Find Broken Links in Selenium**

On a webpage, generally, there are multiple links/hyperlinks, and they serve particular purposes. A few links redirect to third-party websites such as Wikipedia and some redirect to the same website. They are a beautiful way to reduce the searching time. For example, If I am reading an article about selenium and says that [***Selenium WebDriver***](https://www.toolsqa.com/selenium-webdriver/selenium-testing/) is used for automation testing, I can link the automation testing word, which would redirect to a detailed article about automation testing. This will save time for someone who did not know about the automation testing and searched it separately. Therefore, they are important and these broken links in Selenium require testing and verification before publishing a web page.

A working URL will always have an [***HTTP response***](https://www.toolsqa.com/client-server/http-response/) code 200, which is a success and valid. While testing these links, it is difficult for a user to manually click and check all the broken links on a webpage. Therefore, we can search for all the links on a web page and check the status of each of the links to validate whether it is a valid link or not. The same is the case with Images, where we can check for the valid and visible image by validating that the images' src link is valid. This article will explain various ways to validate whether certain images and links are valid or are in broken status while browsing through the all/broken links in selenium tests.

* What are links on a Web Page?
  + What are HTTP Status Codes?
  + Also, what are broken links on a Web Page?
  + What are broken images on a Web Page?
* How to find all the links on a Web Page in Selenium?
  + How to find the broken links in Selenium tests?
  + And how to find broken images in Selenium tests?

## What are links on a Web Page?

***Hyperlinks***, usually called links, are those HTML tags/elements on a web page which we use for redirection to another web page. This happens when a user performs a click operation on these hyperlinks. The user can instantly reach the target page by hitting the link, and the link activates. Each link/hyperlink always contains a target or URL, the URL of the page that will open when the link will be clicked. This link should be valid, so we can open the needed page when someone clicks on the mentioned hyperlink.

Now how do we categorize whether a link is valid or not? When we hit any URL, we know that it returns some HTTP codes, which depending on the return value, signifies whether the mentioned link is valid. Let's quickly understand the meanings of various HTTP status codes which can return while hitting a URL:

### ***What are HTTP Status Codes?***

***A server generates HTTP Status codes*** in response to the request submitted by the client to the server. There are five types of responses to which we can segregate ***HTTP***  response status codes. The first digit of the status-code is the response type, and the last two digits have different interpretations associated with the status code. There are different ***HTTP*** status codes, and a few of them are as below:

* ***200*** – Valid Link/success
* ***301/302*** - Page redirection temporary/permanent
* ***404*** – Page not found
* ***400*** – Bad request
* ***401*** – Unauthorized
* ***500*** – Internal Server Error

We will be using these HTTP codes in our tests to ensure that the link is valid or not.

### ***What are broken links on a Web page?***

A ***broken link,*** often called a ***dead link,*** is any link on a web page that no longer works because there is an underlying issue with the link. When someone clicks on such a link, sometimes an error message is displayed like a page not found. There may not be any error message at all.  These are essentially invalid HTTP requests and have ***4xx*** and ***5xx*** status code. Some common reasons for a broken link on a webpage can be:

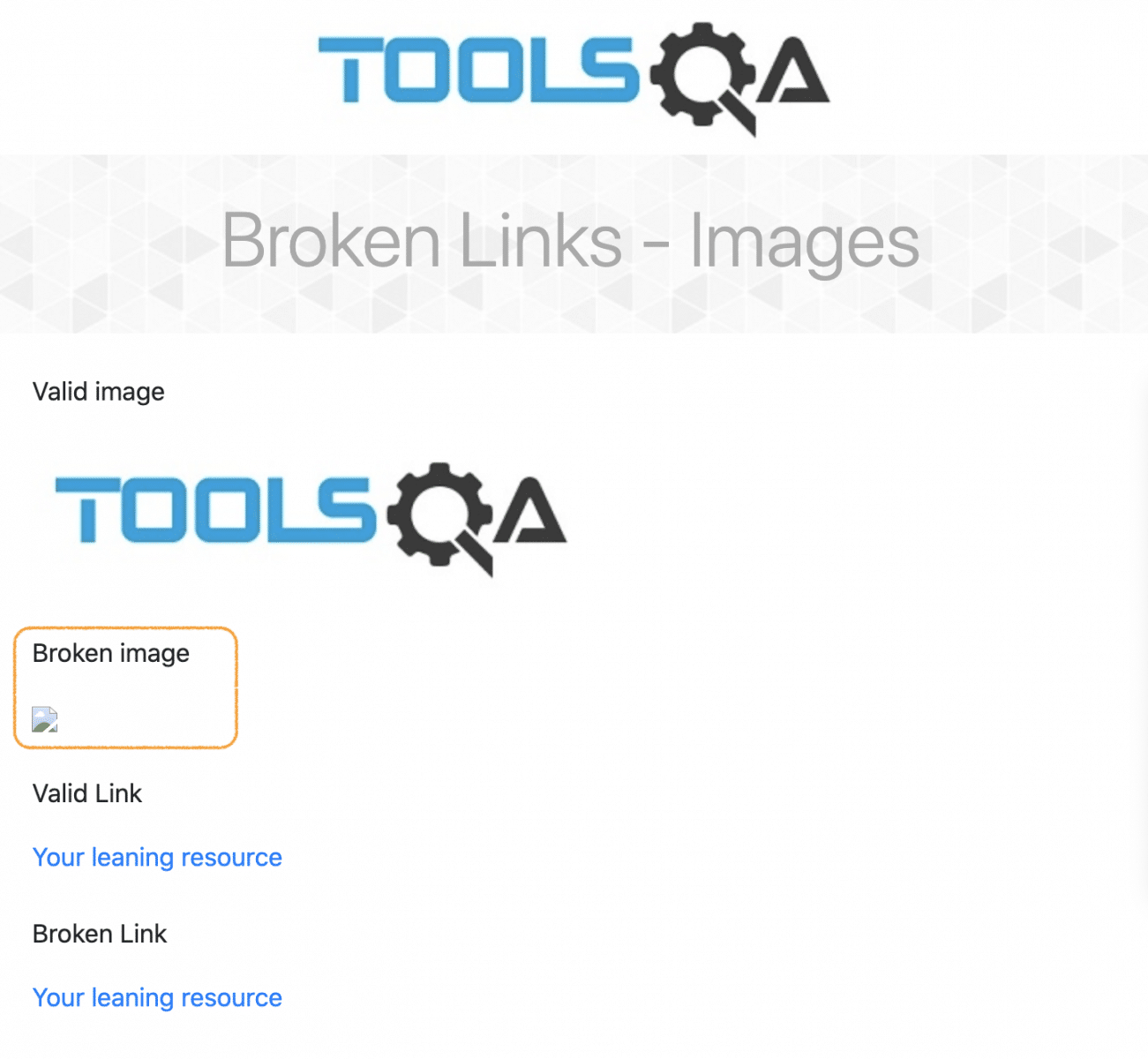
* The destination web page is down, moved, or no longer exists.
* A web page moved without adding a redirect link.
* The user entered an improper/misspell URL.
* The web page link removed from the website.
* With activated firewall settings, also the browser cannot access the destination web page at times.

### ***What are broken images on a Web Page?***

There are cases where an image on the web page does not load properly, and we see ***"Failed to load image"*** or similar error messages. In such cases, the image is either corrupt or the image is not at the specified path. A ***broken image*** on a web page is a link that is associated with the image, and the link is not working. There can be three possible reasons because of which images doesn't show up on web pages:

* Firstly, the image file is not located in the same path specified in your ***<img src " ">*** tag.
* Secondly, the image does not have the same path or filename.
* Thirdly, the image file at the location is either corrupt or undergoes damage, or maybe it's not compatible with a specific browser, and rendering fails in that browser only.

The below image shows how a broken image can look like:

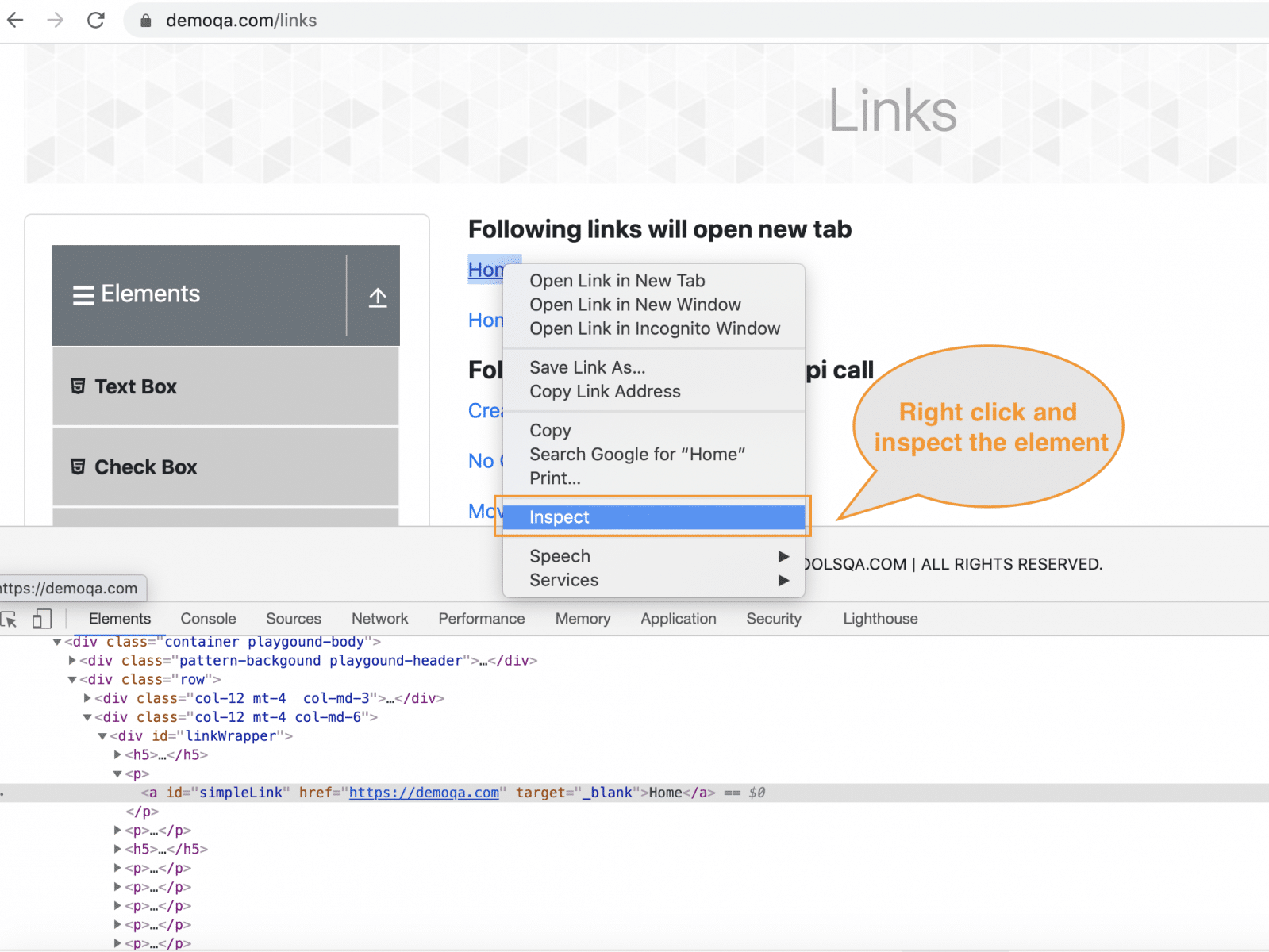


***Note:*** Images can be broken on a web page, even if the link is valid on the page. In such a case, the issue is with either the image file itself or the browser's image rendering.

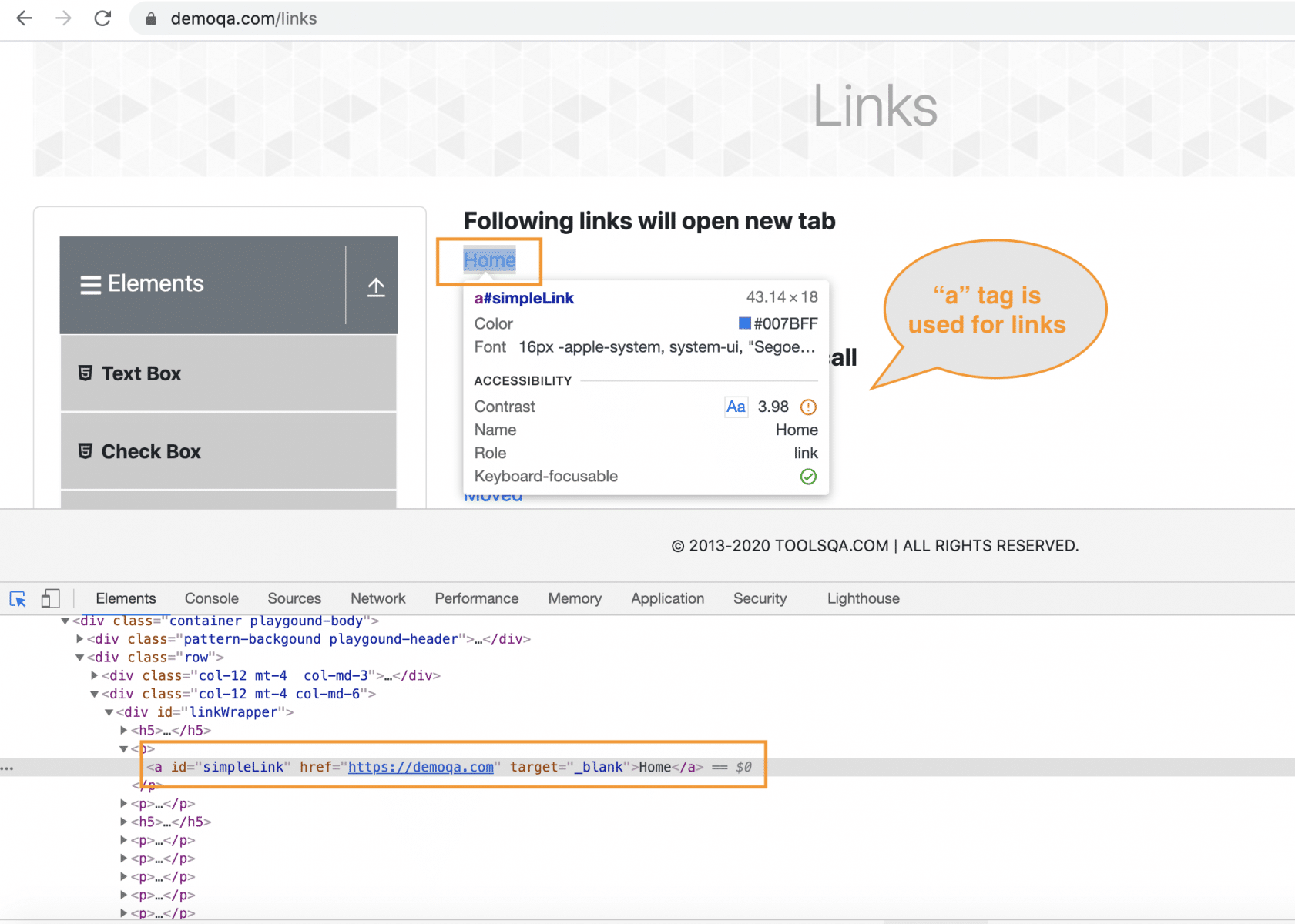
## How to find all the links on a Webpage in Selenium?

Before finding out the broken links in Selenium, it is better to understand the overall generic concept by finding all the links on a webpage. The hyperlinks are generally implemented on a web page using the HTML Anchor (*<a>*) tag. So, if you identify and locate all the anchor tags on a web page and then get the corresponding URLs, we will be able to traverse through all the links on the web page. Let's understand this using the following example:

1. Navigate to the desired webpage, [***"https://demoqa.com/links"***](https://demoqa.com/links).
2. Right-click on the Web element and click on the ***Inspect*** option from the dropdown.



1. Fetch the element with ***tag name*** ='a', and we will be using this tag for checking all the links



The below-mentioned code will help you fetch the links (tags) from the above web page and test them.

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.List;

public class GetAllURLs {

public static void main(String[] args) {

//Create WebDriver instance and open the website.

System.setProperty("webdriver.chrome.driver","./src/main/resources/chromedriver");

WebDriver driver = new ChromeDriver();

driver.manage().window().maximize();

driver.get("https://demoqa.com/links");

String url="";

List<WebElement> allURLs = driver.findElements(By.tagName("a"));

System.out.println("Total links on the Wb Page: " + allURLs.size());

//We will iterate through the list and will check the elements in the list.

Iterator<WebElement> iterator = allURLs.iterator();

while (iterator.hasNext()) {

url = iterator.next().getText();

System.out.println(url);

}

//Close the browser session

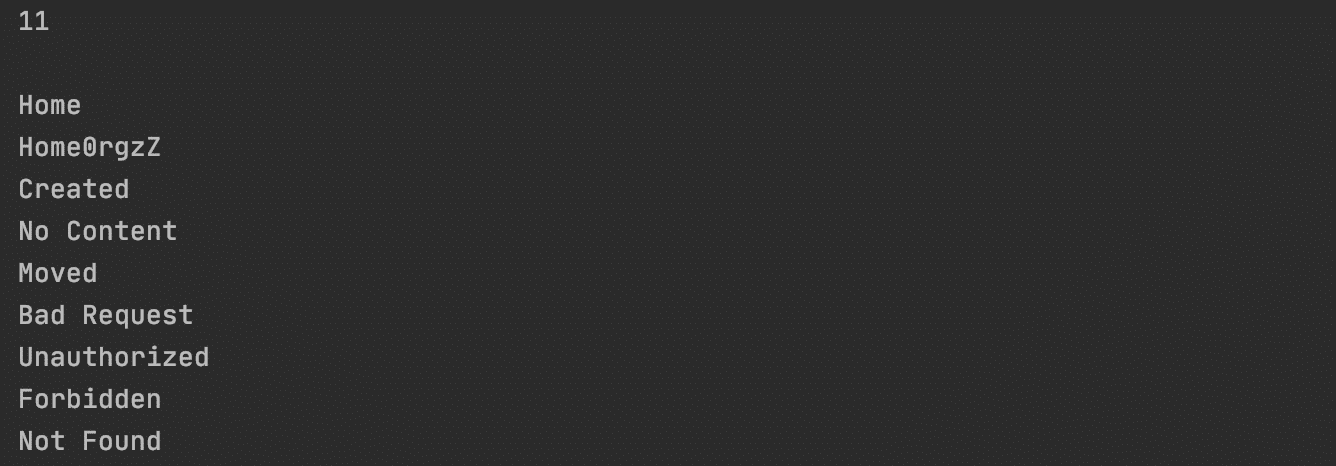
driver.quit();

}

}

***Code walkthrough:***

* Open URL and inspect the desired element.
* ***List<WebElement> allURLss = driver.findElements***(By.tagName("a")); In this we will get list of WebElements with tagname 'a'.
* Traverse through the list using the [***Iterator.***](https://www.toolsqa.com/java/list-interface/#:~:text=Method%3A%20iterator()&text=For%20the%20reader%20unfamiliar%20with,point%20to%20the%20next%20page.&text=%2F%2F%20Get%20a%20list%20iterator%20over%20the%20elements%20in%20the%20list.,-ListIterator%20listIterator)
* Print the links text using ***getText()*** method.
* Close the browser session with the ***driver.quit()*** method.



After executing the above code, we receive a count of links, which is 11. Additionally, we also retrieve labels of each link and print it in the console.

### ***How to find broken links in Selenium tests?***

As we discussed, we can check the status code of the link URL to validate whether it is a valid link or not. Let's modify the above code snippet to check how we can validate whether a certain link is valid or not:

package testCases;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import java.net.HttpURLConnection;

import java.net.URL;

import java.util.List;

public class BrokenLinks {

public static void main(String[] args) {

System.setProperty("webdriver.chrome.driver","./src/resources/chromedriver");

WebDriver driver = new ChromeDriver();

driver.manage().window().maximize();

driver.get("https://demoqa.com/broken");

//Storing the links in a list and traversing through the links

List<WebElement> links = driver.findElements(By.tagName("a"));

// This line will print the number of links and the count of links.

System.out.println("No of links are "+ links.size());

//checking the links fetched.

for(int i=0;i<links.size();i++)

{

WebElement E1= links.get(i);

String url= E1.getAttribute("href");

verifyLinks(url);

}

driver.quit();

}

public static void verifyLinks(String linkUrl)

{

try

{

URL url = new URL(linkUrl);

//Now we will be creating url connection and getting the response code

HttpURLConnection httpURLConnect=(HttpURLConnection)url.openConnection();

httpURLConnect.setConnectTimeout(5000);

httpURLConnect.connect();

if(httpURLConnect.getResponseCode()>=400)

{

System.out.println(linkUrl+" - "+httpURLConnect.getResponseMessage()+"is a broken link");

}

//Fetching and Printing the response code obtained

else{

System.out.println(linkUrl+" - "+httpURLConnect.getResponseMessage());

}

}catch (Exception e) {

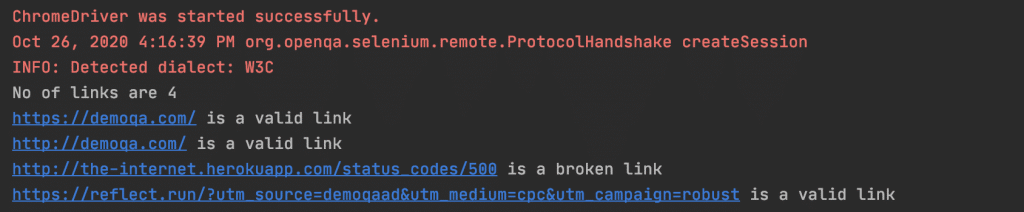
}

}

}

* Create a WebDriver instance and open URL in the browser [***"https://demoqa.com/broken"***](https://demoqa.com/broken).
* ***HttpURLConnection httpURLConnect=***(HttpURLConnection)url.openConnection(): we will check the HTTP status of each using HttpURLConnection class in Java.
* ***httpURLConnect.setConnectTimeout***(5000): It is important to wait before creating a connection as the URL may take time to load. We have set the Connection timeout of 5 seconds.
* ***httpURLConnect.connect()*** : Now creation of connection happens.
* ***getResponsecode():*** We will fetch the response code and print ***OK*** if the URL works fine. Else will give an error.

***Console Output:*** We are getting ***4 links*** from the webpage, and the HTTP status code of each link is displayed.



All the web page links should work properly to avoid a bad user experience and keep the user engaged. So this way, by checking the status code for each of the links, we can identify whether a particular link is in broken status or not.

### ***How to find broken images in Selenium tests?***

As discussed, an image appears broken either due to an invalid src link of the image or the image's bad rendering. To be 100% sure that whether the image is broken or not, we will need to validate both the perspective of the image, i.e., the URL of the image should be valid, i.e., should return status code as 200, and the image should render correctly on the browser window, which we can validate using JavaScript. Marker 1 highlights a valid image in the above image, and Marker 2 highlights an invalid/broken image. Now let's see how we can locate and identify the same in Selenium tests:

import org.openqa.selenium.By;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import java.util.List;

import java.net.HttpURLConnection;

import java.net.URL;

public class BrokenImages {

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://www.demoqa.com/broken");

// Storing all elements with img tag in a list of WebElements

List<WebElement> images = driver.findElements(By.tagName("img"));

System.out.println("Total number of Images on the Page are " + images.size());

//checking the links fetched.

for(int index=0;index<images.size();index++)

{

WebElement image= images.get(index);

String imageURL= image.getAttribute("src");

System.out.println("URL of Image " + (index+1) + " is: " + imageURL);

verifyLinks(imageURL);

//Validate image display using JavaScript executor

try {

boolean imageDisplayed = (Boolean) ((JavascriptExecutor) driver).executeScript("return (typeof arguments[0].naturalWidth !=\"undefined\" && arguments[0].naturalWidth > 0);", image);

if (imageDisplayed) {

System.out.println("DISPLAY - OK");

}else {

System.out.println("DISPLAY - BROKEN");

}

}

catch (Exception e) {

System.out.println("Error Occured");

}

}

driver.quit();

}

public static void verifyLinks(String linkUrl)

{

try

{

URL url = new URL(linkUrl);

//Now we will be creating url connection and getting the response code

HttpURLConnection httpURLConnect=(HttpURLConnection)url.openConnection();

httpURLConnect.setConnectTimeout(5000);

httpURLConnect.connect();

if(httpURLConnect.getResponseCode()>=400)

{

System.out.println("HTTP STATUS - " + httpURLConnect.getResponseMessage() + "is a broken link");

}

//Fetching and Printing the response code obtained

else{

System.out.println("HTTP STATUS - " + httpURLConnect.getResponseMessage());

}

}catch (Exception e) {

}

}

}

When we run the above test, we will see the output as shown below:



As we can see, even though the URL for the image was returning a valid status code, it still appears as broken, and we can identify the same using the corresponding JavaScript code.