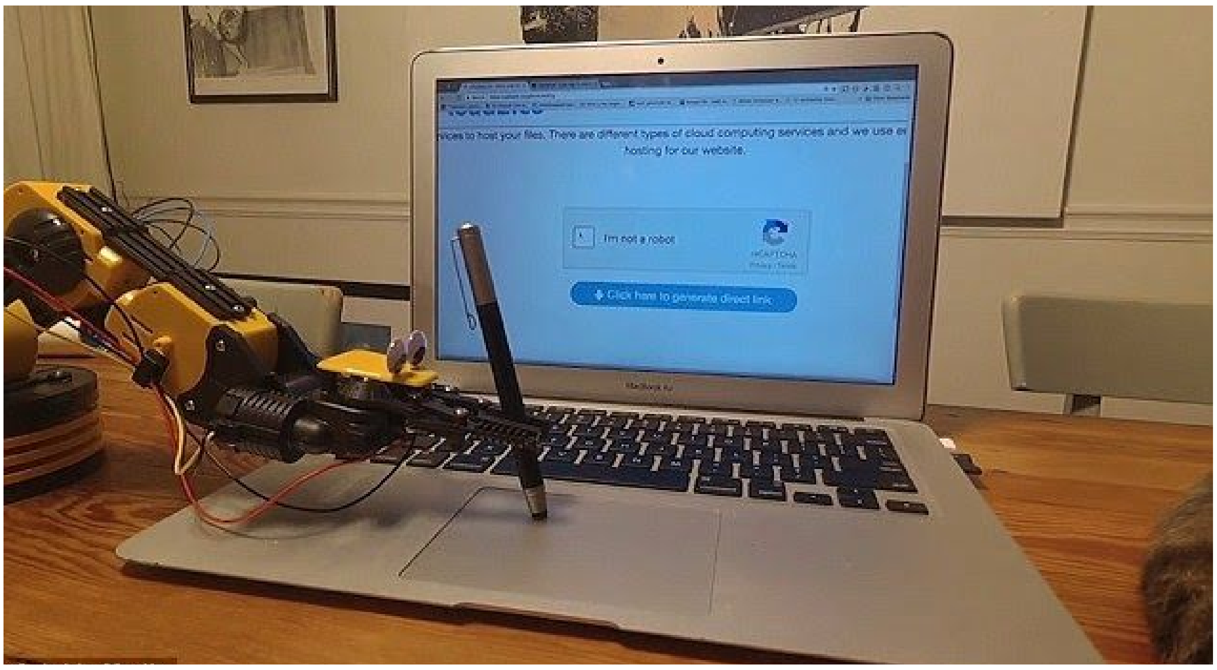
How to Handle Robot class in Selenium?

When we hear the word robot the first thing that comes to our mind is a machine or a program that completes its task within a specified amount of time and doesn't need manual intervention. In this article we will discuss Robot class in selenium and certain functions in Selenium that helps in controlling mouse and keyboard.



This article on Robot class will make your understanding more clear in Selenium. This article will discuss the following topics in a great detail:

What Is Robot Class in Selenium?

Robot Class facilitates Selenium to use actual mouse and keyboard events rather than simulated mouse. Robot Class can handle window based pop-pos like(alerts, download based pops, etc) or native applications like(Calculator, Notepad, etc). Robot Class in Selenium also helps in completing the task in specified time.

Robot Class is invoked using the following syntax:

```
Robot robot = new Robot()

robot.mouseMove(coordinates.getX(), coordinates.getY());
```

Importance of Robot Class in Selenium

Here one important question arises:

why do we need robot class when we can perform actions on the keyboard as well as hover the mouse on the location on the web page?

Robot class is used to handle and simulate the mouse and keyboard functions. We don’t have to click any button while automating a web page. It can handle all the pop-ups as well as the notification section of the web page. It also helps you when you want to upload a file on to an application, this can be done using this robot class. Hope you all have understood what is robot class and why it is used in Selenium.

Robot Class in Java

Robot Class was introduced in Java in version 1.3. The Robot Class in Java AWT package is to generate native system input events for the purpose of automation testing, self running demos and other applications where control of mouse and keyboard is needed. The main purpose of this class is to provide automated testing of java platform implementations.

Methods of Robot class

Mouse Actions:

1. **mouseMove(X,Y):** Move mouse pointer to given screen coordinates.
2. **mouseRelease(buttons):** Releases one or more mouse buttons.
3. **mousePress(buttons):** Presses one or more mouse buttons.
4. **mouseWheel(wheelAmt):** Rotates the scroll wheel on wheel-equipped-mice.
5. **getPixelColor(X,Y):** Returns the color of the pixel at a given screen coordinate.

Keyboard Actions:

1. **keyPress(keycode):** Presses a given key.
2. **keyRelease(keycode):** Releases a given key.

For example, Robot.mouseMove will actually move the mouse cursor instead of just generating mouse move events.

Well have you confused between Robot class and Action class, because they both seem to handle mouse and keyboard events. Lets help you out.

Difference between Action class and Robot class in Selenium

The difference between Action class and Robot class is mentioned below:

Action class	Robot class
Action class simulates a mouse and Keyboard.	Robot class enables the actual mouse and keyboard, because of this reason you can see the movement of the mouse cursor.
Action class doesn’t affect parallel running.	Robot class affects parallel running as there is only one mouse connected to the system.
Action class uses the native browser commands.	Robot class doesn’t use browser based commands.
Action class is limited to browser application.	Robot class can be used along with all applications.
Action class is from org.openqa .selenium.interactions.Actions.	Robot class is from java.awt.Robot.

Let's look at some of the applications of Robot Class in the real world.

Enter Text Using Robot class in Selenium

Test Scenario: In the following code we use automation to automate browserstack.com webpage, to enter the Full Name of the User automatically without any manual intervention..

Explanation:

To perform the keyboard action, we use

Step 1: Establish the connection with the web driver, Here we have used ChromeDriver.

Step 2: Enter the corresponding url, the pop will appear during navigation.

Step 3: Find the element on the web page which you want to navigate. Here we have used Get started for free.

Step 4: To handle the pop ups an instance of robot class is created.

Step 5: Let's explain briefly some of the methods used in the below Code Snippet.

- **r.KeyPress(KeyEvent.VK_TAB):** This command presses the **TAB button** to navigate to the text field(Full Name).
- **r.KeyPress(KeyEvent.VK_A)**: This command pressed the the **button A** in the corresponding field, and accordingly enter the name as **ADMIN.**

Step 6: After the use of the driver you must **quit** the driver.

To perform the mouse action, we use

Step 1: Use the MouseMove() method that takes X and Y coordinates as the parameter, to move the mouse to the given coordinates. **Note:** Screen Coordinates may differ for different users and different screen sizes.

Step 2: After reaching the desired coordinates we need to perform the left click operation. For this purpose we use **robot.mousePress(InputEvent.BUTTON1_MASK).**

Step 3: After a button is pressed it must be released, so we will use **robot.mouseRelease(InputEvent.BUTTON1_MASK)**, this command will release the left click of the mouse.

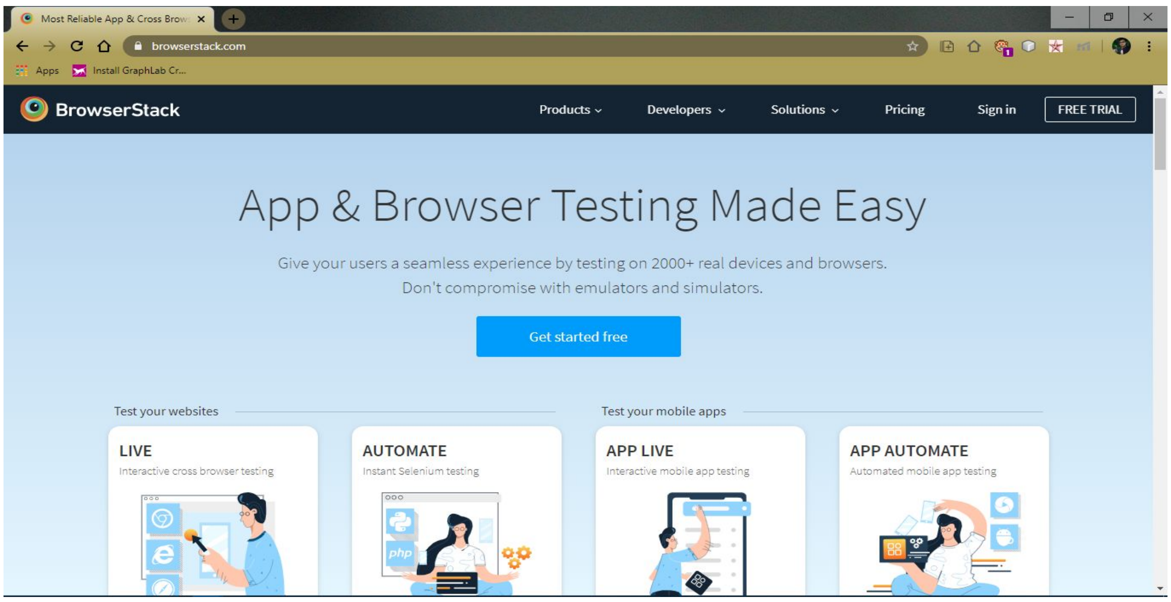
Code Snippet:

```
import java.awt.AWTException;
import java.awt.Robot;
import java.awt.event.KeyEvent;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium..chrome.ChromeDriver;
public class Check {
    public static void main(String[] arg) throws AWTException, InterruptedException
    {
System.setProperty("webdriver.chrome.driver","C:UsersXYZDesktopchromedriver_win64chromedr
iver.exe");
        WebDriver driver = new ChromeDriver();
         driver.get("https://www.browserstack.com/");  //corresponding url
         Thread.sleep(3000);
          //Clicks the following button
         driver.findElement(By.linkText("Get started free")).click();
         Robot r = new Robot();  //instance of Robot class
         Thread.sleep(3000);
         r.KeyPress(KeyEvent.VK_TAB);  //Presses TAB button
         r.KeyPress(KeyEvent.VK_A);  //Presses the button A from keyboard
         r.keyRelease(KeyEvent.VK_A); //Releases the button A from keyboard
         r.KeyPress(KeyEvent.VK_D);  //Presses the button D from keyboard
         r.keyRelease(KeyEvent.VK_D); //Releases the button D from keyboard
         r.KeyPress(KeyEvent.VK_M);  //Presses the button M from keyboard
         r.keyRelease(KeyEvent.VK_M); //Releases the button M from keyboard
         r.KeyPress(KeyEvent.VK_I);  //Presses the button I from keyboard
         r.keyRelease(KeyEvent.VK_I); //Releases the button I from keyboard
         r.KeyPress(KeyEvent.VK_N);  //Presses the button N from keyboard
         r.keyRelease(KeyEvent.VK_N); //Releases the button N from keyboard
         driver.quit()  //quits the driver
    }
}
```

Let's look at the output, what our code really does.

Output:

It first fetches the URL of webpage browserstack.com



Most Reliable App & Cross Brow ×

browserstack.com

Apps Install GraphLab Cr...

BrowserStack Products Developers Solutions Pricing Sign in FREE TRIAL

App & Browser Testing Made Easy

Give your users a seamless experience by testing on 2000+ real devices and browsers.
Don't compromise with emulators and simulators.

Get started free

Test your websites Test your mobile apps

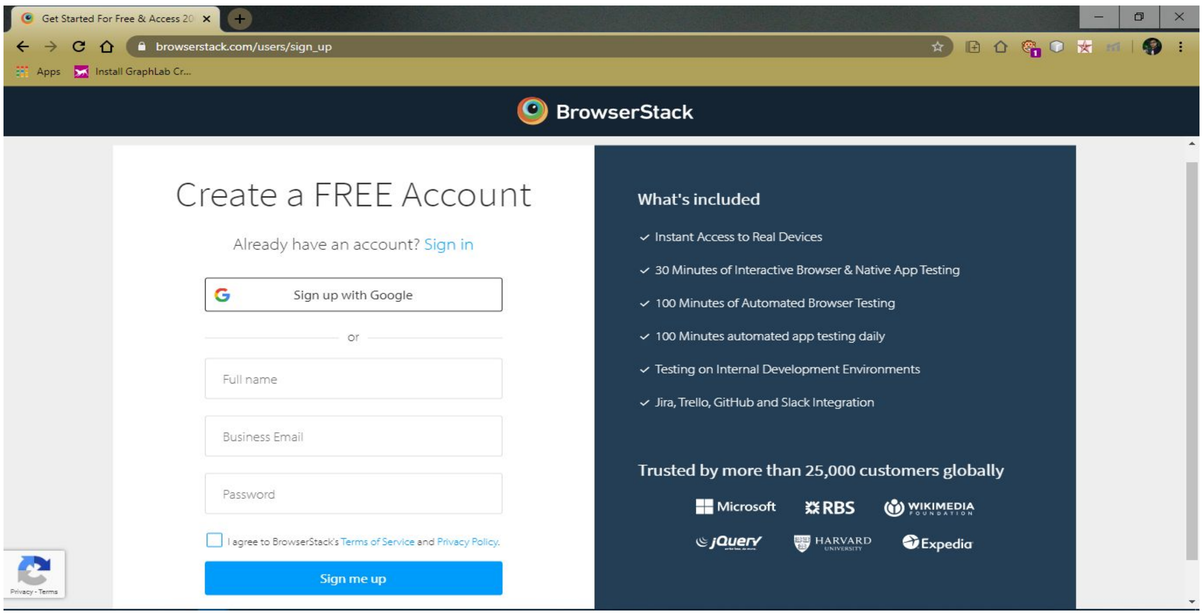
LIVE
Interactive cross browser testing

AUTOMATE
Instant Selenium testing

APP LIVE
Interactive mobile app testing

APP AUTOMATE
Automated mobile app testing

It moves to next webpage Get started for free, using findElement, this is specified in step 3 above



Get Started For Free & Access 20 ×

browserstack.com/users/sign_up

Apps Install GraphLab Cr...

BrowserStack

Create a FREE Account

Already have an account? Sign in

Sign up with Google

or

Full name

Business Email

Password

I agree to BrowserStack's Terms of Service and Privacy Policy.

Sign me up

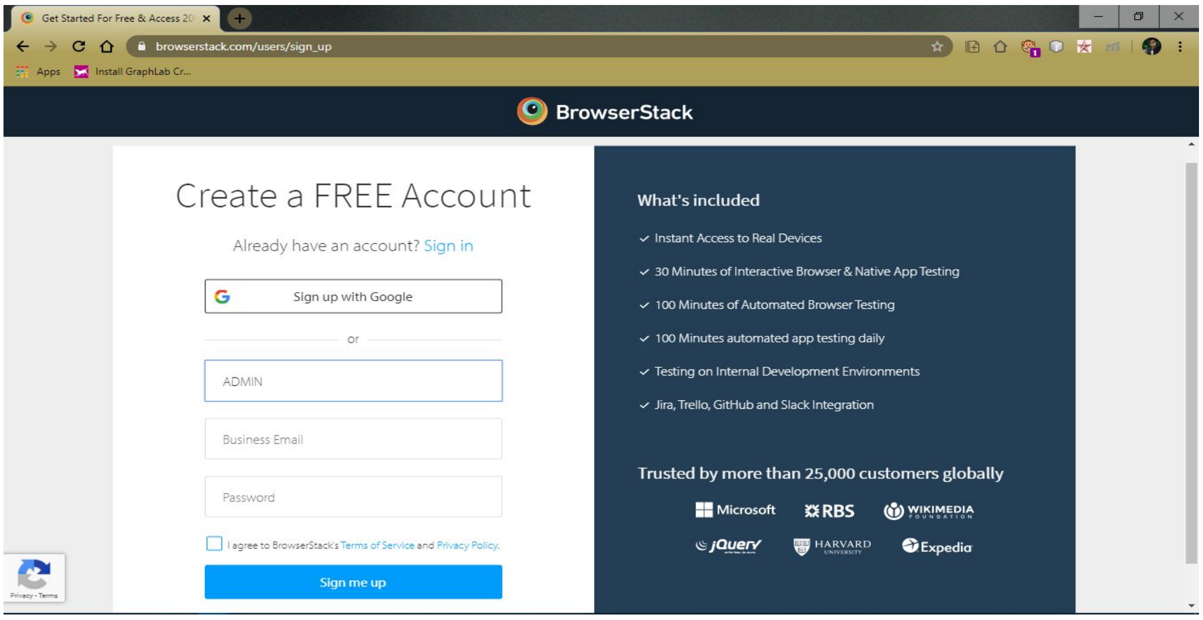
What's included

- Instant Access to Real Devices
- 30 Minutes of Interactive Browser & Native App Testing
- 100 Minutes of Automated Browser Testing
- 100 Minutes automated app testing daily
- Testing on Internal Development Environments
- Jira, Trello, GitHub and Slack Integration

Trusted by more than 25,000 customers globally

Microsoft RBS WIKIMEDIA jQuery HARVARD Expedia

Press the TAB button from Keyboard and enter ADMIN in Full Name field



Create a FREE Account

Already have an account? Sign in

ADMIN

Business Email

Password

☐ I agree to BrowserStack's Terms of Service and Privacy Policy.

What's included

- ✓ Instant Access to Real Devices
- ✓ 30 Minutes of Interactive Browser & Native App Testing
- ✓ 100 Minutes of Automated Browser Testing
- ✓ 100 Minutes automated app testing daily
- ✓ Testing on Internal Development Environments
- ✓ Jira, Trello, GitHub and Slack Integration

Trusted by more than 25,000 customers globally

Now, let's see how we can upload file using Robot class in Selenium

File Upload Using Robot class in Selenium

Test Scenario: In the following code, we have used automation to automatically upload files on gofile.io webpage, without manual intervention. Each step of the Code Snippet is explained below in a great detail.

Explanation:

Step 1: Establish the connection with the web driver, Here we have used ChromeDriver.
Step 2: Enter the corresponding url, the pop will appear during navigation.
Step 3: Find the element on the web page which you want to navigate. Here we have used found the element using **id('btnChoosefiles').**
Note: To find it by name or by id, right click on the respective element, goto to Inspect and find the corresponding element locator.
Step 4: For copy and paste options, **StringSelection** class is used. Here we have copied the path of the specified file(tutorial.pem) which we want to upload.
Step 5: getSystemClipboard() method is used to copy the elements to the clipboard.

Step 6: Instance of Robot class is created to copy the file path from clipboard to desired location.

Step 7: We use CTRL + V to paste the file path using the method **robot.KeyPress(KeyEvent.VK_CONTROL)**, here VK_CONTROL represents the CTRL key. Similarly, the use of the V key.

Note: Here we have directly pasted the content from the clipboard after clicking the desired button, because by default the control is in the same location. If the focus is elsewhere, you need to move the focus to file name.

Step 8: VK_ENTER simulates ENTER button on the keyboard. This will successfully upload the file.

Code Snippet:

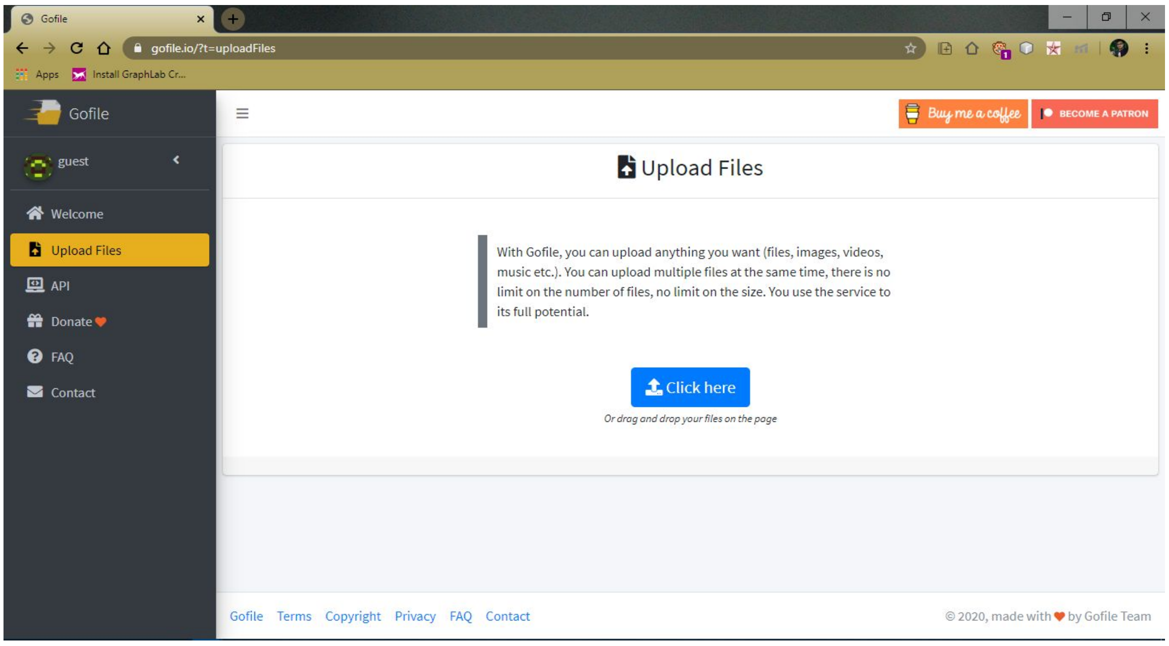
```
import java.awt.Robot;
import java.awt.Toolkit;
import java.awt.datatransfer.StringSelection;
import java.awt.event.KeyEvent;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;
public class UploadFileRobot {

System.setProperty("webdriver.chrome.driver","C:UsersXYZDesktopchromedriver_win64chromedr
iver.exe");
        WebDriver  driver = new ChromeDriver();
        //Path of the webpage
        driver.get("https://gofile.io/?t=uploadFiles");
        driver.findElement(By.id("btnChooseFiles")).click();
        Thread.sleep(2000);
         // This method will set any parameter string to the system's clipboard.
        //StringSelection is a class that can be used for copy and paste operations.
        StringSelection selection = new StringSelection("C:\\Users\Documents\tutorial.pem");
        Toolkit.getDefaultToolkit().getSystemClipboard().setContents(selection, null);
```

```
		try {
		//native keystrokes for CTRL, V and ENTER keys
		Robot robot = new Robot(); //Instance of Robot class
		Thread.sleep(2000);
		robot.keyPress(KeyEvent.VK_CONTROL); //Presses the CTRL button from keyboard
		robot.keyPress(KeyEvent.VK_V); //Presses the button V from keyboard
		robot.keyRelease(KeyEvent.VK_V); //Releases the button A from keyboard
		robot.keyRelease(KeyEvent.VK_CONTROL); //Releases the CTRL button from keyboard
		Thread.sleep(2000);
		robot.keyPress(KeyEvent.VK_ENTER); //Presses the ENTER button from keyboard
		robot.keyRelease(KeyEvent.VK_ENTER); //Releases the ENTER button from keyboard
		} catch (Exception exp) {
			exp.printStackTrace();
		}
		driver.quit(); //quits the driver
}
```

Output:

It first fetches the url gofile.io of webpage

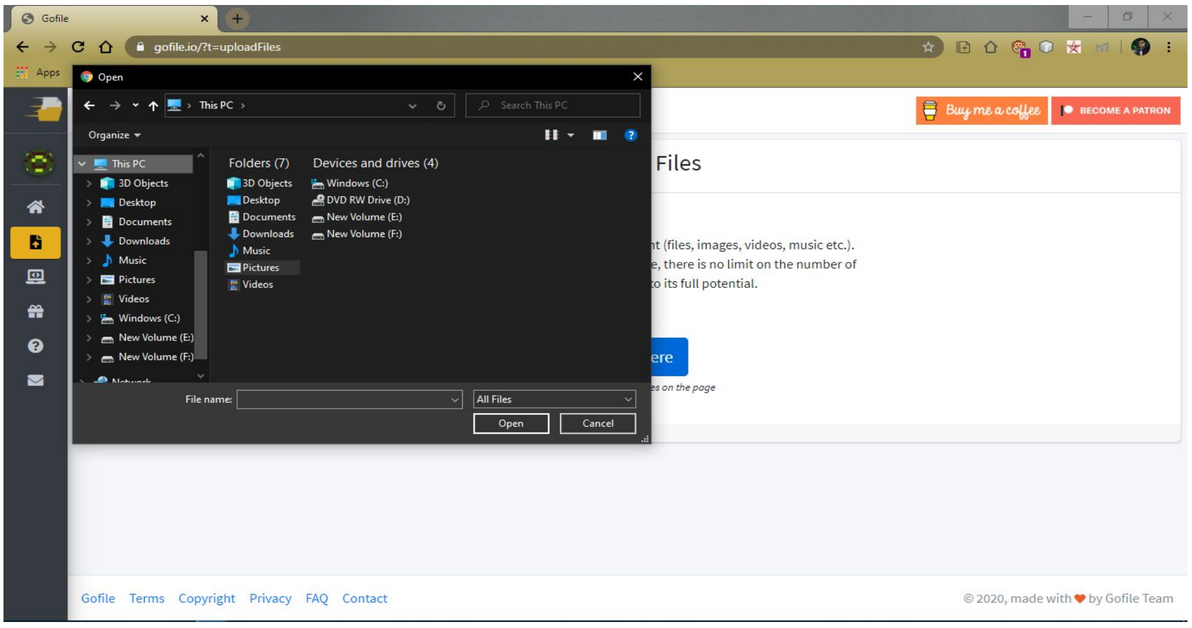


Upload Files

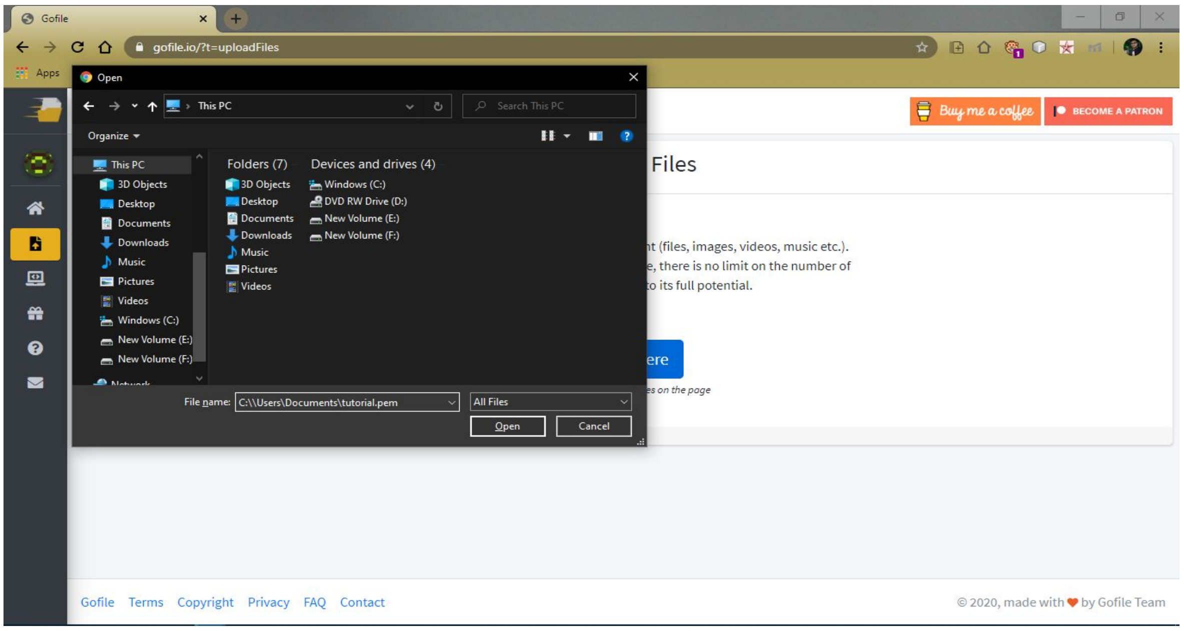
With Gofile, you can upload anything you want (files, images, videos, music etc.). You can upload multiple files at the same time, there is no limit on the number of files, no limit on the size. You use the service to its full potential.

Or drag and drop your files on the page

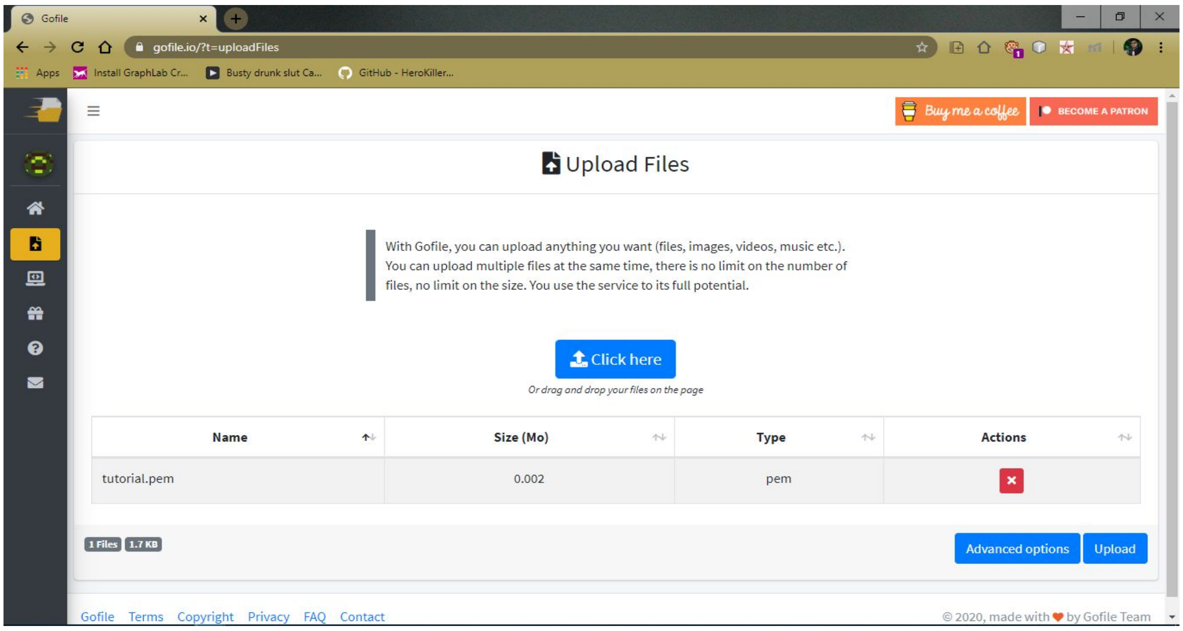
Click on the button Click here to upload files



Copy and paste the file path in File Name(text field).



Press the Enter Button, to upload file successfully



With Gofile, you can upload anything you want (files, images, videos, music etc.). You can upload multiple files at the same time, there is no limit on the number of files, no limit on the size. You use the service to its full potential.

Name	Size (Mo)	Type	Actions
tutorial.pem	0.002	pem	✕

Here we have shown two most common use cases of Robot class in Selenium to perform automation. Similarly we can use Robot class for various other activities like to take screenshots of each page, automatically sign-in on any particular webpage. Moreover, you can think of more sophisticated applications and implement them on your own. Now, let’s discuss some limitations/ drawbacks of using Robot class.

“Automation applied to an inefficient operation will magnify the efficiency” - Bill Gates

Limitations of Robot class in Selenium

Though robot class helps to manage window based pop-up, Robot framework has few disadvantages mentioned below:

1. Most of the methods like mouseMove are dependent on screen resolution, so the method might not perform the same on different machines.

2. Parallel running should be avoided because Robot class facilitates actual mouse commands so a machine cannot have two mice.

3. This class works only on the current instance of window, so behaviour might be different when multiple instances are open.

4. If a key is pressed using keyPress and not released using keyRelease then it will remain pressed and will consume memory in the background.

5. It may raise security exceptions if createRobot permission is not granted.

We hope that you guys have enjoyed this article on ‘Robot class in Selenium and you have a good understanding how robot class handles the mouse and keyboard events. If you have any query feel free to reach out to us.

References

1. https://docs.oracle.com/javase/7/docs/api/java/awt/Robot.html

2. https://www.seleniumeasy.com/selenium-tutorials/webdriver-file-upload-using-robots