Robot class in selenium

Table of content

Robot class in Selenium

Robot class enables selenium to use an actual mouse, **Actions class in selenium** only simulates a mouse, which means **Actions class** does not move the mouse cursor.

Robot class is very easy to use with the automation process. It can be easily integrated with Java **automation frameworks**

Selenium does not provide support to handle window based pop-ups (like download popups, upload popups). **Robot** Class used while we need to handle file upload and download activity using selenium webDriver.

Useful Robot Class Methods for selenium

keyPress();

keyRelease();

mousePress();

mouseRelease();

mouseMove();

mouseWheel(wheelAmt);

delay(ms);

createScreenCapture();

keyPress() : *Presses a given key, The key should be released using the keyRelease method.*
If there is more than one key present in the keyboard for Key Code value(Like SHIFT, ALT, CTRL) will map to the left key.

keyRelease() : *Releases a given keyboard key, if there is more than one key robot class considers the left key.*

mousePress() : *Presses mouse button based on the input value*

mousePress(1) : *Presses Primary key*

mousePress(2) : *Presses Secondary key*

mouseRelease() : *Releases the mouse button*

mouseMove() *: Moves mouse pointer to given screen coordinates, used long with Mouse Press most of the time.*

mouseWheel() : *Scrolls specific amount of notches (check the wheel of the mouse), if the given value is Positive scrolled down, in case of positive value wheel scrolled up*

delay(ms) : *Makes the robot action to sleep for the specified time. To catch any* ***InterruptedExceptions*** *that occur, Thread.sleep() may be used instead.*

AutoIT with Selenium

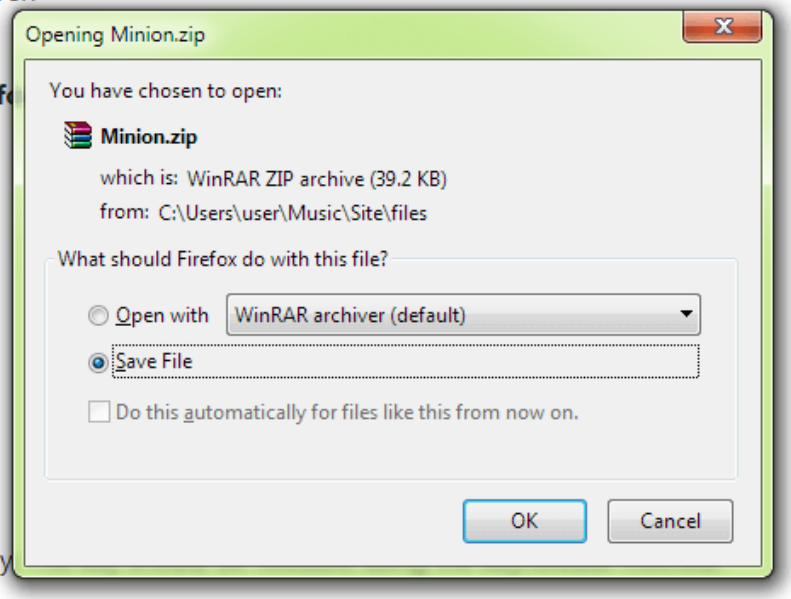
Handle Download popUp using Robot class in selenium

We can handle the download popup using **Robot** class, Just we have to press a couple of TABS and ENTER keys.

Avoid using the mouse with the robot class when you cannot find the exact location of the buttons. Location of the button or the elements may change due to the size of the screen we have for testing.

Example : Consider the Below download popup.

popups, upload popups). Robot Class used while we need to handle file upload and download activity using selenium webDriver.



Useful Robot Class Methods f

1. keyPress();

2. keyRelease();

3. mousePress();

4. mouseRelease();

5. mouseMove();

6. mouseWheel(wheelAmt);

7. delay(ms);

8. createScreenCapture();

keyPress() : Presses a given key

If there is more than one key present in keyboard for Key Code value(Like SHIFT, ALT, CTRL) will map to the left key.

To Handle above popUp, follow below steps:

> Once the popup is active, check which element is in an active state.

> In above popup **Save File** radio button is active

> Calculate how many tabs it takes to reach **Save File** or **OK** button

> It takes two TABS to reach the **Save File** button.

> Once we reach the required button, press EnterKey.

> Download link **Download Minion**

```
// PLEASE DO WRITE IMPORT STATEMENTS
public class DownloadUsingRobot {
   public static void main(String[] args) throws Exception {
   // set the geckodriver.exe property
         System.setProperty("webdriver.gecko.driver", "C:/~/geckodriver.exe");
         // open firefox
         WebDriver driver = new FirefoxDriver();
         driver.manage().timeouts().implicitlyWait(1, TimeUnit.MINUTES);
```

```
        // open web page
        driver.get("https://chercher.tech/files/minion.zip");
        Thread.sleep(3000);
        // create object to robot class
        Robot robot = new Robot();
        // press tab first time
        robot.keyPress(KeyEvent.VK_TAB);
        // press tab second time
        robot.keyPress(KeyEvent.VK_TAB);
        // press enter key
        robot.keyPress(KeyEvent.VK_ENTER);
    }
}
```

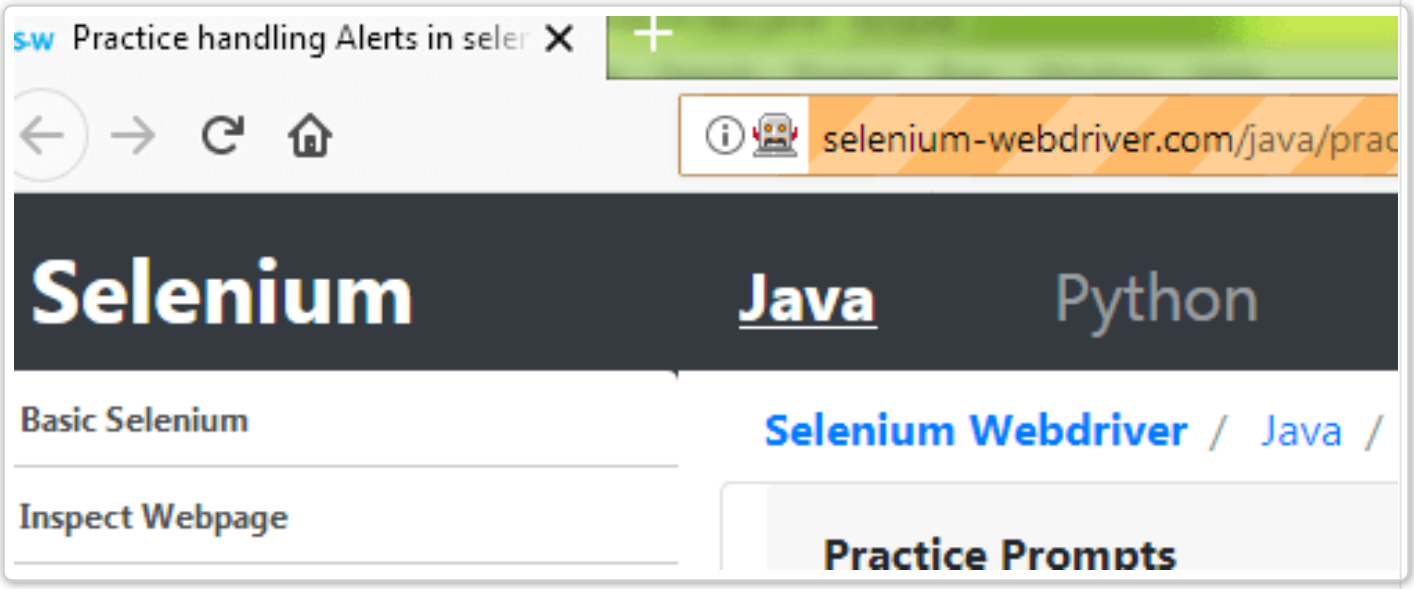
Capture Screenshot with Robot Class in Selenium

`createScreenCapture` takes a **screenshot** of the given rectangle shape. This method accepts a Rectangle type parameter.

Robot class takes a screenshot irrespective of application if the application is minimized robot class takes a screenshot of the desktop.

```
// PLEASE DO WRITE IMPORT STATEMENTS
public class RobotClass {
    public static void main(String[] args) throws Exception {
        // set the geckodriver.exe property
        System.setProperty("webdriver.gecko.driver", "C:/~/geckodriver.exe");
        // open firefox
        WebDriver driver = new FirefoxDriver();
        driver.manage().timeouts().implicitlyWait(1, TimeUnit.MINUTES);
        // open webpage
        driver.get("https://chercher.tech/practice/practice-pop-ups-selenium-webdriver");
        Robot robot = new Robot();
        int x = 10;
        int y = 10;
        int width = 500;
        int height = 200;
        Rectangle area = new Rectangle(x, y, width, height);
        BufferedImage bufferedImage = robot.createScreenCapture(area);
        File imageFile = new File("D:\Robot.png");
        ImageIO.write(bufferedImage, "png", imageFile);
    }
}
```

Output :



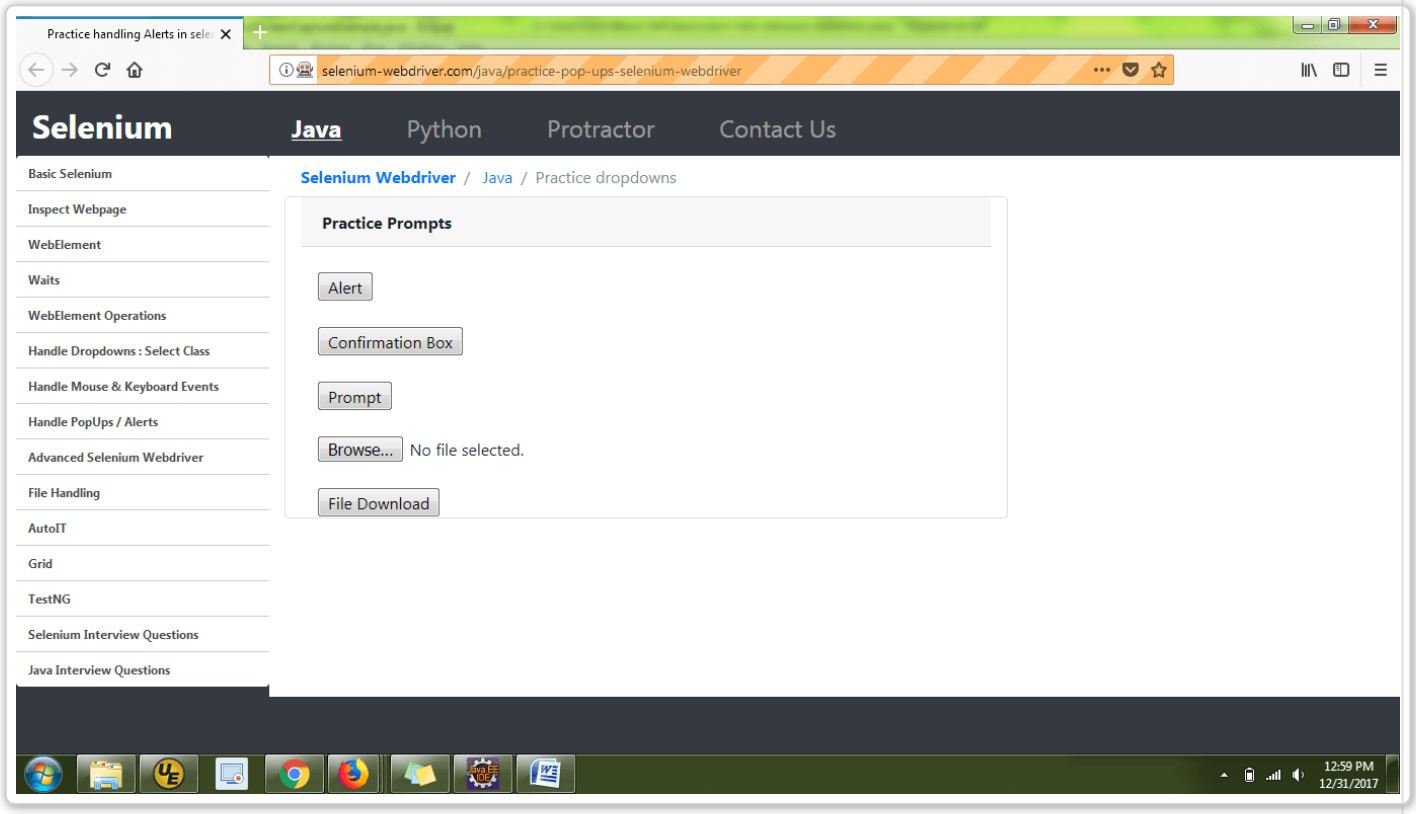
Full screenshot with Robot class in Selenium We can also take a full screenshot of the Desktop (any application which is active currently).

```
// PLEASE DO WRITE IMPORT STATEMENTS
public class FullScreenCaptureExample {
  public static void main(String[] args) throws Exception {

      // set the geckodriver.exe property
      System.setProperty("webdriver.gecko.driver", "C:/~/geckodriver.exe");
      // open firefox
      WebDriver driver = new FirefoxDriver();
      driver.manage().timeouts().implicitlyWait(1, TimeUnit.MINUTES);
      // open webpage
      driver.get("https://chercher.tech/java/practice-pop-ups-selenium-webdriver");
      // create object to robt class
      Robot robot = new Robot();
      // create rectangle for full screenshot
      Rectangle screenRect = new Rectangle(Toolkit.getDefaultToolkit().getScreenSize());
      // capture screen of the desktop
      BufferedImage screenFullImage = robot.createScreenCapture(screenRect);
      // save the screenshot to local system
      ImageIO.write(screenFullImage, "png", new File("D:\FullScreenshotRobot.png"));

      System.out.println("Full Desktop screenshot saved!");
  }
}
```

Output :



Take Full Screenshot in selenium

Difference between Actions class and Robot class

Actions 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.*

Actions class *does not affect parallel running but Robot class affects the parallel running as there is only one mouse connected to the system*

Actions class uses the native browser commands, ***Mouse class*** *does not use browser-based commands*

Actions class limited to the browser application, Robot class is can be used along with all the applications

Actions class *is from* ***org.openqa.selenium.interactions.Actions****, Robot class is from java.awt.Robot*

Difference between AutoIt and Robot class

AutoIT follows .au3 extension to execute it we need to convert the AutoIT script to a .exe file, In case of Robot class we do not have to package it, Robot class can be used along with the java code itself

AutoIT is platform dependent and Robot class is not platform dependent (but java dependent)

AutoIT works based on the window's locator but Robot class does not use locators, Robot class just enables the mouse and keyword actions

AutoIT also used along with OTP/UFT, Robot class used along with Java tools

AutoIT used for lengthy process automation as well, but Robot class used for few steps only

Xpath in Selenium

Exception with Robot Class

Robot class throws a few exceptions, we have to explicitly handle the exception with java.

AWTException - if the platform configuration does not allow low-level input control. This exception is always thrown when GraphicsEnvironment.isHeadless() returns true.

IllegalArgumentException - if the screen is not a screen GraphicsDevice.

SecurityException - if createRobot permission is not granted

There are more than **40 exceptions present** in selenium apart from awt exceptions

Read QR by taking as Screenshot in Selenium

Drawbacks of Robot Class in Automation

Though Robot class helps to handle window based popup, below drawbacks of robot class

MouseMove method is specific to the dimension of the device screen size.

Screenshot method is not controlled by a specific operation

Parallel running must be avoided as Robot class Executes actual mouse commands, so a computer cannot have two mice.

It may fail if permissions are not granted

It doesn't work well with selenium Grid/remote execution

Performs all actions on the active application on the desktop

When you use the keyPress event of this class then you should use keyRelease event also to perform the short keys on to the browser. For example, you want to click Ctrl+C if you forget to add keyRelease event then it will remain pressed and consume memory in the background

will remain pressed and consume memory in the background