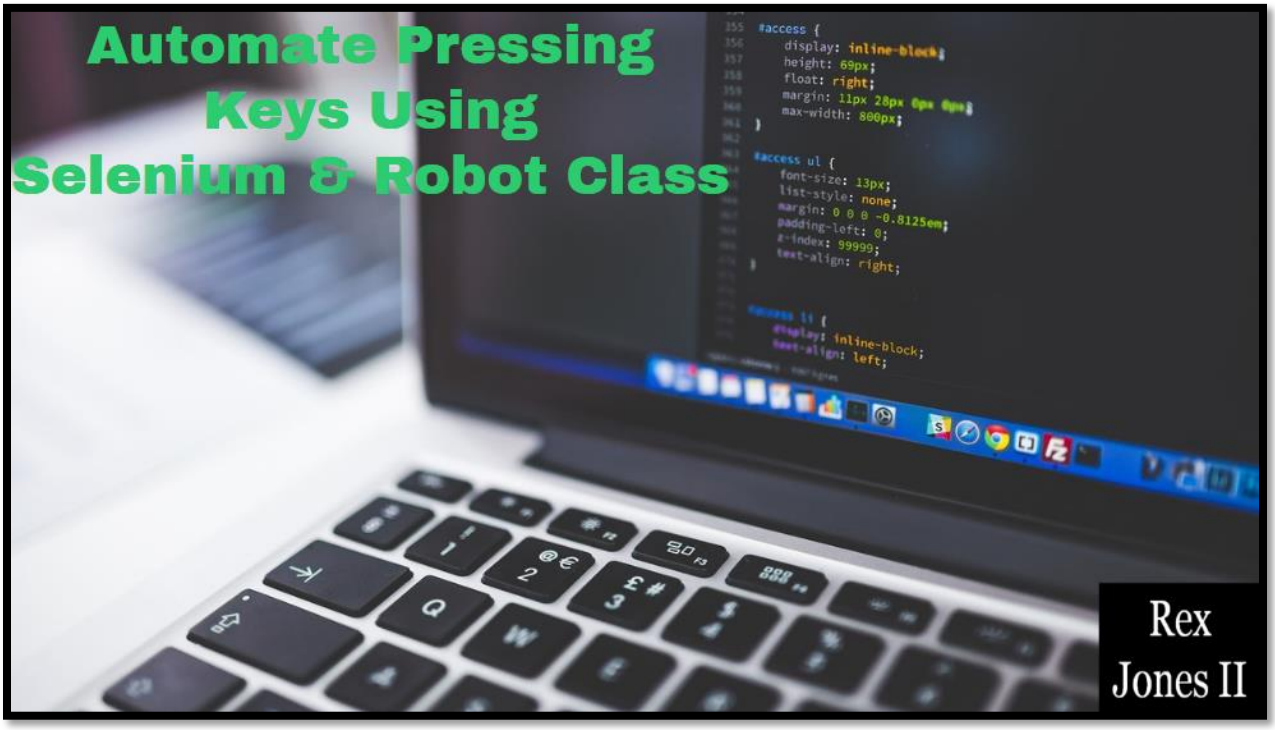
Rex Jones II

Automate Pressing Keys Using Selenium & Robot Class



Hello and Welcome. In this video, we are going to automate pressing keys on the keyboard. This jQuery application allows us to search but does not allow us to click the Search icon. Notice, how I typed api but nothing happens after clicking the icon. However, this other jQuery application is different. We can search for api, click the Search icon, and we see Results. In order to continue with our search on the first https://www.linkedin.com/m/logout/application, we must press the Enter key then we see some results.

Let's automate pressing the Enter key by inspecting the WebElement. The id attribute is not available so we are going to use the name attribute which has a value of s.

Rex
Jones II

```
Elements  Console  Sources  Network  Performance  Memory  Application  Security
<div class="menu-top-container">…</div>
<form method="get" class="searchform" action="//api.jquery.com/" _lpchecked="1">
▶<button type="submit" class="icon-search">…</button>
▼<label>
   <span class="visuallyhidden">Search jQuery API Documentation</span>
  ▼<span class="algolia-autocomplete" style="position: relative; display: inline-block;
  direction: ltr;">
      <input type="text" name="s" value placeholder="Search" class="ds-input" autocomplete=
      "off" spellcheck="false" role="combobox" aria-autocomplete="list" aria-expanded="false
      aria-label="search input" aria-owns="algolia-autocomplete-listbox-0" dir="auto" style=
      "position: relative; vertical-align: top;"> == $0
```

Let's go to Eclipse. We will use the Selenium's Keys class and the Robot class to press the Enter Key. Find the element by writing driver.findElement(By.*name*("s")).sendKeys("Test"); On the next line, we will press the Enter key. driver.findElement(By.*name*("s")).sendKeys

```
@Test
public void useKeysClass ()
{
   driver.findElement(By.name("s")).sendKeys(CharSequence... keysToSend
   driver.findElement(By.name("s")).sendKeys(keysToSend);
}
                                             keysToSend
                                             null
```

The Enter key is pressed using the sendKeys method. Now, we write Keys dot and there are a lot of methods: Arrow Down, Left, Right, Up, Backspace, Cancel but we want Enter. That's how we automate using the Selenium class.

```
values() : Keys[] - Keys
ADD : Keys - Keys
ALT : Keys - Keys
ARROW_DOWN : Keys - Keys
ARROW_LEFT : Keys - Keys
ARROW_RIGHT : Keys - Keys
ARROW_UP : Keys - Keys
BACK_SPACE : Keys - Keys
CANCEL : Keys - Keys
```

```
@Test
public void useKeysClass ()
{
   driver.findElement(By.name("s")).sendKeys("Test");
   driver.findElement(By.name("s")).sendKeys(Keys.ENTER);
}
```

Now, let's automate using the Robot class. We start with the Robot class robot = new Robot (); Import the class. The Robot class is used where control of the mouse and keyboard is needed. Add a throws declaration. Next, we find the element driver.findElement(By.*name*("s")).sendKeys This time, we are going to use ("json");

```
@Test
public void useRobotClass () throws Exception
{
    Robot robot = new Robot ();
    driver.findElement(By.name("s")).sendKeys("json");
```

robot dot and the Robot class has over 10 methods. We are going to press the Enter key so we select keypress then KeyEvent. KeyEvent is also a class. It indicates the keystroke. The keystroke we want is VK underscore ENTER. There's one more statement. We pressed the Enter key but have not released the Enter key. Therefore, our last statement is robot.keyRelease(KeyEvent.*VK_ENTER*);

```
@Test
public void useRobotClass () throws Exception
{
    Robot robot = new Robot ();
    driver.findElement(By.name("s")).sendKeys("json");

    robot.keyPress(KeyEvent.VK_ENTER);
    robot.keyRelease(KeyEvent.VK_ENTER);
}
```

Let's Run. Both Methods Passed. That's it for pressing keys on the keyboard using the Selenium class and Robot class.