A Tutorial for Using Selenium

Edit by Chi-Hui Lin

Abstract

Selenium provides a simple API for web developers to write functional/acceptance tests using Selenium Webdriver. Nevertheless, data scientists apply it to interact with websites and further scrape information from websites. This tutorial will introduce Selenium's features and further demonstrate them with simple projects. We provide our python programs on Github.

Comparison Between Requests and Selenium for Data Exploration

Requests do not interact with web applications and only deal with the HTML code. Hence, it performs faster on simple tasks, such as getting a web page/extracting its content, than Selenium. On the other hand, although Selenium spends more time extracting its content, it can scrape specific information quickly and interact with the websites, like click a button or search a keyword.

Moreover, we have demonstrated both tools' performance to visit the google webpage and return the objects. Requests return a "Response" object for users to handle information in HTML code in around 0.2 seconds. Selenium responds to a "WebDriver" object in around 0.8 seconds. It takes almost four times longer period to get a response. However, we can see that the returned objects are different. WebDriver indeed provides more functions. Users could use it to locate the element and then interact with it. We will explain it in the following paragraphs.

Installation Process of Selenium

First of all, we would like to run through the installation process to ensure our Selenium works appropriately. We are going to install the driver of the browser before installing the Python library of Selenium. Selenium requires the driver to use a web browser, such as Firefox, Chrome, or others.

- For Windows Users
    1. Check the version of the browser
        1. Chrome/Firefox > Open menu > About Google Chrome/Firefox
    2. Download the related driver for your browser, like Chrome or Firefox
    3. Set the Path variable
        1. Control Panel > System and Security > System > Advanced system settings > Advanced tab-Environment Variables Button
        2. Add the path to the driver directory to Path variables
        3. Restart the computer
- For Linux Users
    1. Set the PATH variable

    ```
    # show you which directory shell will search for executable files
    echo %PATH
    # If it does not show the working directory, move to next step.
    # If not, pass it.

    # Edit the PATH variables and applying the following code with
    # your working directory information inside.
    nano ~/.bashrc
    ```

        1. Add the following line of code in .bashrc file

```
#Add the following code in .bashrc file
export PATH="$PATH:/path/to/the/driver"
```

2. Install the browser driver
    1. Make sure the python is at least above version 3 and then install it with the following commands

    ```
    python3 -m pip3 install webdrivermanager
    webdrivermanager firefox --linkpath /path/to/the/driver
    ```

We can then install the python library through a single line of instruction and successfully set up the environment.

```
python -m pip install selenium
```

Steps to Use Selenium

People may use it in different ways. The tutorial aims for an introduction for beginners. Hence, we provide five primary steps to use Selenium.
First, we navigate to a website and wait for the web driver to load the page. If we skip the waiting step, we might fail to interact with web applications. We name objects on a webpage, such as a textbox or a button, by elements. Elements provide functions for users. Users need to locate it before doing interaction with it. We could locate it with multiple information, including CSS, XPath, class name, and interact with it with multiple provided methods. We will show the details in the following paragraph. Finally, we finish interacting with elements on the website by the following steps. There is one more step. We need to quit the browser, and the browser will close all windows.

- Basic 5 steps
    1. Navigate to a website
    2. Wait Until the page loading the element
    3. Locate the element
    4. Interact with the element
    5. Close the browser windows

We suggest users follow our tutorials and type lines of code using Python terminal to get familiar with Selenium. Then, we will edit a Python file to observe the cause of skipping the waiting step. We also provide code on Github.

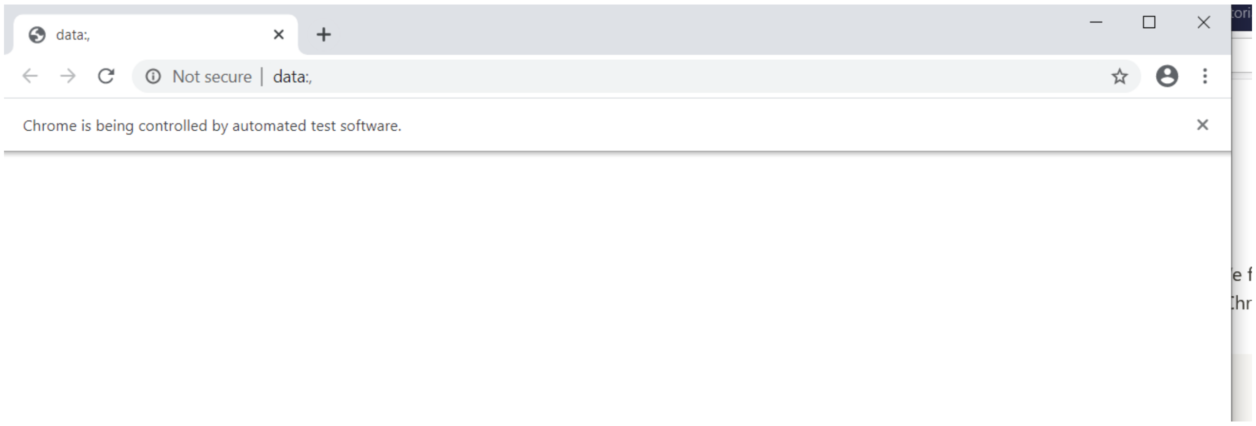
Navigate to A Website

Selenium could take merely three lines of code to navigate to a website. We firstly import webdriver from Selenium and assign a driver for future use. It could be the driver of Chrome, Firefox, or others. We show users multiple driver choices.

```
from selenium import webdriver

################################################################################
# STEP 1: Get to a web page
################################################################################
## For Chrome users:
driver=webdriver.Chrome()

## For Firefox users:
driver=webdriver.Firefox()

## For Chrome users using Operation System without Graphic Interface:
op = webdriver.ChromeOptions()
op.add_argument('headless')

## For Chrome users without setting PATH variables yet:
```

```
PATH = "D:\path\to\the\driver\chromedriver.exe"
driver = webdriver.Chrome(PATH)
```

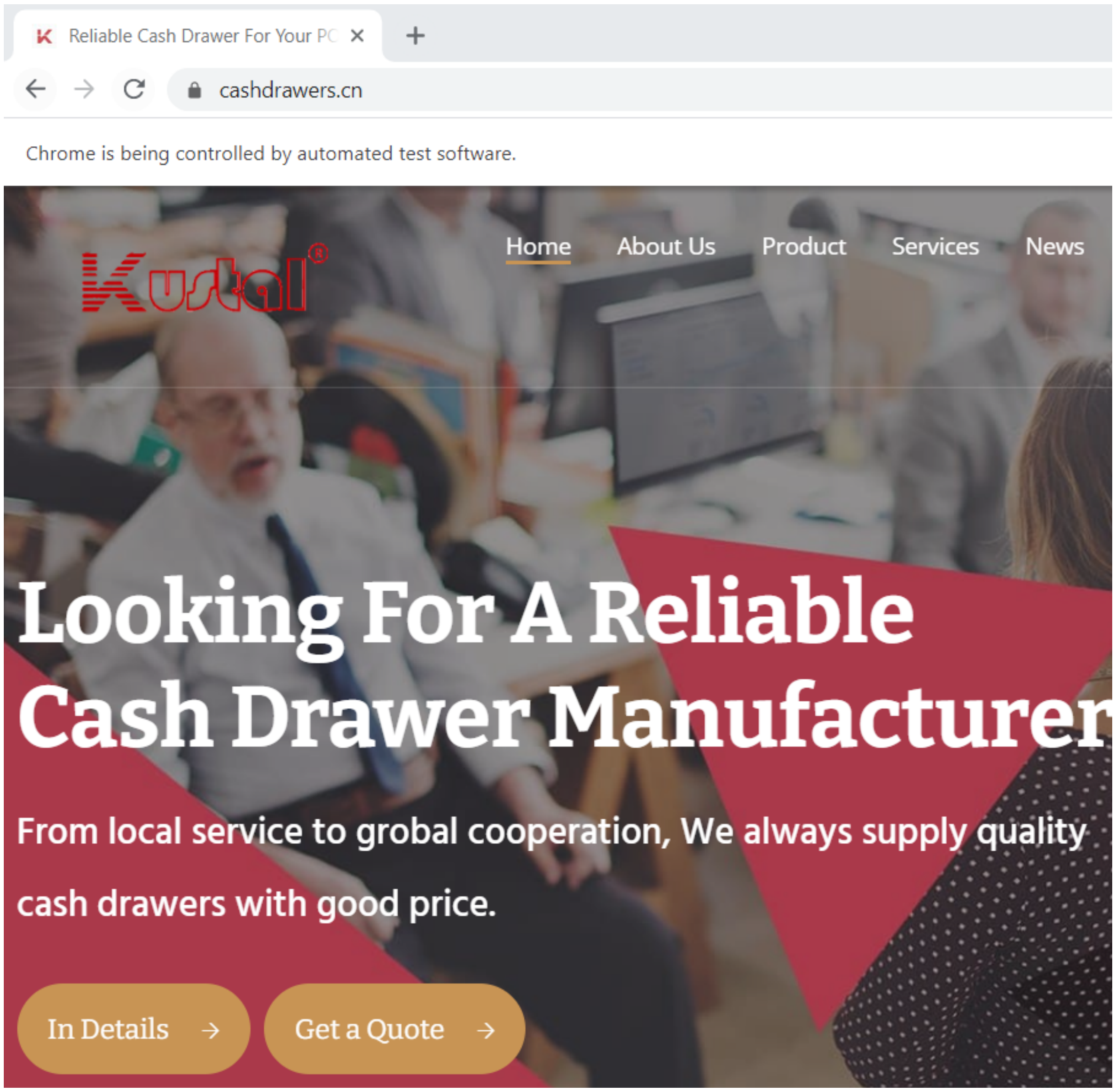
If we type programs correctly and the chosen browser is not headless, the browser will show up as shown in the following picture.



Afterward, the program could access the website. We want to mention that we choose to visit this website because it uses the AJAX technique. It will cause page loading issues if we skip the waiting stage. We could observe it later.

```
################################################################################
# STEP 1: Get to a web page
################################################################################
driver.get("https://www.cashdrawers.cn/")
```

The following page will show up.

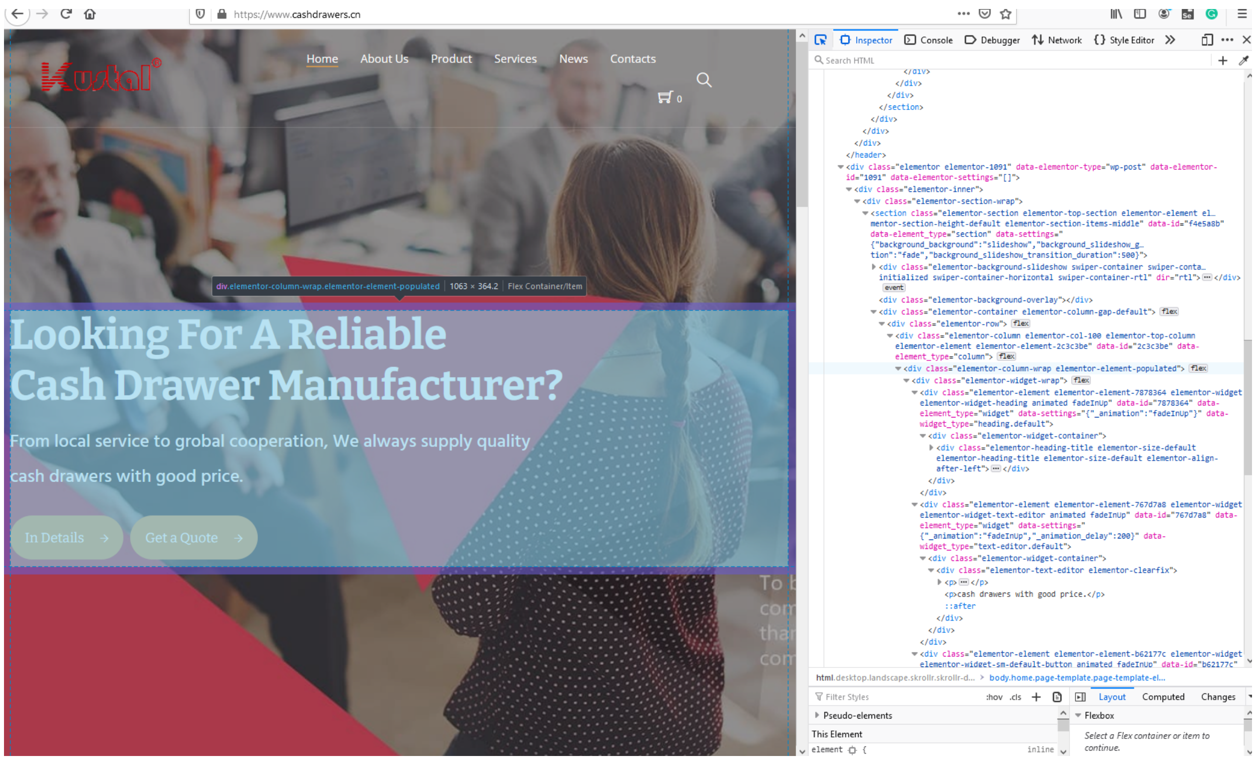

Reliable Cash Drawer For Your PC
cashdrawers.cn
Chrome is being controlled by automated test software.
Kustal
Home
About Us
Product
Services
News
Looking For A Reliable Cash Drawer Manufacturer
From local service to grobal cooperation, We always supply quality cash drawers with good price.
In Details
Get a Quote

Locate the Element

We want to click the "In Details" button in the lower part of the webpage image. Hence, the driver requires to locate the element in order to interact with it. Selenium provides the following methods to locate elements on a page. If we look for a specific element on a specific page, id, name, link_text are excellent choices. On the other hand, class_name could provide a handy way to locate multiple similar elements, like buttons. However, using XPath could provide us both. Locating an element by XPath is like finding an element using its location in the whole web structure. Most of the popular web pages do not alter their structures frequently. Hence, it is suitable to locate a particular element by XPath. Moreover, similar elements are often neighbors. Therefore, we can locate them by using a parent directory of them as the XPath.

```
# Cheet Sheet
# Selenium provides the following methods to locate elements in a page:
#   find_element_by_id
#   find_element_by_name
#   find_element_by_xpath
#   find_element_by_link_text
#   find_element_by_partial_link_text
#   find_element_by_tag_name
#   find_element_by_class_name
#   find_element_by_css_selector
# To find multiple elements (these methods will return a list):
#     find_elements_by_name
#     find_elements_by_xpath
#     find_elements_by_link_text
#     find_elements_by_partial_link_text
#     find_elements_by_tag_name
#     find_elements_by_class_name
#     find_elements_by_css_selector
```

Nevertheless, it is always challenging to find suitable attributes to locate the element if we inspect the element directly, as shown in the following image. We occasionally cannot find the attributes, such as id, name, or the chosen attribute, such as class_name, locate on others, instead of our target. Even though XPath has no these issues, it is hard to track if the page has much content.


https://www.cashdrawers.cn
Home
About Us
Product
Services
News
Contacts
div.elementor-column-wrap.elementor-element-populated | 1063 × 364.2 | Flex Container/Item
Looking For A Reliable Cash Drawer Manufacturer?
From local service to grobal cooperation, We always supply quality cash drawers with good price.
In Details
Get a Quote
Inspector
Console
Debugger
Network
Style Editor
Search HTML
Filter Styles
:hov
.cls
Layout
Computed
Changes
Pseudo-elements
This Element
Flexbox
Select a Flex container or item to continue.

```
            </div>
          </div>
        </section>
      </div>
    </div>
  </div>
</header>
<div class="elementor elementor-1091" data-elementor-type="wp-post" data-elementor-id="1091" data-elementor-settings="[]">
  <div class="elementor-inner">
    <div class="elementor-section-wrap">
      <section class="elementor-section elementor-top-section elementor-element el…mentor-section-height-default elementor-section-items-middle" data-id="f4e5a8b" data-element_type="section" data-settings="{"background_background":"slideshow","background_slideshow_g…tion":"fade","background_slideshow_transition_duration":500}">
        <div class="elementor-background-slideshow swiper-container swiper-conta… initialized swiper-container-horizontal swiper-container-rtl" dir="rtl"> ··· </div> event
        <div class="elementor-background-overlay"></div>
        <div class="elementor-container elementor-column-gap-default"> flex
          <div class="elementor-row"> flex
            <div class="elementor-column elementor-col-100 elementor-top-column elementor-element elementor-element-2c3c3be" data-id="2c3c3be" data-element_type="column"> flex
              <div class="elementor-column-wrap elementor-element-populated"> flex
                <div class="elementor-widget-wrap"> flex
                  <div class="elementor-element elementor-element-7878364 elementor-widget elementor-widget-heading animated fadeInUp" data-id="7878364" data-element_type="widget" data-settings="{"_animation":"fadeInUp"}" data-widget_type="heading.default">
                    <div class="elementor-widget-container">
                      <div class="elementor-heading-title elementor-size-default elementor-heading-title elementor-size-default elementor-align-after-left"> ··· </div>
                    </div>
                  </div>
                  <div class="elementor-element elementor-element-767d7a8 elementor-widget elementor-widget-text-editor animated fadeInUp" data-id="767d7a8" data-element_type="widget" data-settings="{"_animation":"fadeInUp","_animation_delay":200}" data-widget_type="text-editor.default">
                    <div class="elementor-widget-container">
                      <div class="elementor-text-editor elementor-clearfix">
                        <p> ··· </p>
                        <p>cash drawers with good price.</p>
                        ::after
                      </div>
                    </div>
                  </div>
                  <div class="elementor-element elementor-element-b62177c elementor-widget elementor-widget-sm-default-button animated fadeInUp" data-id="b62177c"
html.desktop.landscape.skrollr.skrollr-d… > body.home.page-template.page-template-el…
element { inline
```

Luckily, Selenium presented a tool for developers to track and test user actions on webpages. This tool could allow us to locate elements promptly. We provide a link here and will elaborate on it after we run through all five steps.

Now, we come back to this chapter's topic, finding the element "In Details" button. Before finding it by Selenium driver, we find it and click it manually. It then opens a new tab in the browser. Hence, we know the button indeed contains a link. We take a look at the cheat sheet and choose a useful attribute, link_text, for this case.

```
################################################################################
# STEP 3: Find the element
################################################################################
element = driver.find_element_by_link_text("In Details")
# print the element text in order to ensure we locate element correctly
print("Successfully locate the element by the Link Text: ", element.text)
```

If there is nothing wrong, the following message should show up, and we can ensure the driver has located the desired element.

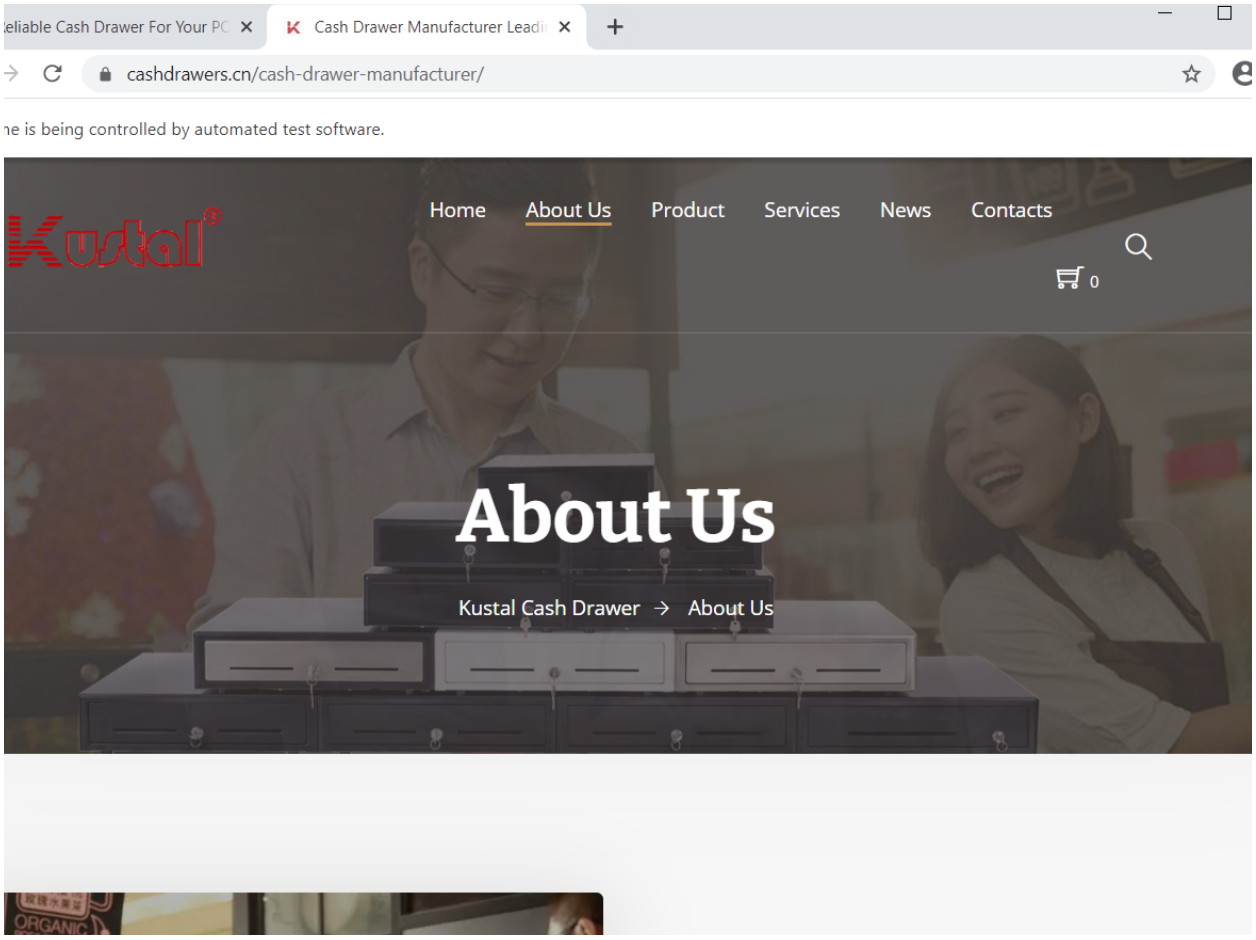
```
Successfully locate the element by the Link Text: In Details
```

Interact with the Element

We can move forward to the next step, interacting with the element. We click it!

```
################################################################################
# STEP 4 Interact with the element
################################################################################
element.click()
```

If following the steps correctly, the interaction will direct the driver to open another page, as shown in the following image. It is like how we do this manually.



eliable Cash Drawer For Your PC
Cash Drawer Manufacturer Leadi
cashdrawers.cn/cash-drawer-manufacturer/
ne is being controlled by automated test software.
Home
About Us
Product
Services
News
Contacts
0
About Us
Kustal Cash Drawer → About Us

Everything works fine even though we skip the waiting step. Some people might meet the issue if they execute whole programs directly, instead of using python terminal. We will discuss the cause of this issue later. For now, we will introduce the last step.

Close the Browser

We mainly suggest two methods regarding this. Users could choose to let the driver close the current tab or all tabs.

```
#Close the current tabs
driver.close()
# Close all tabs and then quit the browser
driver.quit()
```

Wait Until the page loading the element

Now, instead of executing the lines of code on the python terminal one by one, we execute a python program, including the following code. We could check whether it still performs without issues when skipping the "Wait" step.

```
from selenium import webdriver
import time
###############################################################################
# STEP 1: Get to a web page
###############################################################################
driver = webdriver.Chrome()
driver.get("https://www.cashdrawers.cn/")
###############################################################################
# STEP 3: Find the element
###############################################################################
element = driver.find_element_by_link_text("In Details")
print("Successfully locate the element by the Link Text: ", element.text)
###############################################################################
# STEP 4 Interact with the element
###############################################################################
element.click()
###############################################################################
# STEP 5 Close the browser
###############################################################################
time.sleep(5)
driver.quit()
```

The program is in trouble this time. Before it, we execute each line of code with a waiting period caused by human typing speed. However, we leave the program to execute all lines of code without delay in this case. Moreover, the page requires time to load the page. Hence, this experiment meets the following issue.

A ***NoSuchElementException*** occurs while executing the code, "***element = driver.find_element_by_link_text("In Details")***". It explains that there is no such an element with a link text "In Details". We know this is because we did not wait enough time for the page to load this element, but how can we avoid it?

```
(wdi) D:\Codes\wdi\Selenium_Scraping_Tutorial>python 5steps_tutorial.py

DevTools listening on ws://127.0.0.1:53429/devtools/browser/51a5d968-4090-4f09-83b7-c8daaefe9840
Traceback (most recent call last):
  File "D:\Codes\wdi\Selenium_Scraping_Tutorial\5steps_tutorial.py", line 39, in <module>
    element = driver.find_element_by_link_text("In Details")
  File "D:\Users\linja\anaconda3\envs\wdi\lib\site-packages\selenium\webdriver\remote\webdriver.py", line 428, in find_element_by_link_text
    return self.find_element(by=By.LINK_TEXT, value=link_text)
  File "D:\Users\linja\anaconda3\envs\wdi\lib\site-packages\selenium\webdriver\remote\webdriver.py", line 976, in find_element
    return self.execute(Command.FIND_ELEMENT, {
  File "D:\Users\linja\anaconda3\envs\wdi\lib\site-packages\selenium\webdriver\remote\webdriver.py", line 321, in execute
    self.error_handler.check_response(response)
  File "D:\Users\linja\anaconda3\envs\wdi\lib\site-packages\selenium\webdriver\remote\errorhandler.py", line 242, in check_response
    raise exception_class(message, screen, stacktrace)
selenium.common.exceptions.NoSuchElementException: Message: no such element: Unable to locate element: {"method":"link text","selector":"In
  (Session info: chrome=86.0.4240.198)
```

We could employ the wait functions to do this. There are two types of wait functions, explicit and implicit wait. Explicit wait allows users to define the specific condition to occur before locating the element. For instance, we could define the *condition as* the page showing the element, and the program will do nothing but wait until the condition occurs. Besides, the condition may not happen at all, so we also require defining the waiting period's upper limit.

On the other hand, implicit wait provides users to adjust the default waiting time when trying to find any elements not immediately available. In other words, applying it allows the program to wait a certain amount of time for the next attempt while it fails to locate an element. We suggest defining the implicit waiting time in an amount that will not overtly influence the program's efficiency, and employing an explicit wait function for a specific condition requires more time.

We show a simple example to add waiting functions in order to fix the issue, ***NoSuchElementException***.

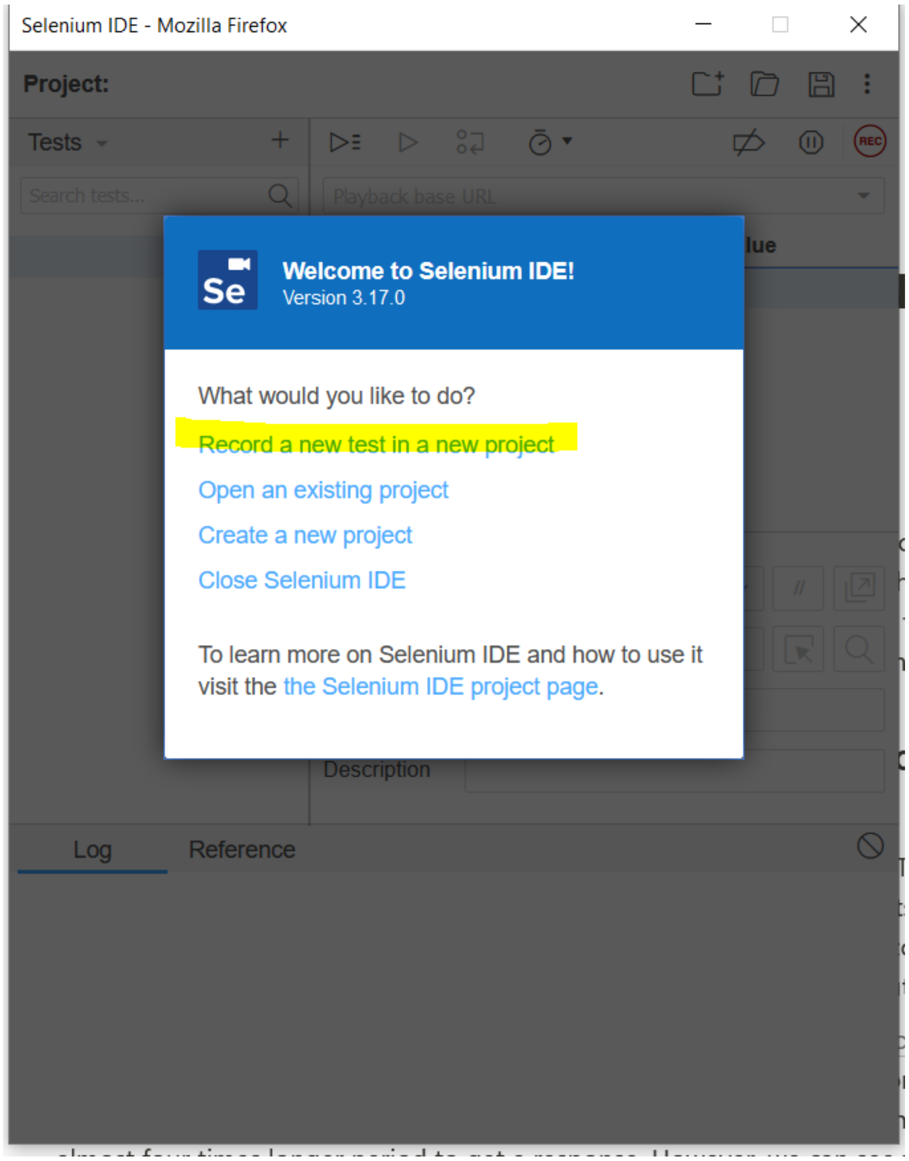
```
from selenium import webdriver
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
################################################################################
# STEP 1: Get to a web page
################################################################################
driver = webdriver.Chrome()
driver.get("https://www.cashdrawers.cn/")
# ################################################################################
# # STEP 2: Wait
# ################################################################################
# # Explicit Wait
WebDriverWait(driver, 10).until(
  EC.presence_of_element_located((By.LINK_TEXT, "In Details"))
)
# # Wait Until the news button is showing and then click
# # Implicit Wait
# driver.implicitly_wait(10)
################################################################################
# STEP 3: Find the element
################################################################################
element = driver.find_element_by_link_text("In Details")
print("Successfully locate the element by the Link Text: ", element.text)
################################################################################
# STEP 4 Interact with the element
################################################################################
element.click()
################################################################################
# STEP 5 Close the browser
################################################################################
time.sleep(5)
driver.quit()
```

The Trick to Use Selenium IDE: Find the key attribute to locate elements

Finding the attributes to locate the element could be complicated and a time-cost task. However, we noticed that using Selenium's IDE tool save us much time to do this. Users could learn to use it by following this additional tutorial.

First of all, we need to download the extension for our chosen browse, such as Firefox, Chrome, or others. Afterward, we can enable it by clicking the browser window's upper right, and we could see the following windows jumping out.

We then click *Record a new test in a new project*, as shown in the highlighted part. The IDE will then ask users to fill in the project's name and insert a URL. In order to follow the tutorial, please insert the provided URL.



Selenium IDE - Mozilla Firefox

Welcome to Selenium IDE!
Version 3.17.0

What would you like to do?

Record a new test in a new project

Open an existing project

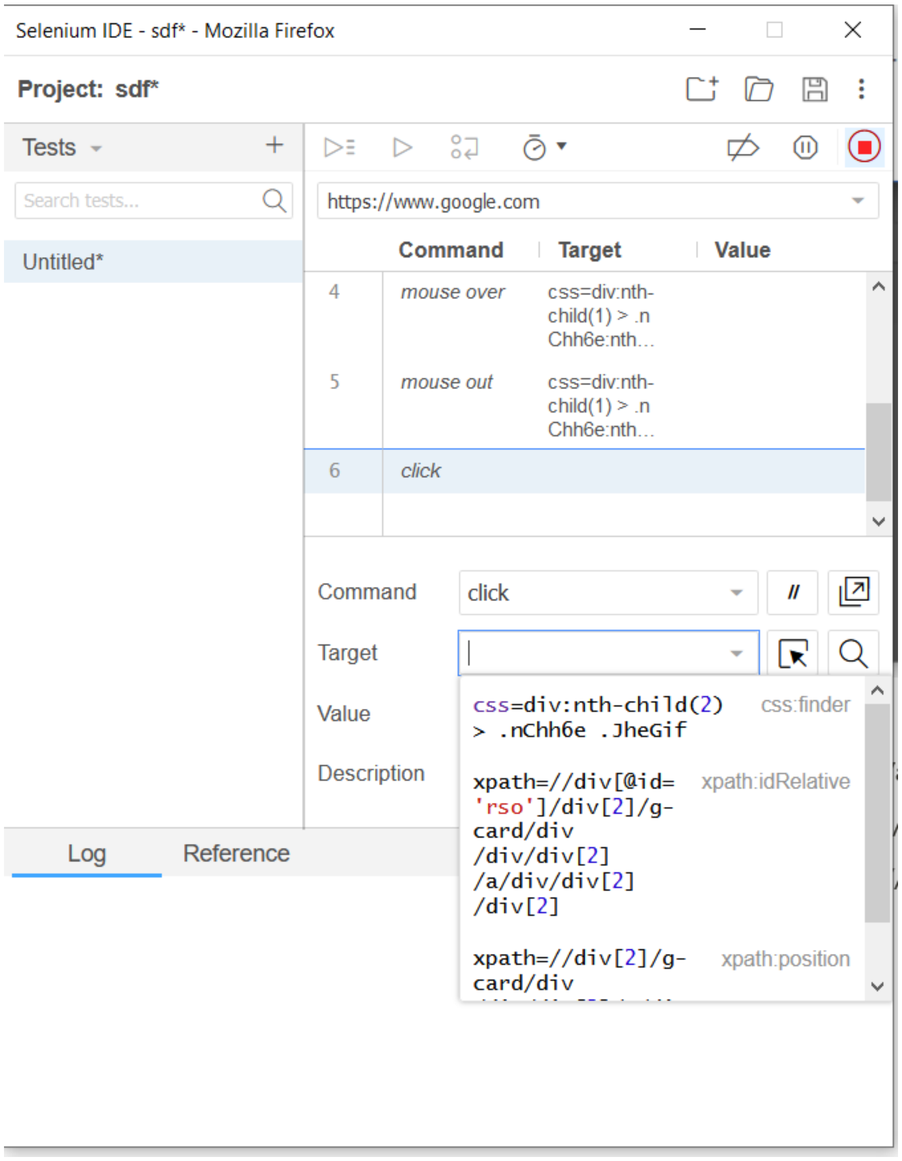
Create a new project

Close Selenium IDE

To learn more on Selenium IDE and how to use it visit the the Selenium IDE project page.

After following the steps correctly, we could finally use it to play with javascript applications, and IDE will record our interactions in the right part of the window. In the window, the *command* shows the interactions and targets the element with which users interact. Let us use it!

We click the first news link on the website, as shown in the following image, and the IDE window makes records, as shown in the following image. The command column contains a *click* action. We could then check the target column and observe the attributes of the clicked element. IDE presents us with several attributes to locate the element, such as XPath, CSS, or others.



Selenium IDE - sdf* - Mozilla Firefox

Project: sdf*

Tests

Untitled*

https://www.google.com

	Command	Target	Value
4	*mouse over*	css=div:nth-child(1) > .nChh6e:nth…	
5	*mouse out*	css=div:nth-child(1) > .nChh6e:nth…	
6	*click*		

Command: click

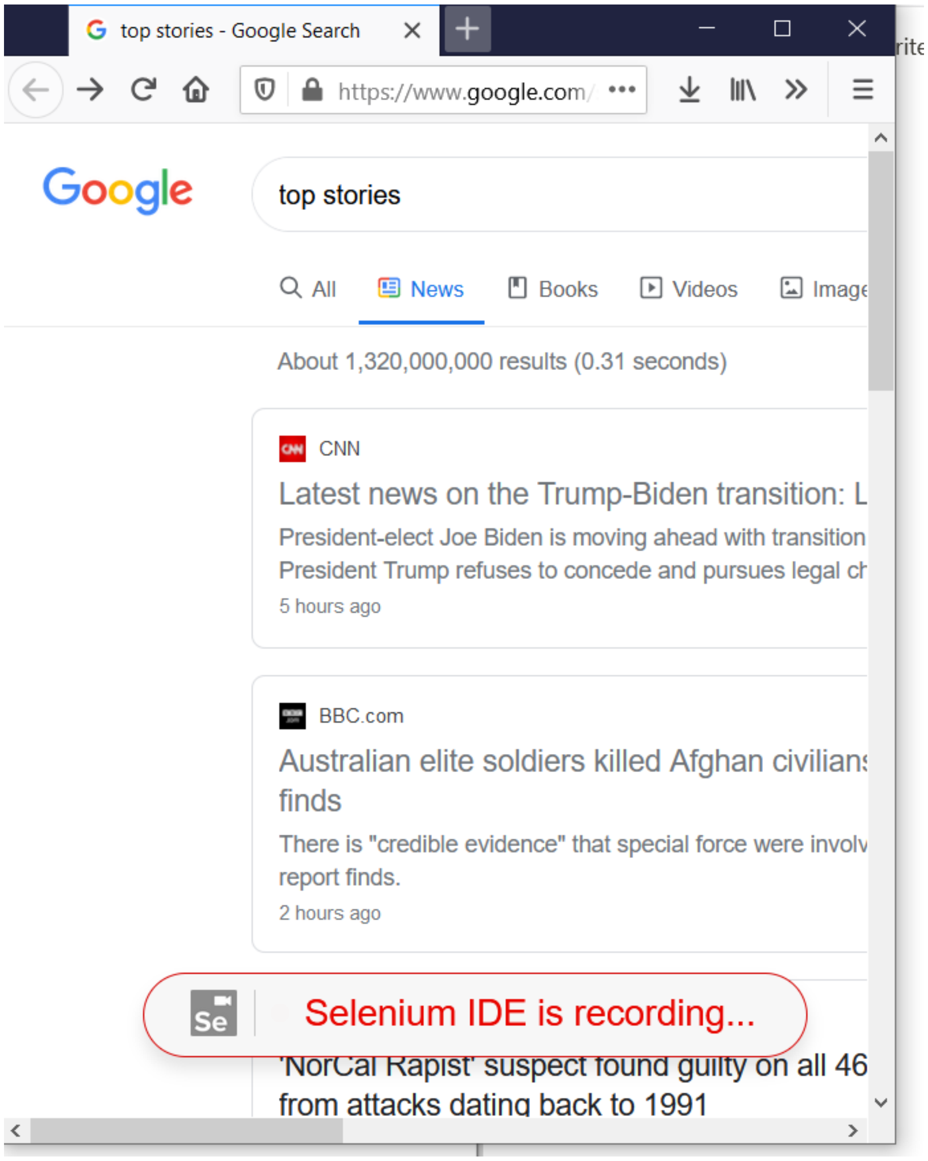
Target:

Value

Description

```
css=div:nth-child(2) > .nChh6e .JheGif          css:finder

xpath=//div[@id='rso']/div[2]/g-card/div/div/div[2]/a/div/div[2]/div[2]          xpath:idRelative

xpath=//div[2]/g-card/div          xpath:position
```

Log   Reference



About 1,320,000,000 results (0.31 seconds)

CNN
Latest news on the Trump-Biden transition: L
President-elect Joe Biden is moving ahead with transition
President Trump refuses to concede and pursues legal ch
5 hours ago

BBC.com
Australian elite soldiers killed Afghan civilians
finds
There is "credible evidence" that special force were involv
report finds.
2 hours ago

'NorCal Rapist' suspect found guilty on all 46
from attacks dating back to 1991

We have extracted all news information, including title, content, and links, using Selenium and Selenium IDE in our Github's code.

Some users may feel complicated to deal with several functions. Therefore, we provide a cheet sheet of important functions. Thank you very much for spending time reading this. We hope you are much more familiar with Selenium on data exploration.

```
# Cheet Sheet
# Selenium provides the following methods to locate elements in a page:
#   find_element_by_id
#   find_element_by_name
#   find_element_by_xpath
#   find_element_by_link_text
#   find_element_by_partial_link_text
#   find_element_by_tag_name
#   find_element_by_class_name
#   find_element_by_css_selector
# To find multiple elements (these methods will return a list):
#     find_elements_by_name
#     find_elements_by_xpath
#     find_elements_by_link_text
#     find_elements_by_partial_link_text
#     find_elements_by_tag_name
#     find_elements_by_class_name
#     find_elements_by_css_selector
# class selenium.webdriver.common.by.By[source]
#   Set of supported locator strategies.
#   CLASS_NAME = 'class name'
#   CSS_SELECTOR = 'css selector'
#   ID = 'id'
#   LINK_TEXT = 'link text'
#   NAME = 'name'
#   PARTIAL_LINK_TEXT = 'partial link text'
#   TAG_NAME = 'tag name'
#   XPATH = 'xpath'
# class selenium.webdriver.common.keys.Keys[source]
#   Set of special keys codes.
#   ADD = '\ue025'
#   ALT = '\ue00a'
#   ARROW_DOWN = '\ue015'
#   ARROW_LEFT = '\ue012'
#   ARROW_RIGHT = '\ue014'
#   ARROW_UP = '\ue013'
#   BACKSPACE = '\ue003'
#   BACK_SPACE = '\ue003'
#   CANCEL = '\ue001'
#   CLEAR = '\ue005'
#   COMMAND = '\ue03d'
#   CONTROL = '\ue009'
#   DECIMAL = '\ue028'
#   DELETE = '\ue017'
#   DIVIDE = '\ue029'
#   DOWN = '\ue015'
#   END = '\ue010'
#   ENTER = '\ue007'
#   EQUALS = '\ue019'
#   ESCAPE = '\ue00c'
#   F1 = '\ue031'
#   F10 = '\ue03a'
#   F11 = '\ue03b'
#   F12 = '\ue03c'
#   F2 = '\ue032'
#   F3 = '\ue033'
#   F4 = '\ue034'
#   F5 = '\ue035'
#   F6 = '\ue036'
#   F7 = '\ue037'
#   F8 = '\ue038'
#   F9 = '\ue039'
#   HELP = '\ue002'
#   HOME = '\ue011'
#   INSERT = '\ue016'
#   LEFT = '\ue012'
#   LEFT_ALT = '\ue00a'
#   LEFT_CONTROL = '\ue009'
#   LEFT_SHIFT = '\ue008'
#   META = '\ue03d'
#   MULTIPLY = '\ue024'
#   NULL = '\ue000'
#   NUMPAD0 = '\ue01a'
#   NUMPAD1 = '\ue01b'
#   NUMPAD2 = '\ue01c'
#   NUMPAD3 = '\ue01d'
#   NUMPAD4 = '\ue01e'
#   NUMPAD5 = '\ue01f'
#   NUMPAD6 = '\ue020'
#   NUMPAD7 = '\ue021'
#   NUMPAD8 = '\ue022'
#   NUMPAD9 = '\ue023'
#   PAGE_DOWN = '\ue00f'
#   PAGE_UP = '\ue00e'
#   PAUSE = '\ue00b'
#   RETURN = '\ue006'
#   RIGHT = '\ue014'
#   SEMICOLON = '\ue018'
#   SEPARATOR = '\ue026'
#   SHIFT = '\ue008'
#   SPACE = '\ue00d'
#   SUBTRACT = '\ue027'
#   TAB = '\ue004'
#   UP = '\ue013'

# Expected Condition
#   EC.title_is
#   title_contains
#   presence_of_element_located
#   visibility_of_element_located
#   visibility_of
#   presence_of_all_elements_located
#   text_to_be_present_in_element
#   text_to_be_present_in_element_value
#   frame_to_be_available_and_switch_to_it
#   invisibility_of_element_located
#   element_to_be_clickable
#   staleness_of
#   element_to_be_selected
#   element_located_to_be_selected
#   element_selection_state_to_be
#   element_located_selection_state_to_be
#   alert_is_present

# selenium.webdriver.remote.webelement
#   clear()[source]
#       Clears the text if it’s a text entry element.
#   click()[source]
#       Clicks the element.
#   send_keys(*value)
#     Simulates typing into the element.
#   screenshot(filename)
#     Saves a screenshot of the current element to a PNG image file
#   get_attribute(name)
#     Gets the given attribute or property of the element.
#   text
#       The text of the element.
```