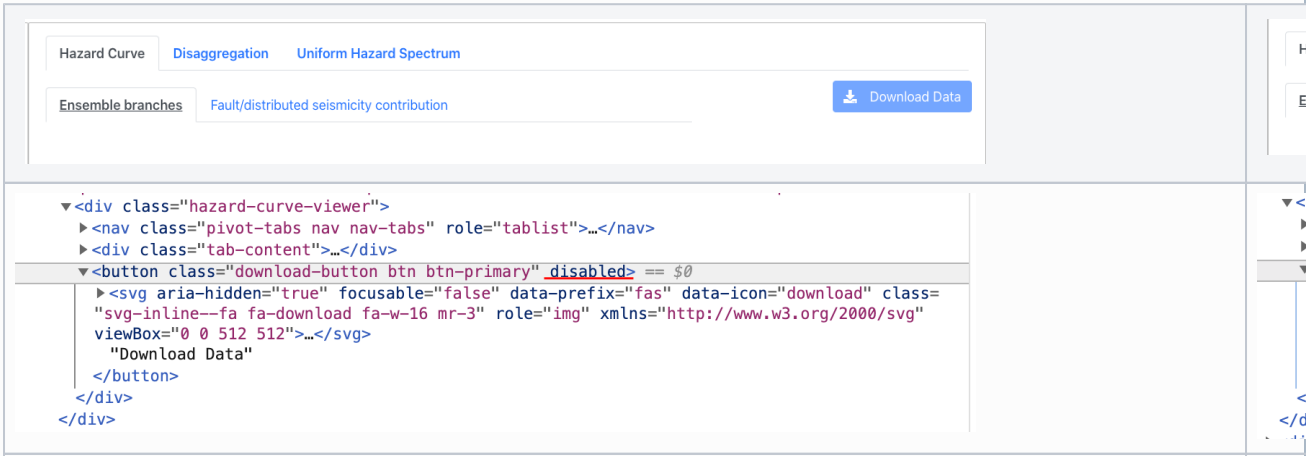
Writing an automated test with Selenium

Step-by-step guide

1. Execute Selenium IDE
2. Start recording and operate website as normal
3. Stop recording
4. Play the testing and see if the steps are correctly automated
5. Add conditional waits
6. Export to Pytest script
7. Final tests

Conditional Waits

Button

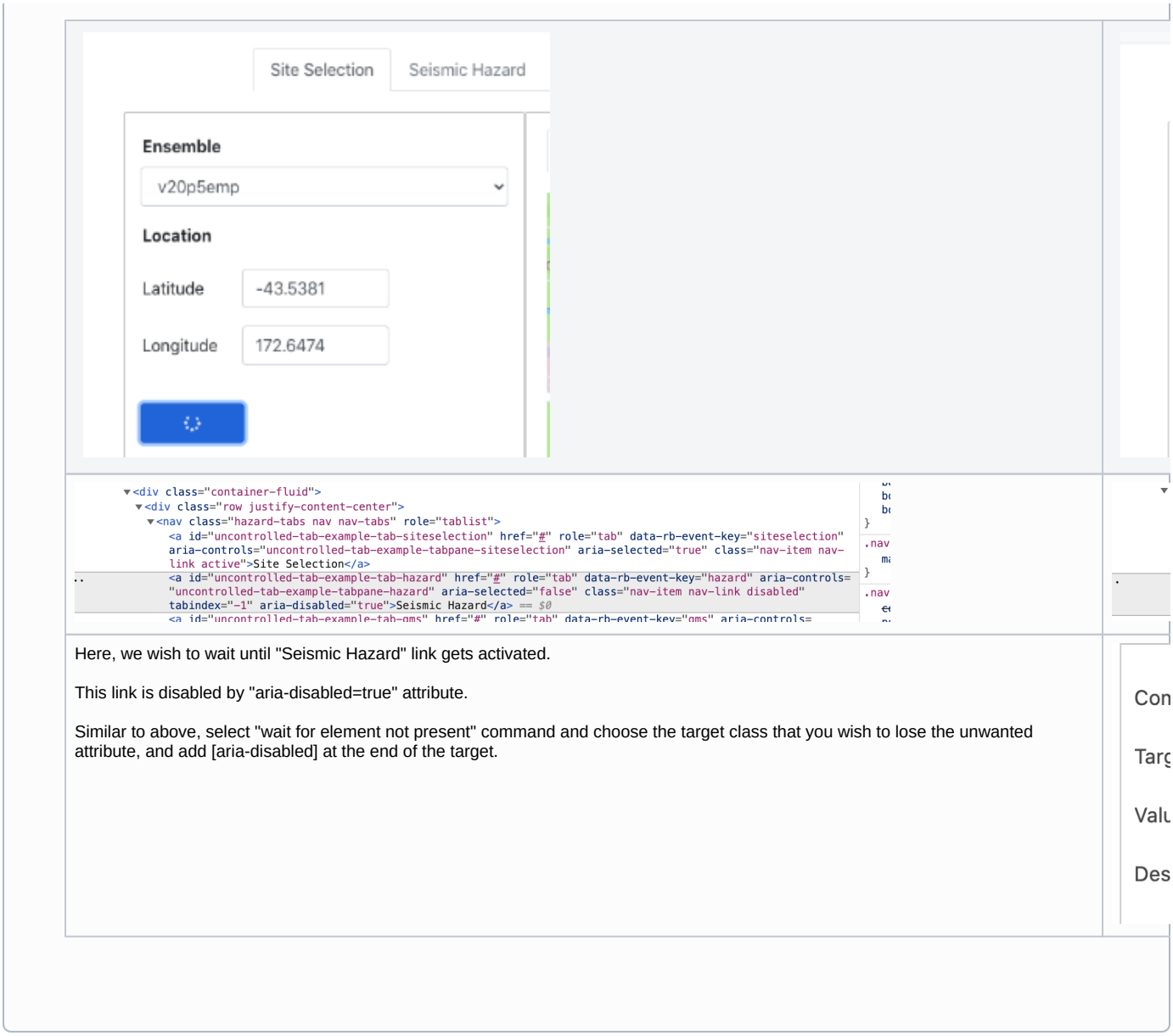


Button element has "disabled" attribute, which is removed when the button is activated.

Select "wait for element not present" command and choose the css class linked to the button, and add [disabled] to specify it is the unwanted element to move on to the next step.

Value 30000 is the timeout in milliseconds.

Text Link



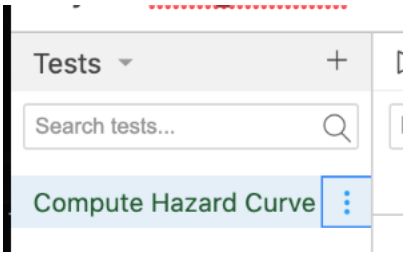
Here, we wish to wait until "Seismic Hazard" link gets activated.

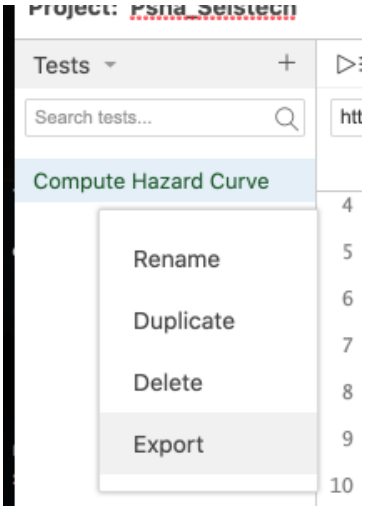
This link is disabled by "aria-disabled=true" attribute.

Similar to above, select "wait for element not present" command and choose the target class that you wish to lose the unwanted attribute, and add [aria-disabled] at the end of the target.

Pytest Export

Go to Tests mode, and click ... to export the test as a Pytest script.





The auto-generated Pytest code is not 100% though. Some unnecessary mouse clicks might have been recorded, which I commented out as below.

```
# Generated by Selenium IDE
import pytest
import time
import json
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support import expected_conditions
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities

class TestComputeHazardCurve():
  def setup_method(self, method):
    self.driver = webdriver.Chrome()
    self.vars = {}

  def teardown_method(self, method):
    self.driver.quit()

  def test_computeHazardCurve(self):
    self.driver.get("https://development.seistech.nz/")
    self.driver.set_window_size(1680, 1027)
    self.driver.find_element(By.ID, "qs-login-btn").click()
    self.driver.find_element(By.ID, "username").click()
    self.driver.find_element(By.ID, "username").send_keys("sungeunbae@live.com")
    self.driver.find_element(By.ID, "password").send_keys("XXXXX")
    self.driver.find_element(By.NAME, "action").click()
    self.driver.find_element(By.LINK_TEXT, "Hazard Analysis").click()
#    element = self.driver.find_element(By.LINK_TEXT, "Hazard Analysis")
#    actions = ActionChains(self.driver)
#    actions.move_to_element(element).perform()
    self.driver.execute_script("window.scrollTo(0,0)")
#    element = self.driver.find_element(By.CSS_SELECTOR, "body")
#    actions = ActionChains(self.driver)
#    actions.move_to_element(element, 0, 0).perform()
    self.driver.find_element(By.ID, "site-selection").click()
    time.sleep(3)
    WebDriverWait(self.driver, 10000).until(expected_conditions.invisibility_of_element_located((By.
CSS_SELECTOR, "#uncontrolled-tab-example-tab-hazard[aria-disabled]")))
    self.driver.find_element(By.XPATH, "//a[contains(.,\'Seismic Hazard\')]").click()
    self.driver.find_element(By.ID, "IMs").click()
    dropdown = self.driver.find_element(By.ID, "IMs")
    dropdown.find_element(By.XPATH, "//option[. = 'PGA']").click()
    self.driver.find_element(By.ID, "IMs").click()
    self.driver.find_element(By.ID, "im-select").click()
    WebDriverWait(self.driver, 30000).until(expected_conditions.invisibility_of_element_located((By.
CSS_SELECTOR, ".hazard-curve-viewer > .download-button[disabled]")))
    self.driver.find_element(By.CSS_SELECTOR, ".hazard-curve-viewer > .download-button").click()
    time.sleep(10)
```

When I ran this, however, I had an error message as below. It was supposed to "click" the "Hazard Analysis" link straight after the authentication, but it didn't wait for the link to show up.

```
(sung) sungbae@~/Downloads$ pytest ./test_computeHazardCurve.py
...
=================================================================== FAILURES
===============================================================================
_____________________________________________________ TestComputeHazardCurve.test_computeHazardCurve
_____________________________________________________

self = <test_computeHazardCurve.TestComputeHazardCurve object at 0x7fdb4b91ca58>

      def test_computeHazardCurve(self):
        self.driver.get("https://development.seistech.nz/")
        self.driver.set_window_size(1680, 1027)
        self.driver.find_element(By.ID, "qs-login-btn").click()
        self.driver.find_element(By.ID, "username").click()
        self.driver.find_element(By.ID, "username").send_keys("sungeunbae@live.com")
        self.driver.find_element(By.ID, "password").send_keys("XXXX")
        self.driver.find_element(By.NAME, "action").click()
>       self.driver.find_element(By.LINK_TEXT, "Hazard Analysis").click()


raise exception_class(message, screen, stacktrace, alert_text)
> raise exception_class(message, screen, stacktrace)
E selenium.common.exceptions.NoSuchElementException: Message: no such element: Unable to locate element:
{"method":"link text","selector":"Hazard Analysis"}
E (Session info: chrome=85.0.4183.83)
../sung/lib/python3.7/site-packages/selenium/webdriver/remote/errorhandler.py:242: NoSuchElementException
=========================================================================== 1 failed in 10.69s
===============================================================================
```

Add the following before line 29 to wait for the element to load first.

```
WebDriverWait(self.driver, 30000).until(expected_conditions.presence_of_element_located((By.XPATH, "//a[contains
(text(),\'Hazard Analysis\')]")))
```

Why do we need time.sleep(XXX)?

Lines 38 and 48 have time.sleep(XXX), which can be added from Selenium IDE using "pause" command, which waits for a specified number of seconds unconditionally.

This is a simple way to fix a timing issue. For example, line 38 is necessary because "Seismic Hazard" link can be already enabled if the user comes back to the "Site Selection" tab from the "Seismic Hazard" tab. When the "Set" button is clicked, there is a very short delay before "Seismic Hazard" link gets disabled. Without the time.sleep(3), it checks that the link is active, but when it clicks the link when it has already disabled - and the test gets stuck!

At the end of the test, Pytest kills the browser. The sleep at line 48 is necessary as the test immediately exits as soon as it "clicks" the download link - no download is actually taking place.

It is certainly better to check the existence of download file and check its correctness.