Automation Testing Code Documentation

The provided code is an example of using Selenium WebDriver with Python to automate the interaction with a web page. The code performs a series of test cases that involve selecting the source language and entering text in an online translation tool. Here's a documentation of the code:

1. Imports

The code starts by importing the necessary modules:

```python
import time
from selenium import webdriver
from selenium.webdriver.common.by import By
```

- The `time` module is imported to provide time-related functionality.
- The `webdriver` module from Selenium is imported to facilitate browser automation.
- The `By` class from `selenium.webdriver.common.by` is imported to allow selection of web elements using different criteria.

2. WebDriver Setup

Next, the code sets up the WebDriver by creating an instance of the Chrome WebDriver:

```python
driver = webdriver.Chrome()  # Replace with the actual path to the ChromeDriver executable
```

- The `webdriver.Chrome()` creates an instance of the Chrome WebDriver. Make sure to replace it with the actual path to the ChromeDriver executable on your system.

3. Test Case: Selecting Source Language and Entering Text

The main test case is defined as the function `test_select_source_language_and_enter_text()`. This function performs a series of steps to select the source language and enter text in the translation tool.

Here's an overview of the steps involved:

1. Opening the web page:

```python
driver.get("durgasree.site/index.html")  # Replace with the actual file path
```

- The `get()` method is used to navigate to the specified URL. Replace the URL with the actual file path or URL of the web page you want to test.

2. Waiting for elements and finding web elements:

```python
time.sleep(2)  # Wait for 2 seconds (adjust as needed)
source_language_select = driver.find_element(By.CSS_SELECTOR, ".controls .row.from select")
source_text_area = driver.find_element(By.CSS_SELECTOR, ".text-input .from-text")
```

- The `time.sleep()` function is used to introduce a delay to wait for the source language select element to be present on the page.

- The `find_element()` method is used to locate the source language select element and the source text area element on the page. It uses the `By.CSS_SELECTOR` strategy to locate elements based on CSS selectors.

3. Selecting the source language:

```python
source_language_code = "en"  # Replace with the desired source language code, for example, "en-GB" for English
source_language_select.click()
source_language_option = driver.find_element(By.CSS_SELECTOR, f".controls .row.from select option[value='{source_language_code}']")
source_language_option.click()
```

- The desired source language is specified by setting the `source_language_code` variable.

- The source language select element is clicked to open the dropdown.

- The source language option is found and clicked based on the specified language code.

4. Verifying the selected source language:

```python
selected_option = source_language_select.find_element(By.CSS_SELECTOR, "option:checked")
assert selected_option.get_attribute("value") == source_language_code
```

- The currently selected source language is verified by retrieving the selected option's value and comparing it to the expected language code.

5. Entering text in the source text area:

```python
text_to_translate = "Hello, how are you?"  # Replace with the desired text to translate
source_text_area.clear()
source_text_area.send_keys(text_to_translate)
```

- The desired text to translate is specified by setting the `text_to_translate` variable.
- The text area is cleared, and the text is entered using the `clear()` and `send_keys()` methods.

6. Additional steps for Test Case 1:

```python
time.sleep(5)  # Wait for 5 seconds (adjust as needed)
target_language_select = driver.find_element(By.CSS_SELECTOR, ".controls .row.to select")
translate_btn = driver.find_element(By.CSS_SELECTOR, "button")
```

- A delay is introduced to wait for additional elements to be present.
- The target language select element and the translate button are found.

7. Selecting the target language:

```python
target_language_code = "es"  # Replace with the desired target language code, for example, "es-ES" for Spanish
target_language_select.click()
target_language_option = driver.find_element(By.CSS_SELECTOR, f".controls .row.to select option[value='{target_language_code}']")
```

target_language_option.click()

```

- The desired target language is specified by setting the `target_language_code` variable.

- The target language select element is clicked to open the dropdown.

- The target language option is found and clicked based on the specified language code.

8. Verifying the selected target language:

```python

selected_option = target_language_select.find_element(By.CSS_SELECTOR, "option:checked")

assert selected_option.get_attribute("value") == target_language_code

```

- The currently selected target language is verified by retrieving the selected option's value and comparing it to the expected language code.

9. Clicking the translate button:

```python

translate_btn.click()

```

- The translate button is clicked to initiate the translation.

10. Additional steps for Test Case 2:

```python

time.sleep(5)  # Wait for 5 seconds (adjust as needed)

```

- A delay is introduced to allow time for the translation to complete or for any other necessary actions.

4. Running the Test Cases and Closing the WebDriver

The code then runs the `test_select_source_language_and_enter_text()` function to execute the defined test cases. Finally, the WebDriver is closed:

```python

# Run all the test cases

test_select_source_language_and_enter_text()

# Close the WebDriver

driver.quit()

```

It's important to note that to run this code successfully, you need to have the appropriate WebDriver installed (in this case, ChromeDriver) and provide the correct path to the WebDriver executable.

That's the documentation for the provided code.
```