Selenium IDE with Assessor+

Assessor+

- **Assessor+** is a Selenium IDE plugin that empowers **Capture & Replay** by supporting **PageObjects** automatic generation

- **Assessor+** can be used
  1. During the Selenium IDE recording phase, to **add annotations** to test cases about PageObjects and methods
  2. Once Selenium IDE have been recorded and exported, to automatically **translate the annotations** into Java PageObjects

Pre-requisites: Java, Selenium IDE & Others

- If not already present, install **Mozilla Firefox** from https://www.mozilla.org/en-US/firefox/new/
- If not already present, install **Java 8+** from https://www.java.com/it/
- If not already present, install **Maven** from https://maven.apache.org/install.html
- Install **Selenium IDE** from https://addons.mozilla.org/en-US/firefox/addon/selenium-ide/

Pre-requisites: Assessor+ Recorder

1. Clone repository from https://github.com/S4064172/AssessorExtension

2. Type "about:debugging" in Firefox browser

3. In the page that opens, click "This Firefox" link

4. Click Load Temporary Add-on... button

5. Browse to *manifest.json* in Assessor+ repo cloned in step 1

Note that steps 2-5 must be repeated any time the browser is closed

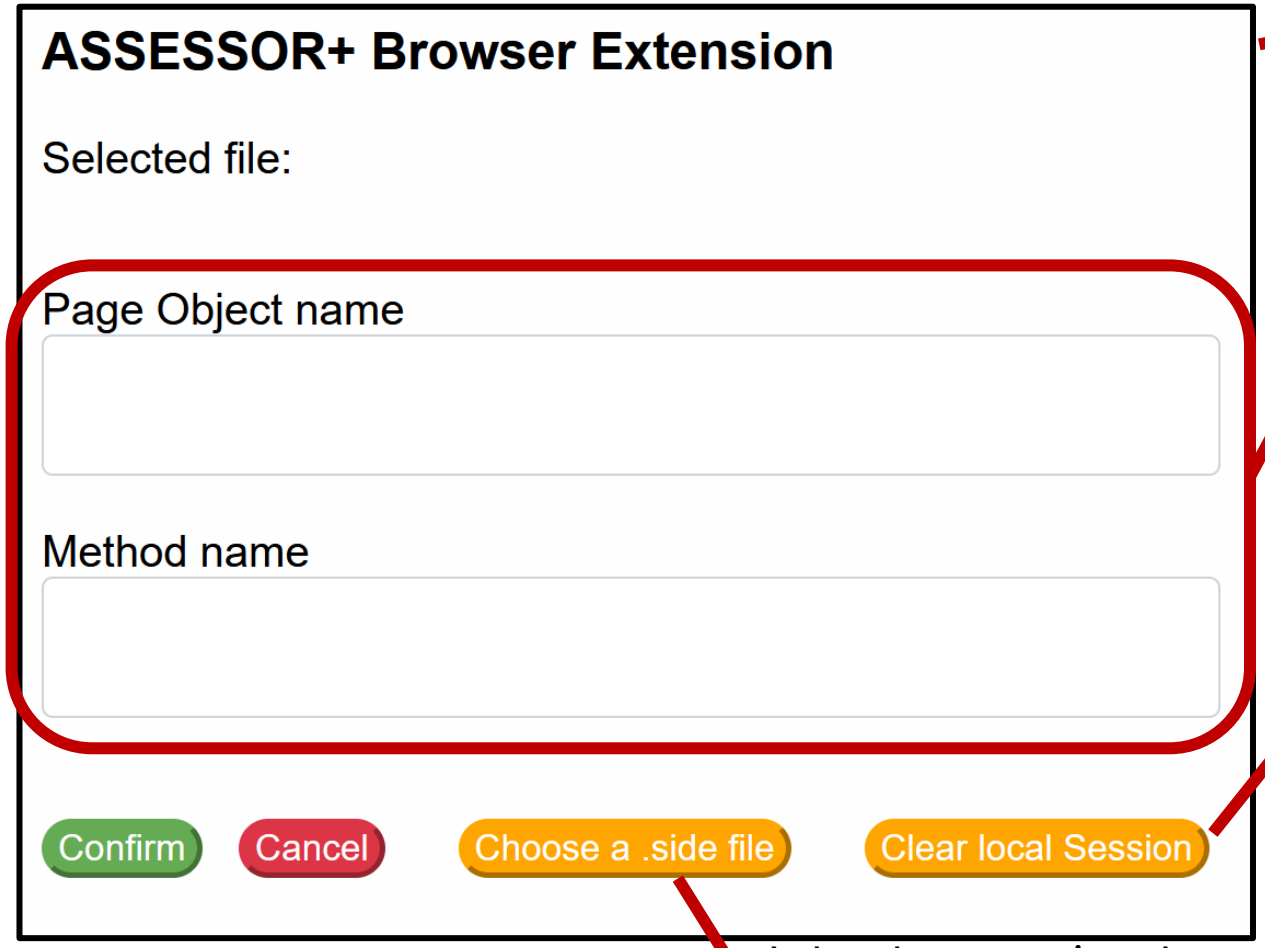
Pre-requisites: Assessor+ Generator

1. Clone repository from https://github.com/S4064172/Assessor

2. Compile project using **Maven**
   - From cmd inside the folder, run **mvn install**
   - A JAR file named **assessor-1.0.0-jar-with-dependencies** will be created under **/target** folder
   - We'll use the JAR later!

Assessor+ UI

This window is opened by pressing F8 when Selenium IDE is recording



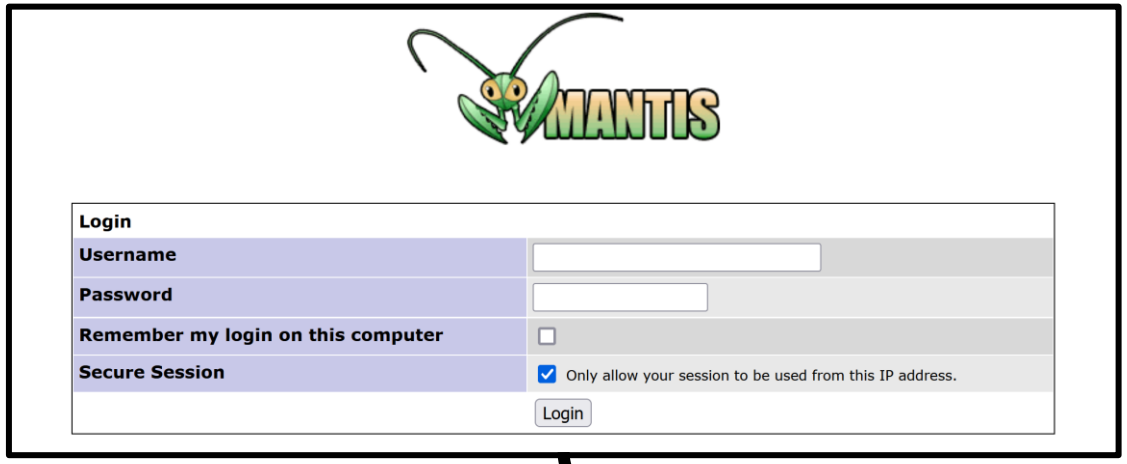
These fields are used to annotate a test case with PageObject and associated method names. Whatever follows the annotation, during the recording phase, and until the next annotation is added, is considered part of that PageObject and method.

It clears the references to test suites loaded for reuse purposes. It is strongly recommended to clean up Assessor+ state when test suites for different applications have to be created

It loads a previously saved test suite (.side extension) to refer to already defined Page Objects and methods for reuse purposes

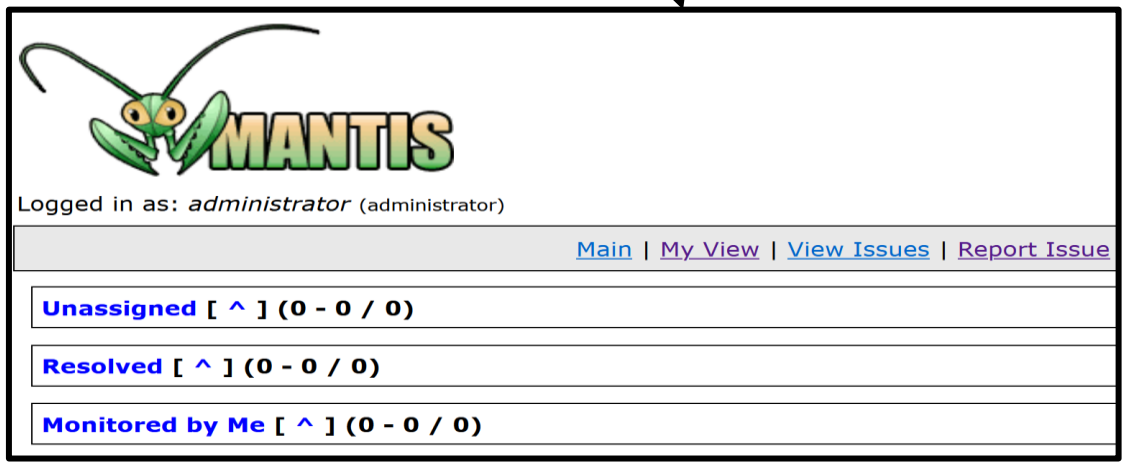
Assessor+ Annotations Example

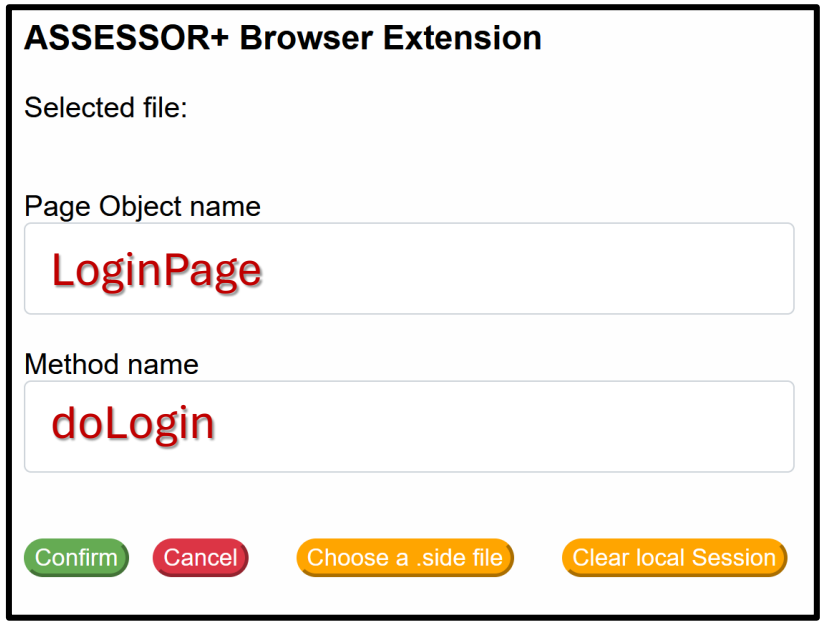
Login page



Log in

Home page





The annotation precedes the login recording

Testing Procedure with Assessor+

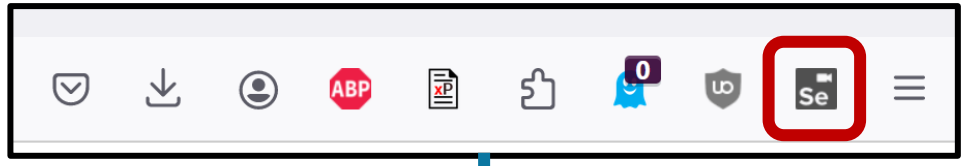
1. Create Selenium IDE Project
2. Create Test Case from a Gherkin
3. Start Recording
4. Add Annotation
5. Interact over the UI
6. Stop Recording
7. Export Test Suite
8. Generate Files from Test Suite
9. Import Files into Java Project

Steps 4-5 are repeated as long as needed to record POs/methods within a test case

Steps 2-6 are repeated as long as needed to record test cases

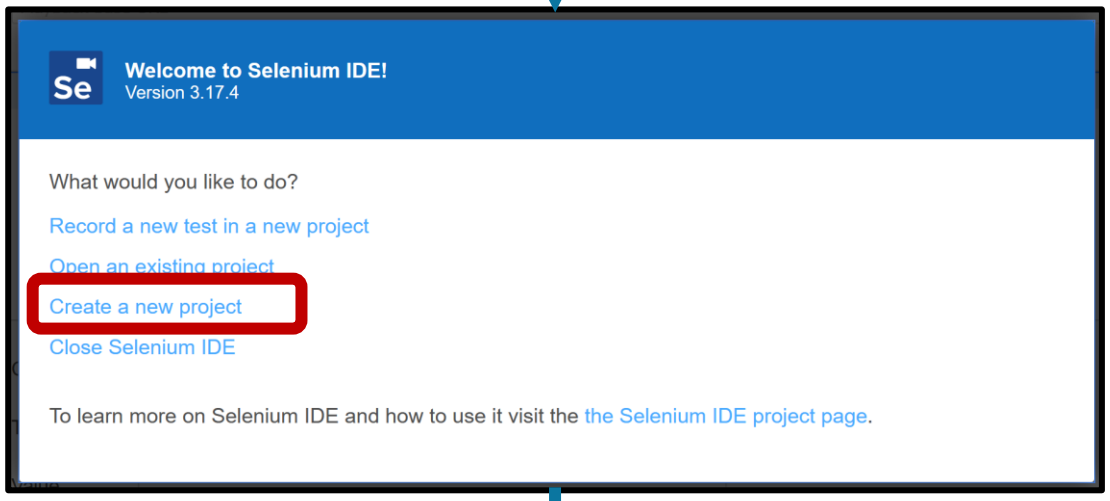
Steps in red require Assessor+

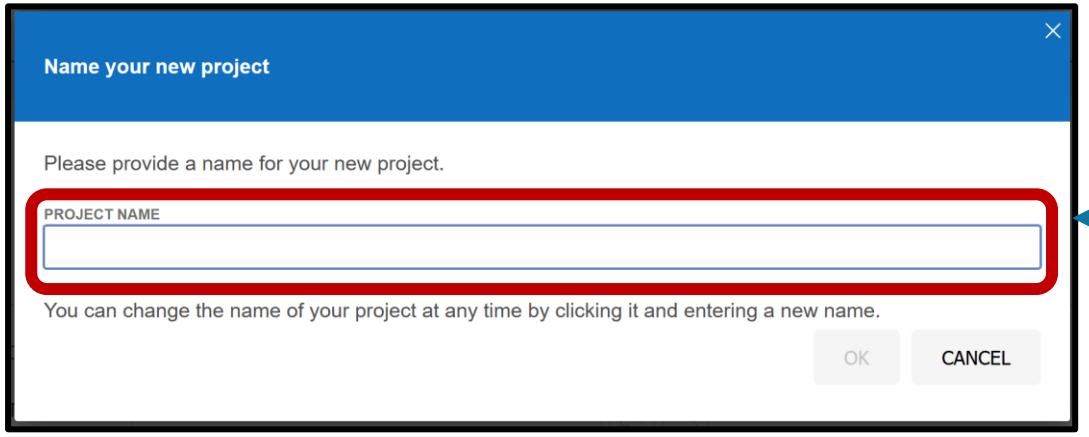
Create Selenium IDE Project (1)



1. Open Selenium IDE from browser

2. Select "Create a new project" (or open an existing project if you already produced some tests before)

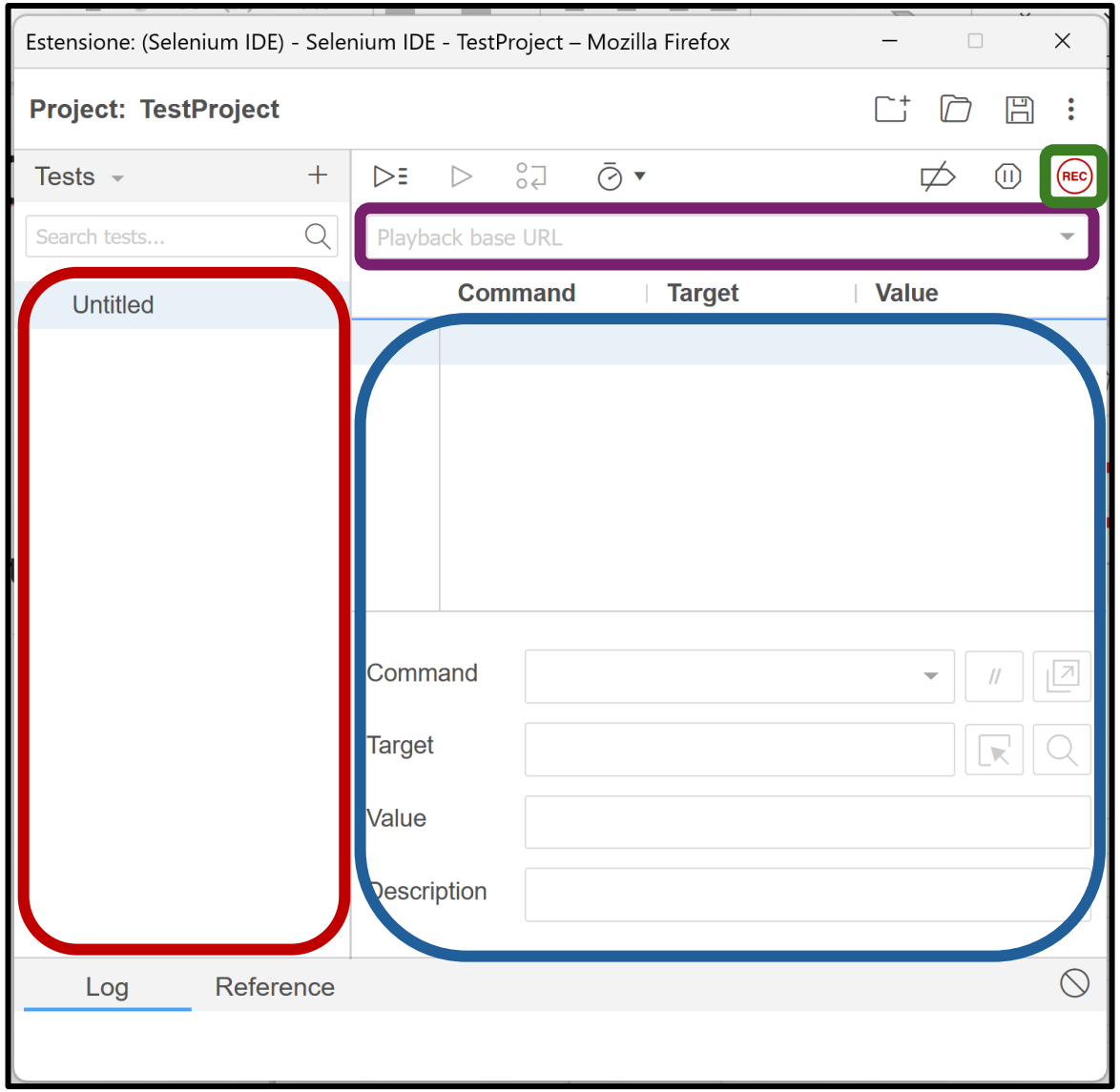




3. Give it a name and confirm

Create Selenium IDE Project (2)

Test cases panel



Start/stop recording button

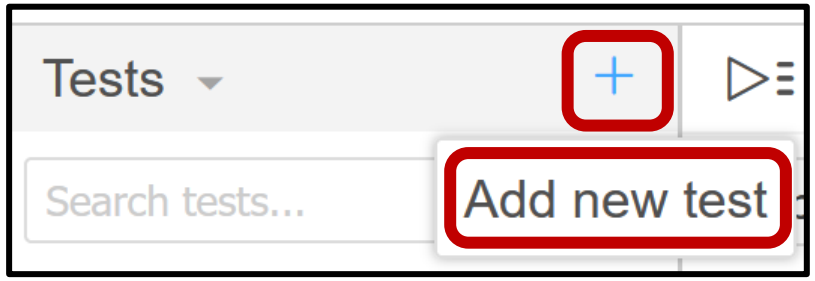
Base URL of the Web app under test (e.g., https://localhost/myapp)

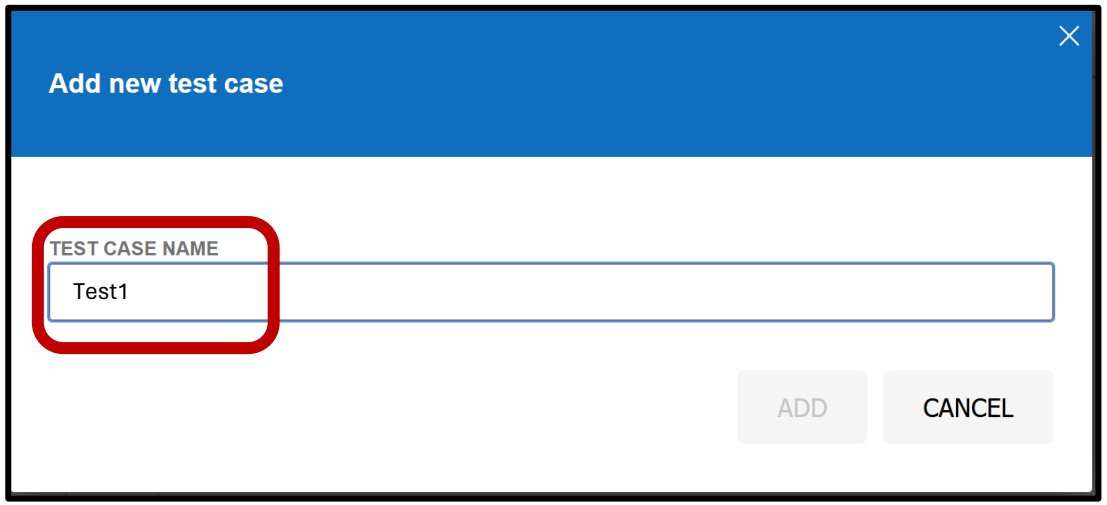
Selenese instructions (triples) panel:

- **Command** (the action to perform, e.g., type)
- **Target** (the Web element target of the command, e.g., id=«username»)
- **Value** (the optional value required by the command, e.g., «administrator»)

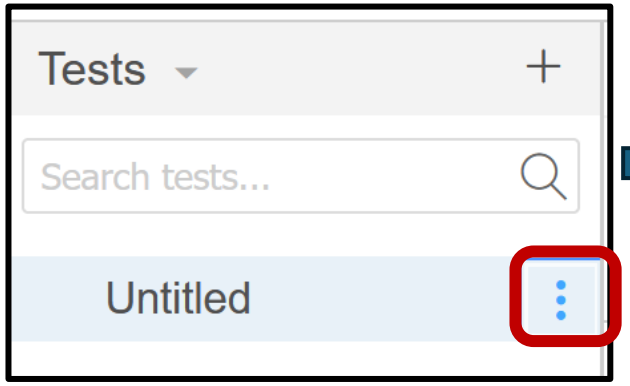
Create Test Case

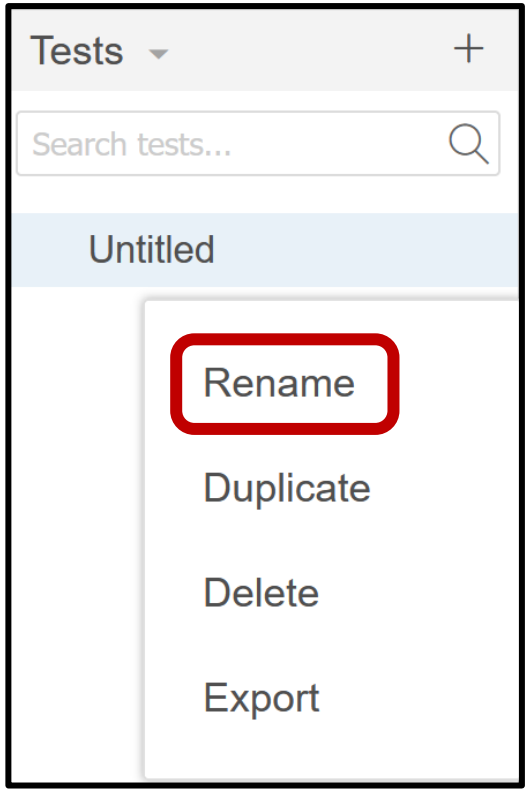
Add test case

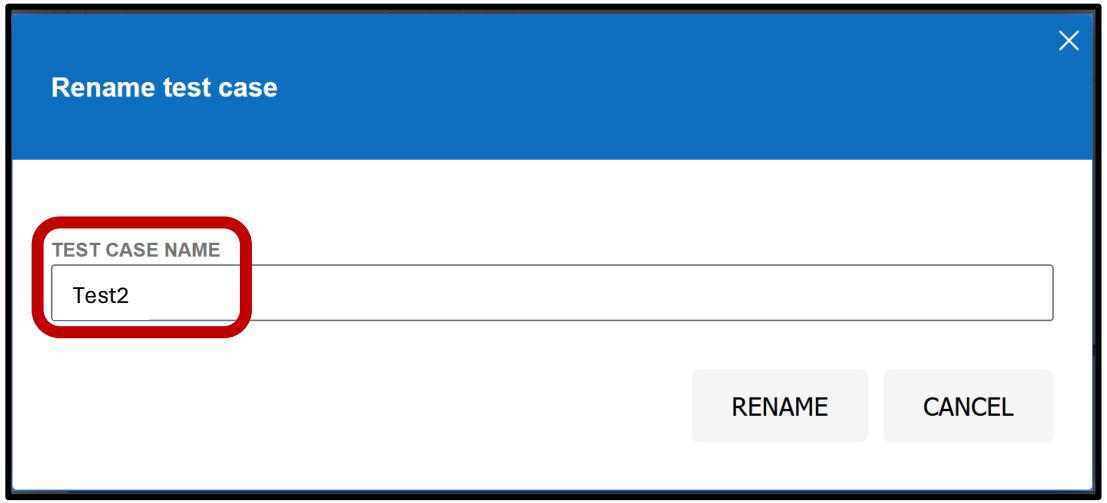




Rename test case



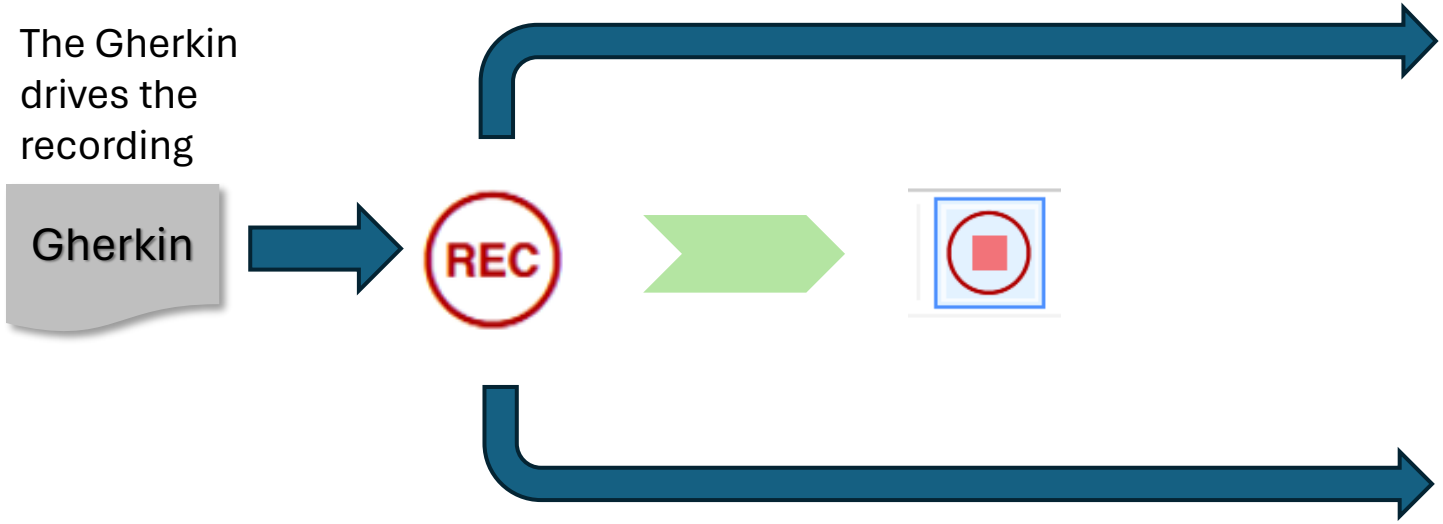


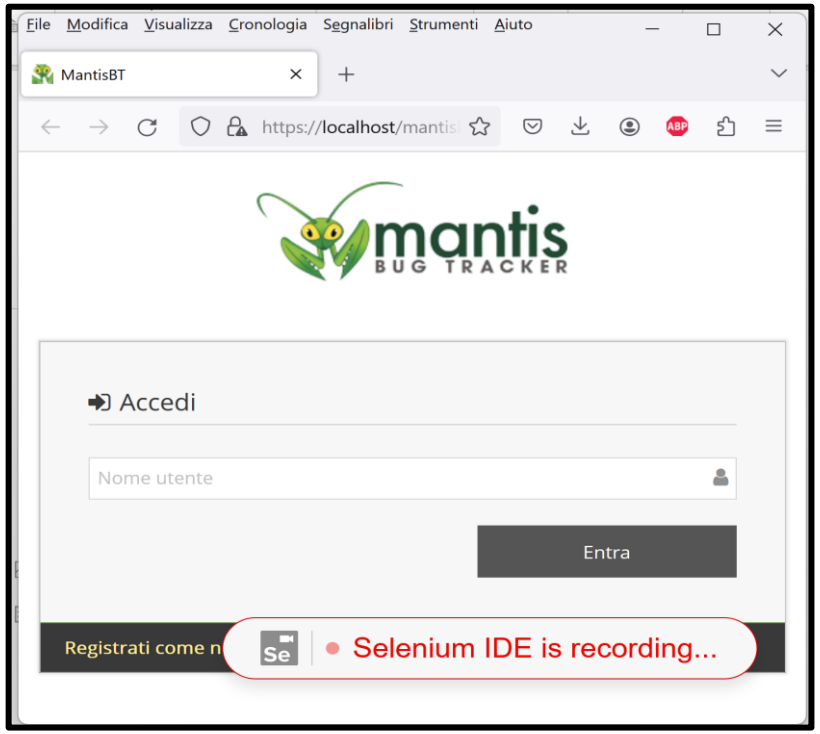


Start/Stop Recording

Clicking the **REC button** will activate the recording mode and open the browser **<u>at the page set as base URL</u>**

The Gherkin drives the recording



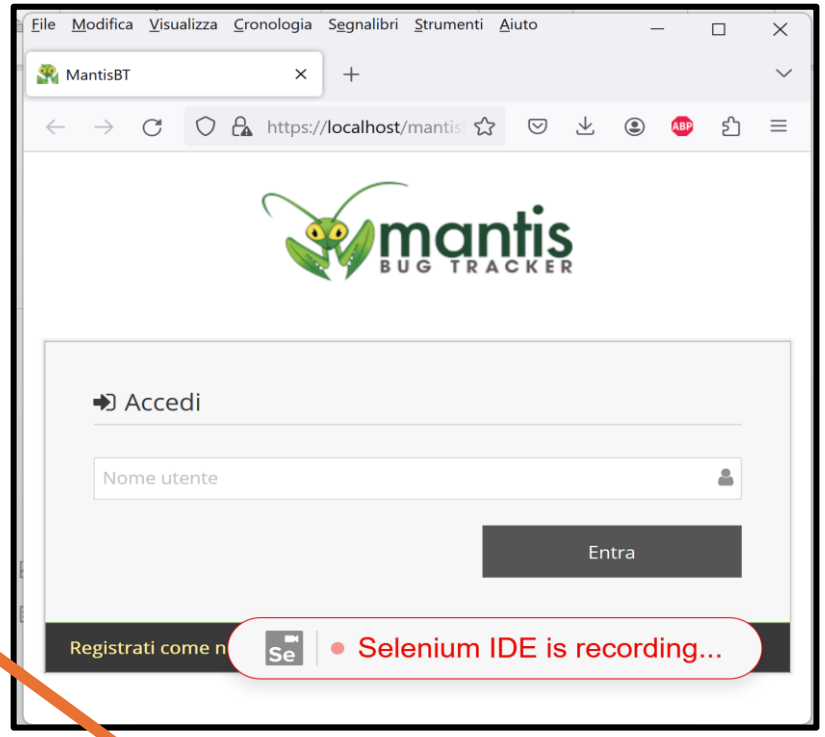


Selenium IDE will add the **open command** that reflects the opening of the browser and a **set window size command**. From now on, Selenium IDE will record any interaction over the UI

	Command	Target	Value
1	*open*	https://localhost/mantisbt/login_page.php	
2	*set window size*	599x579	

Start/Stop Recording

Clicking the **REC button** will activate the recording mode and open the browser **<u>at the page set as base URL</u>**



The Gherkin drives the recording

Gherkin

Sometimes a test might not add the URL to open as target (set it manually if that is the case)

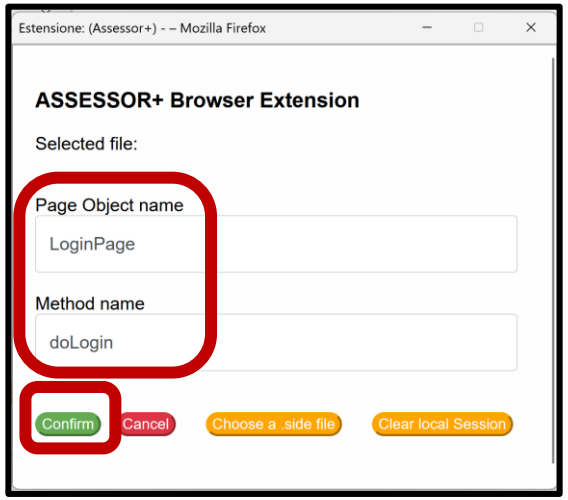
	Command	Target	Value
1	*open*	https://localhost/mantisbt/login_page.php	
2	*set window size*	599x579	

Selenium IDE will add the **open command** that reflects the opening of the browser and a **set window size command**. From now on, Selenium IDE will record any interaction over the UI

Add Annotation + Interact over the UI

F8 to open this window the first time

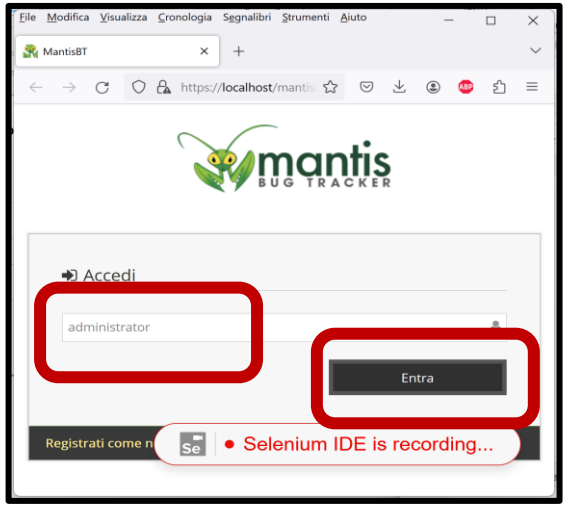
Add Annotation



3	*echo*	{ASSESSOR}:LoginPage:doLogin

Entering PO and method names and clicking Confirm will add an annotation indicating the begin of the PO method

Interactions over UI

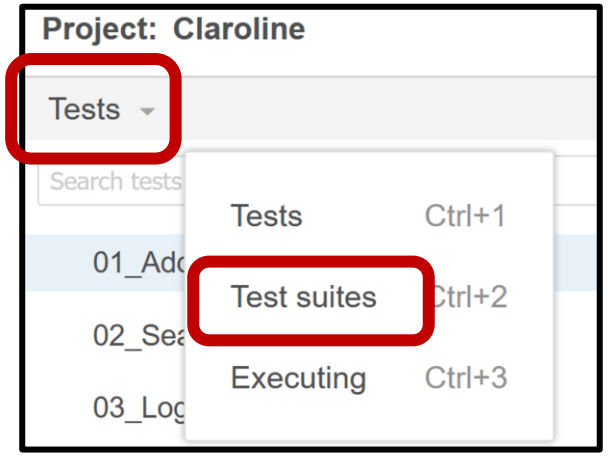


4	*click*	id=username	
5	*type*	id=username	administrator
6	*click*	css=.width-40	

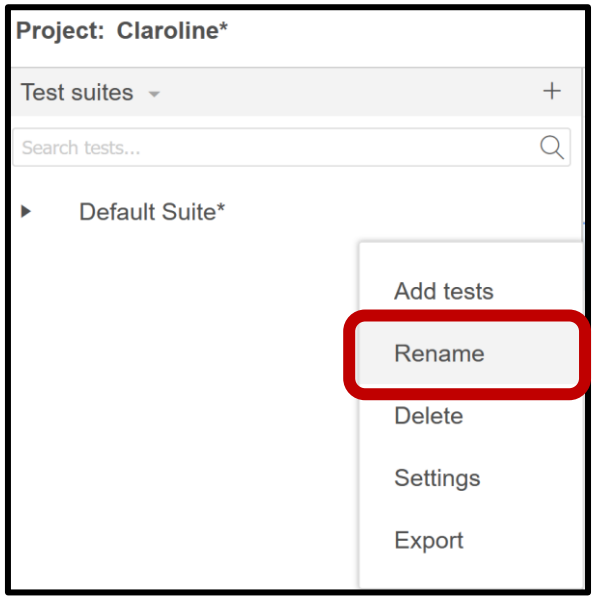
Any following interaction over the UI will be recorded and encapsulated within that PageObject method

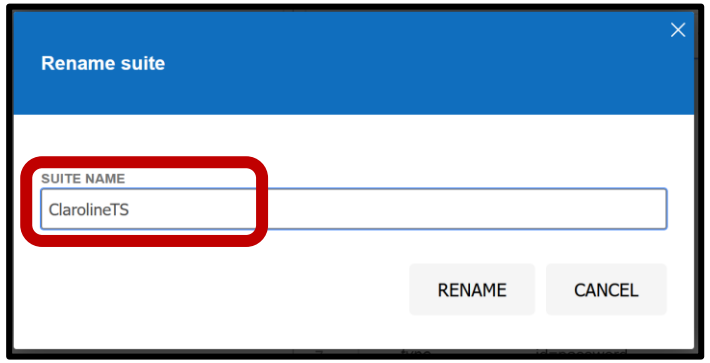
Export Test Suite (1)

1. Switch to Test Suite perspective

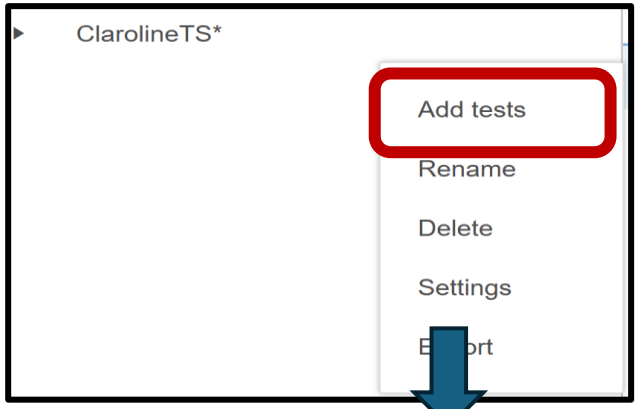


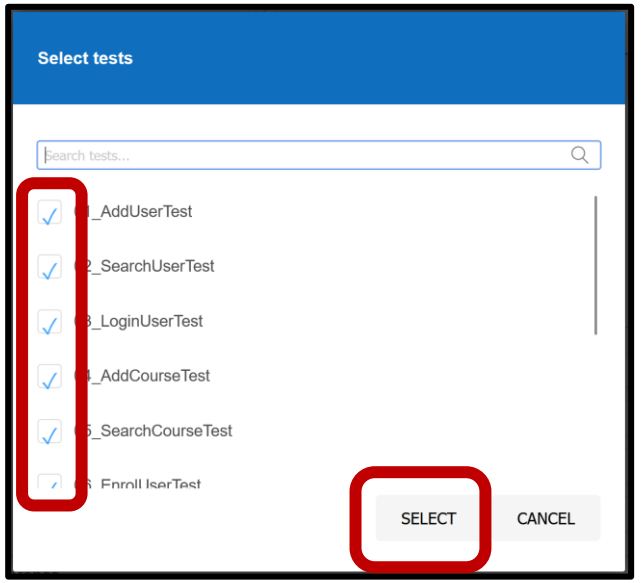
2. Rename Test Suite



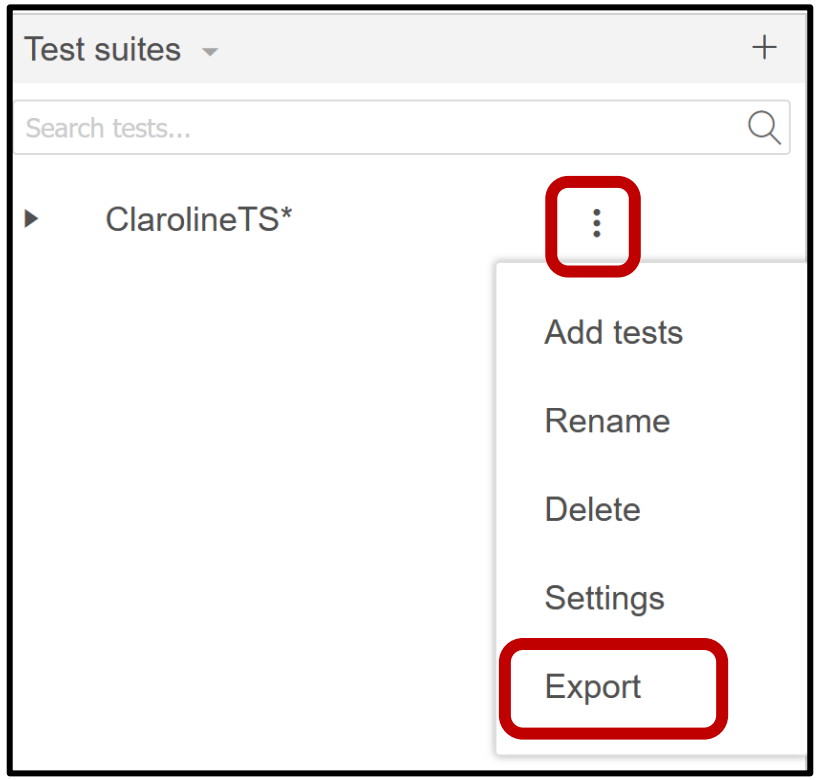


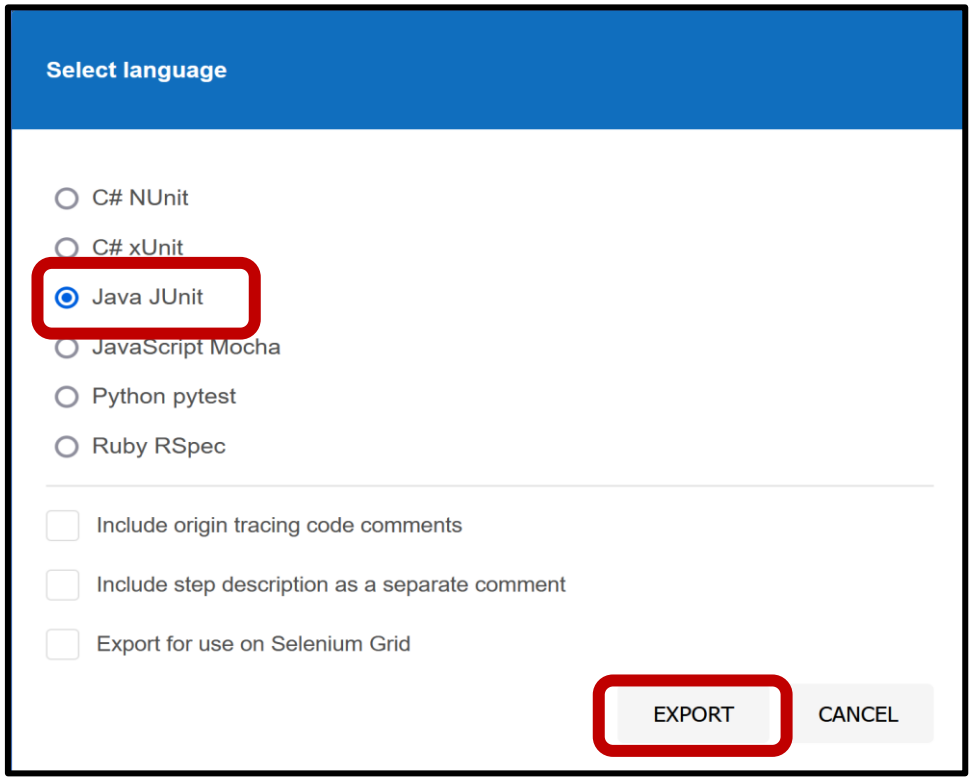
3. Add Test Cases to Test Suite

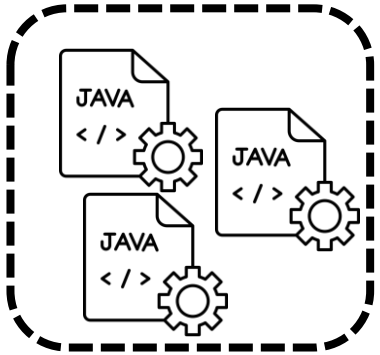




Export Test Suite (2)







TS-folder

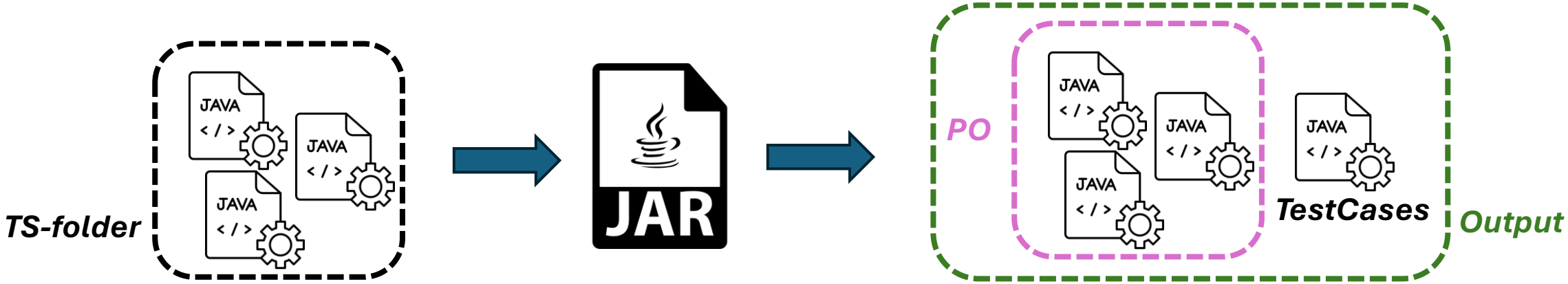
Save the files into a folder (e.g., ***TS-folder***)

Generate Files from Test Suite

- To generate executable test scripts + POs we will use the Assessor+ JAR file produced steps earlier

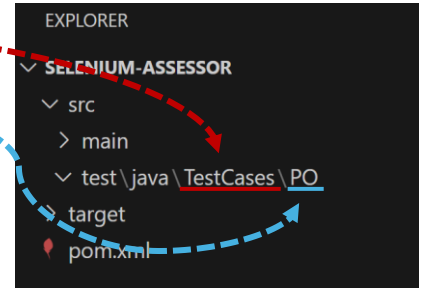
- Retrieve the JAR, then from cmd, execute: **java -jar assessor-1.0.0-jar-with-dependencies.jar path/to/TS-folder**

- Inside the target folder (***TS-folder*** in the example), an **Output** folder will be created with the TestCases and the Page Objects classes according to the annotations entered via Assessor+



Import Files into Java Project

- For each Web app to test, import **TestCases** and **POs** into a new instance of VSCode *selenium-assessor* project
  - Place **TestCases** class under **test\java\TestCases** package
  - Place **POs** classes under **test\java\TestCases\PO** package



- Since *selenium-assessor* project refers via Maven to the most recent version of Selenium, which integrates the **Driver Manager** module, you will not have to set the path to the driver manually, thus you can remove the line in red under the **TestCases** class produced by Assessor

```
@BeforeClass()
    public static void setup() {
                        System.setProperty("webdriver.gecko.driver", "InsertGeckoPathHere");
            }
```

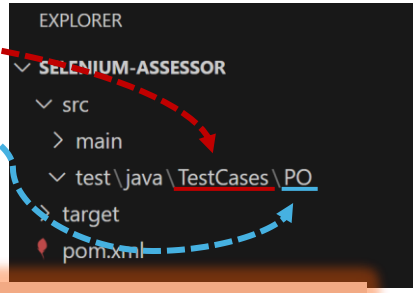
- One last line to change is the following under **MyUtils** class

```
WebDriverWait waitFor = new WebDriverWait(driver, 10);
```

```
WebDriverWait waitFor = new WebDriverWait(driver, Duration.ofSeconds(10));
```

Import Files into Java Project

- For each Web app to test, import **TestCases** and **POs** into a new instance of VSCode *selenium-assessor* project
  - Place **TestCases** class under **test\java\TestCases** package
  - Place **POs** classes under **test\java\TestCases\PO** package



- Since *selenium-assessor* proj Selenium, which integrates the **Driver Manager** module, you thus you can remove the line in red under the **TestCases** clas

You can now run the automatically generated test cases & Page Objects. Keep in mind that some fixtures might still be needed!

```
@BeforeClass()
    public static void set
                    Syste                    );
            }
```

- One last line to change is the following under **MyUtils** class

```
WebDriverWait waitFor = new WebDriverWait(driver, 10);
```

```
WebDriverWait waitFor = new WebDriverWait(driver, Duration.ofSeconds(10));
```

1. **Always read the Gherkin(s) before recording**
2. **Do manual fixtures when needed**
   - E.g., Check that the base URL to open the app is not empty
   - E.g., Remove some unwanted mouse over/clicks
   - E.g., Add some specific asserts or missed commands
3. **Do not record the browser closing**
4. **Keep assertions simple**
   - E.g., if it is stated in the Gherkin that an error message is displayed with a red background, or that such message is placed on the right in the GUI, just assert that the message exists
5. **Avoid using tabs to move between text boxes**
6. **Double check the recordings, test them, and clean up app data**

Even with these precautions, some manual fixtures over the test cases and PageObjects generated after the exporting might still be needed

References

- Selenium Manager: https://www.selenium.dev/documentation/selenium_manager/

- Setup Selenium WebDriver in VSCode: https://funnelgarden.com/setup-selenium-with-java-on-visual-studio-code/