**Introduction to IDE:**

Selenium IDE (Integrated Development Environment) is primarily a record/run tool that a test case developer uses to develop Selenium Test cases. Selenium IDE is an easy to use tool from the Selenium Test Suite and can even be used by someone new to developing automated test cases for their web applications.

Selenium IDE requires no additional setup other than installing the extension on browser. One of the driving philosophies is to provide an easy to use tool that will give instant feedback.

Features of IDE:

**Resilient Tests** - Selenium IDE records multiple locators for each element it interacts with. If one locator fails during playback, the others will be tried until one is successful.

**Test Case Reuse -**Through the use of the run command, you can re-use one test case inside of another (e.g., allowing you to re-use your login logic in multiple places throughout a suite).

**Control Flow -** Selenium IDE ships with an extensive control flow structure, with available commands like if, while and times.

**Pros of Using Selenium IDE:**

1. Provides you the capability of automatically recording your test cases based upon the interactions with the browser
2. Gives developers greater flexibility in executing the test cases. Either the test developer can run the entire test suite consisting of multiple test cases or execute a single test case
3. Operates based on the rich set of Selenese commands, which helps the IDE understand what needs to be done
4. Allows the test developers to set breakpoints for the purpose of debugging test cases
5. Test cases can be re-used using the run command. (e.g. allowing you to re-use the logic of login or reload on multiple places in the entire suite)
6. Use of multiple locators for each element in the IDE ensures successful execution

**Cons of Selenium IDE:**

1. **Overhead and upkeep: - Can’t handle dynamic web application**

As your web application changes and evolves, some of the paths Selenium uses to locate objects may change. For example, if the “name” or “id” attributes of an element change, the test may no longer be able to locate it and require an update. If new elements are added to a given page, it will also change the “xpath” to the original elements potentially rendering that element unreachable by Selenium. This is true for most web testing automation software (including Selenium WebDriver) but is worth stating. The larger and more robust the test suite, the more upkeep it will require. This is one reason that SIDE is ideal for short test cases like smoke testing, but less so for something like full regression testing.

1. **No data driven scripts / Database Connection:**

At the time that this article was published, SIDE did not support the ability to import data from an external file like a .csv file, and then execute a test for each record within it. According to some contributors to the project on Github.com, this is both in development for native support, and there are plug-ins in development as well.

1. **No built-in reporting capabilities:**

As of now, there are no reporting capabilities built into SIDE, but there are several plug-ins that address this.

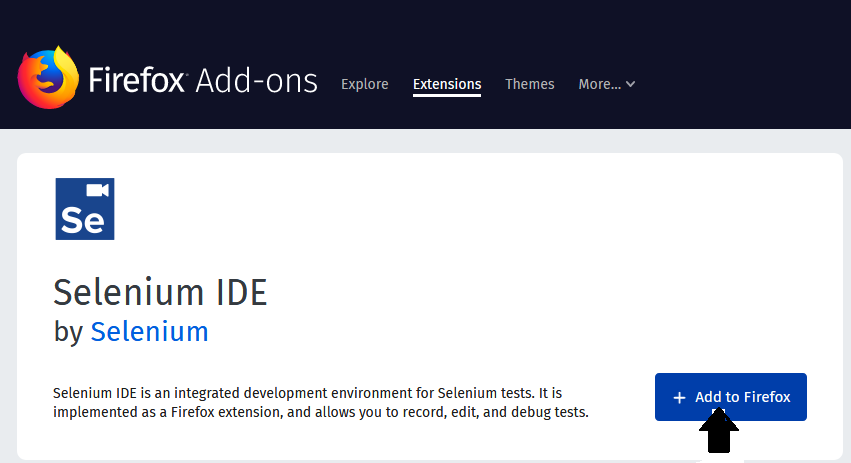
1. **No technical support:**

One of the downsides of the product being free is that there is nobody to contact when things go wrong. However, as stated in the “Pro” column, there is a large community of users and many of the issues you are likely to encounter have already been addressed on blogs and sites like stackoverflow.com

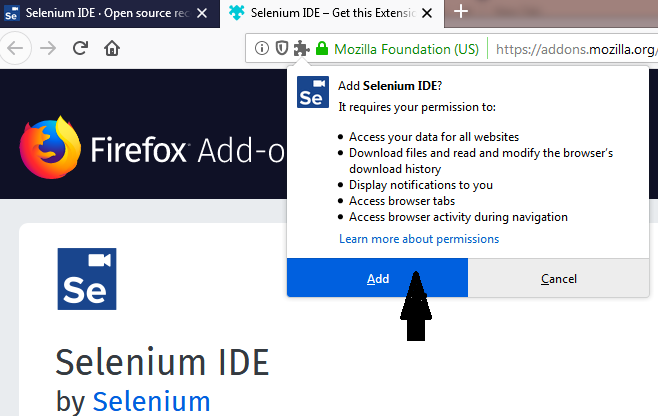
**Download Selenium –IDE [Firefox]:**

* Launch the Firefox browser.
* Then go to Google search for downloading the selenium IDE.
* Click on <https://addons.mozilla.org/en-GB/firefox/addon/selenium-ide/> link, and it will navigate you to the Firefox add-ons.

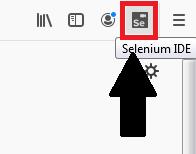
                    After that, clicks on add**to the Firefox**button and then click on install.

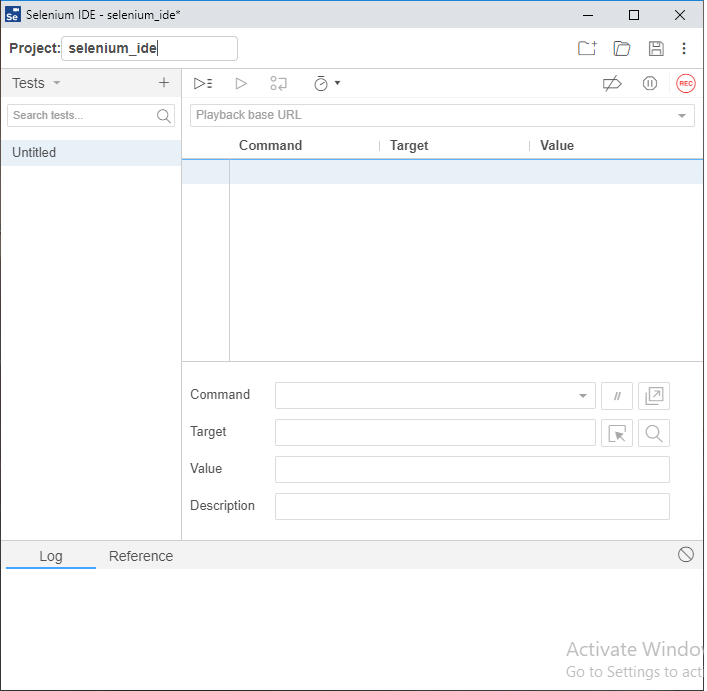


* A dialog box will be shown and asking to add Selenium IDE as an extension to your Firefox browser.



* Click on the **Add** button.
* Restart the browser when the installation is done.
* To open Selenium–IDE, go for the Firefox browser.
* Click on the Selenium IDE icon.

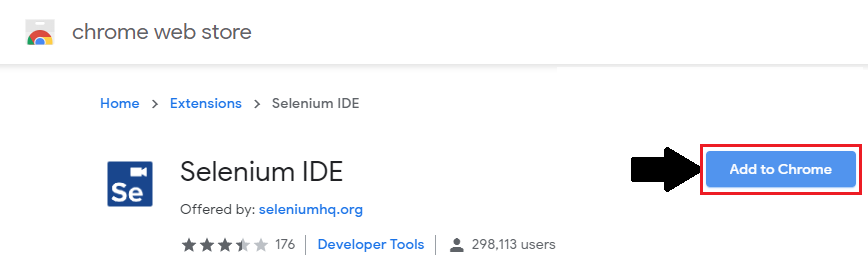


* Selenium–IDE browser is launched, and now, click on create a new project.
* After creating the project, it will show the Selenium–IDE interface.

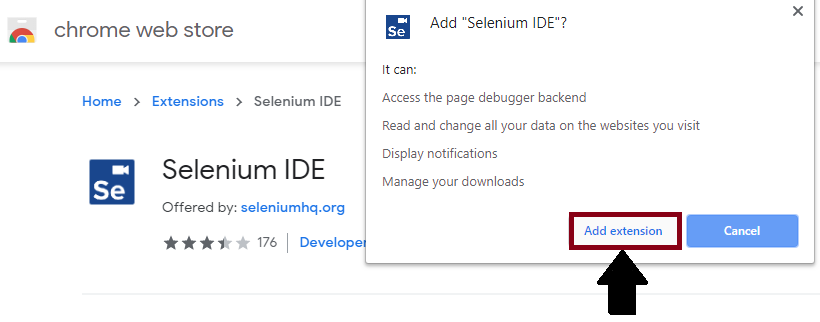
**Download Selenium–IDE [Google Chrome]:**

* Launch the Chrome browser.
* Then go to Google search for downloading the selenium IDE.
* Click on <https://chrome.google.com/webstore/detail/selenium-ide/mooikfkahbdckldjjndioackbalphokd> link, and it will navigate you to the chrome web store.

                    After that**, click** on **Add to Chrome**button and then click on install.



* A dialog box will be shown and asking to add Selenium IDE as an extension to your chrome browser.



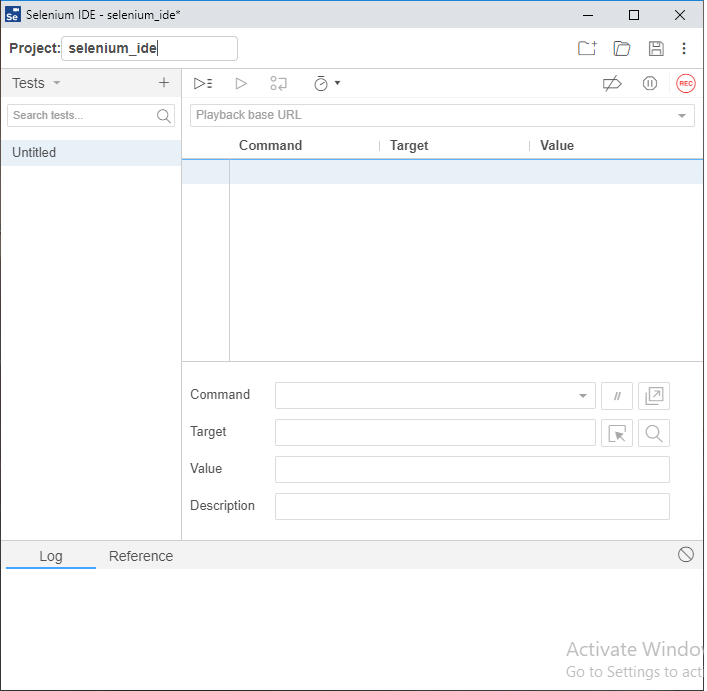
* Click on **Add Extension**button.
* Restart the browser when the installation is done.
* To open Selenium –IDE, go for chrome browser.
* Click on the Selenium IDE icon.



* Selenium–IDE browser is launched and now click on create a new project.



* After creating the project, it will show the Selenium–IDE interface.



# First test case of Selenium-IDE

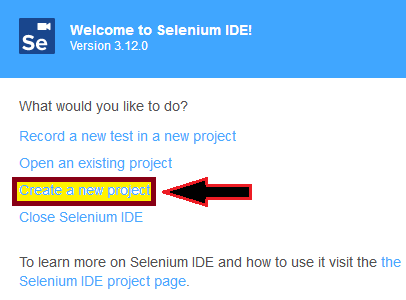
The entire test script execution process in Selenium -IDE can be divided into the following steps:

* Recording (record the user interactions with the browser)
* Playing back (executing the recorded script)
* Save the test case suite
* Export the test script

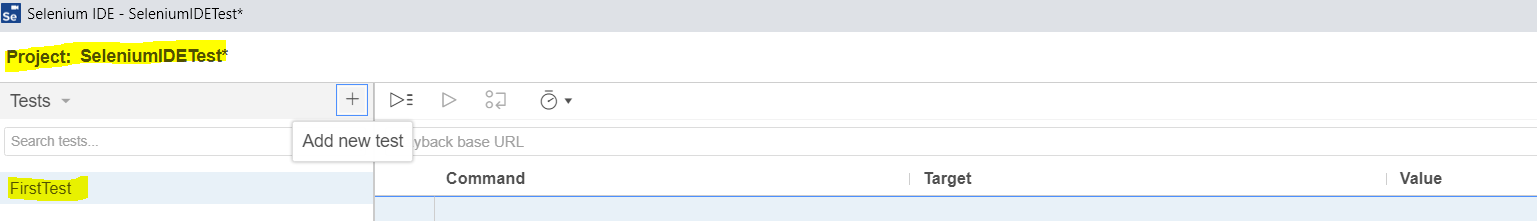
Now, we will look into the implementation of the above three steps.

**Recording:**

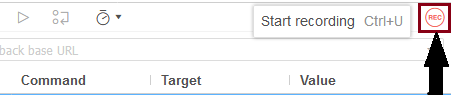
* Launch the Firefox browser.
* Click on the Selenium icon, which was present on the top right corner of the Firefox browser.
* The browser will launch the default interface of Selenium-IDE, and asking you to create a project.



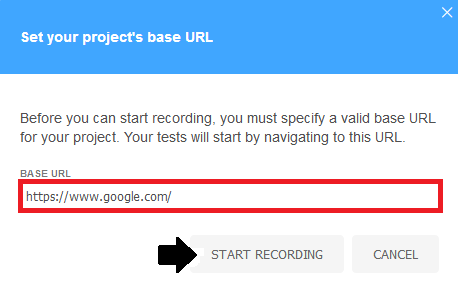
* Give a project name, e.g., **SeleniumIDETest**
* Give a test name, e.g.-**FirstTest.**



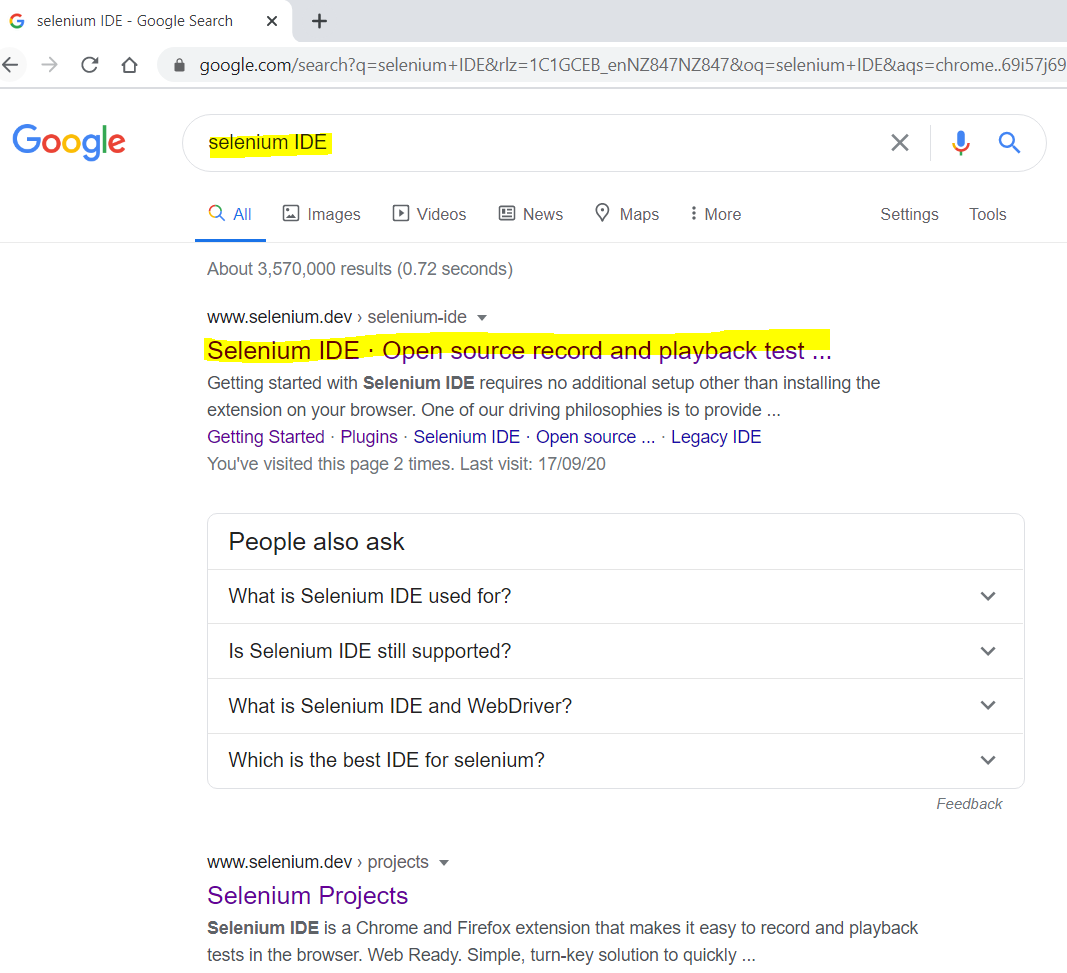
* Click on the **Start Recording Button**, which is present at the top right corner on the Selenium-IDE interface.



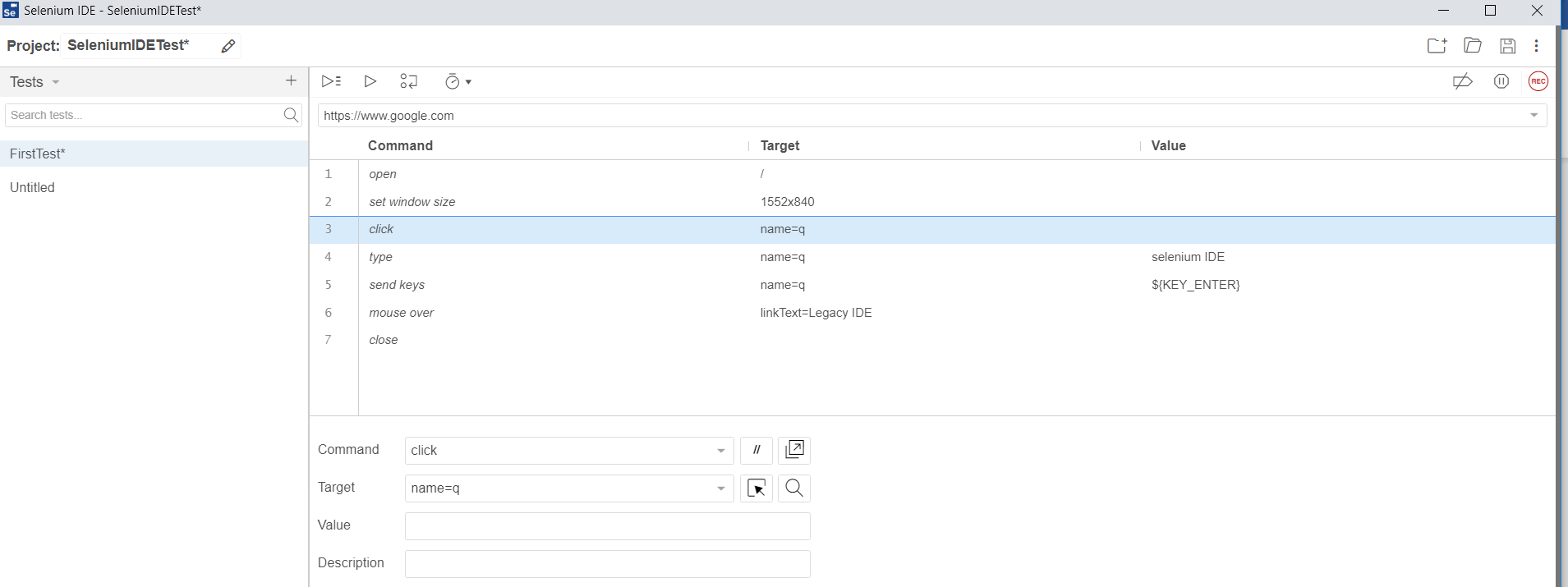
* After click on the start recording button, one pop-up window will be shown where you have to give a base URL. e.g., <https://www.google.com/> and again click on the start recording button.



* It will redirect you to the Google search page, where you can type **Selenium IDE** and click on the first link.



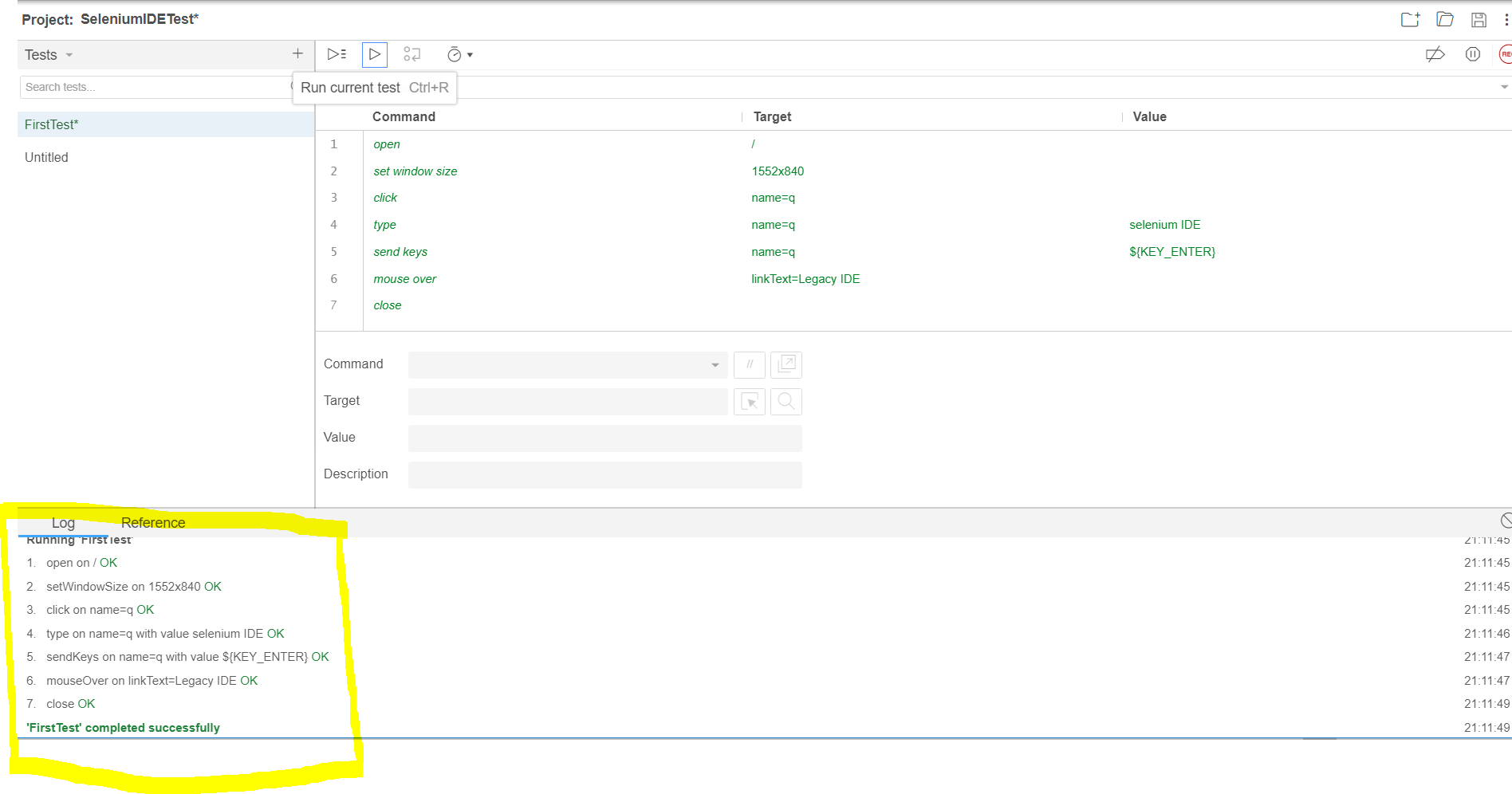
* It will redirect to the home page of **seleniumdev .com**.
* Go to selenium –IDE and click on the **stop recording button**, it will stop your action with browser immediately.
* And the text editor box has recorded all the action performs on the browser.



Now, we are going to the next step, which includes executing the recorded script.

**Playing Back:**

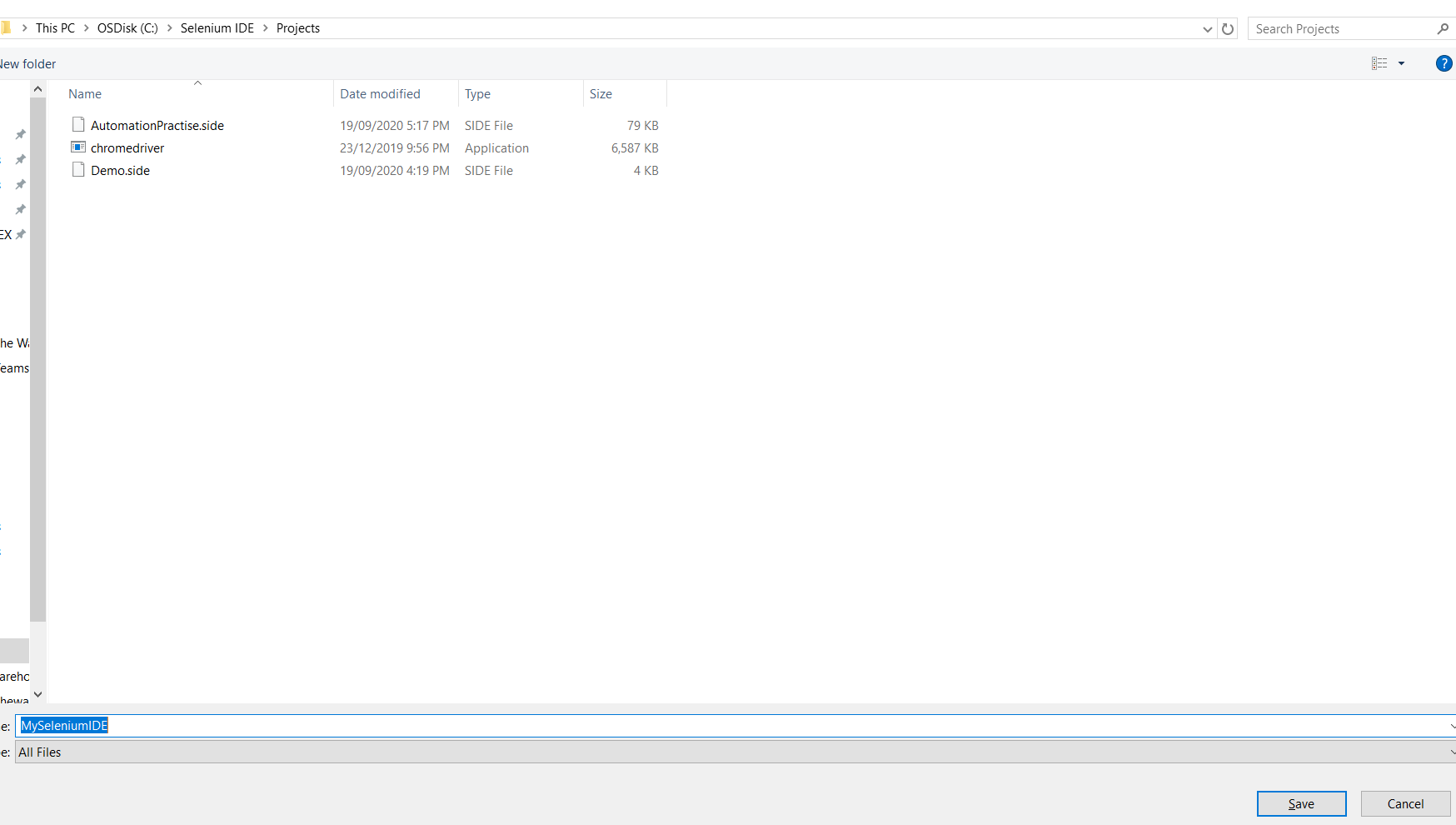
* Click on **Run Current test** button, which is present on the toolbar menu of the Selenium-IDE.
* It will execute all the interactions with the browser and gives you an overall summary of the completed test scripts.
* Executed test scripts summary will be displayed on the log pane.



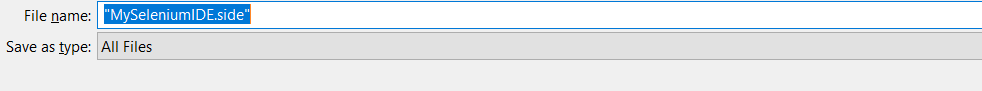
Now, we are coming to our next segment, where we can learn about how we can save the project.

**Save Project:**

* Click on the **save project**button present on the extreme right corner of the menu bar.
* Save the test case suite as **MySeleniumIDE**.



* The test case suite can be found at the location provided in the above steps. save as **.side** format.

test case suite can be found at the location

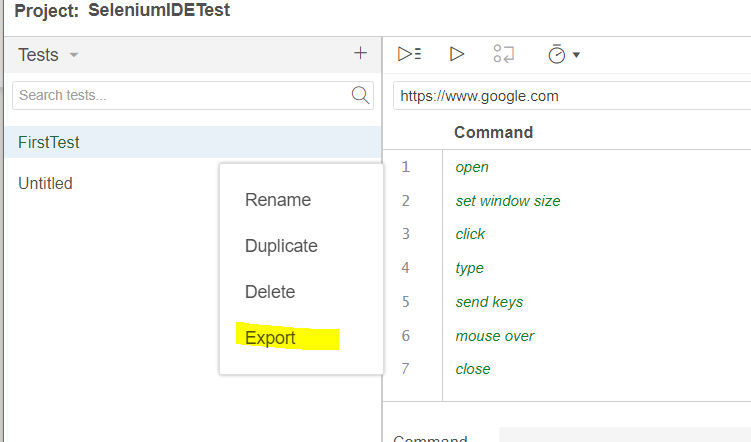
**Export Test Script**

There is no export option in the latest version of Selenium IDE because there is no file menu present in the IDE interface.

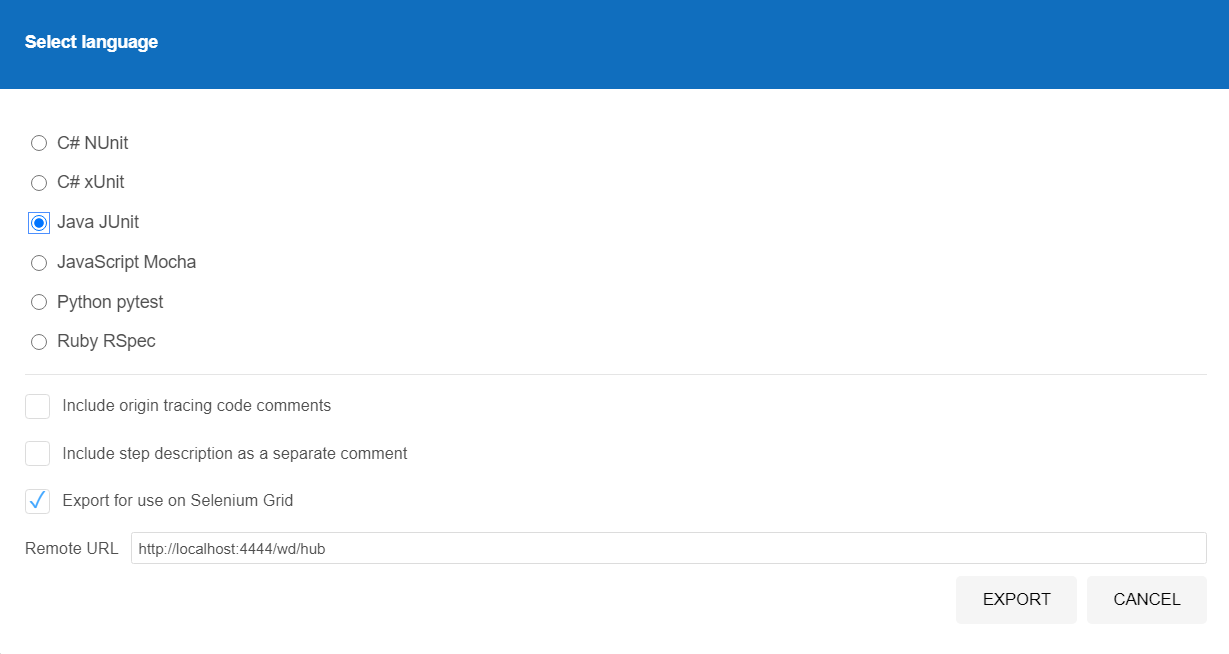
To export our test case in different languages in selenium ide, follow below steps:

For our better understating that how we export our test script in a different language, we are considering the above example

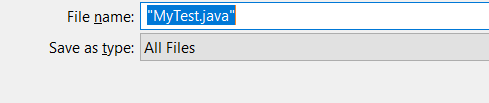
* First right click on the existing test case **FirstTest and click on the export button.**



* After click on the export button, one pop-up window will be opened and ask you to select one of the programming languages.
* Select one of the options and click on the**export** button.



* Export your test case file in your local system and save it with a particular name.



* After saving the exported file in our local system, we will open that file. You can open that exported file, go to your Selenium IDE interface and click on the particular file.
* Your exported test looks like this

// Generated by Selenium IDE

import org.junit.Test;

import org.junit.Before;

import org.junit.After;

import static org.junit.Assert.\*;

import static org.hamcrest.CoreMatchers.is;

import static org.hamcrest.core.IsNot.not;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.remote.RemoteWebDriver;

import org.openqa.selenium.remote.DesiredCapabilities;

import org.openqa.selenium.Dimension;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.interactions.Actions;

import org.openqa.selenium.support.ui.ExpectedConditions;

import org.openqa.selenium.support.ui.WebDriverWait;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.Alert;

import org.openqa.selenium.Keys;

import java.util.\*;

import java.net.MalformedURLException;

import java.net.URL;

public class FirstTestTest {

private WebDriver driver;

private Map<String, Object> vars;

JavascriptExecutor js;

@Before

public void setUp() throws MalformedURLException {

driver = new RemoteWebDriver(new URL("http://localhost:4444/wd/hub"), DesiredCapabilities.chrome());

js = (JavascriptExecutor) driver;

vars = new HashMap<String, Object>();

}

@After

public void tearDown() {

driver.quit();

}

@Test

public void firstTest() {

driver.get("https://www.google.com/");

driver.manage().window().setSize(new Dimension(1552, 840));

driver.findElement(By.name("q")).click();

driver.findElement(By.name("q")).sendKeys("selenium IDE");

driver.findElement(By.name("q")).sendKeys(Keys.ENTER);

{

WebElement element = driver.findElement(By.linkText("Legacy IDE"));

Actions builder = new Actions(driver);

builder.moveToElement(element).perform();

}

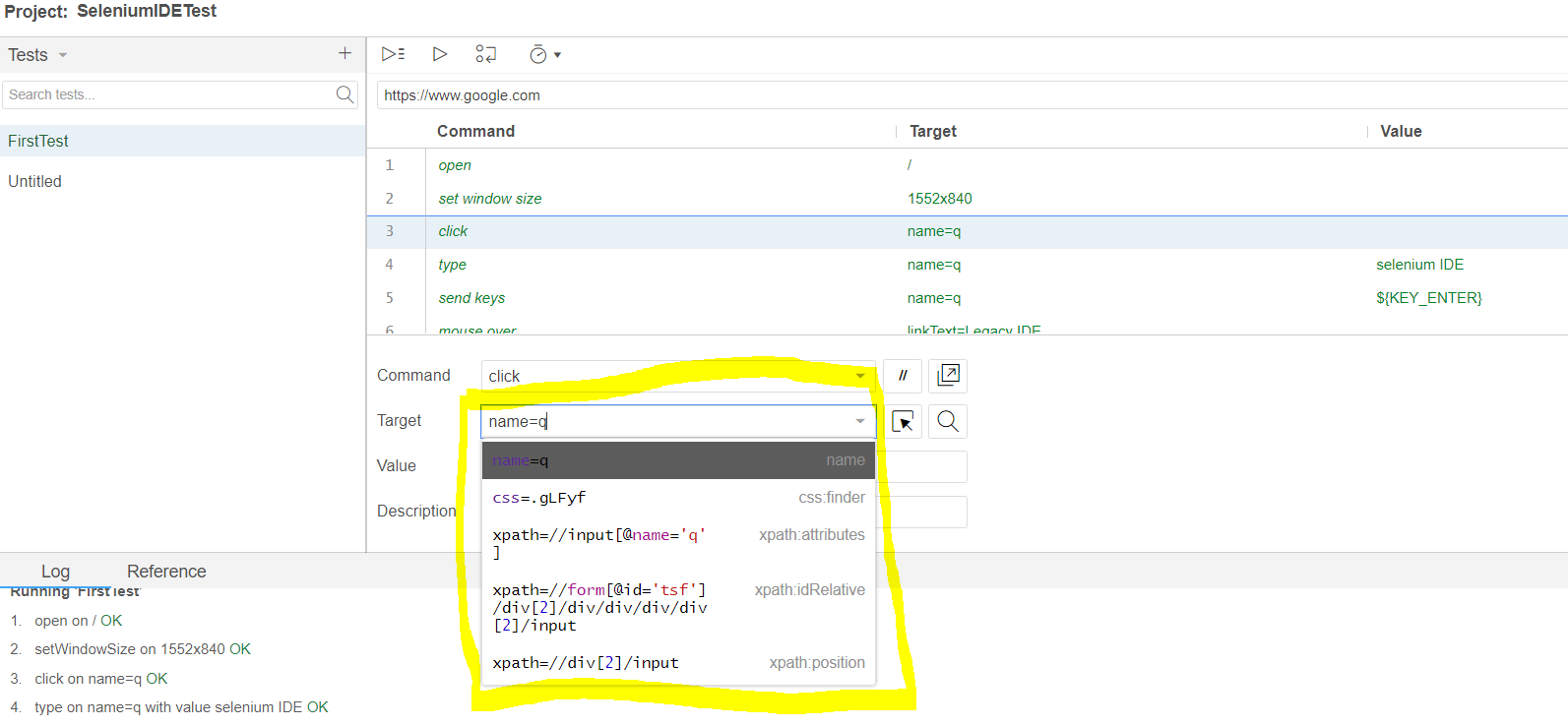
driver.close();

}

}

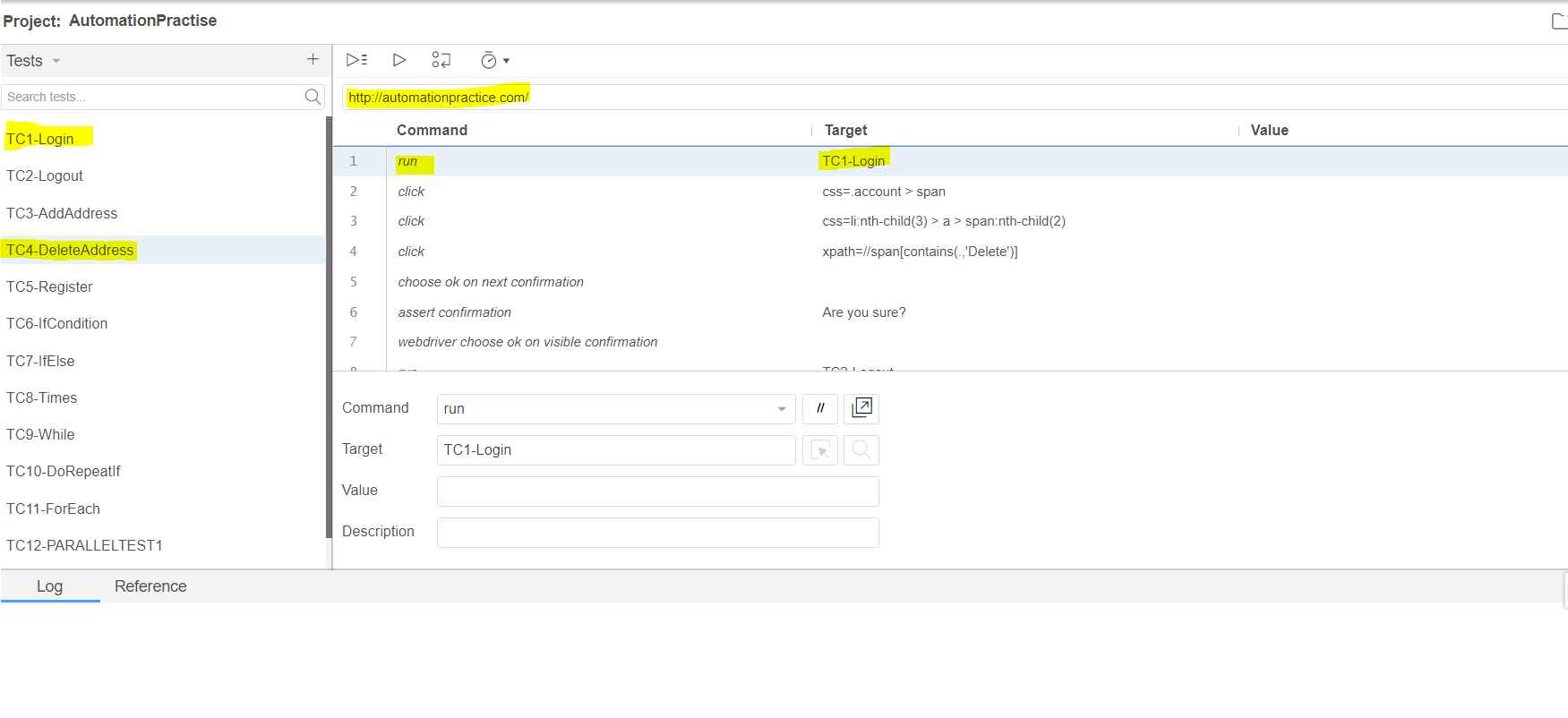
# Characteristic of Selenium-IDE

**No Brittle Test:** New IDE captures more locators for a recorded element.

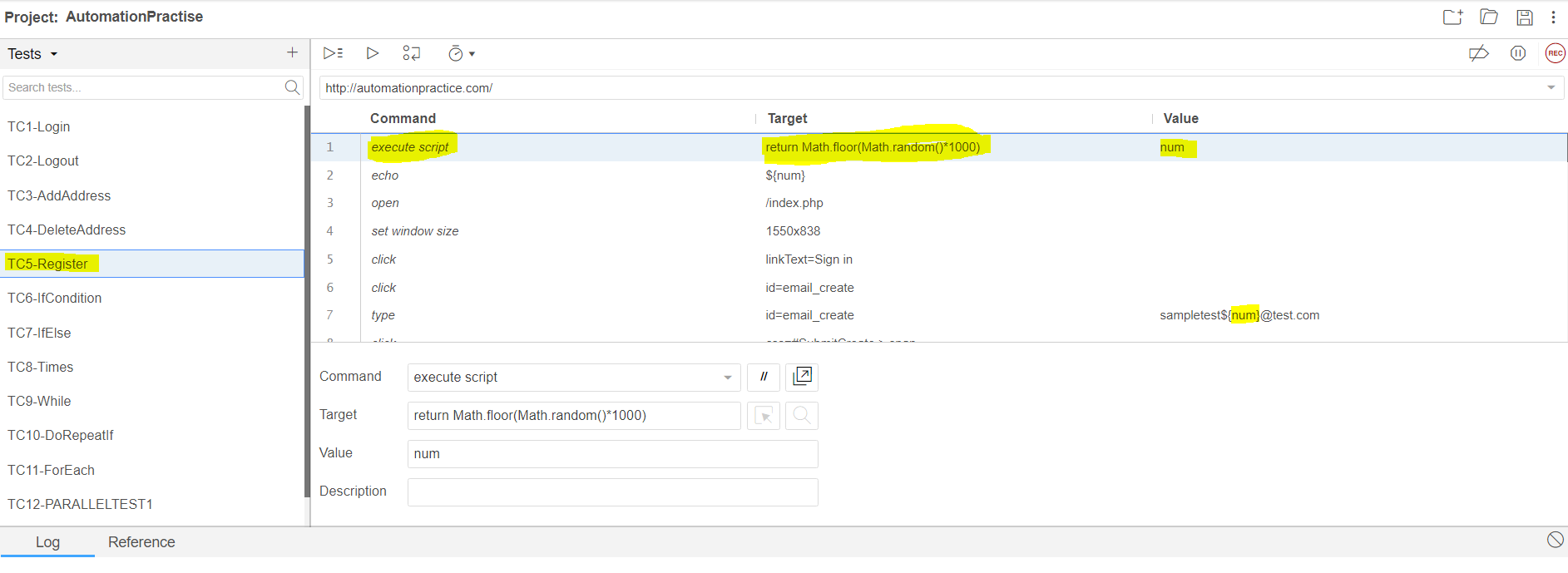
The below highlighted element has name,xpath,css to identify the same elements

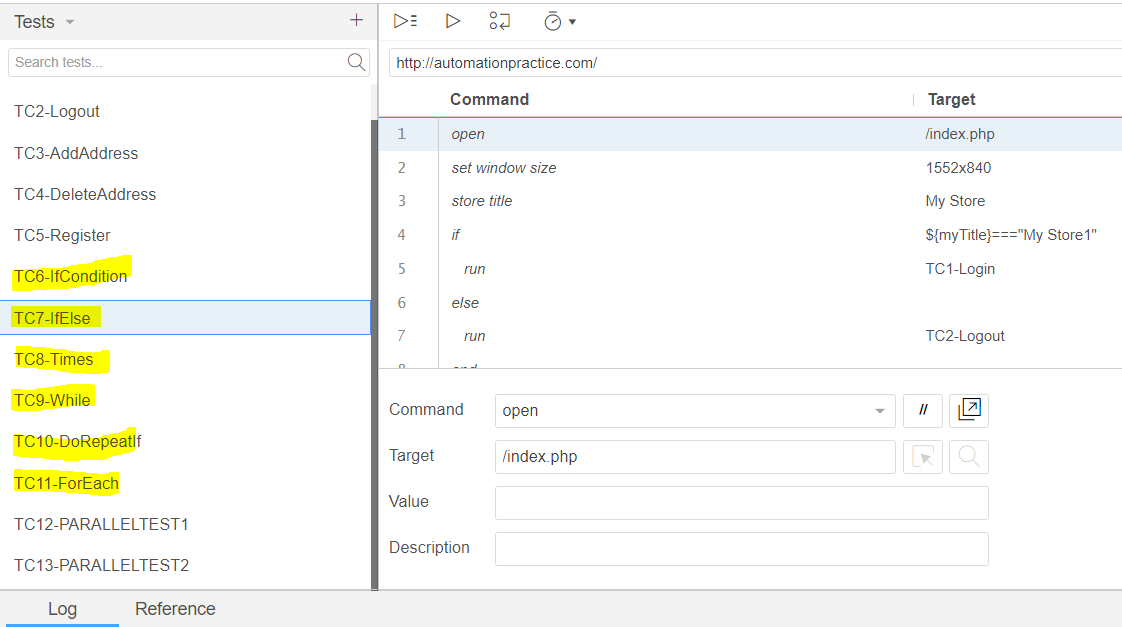
**Reusable / Modular Test:** Possible to have a test inside another test

Below snapshot shows, TC4-DeleteAddress has used TC1-Login in it



**Supports for Embedded Code:** Allows javascript code inside test to make the test more dynamic



**Conditional Logics:** supports if, else if, while, times, do repeat if, for each loops 

**Parallel Execution:** This can be done only through runner via CLI (Command Line Interface). In order to achieve parallel execution, follow the below steps to install runner.

**Pre-requisite:** Make sure your system has node.js installed, else refer the website <https://phoenixnap.com/kb/install-node-js-npm-on-windows>

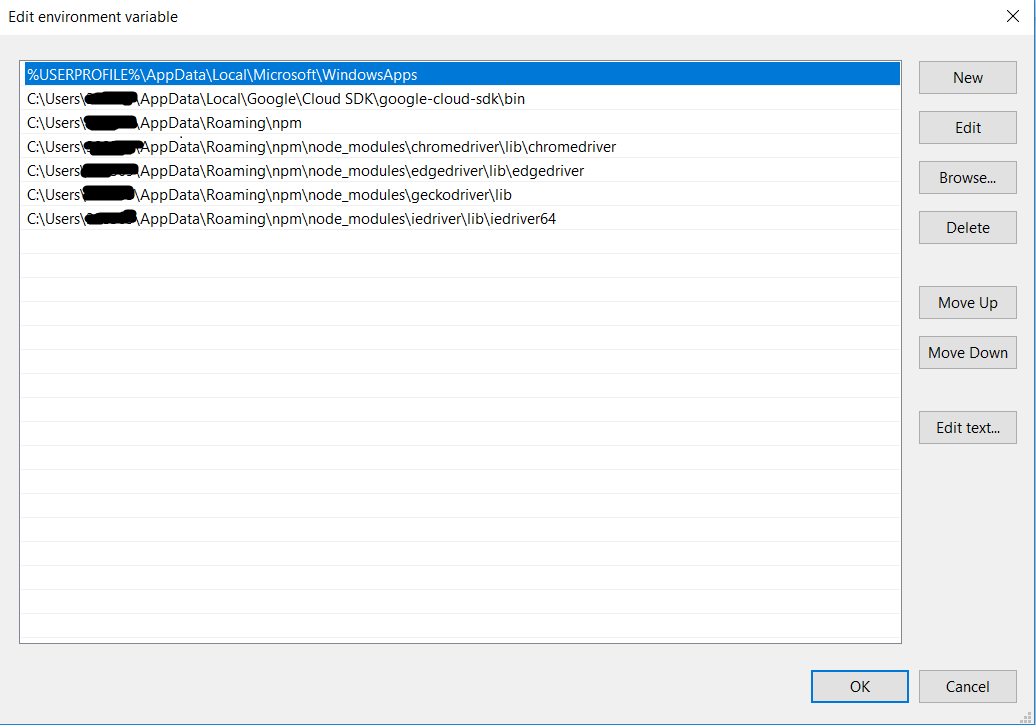
**Test Suite Side:** Enable the Parallel Running Check box from Selenium IDE against the Test suite

Step 1: Go to “cmd” prompt

Step 2: install selenium runner using “npm install -g selenium-side-runner”

Step 3: install browser drivers using “npm install -g chromedriver”

Step 4: Repeat step 3 for all other browser, replace the chrome by your desired browser driver

Step 5: Add driver path and npm in the System Environment Variabless

Step 6: Go to the project path where your “.side” file resides and open command prompt from there

Step 7: Type “selenium-side-runner <proejct name>”, example “**selenium-side-runner AutomationPractise.side**”

