Selenium 4 Beginners

Rex Jones II

(Part 2) Building Blocks For Selenium

Table of Contents

Recap (Part 1) Building Blocks For Selenium

Before discussing TestNG, give me a few moments to recap (Part 1) Building Blocks for Selenium.

According to Selenium's HQ Browser Automation website, there are 7 preferred languages for Selenium: In alphabetical order, the 7 languages are: C Sharp - C#, Java, JavaScript, Perl, PHP, Python, and Ruby.

Java is the most popular programming language according to a tracking index by TIOBE. One of the reasons why I chose Java because it is the most popular language.

TestNG which stands for Test Next Generation is a Test Framework for Java. It was designed for automation testers. The other Java Test Framework is JUnit which inspired TestNG. TestNG can perform Unit Testing like other frameworks but added more features for automation engineers. This is a classic example of popularity does not equate to superiority. I believe JUnit is the most popular framework but TestNG is the most powerful framework. TestNG is superior.

Selenium has 4 components: Selenium IDE, Selenium Remote Control also known as Selenium RC, Selenium WebDriver, and Selenium Grid. Our tutorial sessions will cover Selenium WebDriver.

Okay, we left off at this point with (Part 1). How many of you watched (Part 1) Building Blocks For Selenium at least 1 more time? Did you execute the same Test Script Search_T_SHIRTS? Of course, it's all optional but the main practice was to create a Test Script to save your registration for Demo Web Shop. I added a sample of my automation code in the zip file that you had access to download.

Here's my automation code from Part 1's practice. Did you write all your steps within your main method like me? If so, it's okay. That's a common practice between trainers and people watching the trainer's session. I've attended several training sessions and attendees watching the sessions copy the trainer even when there's bad practice. I understand, the trainer may know the best practice. My only concern is the attendee may not understand the best practice and take that philosophy to a project. In defense of the trainer, I think most trainers want to cover many concepts so they have limited time.

Remember in (Part 1) Building Blocks For Selenium, I mentioned a method should perform only 1 task. It's not wrong but it's not a good programming practice to add multiple tasks into 1 method. One of the benefits of using 1 task per method is the ability to use that method over and over. Repeatability - we can call that method again. Calling a method is when the steps of that 1 method are executed from other locations. I placed all those steps into the main method because we have not covered.

TestNG. In this session and the sessions going forward we will use 1 task per method. Eventually I am going to show you how to create and call some of those methods. For now, let's talk about TestNG annotations and assertions.

Annotations

The purpose of annotations is to bring programming and testing closer together. According to dictionary.com, an annotation is a critical or explanatory note. Therefore, a TestNG annotation is a critical or explanatory note for a Java method. There are many TestNG annotations but every time a Java method is annotated by: @BeforeTest, @Test, and @AfterTest. The method runs at a certain time.

@BeforeTest runs its Java method 1 time before the first test method. If there's only 1 test method then the Java method will run once before the test method.

@Test sets its Java method as a test method. We have the ability to set 1 Java method as a test method or several Java methods as a test method.

@AfterTest runs its Java method 1 time after the last test method. If there's only 1 test method then the Java method will run once after the test method.

Demo

Classes / Methods

Now, let's get into our demo of TestNG. We are going to revamp our Test Script Search_T_SHIRTS to include TestNG annotations by creating a new class called Search_T_SHIRTS_TestNG.

After creating a class, the next step is to create a few methods to perform each task. The methods will be called Set Up, Automation Sign In, Search T-Shirts, Sign Out, and Tear Down.

public is an access modifier which means this method can be accessed by outside classes. void means the method does not return a value. adminSignIn is the method's name. All method declarations end with a parenthesis. An empty parenthesis means the method does not receive a value.

Within Package Explorer, the methods are displayed in alphabetical order. Notice how each method has a green circle in front of its name. The green circle indicates it is a public method. However, a private method will show a red square. private means this method cannot be accessed by other classes.

Let's bring the programming part closer to testing by writing an annotation for each Java method.

Add Annotations

Before we test, we must set up our test. Therefore, I will add an annotation called @BeforeTest to the setup Java method. This method runs before all test methods. Next, the @Test annotation maps the automationSignIn, search_T_Shirts, and signOut methods. These test methods run after our test is set up. Finally, is the @AfterTest annotation which runs after all 3 test methods and will be tied to the tearDown method. The errors for each annotation are in this program because they have not been imported. Let us import the TestNG annotations.

Import @BeforeTest by selecting org.testng.annotations.

Import @Test but there is an option for org.testng.annotations and 2 JUnit Libraries. The options to add JUnit 4 and JUnit 5 Library to build path show up because JUnit also has an annotation called @Test. We select TestNG.

Import @AfterTest by selecting org.testng.annotations. The imports are shown at lines 3 - 5.

Now, let's copy and paste the automation code from Search_T_SHIRTS to Search_T_SHIRTS_TestNG. Before we test, we set up our test by determining the browser and loading the web page. In this case, we are going to load Chrome and go to the Automation Practice application. Copy lines 11 – 15.

Minimize the setUp method. Hover over the plus sign if you want to view statements within the method.

Next, we sign into the application by clicking the Sign In hyperlink, entering an Email Address, Password, then clicking the Sign In button. Therefore, we copy lines 17 – 23 to the automationSignIn test method. Press the Shift and Tab key to move the statement to the left.

After signing into the application, we search for some Orange T-Shirts by clicking the T-SHIRTS hyperlink, entering Orange in the search box, then clicking the Search button. We will copy lines 25 – 29 to the search_T_Shirts test method.

The signOut method is the last test method and we copy line 31. Click the Sign out hyperlink.

After we test, we will tear down our test by closing the web browser. Line 33.

Notice how the last 4 methods have an error. However, there is not an error for the first method. There is a red line underneath driver in each method. You can ignore the red lines within the comment statement. We are going to focus on driver which has a problem.

The setUp method does not have an error because we instantiated driver in this method by writing WebDriver driver. Instantiate means we provided an instance of driver. Currently, driver is a local variable and only visible within the setUp method. We need to make driver a class variable where it's visible to all methods in this class. You may hear some people refer to a class variable as a global variable.

In reality, Java does not have a term called global variable. We make driver, a class variable by placing it within the class and outside of all methods. Now the errors are gone.

Let's run. All 3 test methods PASSED: automationSignIn, search_T_SHIRTS, and signOut. The Results tab also shows they Passed. A question people ask is "Where are the methods setup and tearDown that is associated to BeforeTest and AfterTest? We don't see them. Why are they not shown in the Console or Results tab?". We don't see those methods because they are not test methods. Annotations that start with @Before or @After are Configuration annotations.

You can bypass writing System.setProperty if you add your Drivers to Environment Variables. I created a document and a video called How To ByPass System.setProperty if you do not want to write this statement. I am going to comment this line using the single line comment syntax.

Comments

Let's discuss comments. There is a healthy debate whether should we use comments? Some people say we should use comments and some people say we should not use comments. For those who say, we should use comments. When is the appropriate time to write a comment? I like Paul Grizzaffi and his former professor's view on comments. We should write comments when describing why our code is doing something rather than what our code is doing.

With experience, we can read code and know what the code is doing. For example, a couple of the comments are "Enter Password and Click Sign in button". You can read my code and determine I am entering a Password and clicking the Sign in button. Both comments describe what the code is doing. I would not keep these comments if this was a project assignment.

On the other hand, entering the steps and making them comments can serve as a guideline. Therefore, feel free to copy and paste the steps then decide if you want to delete or keep your comments.

Instantiate WebElement

Do you see the concepts of automation? First, we find the web element then we perform an action on the web element. We find the Sign out hyperlink then we click the Sign out hyperlink. Have you noticed that I have not used the WebElement class like it was demoed in the Data Types class or shown in our presentation slide?

Let's go back to our presentation slide, we see WebElement as the class and element as the object followed by driver.findElement(). So far, we started with driver.findElement(). WebElement only needs to be declared with an object if we plan to perform more than one action on an element.

More than one action such as checking if a hyperlink is displayed then clicking the link. If we perform 1 action on an element, then it is optional to declare the WebElement class.

Here's how you would write the same statement using WebElement. Driver Dot Find Element By Class Name Logout assign the value to WebElement hyperlinkSignOut =. If we have a word that we do not want to type, we can bring up intellisense. Type some of the word hyp, press CTRL + SPACE then we see hyperlinkSignOut Dot Click

Find then perform. We can check to see if the Sign out hyperlink is displayed by writing hyper CTRL + SPACE Dot Is Displayed. Here is more than one action on the same element. Checking to see if the link is displayed and clicking the link.

Let me show you how to change multiple words at one time. For instance, if we think hyperlinkSignOut is too long. We can select the object then press ALT + SHIFT + R then rename to linkSignOut. Another way is go to Refactor then select Rename.

Run again. If you get an error, it's probably because you commented out the System.setProperty line without bypassing that statement. Let me ask you a question. Up to this point, what have we tested? So far, we have automated signing into an application, searching for Orange T-Shirts, signing out of the application, and closing the browser but have not performed our test yet. Look what happens when I comment all code from the search_T_Shirts method using a multi-line comment syntax. Execution will sign in then sign out. It will not search for Orange T-Shirts. Run

The search_T_Shirts method passed although our Test Script did not search for a T-Shirt.

Why did it pass without any action like automationSign and signOut? It passed because the syntax for that method was correct. A method does not require a body which is where the action takes place.

Well then, how do we verify if our test actually passed or failed?

Assertions

Assertions - Assertions help us to verify if our test passed or failed. There are several Assert methods but we will cover assertEquals and assertTrue. assertEquals verifies the actual and expected values are equal. assertTrue verifies a condition is true.

Now, let's demo assertions starting with our new Test Case: Verify Correct User And T-SHIRT Color. Step 6 and Step 10 are the new steps that will use an assertion to verify the correct User and T-Shirt color.

We are going to verify the application displays correct user as Selenium Automation and verify the application searches for the correct color ORANGE.

Let's verify we have the correct user after we sign into our application. I am going to use linkText as the locator.

First, we are going to find the element Driver dot Find Element By Link Text Selenium Automation. We have found the element, now let's get the element from our test application. (Dot Get Text) Get Text is a method that gets the information. Do you see the intellisense shows String – WebElement? This means the getText method returns a web element and the web element will be a String. Therefore, after we find the text and get the text, we will assign it to a String. We will name the String actualUser.

Next, we compare the text, so let's write String expectedUser = "selenium automation". All lowercase and a space between both words. Finally, our test is to verify the Actual User and Expected User are the same. The Assert class has a lot of methods. However, we will use Assert True as the method. Make sure Assert has an uppercase A. (Assert Dot assert True (Actual User Dot Equals, Expected User)).

Line 43 gets the text from our application and assigns the value to actualUser. Line 45 assigns selenium automation to expectedUser because that's the value we expect from the application. Line 47 verifies if the actualUser and expectedUser are equal to each other.

Let's verify the application searches for Orange after we search for T-Shirts. Copy and paste the Test Step after clicking Search button. Driver dot Find Element By XPath. We are going to get the text and assign the value to actualColor.

Go to Chrome and get the xpath value. Inspect the color, copy xpath, and paste the value. The expectedColor is Orange.

Assert Dot Assert Equals (Actual Color Comma Expected Color). We can also add a message. The message will only show up if there is a fail. Let's add The Actual and Expected Colors Do Not Match.

I am going to comment driver.quit() so it don't close the browser. Before I run, I want you to know this run will fail. Do you have an idea why it will fail? Let's look at the code one more time. Run

There are 2 Failures. The first failure is automationSignIn. The Assertion Error states "". Let's look at the difference. The application has uppercase S and A for Selenium Automation while our automation code has lowercase s and a for selenium automation. We can pass this method 1 of 2 ways. Make s and a capital for expectedUser or leave s and a lowercase then change equals to equalsIgnoreCase. ignores the case considerations. In other words, the s and a will get ignored.

Let's look at the second failure search_T_Shirts. The Assertion Error states "The Actual and Expected Colors Do Not Match expected [Orange] but found ["ORANGE"]". It found ORANGE with all capital letters and our code has O as the only capital letter.

The TestNG Results tab will show the same errors. automationSignIn and signOut

We will change our expectedColor to all capital letters. Let's Run but I want you to know again this run will fail again. Do you have an idea?

search_T_Shirts failed expected ORANGE without double quotes but found ORANGE with double quotes.

We must add double quotes. Remember in (Part 1) Building Blocks For Selenium, we saw how an error shows up for xpath values if there are double quotes without a back slash. The error states invalid AssignmentOperator. Well, the same error appears with ORANGE if we only add double quotes.

In this sense, the double quotes have a special meaning to the compiler and requires a backslash. Therefore, we add a backslash in front of the double quotes just like the xpath values.

Now, all methods should Pass. Run one more time.

They PASSED.

That's it for Building Blocks For Selenium: Java, Selenium WebDriver, and TestNG. I hope the demo was helpful.

Questions, Answers, and Practice

Questions

#1. Is it wrong to add multiple tasks in 1 method?

#2. Which annotation sets its Java method as a test method?

#3. What is the purpose of Assertions?

Answers

#1. **#1. Is it wrong to add multiple tasks in 1 method?** No, it is not wrong but it is a bad programming standard to add more than 1 task in a method.

#2. **Which annotation sets its Java method as a test method?** @Test sets its Java method as a test method

#3. **What is the purpose of Assertions?** Assertions verify if a test Pass or Fail.

Practice

If you have time, watch this video (Part 2) Building Blocks For Selenium at least one more time. I may ask you to watch the video again every time as practice. It's similar to a song we may hear something that we did not hear the first time.

Execute the same Test Script from video which verified the User and T-Shirt Color.

Create A Test Script To Verify Demo Web Shop Email And Logout.

We click the Register hyperlink, select Gender, enter Personal Details, enter Passwords, then click the Register button. This time verify if the Email is correct. After verifying if the Email which is the Account, click the Email, click the Save button, then click the Log out hyperlink. The last verification is to verify if the Log out hyperlink changes to Log in.

Let's take a look at your Practice Test Steps to Verify Demo Web Shop Email And Logout. Steps 10 and 13 are the verification steps: Verify the Email hyperlink "Account Information" and Verify User Logged Out via Log in hyperlink.

In this last presentation slide, I added some Java, Selenium WebDriver, and Test NG Books. The first book Java: A Beginners Guide is the book I read to learn Java. It is a well written book. In fact, this book inspired me to write about Java. After I learned Selenium, I realized the first book included programming concepts that are not necessary for a test automation engineer. We only need core Java to be effective at automating applications.

Therefore, I wrote the next 2 books which filters out the additional programming concepts and focus on core Java: Part 1 and Part 2 Java 4 Selenium WebDriver. These 2 books only cover Java. It does not cover Selenium.

The next section are Selenium WebDriver books. Most Selenium books supply a little knowledge about Java and basic automation concepts. I wrote the first Selenium book that only show and explain automation concepts. The second Selenium book Mastering Selenium WebDriver dives into advanced concepts.

The third section of books is TestNG. I wrote the first book "Getting Started With TestNG" which provides a foundation of TestNG. It explains Annotations, Assertions and different test types such as Data Driven Testing and Cross Browser Testing. Next Generation Java Testing by Cedric is the book I focused on to learn TestNG. If I'm not mistaken, I think Cedric is the person who created TestNG.

The fourth section is a link to all of my books which also includes VBScript for QTP/UFT. All books are available in eBook, PDF, and Paperback. This is Amazon's site but they are available anywhere books are sold. The Part 1 eBooks and PDF books are free. I could not make the Paperbacks free. The Part 1 books are geared toward beginners while Part 2 are for people with experience.

You can download your documents by going to https://tinyurl.com/Selenium-Building-Blocks-Part2. The zip file includes Test Cases, the presentation, automation code from this video, and practice automation code for verifying Demo Web Shop.

That's it and Thank You for watching (Part 2) Building Blocks For Selenium.