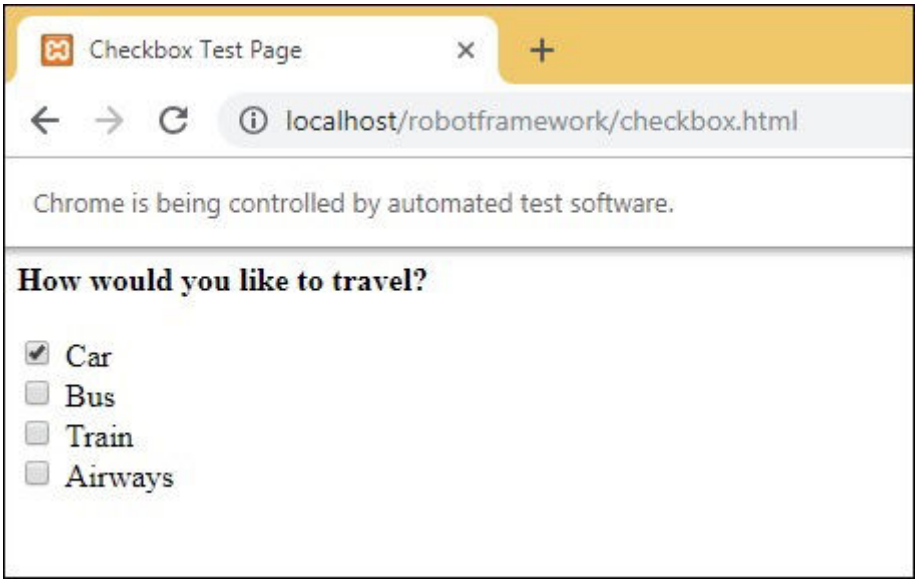
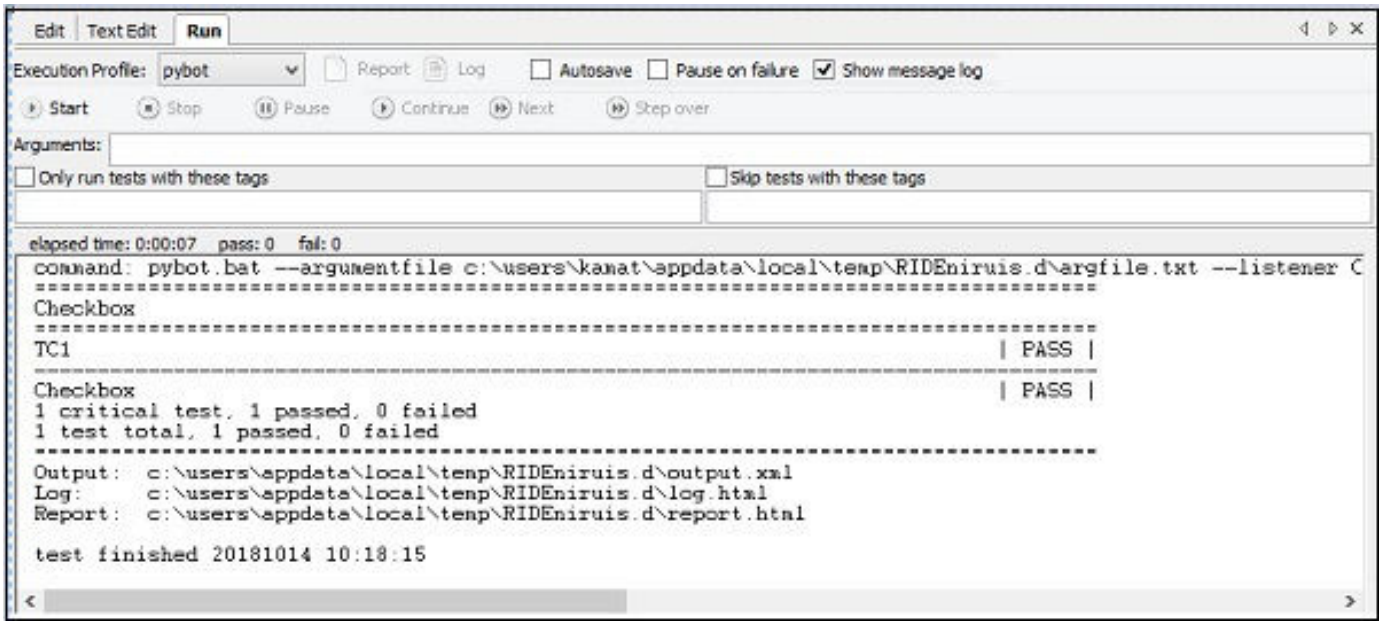
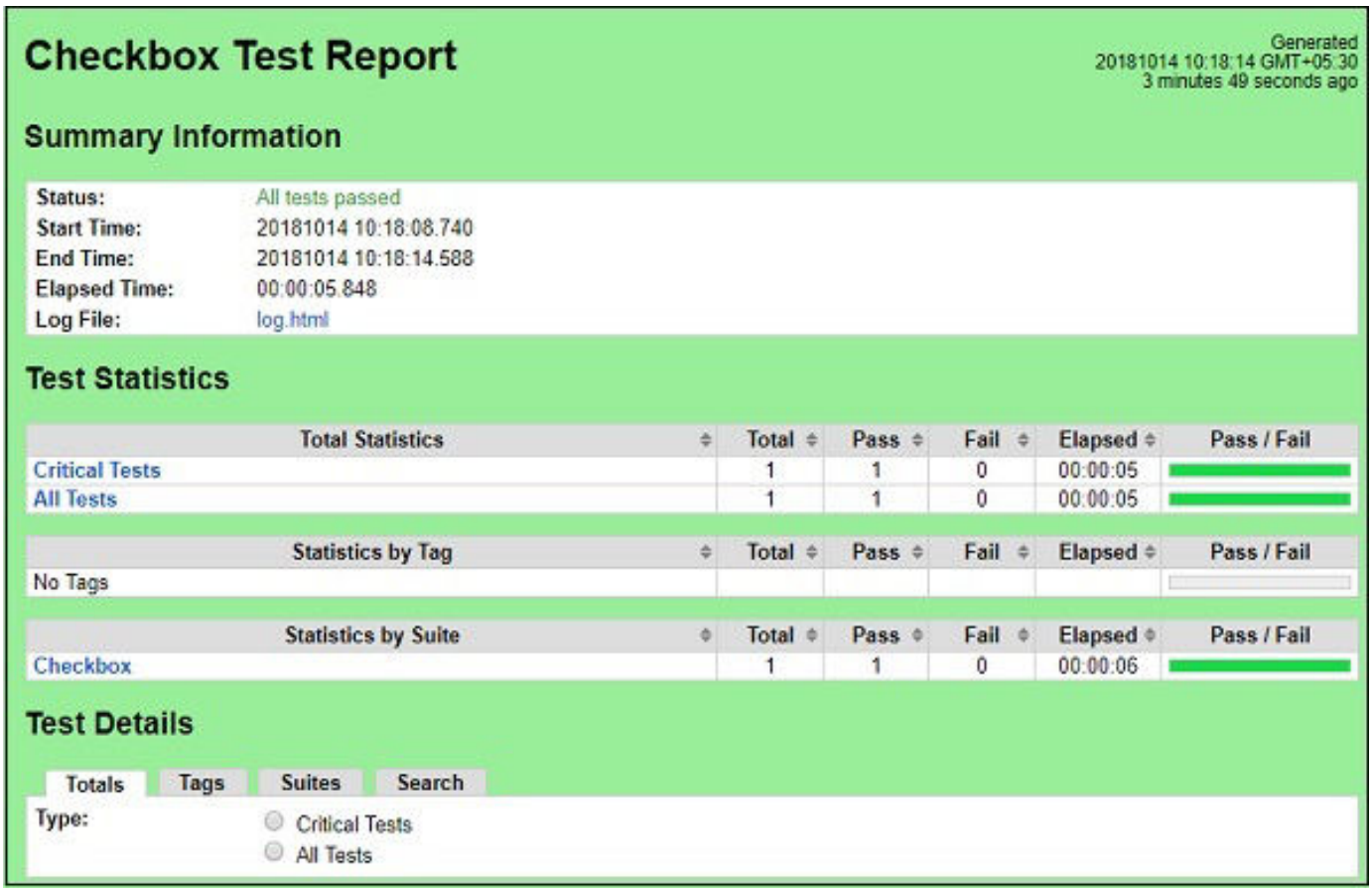


When the test case is executed, it opens the URL **http://localhost/robotframework/checkbox.html** and selects the name Car given in the test case.

Here are the execution details −



Details of Report



Checkbox Test Report

Generated
20181014 10:18:14 GMT+05:30
3 minutes 49 seconds ago

Summary Information

Status:	All tests passed
Start Time:	20181014 10:18:08.740
End Time:	20181014 10:18:14.588
Elapsed Time:	00:00:05.848
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	1	0	00:00:05	
All Tests	1	1	0	00:00:05	

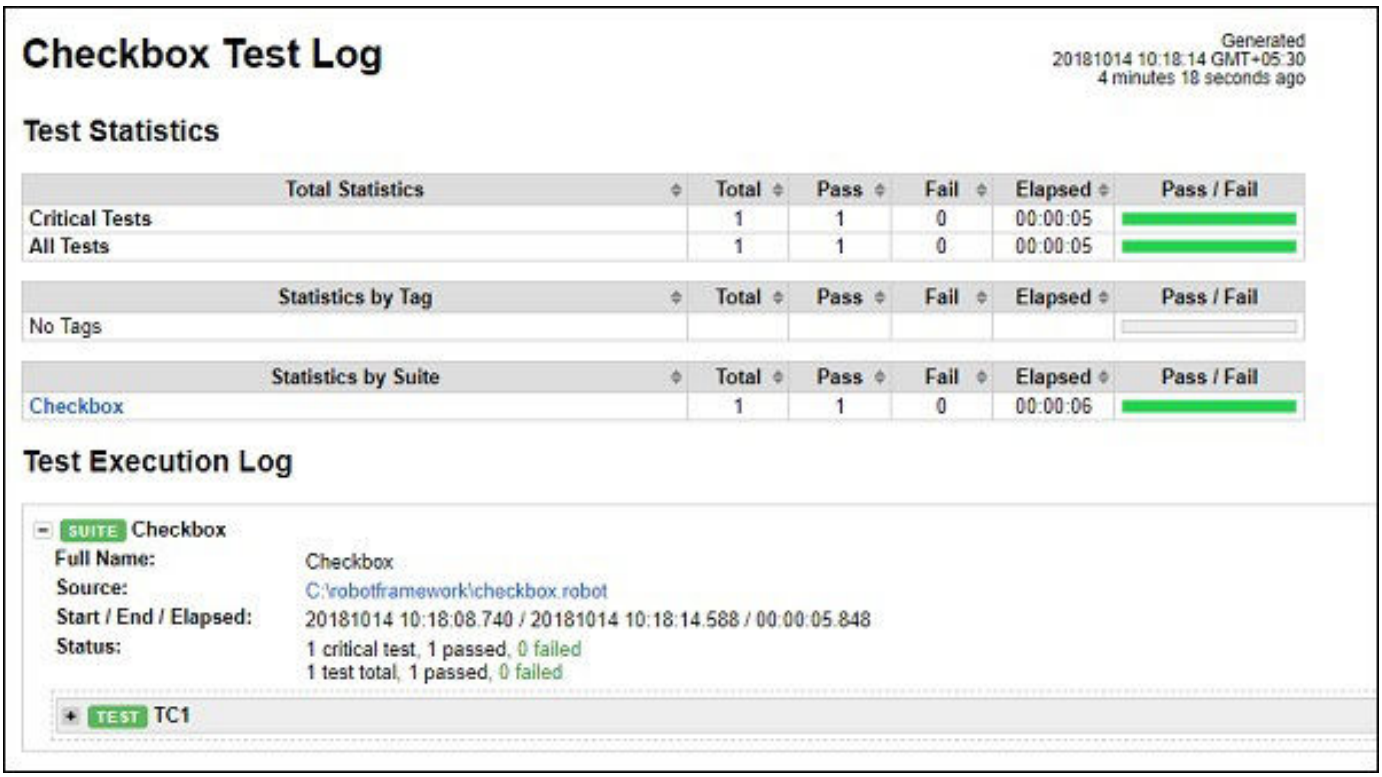
Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Checkbox	1	1	0	00:00:06	

Test Details

Totals | Tags | Suites | Search

Type: Critical Tests / All Tests



Checkbox Test Log

Generated
20181014 10:18:14 GMT+05:30
4 minutes 18 seconds ago

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	1	0	00:00:05	
All Tests	1	1	0	00:00:05	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Checkbox	1	1	0	00:00:06	

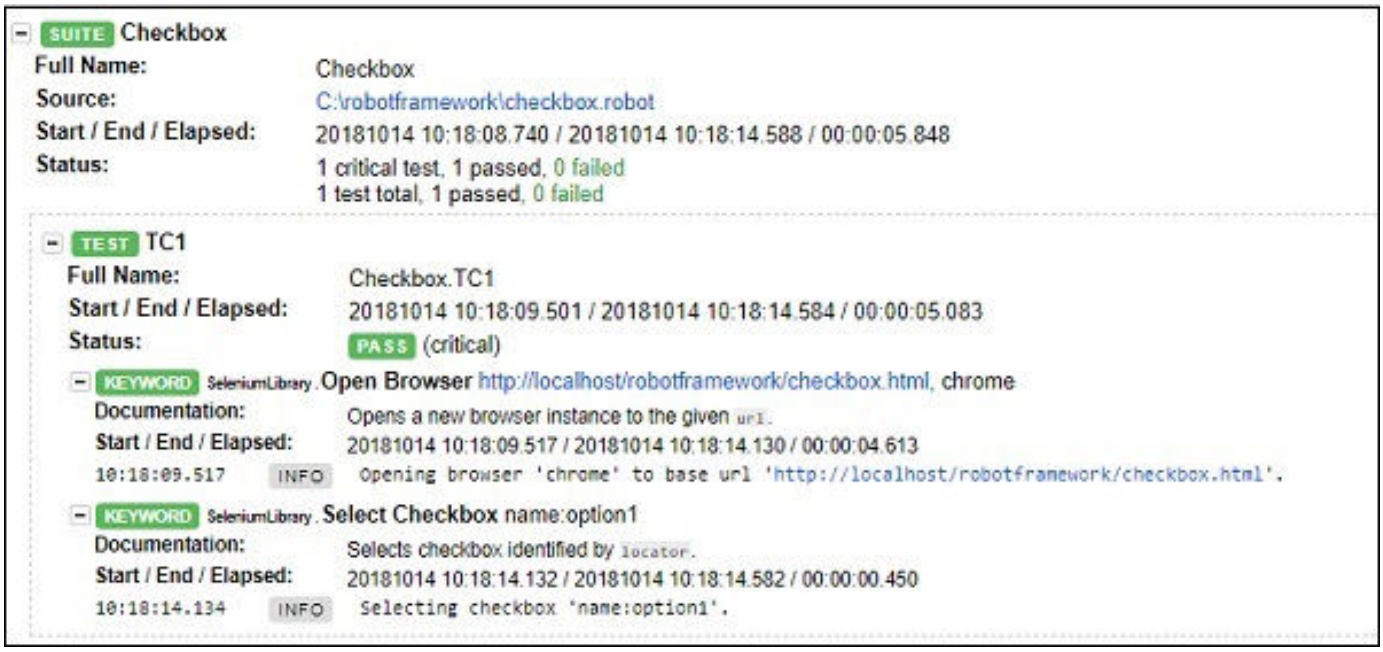
Test Execution Log

SUITE Checkbox

Full Name:	Checkbox
Source:	C:\robotframework\checkbox.robot
Start / End / Elapsed:	20181014 10:18:08.740 / 20181014 10:18:14.588 / 00:00:05.848
Status:	1 critical test, 1 passed, 0 failed 1 test total, 1 passed, 0 failed

TEST TC1

Details of Log



Conclusion

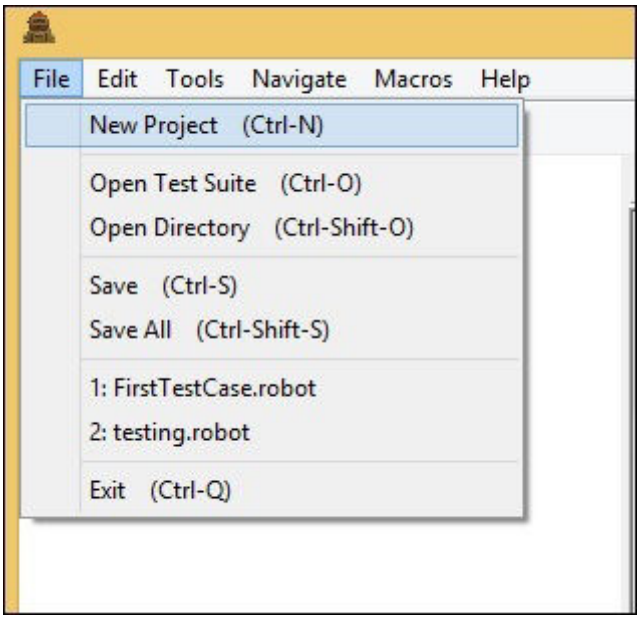
In this chapter, we learnt how we can select a checkbox by giving the locator of the checkbox. The log and Reports give the details of the execution of the test case along with the time spent for each test case.

Robot Framework - Working With Dropdown

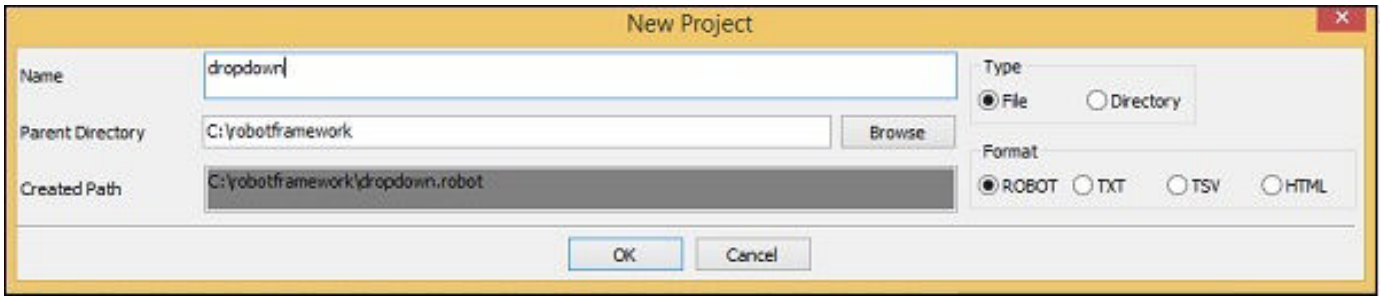
In this chapter, we will learn how to work with dropdown using Selenium Library.

Project Setup for Dropdown Testing

We will first create a project in Ride to work with browsers. Open ride using ride.py from the command line −

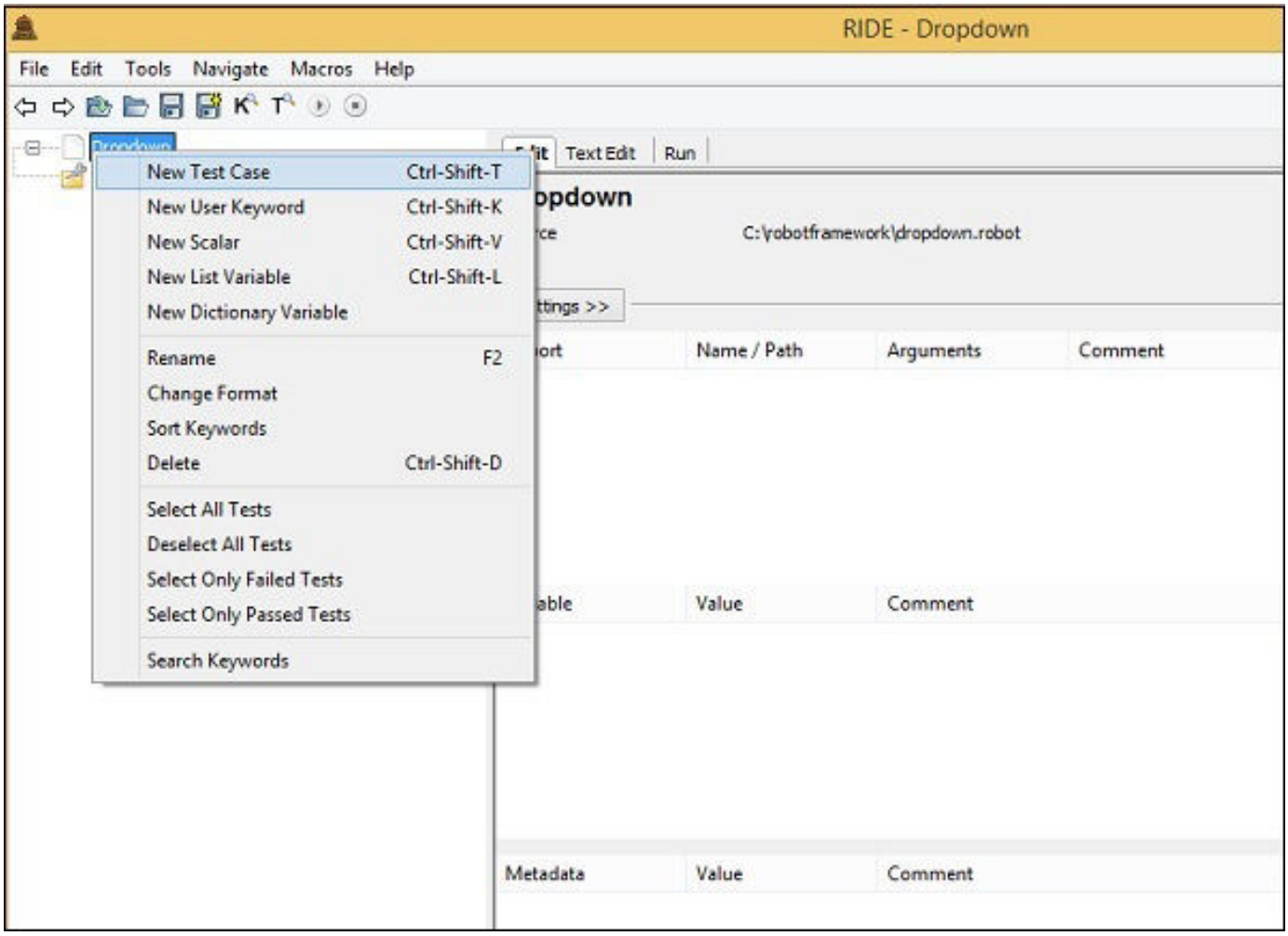


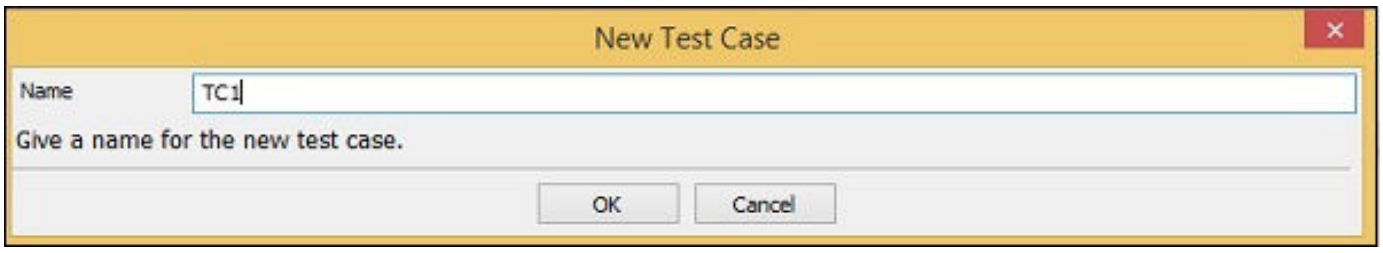
Click *New Project* and give a name to your project.



The name given is dropdown. Click OK to save the project.

Right−click on the name of the project created and click *New Test Case* −

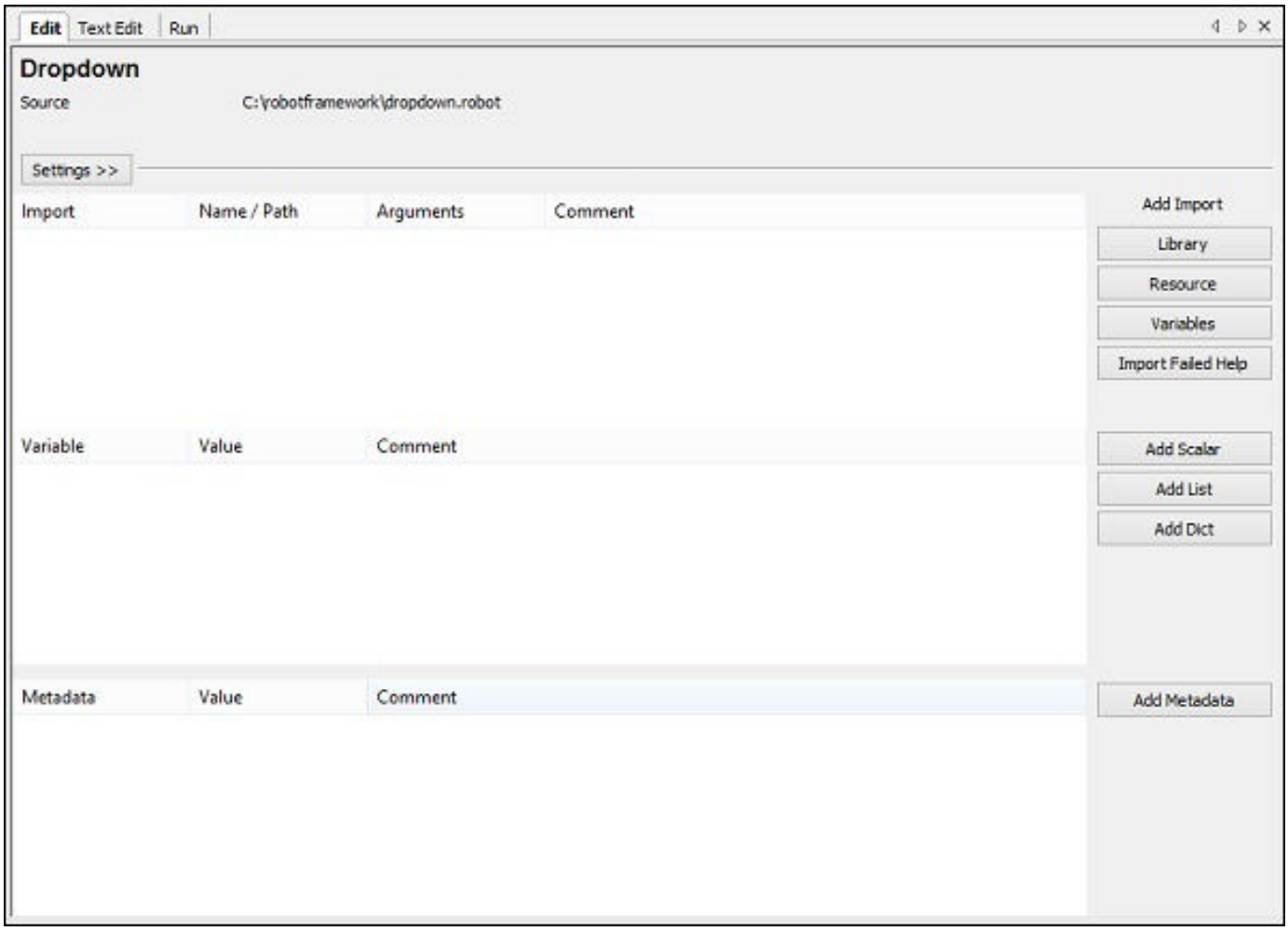




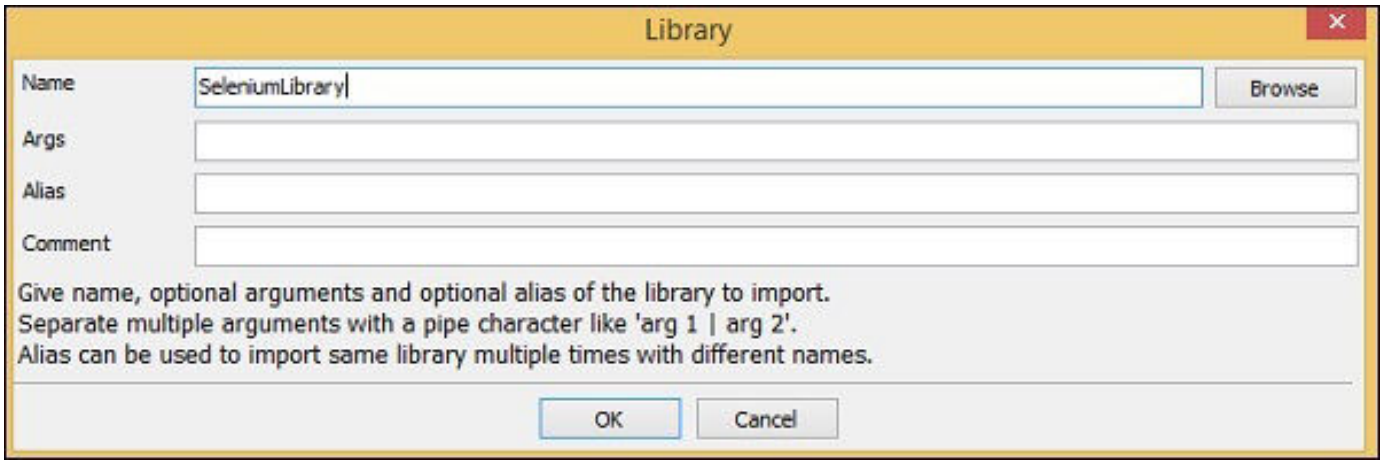
Give name to the test case and click OK to save it.

We are done with the project setup. Now, we will write test cases for the dropdown. Since we need Selenium library, we need to import the same in our project.

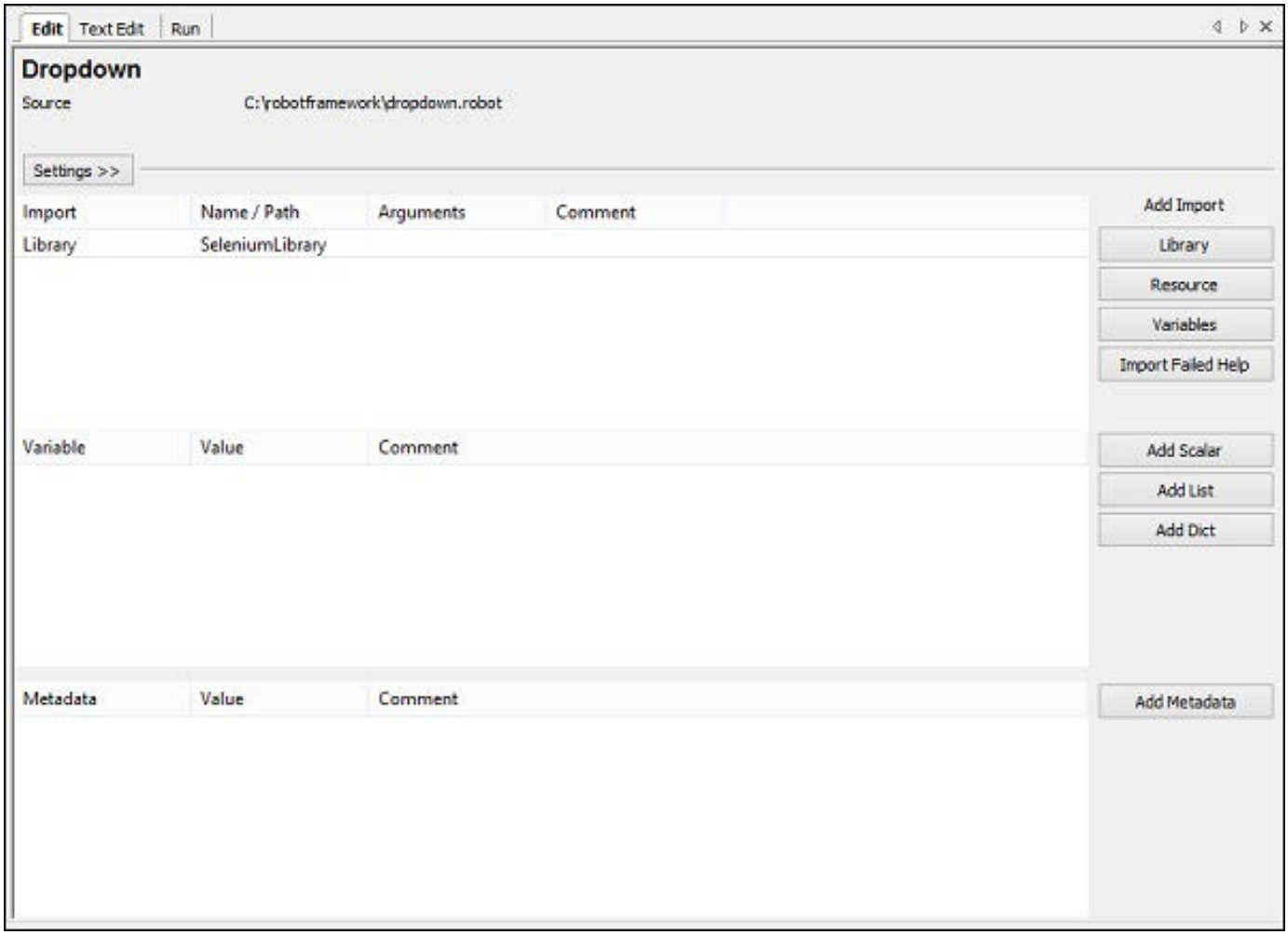
Click on your project on the left side and use *Library* from *Add Import* as shown below −



Now, click *Library*. A screen will appear where you need to enter the library name −

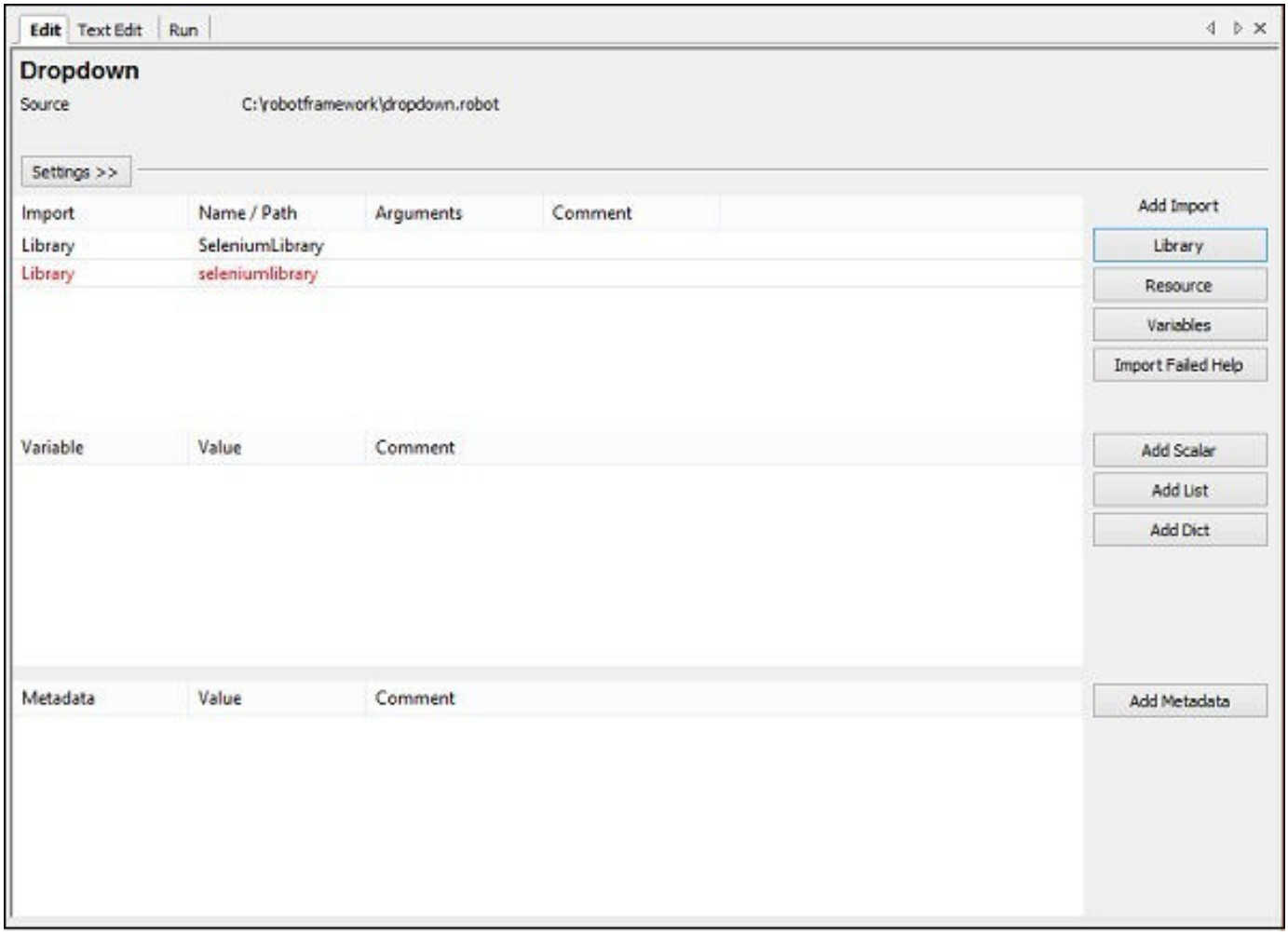


Click *OK* and the library will be displayed in the settings.



The name given has to match with the name of the folder installed in site-packages.

In case the name does not match, the library name will show in red −



Library import in red is as good as the library does not exists inside python.So now we are done with selenium library import.

Test Case for Dropdown

The test case for dropdown will select the value from the dropdown. To go about working with this, we need the locator (identifier) for that dropdown.

Consider the following html display for dropdown −

```
<select name = "carbrand">
   <option value = "">Select car brand..</option>
   <option value = "audi">AUDI</option>
   <option value = "bmw">BMW</option>
   <option value = "chevrolet">CHEVROLET</option>
   <option value = "datsun">DATSUN</option>
</select>
```

For dropdown, *name* is the *locator*. In the above example, the *name* is *carbrand*. We also need the value so that we can select the same. The values in the above example are – *audi, bmw, chevrolet and datsun*.

Now, we will create a test page with dropdown, open the same in the browser and select the value from the dropdown.

The test case details will be as follows −

- Open browser URL − **http://localhost/robotframework/dropdown.html** in chrome
- Enter details of dropdown
- Execute the test case

While writing the keyword for test cases in RIDE, press Ctrl + Spacebar. This gives all the details of the command.

For dropdown, we have three ways of doing it −

- Select From List By Index
- Select From List By Label
- Select From List By Value

We will work on an example to show working for all the cases mentioned above.

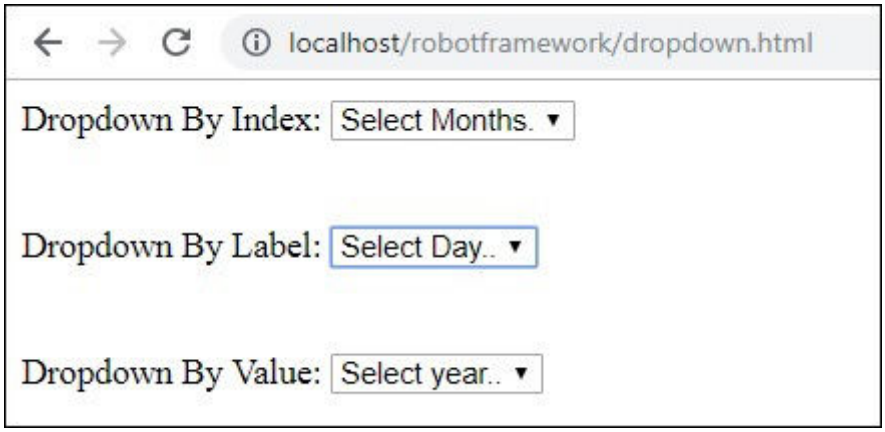
In our test page, we will create 3 dropdowns and will use above test cases to select the dropdown by index, label and value.

dropdown.html

```
<html>
   <head>
      <title>Dropdown</title>
   </head>
   <body>
      <form name="myform" method="POST">
         <div>
            Dropdown By Index:
            <select name = "months">
               <option value = "">Select Months.</option>
               <option value = "Jan">January</option>
               <option value = "Feb">February</option>
               <option value = "Mar">March</option>
               <option value = "Apr">April</option>
               <option value = "May">May</option>
               <option value = "Jun">June</option>
               <option value = "Jul">July</option>
               <option value = "Aug">August</option>
               <option value = "Sept">September</option>
               <option value = "Oct">October</option>
               <option value = "Nov">November</option>
               <option value = "Dec">December</option>
            </select>
         </div>
         <br/>
         <br/>
         <div>
            Dropdown By Label:
            <select name = "days">
               <option value = "">Select Day..</option>
```

```
            <option value = "01">01</option>
            <option value = "02">02</option>
            <option value = "03">03</option>
            <option value = "04">04</option>
            <option value = "05">05</option>
            <option value = "06">06</option>
            <option value = "07">07</option>
            <option value = "08">08</option>
            <option value = "09">09</option>
            <option value = "10">10</option>
            <option value = "11">11</option>
            <option value = "12">12</option>
            <option value = "13">13</option>
            <option value = "14">14</option>
            <option value = "15">15</option>
            <option value = "16">16</option>
            <option value = "17">17</option>
            <option value = "18">18</option>
            <option value = "19">19</option>
            <option value = "20">20</option>
            <option value = "21">21</option>
            <option value = "22">22</option>
            <option value = "23">23</option>
            <option value = "24">24</option>
            <option value = "25">25</option>
            <option value = "26">26</option>
            <option value = "27">27</option>
            <option value = "28">28</option>
            <option value = "29">29</option>
            <option value = "30">30</option>
            <option value = "31">31</option>
         </select>
      </div>
      <br/>
      <br/>
      <div>
         Dropdown By Value:
         <select name = "year">
            <option value = "">Select year..</option>
            <option value = "0">2000</option>
            <option value = "1">2001</option>
            <option value = "2">2002</option>
            <option value = "3">2003</option>
            <option value = "4">2004</option>
            <option value = "5">2005</option>
            <option value = "6">2006</option>
            <option value = "7">2007</option>
            <option value = "8">2008</option>
            <option value = "9">2009</option>
```

```
                <option value = "10">2010</option>
                <option value = "11">2011</option>
                <option value = "12">2012</option>
                <option value = "13">2013</option>
                <option value = "14">2014</option>
                <option value = "15">2015</option>
                <option value = "16">2016</option>
                <option value = "17">2017</option>
                <option value = "18">2018</option>
            </select>
        </div>
    </form>
  </body>
</html>
```



We will add test cases for all 3 dropdown selection in Ride.

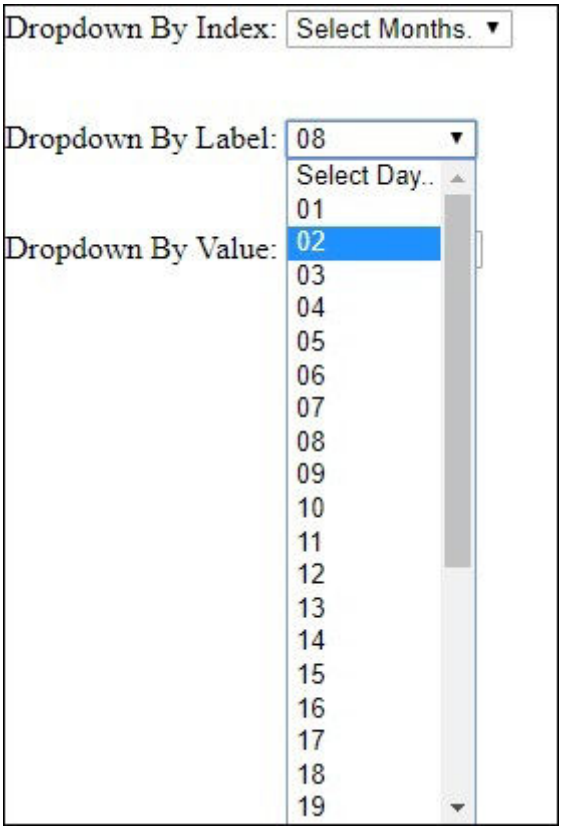
For index, we need to pass the locator of that dropdown – name or id and the index of the element that needs to be selected.

Select List by Index – Example

```
<select name = "months">
   <option value = "">Select Months.</option> // index 0
   <option value = "Jan">January</option> //index 1
   <option value = "Feb">February</option> // index 2
   <option value = "Mar">March</option> // index 3
   <option value = "Apr">April</option> // index 4
   <option value = "May">May</option> // index 5
   <option value = "Jun">June</option> // index 6
   <option value = "Jul">July</option> // index 7
   <option value = "Aug">August</option> // index 8
   <option value = "Sept">September</option> //index 9
   <option value = "Oct">October</option> //index 10
   <option value = "Nov">November</option> //index 11
   <option value = "Dec">December</option> // index 12
</select>
```

Now, we want to select month as May so the index to be given in the test case is 5.

Label is seen when you open the dropdown on screen.



If you want to select a day, you can choose one from the dropdown.

Select From List by Value

Here is the list of the year. The list has values from 0 to 18.

```
<select name = "year">
   <option value = "">Select year..</option>
   <option value = "0">2000</option>
   <option value = "1">2001</option>
   <option value = "2">2002</option>
   <option value = "3">2003</option>
   <option value = "4">2004</option>
   <option value = "5">2005</option>
   <option value = "6">2006</option>
   <option value = "7">2007</option>
   <option value = "8">2008</option>
   <option value = "9">2009</option>
   <option value = "10">2010</option>
   <option value = "11">2011</option>
   <option value = "12">2012</option>
   <option value = "13">2013</option>
   <option value = "14">2014</option>
   <option value = "15">2015</option>
   <option value = "16">2016</option>
   <option value = "17">2017</option>
```

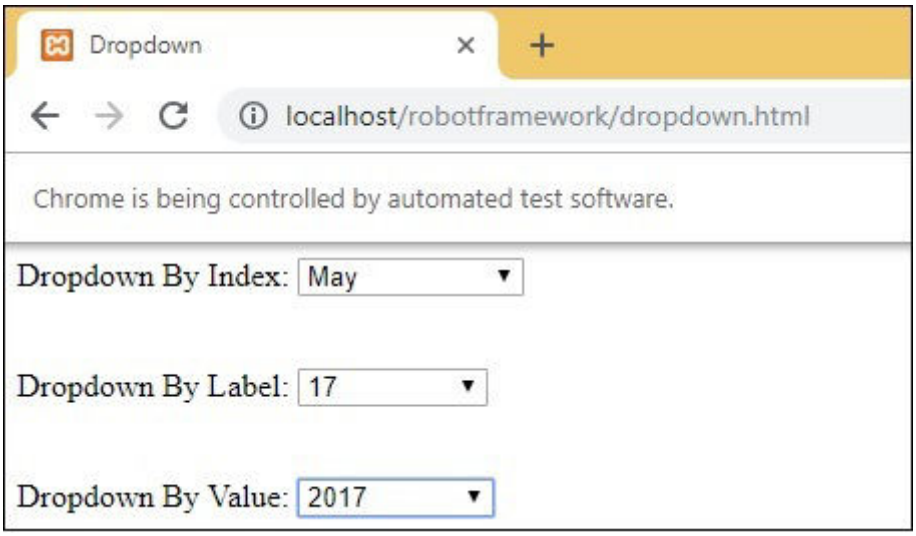
```
   <option value = "18">2018</option>
</select>
```

If you want to select any year, take the value corresponding to the year and add the same in test case. For example, if you want to select year 2017 the value is 17.

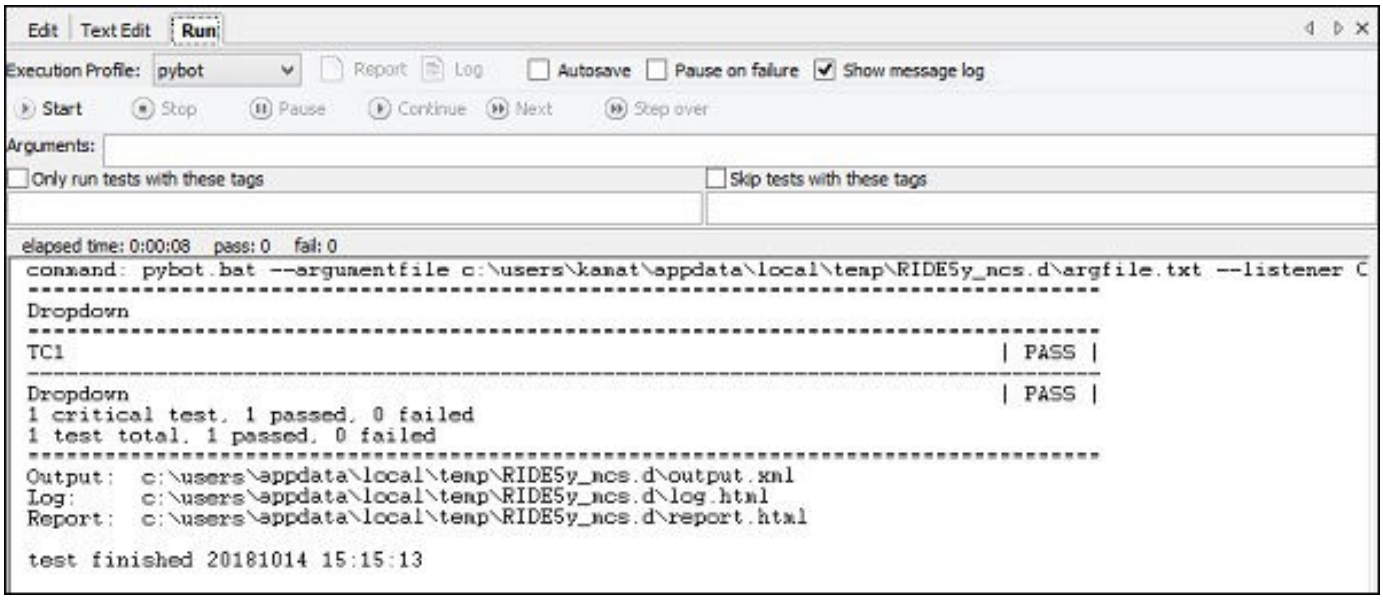
Here is the final list of test cases −

1	Open Browser	http://localhost/robotframework/d	chrome		
2	Select From List By Index	name:months	5		
3	Select From List By Label	name:days	17		
4	Select From List By Value	name:year	17		
5					
6					
7					

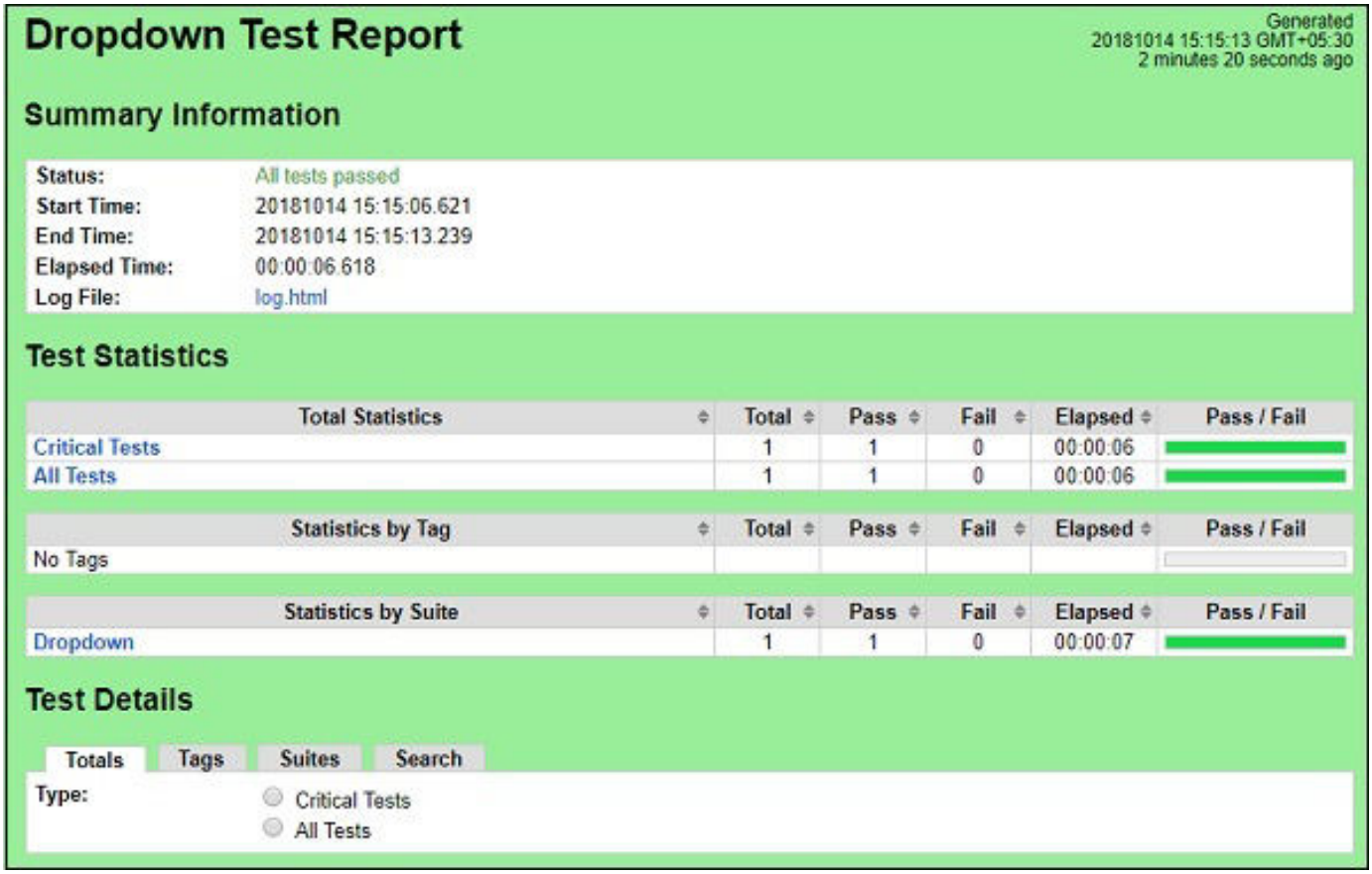
After execution, here is the selection done for dropdowns based on the test case −



Execution Details



Report Details



Dropdown Test Report

Generated
20181014 15:15:13 GMT+05:30
2 minutes 20 seconds ago

Summary Information

Status:	All tests passed
Start Time:	20181014 15:15:06.621
End Time:	20181014 15:15:13.239
Elapsed Time:	00:00:06.618
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	1	0	00:00:06	
All Tests	1	1	0	00:00:06	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

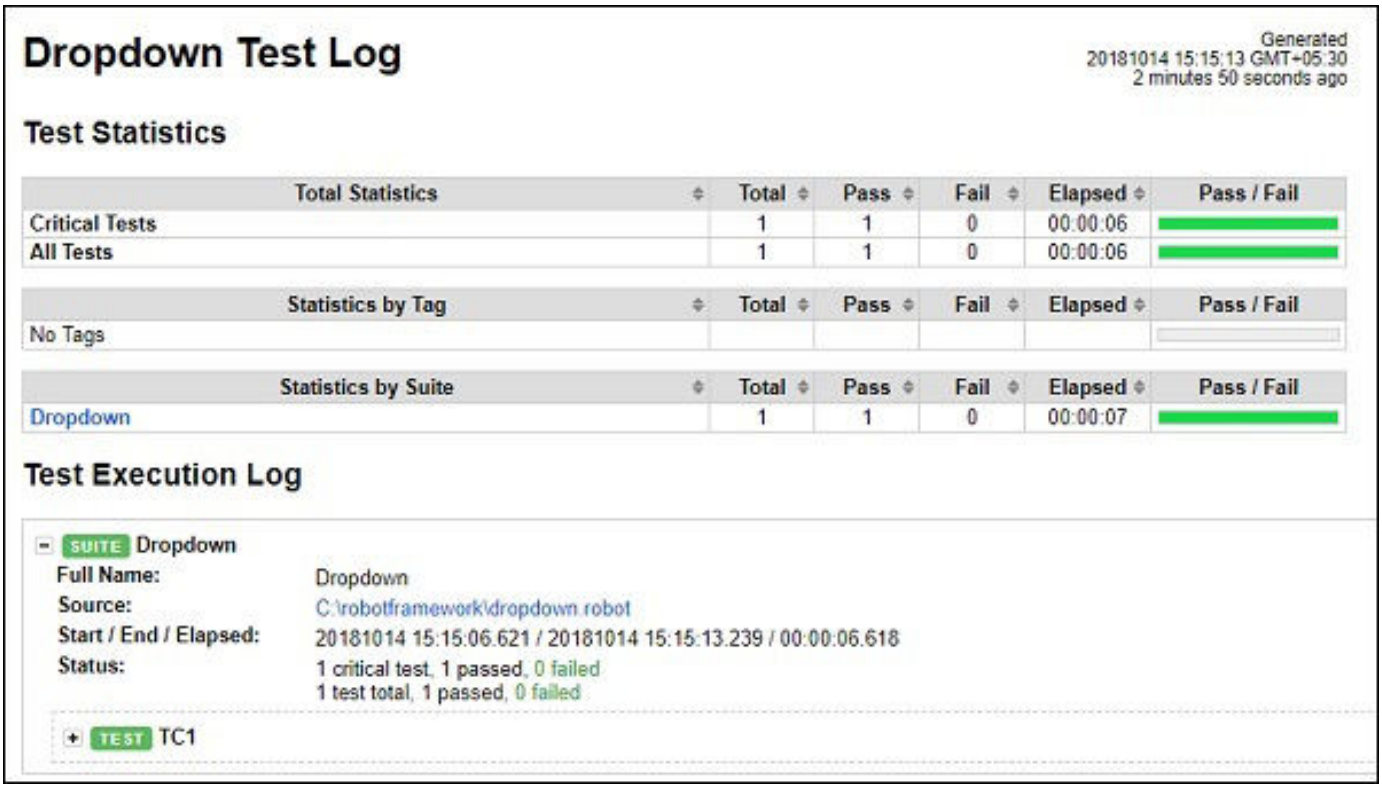
Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Dropdown	1	1	0	00:00:07	

Test Details

Totals | Tags | Suites | Search

Type: ○ Critical Tests ○ All Tests

Log Details



Dropdown Test Log

Generated
20181014 15:15:13 GMT+05:30
2 minutes 50 seconds ago

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	1	0	00:00:06	
All Tests	1	1	0	00:00:06	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Dropdown	1	1	0	00:00:07	

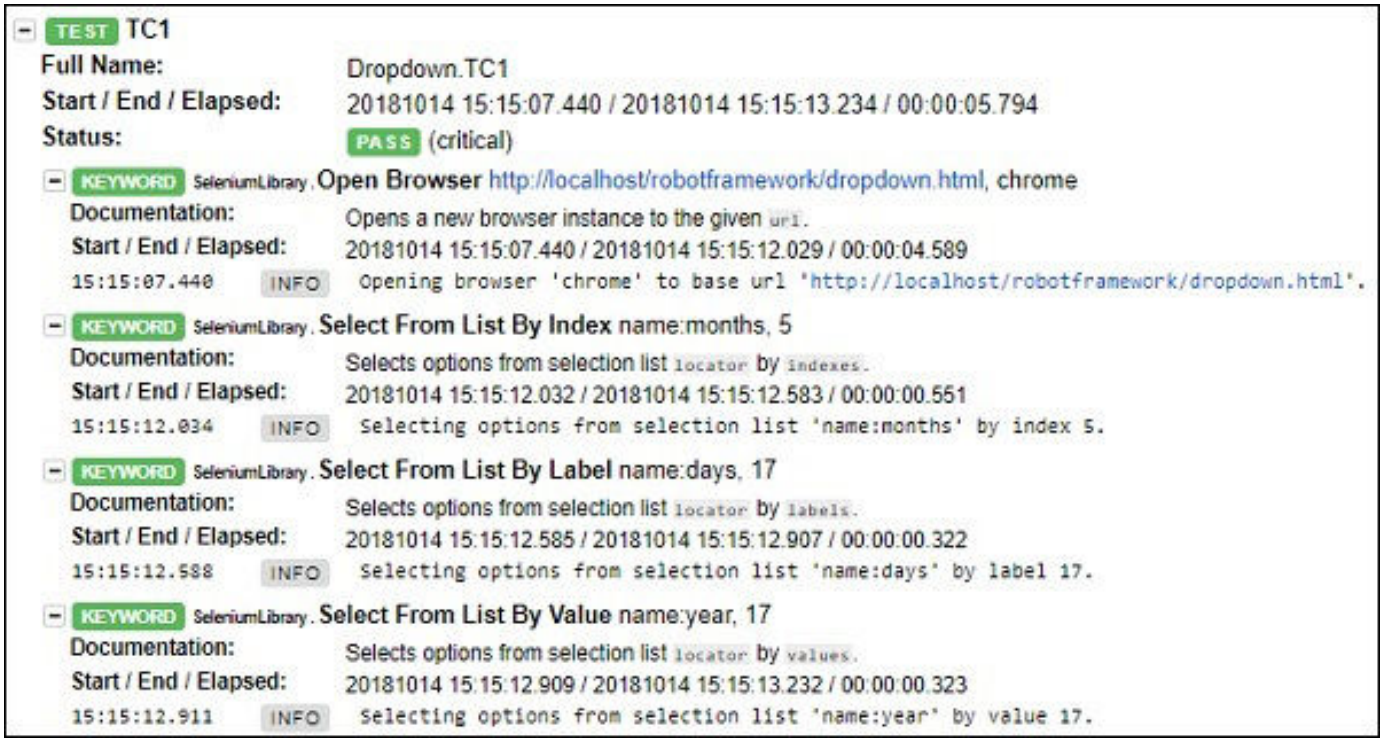
Test Execution Log

SUITE Dropdown

Full Name:	Dropdown
Source:	C:\robotframework\dropdown robot
Start / End / Elapsed:	20181014 15:15:06.621 / 20181014 15:15:13.239 / 00:00:06.618
Status:	1 critical test, 1 passed, 0 failed 1 test total, 1 passed, 0 failed

TEST TC1

Test case details from log



Conclusion

We have seen how to work with dropdown by value, index and label. We can refer to logs and reports to get the details of the test case executed.

Robot Framework - Working With Keywords

In Robot Framework, test cases are constructed in test case tables using keywords. In this chapter, we will cover the details on keywords used in Robot Framework. There are 2 types of keywords used in Robot −

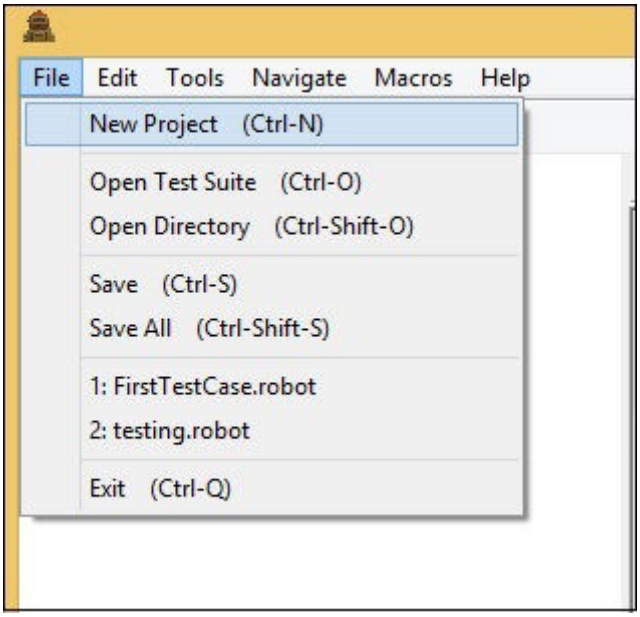
- Library Keywords
- User Defined Keywords

Library Keywords

Library Keywords are keywords that come from the library we import in Robot Framework. We will now take a look at the Selenium library, which helps us interact with the browser. We will discuss some of the important keywords associated with selenium library.

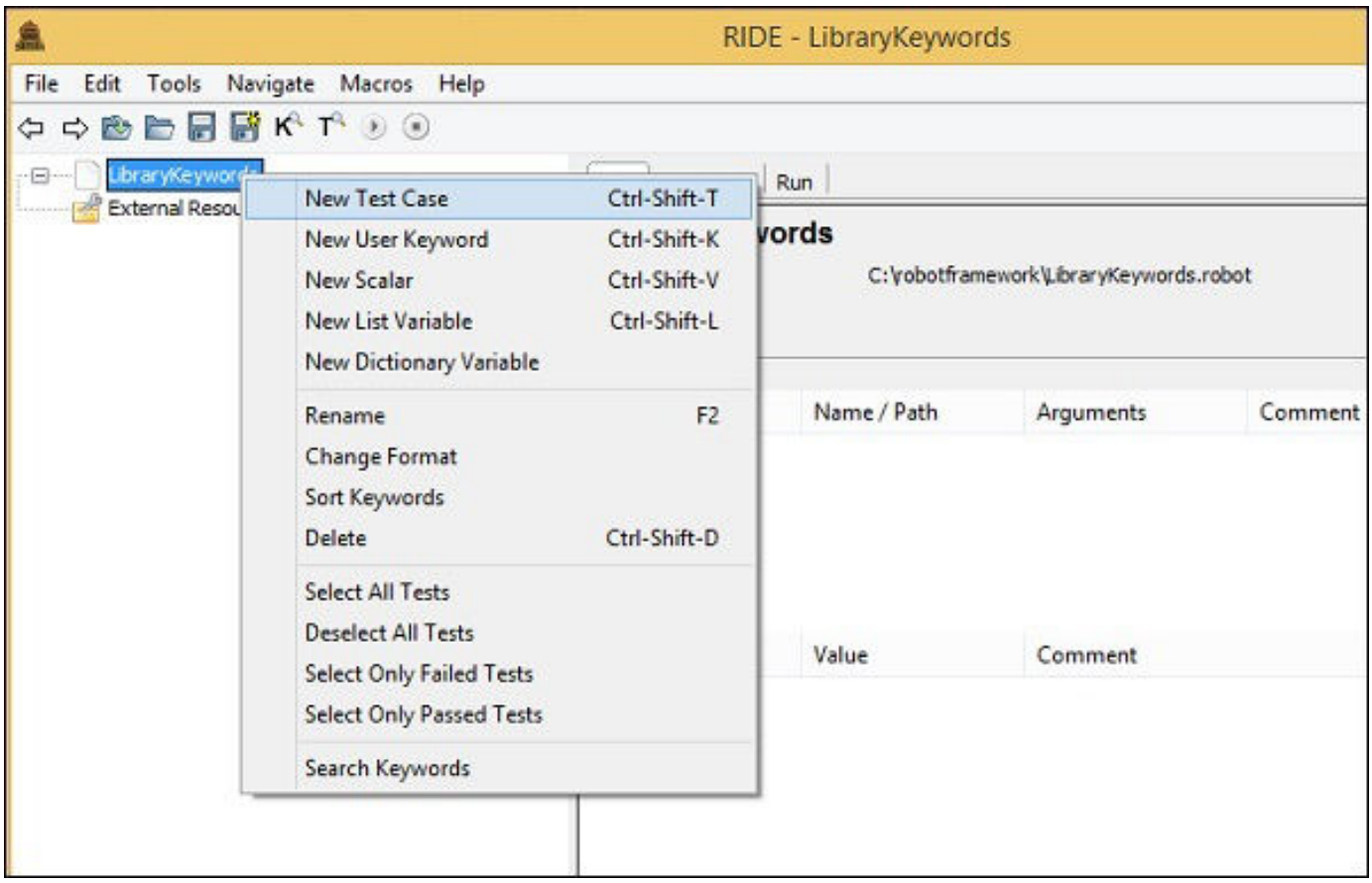
Follow the steps shown below to import Selenium library −

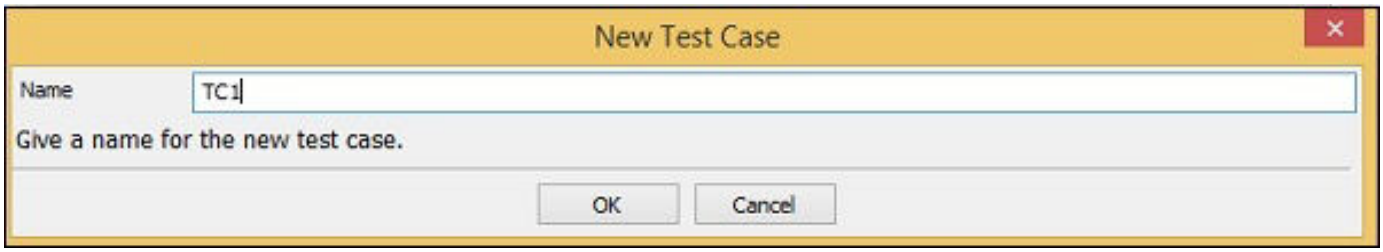
The details relating to the installation of Selenium library is discussed in chapter **“Working with Browsers using Selenium Library ”**. Open ride using ride.py from the command line.



Click on New Project and give name to your project. The name given to the project is **LibraryKeywords**.

Right-click on the name of the project created and click on *New Test Case* −

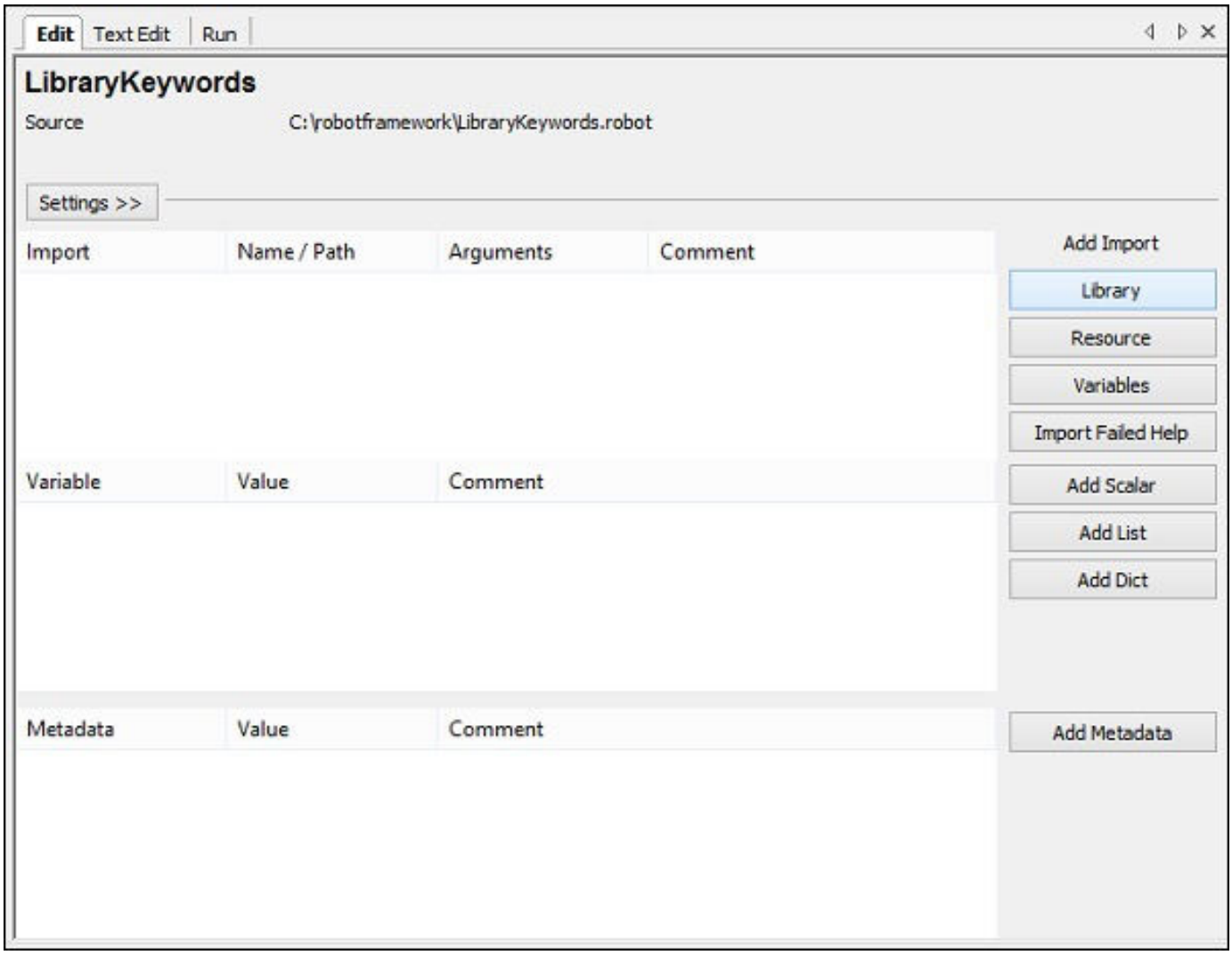




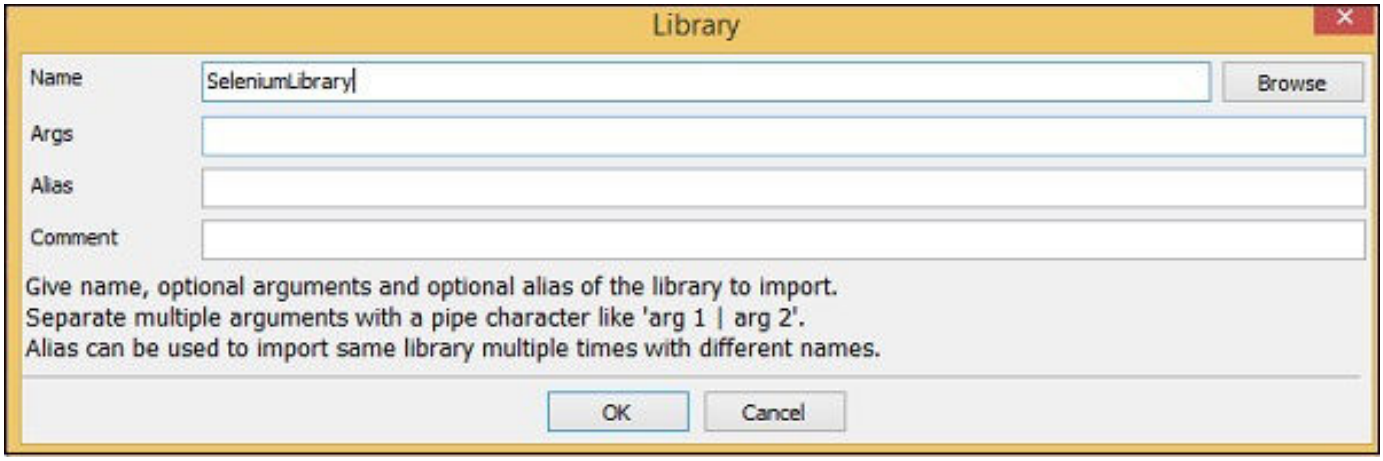
Give a name to the test case and click OK.

We are done with the project setup. Now, we will write test cases to show the working of library keywords. Since we need Selenium library, we need to import the same in our project.

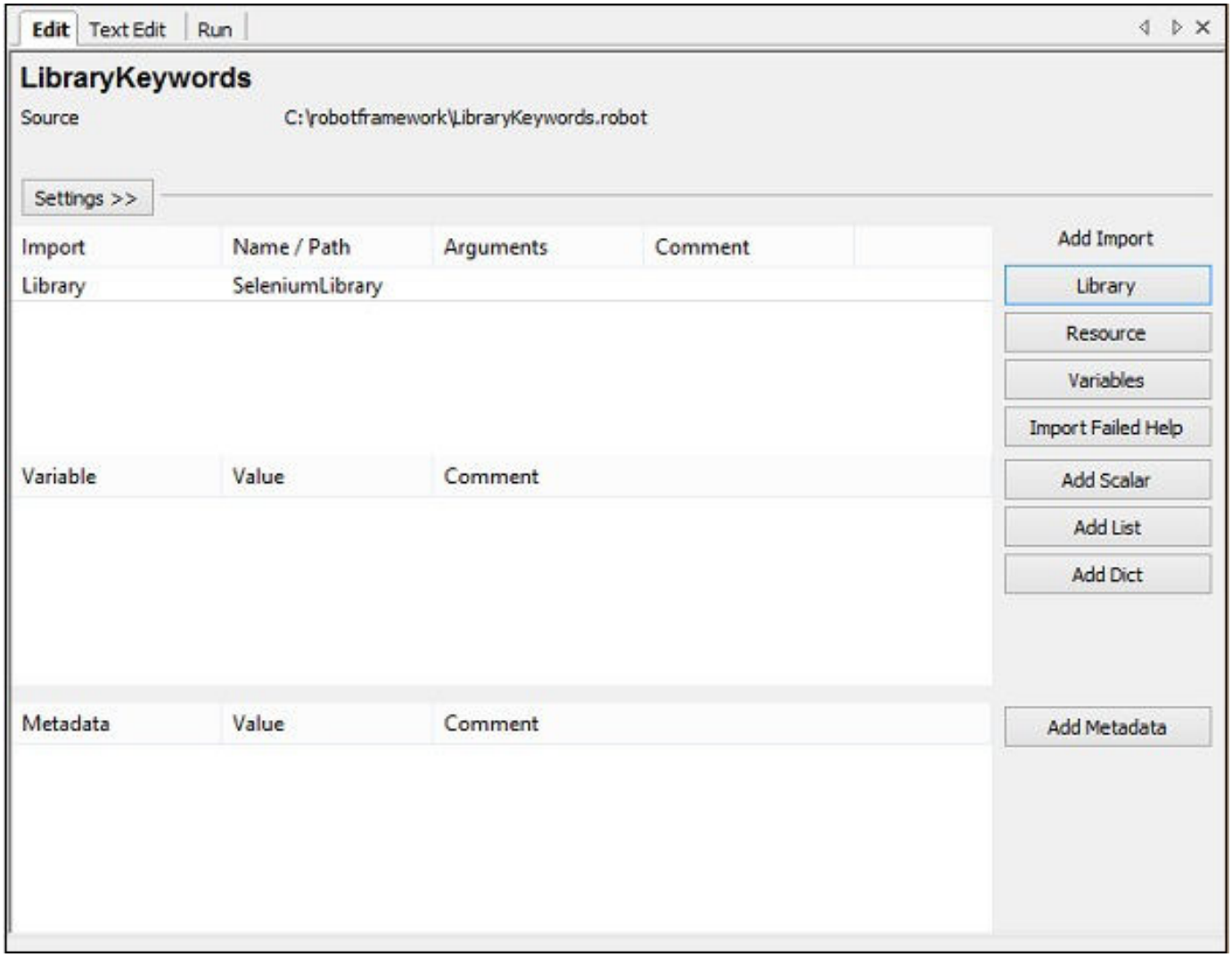
Click on your project on the left side and click Library.



Upon clicking Library, a screen will appear where you need to enter the library name −



Click OK and the library will get displayed in the settings.

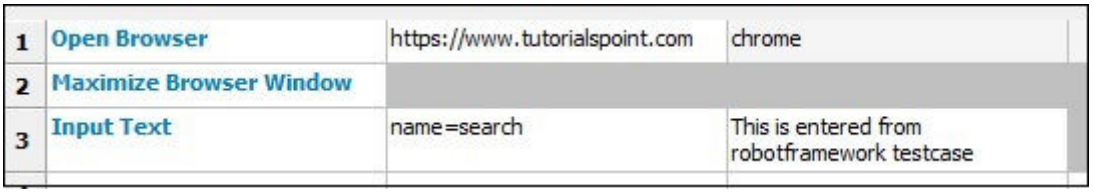


The name given has to match with the name of the folder installed in site-packages.

Now will create test case in the project created and use a few important keywords.

Click on your test case created TC1 and in the tabular form enter the keywords to open the browser and enter data inside the form opened.

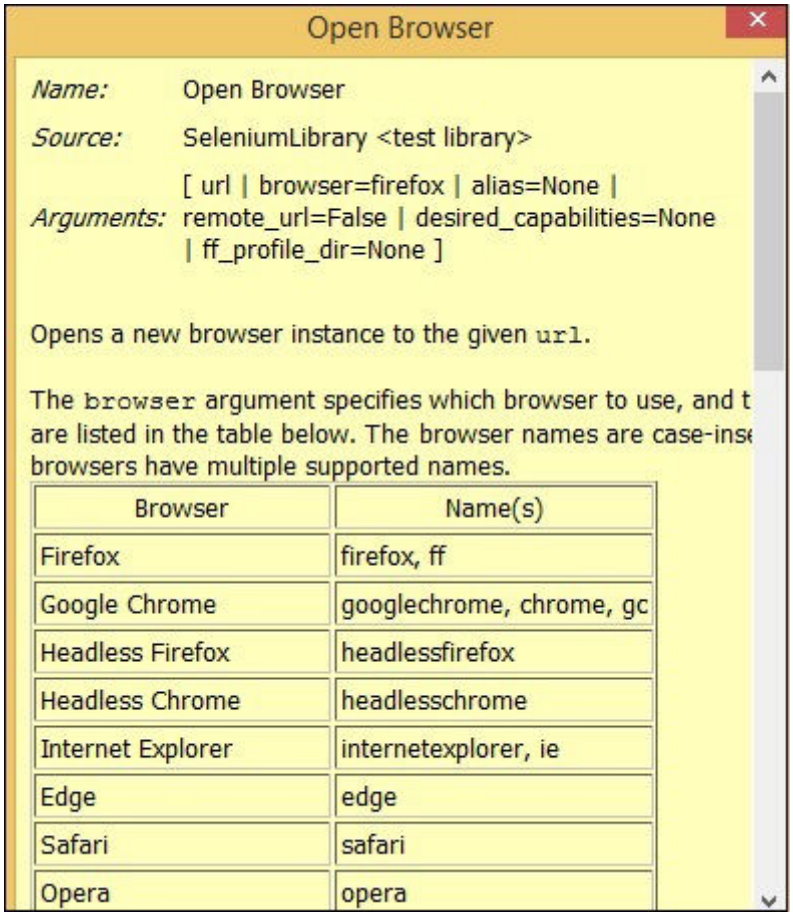
Here is a simple test case using Library Keywords −



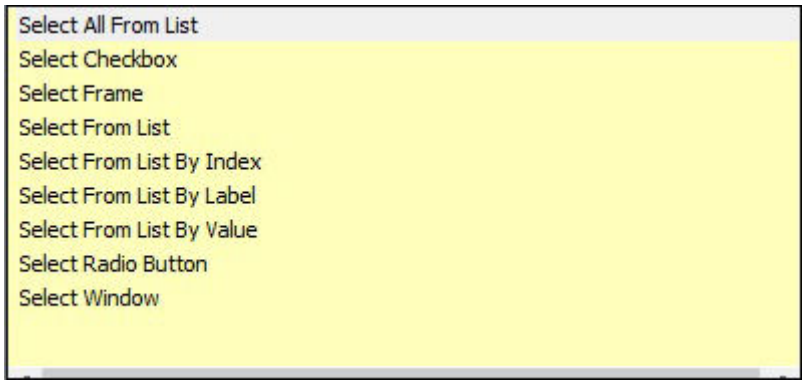
To get more details of this keyword, while typing the keyword press ctrl + spacebar. It will show the details of the library keyword entered.

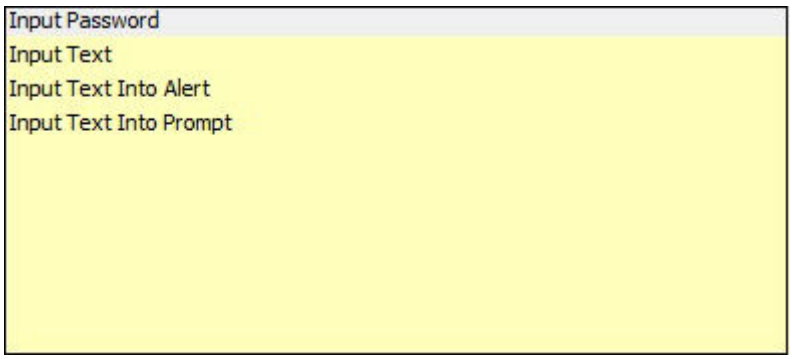
Here is an example for Open Browser, and if any help required for that keyword you can make use of ctrl + spacebar while typing the keyword.

Open Browser Keyword Details

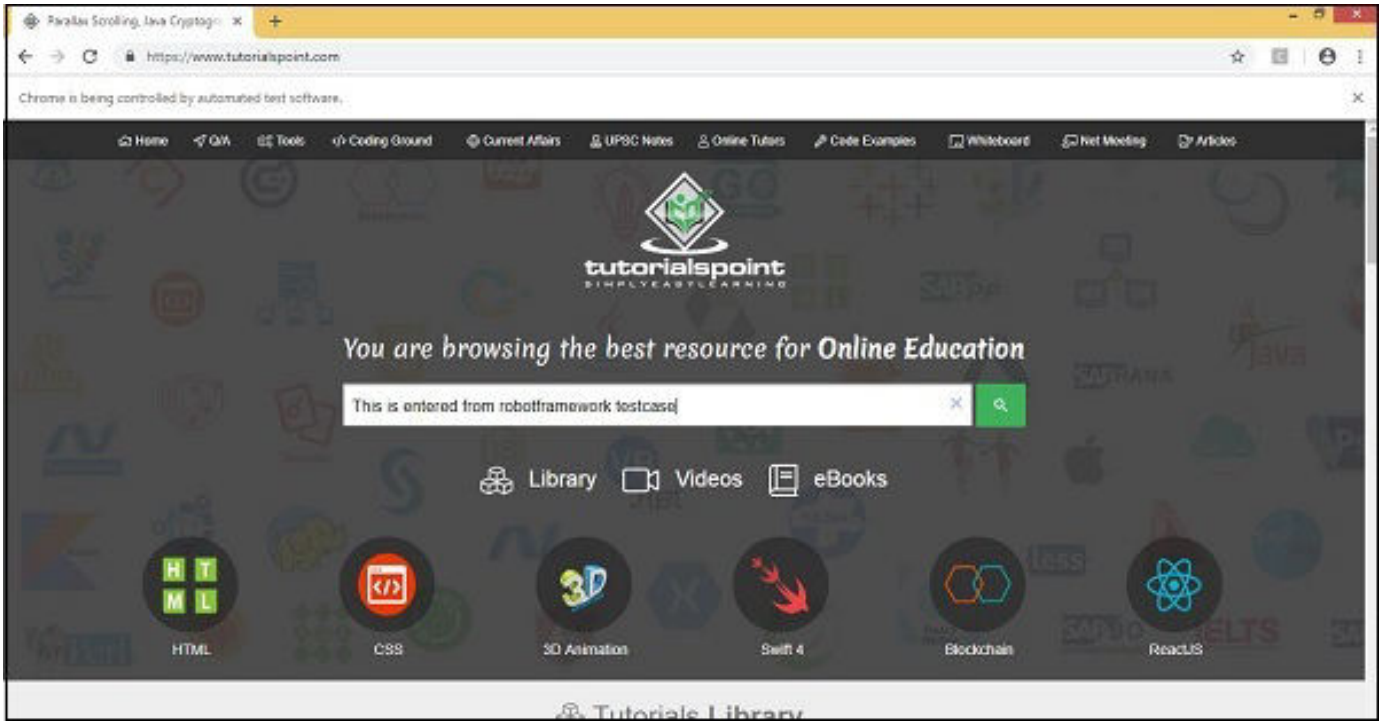


Similarly, we have Library keywords to work with Input, Radio, Text, etc





We will execute the test case we entered to open the browser with URL − https://www.tutorialspoint.com/ and enter details in the input text.



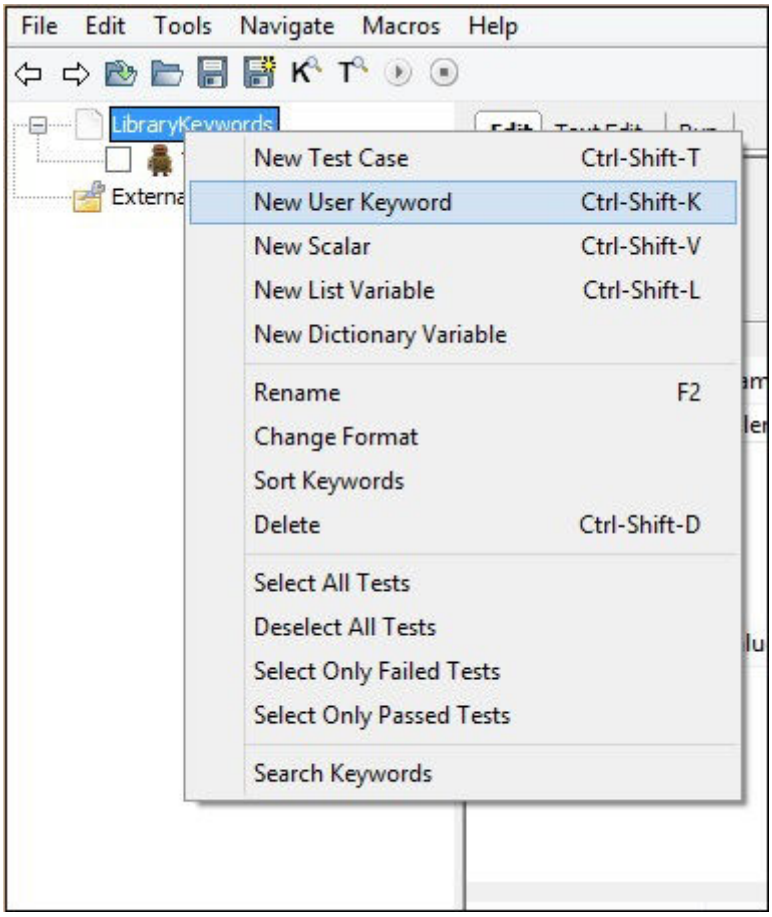
We have executed the test case. You can see the textbox has all the details we gave in the test case.

User-defined Keywords

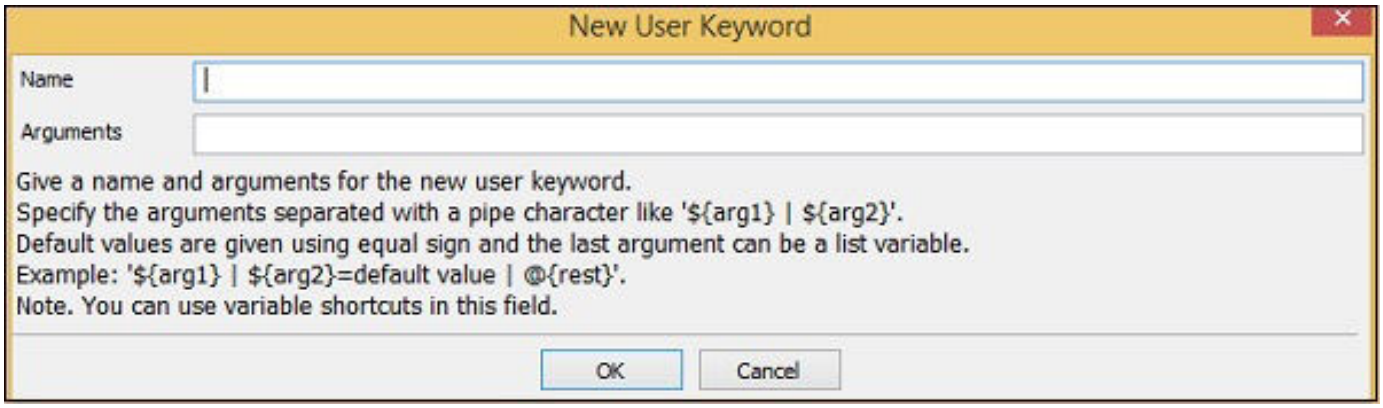
User-defined keywords can be created to perform a particular action in the test case or it can also be created using the library keywords and built-in keywords in robot framework. We will work on an example and see how we can create keywords for our test case.

We will use the same project that we created above and create user-defined keywords in that and use in the test case.

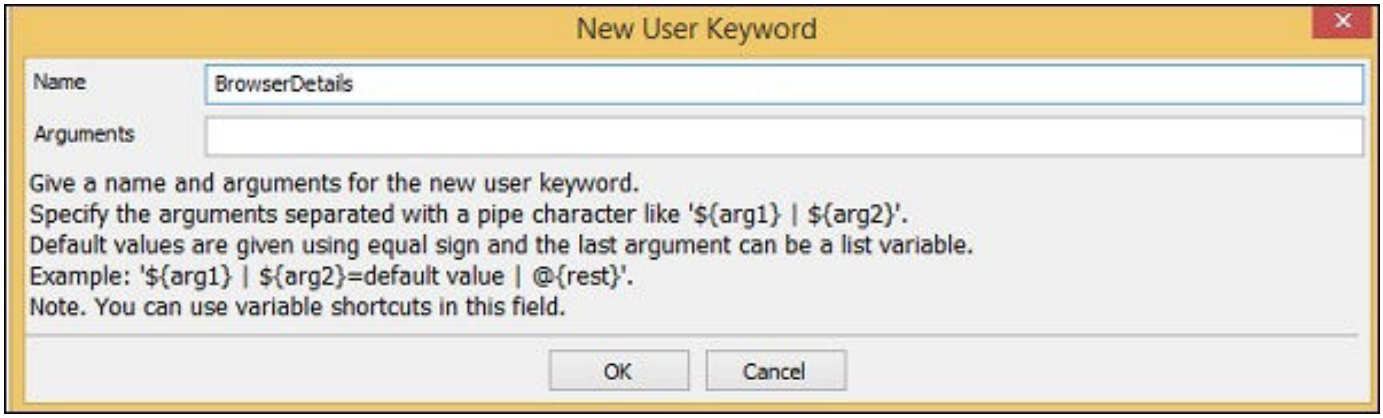
To create keyword in Ride, right-click on your project and click on New User Keyword as shown below −



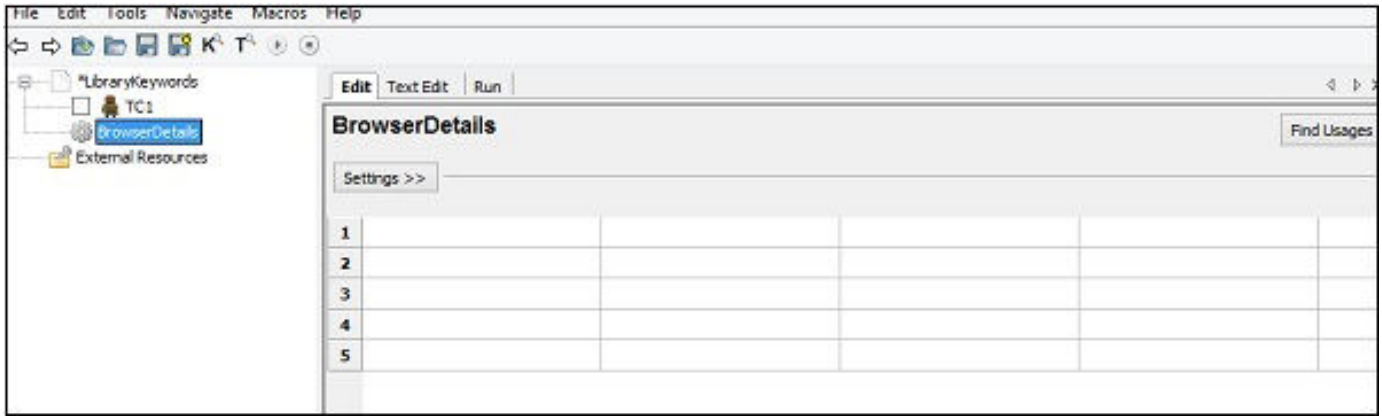
Upon clicking New User Keyword, a screen appears as shown below −



Enter the Name of the keyword and click OK. The screen also shows Arguments. We will discuss what arguments have to do with Keywords in a subsequent section.

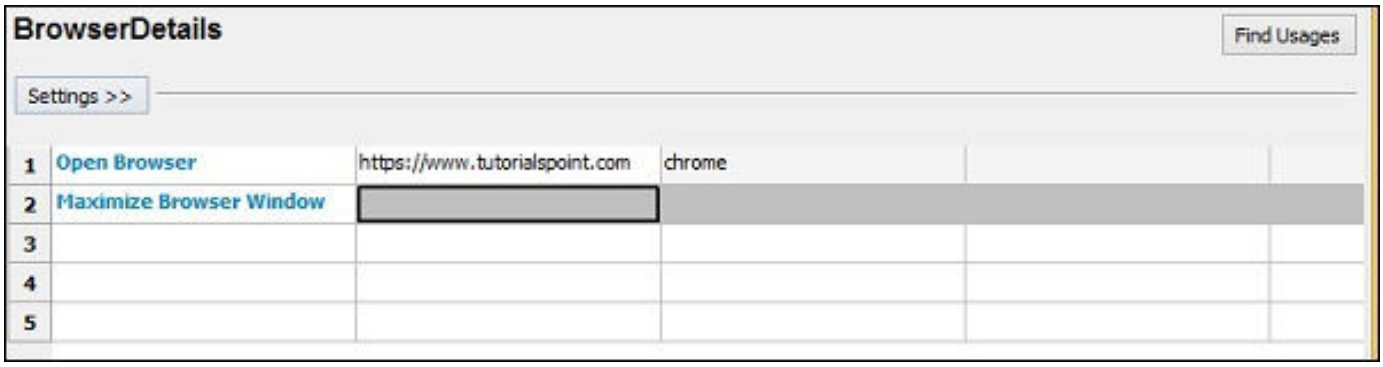


We have given the name BrowserDetails to the keyword. Click OK to save it. The keyword BrowserDetails is created.



To test the URL in the browser, we repeatedly have to enter open browser, **maximize browser** keywords.

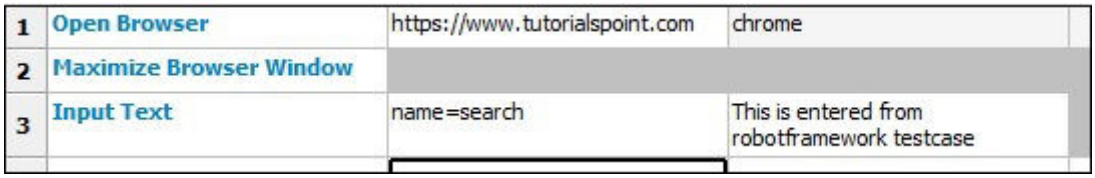
Now, we will create a user-defined keyword that will have *open browser* and *maximize browser details*. The keyword created will be used in our test case.



Our BrowserDetails keyword is a combination of other keywords used repeatedly.

Now, we will use the keyword created in the test case as shown below.

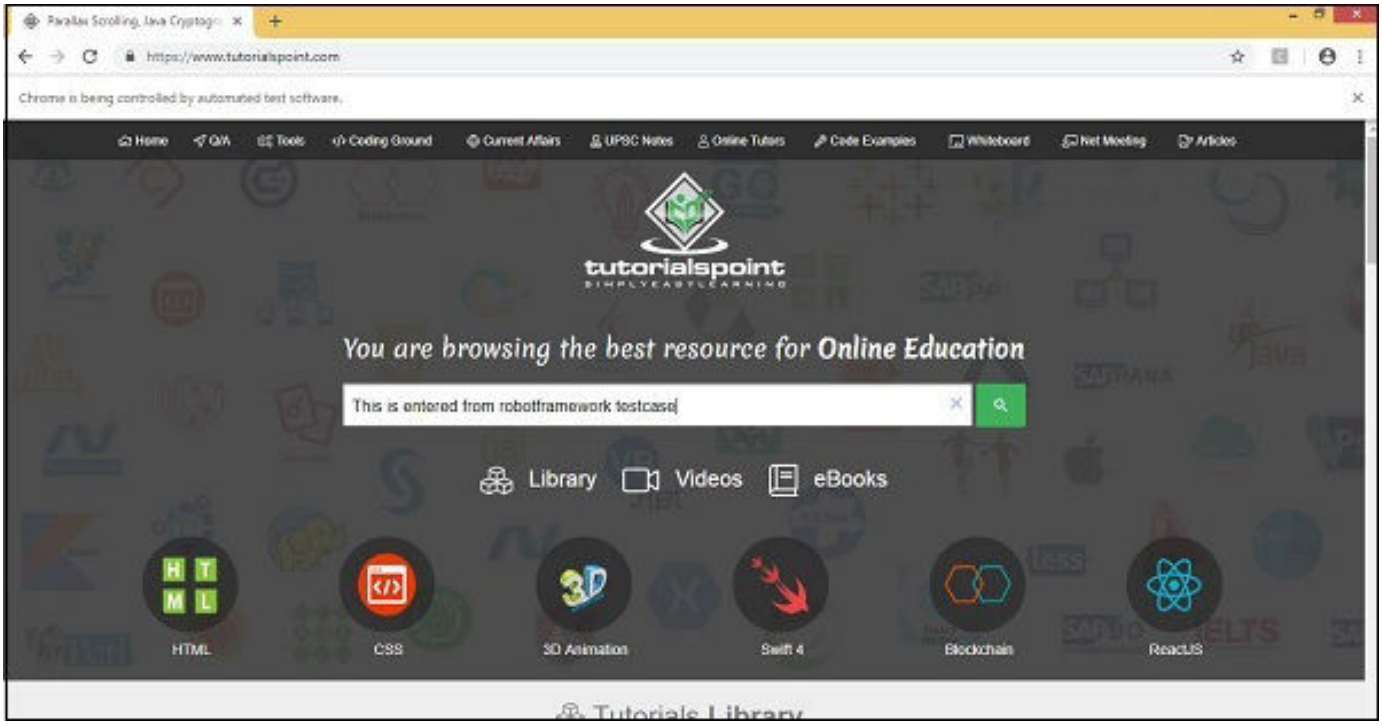
Test case



Considering the above test case, we are going to use the user-defined keyword *BrowserDetails*.

We will now replace 1 and 2 keywords with the user-defined keyword −

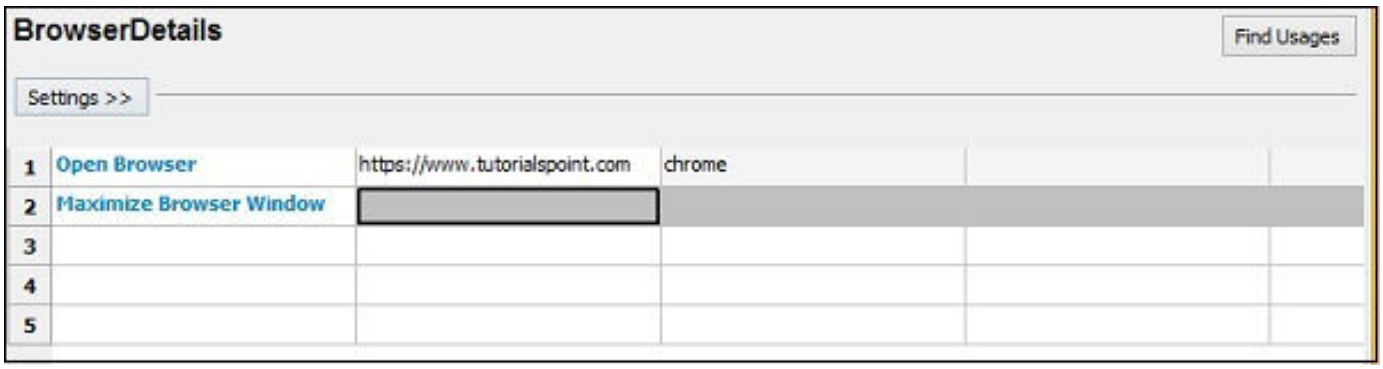
Let us now run the test case to see the output −



The execution of the test case works perfectly fine.

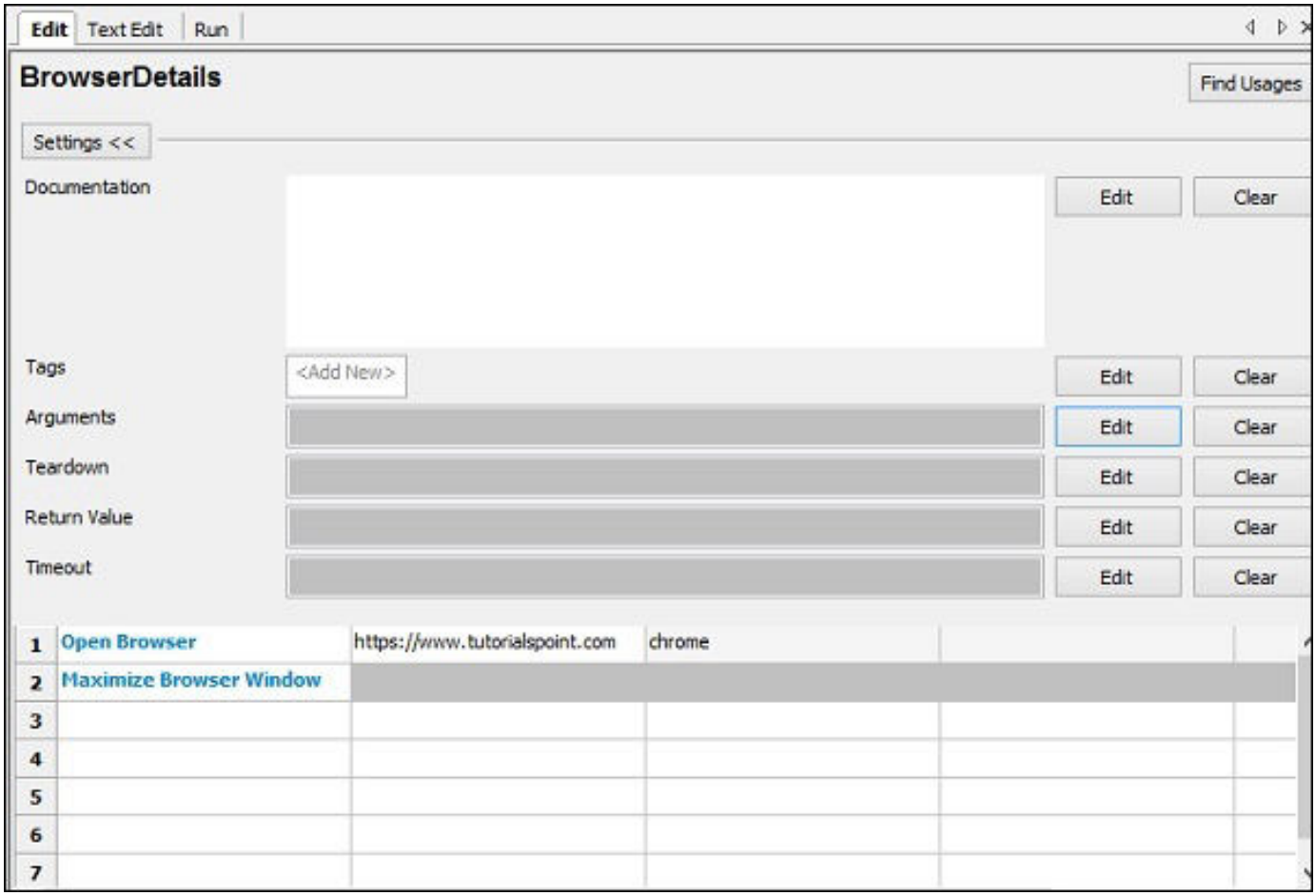
Now, we will see the use-case of arguments in keywords.

Here is the keyword that we created −

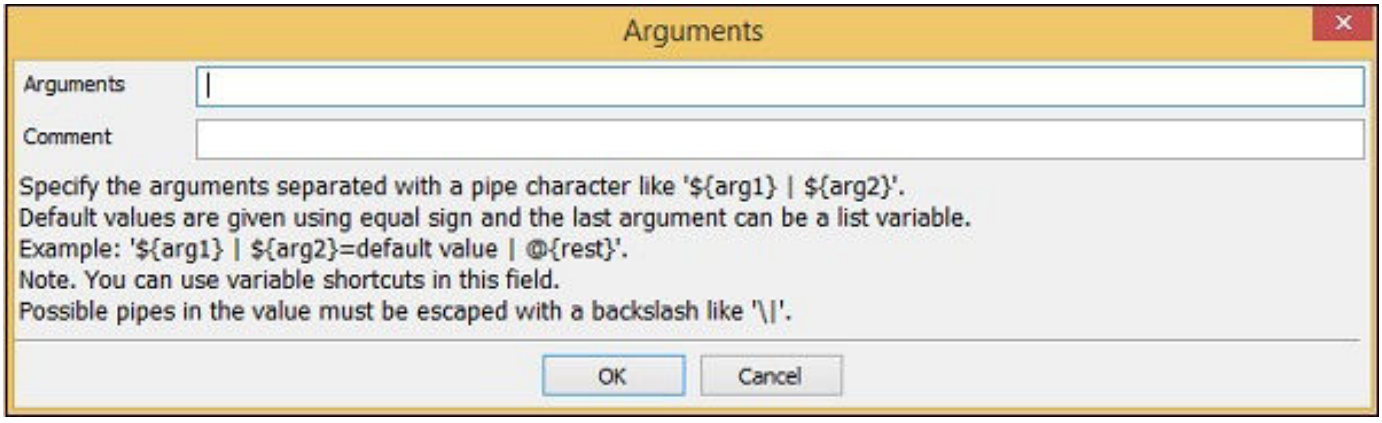


The name of the keyword is *BrowserDetails*. We can use this keyword in other test cases created under the project. The keyword contains the URL of the browser hardcoded. If we want to use the keyword in another test case with a different URL, it will not be possible.

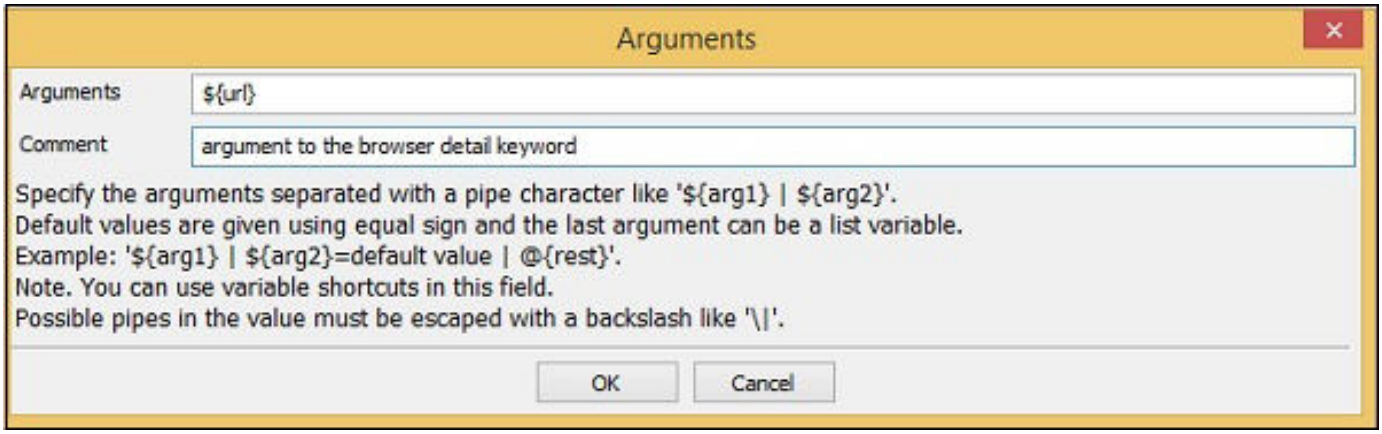
We can use arguments to help us with the hardcoded parameters. We will go back to the keyword created and make use of arguments.



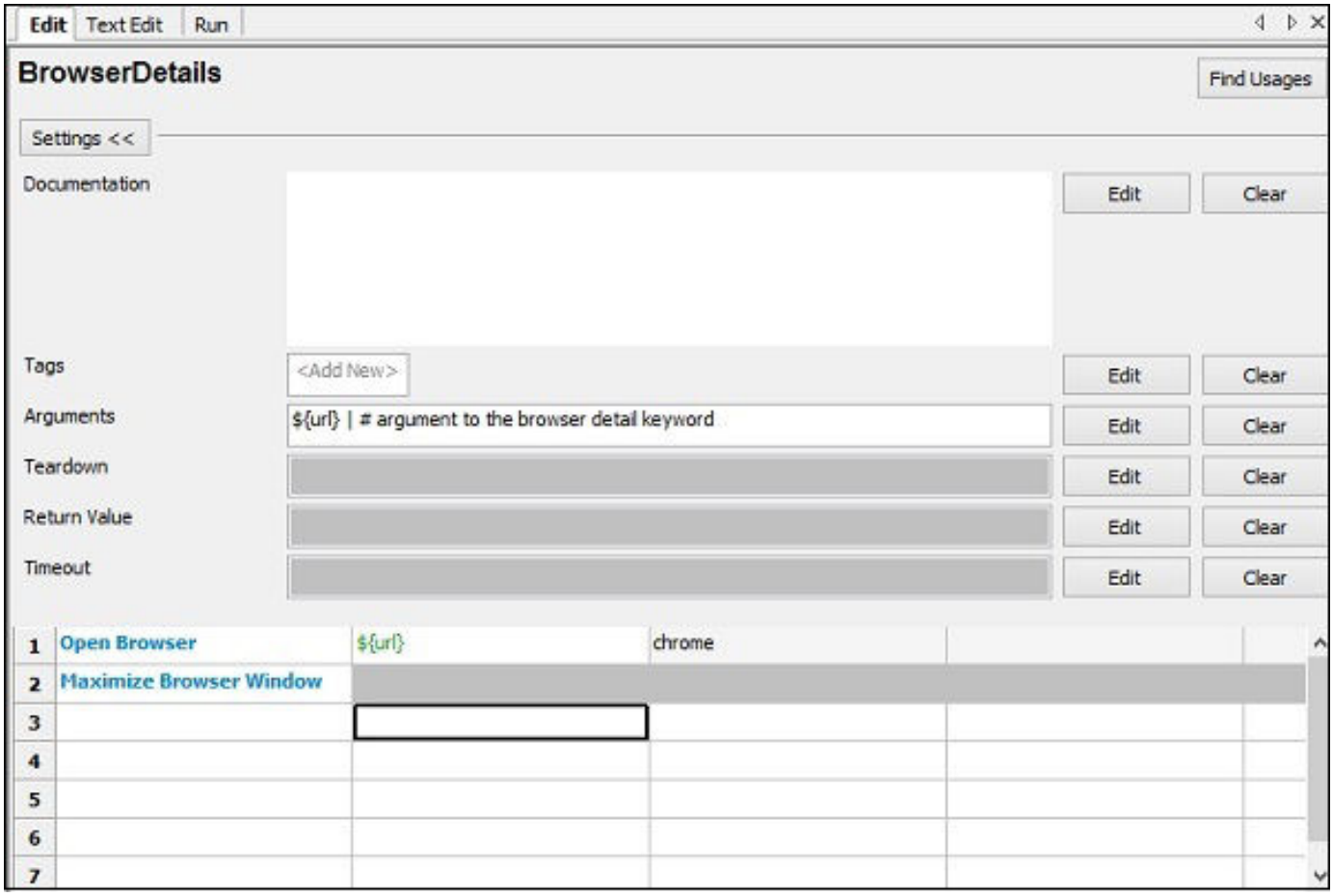
Click on Edit against the Arguments.



Enter the argument to be used with the keyword.



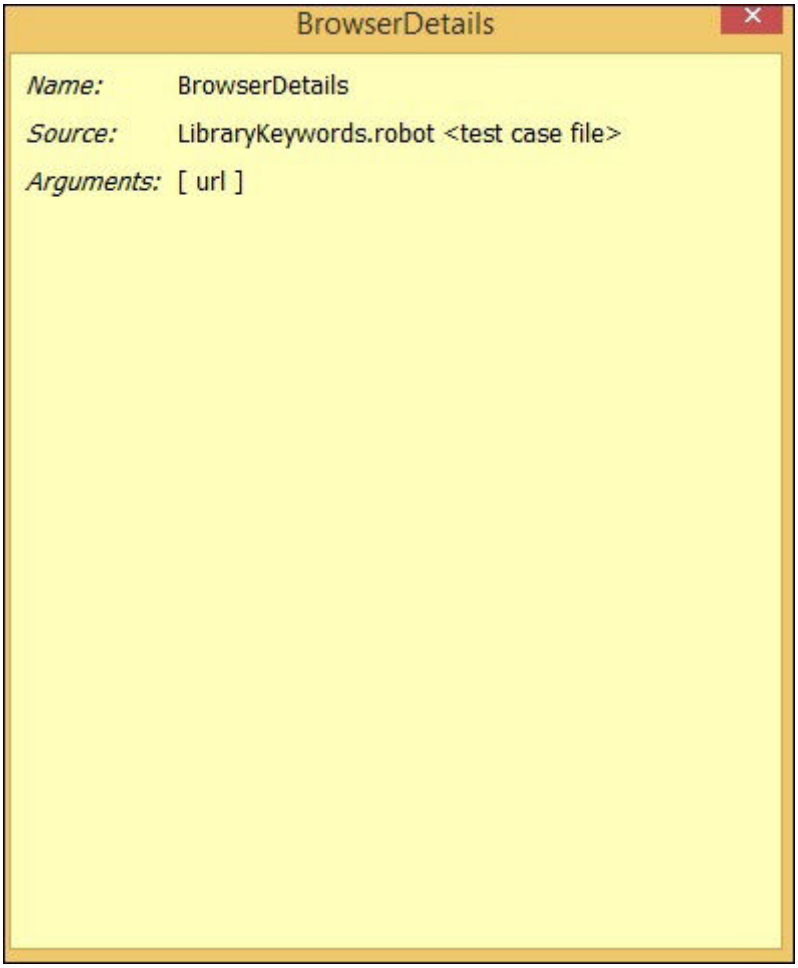
If there is more than 1 argument, you can separate them using pipe (|). We will now use the argument in the Keyword specified as follows −



Go back to your test case. Now, you need to pass the value which is the URL to be used for the test case.

In the test case, when you type the user-defined keyword and press Ctrl + Spacebar, it gives the details of the keyword along with the arguments.

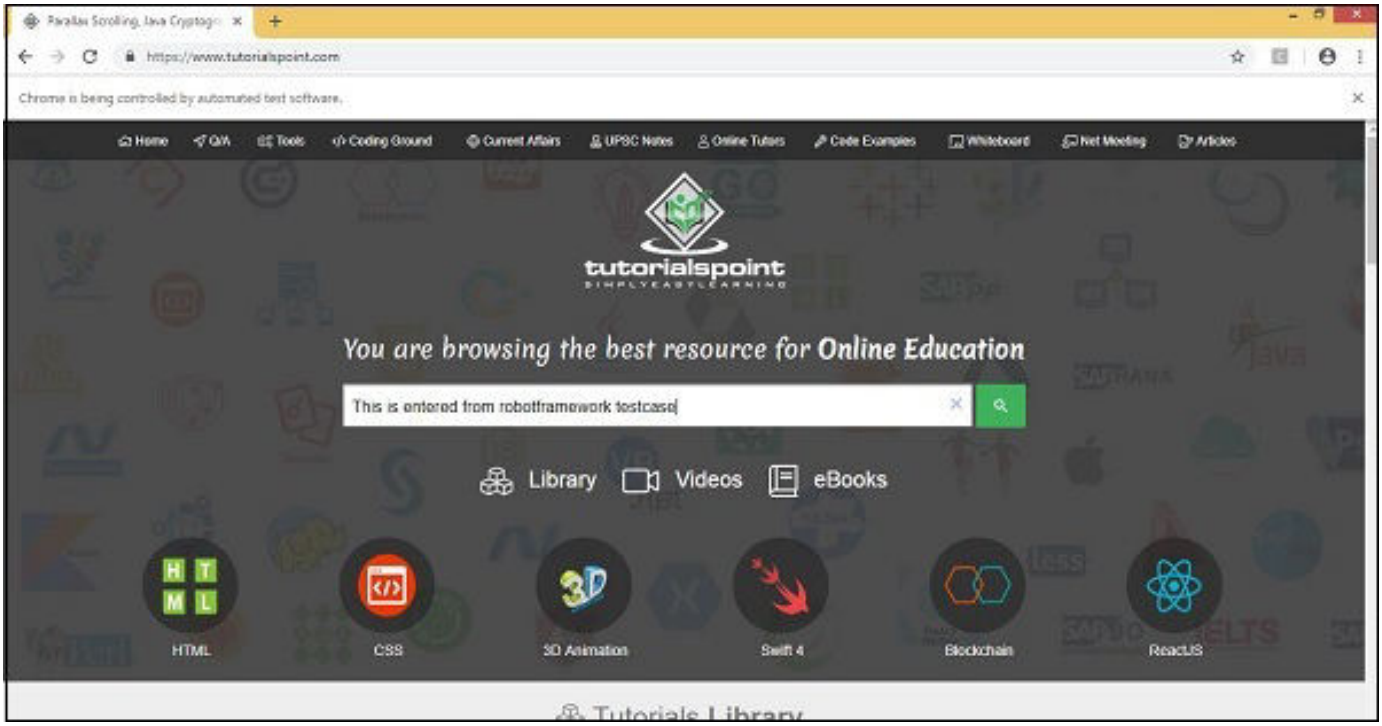
Following are the details for keyword BrowserDetails −



The test case now will have the URL to be passed as argument.

1	BrowserDetails	https://www.tutorialspoint.com	
2	Input Text	name=search	This is entered from robotframework testcase
3			
4			

Let us now run the test case to see the output −



The Keyword and the arguments passed to the user-defined keyword are working fine.

Let us now change the URL; we will use https://www.google.com/

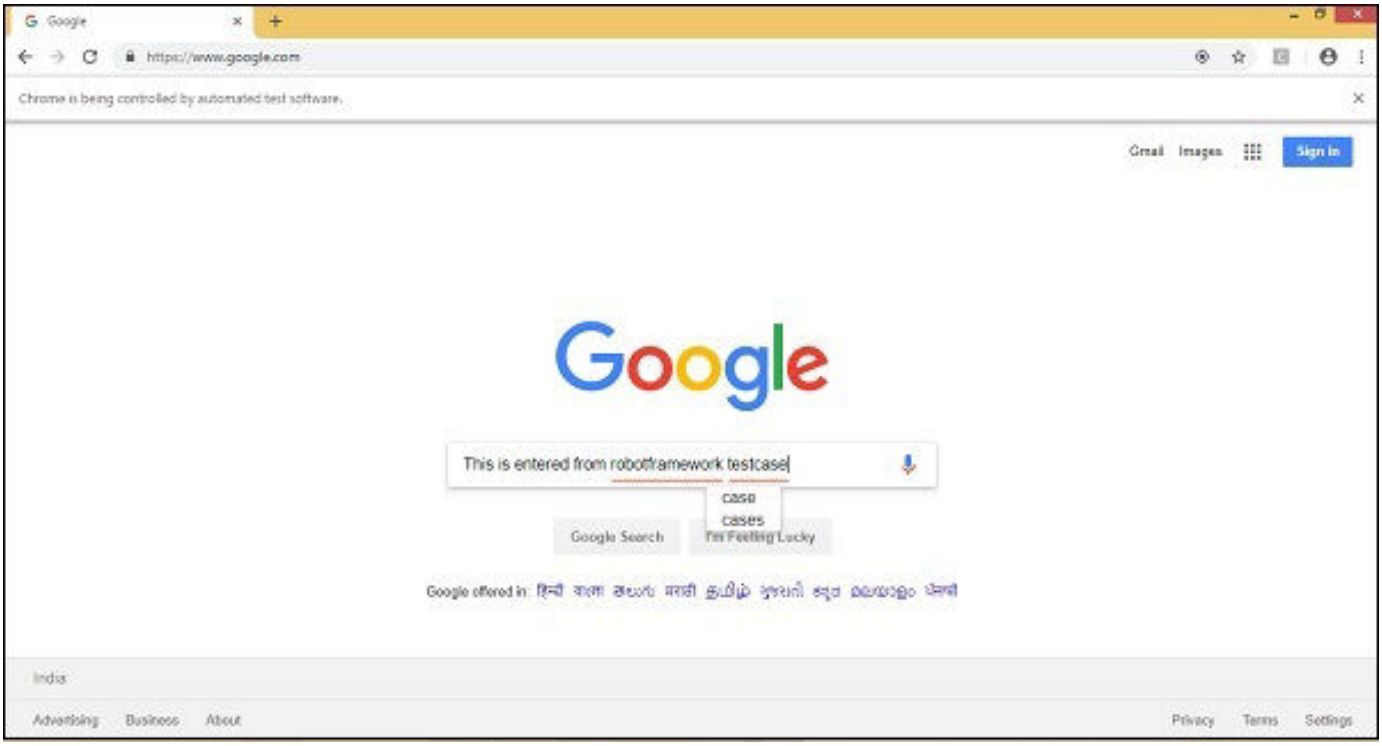
1	BrowserDetails	https://www.google.com	
2	Input Text	id=lst-ib	This is entered from robotframework testcase
3			
4			

The URL for keyword BrowserDetails is changed to https://www.google.com/

We have changed the argument to Input Text to the id available from google site. To get the id or name or class of the input field, you can inspect and check in the browser.

Let us run the above test case and see the output.

Upon successful execution, the above test case generates the following output −



Conclusion

In this chapter, we have seen how to get help for built-in keywords. We have also seen how to create user-defined keywords, which can be a combination of library keywords and built-in keywords.

Robot Framework - Working With Variables

In this chapter, we will discuss how to create and use variables in Robot Framework. Variables are used to hold a value, which can be used in test cases, user-defined keywords, etc.

We are going to discuss following variables available in Robot Framework

- Scalar Variable
- List Variable
- Dictionary Variable

We will understand the working of each of this variable with the help of test cases in Ride.

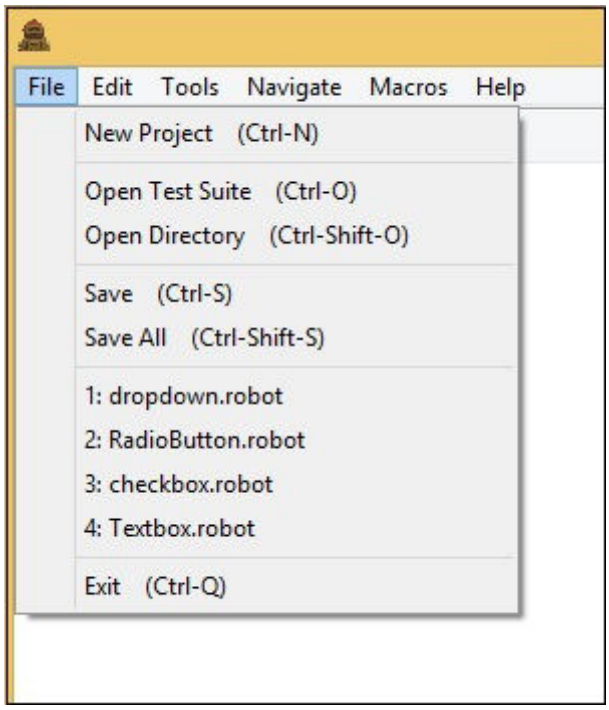
Scalar Variable

Scalar variables will be replaced with the value they are assigned. The syntax for scalar variable is as follows −

```
${variablename}
```

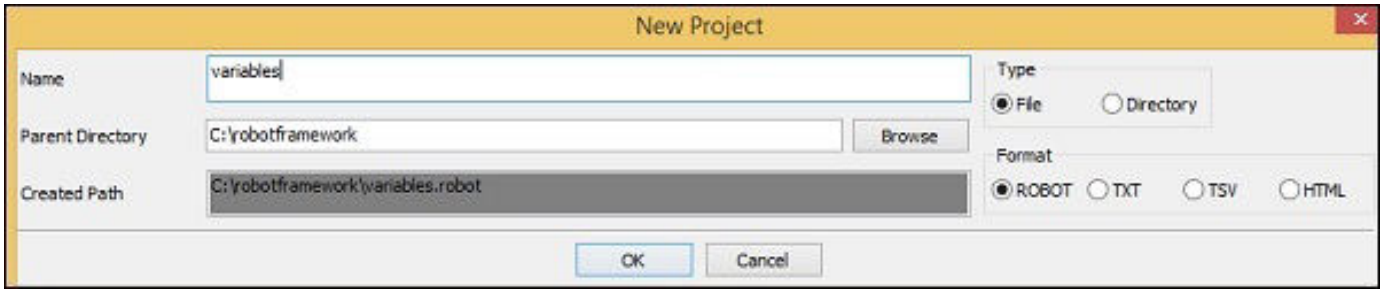
We can use scalar variable to store strings, objects, lists, etc. We will first create a simple test case and make use of scalar variable in it.

Open RIDE using **ride.py** in the command line and create a new project.



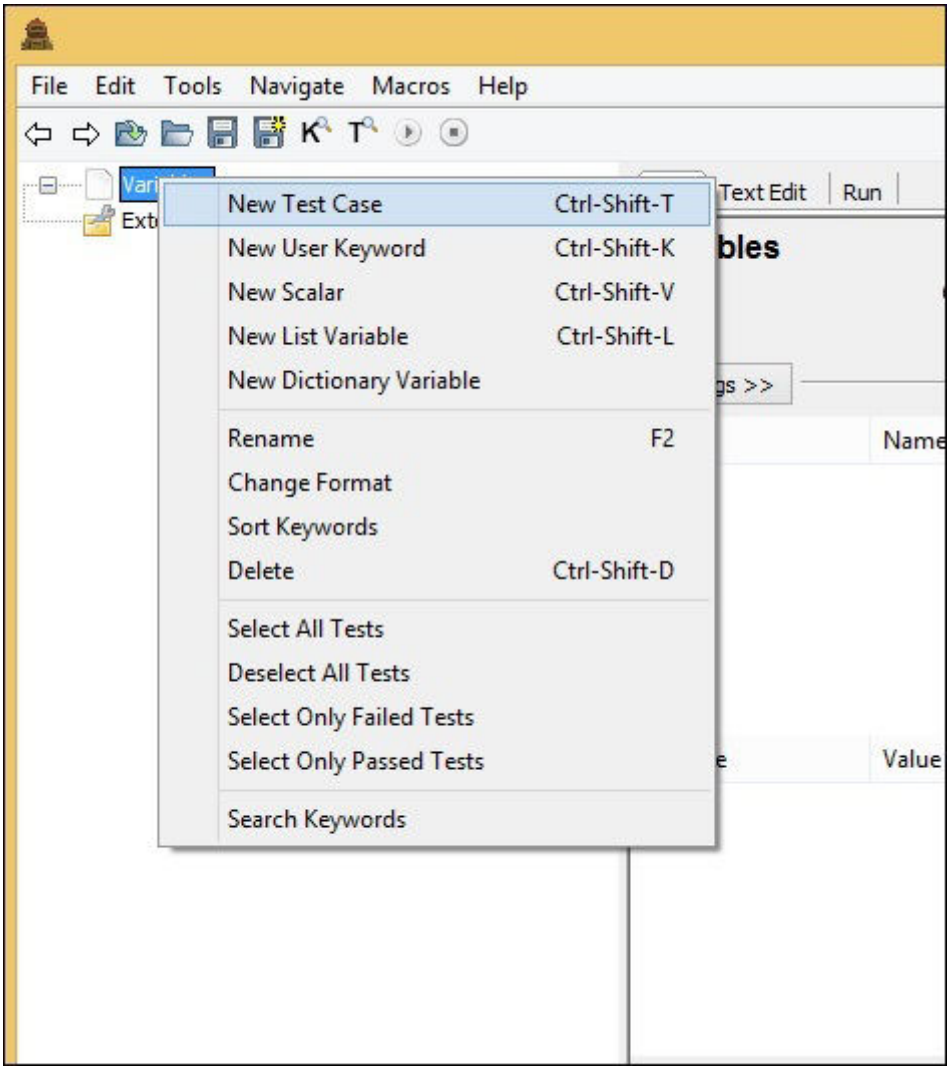
Click **New Project**.

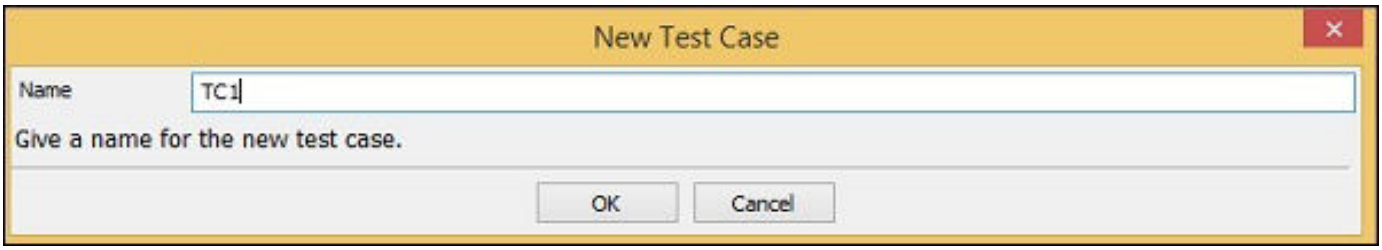
Now, give a name to your project.



The name given is *variables*. Click OK to save the project.

Right-click on the name of the project created and click on *New Test Case* −

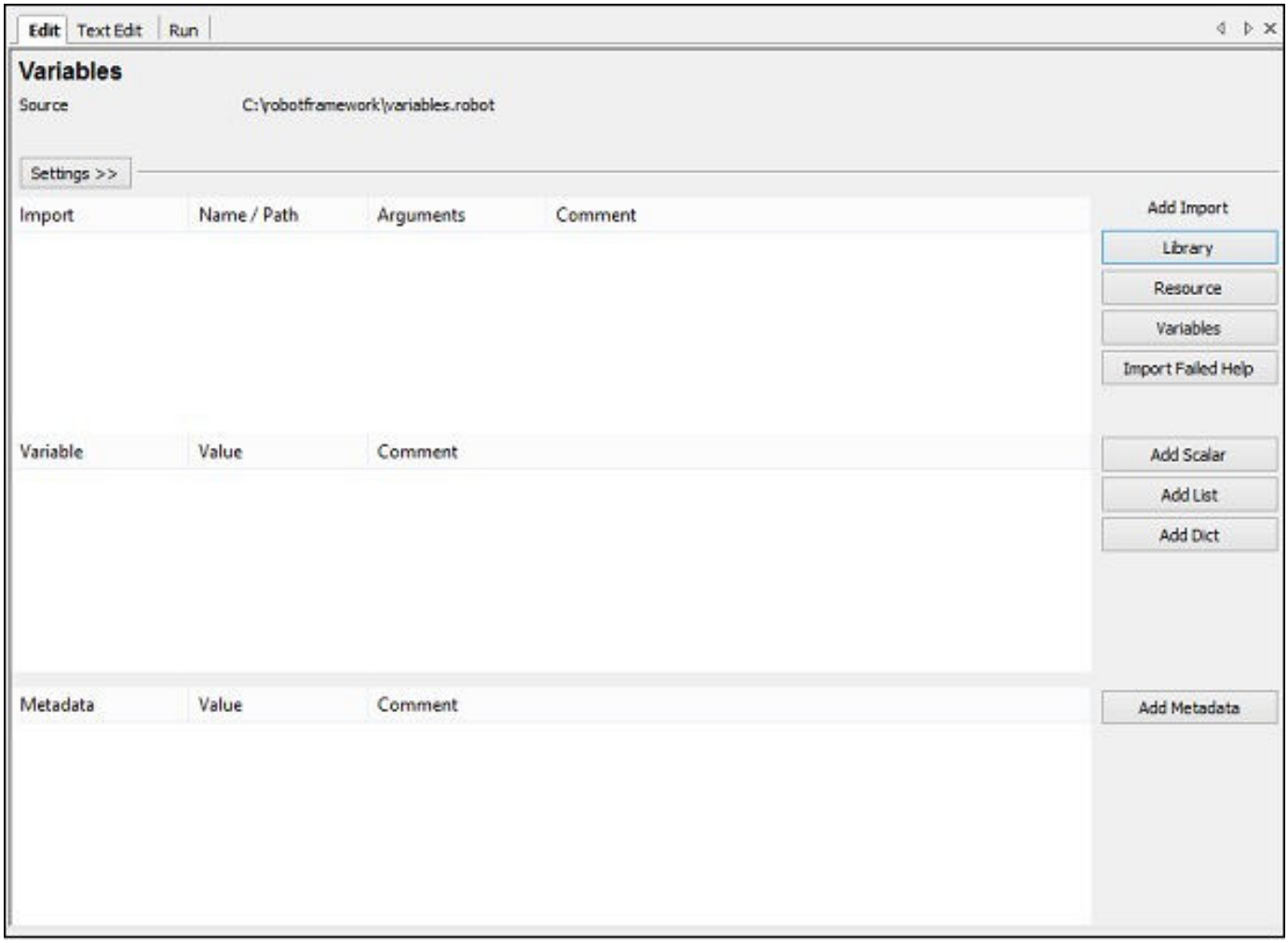




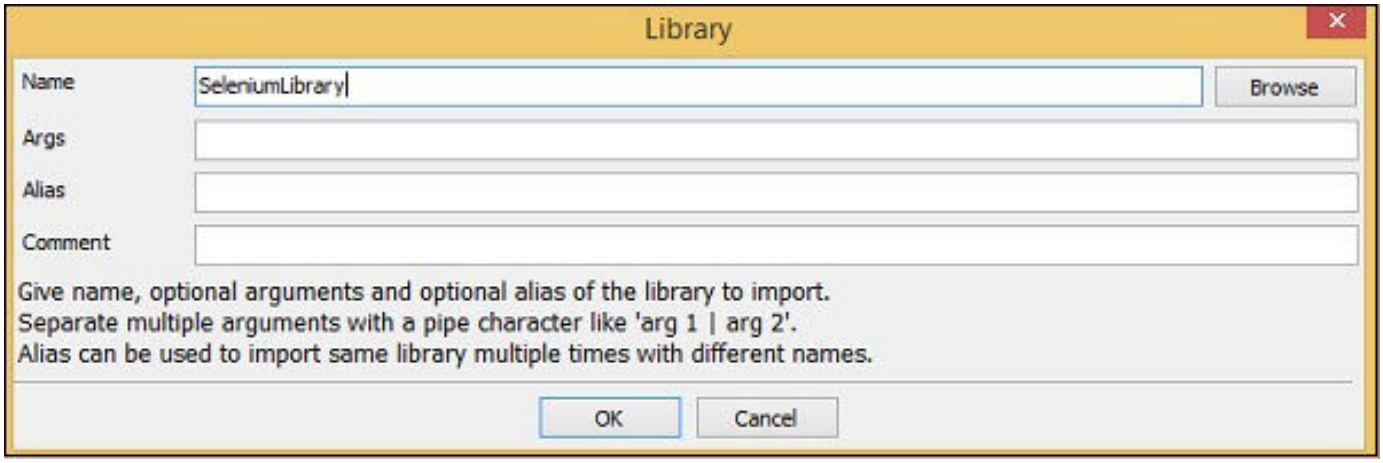
Give a name to the test case and click OK.

We are done with the project setup and now will write test cases for the scalar variables to be used in our test case. Since we need Selenium library, we need to import the same in our project.

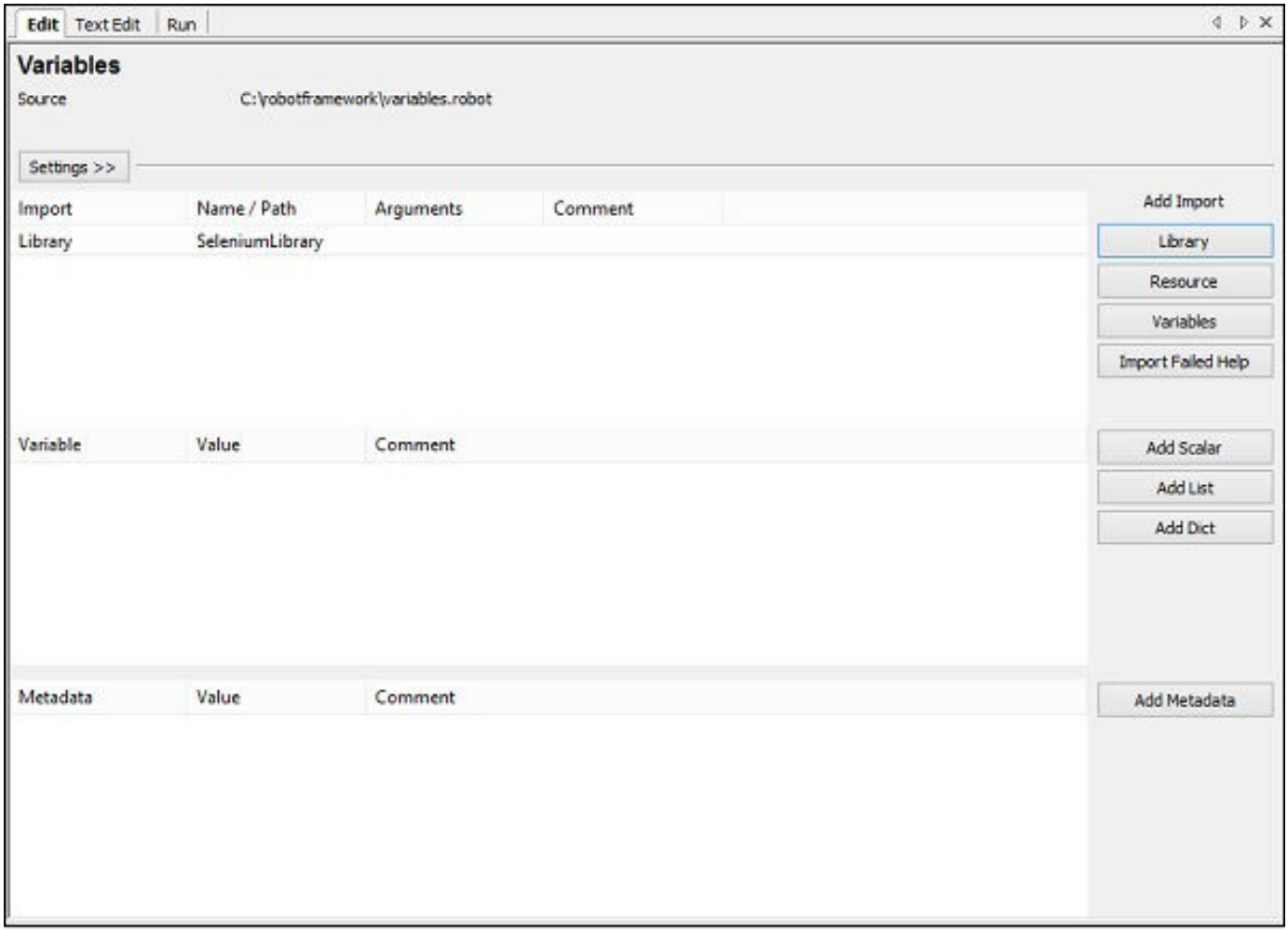
Click on your project on the left side and use Library from Add Import −



Upon clicking Library, a screen will appear where you need to enter the library name −

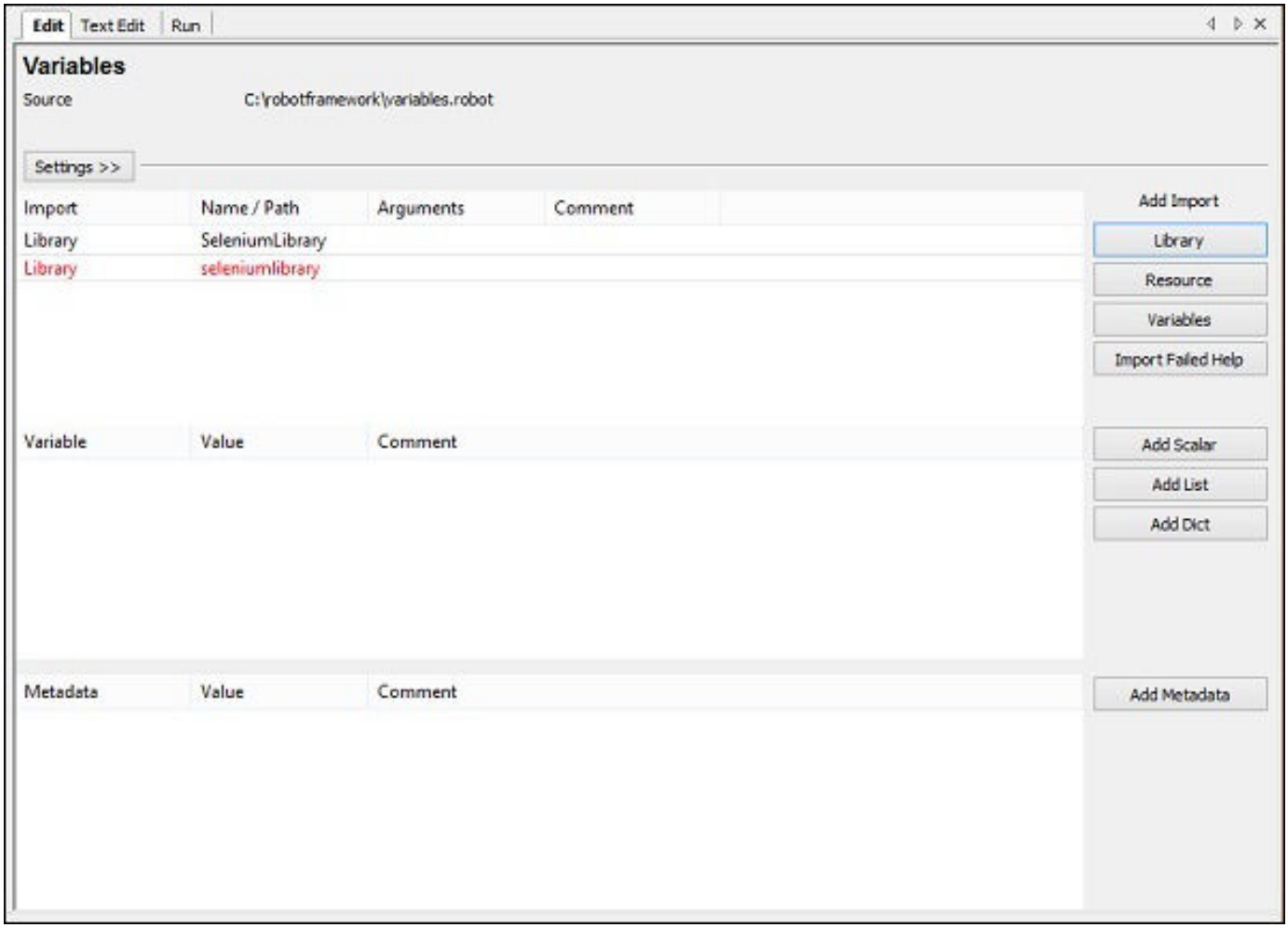


Click OK and the library will get displayed in the settings.



The name given has to match with the name of the folder installed in site-packages.

If the name does not match, the library name will be shown in red −

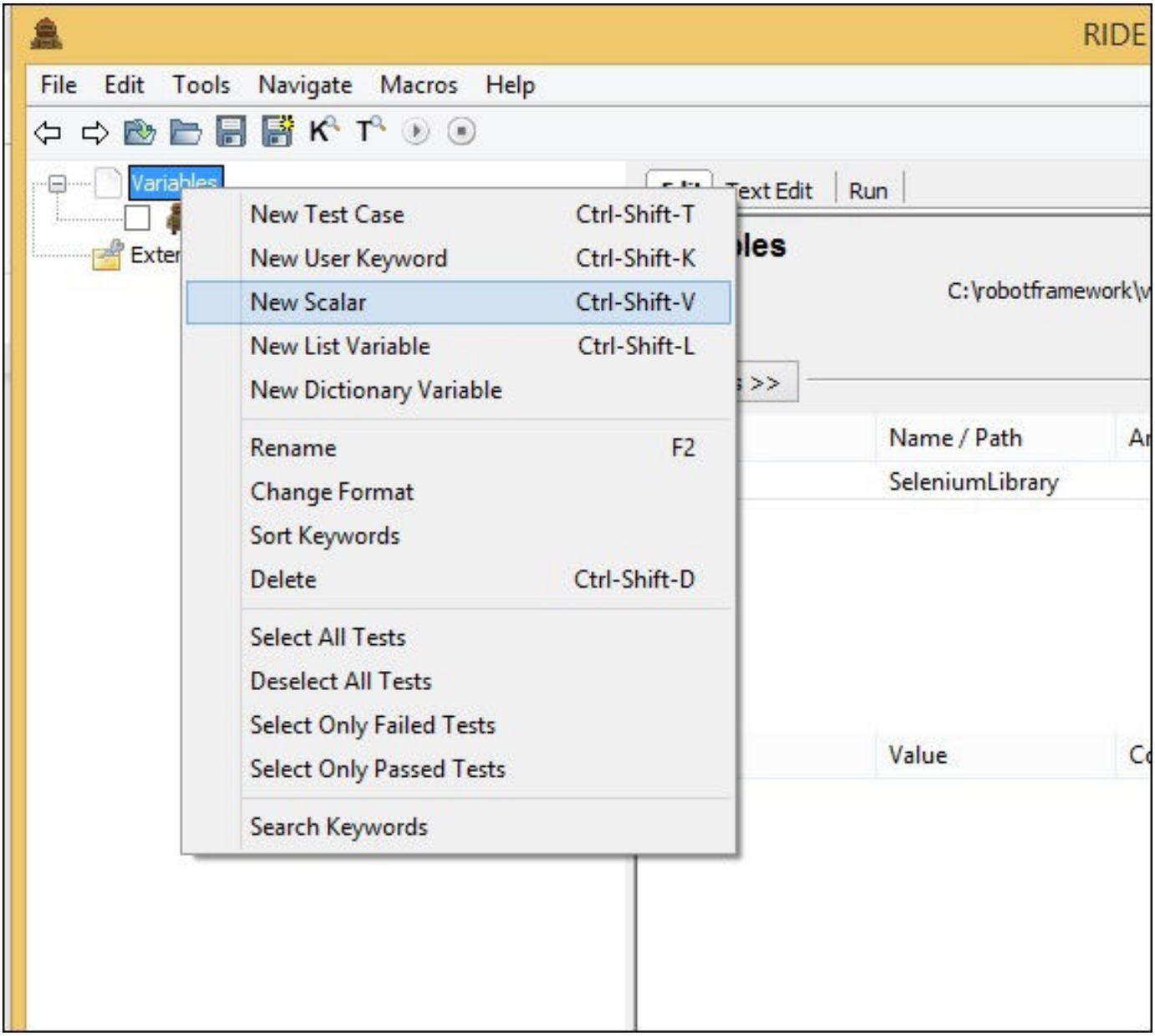


Test Case for Scalar Variable

In the above test cases we hardcoded the values like the URL, email, password, which we are giving to the test case. The values used can be stored in a variable and instead of hardcoding, we can use the variable in those places.

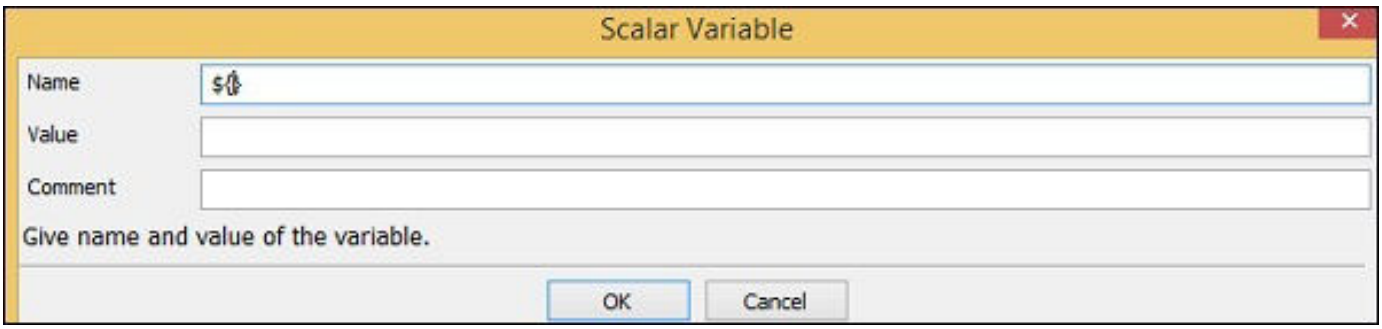
1	Open Browser	http://localhost/robotframework/lo	chrome		
2	Input Text	id:email	admin@gmail.com		
3	Input Password	id:passwd	admin		
4	Click Button	id:btnsubmit			
5					

To create scalar variable, right-click on your project and click on *New Scalar* as shown below −

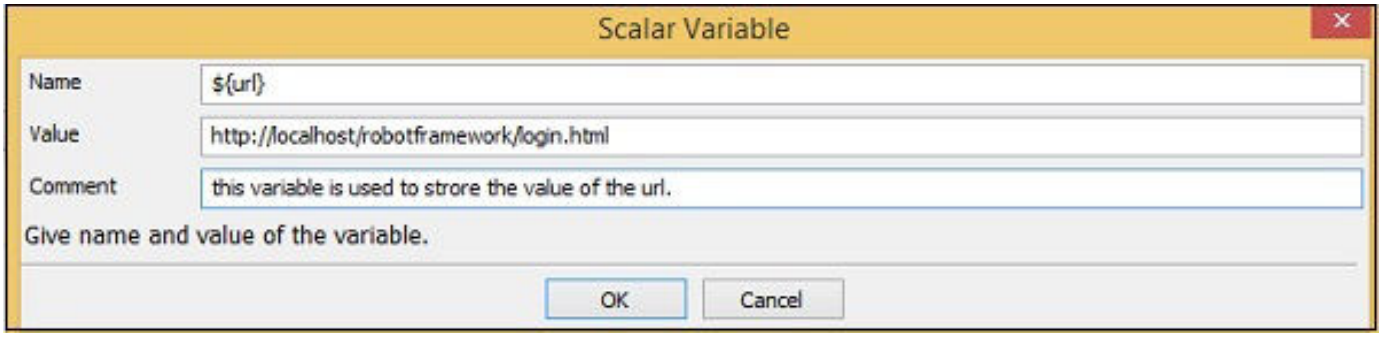


Clicking on New Scalar will open the following screen to create the variable and the value we need to replace with when the variable in used inside test cases.

We get ${} for the Name field.

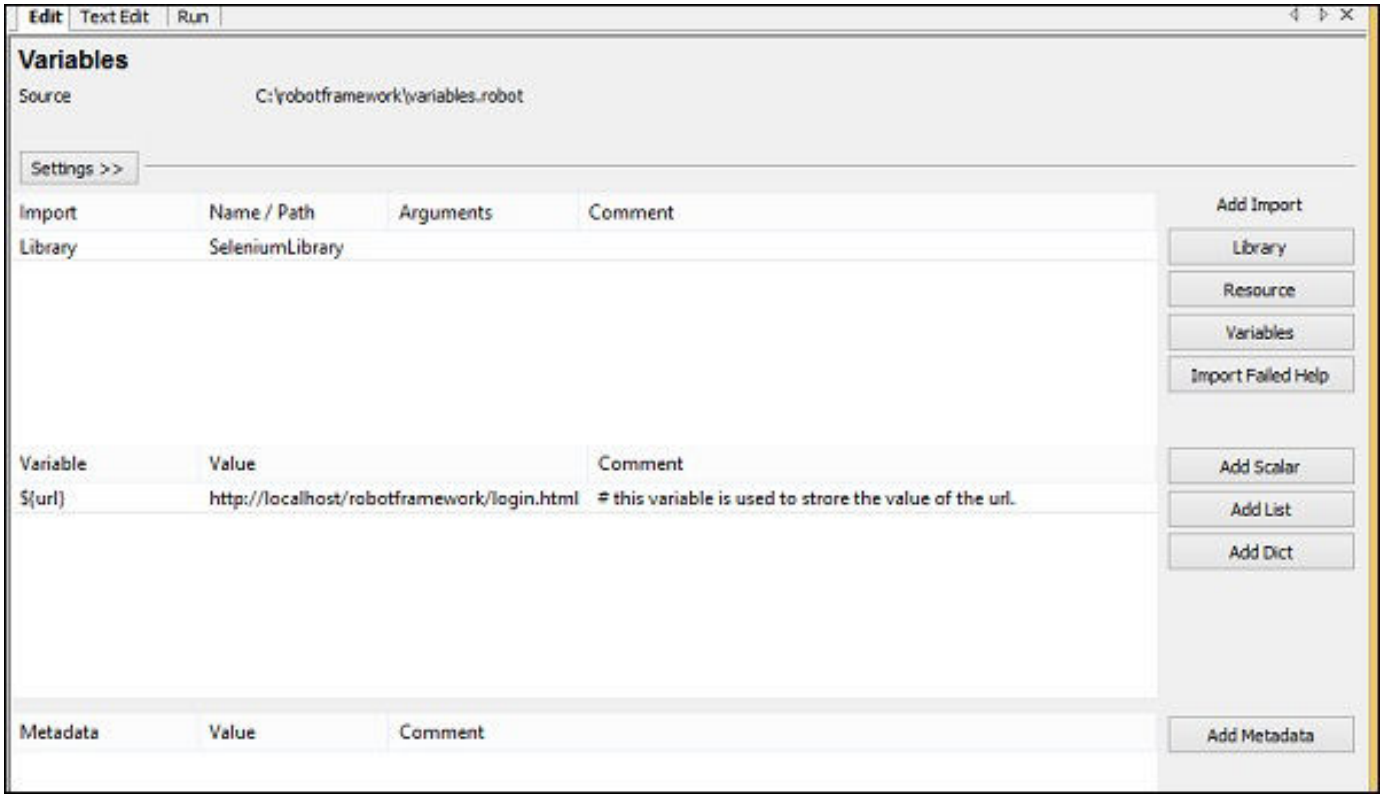


Here we need to enter the name of the variable inside the curly braces as shown in the screen below −

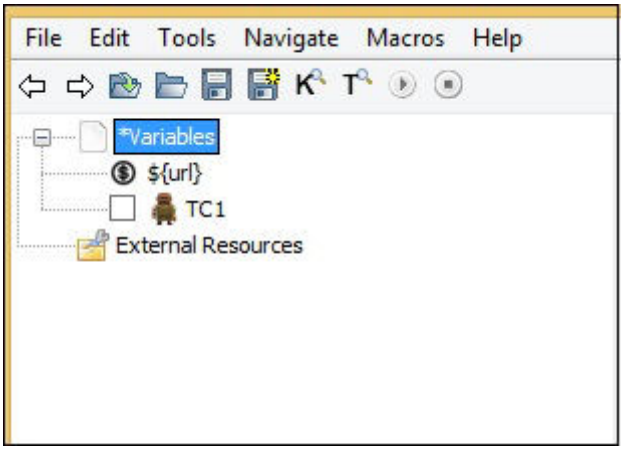


The name of the variable is ${url}. The value is − **http://localhost/robotframework/login.html**.

We added the comment as shown above. Click OK to save the scalar variable. The details of the variable are added as shown below −



The variable name is shown under the project created as follows −



Let us now use the scalar variable created inside our test case.

Test case with URL hardcoded

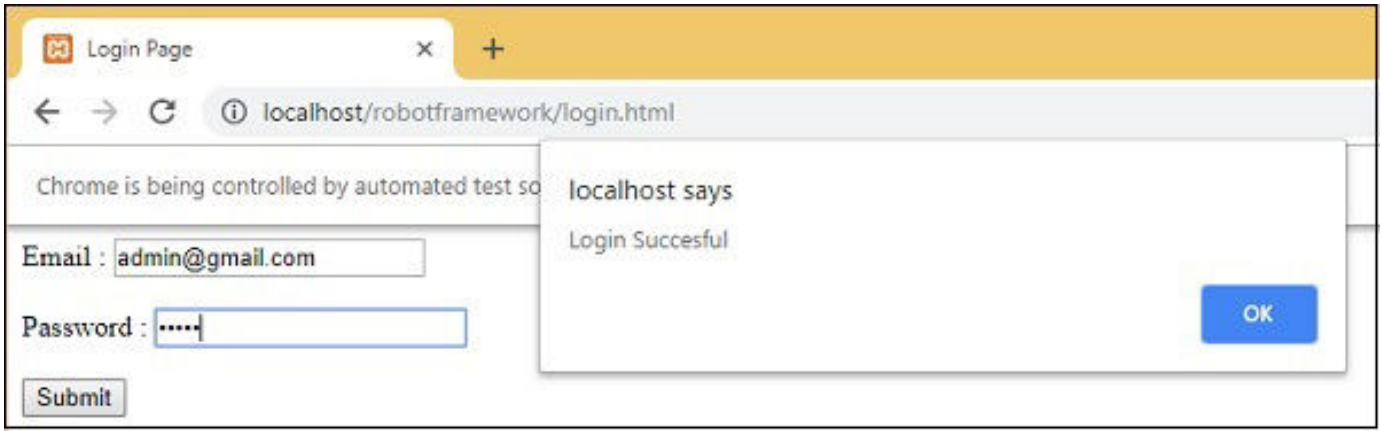
1	Open Browser	http://localhost/robotframework/lo	chrome		
2	Input Text	id:email	admin@gmail.com		
3	Input Password	id:passwd	admin		
4	Click Button	id:btnsubmit			
5					

In the above test case, we have to replace the URL with the variable we just created above.

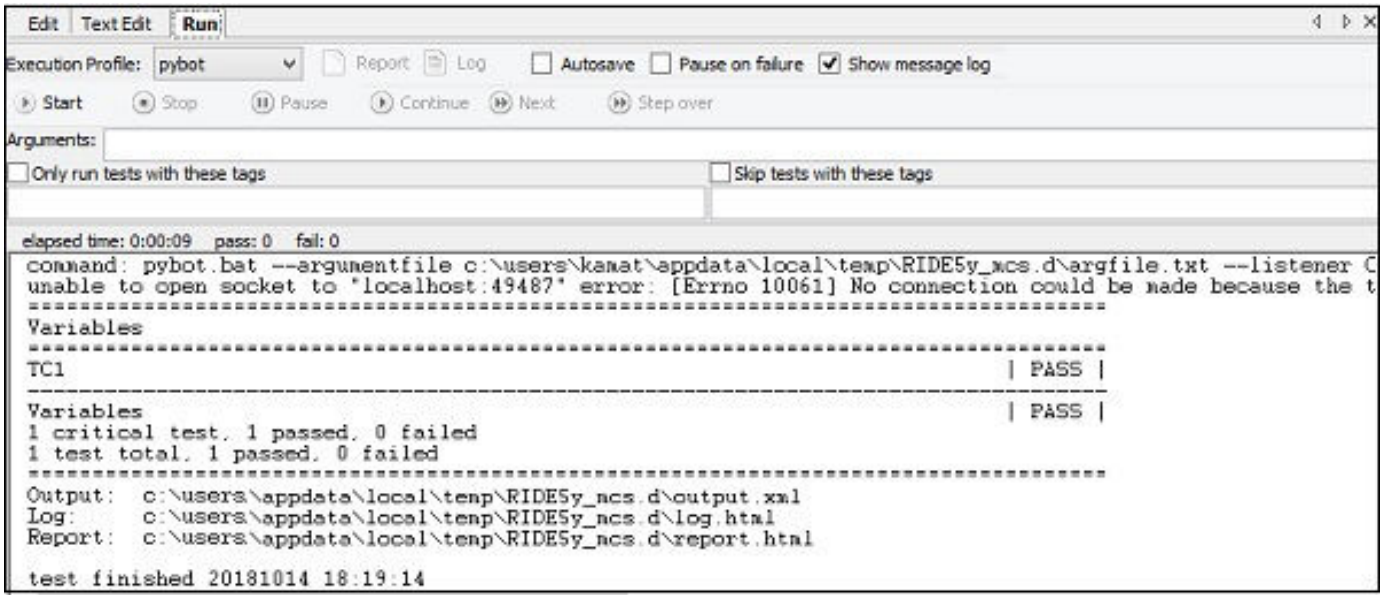
Test Case with Scalar Variable for URL

1	Open Browser	${url}	chrome		
2	Input Text	id:email	admin@gmail.com		
3	Input Password	id:passwd	admin		
4	Click Button	id:btnsubmit			
5					

Now, we will run the test case to see if it is taking the URL from the variable. Below is the output that we get when we run it. The URL **http://localhost/robotframework/login.html** is picked up from the scalar variable we created.



Execution Details



The advantage of using variables is that you can change the value for that variable and it will be reflected in all test cases. You can use the variables in many test cases that you create under

that project. Hardcoding of values can be a serious problem when you want to change something, you will have to go to individual test case and change the values for it. Having variables in one place gives us the flexibility to test the way we want with different values to the variables.

Now, we will look into the next type of variable called the List variable.

List Variable

List variable will have an array of values. To get the value, the list item is passed as the argument to the list variable.

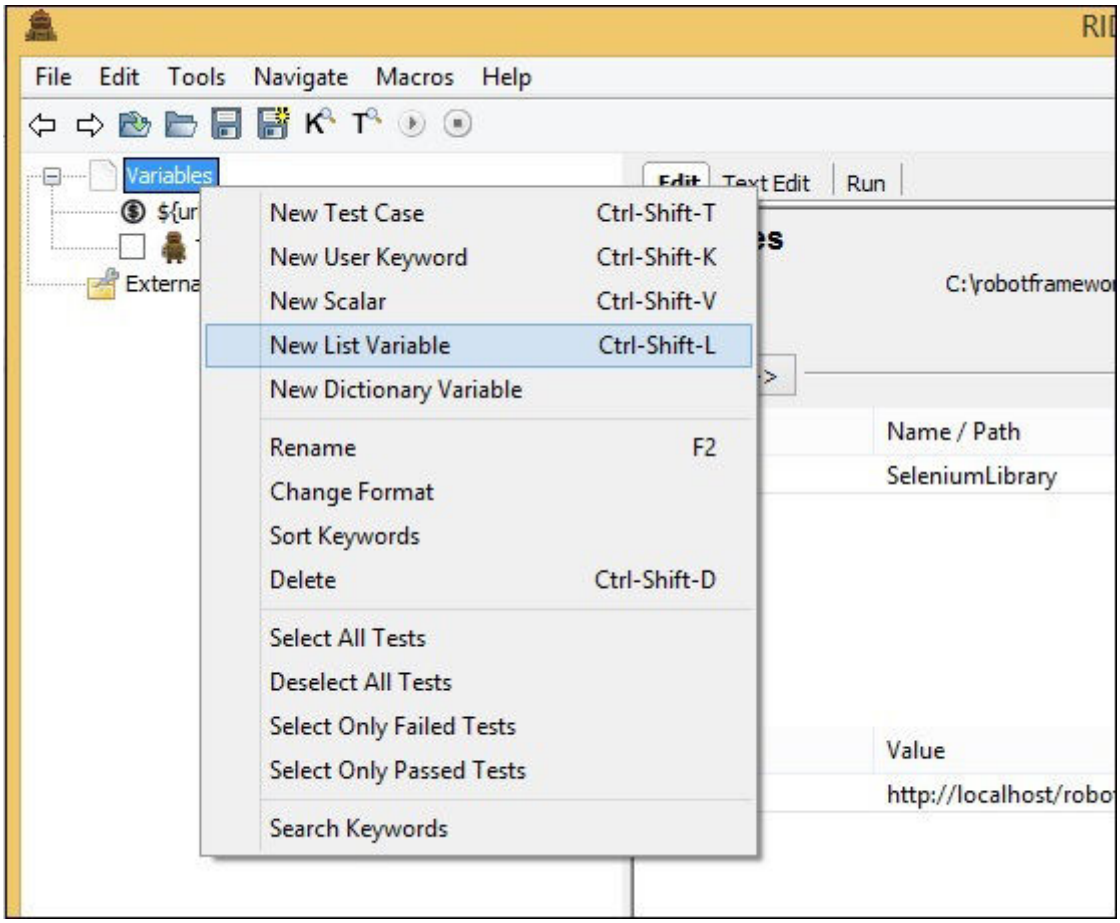
Syntax

```
@{variablename}
```

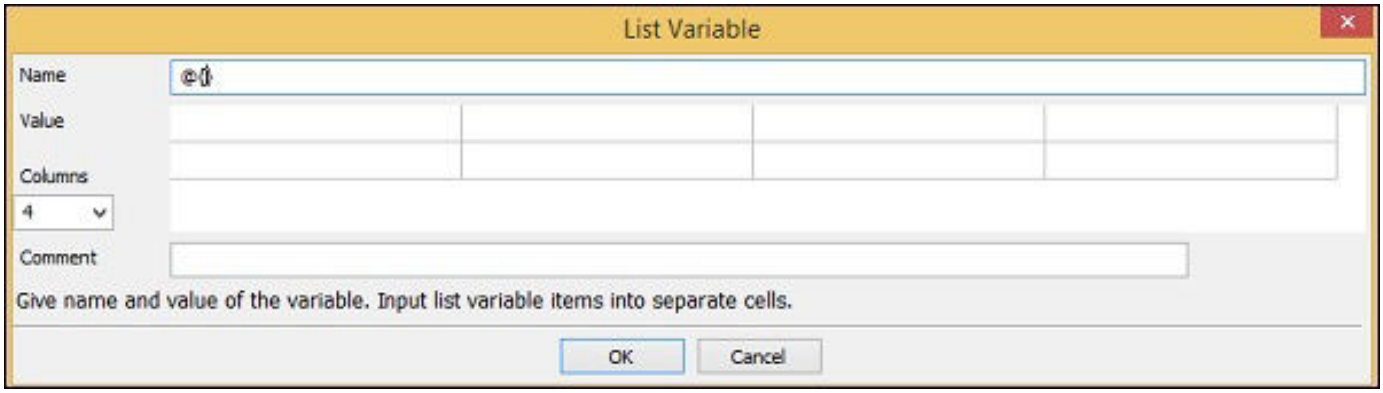
Suppose we have values A, B. To refer the values, we need to pass the list item as follows −

```
@{variablename}[0] // A
@{variablename}[1] // B
```

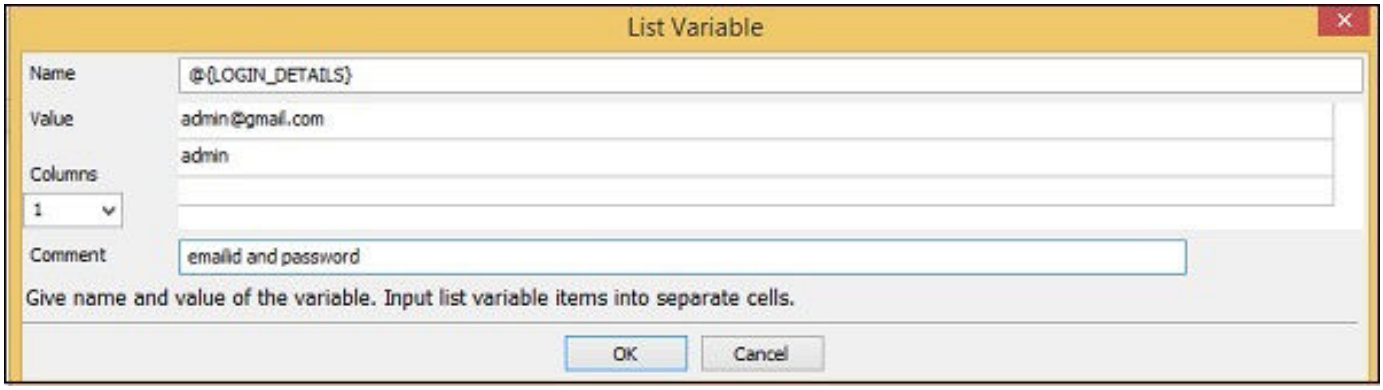
To add list variable, right-click on the project and click **New List Variable**.



Upon clicking *New List Variable*, a screen appears where we can enter the values −

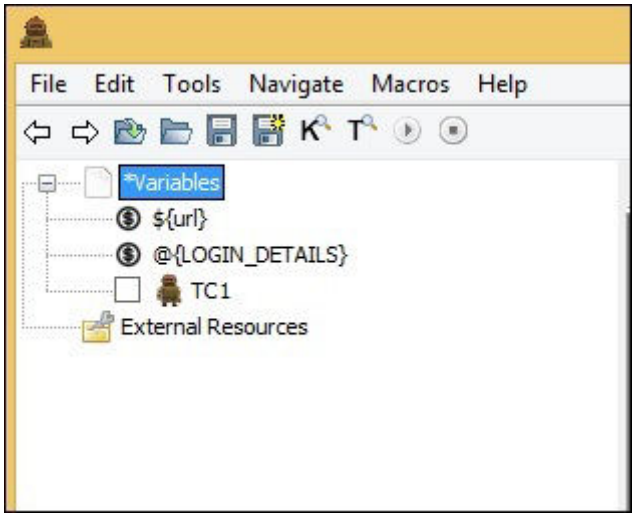


The Name is given as @{} followed by Value. It also has 4 Columns selected. Right now, we will use just Column 1 and create the list variable, which will have values, email id and password as follows −

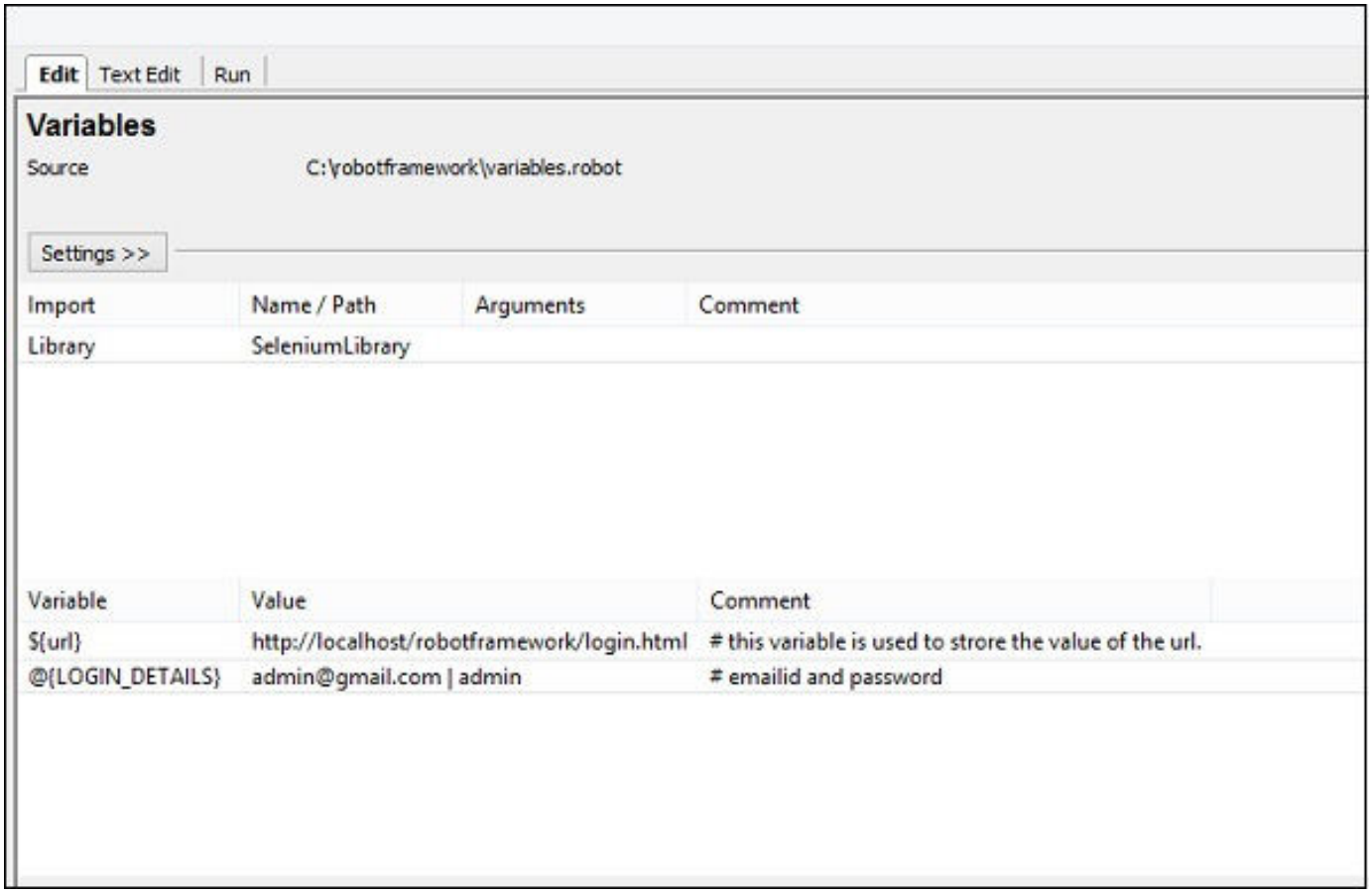


The name of the list variable is **@{LOGIN_DETAILS}** and the values given are **admin@gmail.com** and **admin**, which has email id and password for the login page.

Click OK to save the list variable. The variable is listed below the project as shown here −



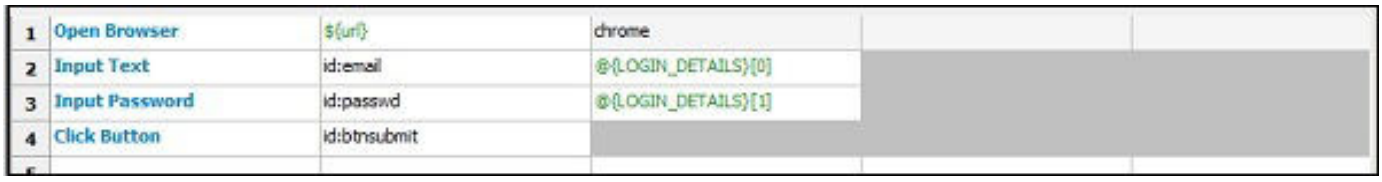
The details of variables used are listed in the settings tab −



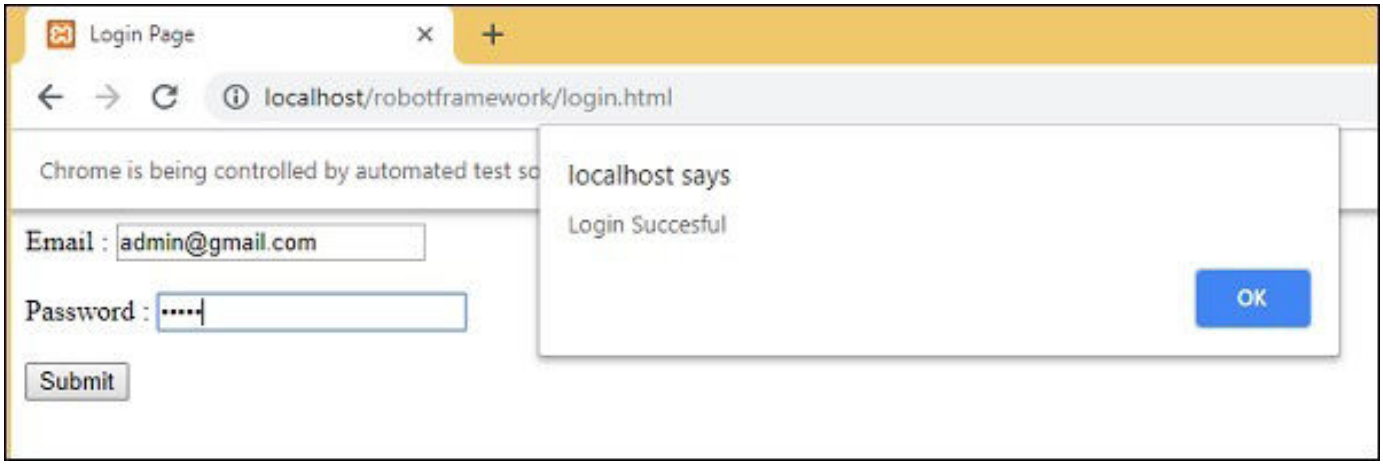
Now, we will add the list variable inside the test cases as shown below.

Here, we have hardcoded values for the Input Text and Password. Now, we will change it to use the list variable.

Using List Variable

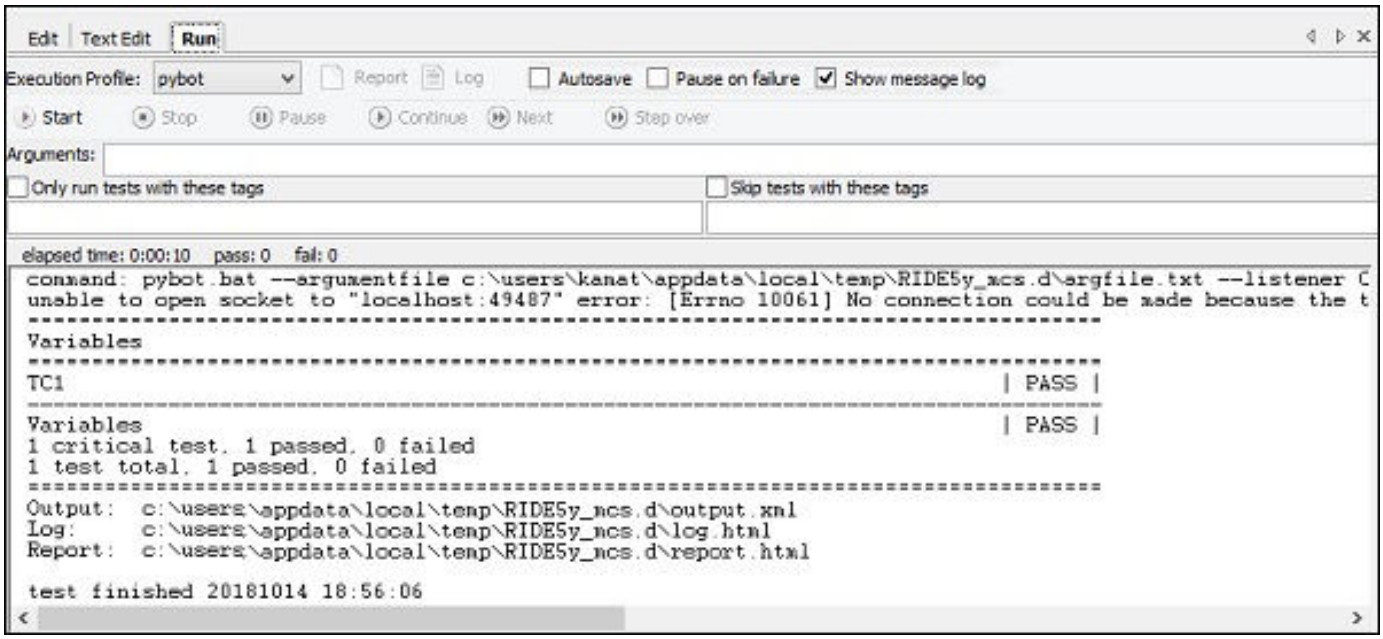


Now, we will execute the test case to see if it is taking the values from the list variable −



It has taken the email id and password from the list variable as shown above in the test screen.

The following screenshot shows the execution details for the same −



In our next section, we will learn about the Dictionary Variable.

Dictionary Variable

Dictionary Variable is similar to list variable wherein we pass the index as an argument; however, in case of dictionary variable, we can store the details – key value form. It becomes easier to refer when used in the test case instead of using the index as 0, 1, etc.

Syntax

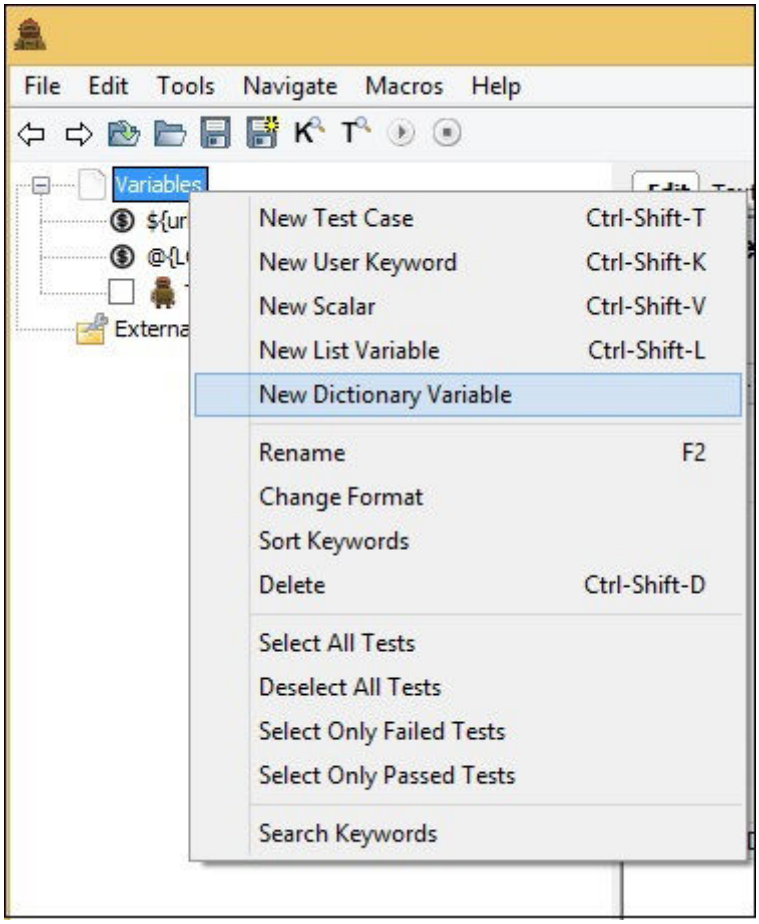
```
&{Variablename}
```

Suppose we are storing the values as key1=A, key2=B. It will be referred in the test case as −

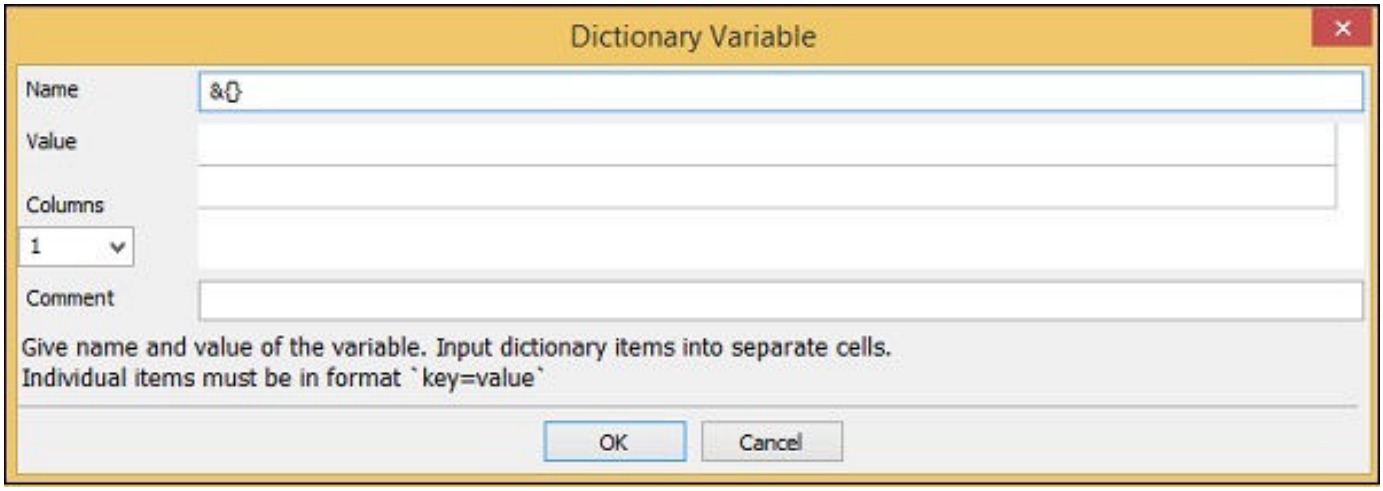
```
&{Variablename}[key1] // A
&{Variablename}[key2] // B
```

Let us create dictionary variable in Ride.

Right-click on Project and click on *New Dictionary Variable*.

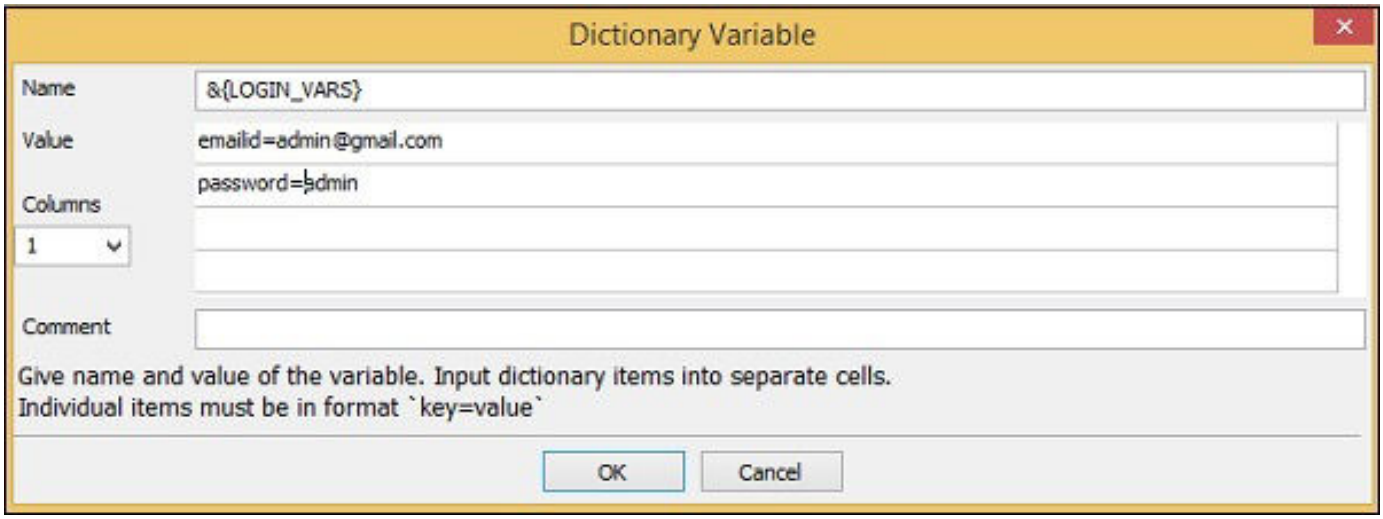


Upon clicking **New Dictionary Variable**, a screen will appear as shown below −

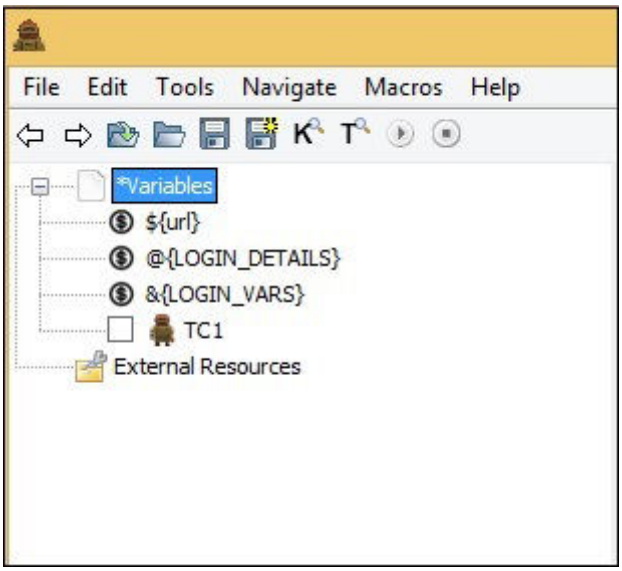


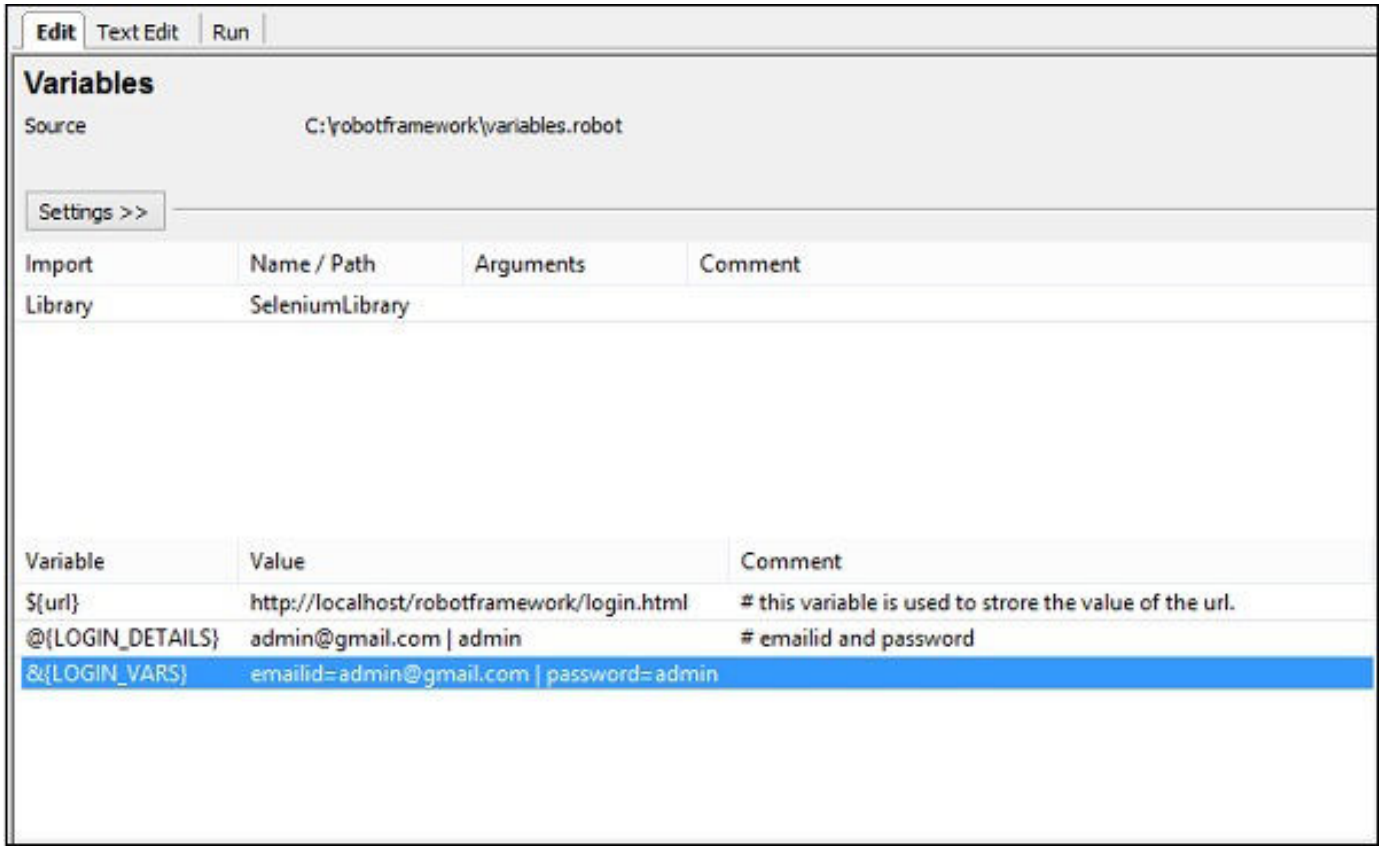
The Name by default in the screen is &{} and it has Value and Columns option.

We will enter the Name and the Values to be used in the test case.



Click OK to save the variable. The variable will be listed under the project and also in the settings as follows −





We will change the test case to take the dictionary values.

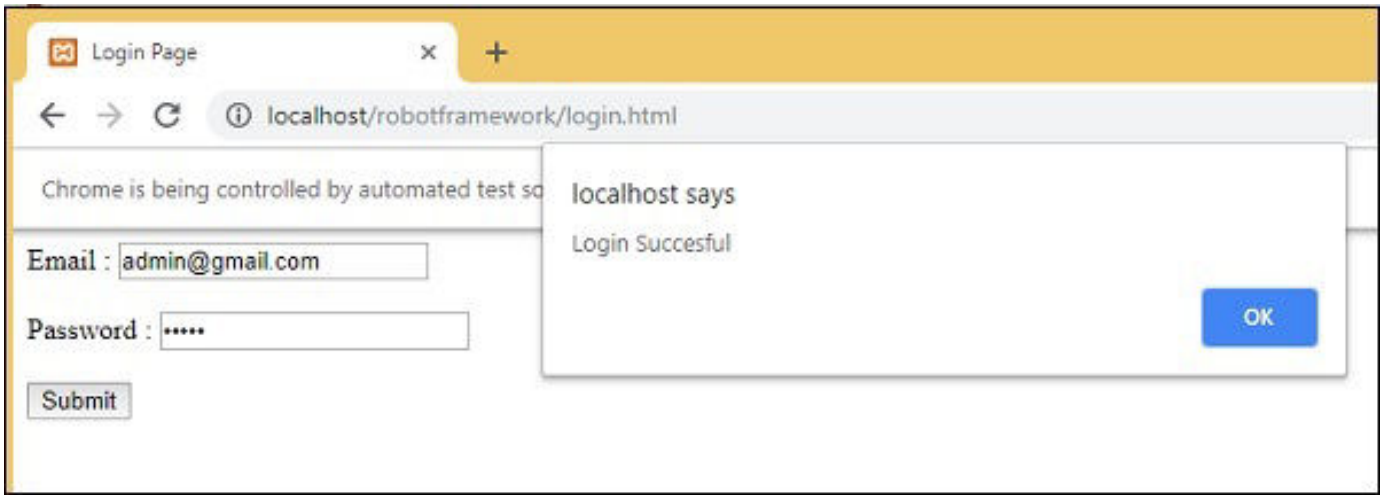
1	Open Browser	${url}	chrome		
2	Input Text	id:email	@{LOGIN_DETAILS}[0]		
3	Input Password	id:passwd	@{LOGIN_DETAILS}[1]		
4	Click Button	id:btnsubmit			
5					

We will change to dictionary variable as shown below.

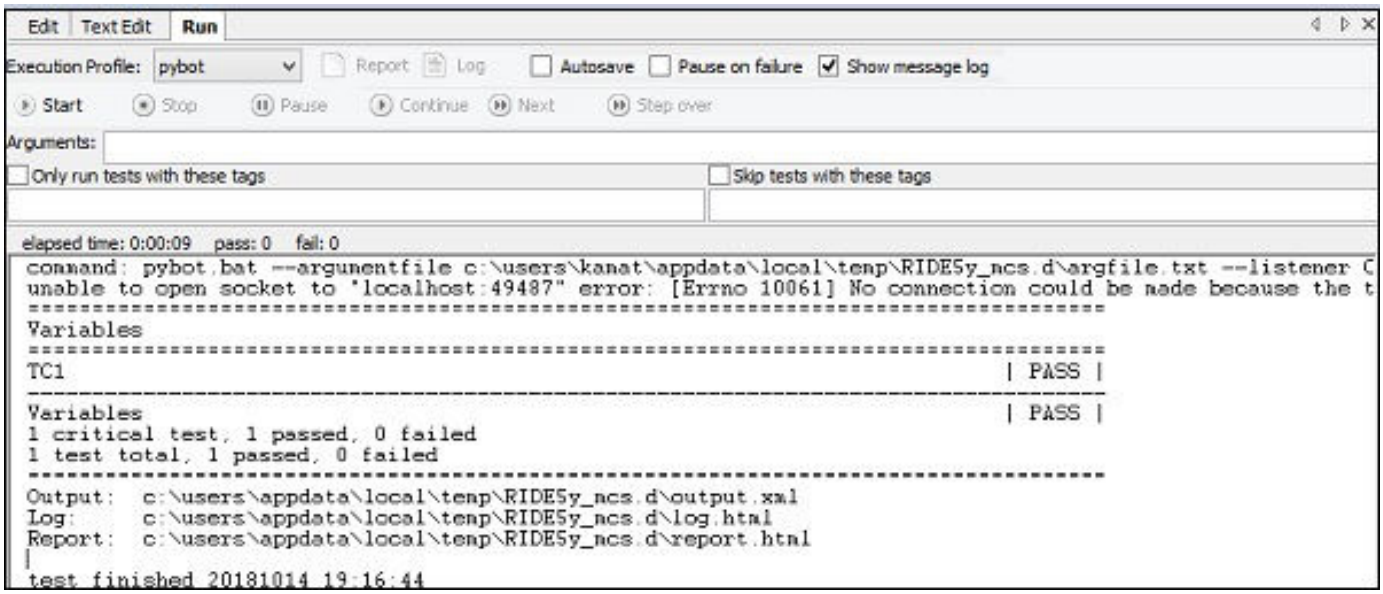
Using Dictionary Variable

1	Open Browser	${url}	chrome		
2	Input Text	id:email	&{LOGIN_VARS}[emailid]		
3	Input Password	id:passwd	&{LOGIN_VARS}[password]		
4	Click Button	id:btnsubmit			

Upon clicking run, we get the following −



The execution details are as follows −

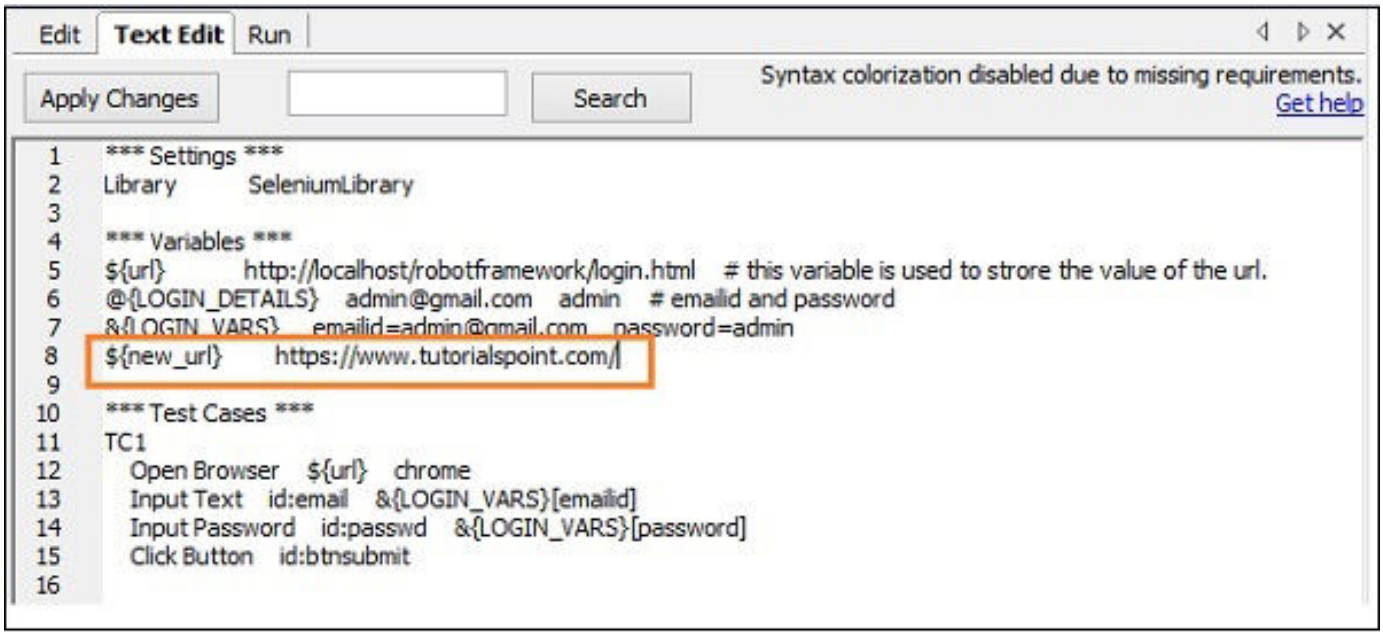


We have seen the Edit and Run Tab so far. In case of TextEdit, we have the details of the test case written. We can also add variables required in TextEdit.

Test case

1	Open Browser	${url}	chrome
2	Input Text	id:email	&{LOGIN_VARS}[emailid]
3	Input Password	id:passwd	&{LOGIN_VARS}[password]
4	Click Button	id:btnsubmit	

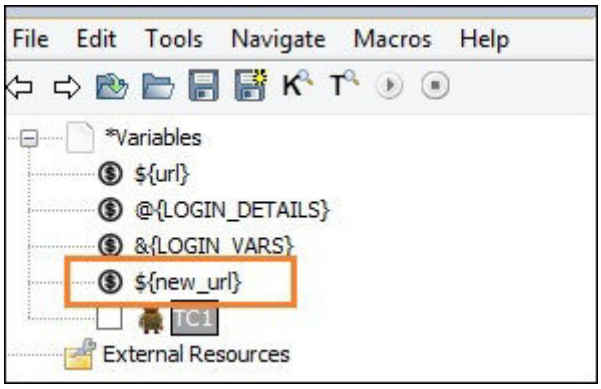
We have used scalar variable and dictionary variable in the above test case. Here is the code so far in TextEdit; this is based on the test case written −



The variables used are highlighted in Red. We can also create variables we want directly in TextEdit as shown below −

We have added a scalar variable called **${new_url}** and the value given is https://www.tutorialspoint.com/ .

Click **Apply Changes** button on the top left corner and the variable will be seen under the project as shown below −



Similarly, other variables − list and dictionary variables can be created directly inside TextEdit tab whenever required.

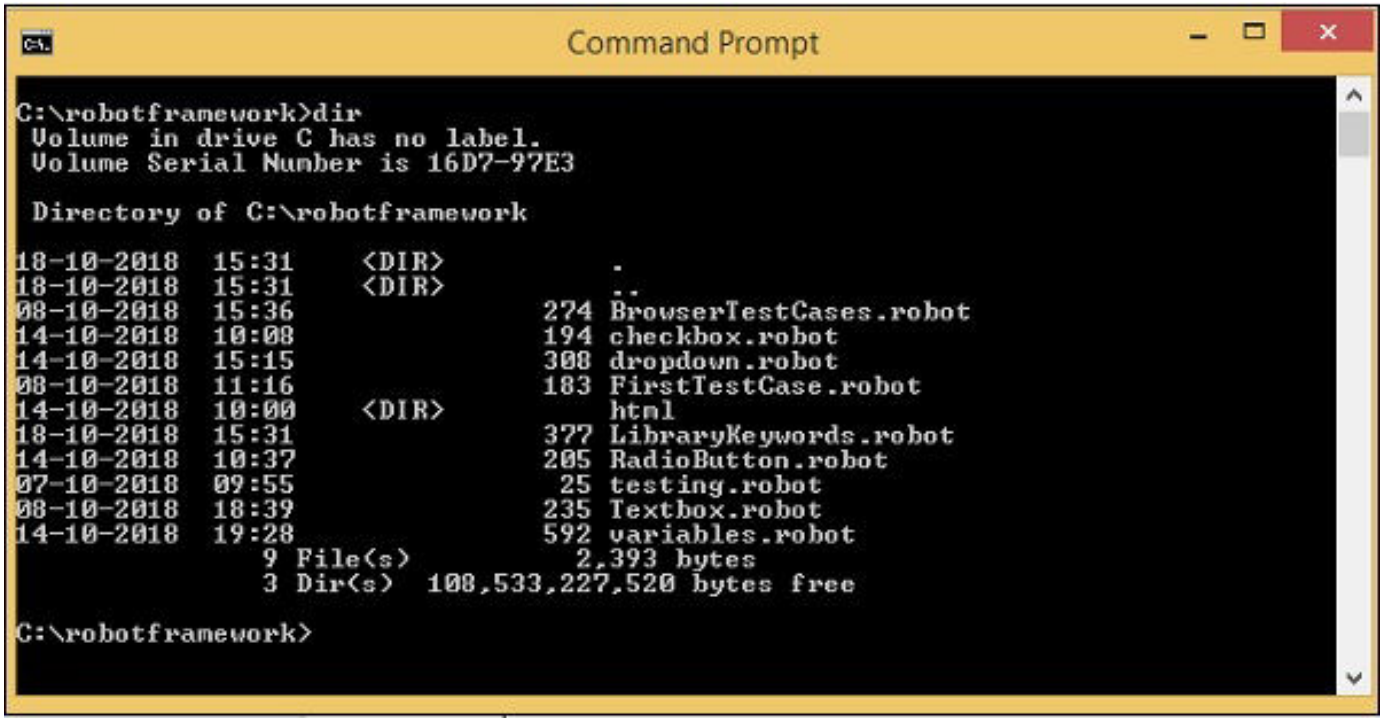
Conclusion

We have seen how to create and use variables. There are three types of variables supported in robot framework − scalar, list and dictionary. We discussed in detail the working of all these variables.

Working With Command Line

In this chapter, we will learn how to make use of the command line to run test cases.

To begin with, let us open the command prompt and go to the folder where your test cases are saved. We have created test cases and saved in the folder **robotframework** in C Drive.



Test cases created so far are available in the folder **C:\robotframework**.

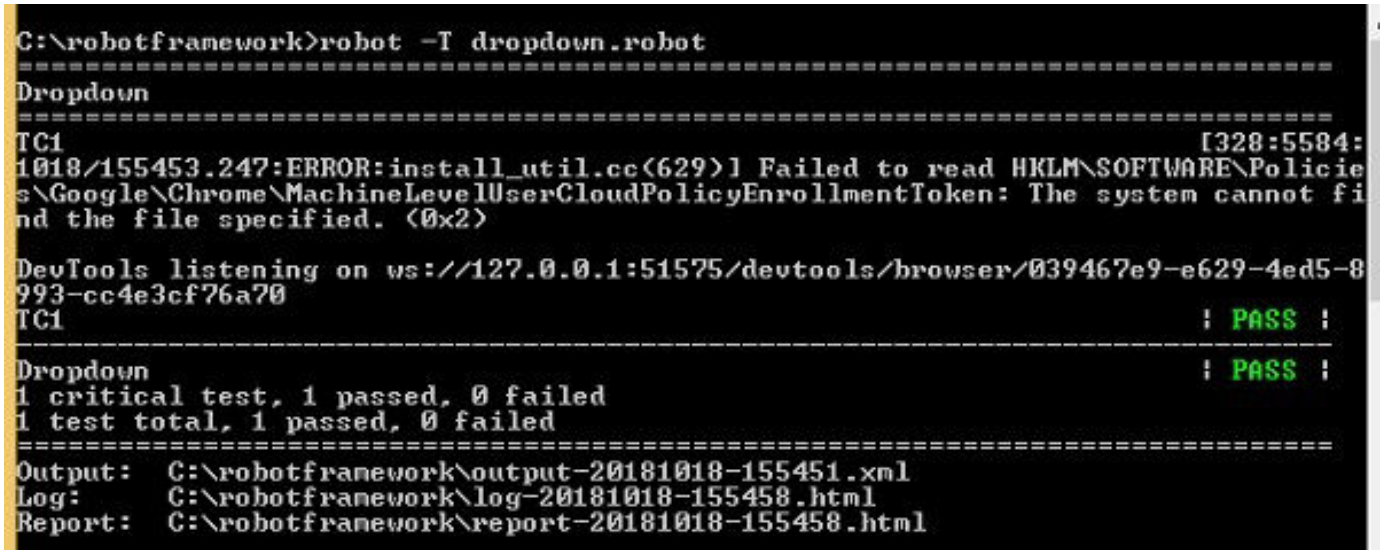
If you have saved your project as a file, the command is −

```
robot -T nameoftestcase.robot
```

If you have saved your project as a directory, the command is −

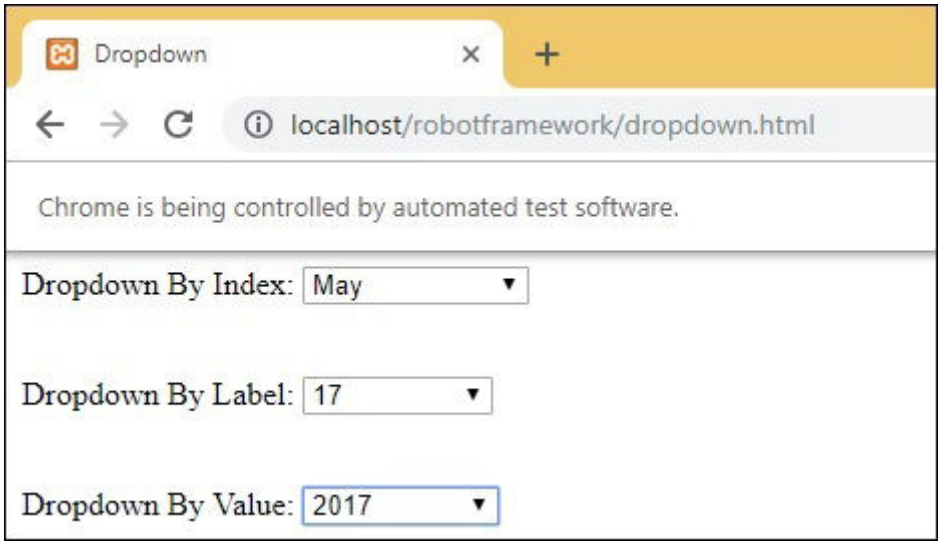
```
robot -T projectname testsuite
```

We will run one of the test created from the folder as shown below −

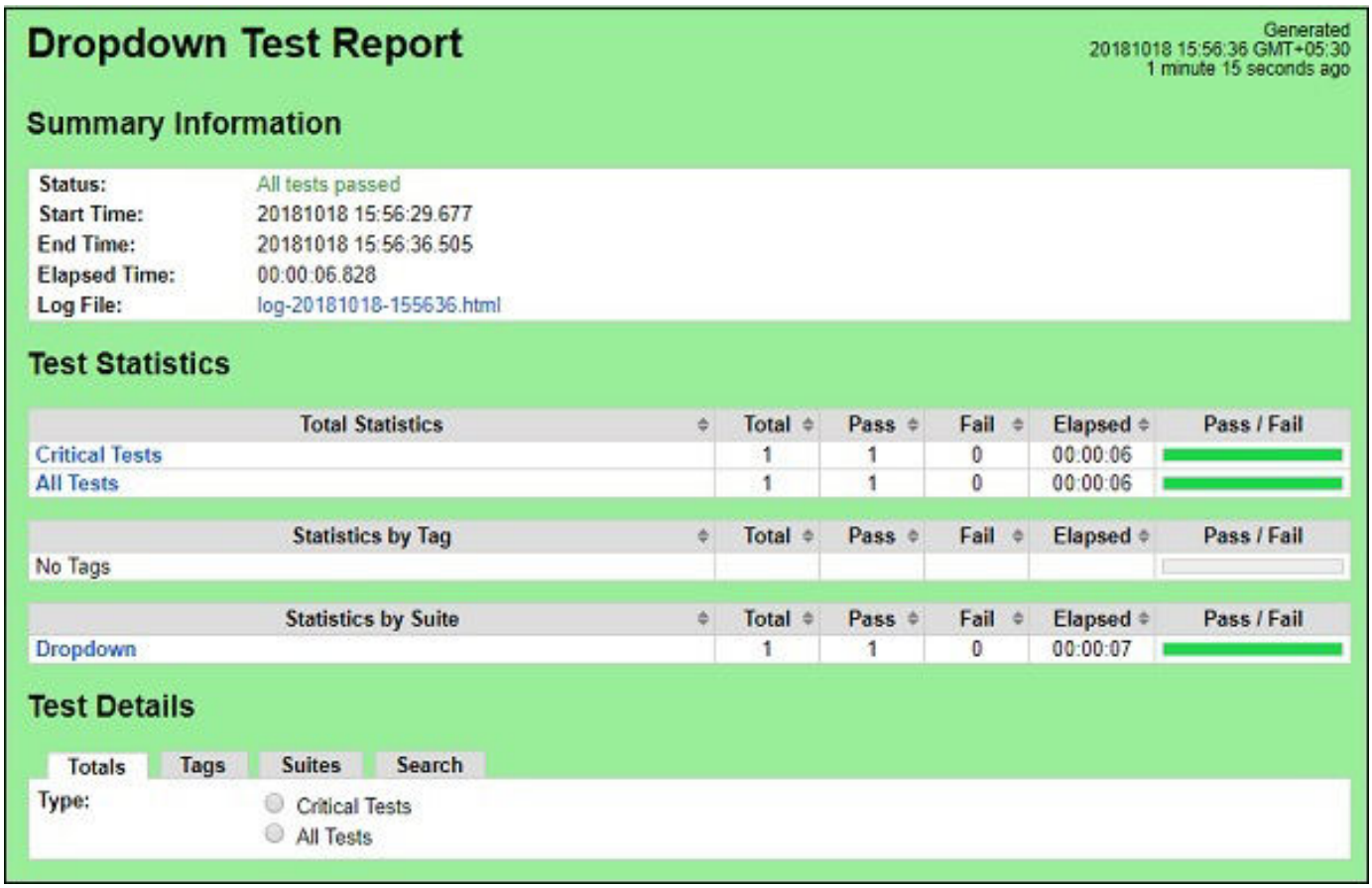


The output, log and report paths are displayed at the end as shown above.

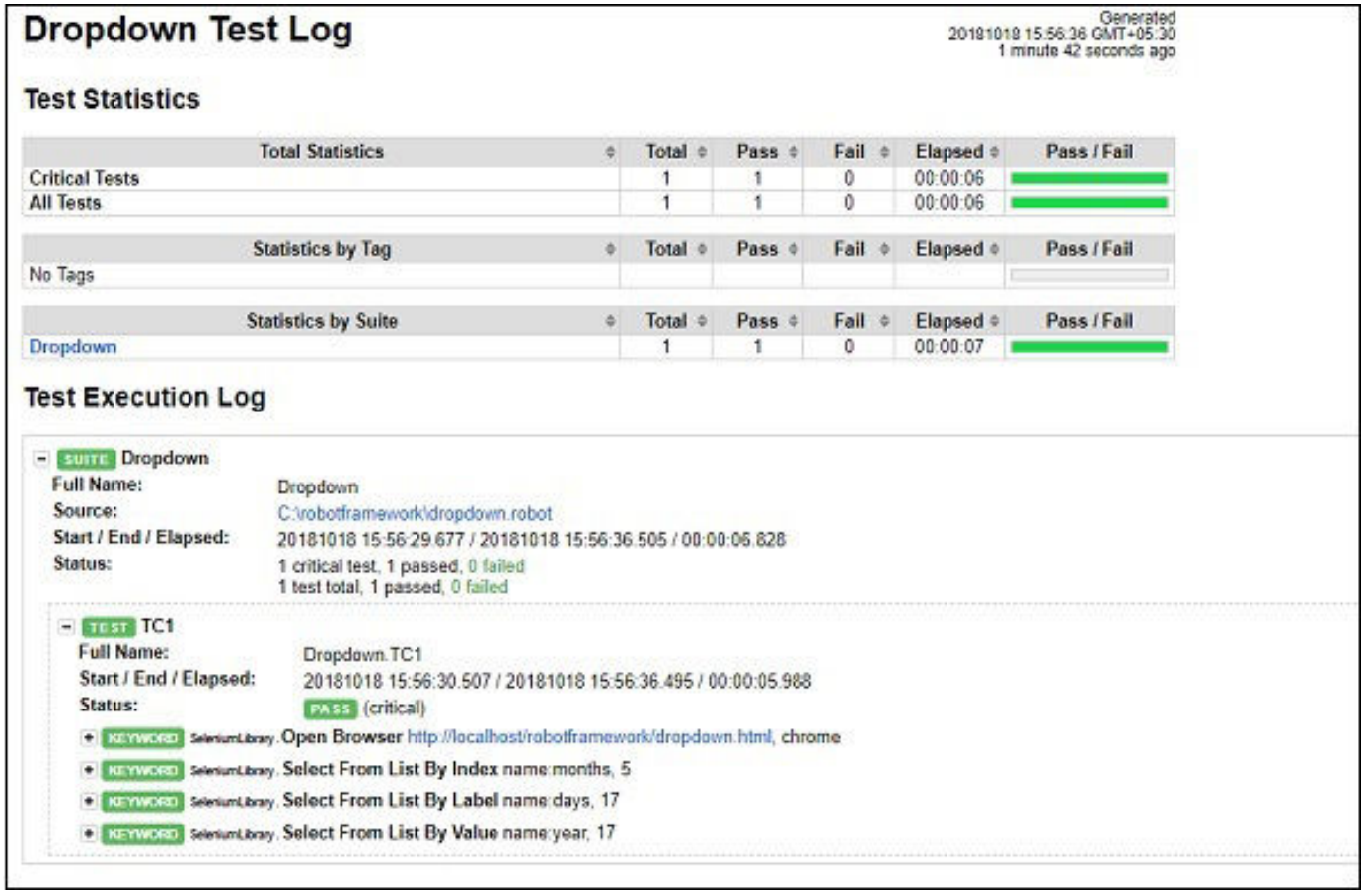
The following screenshot shows the execution details −


Dropdown
localhost/robotframework/dropdown.html
Chrome is being controlled by automated test software.
Dropdown By Index: May
Dropdown By Label: 17
Dropdown By Value: 2017

Report


Dropdown Test Report
Generated
20181018 15:56:36 GMT+05:30
1 minute 15 seconds ago
Summary Information
Status: All tests passed
Start Time: 20181018 15:56:29.677
End Time: 20181018 15:56:36.505
Elapsed Time: 00:00:06.828
Log File: log-20181018-155636.html
Test Statistics
Total Statistics | Total | Pass | Fail | Elapsed | Pass / Fail
Critical Tests | 1 | 1 | 0 | 00:00:06
All Tests | 1 | 1 | 0 | 00:00:06
Statistics by Tag | Total | Pass | Fail | Elapsed | Pass / Fail
No Tags
Statistics by Suite | Total | Pass | Fail | Elapsed | Pass / Fail
Dropdown | 1 | 1 | 0 | 00:00:07
Test Details
Totals
Tags
Suites
Search
Type:
Critical Tests
All Tests

Log



Conclusion

We can use command line to execute robot test cases. The details of the test case pass or fail are displayed in the command line along with log and report URLs.

Working With Setup And Teardown

In this chapter, we will understand two important concepts of testing world − setup and teardown.<

Setup

This is a set of keywords or instruction to be executed before the start of test suite or test case execution.

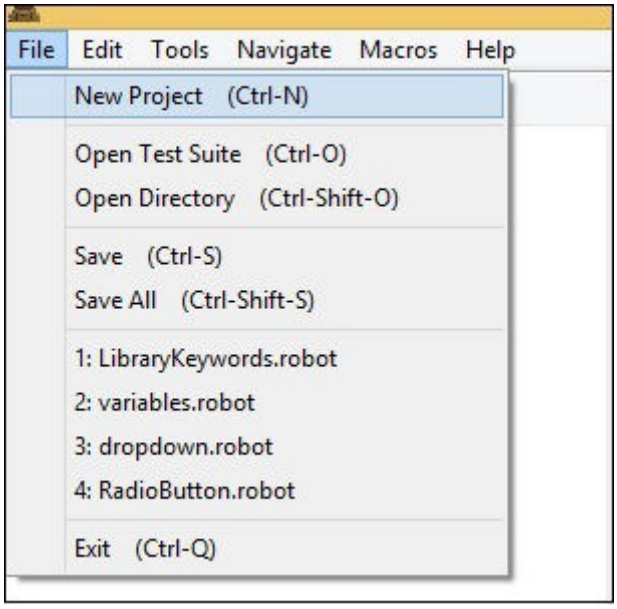
Teardown

This is a set of keywords or instruction to be executed after the start of test suite or test case execution.

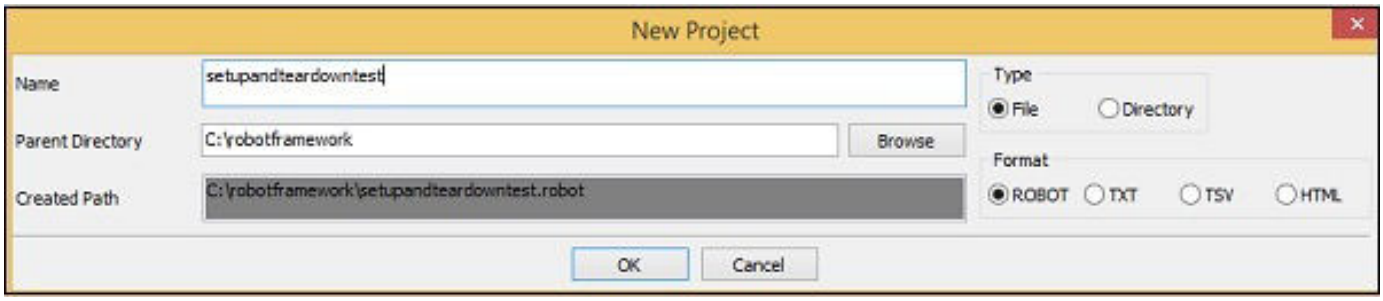
We will work on a project setup, where will use both setup and teardown. The opening and closing of browser are the common steps in test cases.

Now, we will add keyword **open browser** in the setup and close browser in teardown.

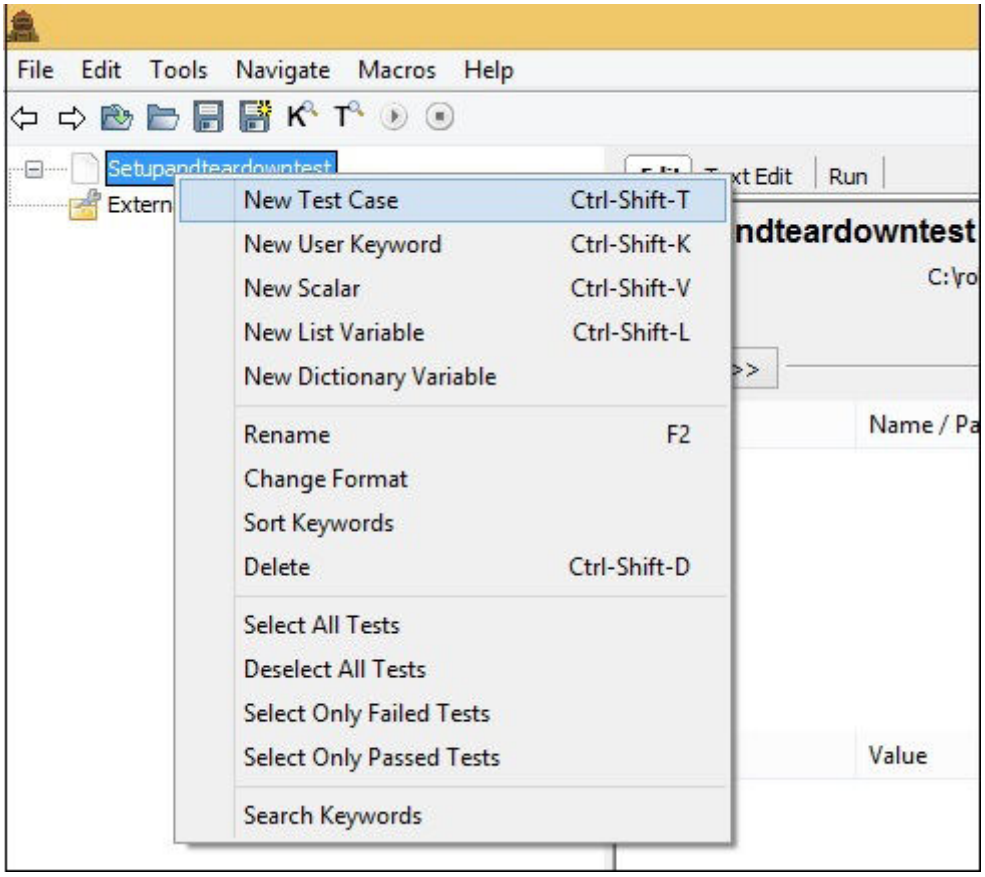
Open Ride using **ride.py** command from command line and create a new project.



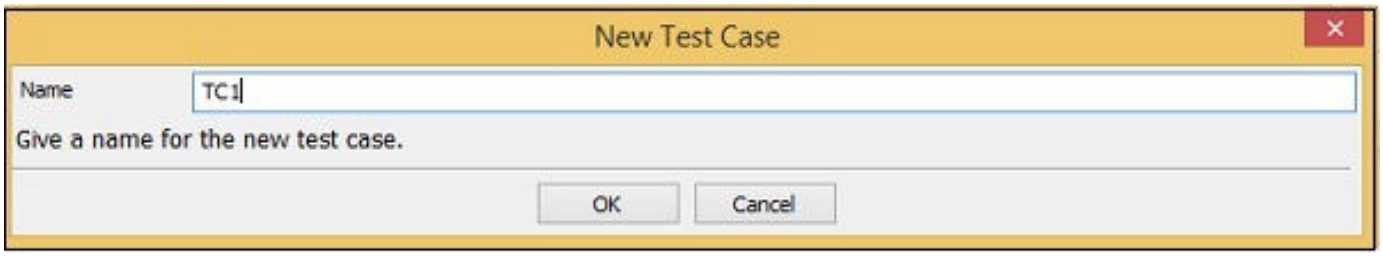
Click *New Project* to create project.



Click OK to save the project.



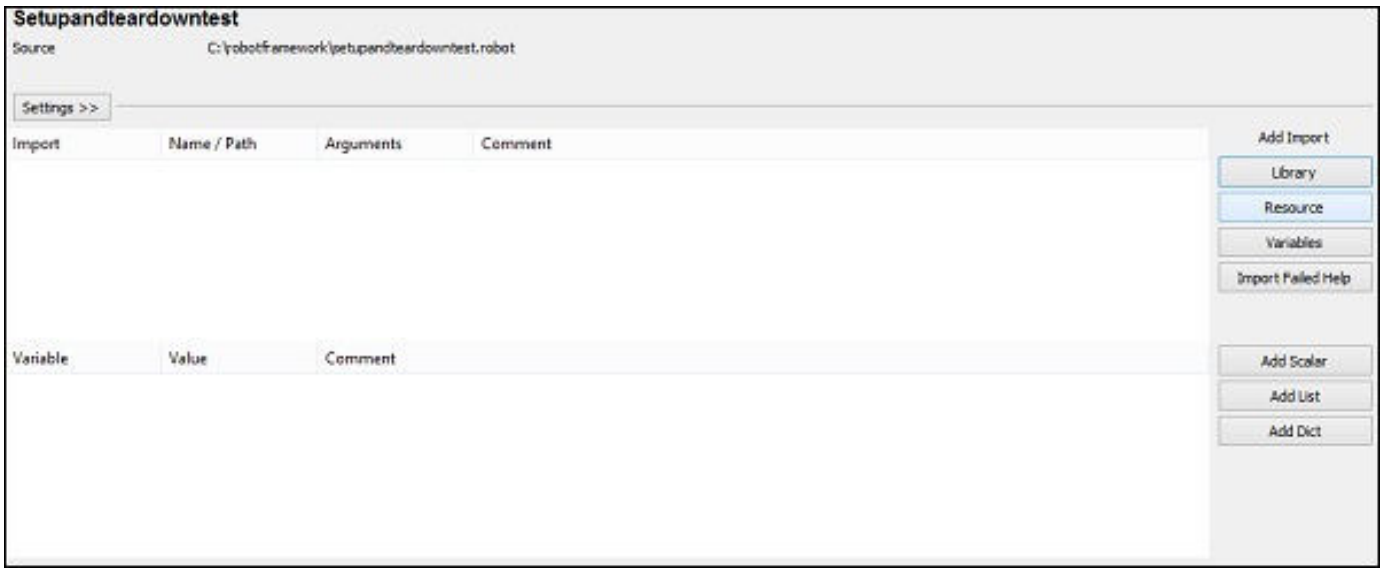
Click **New Test Case** to create one.

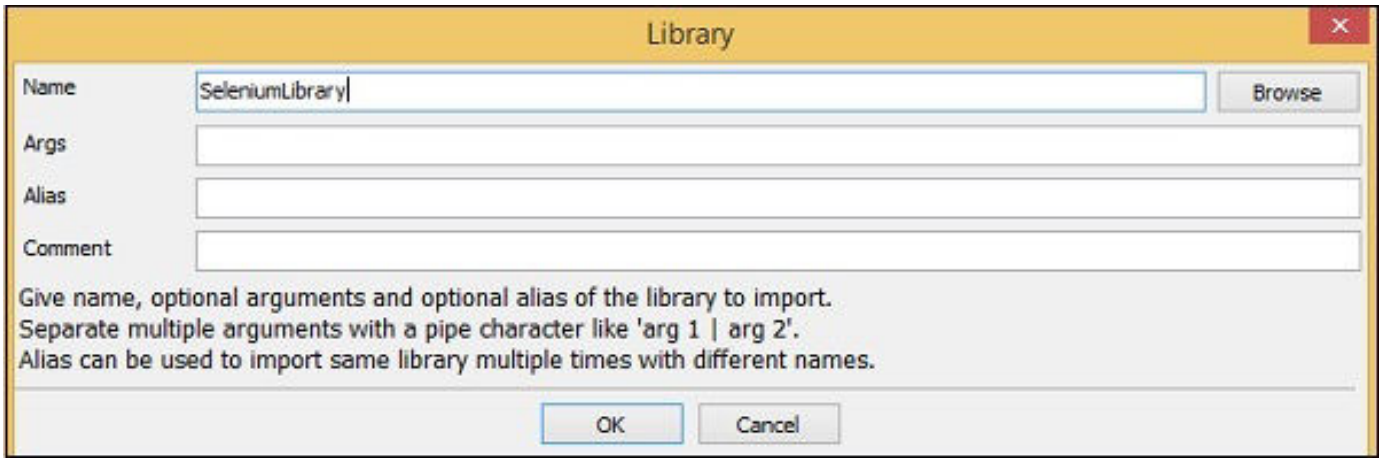


Click OK to save the test case.

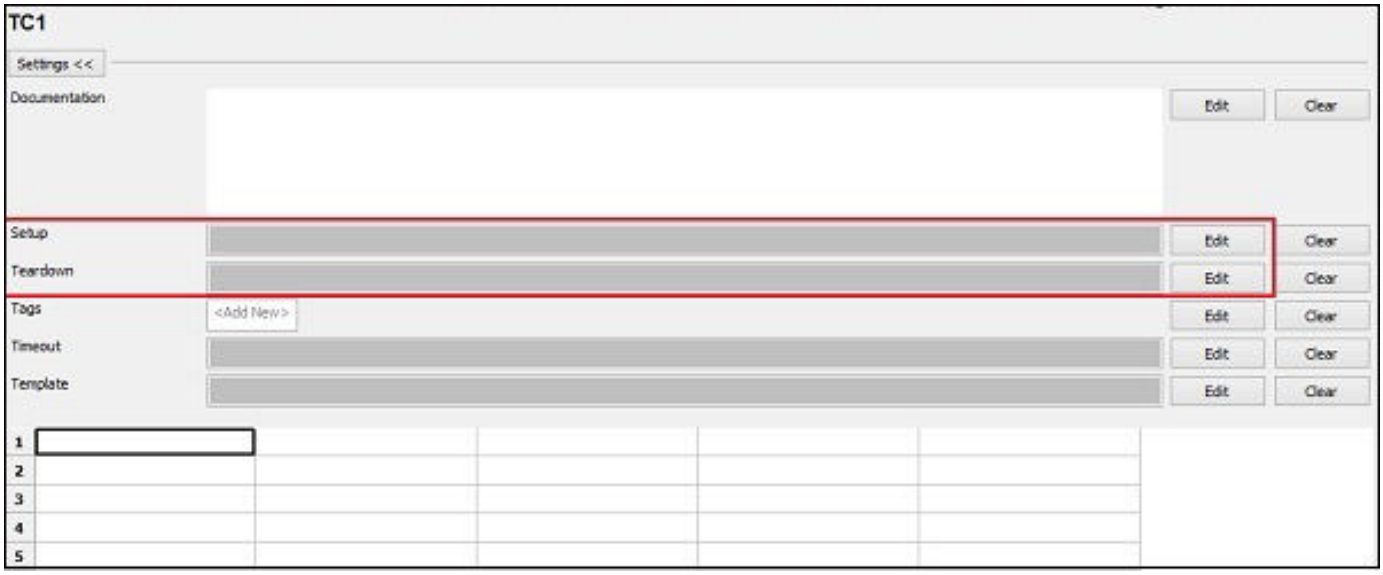
Now we need to import the SeleniumLibrary to use the keywords related to browser and interact with the pages.

To import the library, click Library −



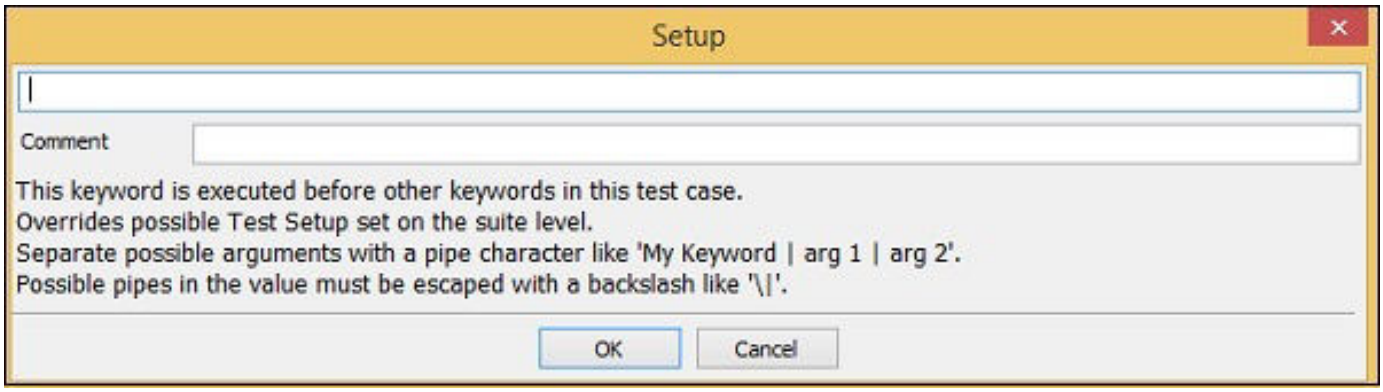


Click OK to save the library.

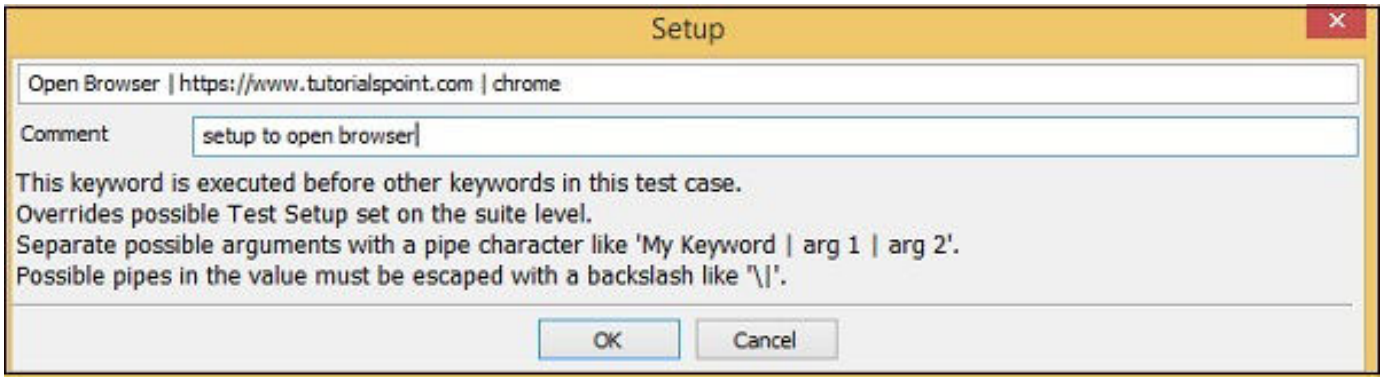


In the above screenshot, the Settings section has *Setup and Teardown* options. For Setup, click **Edit** to enter the keyword.

Now, enter the Keyword −



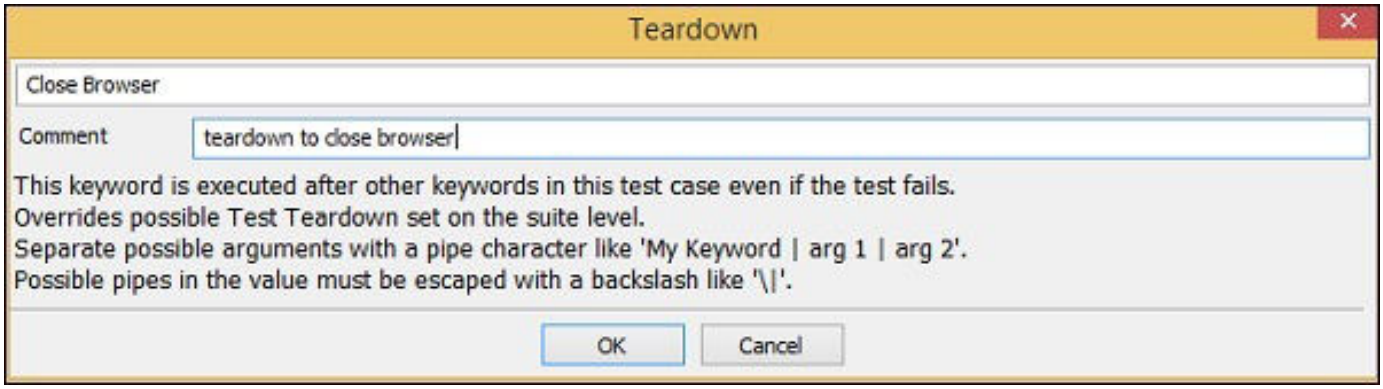
Arguments have to be separated with the pipe character (|).



Click OK to save the Setup. We have entered the Keyword **Open browser** and also added the arguments as shown above.

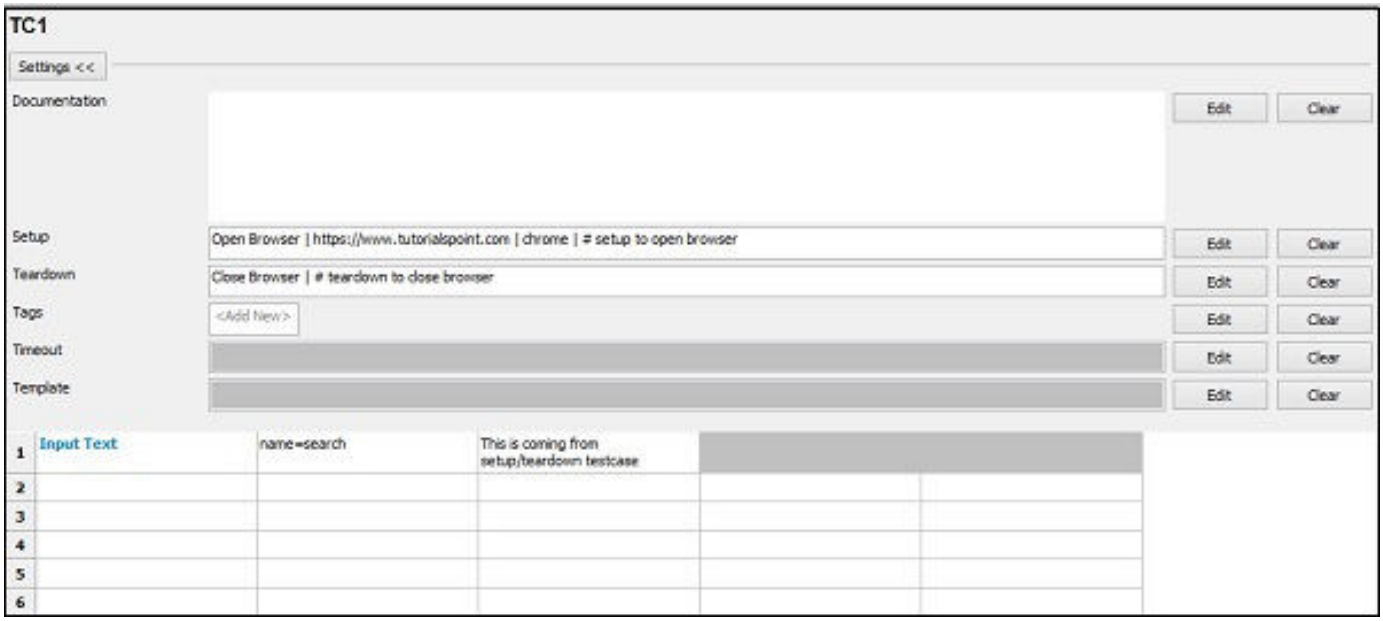
Now, we will enter the teardown case.

Click Edit for Teardown and enter the keyword.



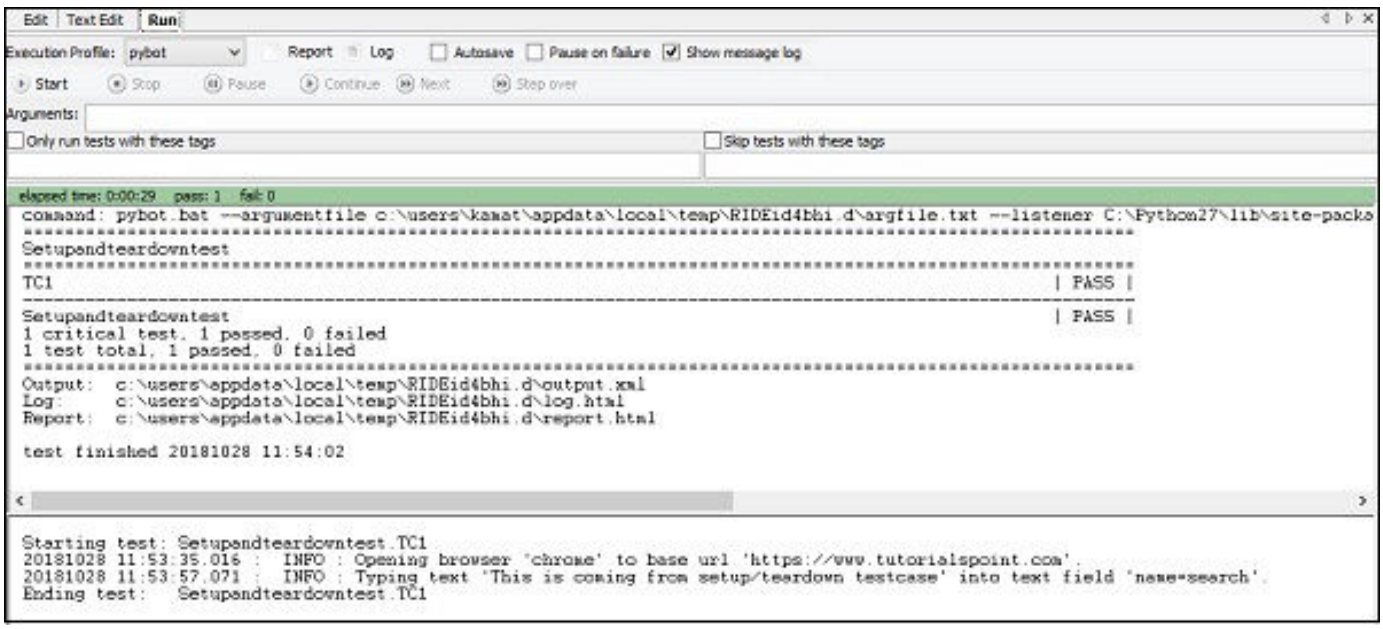
Click OK to save the teardown.

Now, we will enter the keywords for test case.



We have only Input Text in the test case. The opening and closing of the browser is done from Setup and Teardown Settings.

Test Execution Details



Conclusion

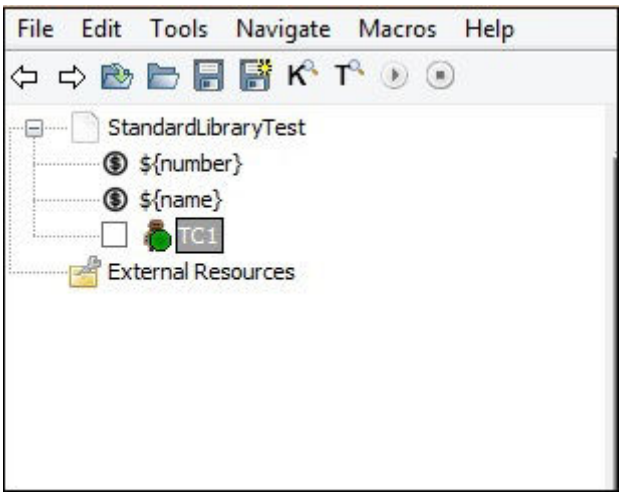
Setup and teardown play a crucial role in the testing world. We have seen how to use setup and teardown in our test cases and also how they are executed.

Working with Built-In Library

In this chapter, we will cover some of the important built-in keywords, which come with the Robot Framework. We can use these keywords along with External libraries for writing test case. We also have the built-in library available with Robot framework by default. It is mostly used for verifications (for example – Should Be Equal, Should Contain), conversions (convert to integer, log to console, etc.).

We will work on a simple test case and will make use of built-in library in that.

We have created project in Ride and Test case as shown below −



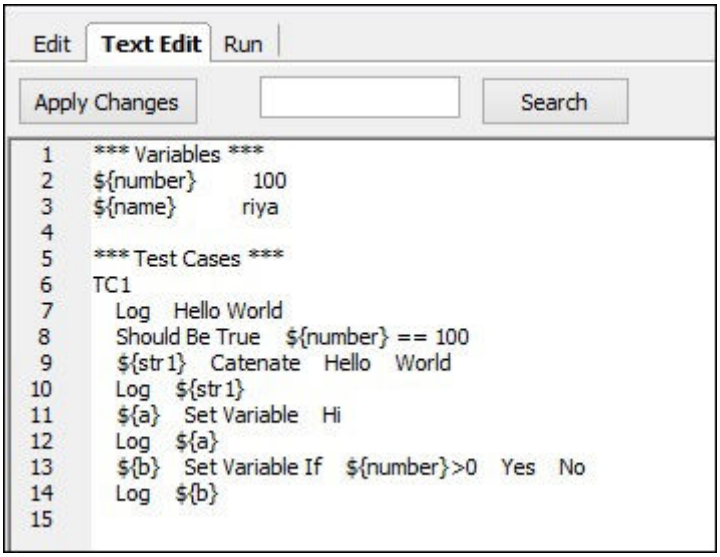
We have created 2 scalar variables – number and name as shown below −

Variable	Value	Comment
${number}	100	
${name}	riya	

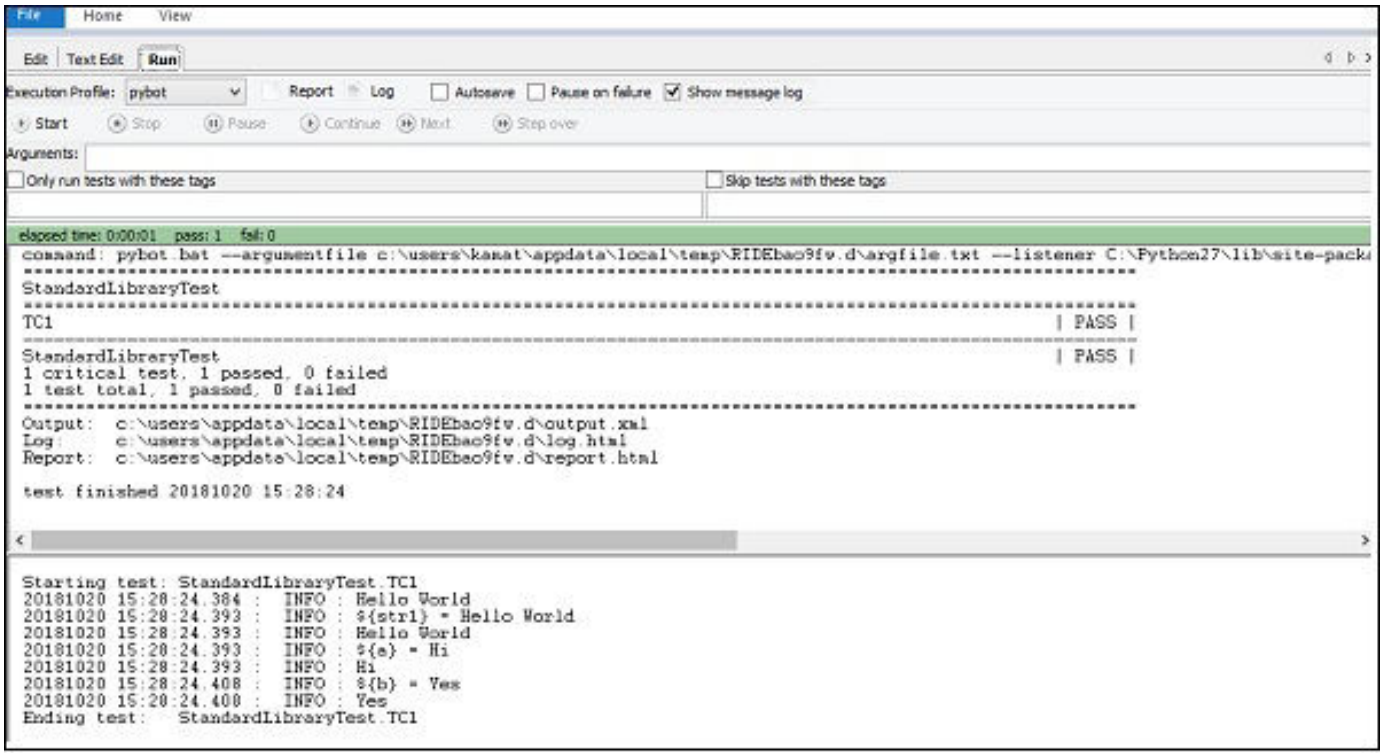
Here are the test cases used for comparing number, string, concatenate, etc. We have used simple keywords in the test cases below. The keywords are shown in tabular format here −

1	Log	Hello World			
2	Should Be True	${number} == 100			
3	${str 1}	Catenate	Hello	World	
4	Log	${str 1}			
5	${a}	Set Variable	Hi		
6	Log	${a}			
7	${b}	Set Variable If	${number}>0	Yes	No
8	Log	${b}			

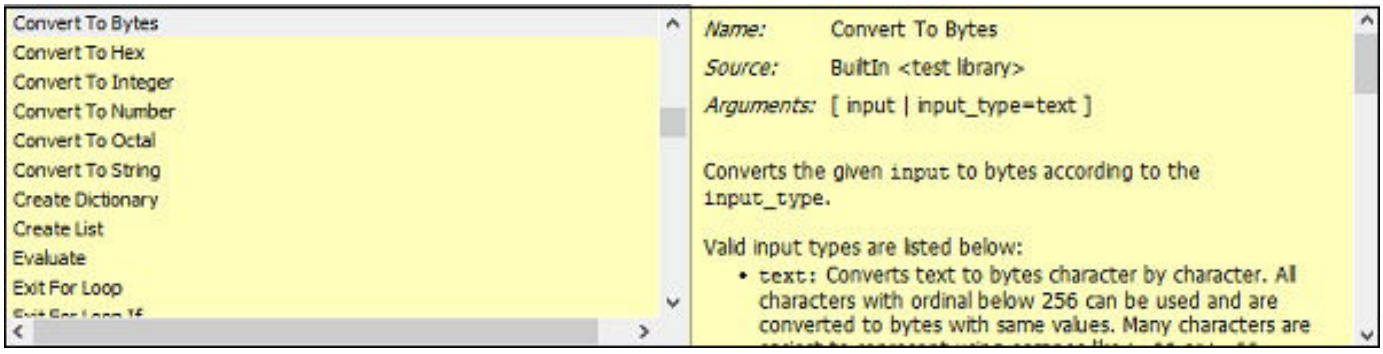
Following is the test code for above test cases from text edit −



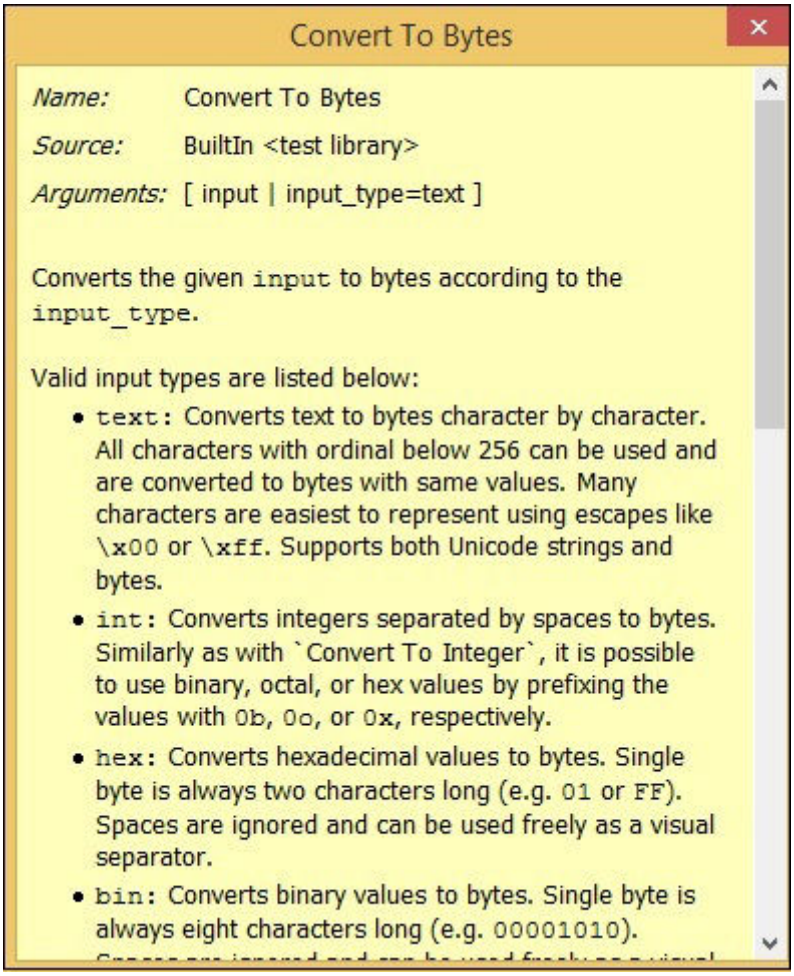
Now, we will execute the test case to see the results −



When you write your keywords in tabular format, press ctrl + spacebar. It gives the list of built-in keywords available with Robot Framework.



It gives the details of each keyword with example in the corresponding window. When we click on the corresponding window, it will open separately as shown below −



Conclusion

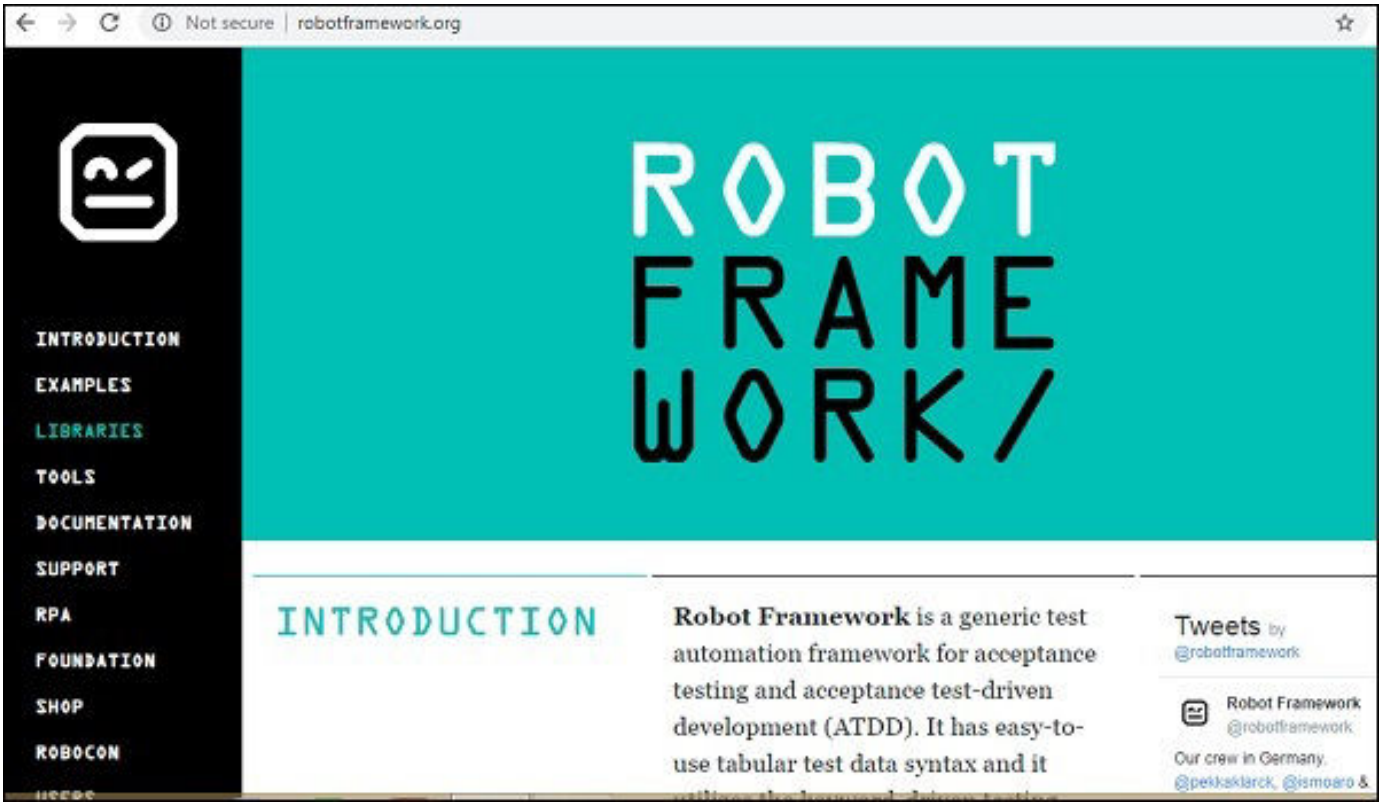
We have seen keywords related to string, numbers, log messages, which are available with robot framework by default. The same can be used along with external library and also can be used to create user-defined keyword to work with test-cases.

Working With External Database libraries

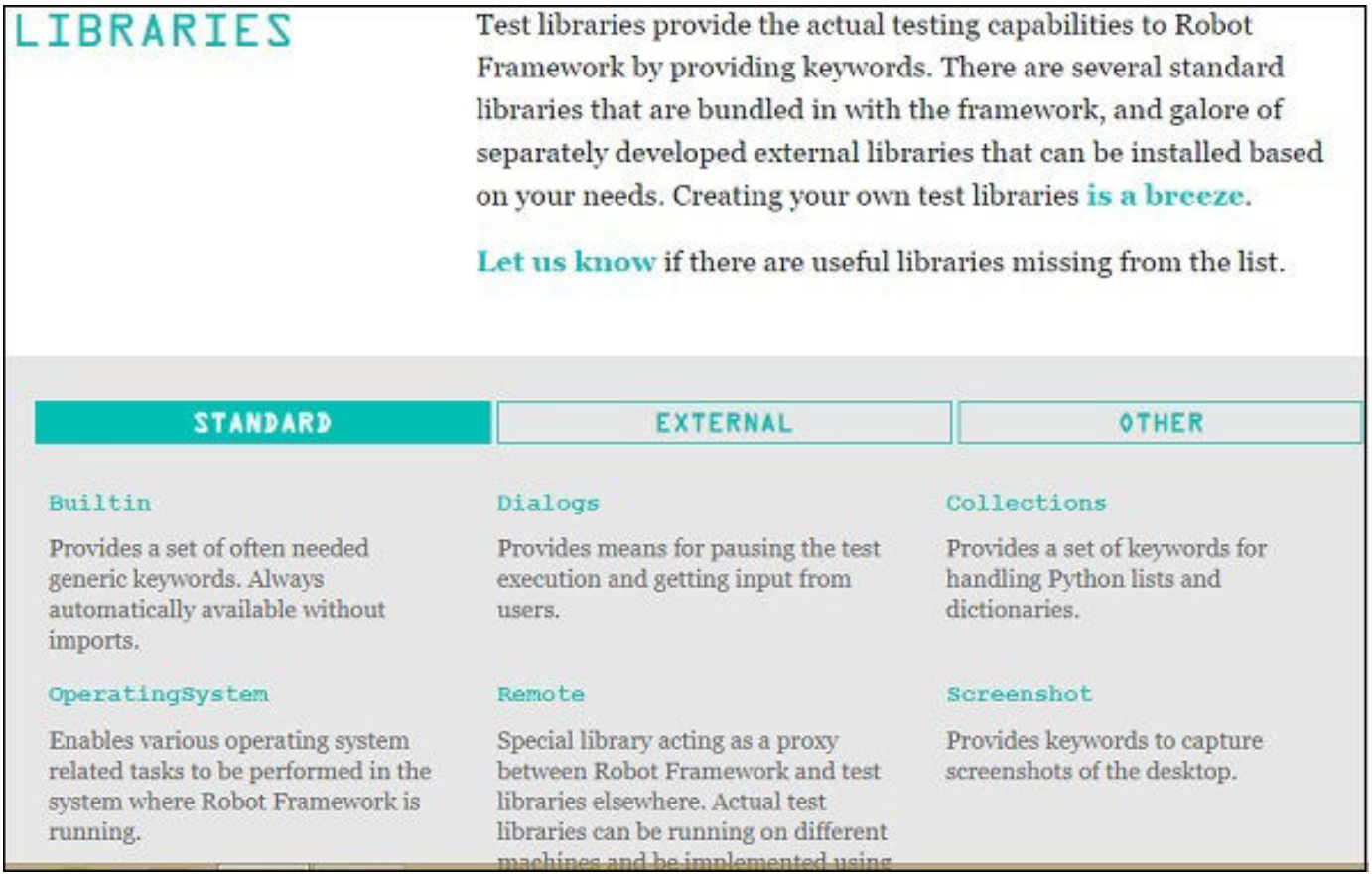
We have seen how we can work with Selenium Library. The detailed installation/importing of Selenium Library is discussed in chapter *"Working with Browsers using Selenium Library"*.

In this chapter, we will discuss database library and how to connect and test database using Robot Framework.

Go to Robot framework site https://robotframework.org/ and click **Libraries** as shown below –

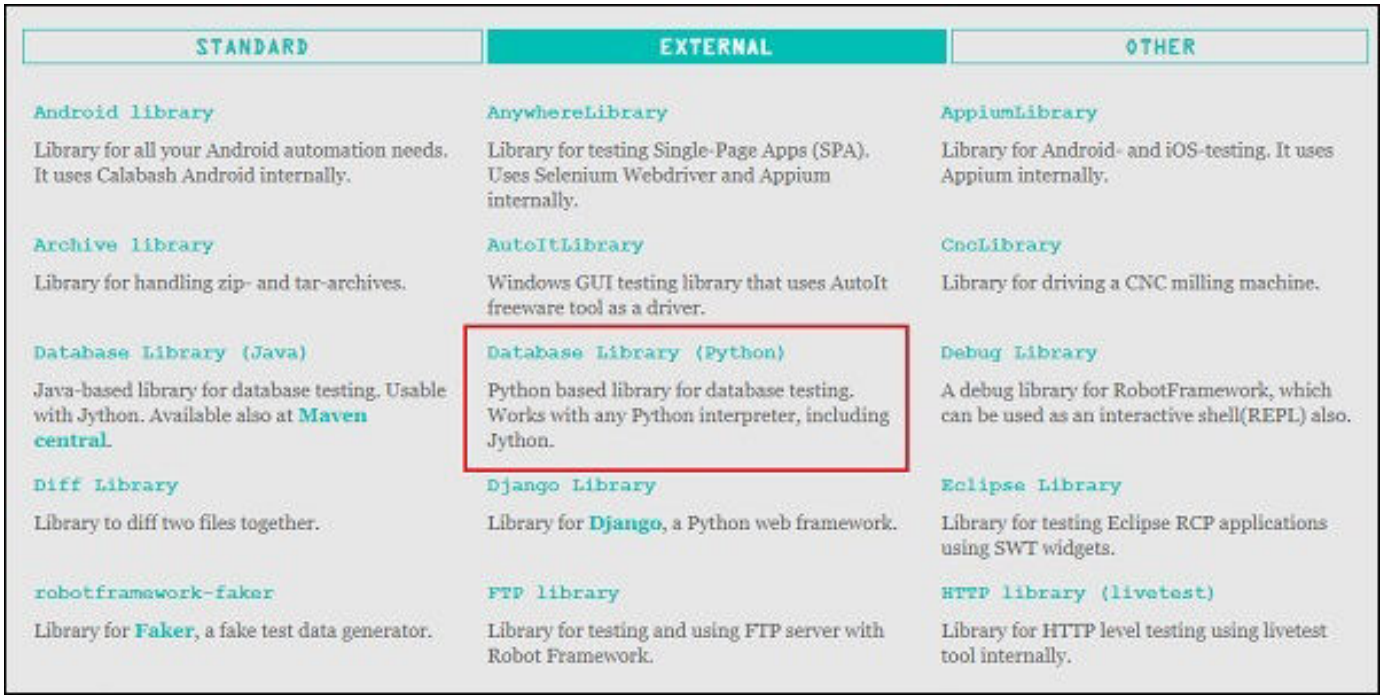


Upon clicking Libraries, you will be redirected to a screen as shown below −



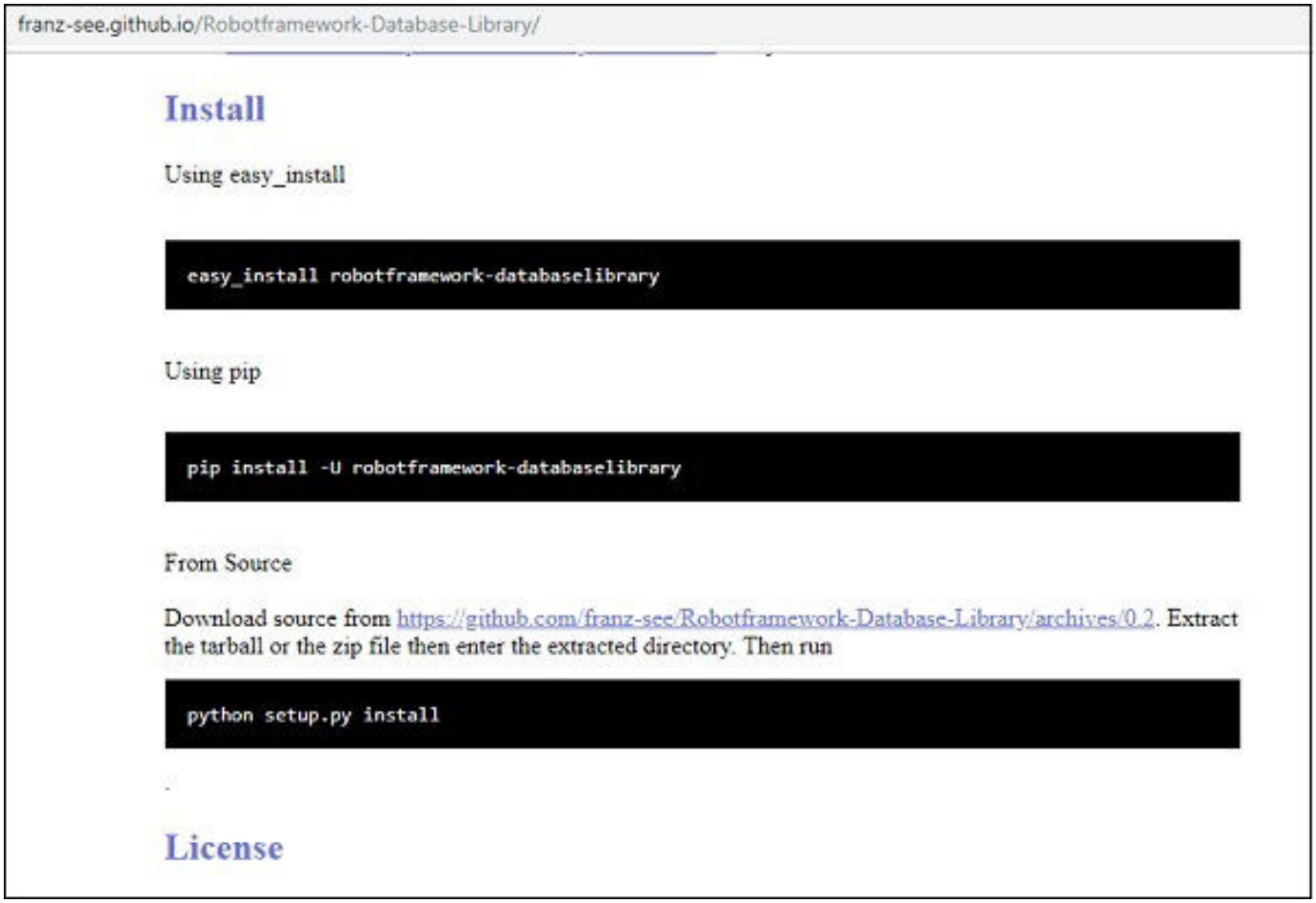
The Libraries are categorized as Standard, External and Other.

We will now take a look at the external library in this chapter. Upon clicking External, the following screen appears −



It shows the list of external libraries supported by Robot Framework. Here, we will focus more on the Database Library (Python). The same has been highlighted in the screenshot above.

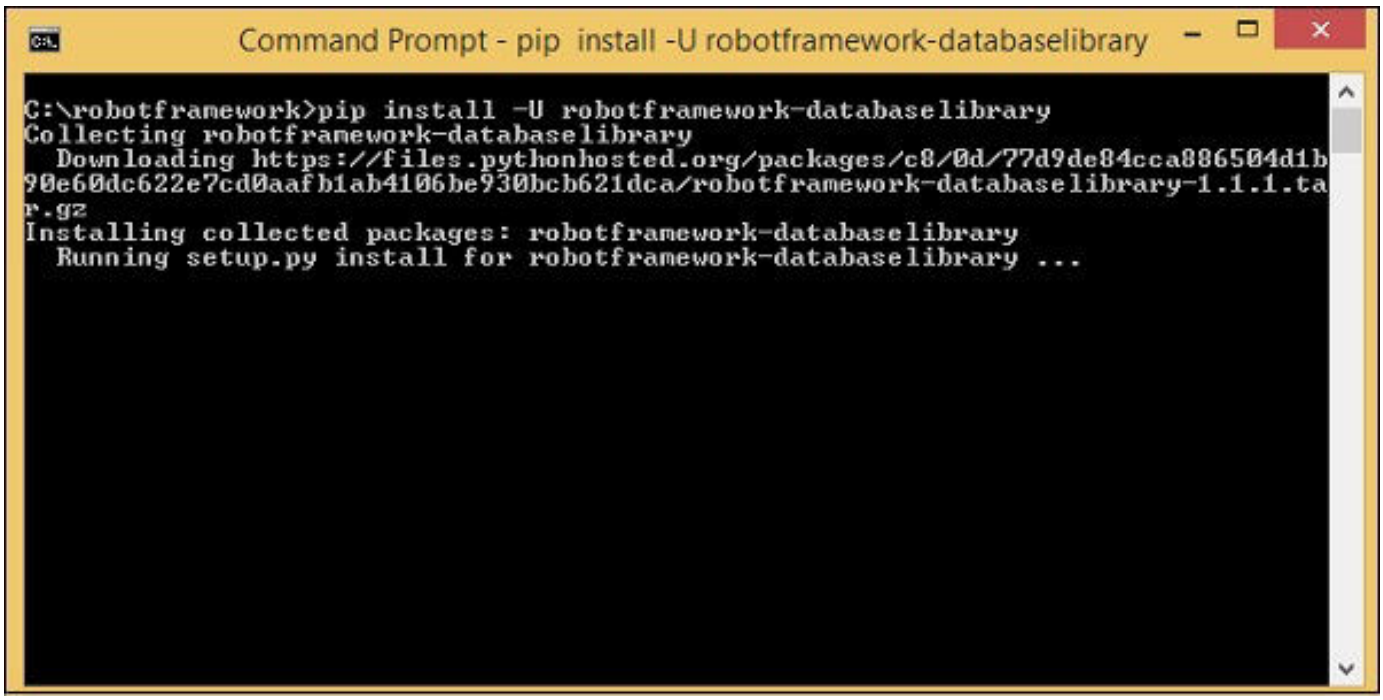
Upon clicking the Database Library (Python), you will be redirected to the screen where the instruction for installation are listed as shown in the following screenshot −



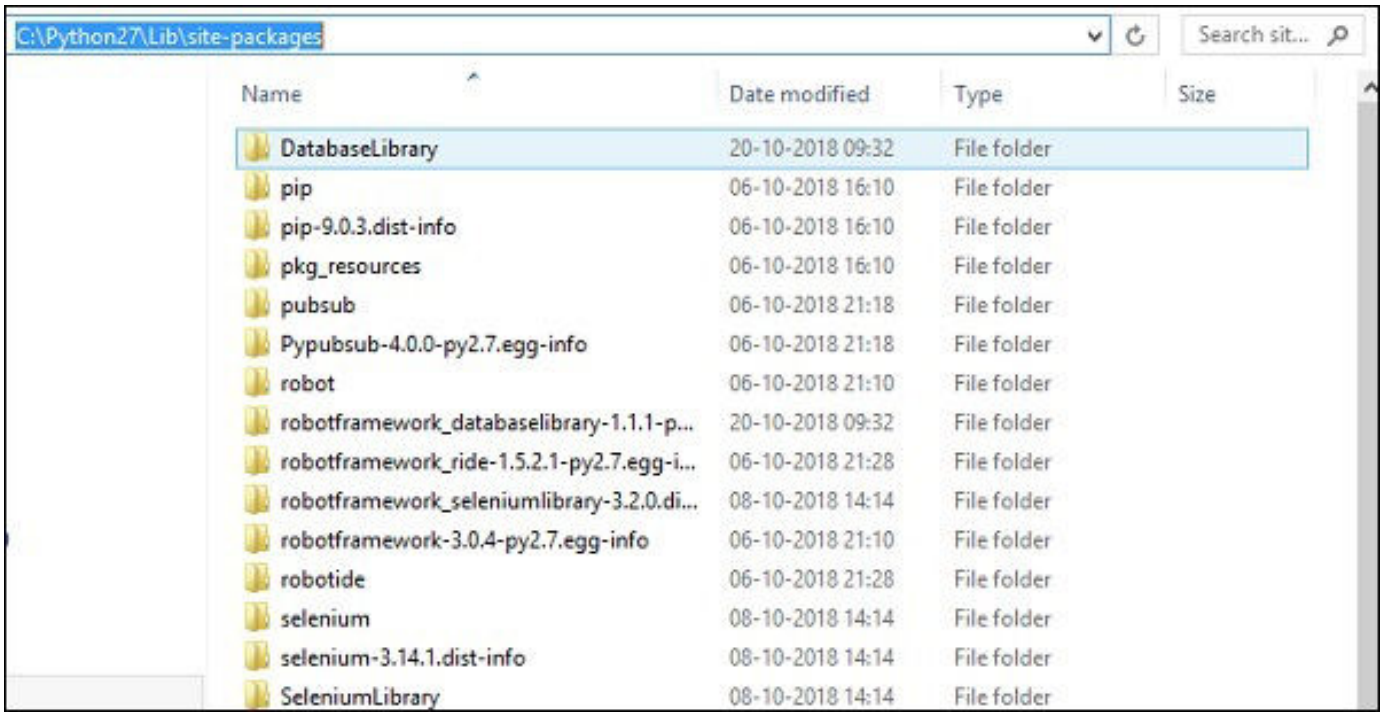
We can install the database library using pip and the command is −

```
pip install -U robotframework-databaselibrary
```

Run the above command in the command line as shown below −



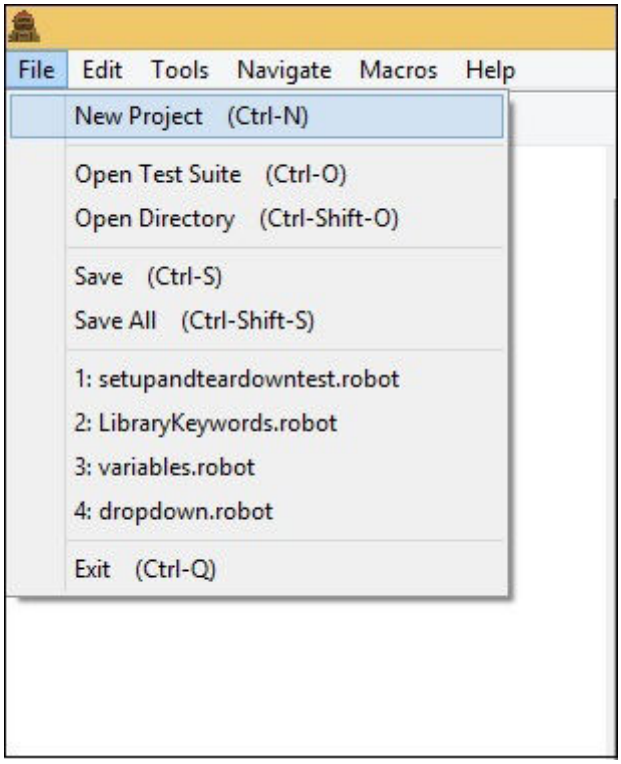
The Library is stored in python lib folder as shown below −



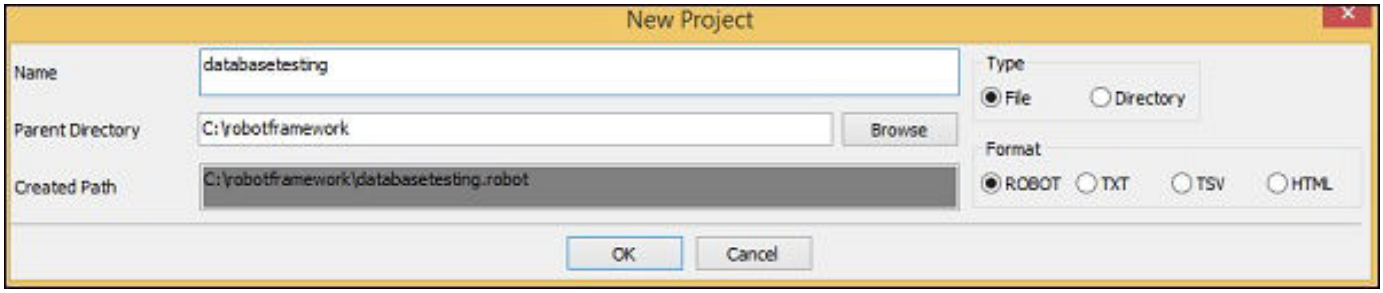
Once the installation is done, the next step is to import the library inside the project and use it with test cases.

Import Database Library

Open ride using **ride.py** from command line and create the project for testing database.

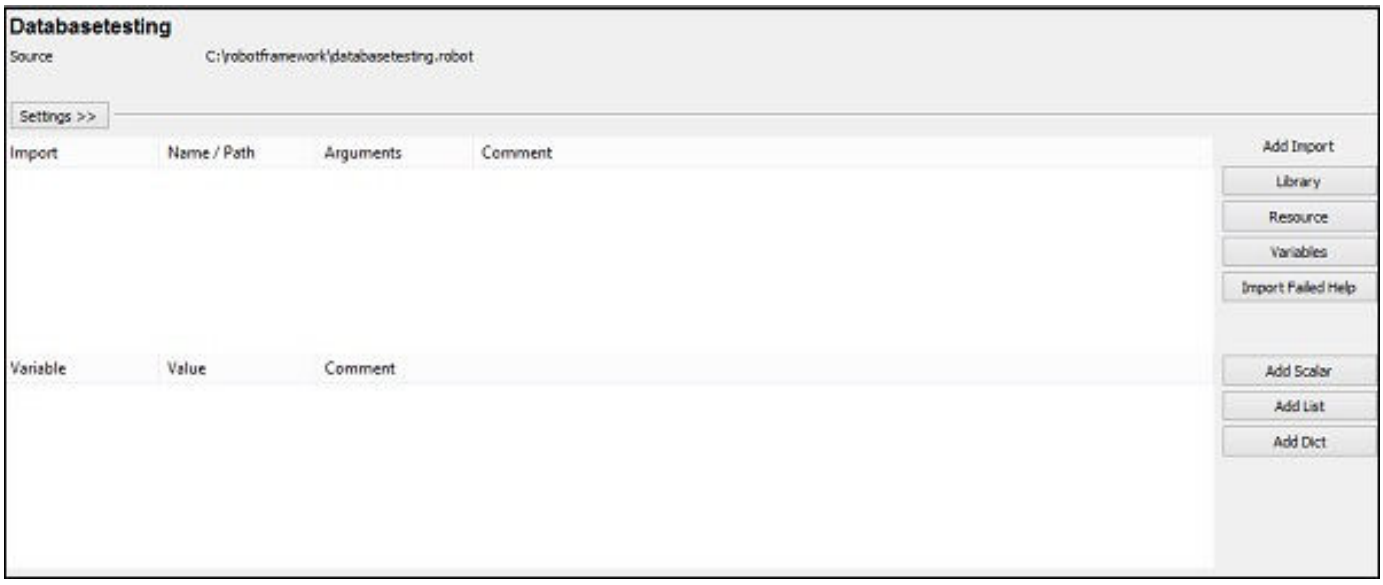


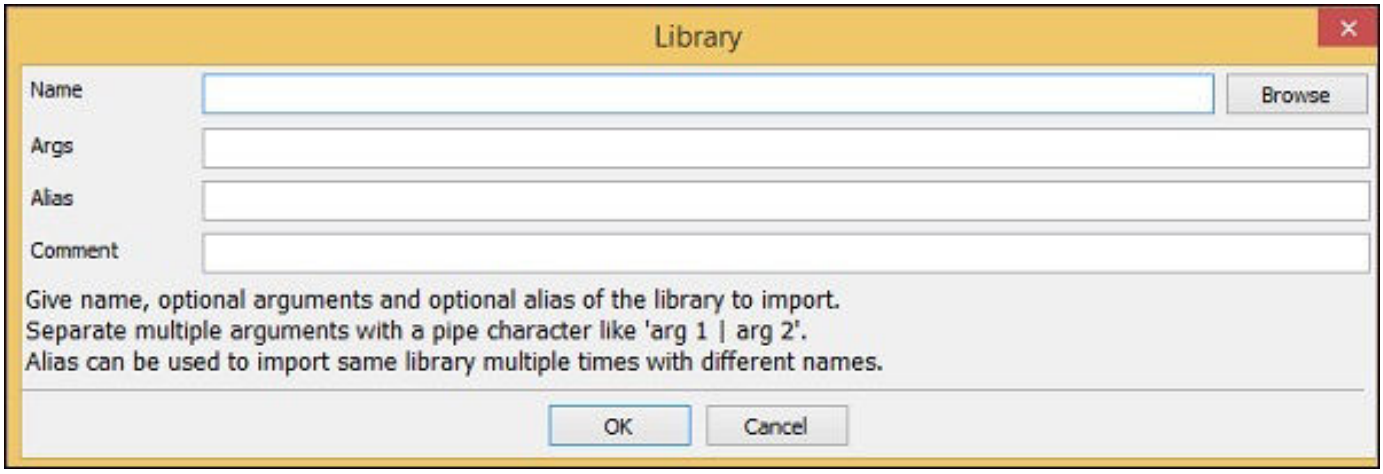
Click New Project and give a name to the project.



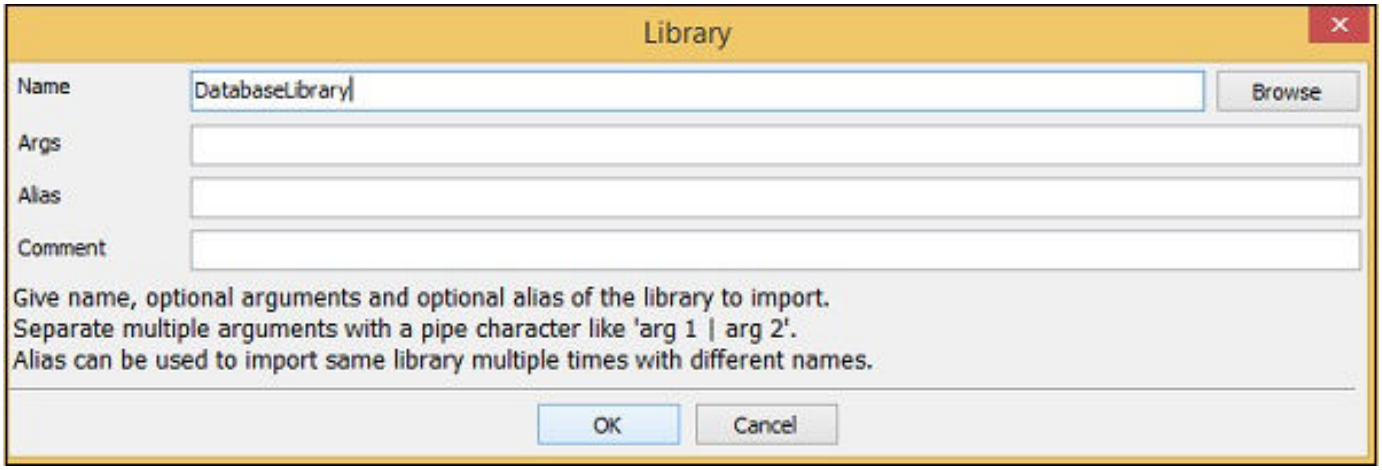
Click OK to save the project.

Click Library below Add Import.

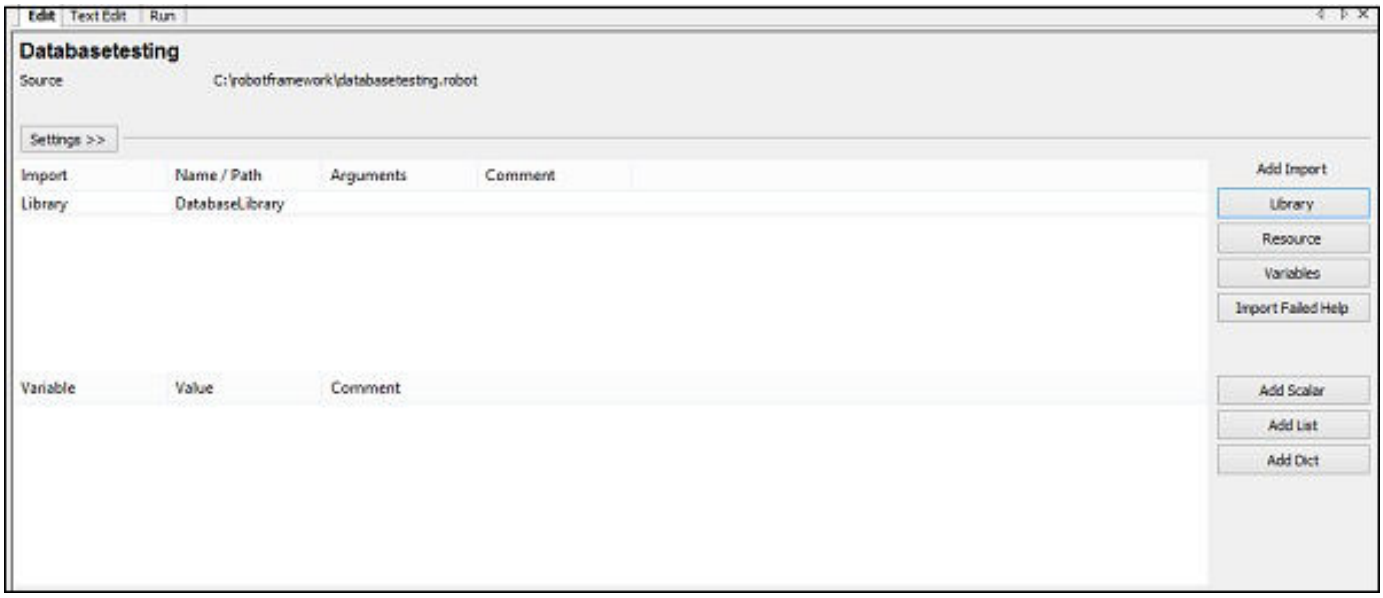




Enter the Name of the Library as DatabaseLibrary as shown below and click OK.



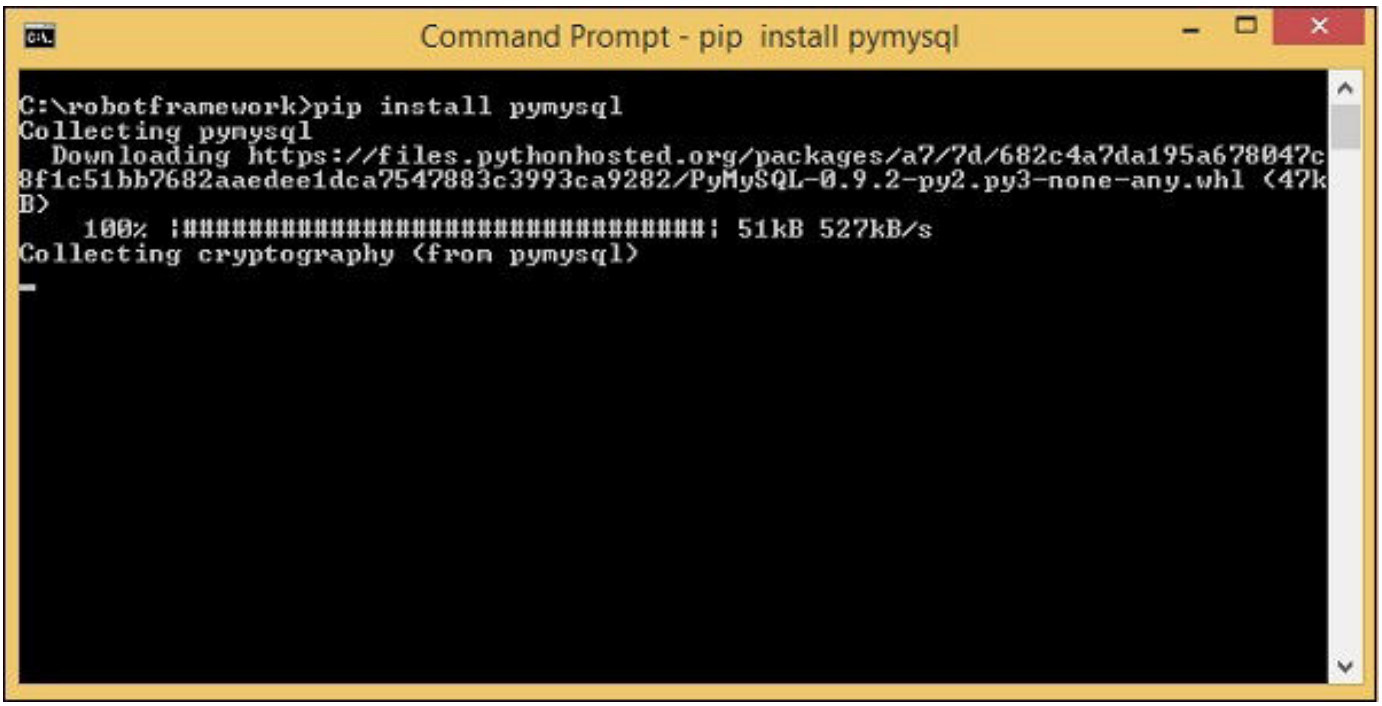
Once saved, the library is as shown below −



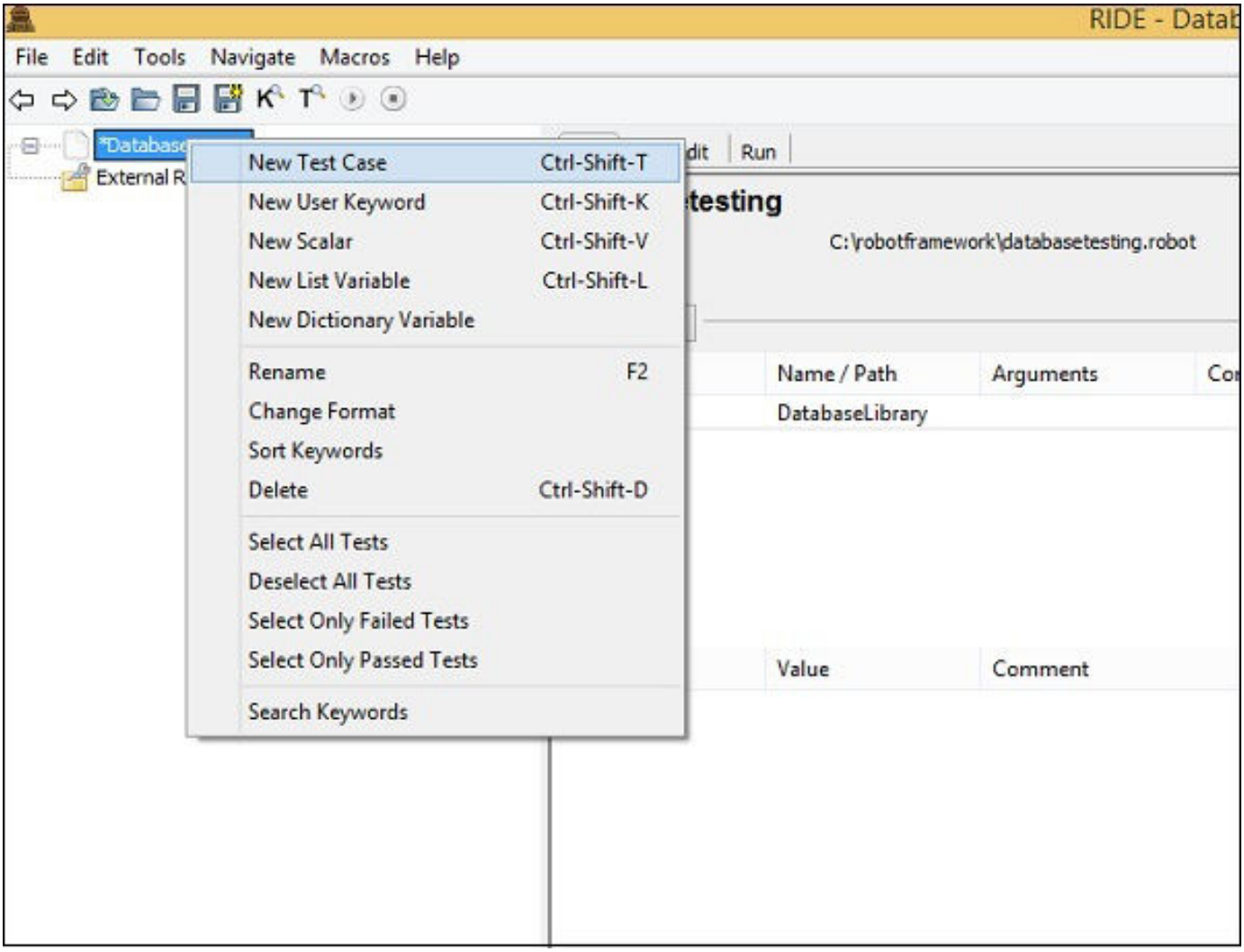
We are going to work with MySQL Database. To work with MySQL, we need to install the module.

Command

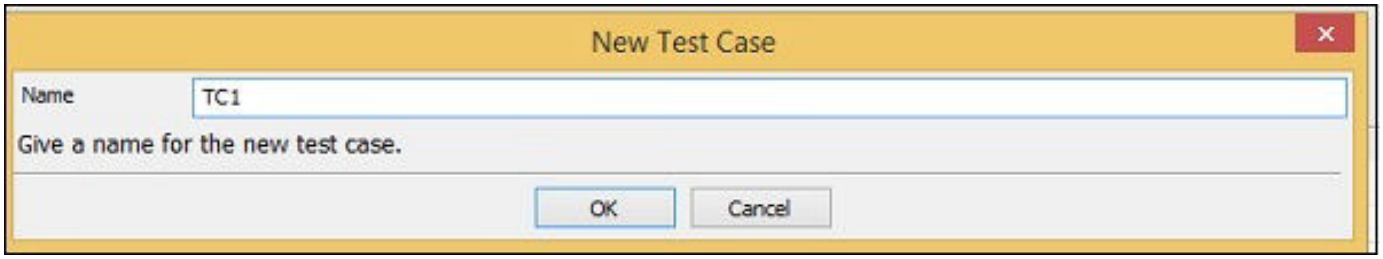
```
pip install pymysql
```



Now create test case under the project created.



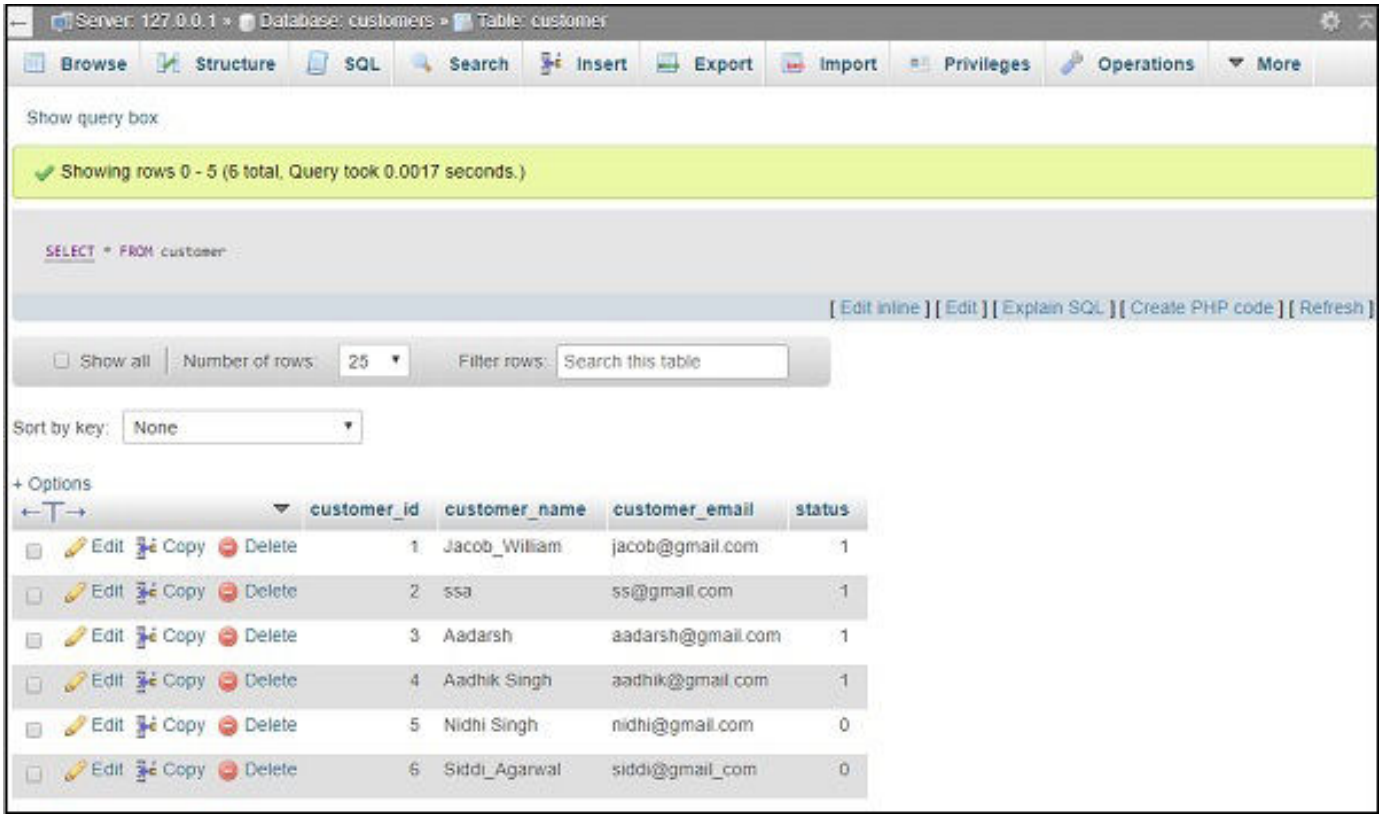
Click New Test Case −



Enter the name of the test case and click OK.

We are going to use the existing database called customers available with us.

We will use phymyadmin to show the customer database −



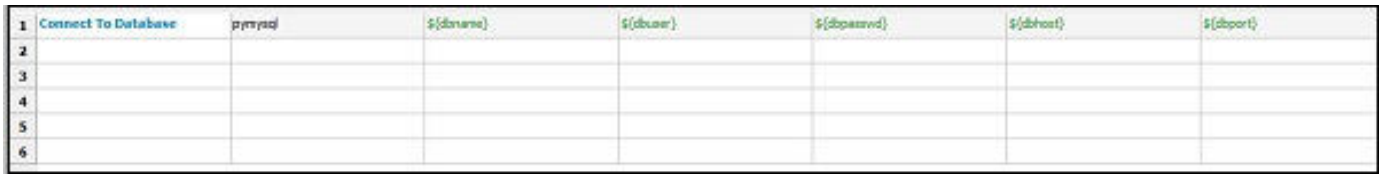
We have a table called customer, which has data distributed in 6 rows. Now will write test-case which will connect to MySQL database customers and fetch the data from customer table.

Before we start, we will create scalar variables which will hold the data for dbname, dbuser, dbpasswd, dbhost, dbport and queryresult to store data, etc. Here are the variables created with values −

Variable	Value	Comment
${dbname}	customers	
${dbuser}	root	
${dbpasswd}	admin	
${dbhost}	localhost	
${dbport}	3306	
@{queryResults}		

The command to connect to database is −

```
Connect To Database pymysql ${dbname} ${dbuser}
${dbpasswd} ${dbhost} ${dbport}
```



We will add some more test cases as shown below −

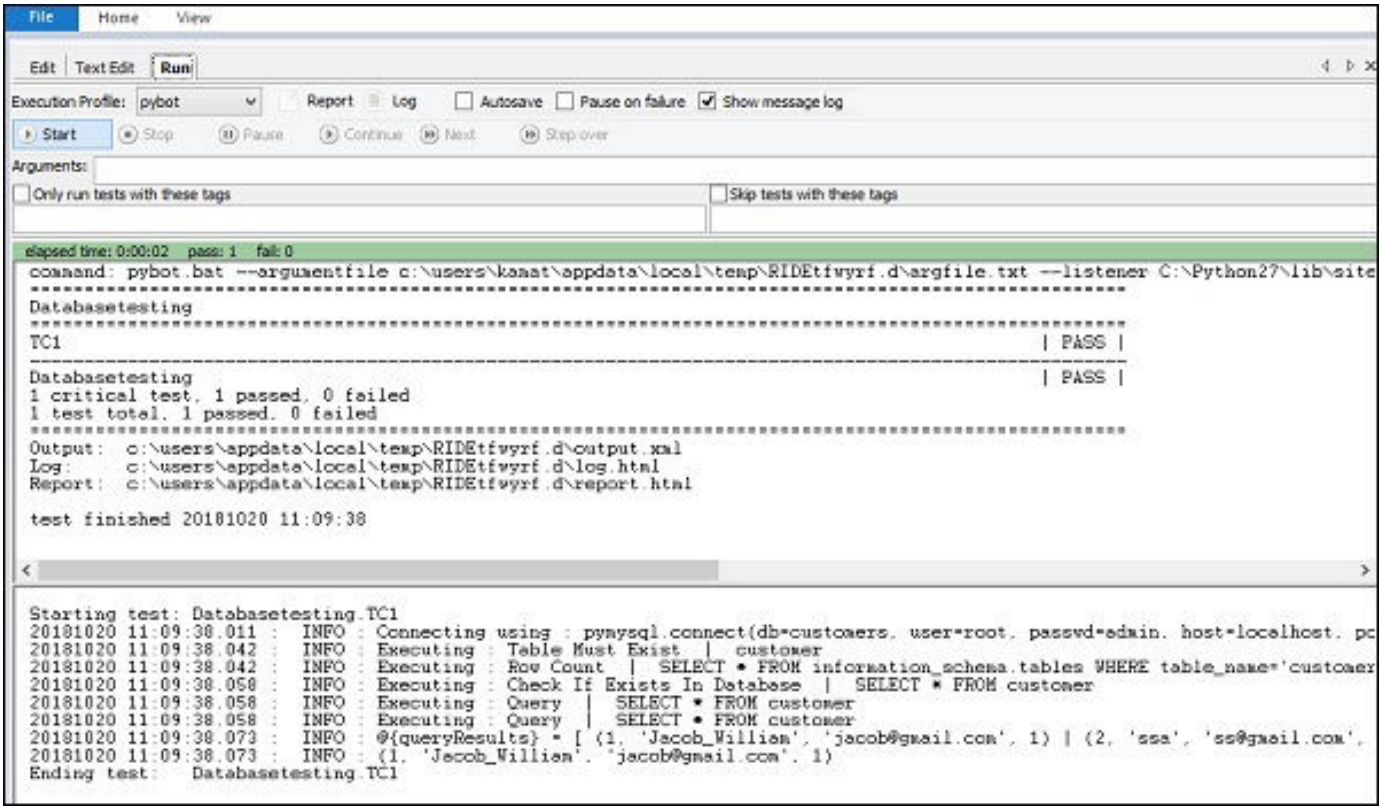
1	Connect To Database	pymysql	${dbname}
2	Table Must Exist	customer	
3	Check If Exists In Database	SELECT * FROM customer	
4	@{queryResults}	Query	SELECT * FROM customer
5	Log	@{queryResults}[0]	
6			
7			
8			

Here are the details −

```
*** Settings ***
Library DatabaseLibrary

*** Variables ***
${dbname} customers
${dbuser} root
${dbpasswd} admin
${dbhost} localhost
${dbport} 3306
@{queryResults}

*** Test Cases ***
TC1

   Connect To Database pymysql ${dbname} ${dbuser}
   ${dbpasswd} ${dbhost} ${dbport}
   Table Must Exist customer
   Check If Exists In Database SELECT * FROM customer
   @{queryResults} Query SELECT * FROM customer
   Log @{queryResults}[0]
```

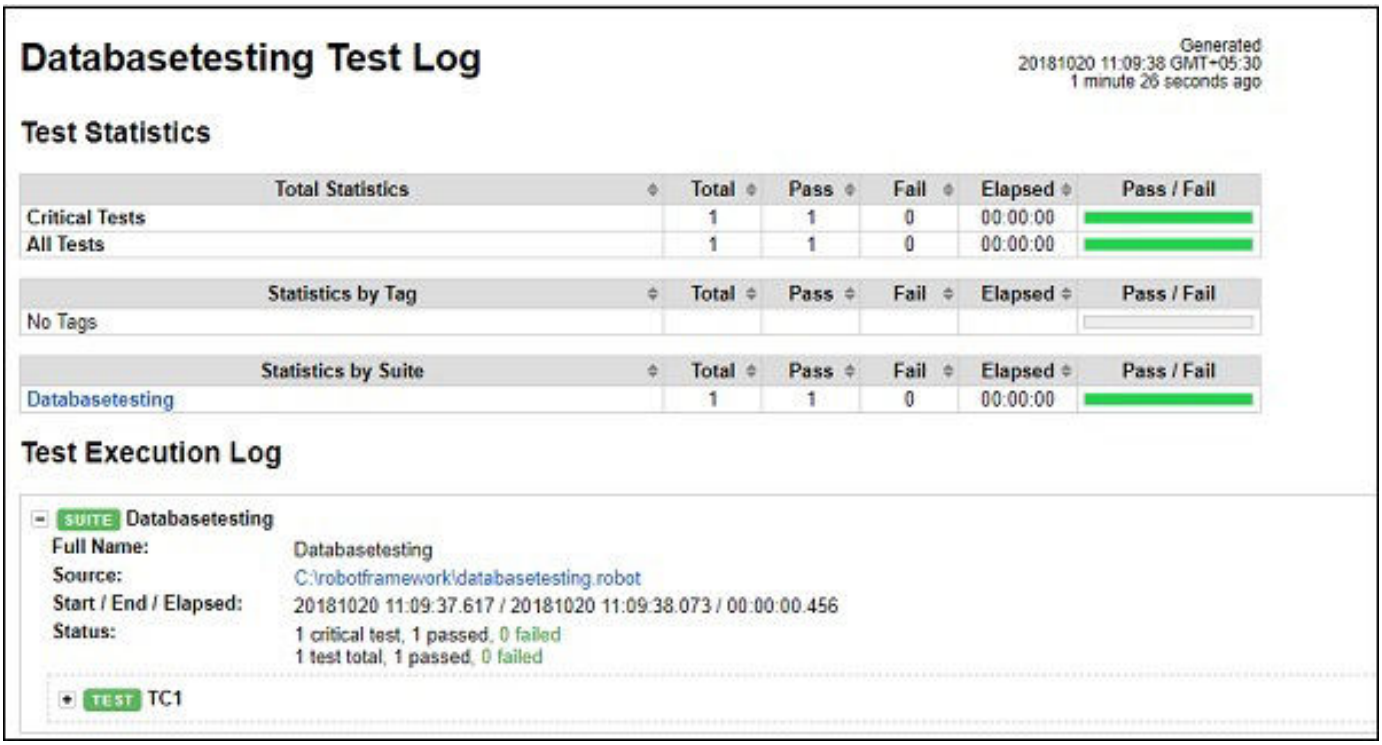
We have connected to the database, checked if table customer exists in database, got the query executed and logged the details of the query.

We will execute the test case and see the output

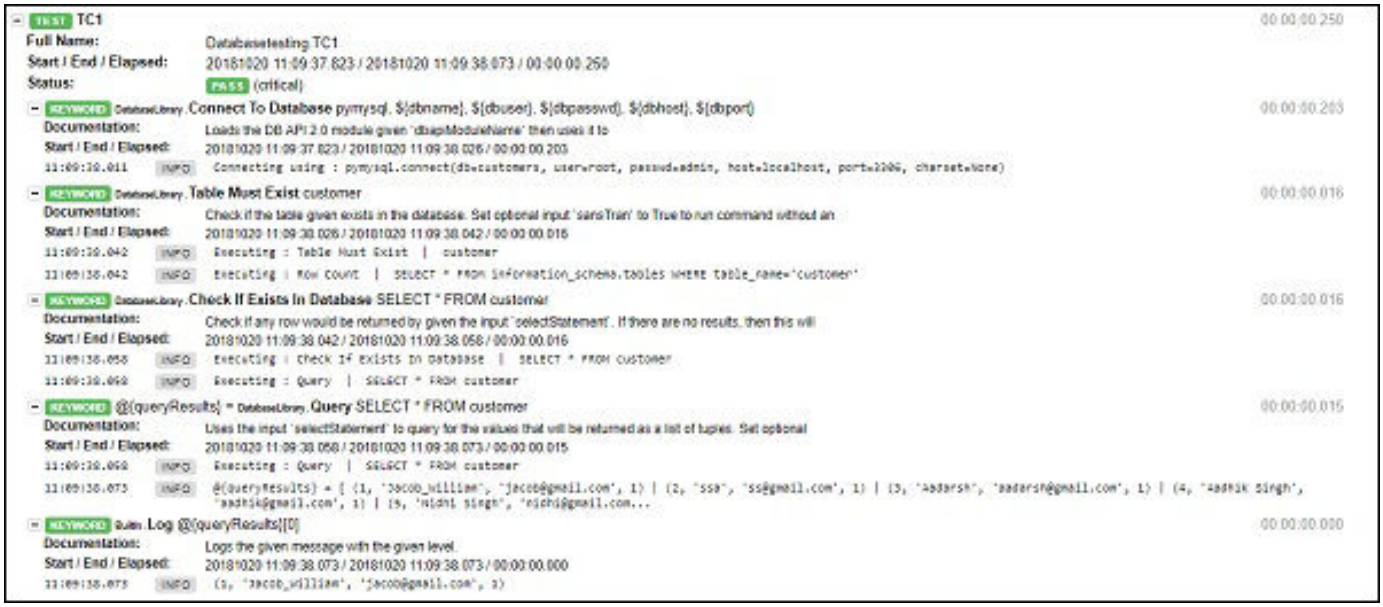


The results from the table are shown for the queryResults.

Log Details



Details of TC1



Conclusion

We have seen how to import database library, and the installation of it. We now know how to connect to MySQL database in Robot Framework and test the tables.

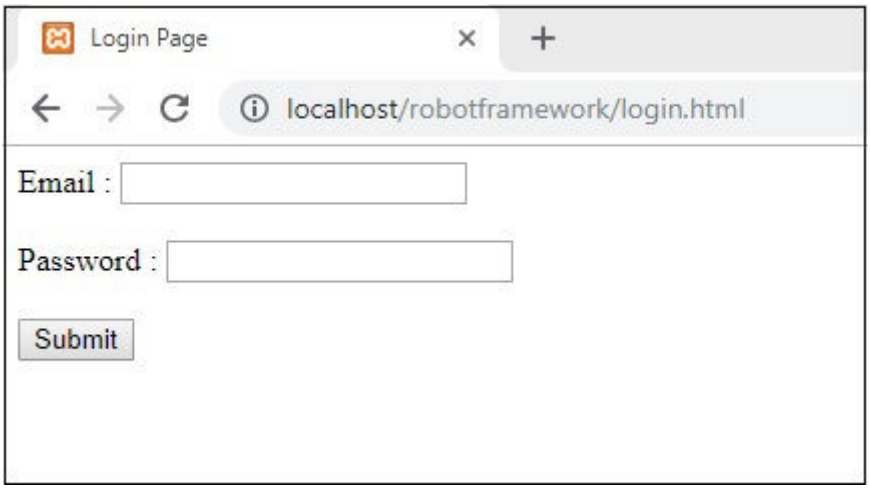
Testing Login Page Using Robot Framework

With Robot Framework, we have learnt the following so far −

- Import Libraries
- Working with variables
- Create custom Keywords
- How to write test-cases
- How to create Setup and teardown
- How to execute test cases
- How to work with data driven test-cases

We will use all of the above features and use it to test login page in this chapter. We have a login page which takes in email-id and password. When you enter correct email id and password, you will be redirected to a welcome page. If the user enters invalid email id or password, the page will get redirected to error page.

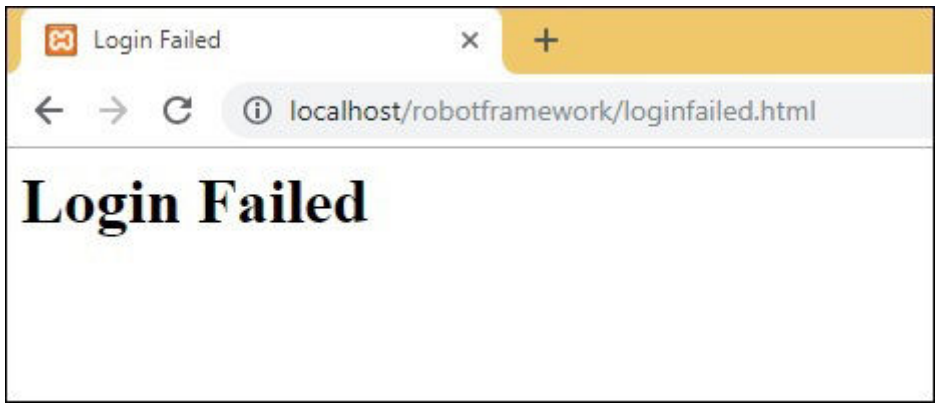
The following screenshot shows a login page −



HTML Code

```
<html>
   <head>
      <title>Login Page</title>
   </head>
   <body>
      <script type="text/javascript">
         function wsSubmit() {
            if (document.getElementById("email").value == "admin@gmail.com" && do
               location.href = "http://localhost/robotframework/success.html";
            } else {
               location.href = "http://localhost/robotframework/loginfailed.html"
            }
         }
      </script>
      <div id="formdet">
         Email : <input type="text" id="email" value="" id="email" /><br/><br/>
         Password : <input type="password" id="passwd" value="" /><br/><br/>
         <input type="submit" id="btnsubmit" value="Submit" onClick="wsSubmit();"
      </div>

   </body>
</html>
```

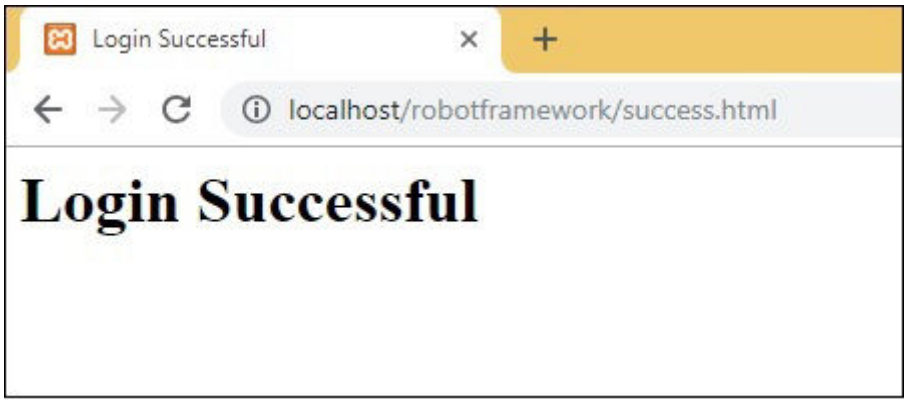
The following screen appears when either the email-id or the password is not valid −



HTML Code

```
<html>
   <head>
      <title>Login Failed</title>
   </head>
   <body>
      <div id="loginfailed">
         <h1>Login Failed</h1>
      </div>
   </body>
</html>
```

The following screen appears when both the email id and password are valid −



HTML Code

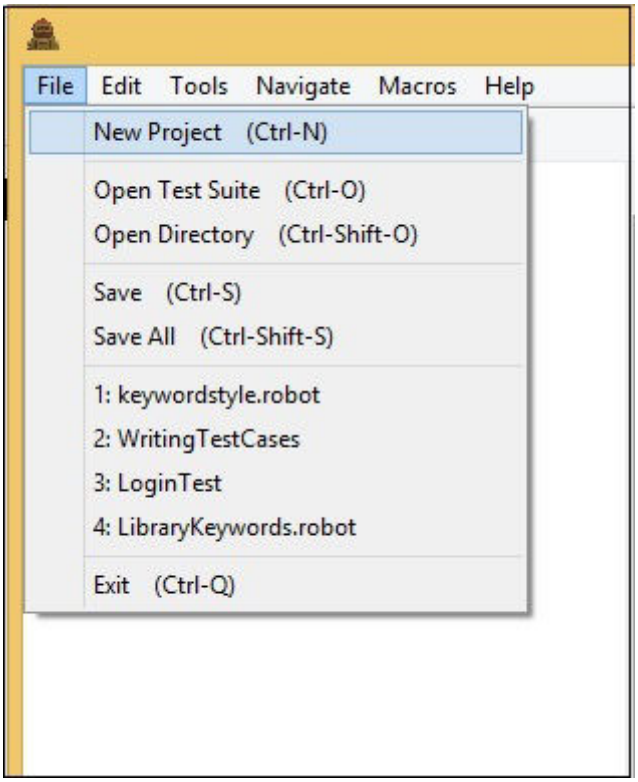
```
<html>
   <head>
      <title>Login Successful</title>
   </head>
   <body>
      <div id="loginfailed">
         <h1>Login Successful</h1>
      </div>
   </body>
</html>
```

Now we are going to write test cases for the above test page. To start with it, we will first run the command to open Ride.

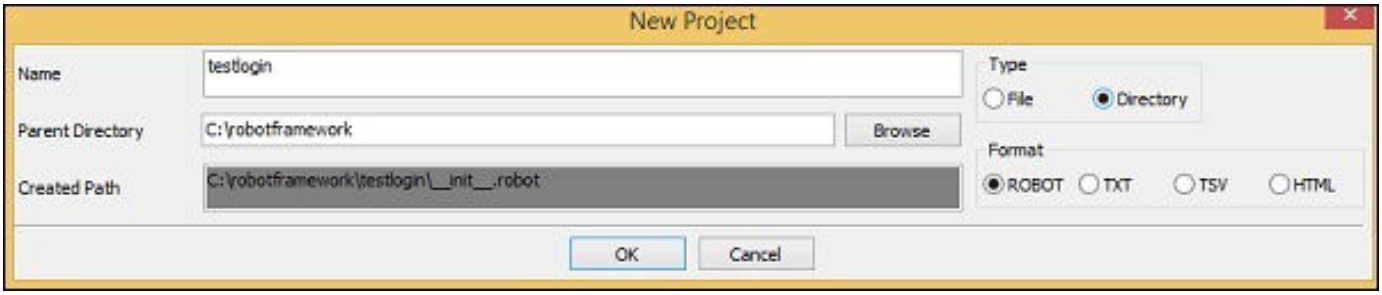
Command

```
ride.py
```

Once done, we will get started with the project setup as shown below −



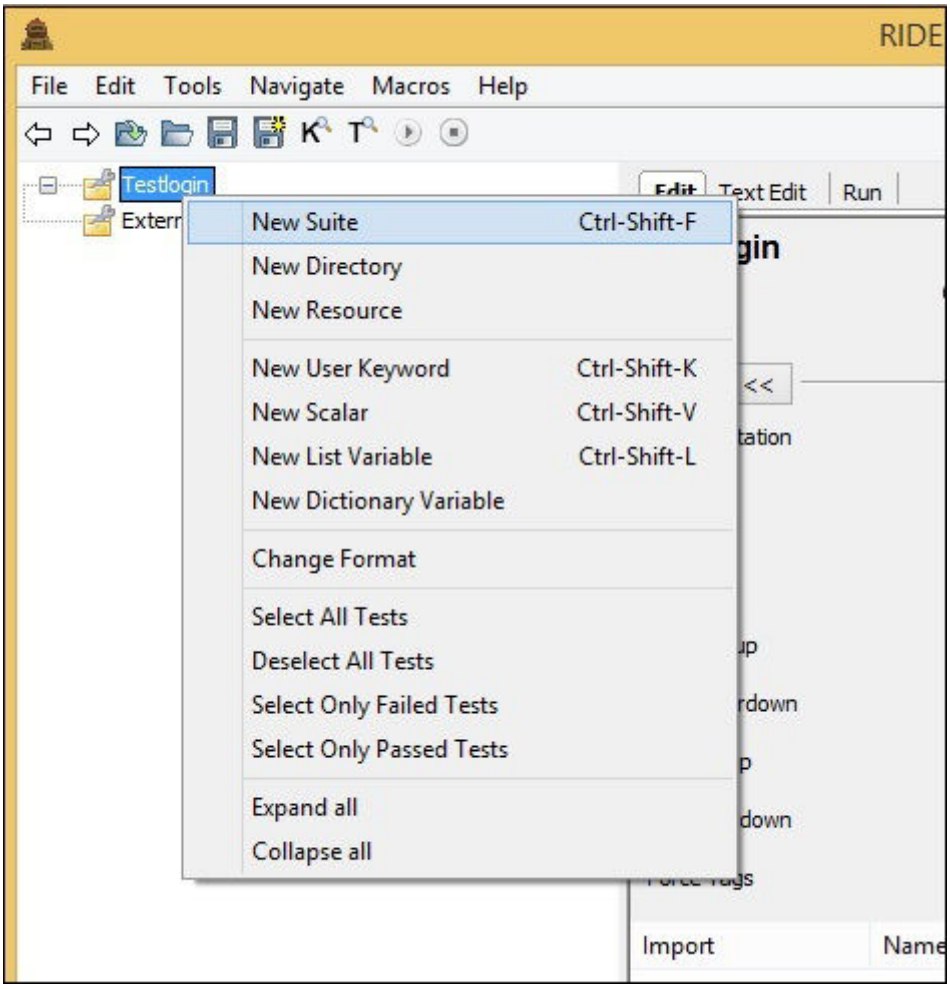
Click New Project and enter the name of the project.



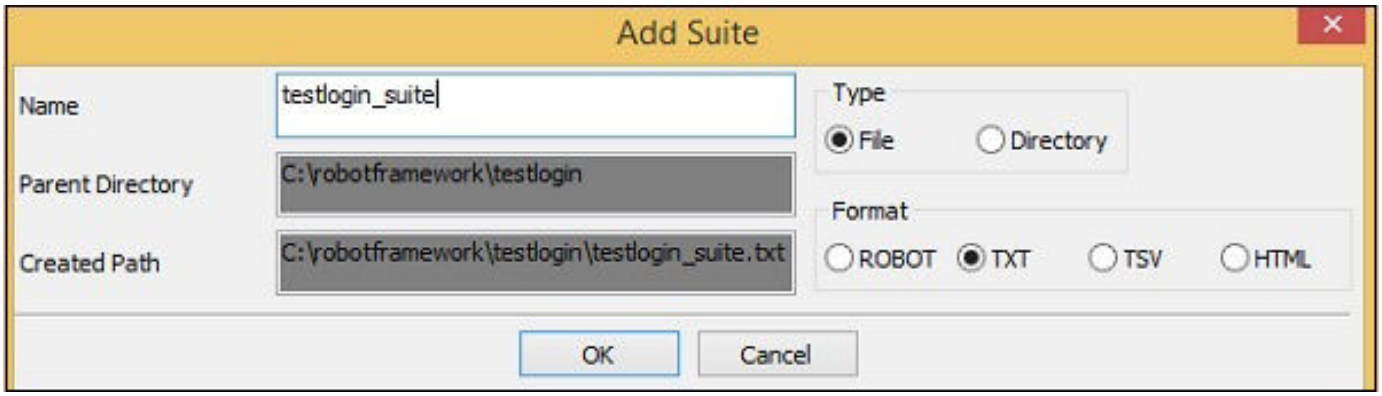
We will save the type of the project as Directory. The name given to the project is testlogin.

Click OK to save the project.

Now, we will create test suite inside the project.

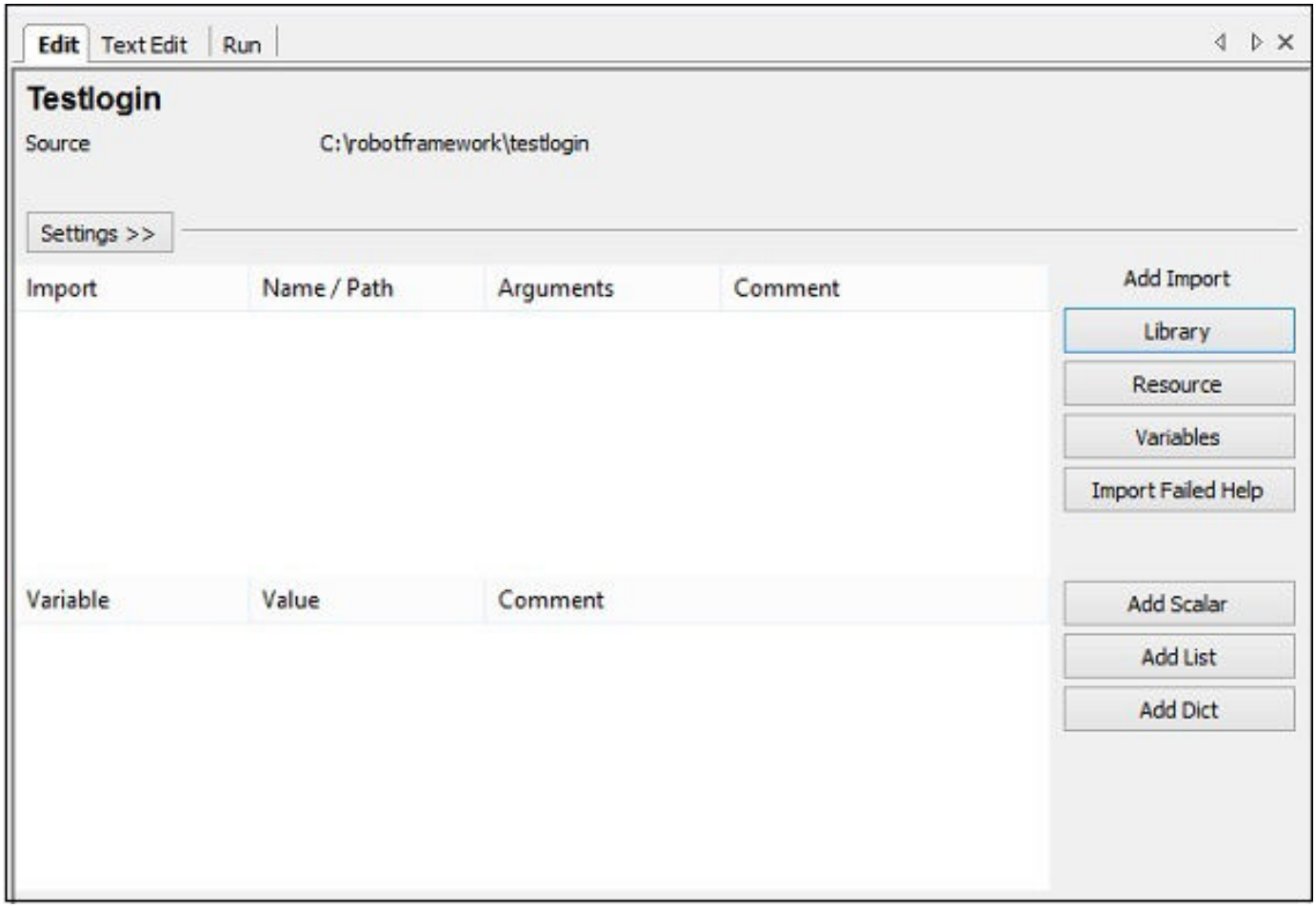


Click New Suite and it will display a screen as shown below −

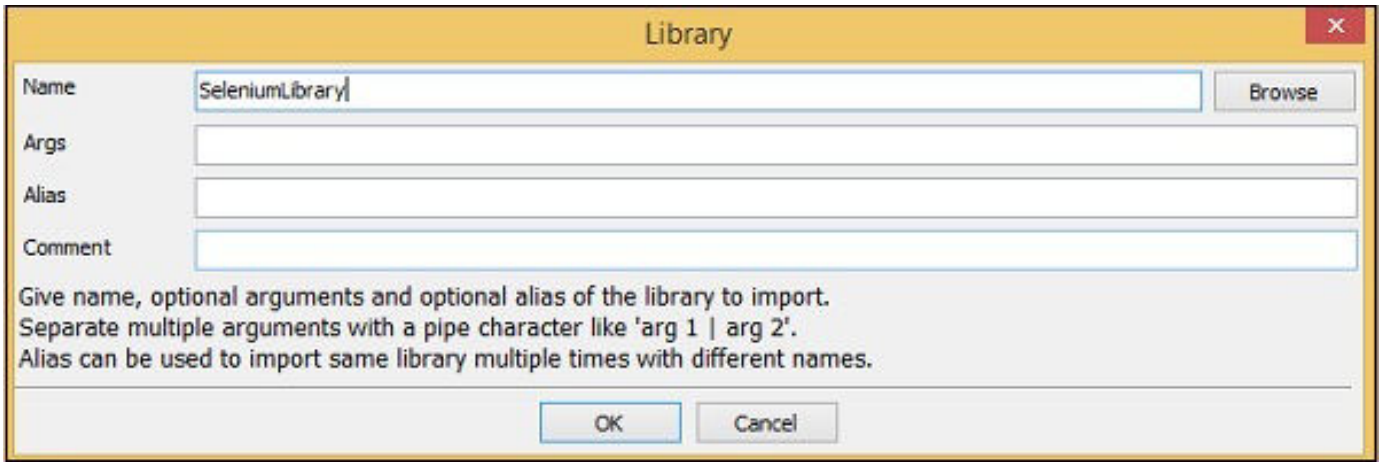


Click OK to save the test suite. We need to import the Selenium Library since we will be working with the browser.

Import Library in the main project and also to the test suite created.

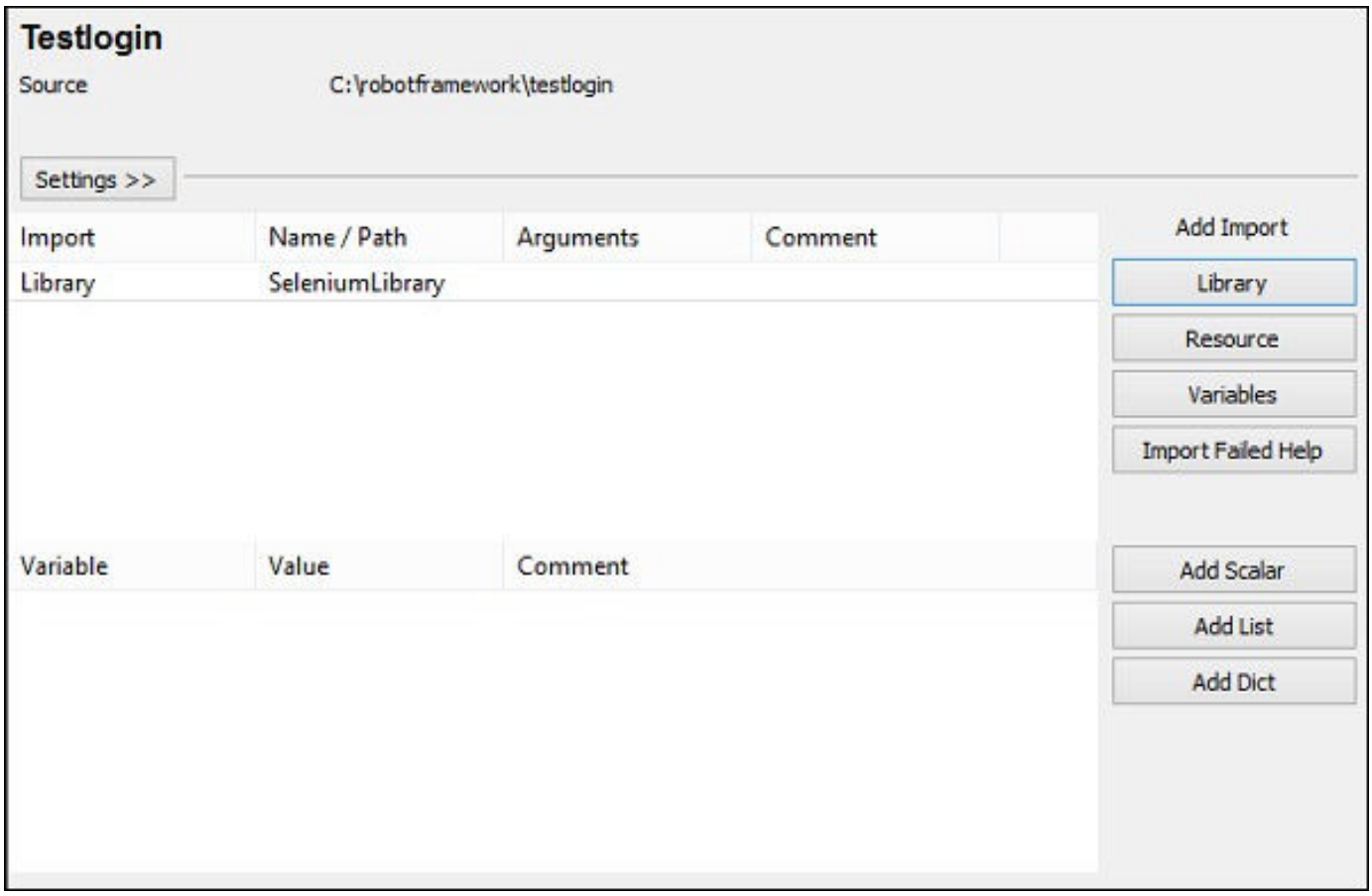


Click Library as in the above screenshot. Upon clicking Library, the following screen will appear.



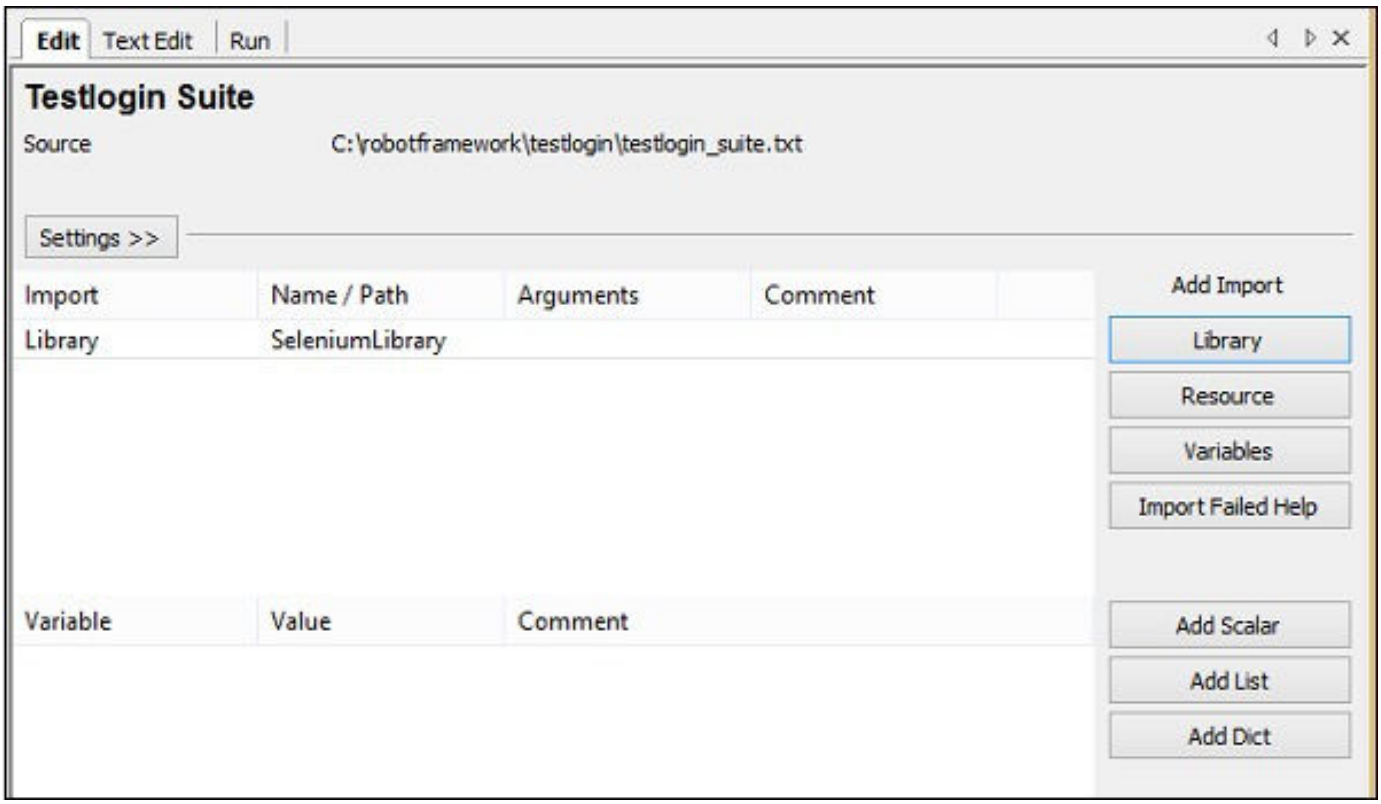
Click OK to save the library for the project.

Once the library is saved for the project, it will display the library in the settings −



Repeat the same step for the Test suite created.

Here is the library added for Test suite −



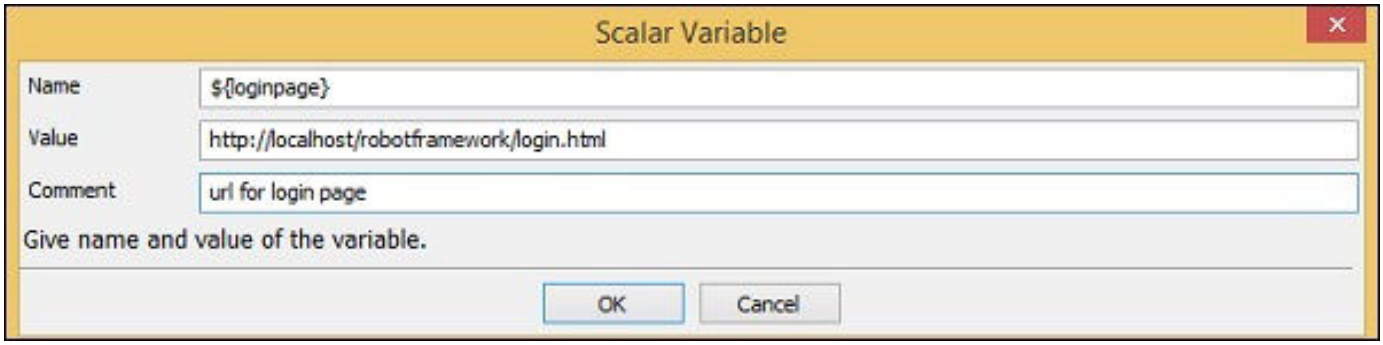
Now in the main Project, we will create a setup and teardown. We would like to open the login page in Chrome browser and maximize the window. In teardown, we will close the browser.

For setup, we will create a user−defined keyword called **Open Login Page**. This keyword will take 2 arguments, login page URL and browser name.

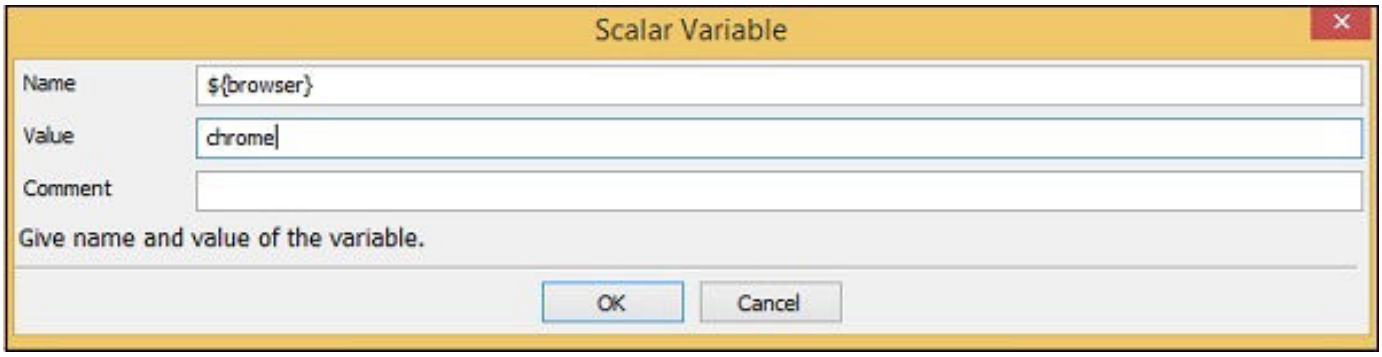
Now, we need 2 scalar variables that will help us store the values – url and the browser name.

In ride, create 2 variables **${loginpage}** and **${browser}** as follows −

${loginpage}

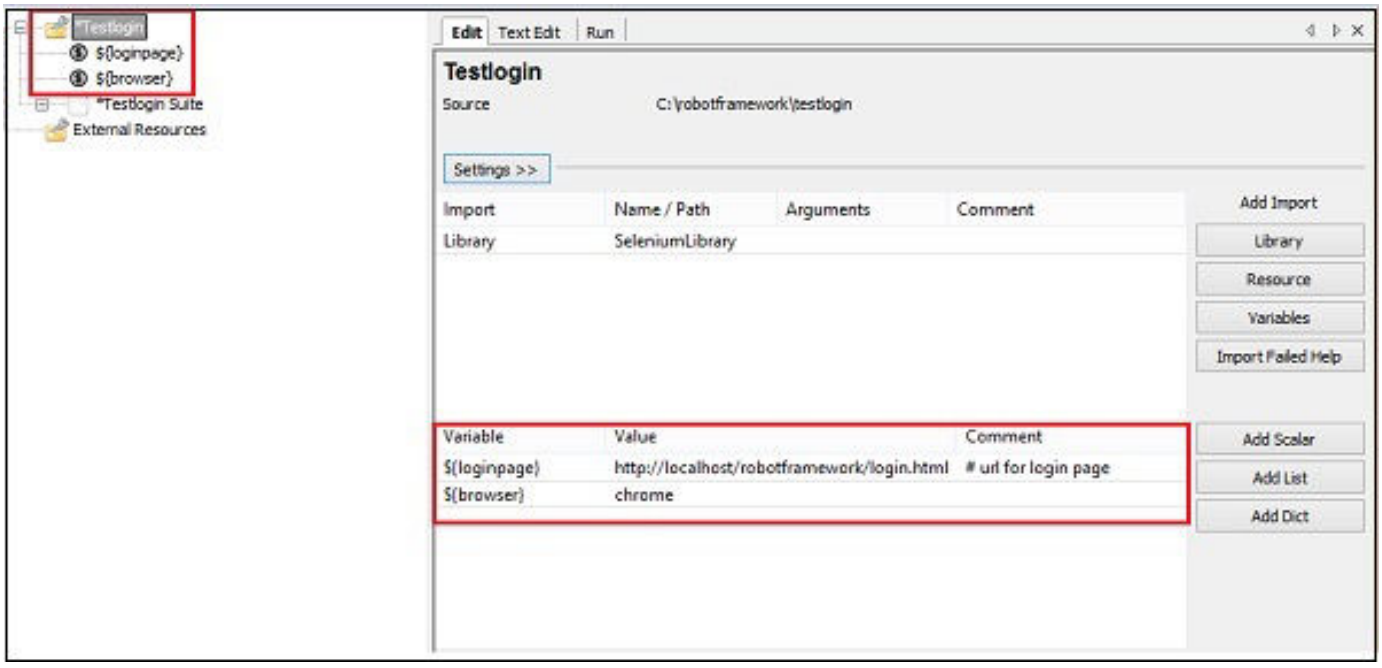


${browser}



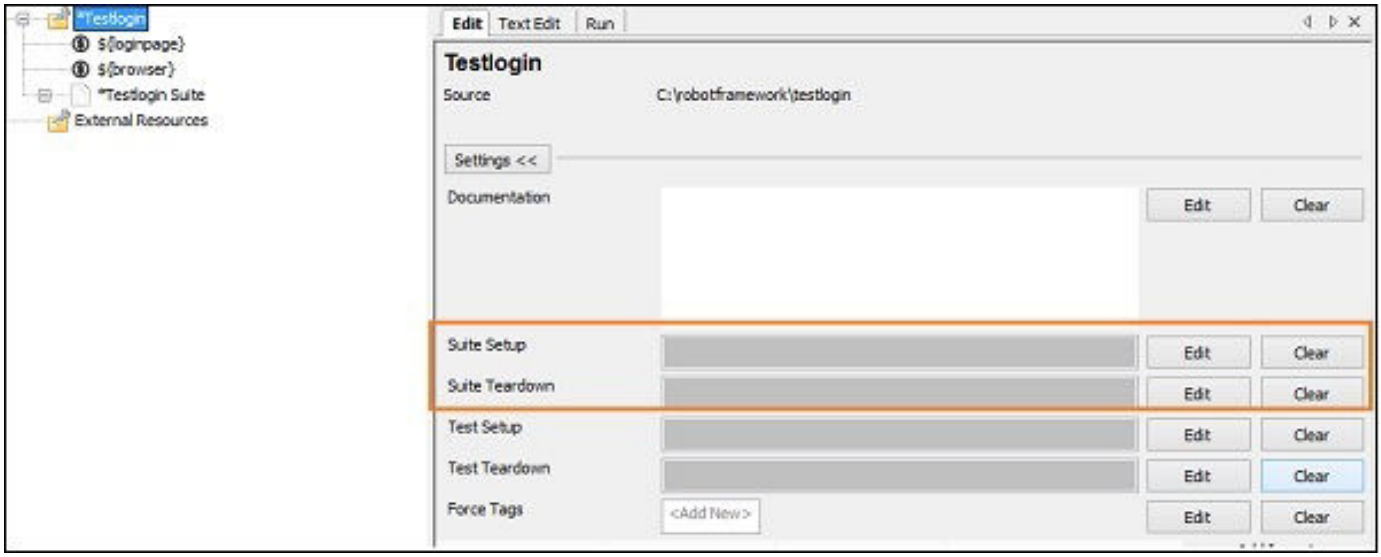
Save both variables.

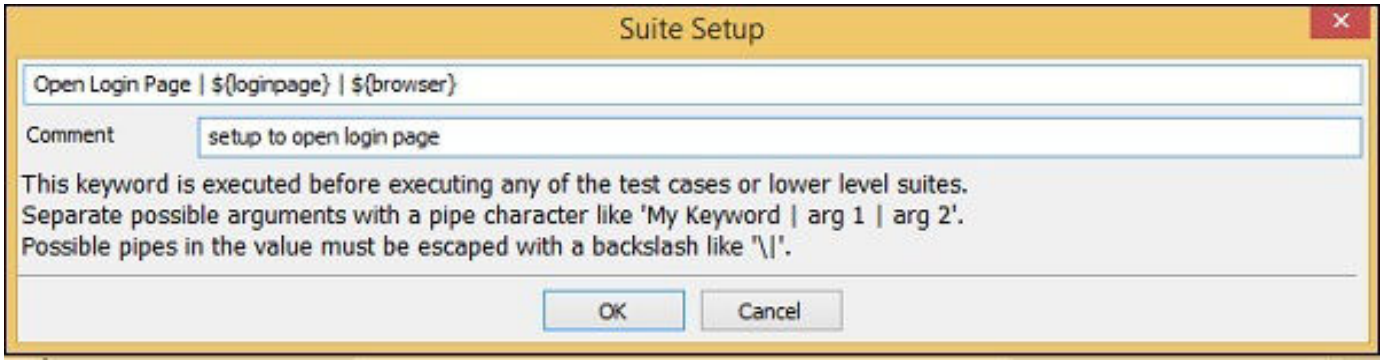
The variables will be displayed under your project as follows −



Now, we will add the setup and teardown for the main project.

Click on the project on the left side. In settings, click Suite Setup.



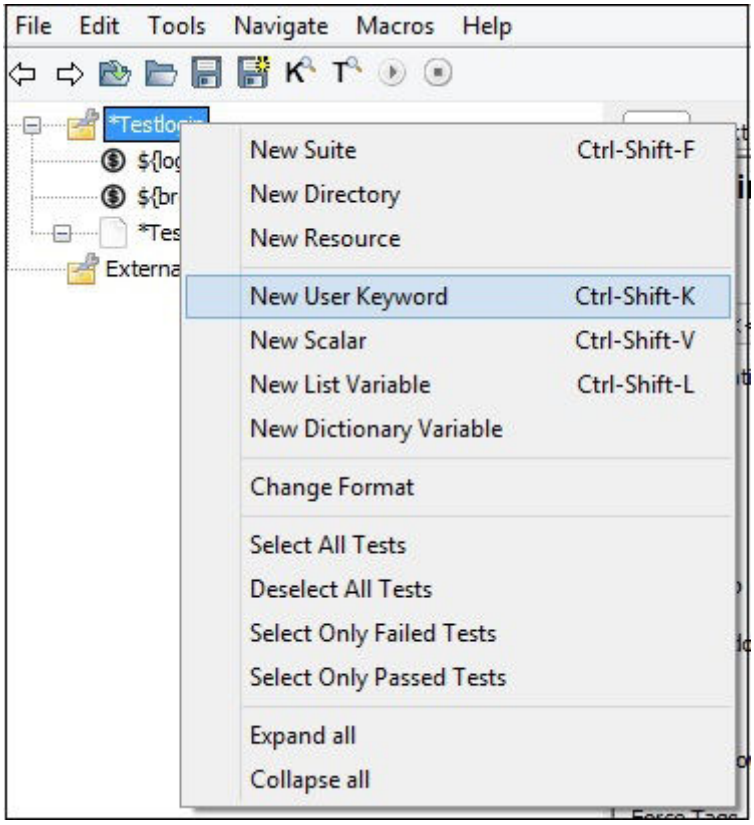


We have created setup that is using user keyword **Open Login Page** with arguments **${loginpage}** and **${browser}**.

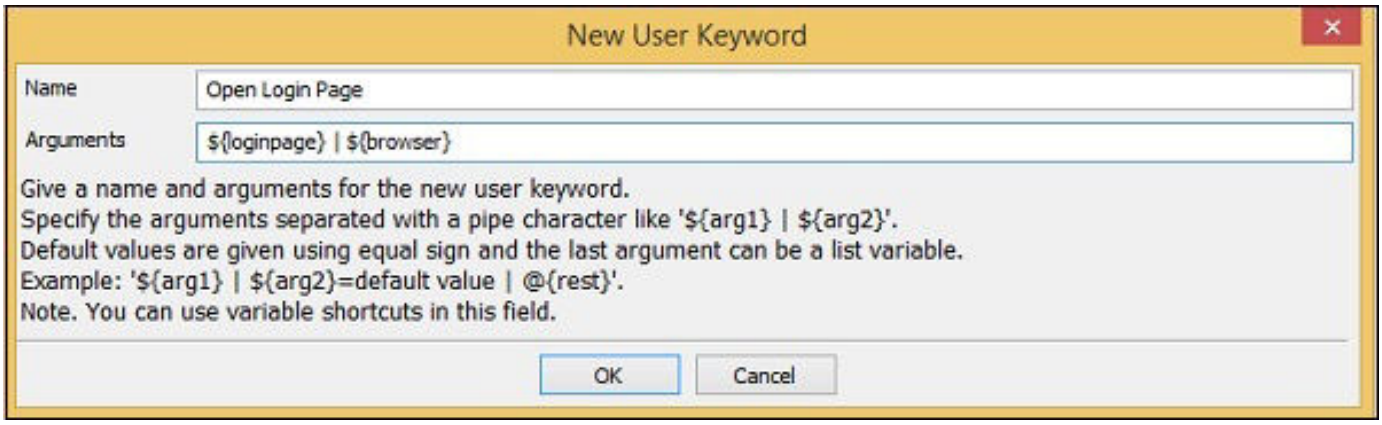
Click OK to save the setup.

Now, we have to create the user−defined keyword **Open Login Page**, which is done as follows −

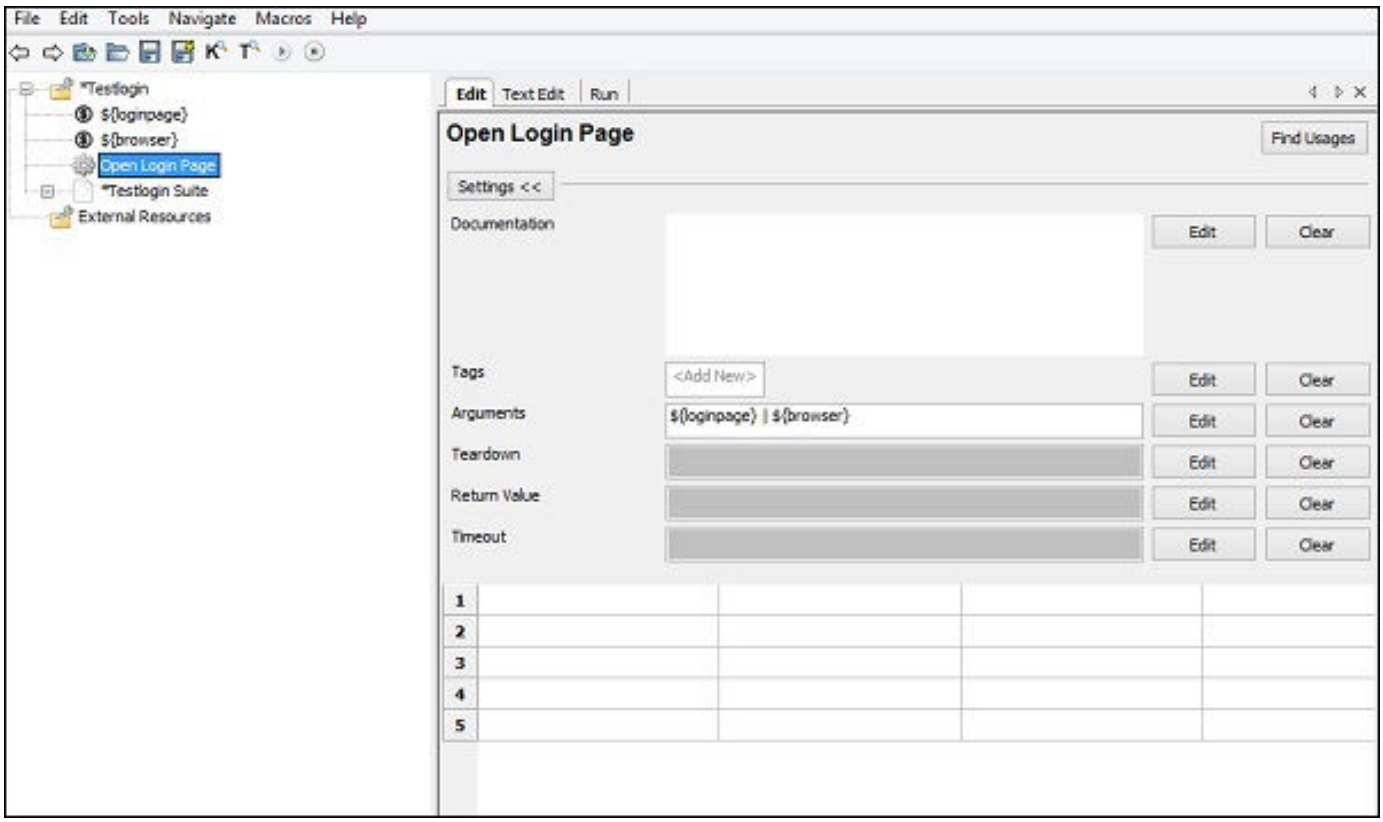
Right−click on the project and click **New User Keyword** −



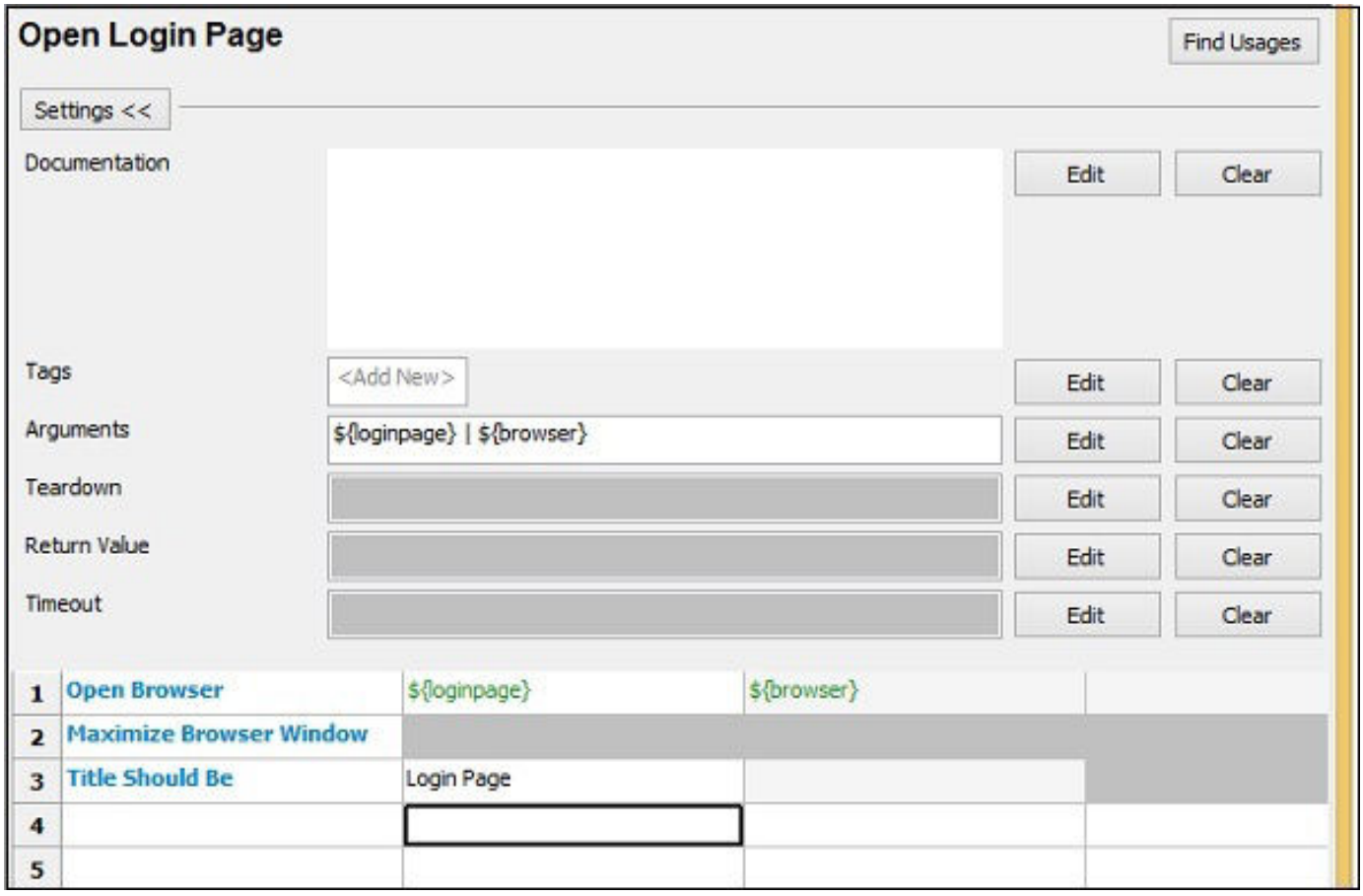
Upon clicking **New User Keyword**, the following screen appears −



Here the Keyword is given 2 arguments − **${loginpage}** and **${browser}**. Click OK to save the user keyword.



Now we need to enter the library keywords, which will open the URL.

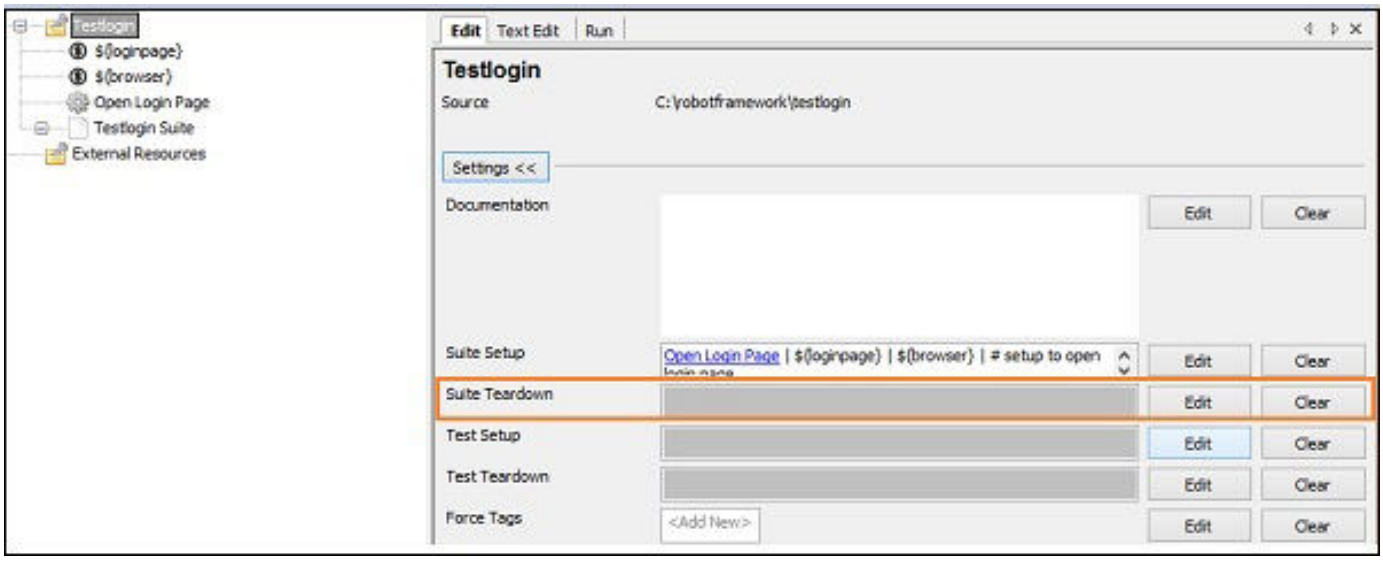


Open Login Page user−defined keyword has the following details −

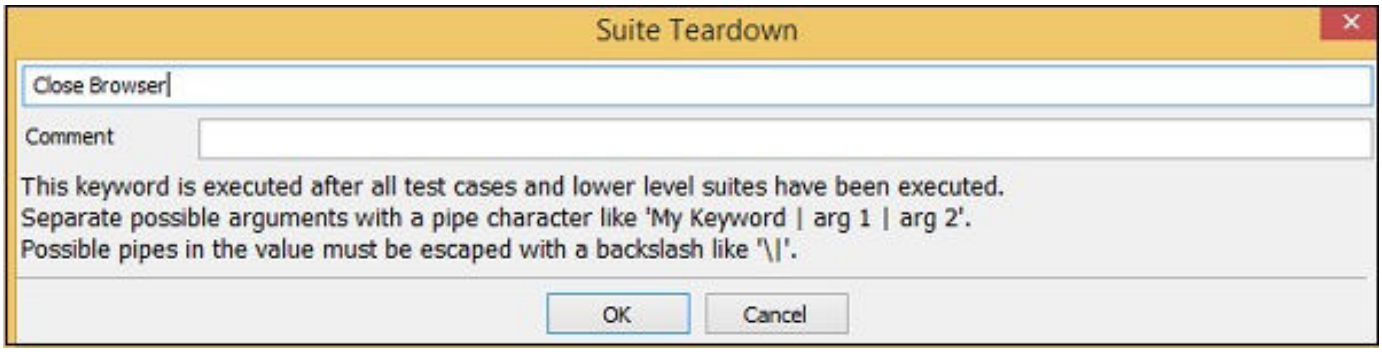
```
*** Keywords ***
Open Login Page
```

```
[Arguments] ${loginpage} ${browser}
Open Browser ${loginpage} ${browser}
Maximize Browser Window
Title Should Be Login Page
```

Now, we will create **Suite Teardown** for the suite.

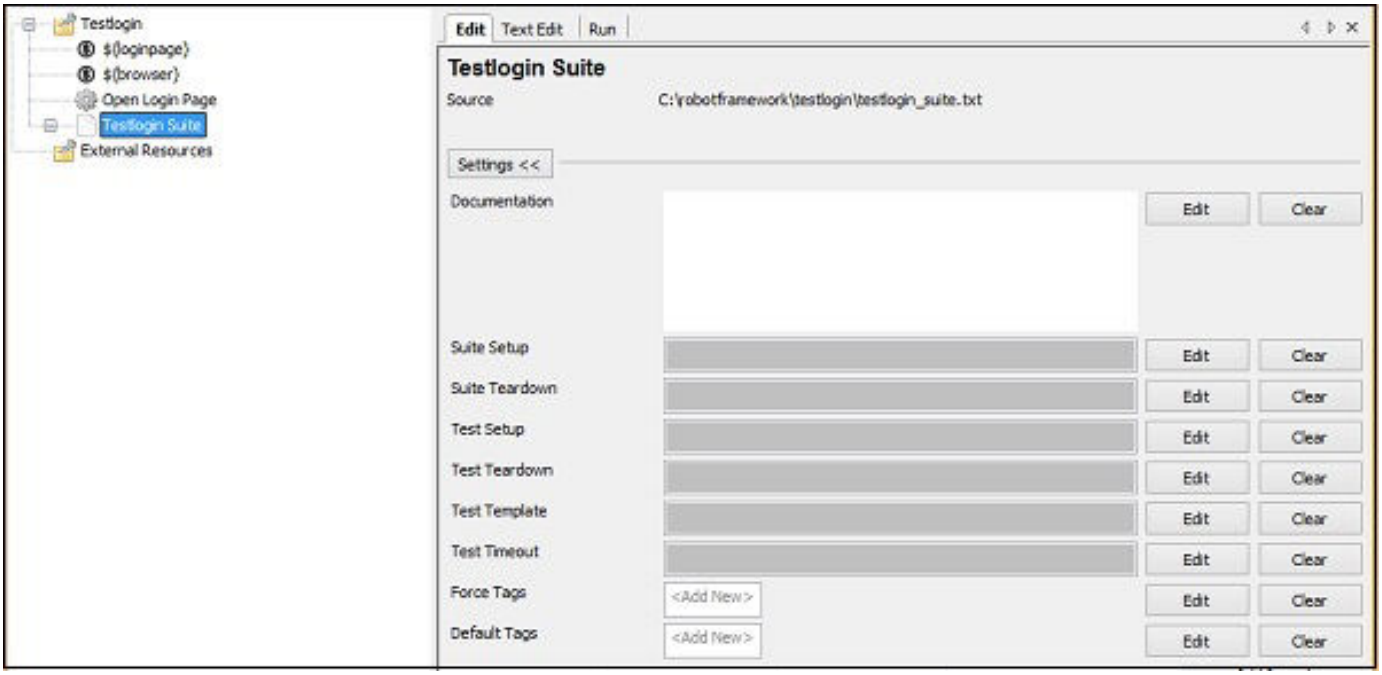


Click Edit for Suite Teardown and enter the details −



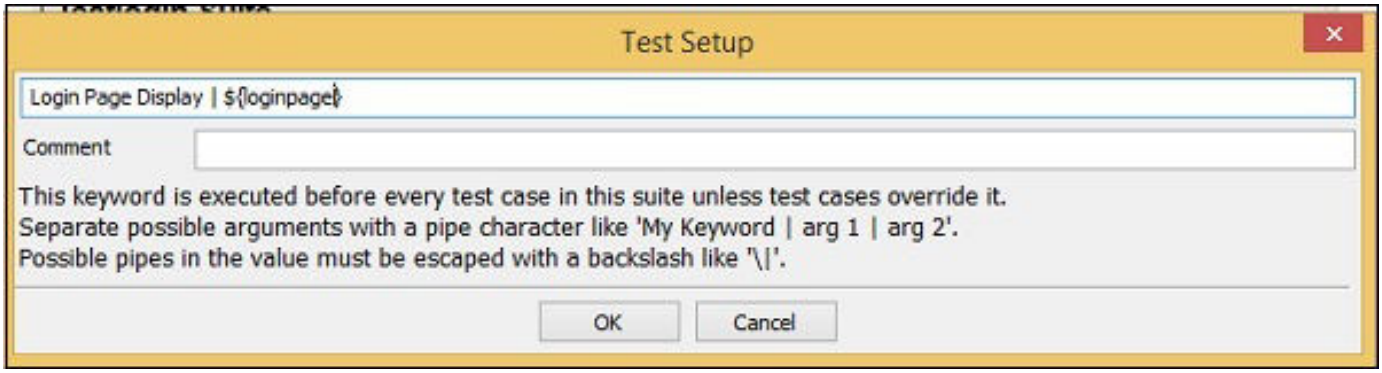
For Suite teardown, we are directly using library keyword, which will close the browser. Click OK to save the suite teardown.

Now, click the Testlogin Suite we have created.



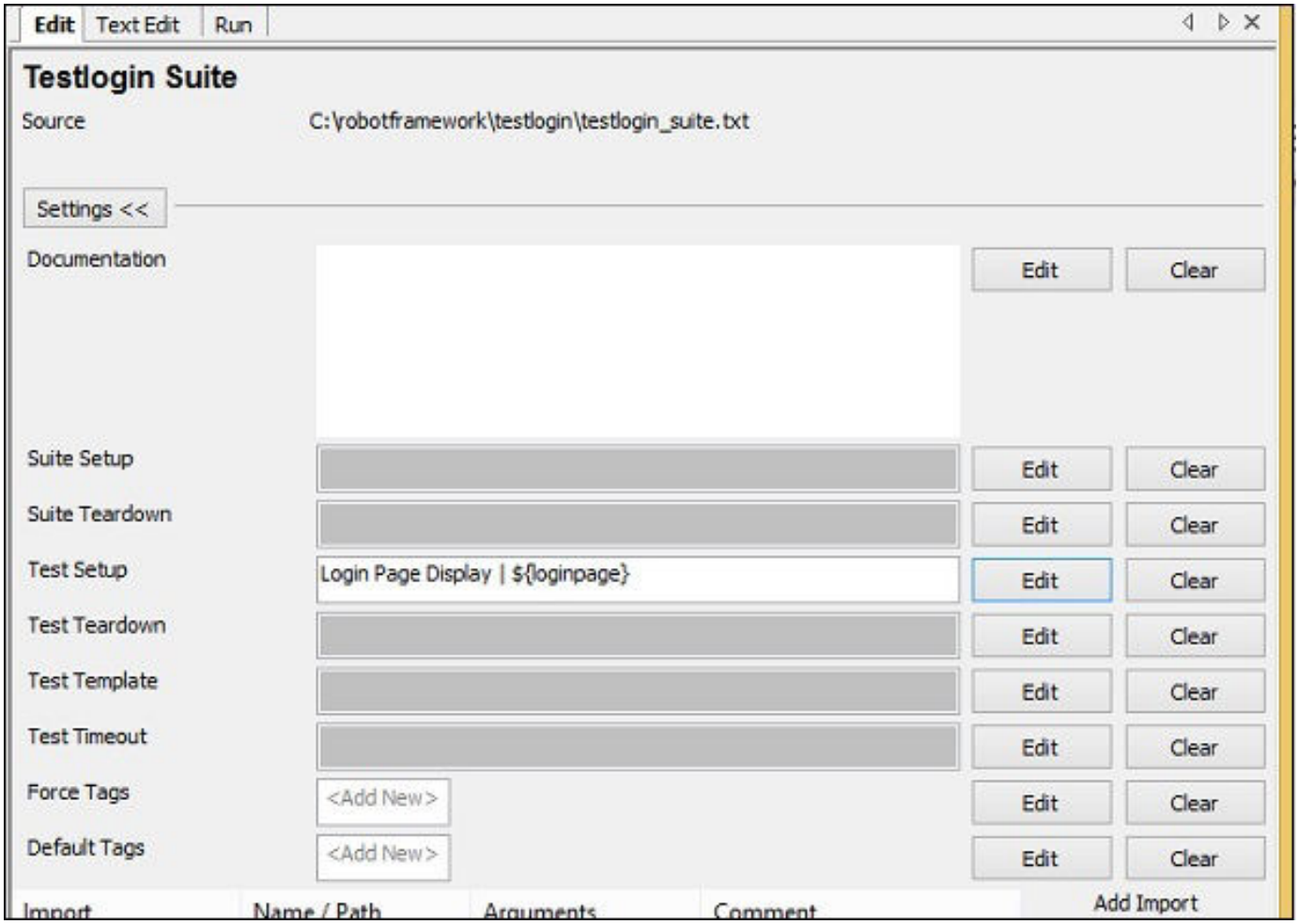
Let us now create a setup for the test suite – Test Setup. This setup needs to get executed first.

Click Edit for Test Setup and enter the details.



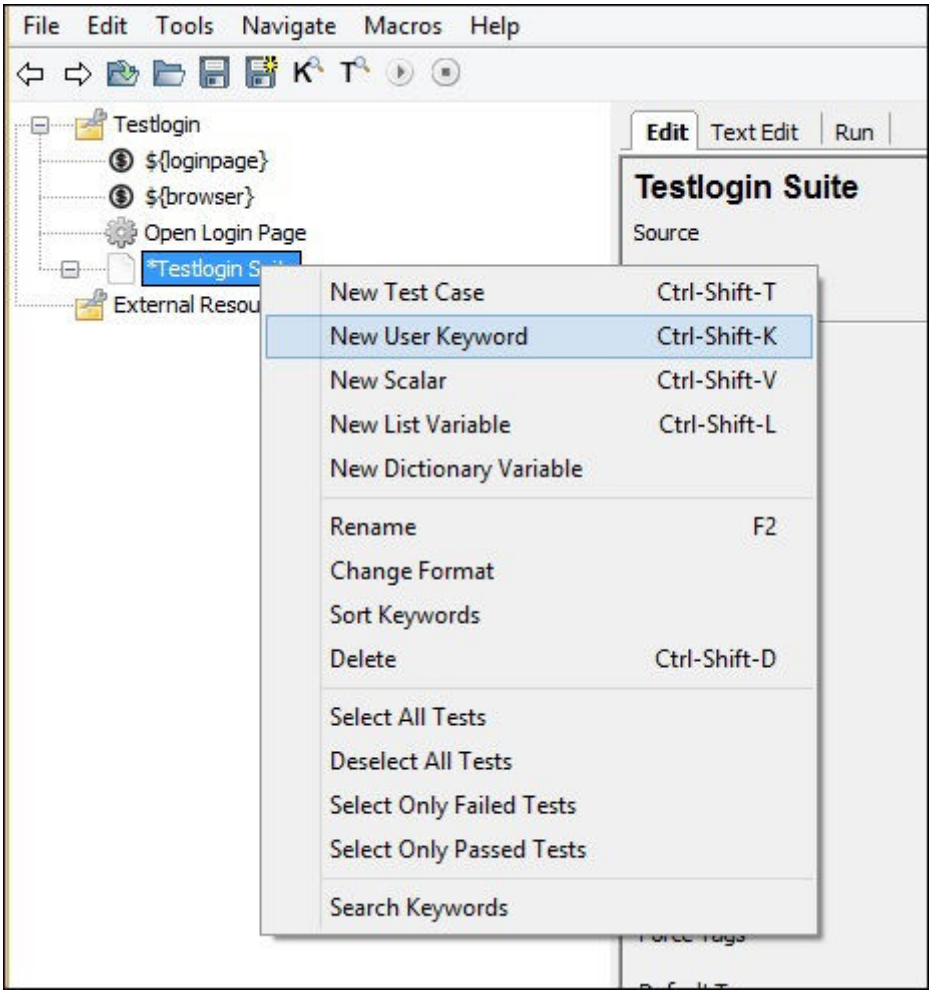
For the Test Setup, we have created User defined Keyword called **Login Page Display**, which will take the argument as **${loginpage}** as in the above screenshot.

Click OK to save the test setup.

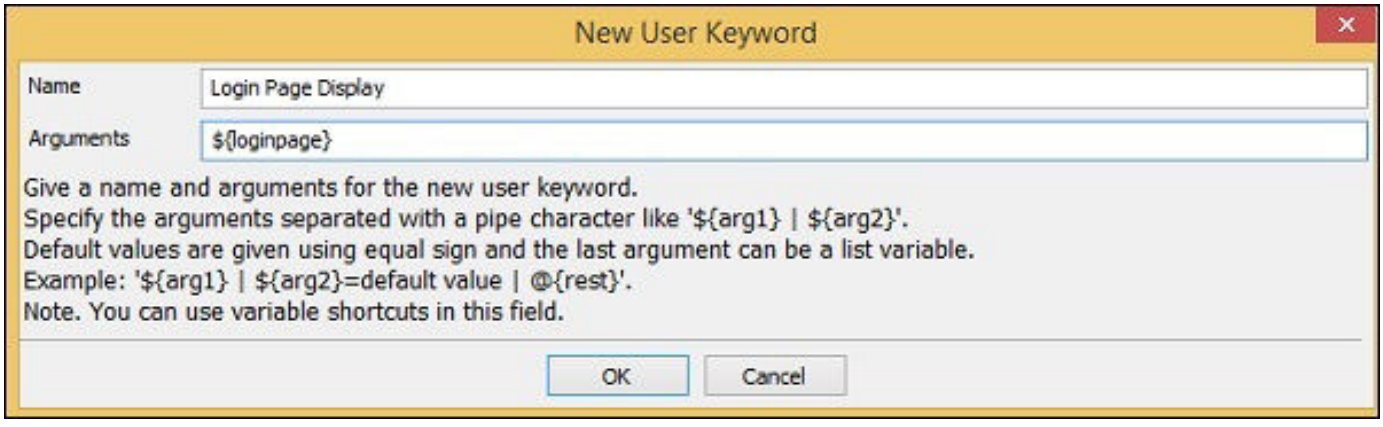


Now, we need to create the user keyword **Login Page Display**.

Right-click on the test suite and click **New User Keyword** as shown below −

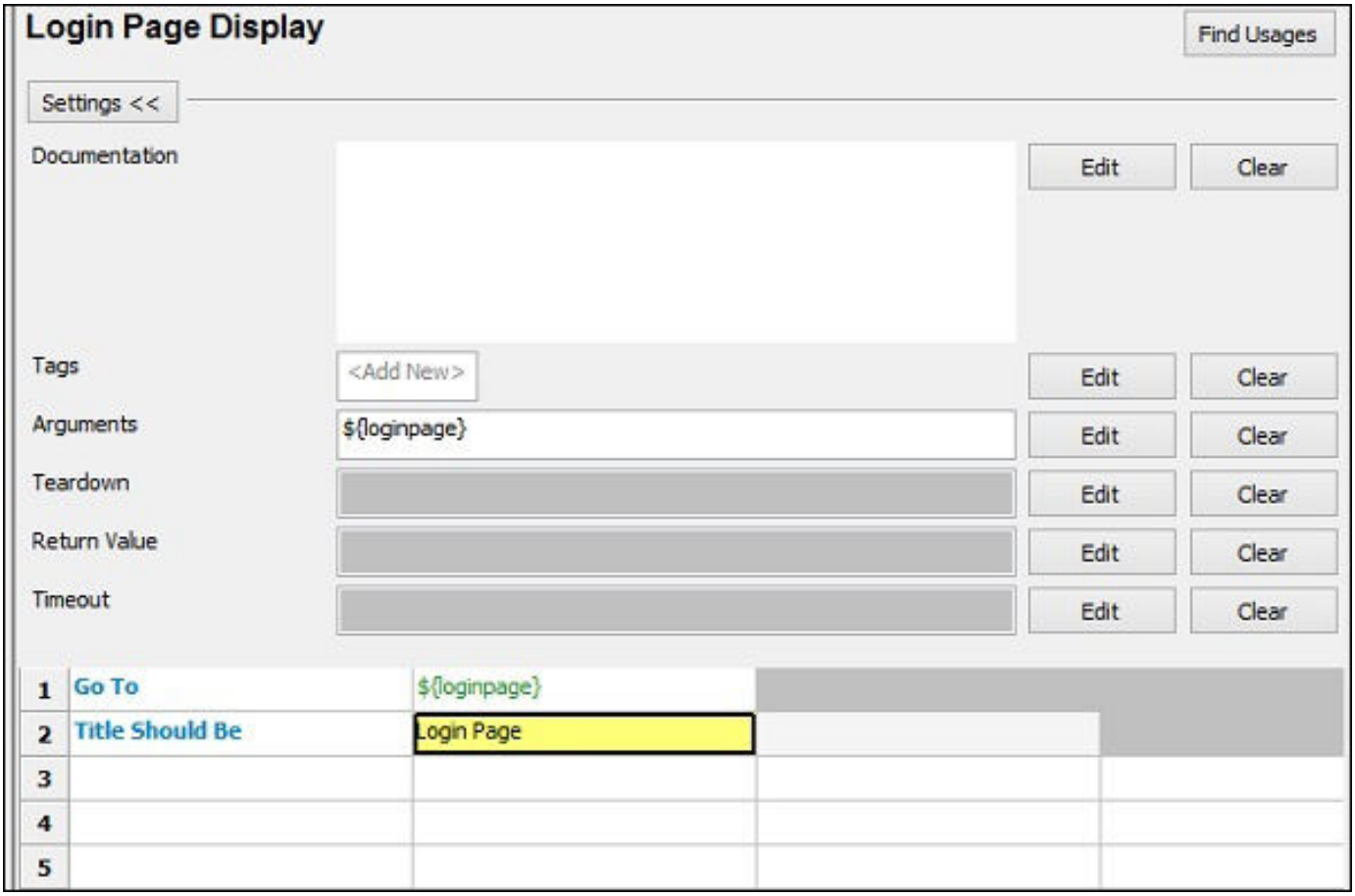


New User Keyword will display the screen as shown below −



Click OK to save the keyword.

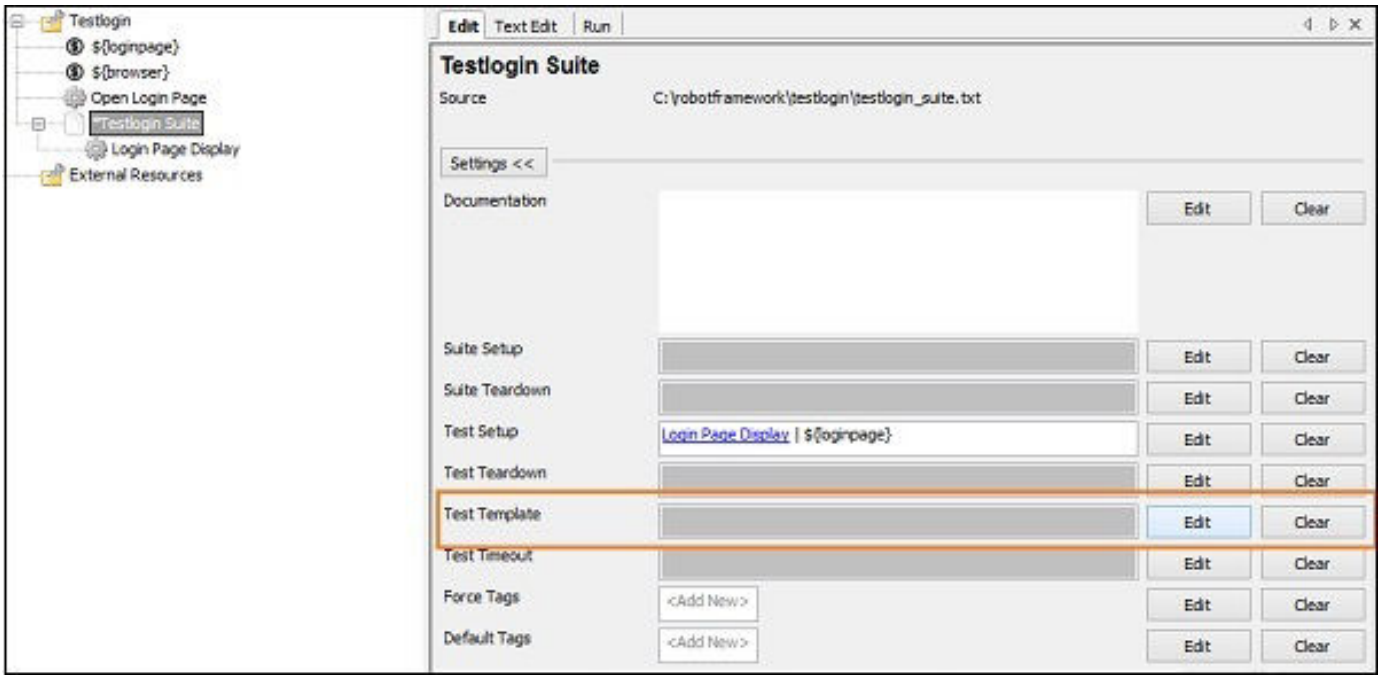
Let us now enter the keyword we need for the user keyword **Login Page Display**.



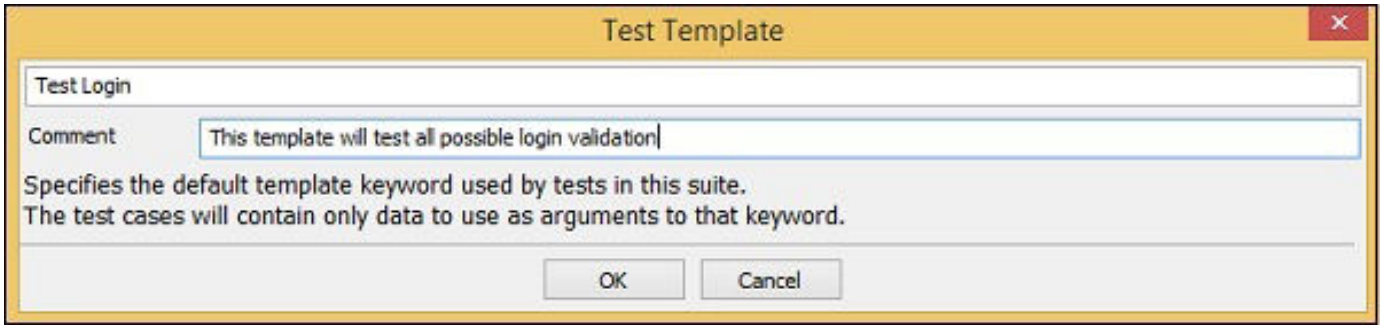
Here we want to go to the **loginpage** and check if the title of the page matches with the value given.

Now, we will add template to the test suite and create data driven test cases.

To create template, click on the suite and on right side click Edit for Test Template.



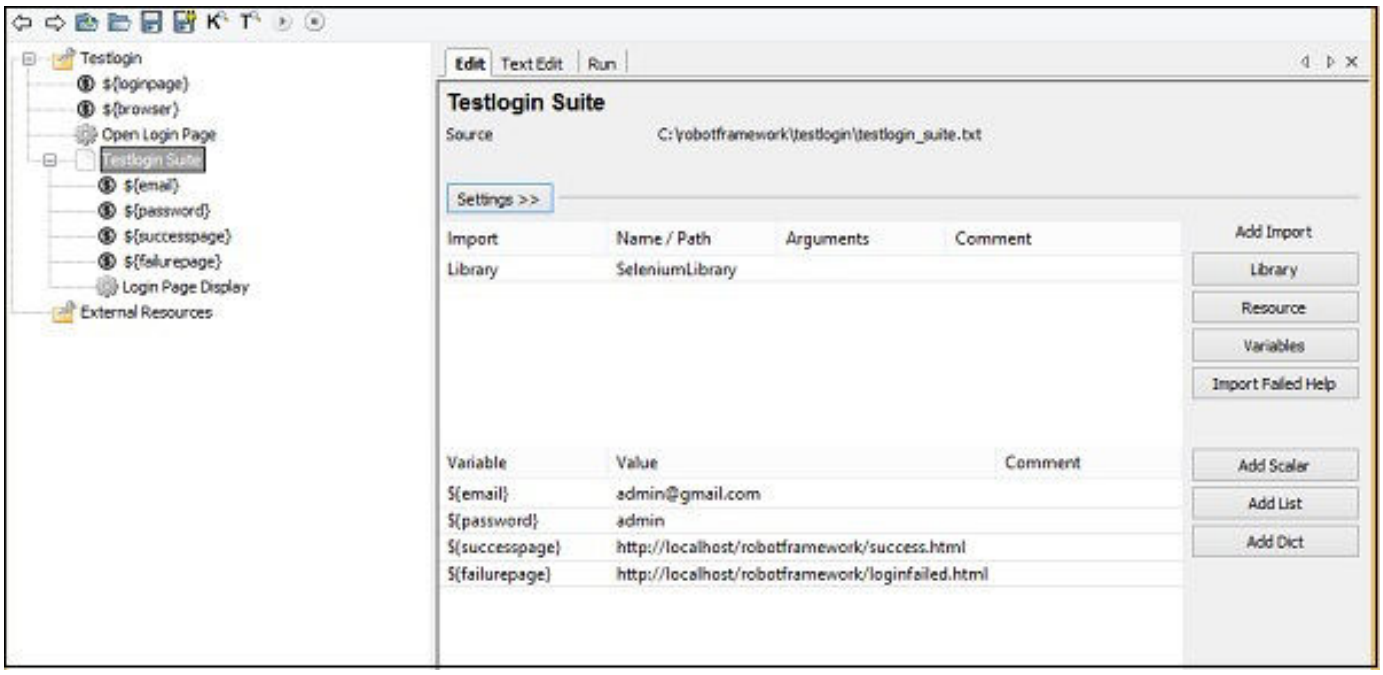
You will be directed to the following screen −



Test Login is again a user-defined keyword. Click OK to save the template.

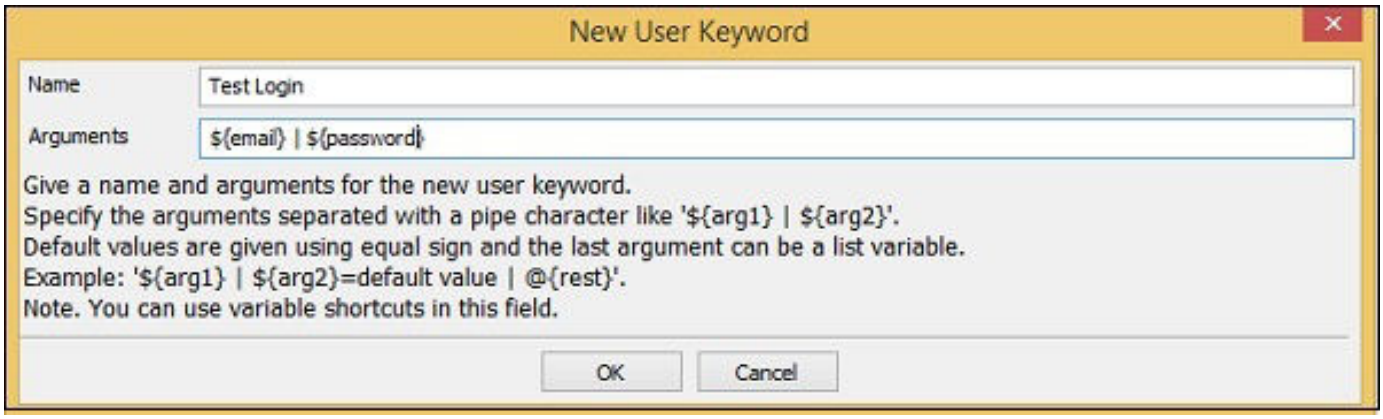
Before we create the Test Login keyword, we need some scalar variables. The scalar variables will have the details of the email-id, password, successpage, failurepage, etc.

We will create scalar variables for test suite as follows −



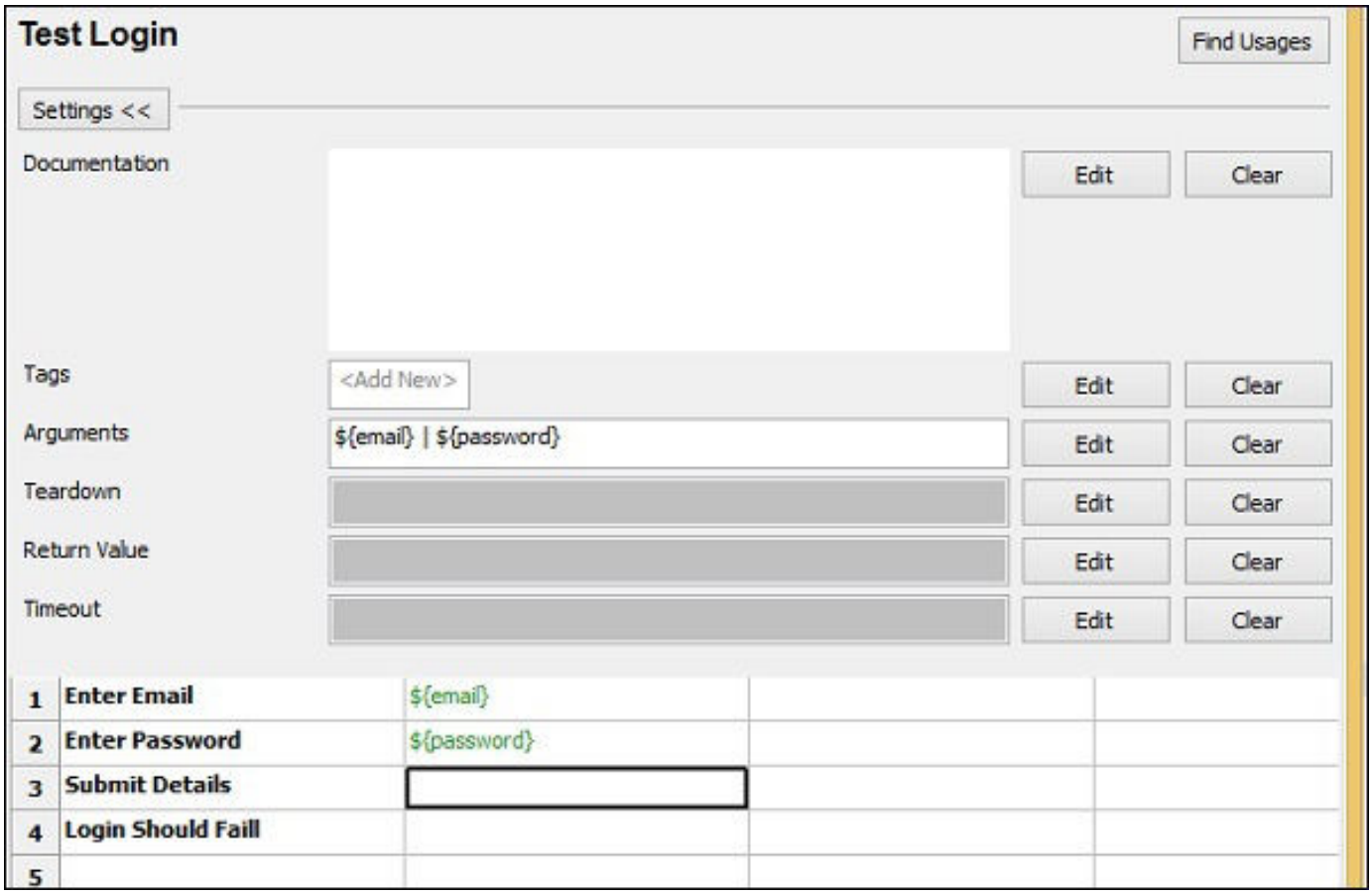
We have created email, password, successpage and failurepage scalar variables as shown in the above screenshot.

Now, we will create **Test Login** User defined Keyword. Right-click on the test suite and click on New User Keyword.

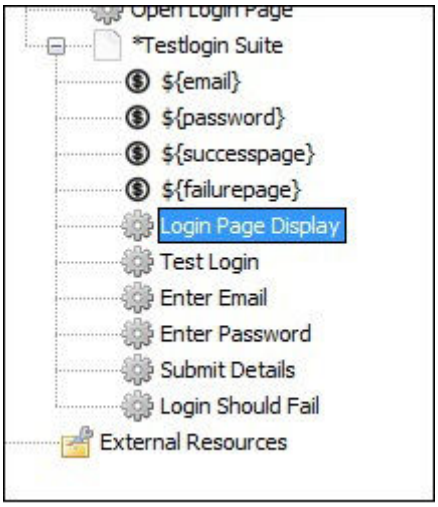


Click OK to save the keyword.

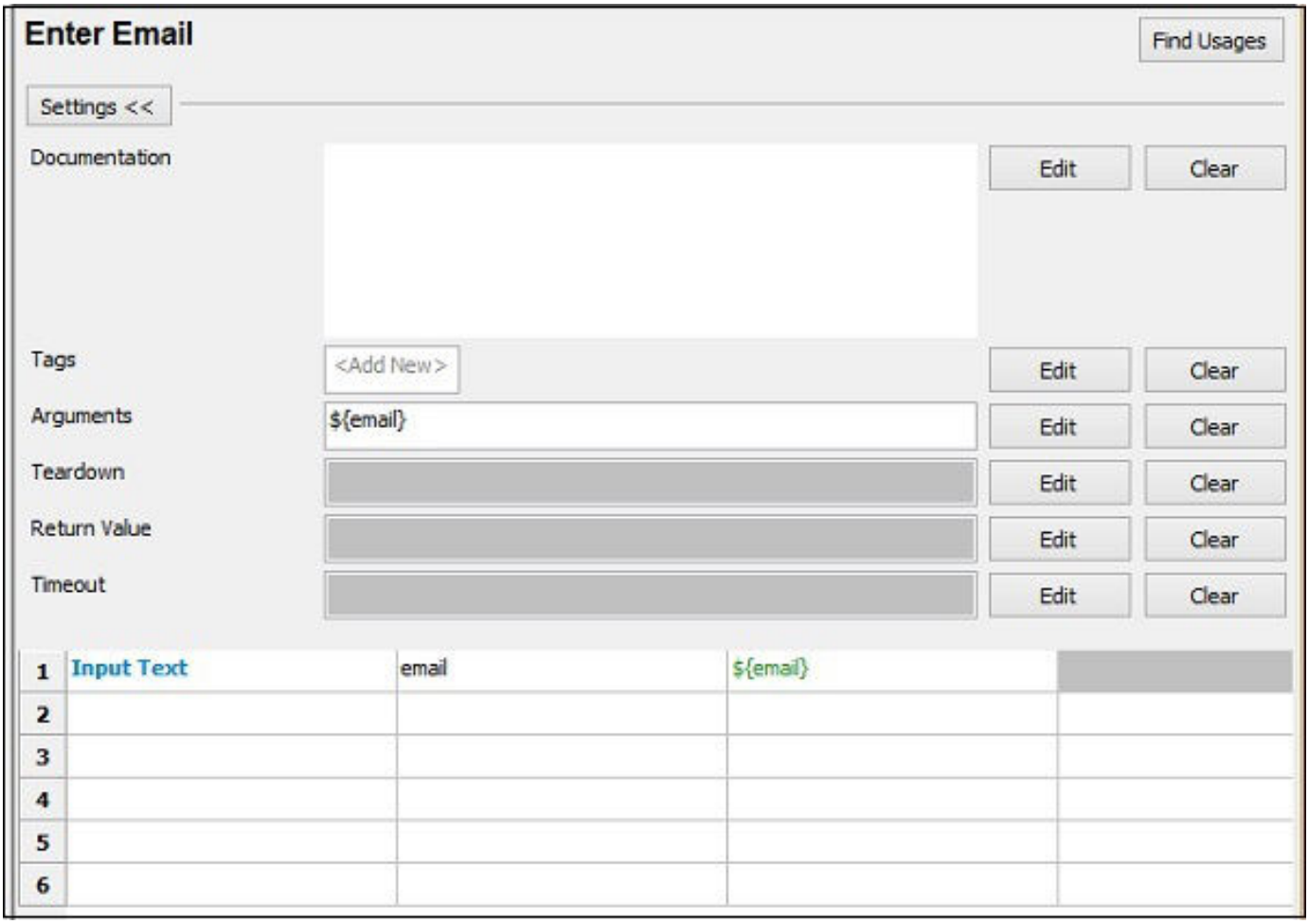
The following screenshot shows the keywords entered for Test Login −



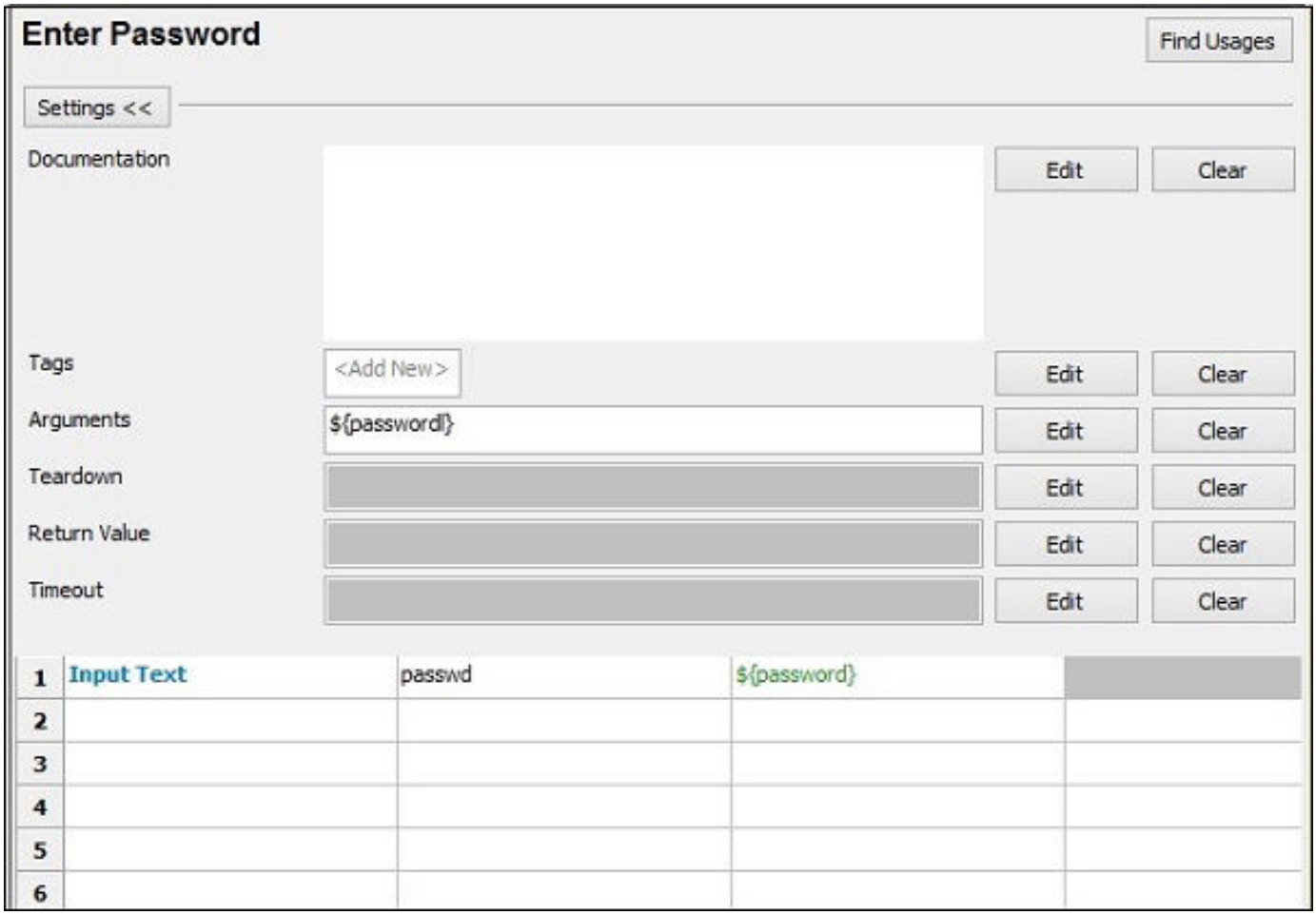
Enter Email, Enter Password, Submit Details and **Login Should Fail** are User Defined Keywords, which are defined as follows −



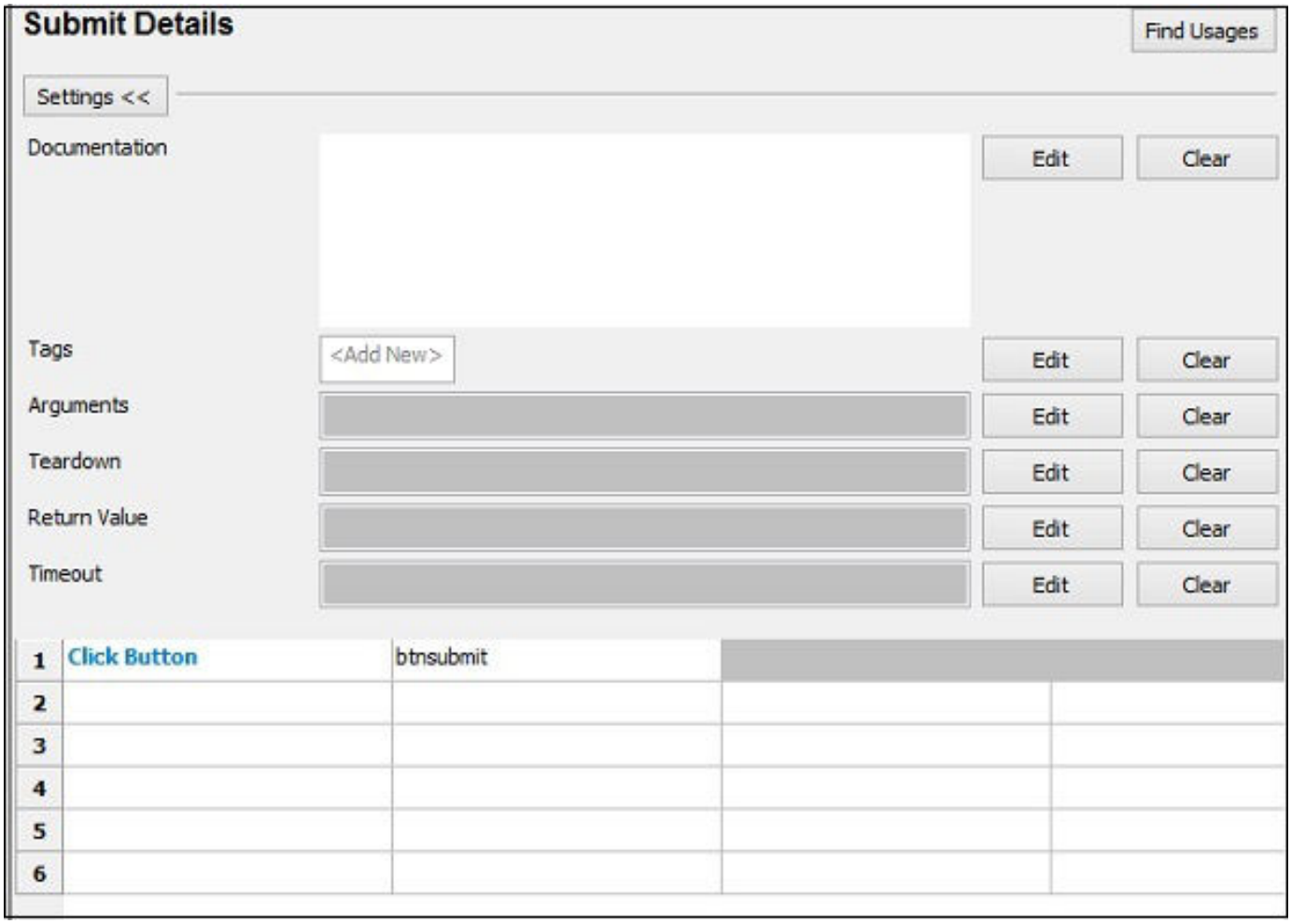
Enter Email



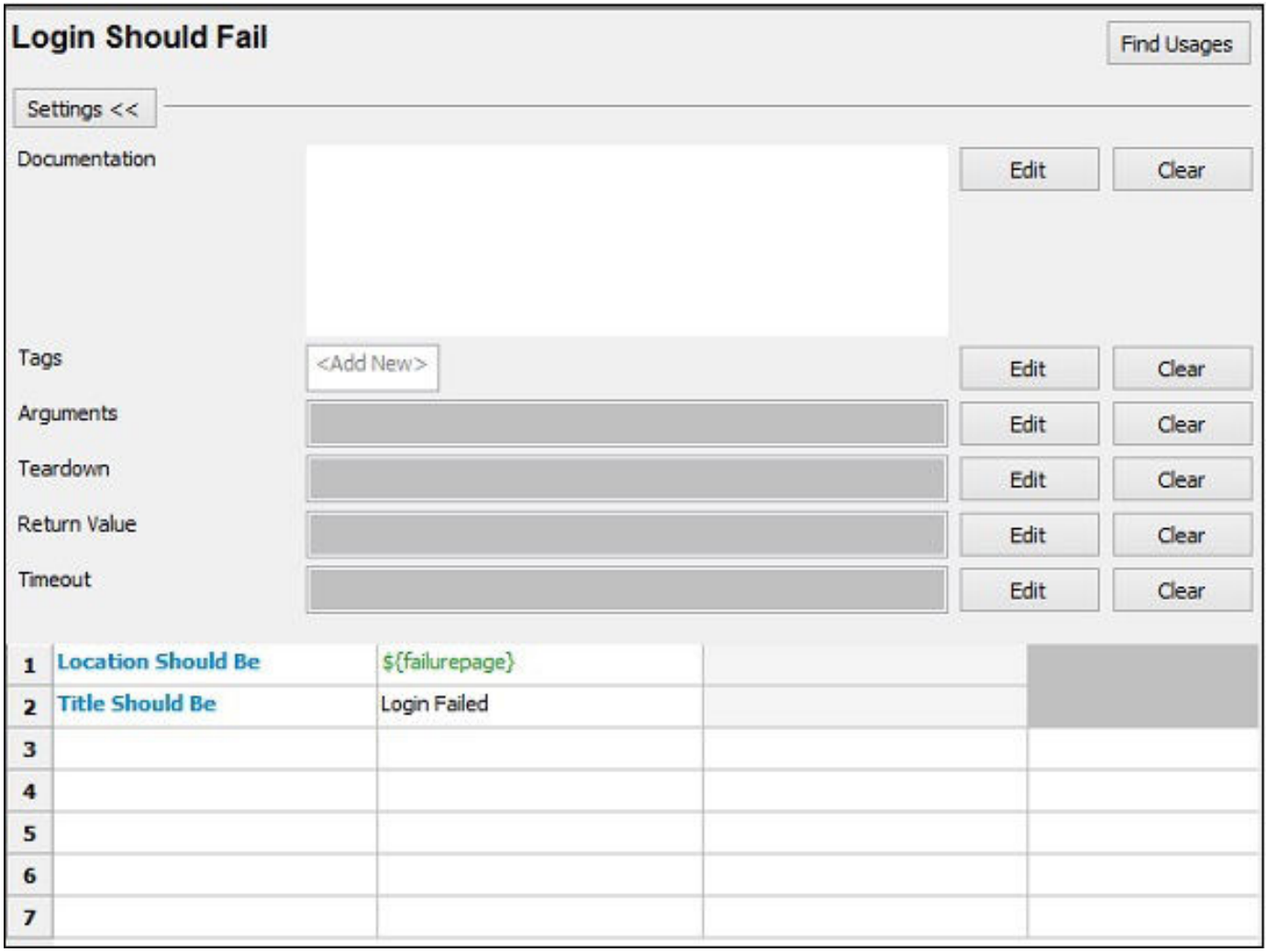
Enter Password



Submit Details

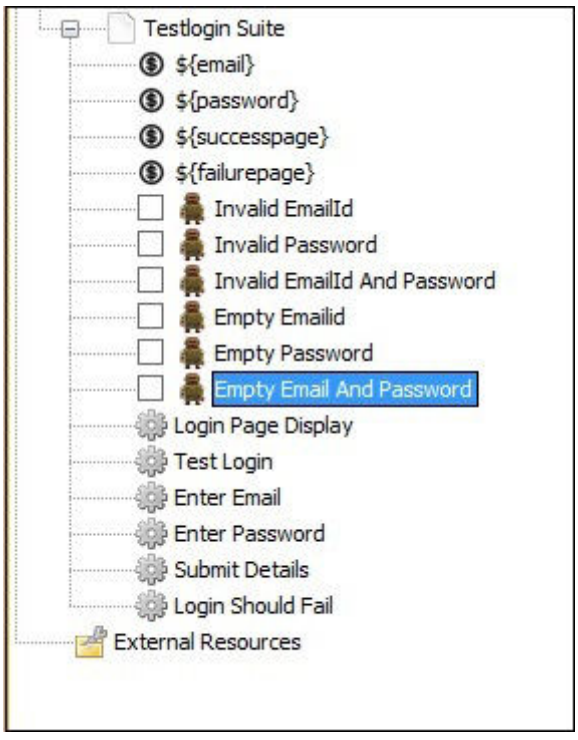


Login Should Fail



Now, we will write test cases, which will take different email id and password details to the template created.

The following is a list of test cases −



Invalid email id Test case

	Email	Password		
1	abcd@gmail.com	${password}		
2				
3				
4				
5				
6				

The email is passed with values abcd@gmail.com and ${password} is the password stored in the variable.

Invalid Password

	Email	Password		
1	${email}	xyz		
2				
3				
4				
5				
6				

Invalid Email Id and Password

	Email	Password		
1	invalid	invalid		
2				
3				
4				
5				
6				

Empty Email Id

	Email	Password		
1	${EMPTY}	${password}		
2				
3				
4				
5				
6				

Empty Password

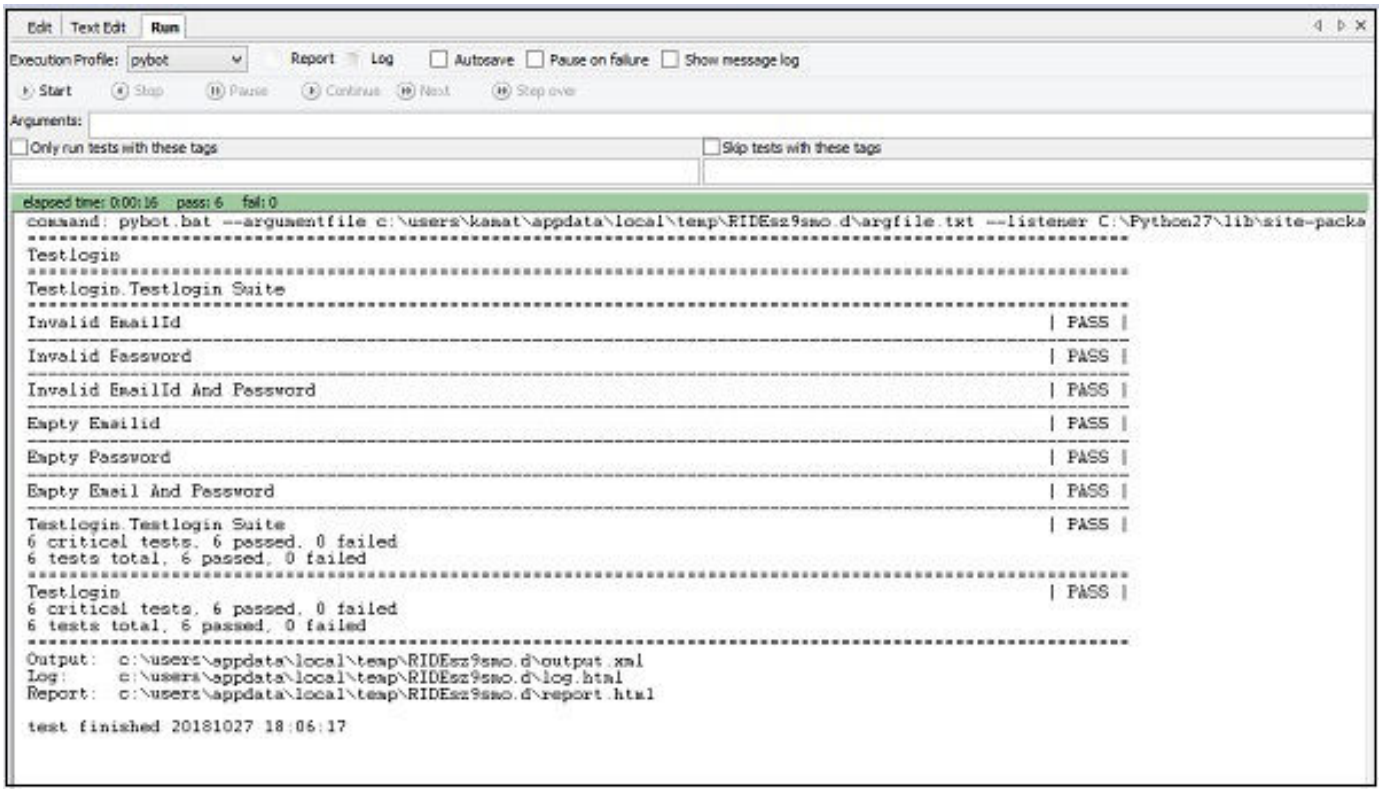
	Email	Password		
1	${email}	${EMPTY}		
2				
3				
4				
5				
6				

Empty Email and Password

	Email	Password		
1	${EMPTY}	${EMPTY}		
2				
3				
4				
5				
6				

Now, we are done with the test cases and can run the same.

Go to the Run tab and click Start to execute the test cases.



Here are the log messages for the test cases −

```
20181027 18:11:40.353 : INFO : Opening browser 'chrome' to base url '
http://localhost/robotframework/login.html'.
```

```
20181027 18:11:45.960 : INFO : Page title is 'Login Page'.
Starting test: Testlogin.Testlogin Suite.Invalid EmailId
20181027 18:11:45.991 : INFO : Opening url 'http://localhost/robotframework/login
20181027 18:11:46.169 : INFO : Page title is 'Login Page'.
20181027 18:11:46.180 : INFO : Typing text 'abcd@gmail.com' into text field 'emai
20181027 18:11:46.706 : INFO : Typing text 'admin' into text field 'passwd'.
20181027 18:11:47.075 : INFO : Clicking button 'btnsubmit'.
20181027 18:11:47.565 : INFO : Current location is 'http://localhost/robotframewo
20181027 18:11:47.584 : INFO : Page title is 'Login Failed'.
Ending test: Testlogin.Testlogin Suite.Invalid EmailId

Starting test: Testlogin.Testlogin Suite.Invalid Password
20181027 18:11:47.600 : INFO : Opening url 'http://localhost/robotframework/login
20181027 18:11:47.767 : INFO : Page title is 'Login Page'.
20181027 18:11:47.783 : INFO : Typing text 'admin@gmail.com' into text field 'ema
20181027 18:11:48.342 : INFO : Typing text 'invalid' into text field 'passwd'.
20181027 18:11:48.701 : INFO : Clicking button 'btnsubmit'.
20181027 18:11:49.035 : INFO : Current location is 'http://localhost/robotframewo
20181027 18:11:49.051 : INFO : Page title is 'Login Failed'.
Ending test: Testlogin.Testlogin Suite.Invalid Password

Starting test: Testlogin.Testlogin Suite.Invalid EmailId And Password
20181027 18:11:49.054 : INFO : Opening url 'http://localhost/robotframework/login
20181027 18:11:49.213 : INFO : Page title is 'Login Page'.
20181027 18:11:49.221 : INFO : Typing text 'invalid' into text field 'email'.
20181027 18:11:49.555 : INFO : Typing text 'invalid' into text field 'passwd'.
20181027 18:11:49.883 : INFO : Clicking button 'btnsubmit'.
20181027 18:11:50.162 : INFO : Current location is 'http://localhost/robotframewo
20181027 18:11:50.176 : INFO : Page title is 'Login Failed'.
Ending test: Testlogin.Testlogin Suite.Invalid EmailId And Password

Starting test: Testlogin.Testlogin Suite.Empty Emailid
20181027 18:11:50.188 : INFO : Opening url 'http://localhost/robotframework/login
20181027 18:11:50.302 : INFO : Page title is 'Login Page'.
20181027 18:11:50.306 : INFO : Typing text '' into text field 'email'.
20181027 18:11:50.486 : INFO : Typing text 'admin' into text field 'passwd'.
20181027 18:11:50.693 : INFO : Clicking button 'btnsubmit'.
20181027 18:11:50.935 : INFO : Current location is 'http://localhost/robotframewo
20181027 18:11:50.958 : INFO : Page title is 'Login Failed'.
Ending test: Testlogin.Testlogin Suite.Empty Emailid

Starting test: Testlogin.Testlogin Suite.Empty Password
20181027 18:11:50.958 : INFO : Opening url 'http://localhost/robotframework/login
20181027 18:11:51.063 : INFO : Page title is 'Login Page'.
20181027 18:11:51.071 : INFO : Typing text 'admin@gmail.com' into text field 'ema
20181027 18:11:51.367 : INFO : Typing text '' into text field 'passwd'.
20181027 18:11:51.561 : INFO : Clicking button 'btnsubmit'.
20181027 18:11:51.796 : INFO : Current location is 'http://localhost/robotframewo
20181027 18:11:51.808 : INFO : Page title is 'Login Failed'.
Ending test: Testlogin.Testlogin Suite.Empty Password
```

```
    g            g         g           p y

Starting test: Testlogin.Testlogin Suite.Empty Email And Password

20181027 18:11:51.811 : INFO : Opening url 'http://localhost/robotframework/login
20181027 18:11:51.908 : INFO : Page title is 'Login Page'.
20181027 18:11:51.916 : INFO : Typing text '' into text field 'email'.
20181027 18:11:52.049 : INFO : Typing text '' into text field 'passwd'.
20181027 18:11:52.193 : INFO : Clicking button 'btnsubmit'.
20181027 18:11:52.419 : INFO : Current location is 'http://localhost/robotframewo
20181027 18:11:52.432 : INFO : Page title is 'Login Failed'.
Ending test: Testlogin.Testlogin Suite.Empty Email And Password
```

Conclusion

We have seen here how to test a login page with different inputs, which will validate if the login is working fine or not. The details of how the execution takes place is given in the log section.