

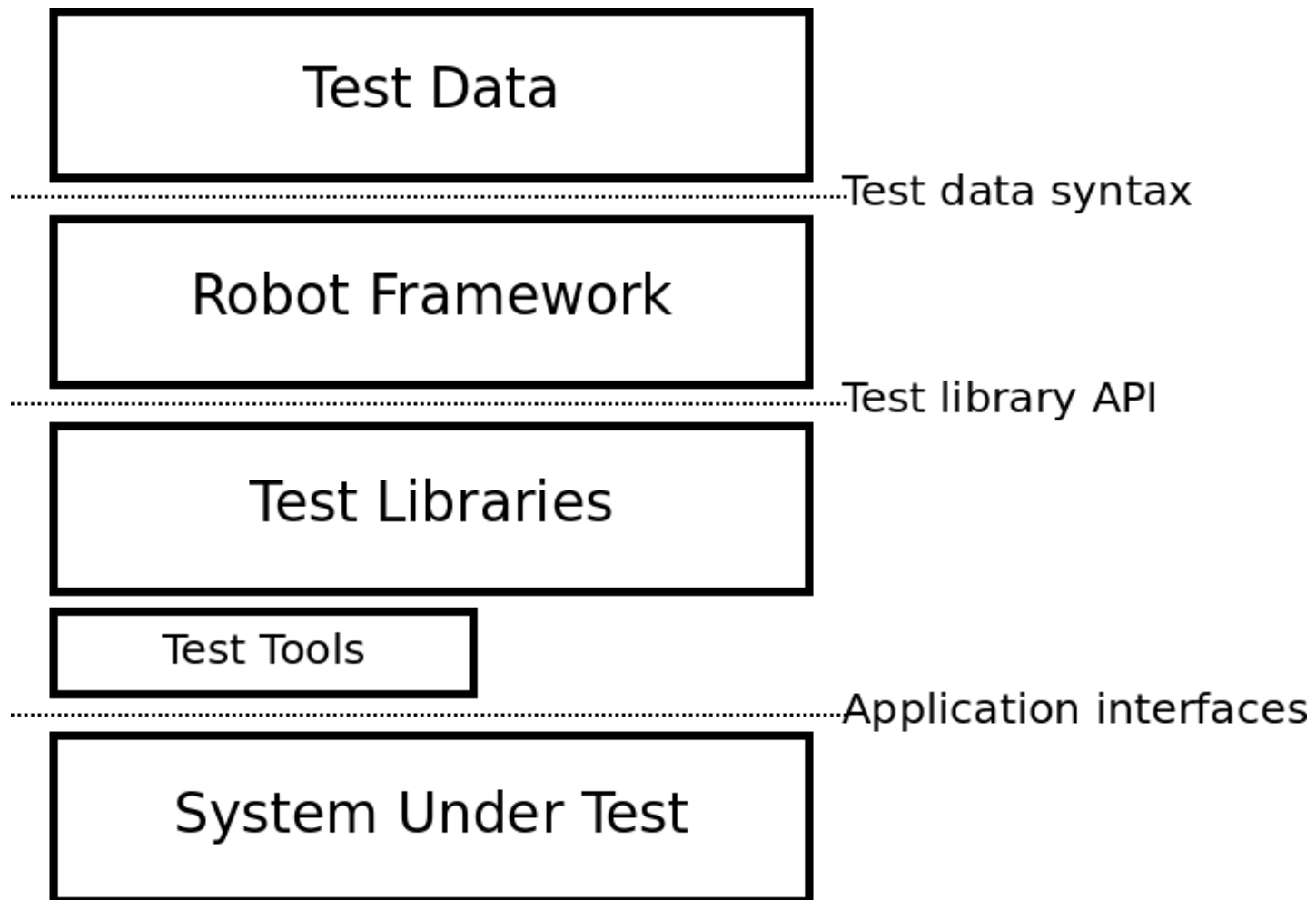


Commencer Avec Robot Framework

Généralités

- Cadre d'automatisation des tests génériques
 - Utilise l'approche de test basée sur les mots clés
 - Convient à la fois à l'automatisation des tests "normaux" et à l'ATDD
- Implémenté avec Python
 - Fonctionne également sur Jython (JVM) et IronPython (.NET)
 - Peut être étendu nativement en utilisant Python ou Java
 - Autres langues prises en charge via une interface à distance
- Open source
 - Hébergé sur GitHub, licence Apache 2
 - Sponsorisé par Nokia Networks
 - Écosystème riche et communauté très active

Architecture de haut niveau



Syntaxe simple basée sur des mots clés

***** Test Cases *****

Valid Login

Open Browser To Login Page

Input Username demo

Input Password mode

Submit Credentials

Welcome Page Should Be Open

[Teardown] Close Browser

Tests basés sur les données

*** Test Cases ***

Invalid Username

Invalid Password

Both Invalid

Empty Username

Empty Password

Both Empty

USER NAME

invalid

\${VALID USER}

invalid

\${EMPTY}

\${VALID USER}

\${EMPTY}

PASSWORD

\${VALID PWD}

invalid

whatever

\${VALID PWD}

\${EMPTY}

\${EMPTY}

Syntaxe Gherkin

***** Test Cases *****

Valid Login

Given browser is opened to login page

When user "demo" logs in with password "mode"

Then welcome page should be open

Mots clés de niveau supérieur

***** Keywords *****

Open Browser To Login Page

Open Browser \${LOGIN_URL} \${BROWSER}

Maximize Browser Window

Set Selenium Speed \${DELAY}

Login Page Should Be Open

Login Page Should Be Open

Title Should Be Login Page

Input Username

[Arguments] \${username}

Input Text username_field \${username}

Input Password

[Arguments] \${password}

Input Text password_field \${password}

API de bibliothèque de tests simples

```
class Selenium2Library:

    def input_text(self, locator, text):
        """Types given `text` into text field `locator`."""
        print "Typing text '%s' into '%s'." % (text, locator)
        element = self._element_find(locator)
        element.clear()
        element.send_keys(text)

    def title_should_be(self, title):
        """Verifies that current page title equals `title`."""
        actual = self.get_title()
        if actual != title:
            raise AssertionError("Title should have been '%s' but was '%s'."
                                % (title, actual))
        print "Page title is '%s'." % title
```

Above is real Selenium2Library code but slightly simplified.

variables

- Facile à créer :

```
*** Variables ***
```

```
${BROWSER}      Firefox  
${HOST}         localhost:7272  
${LOGIN URL}    http://${HOST}/  
${WELCOME URL}  http://${HOST}/welcome.html
```

- Remplacer depuis la ligne de commande :
variable NAVIGATEUR : IE

Balisage

- Métadonnées gratuites pour catégoriser les cas de test
- Statistiques par balises collectées automatiquement
- Sélectionner les cas de test à exécuter
- Spécifier quels cas de test sont considérés comme critiques

Rapports clairs

Login Tests Test Report

Generated
20080613 14:12:20 GMT +02:00
1 minute 0 seconds ago

Summary Information

Status: 6 critical tests failed
Start Time: 20080613 14:12:08.445
End Time: 20080613 14:12:39.666
Elapsed Time: 00:00:31.221

Test Statistics

Total Statistics	Total	Pass	Fail
Critical Tests	10	4	6
All Tests	10	4	6
Statistics by Tag	Total	Pass	Fail
regression	10	4	6
smoke	4	4	0
Statistics by Suite	Total	Pass	Fail
Login Tests	10	4	6
I.Higher Level Login	3	3	0
I.Invalid Login	6	0	6
I.Simple Login	1	1	0

Test Details by Suite

Name	Documentation	Metadata / Tags	Crit.	Status
Login Tests			N/A	FAIL
I.Higher Level Login			N/A	PASS
I.h.Higher Level Valid Login		regression, smoke	yes	PASS
I.h.Even Higher Level Valid Login		regression, smoke	yes	PASS
I.h.Highest Level Login		regression, smoke	yes	PASS
I.Invalid Login			N/A	FAIL
I.i.Invalid Username		regression	yes	FAIL
I.i.Invalid Password		regression	yes	FAIL
I.i.Invalid Username				FAIL

Login Tests Test Report

Generated
20080613 13:39:08 GMT +02:00
1 minute 9 seconds ago

Summary Information

Status: All tests passed
Documentation: Demo test cases for Robot Framework using Selenium test library
Start Time: 20080613 13:38:36.191
End Time: 20080613 13:39:08.068
Elapsed Time: 00:00:31.877

Test Statistics

Total Statistics	Total	Pass	Fail	Graph
Critical Tests	10	10	0	<div></div>
All Tests	10	10	0	<div></div>
Statistics by Tag	Total	Pass	Fail	Graph
regression	10	10	0	<div></div>
smoke	4	4	0	<div></div>
Statistics by Suite	Total	Pass	Fail	Graph
Login Tests	10	10	0	<div></div>
I.Higher Level Login	3	3	0	<div></div>
I.Invalid Login	6	6	0	<div></div>
I.Simple Login	1	1	0	<div></div>

Test Details by Suite

Name	Documentation	Metadata / Tags	Crit.	Status	Message	Start / Elapsed
Login Tests	Demo test cases for Robot Framework using Selenium test library		N/A	PASS	10 critical tests, 10 passed, 0 failed 10 tests total, 10 passed, 0 failed	20080613 13:38:36 00:00:32
I.Higher Level Login			N/A	PASS	3 critical tests, 3 passed, 0 failed 3 tests total, 3 passed, 0 failed	20080613 13:38:36 00:00:17
I.h.Higher Level Valid Login		regression, smoke	yes	PASS		20080613 13:38:36 00:00:06
I.h.Even Higher Level Valid Login		regression, smoke	yes	PASS		20080613 13:38:41 00:00:05
I.h.Highest Level Login		regression, smoke	yes	PASS		20080613 13:38:47 00:00:06
I.Invalid Login			N/A	PASS	6 critical tests, 6 passed, 0 failed 6 tests total, 6 passed, 0 failed	20080613 13:38:52 00:00:10
I.i.Invalid Username		regression	yes	PASS		20080613 13:38:57 00:00:01
I.i.Invalid Password		regression	yes	PASS		20080613 13:38:58 00:00:01
I.i.Invalid Username		regression	yes	PASS		20080613 13:38:59

Journaux détaillés

[-] TEST SUITE: Higher Level Login

[Expand All](#)

Full Name: Login Tests.Higher Level Login
Source: /home/jth/workspace/seleniumlib/demo/login_tests/higher_level_login.html
Start / End / Elapsed: 20090415 07:36:29.500 / 20090415 07:36:55.480 / 00:00:25.980
Overall Status: **FAIL**
Message: 3 critical tests, 0 passed, 3 failed
 3 tests total, 0 passed, 3 failed

[-] TEST CASE: Higher Level Valid Login

[Expand All](#)

Full Name: Login Tests.Higher Level Login.Higher Level Valid Login
Tags: regression, smoke
Start / End / Elapsed: 20090415 07:36:29.533 / 20090415 07:36:36.412 / 00:00:06.879
Status: **FAIL (critical)**
Message: Location should have been '<http://localhost:7272/welcome.html>' but was '<http://localhost:7272/error.html>'

+ SETUP: resource.Open Login Page

+ KEYWORD: resource.Input Username demo

+ KEYWORD: resource.Input Password mode

+ KEYWORD: resource.Click Login Button

[-] KEYWORD: resource.Welcome Page Should Be Open

Start / End / Elapsed: 20090415 07:36:36.340 / 20090415 07:36:36.374 / 00:00:00.034

[-] KEYWORD: SeleniumLibrary.Location Should Be \${WELCOME URL}

Documentation: Verifies that current URL is exactly `url`.

Start / End / Elapsed: 20090415 07:36:36.341 / 20090415 07:36:36.374 / 00:00:00.033

07:36:36.373 INFO Verifying current location is '<http://localhost:7272/welcome.html>'.

07:36:36.374 **FAIL** Location should have been '<http://localhost:7272/welcome.html>' but was '<http://localhost:7272/error.html>'

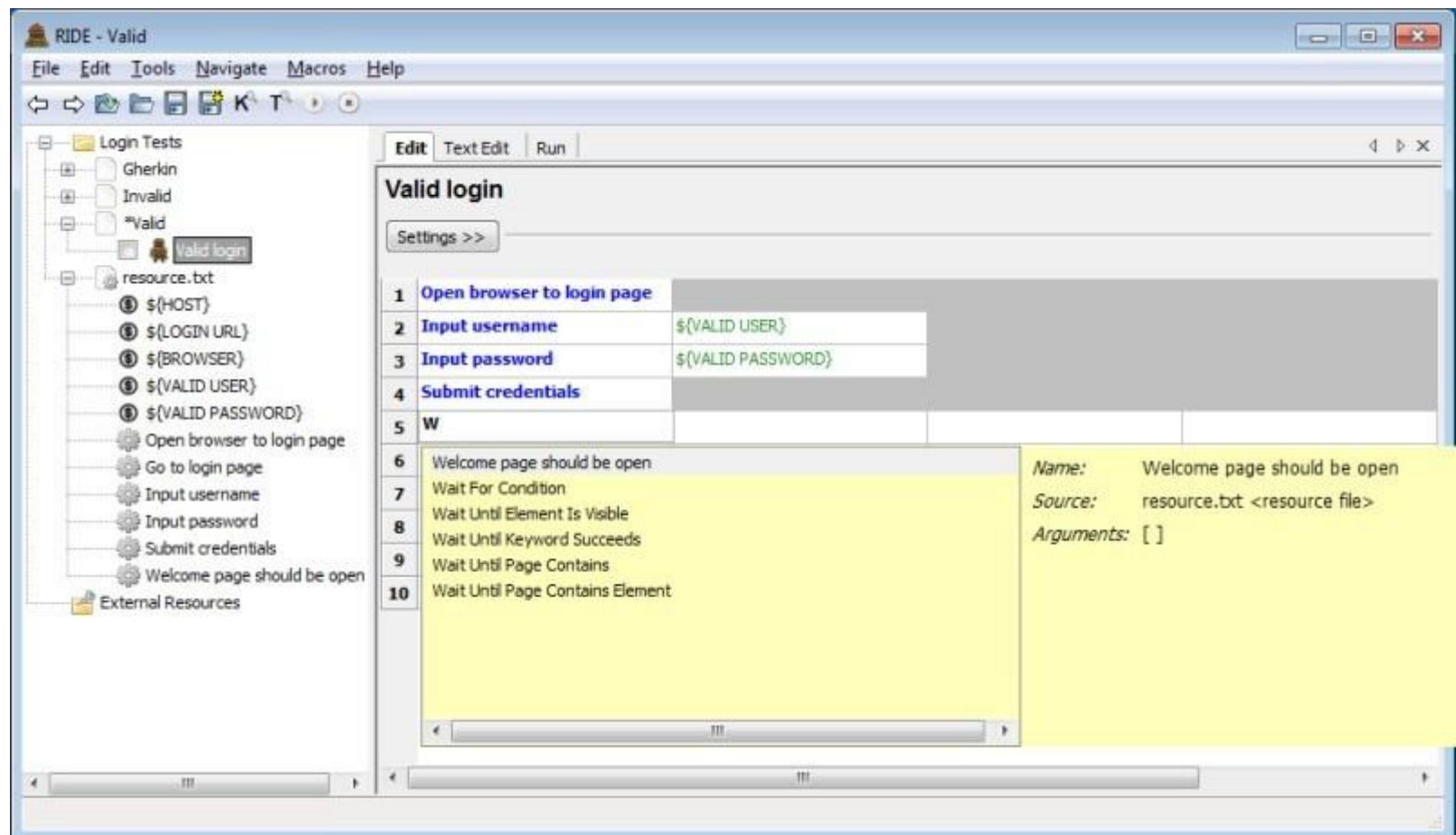
+ TEARDOWN: SeleniumLibrary.Close Browser

Différentes bibliothèques de tests

- Bibliothèques standards
 - Inclus dans l'installation normale
 - Système d'exploitation, Capture d'écran, Chaîne, Telnet, XML, ...
- Bibliothèques externes
 - Doit être installé séparément
 - Selenium2Library, SwingLibrary, DatabaseLibrary, AutoltLibrary, SSHLibrary, HTTPLibrary, ...
- Bibliothèques spécifiques au projet et à l'équipe

Assistance de l'éditeur

- BALADE



- Plugins pour Eclipse, IntelliJ/PyCharm, SubLime, TextMate, Vim, Emacs, Brackets, Atom, ...

Intégration facile

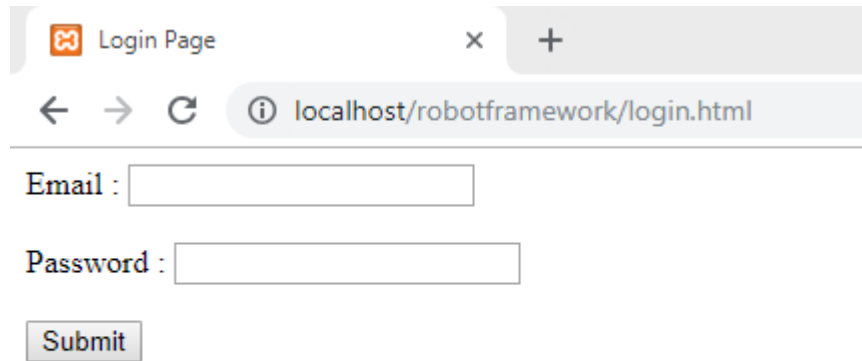
- Les suites de tests sont créées à partir de fichiers et de répertoires
 - Trivial à stocker dans n'importe quel système de contrôle de version
- Interface de ligne de commande simple
 - Lancement facile de l'exécution des tests par des outils externes
- Sortie également au format XML
 - Toutes les informations dans un format lisible par machine
 - Les sorties de différents tests peuvent être combinées
- Plugins pour CI courants et outils de build
 - Jenkins, Fourmi, Maven

Test « Login Page » avec Robot Framework

- Importer des bibliothèques
- Travailler avec des variables
- Créer des mots-clés personnalisés
- Comment écrire des cas de test
- Comment créer une configuration et un démontage
- Comment exécuter des cas de test
- Comment travailler avec des cas de test basés sur des données

Nous utiliserons toutes les fonctionnalités ci-dessus et les utiliserons pour tester la page de connexion dans ce chapitre. Nous avons une page de connexion qui prend en compte l'identifiant de l'e-mail et le mot de passe. Lorsque vous entrez un identifiant de messagerie correct et mot de passe, vous serez redirigé vers une page d'accueil. Si l'utilisateur saisit un identifiant de messagerie invalide ou mot de passe, la page sera redirigée vers la page d'erreur.

La capture d'écran suivante montre une page de connexion :



The screenshot shows a web browser window with a single tab titled "Login Page". The address bar displays "localhost/robotframework/login.html". Below the address bar, there is a form with two input fields. The first field is labeled "Email :" and the second is labeled "Password :". Below these fields is a button labeled "Submit".

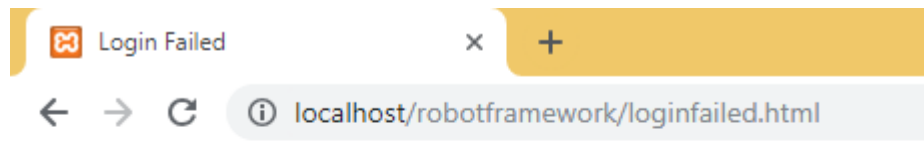
HTML Code

```
<html>
<head>
<title>Login Page</title>
</head>
<body>
<script type="text/javascript">
function wsSubmit() {

    if (document.getElementById("email").value == "admin@gmail.com" &&
document.getElementById("passwd").value == "admin") {
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>
```

L'écran suivant apparaît lorsque l'identifiant de messagerie ou le mot de passe n'est pas valide :

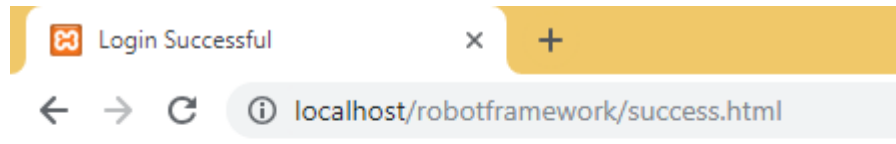


Login Failed

HTML Code

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

L'écran suivant apparaît lorsque l'identifiant de messagerie et le mot de passe sont valides :



Login Successful

HTML Code

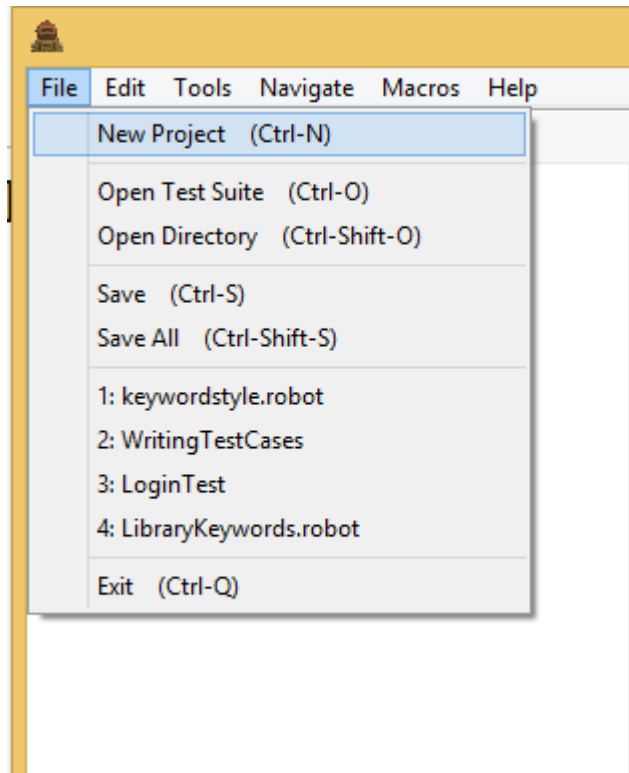
```
<html>
<head>
<title>Login Successful</title>
</head>
<body>
<div id="loginfailed">
<h1>Login Successful</h1>
</div>
</body>
</html>
```

Nous allons maintenant écrire des cas de test pour la page de test ci-dessus. Pour commencer, nous allons d'abord exécuter la commande pour ouvrir Ride.

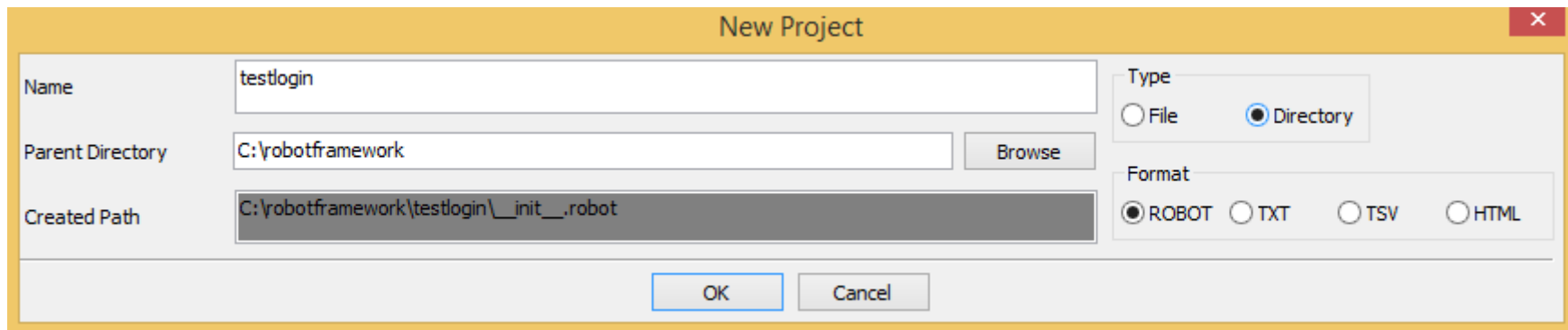
Commande

ride.py

Une fois cela fait, nous allons commencer la configuration du projet comme indiqué ci-dessous :



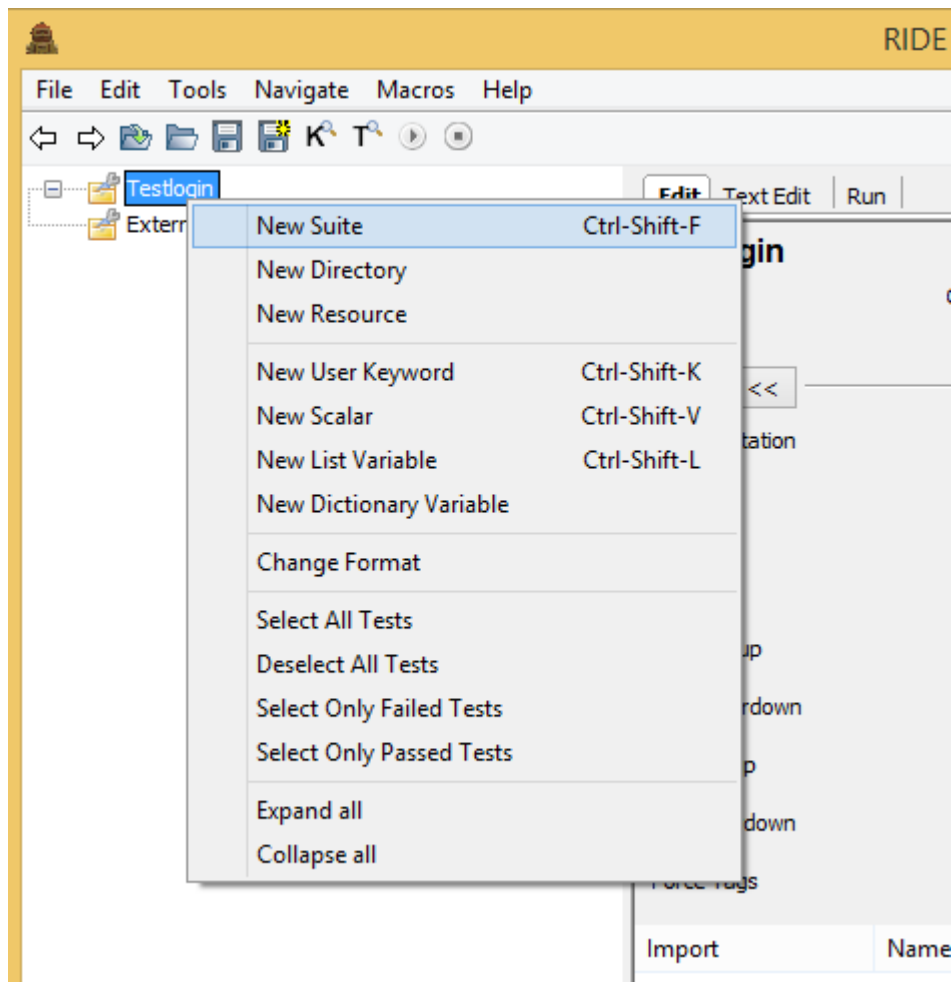
Cliquez sur Nouveau projet et entrez le nom du projet.



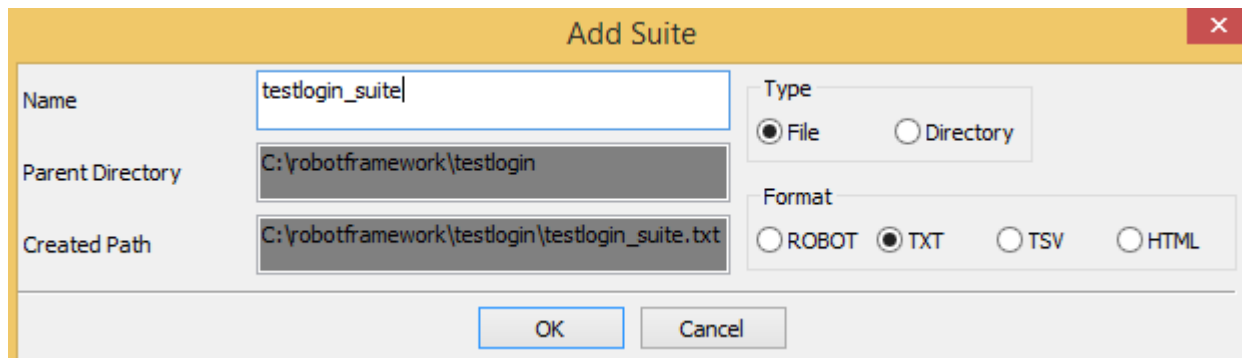
Nous enregistrerons le type de projet sous Directory. Le nom donné au projet est testlogin.

Cliquez sur OK pour enregistrer le projet.

Maintenant, nous allons créer une suite de tests dans le projet.



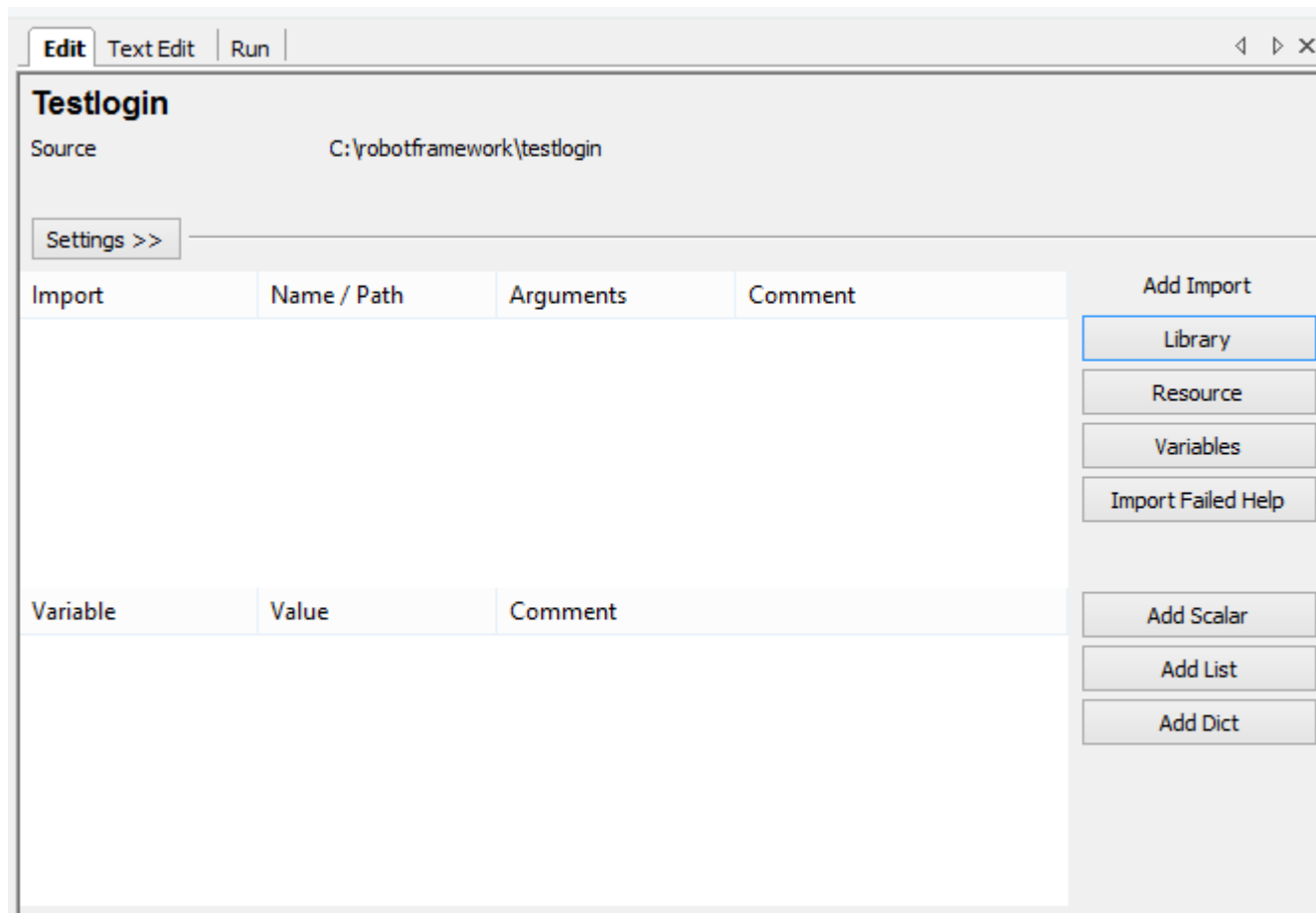
Cliquez sur New Suite et il affichera un écran comme indiqué ci-dessous :

A screenshot of the 'Add Suite' dialog box in Robot Framework. The dialog has a yellow title bar with the text 'Add Suite' and a red close button. It contains three input fields on the left: 'Name' with the value 'testlogin_suite', 'Parent Directory' with the value 'C:\robotframework\testlogin', and 'Created Path' with the value 'C:\robotframework\testlogin\testlogin_suite.txt'. On the right, there are two sections: 'Type' with radio buttons for 'File' (selected) and 'Directory', and 'Format' with radio buttons for 'ROBOT', 'TXT' (selected), 'TSV', and 'HTML'. At the bottom, there are 'OK' and 'Cancel' buttons.

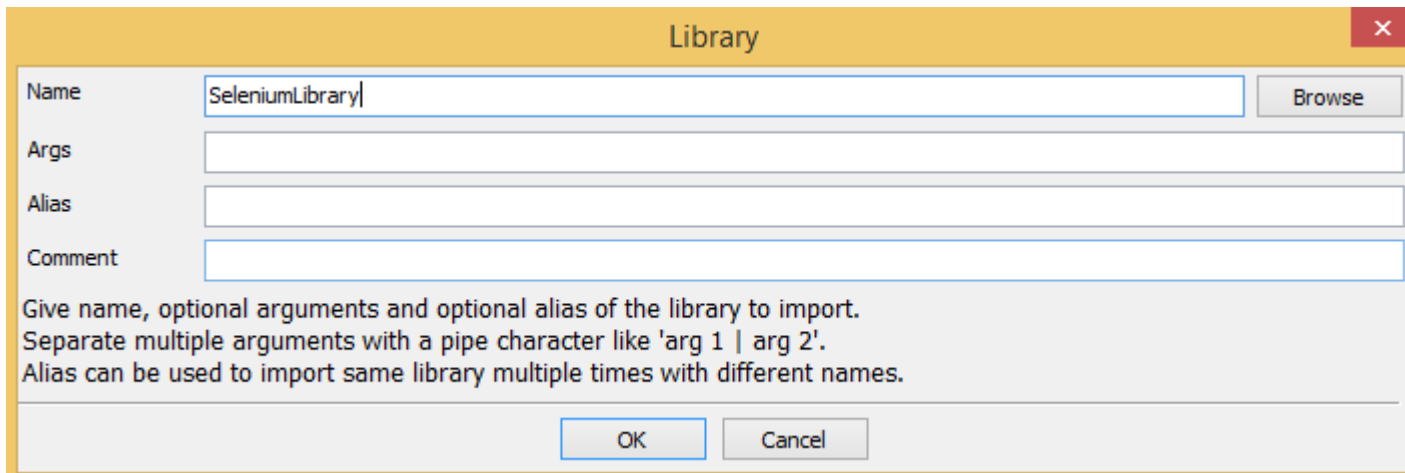
Name	testlogin_suite	Type	<input checked="" type="radio"/> File <input type="radio"/> Directory
Parent Directory	C:\robotframework\testlogin	Format	<input type="radio"/> ROBOT <input checked="" type="radio"/> TXT <input type="radio"/> TSV <input type="radio"/> HTML
Created Path	C:\robotframework\testlogin\testlogin_suite.txt		
<div>OK Cancel</div>			

Cliquez sur OK pour enregistrer la suite de tests. Nous devons importer la bibliothèque de sélénium puisque nous serons travaillées avec le navigateur.

Importer la bibliothèque dans le projet principal et également dans la suite de tests créée.



Cliquez sur Bibliothèque comme dans la capture d'écran ci-dessus. En cliquant sur Bibliothèque, l'écran suivant s'affichera apparaître.



Library

Name: SeleniumLibrary

Args:

Alias:

Comment:

Give name, optional arguments and optional alias of the library to import.
Separate multiple arguments with a pipe character like 'arg 1 | arg 2'.
Alias can be used to import same library multiple times with different names.

Cliquez sur OK pour enregistrer la bibliothèque pour le projet.

Une fois la bibliothèque enregistrée pour le projet, il affichera la bibliothèque dans les paramètres :

Testlogin

Source C:\robotframework\testlogin

Settings >>

Import	Name / Path	Arguments	Comment
Library	SeleniumLibrary		

Variable	Value	Comment
----------	-------	---------

Add Import

Library

Resource

Variables

Import Failed Help

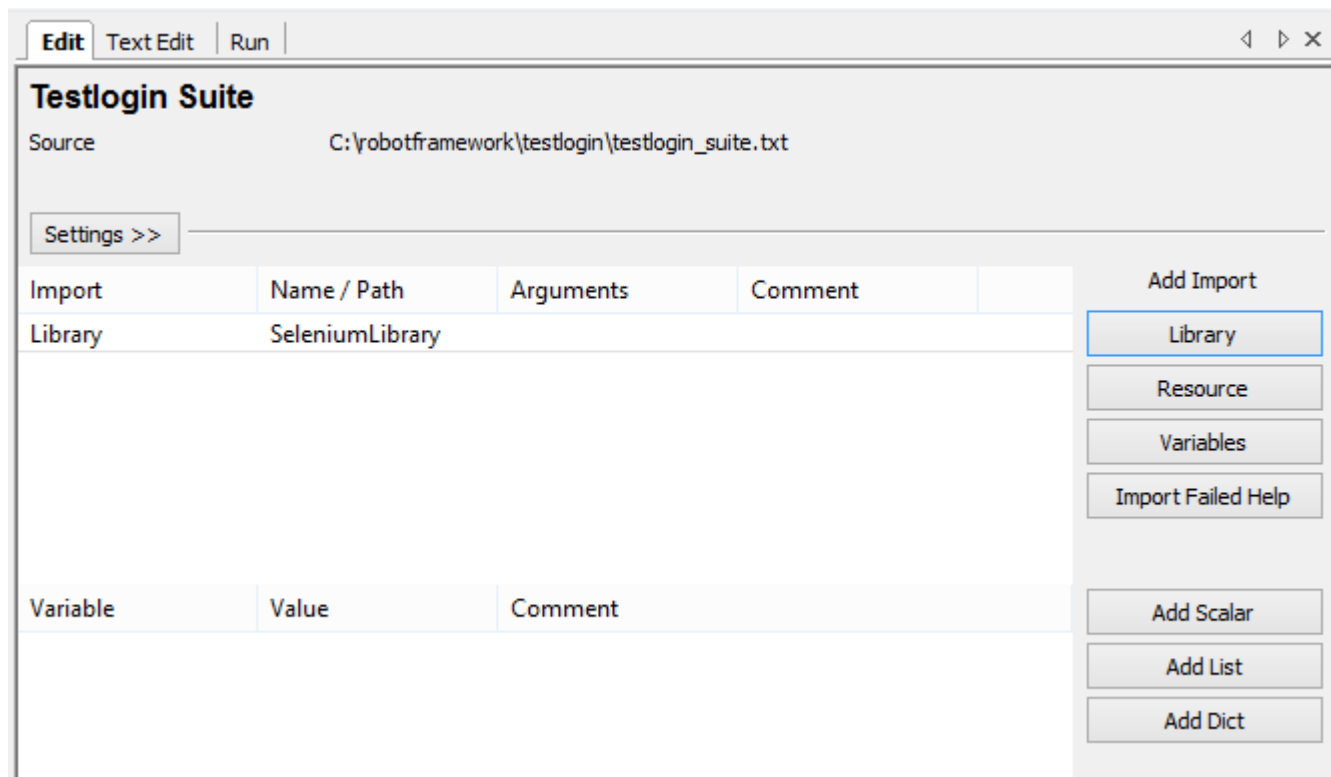
Add Scalar

Add List

Add Dict

Répétez la même étape pour la suite de tests créée.

Voici la bibliothèque ajoutée pour Test suite:



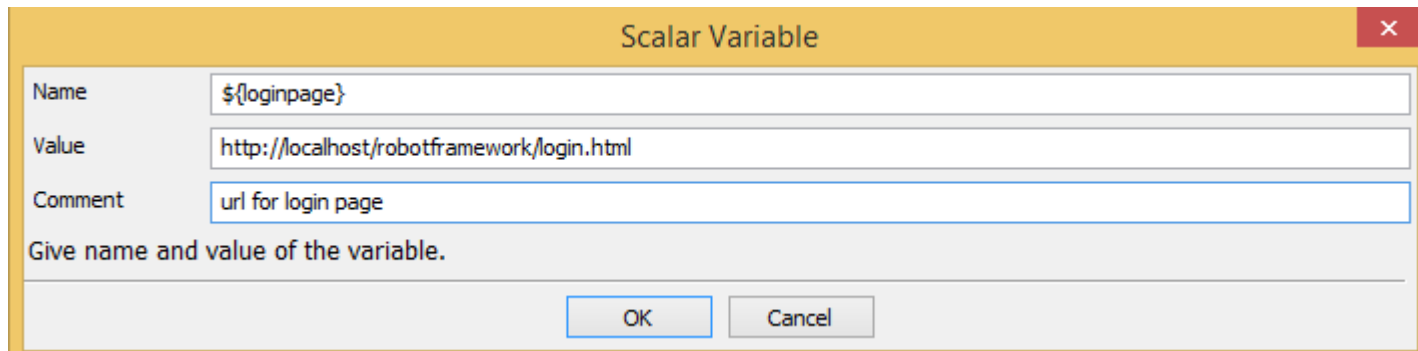
Maintenant, dans le projet principal, nous allons créer une configuration et un démontage. Nous aimerions ouvrir le page de connexion dans le navigateur Chrome et agrandissez la fenêtre. En démontage, nous fermerons le navigateur.

Pour la configuration, nous allons créer un mot-clé défini par l'utilisateur appelé Ouvrir la page de connexion. Ce mot clé prendra 2 arguments, l'URL de la page de connexion et le nom du navigateur.

Maintenant, nous avons besoin de 2 variables scalaires qui nous aideront à stocker les valeurs – url et le navigateur Nom.

Dans ride, créez 2 variables `${loginpage}` et `${browser}` comme suit :

`${loginpage}`



Scalar Variable

Name: `${loginpage}`

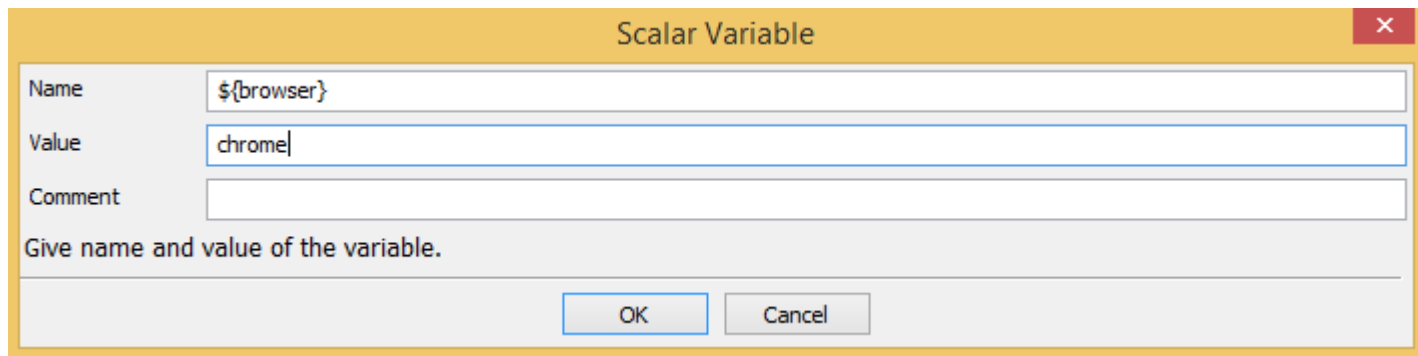
Value: `http://localhost/robotframework/login.html`

Comment: `url for login page`

Give name and value of the variable.

OK Cancel

`${browser}`



Scalar Variable

Name: `${browser}`

Value: `chrome`

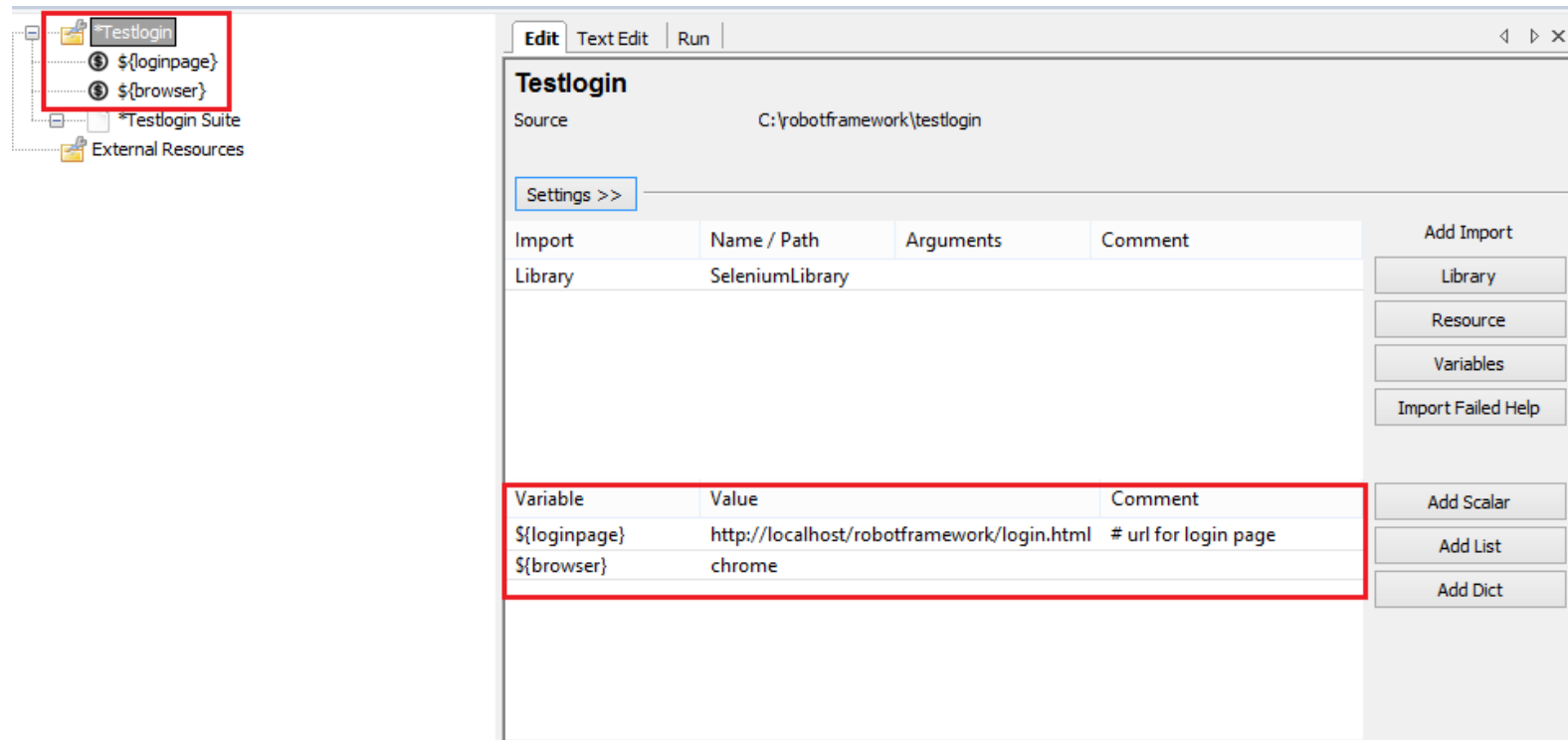
Comment:

Give name and value of the variable.

OK Cancel

Enregistrez les deux variables.

Les variables seront affichées sous votre projet comme suit :

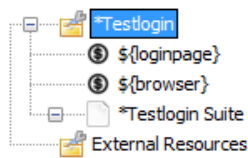


The screenshot shows the Robot Framework IDE interface. On the left, the project tree shows a folder named 'Testlogin' containing two variables: '\${loginpage}' and '\${browser}'. The main window displays the configuration for the 'Testlogin' project, located at 'C:\robotframework\testlogin'. The 'Settings >>' button is visible. Below it, a table lists the imported libraries, showing 'SeleniumLibrary'. At the bottom, a table lists the defined variables:

Variable	Value	Comment
\${loginpage}	http://localhost/robotframework/login.html	# url for login page
\${browser}	chrome	

Maintenant, nous allons ajouter la configuration et le démontage du projet principal.

Cliquez sur le projet sur le côté gauche. Dans les paramètres, cliquez sur Suite Setup.



The screenshot shows the 'Testlogin' window in the Robot Framework Test Editor. The window has tabs for 'Edit', 'Text Edit', and 'Run'. The 'Source' is set to 'C:\robotframework\testlogin'. Below the source, there is a 'Settings <<' button and a 'Documentation' section with 'Edit' and 'Clear' buttons. The 'Suite Setup' and 'Suite Teardown' sections are highlighted with an orange border. Each section has a text input field and 'Edit' and 'Clear' buttons. The 'Test Setup' and 'Test Teardown' sections also have text input fields and 'Edit' and 'Clear' buttons. The 'Force Tags' section has a '<Add New>' button and 'Edit' and 'Clear' buttons.

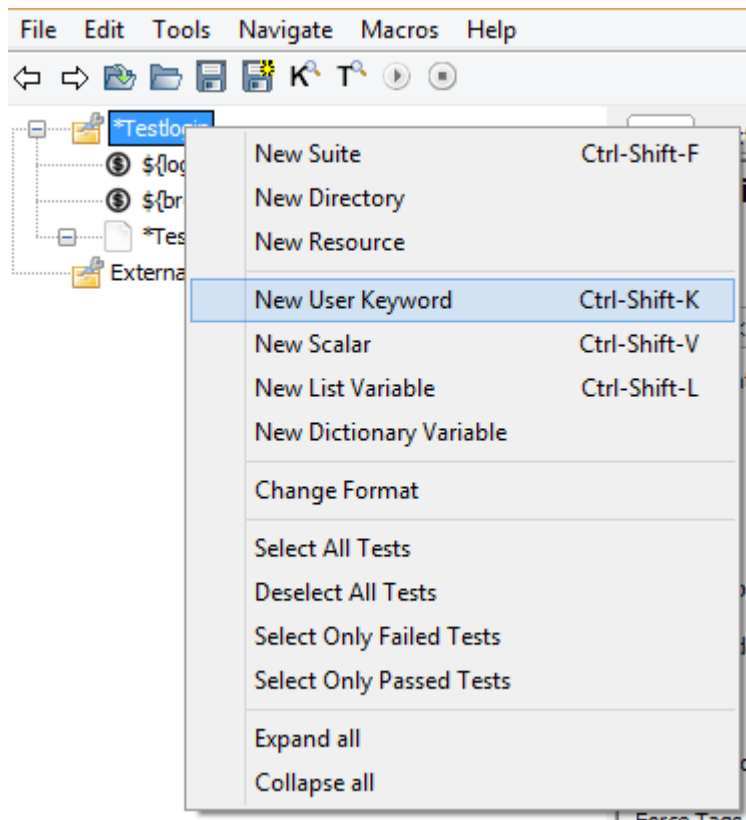
The screenshot shows the 'Suite Setup' dialog box. It has a title bar with a close button. The main area contains a text input field with the text 'Open Login Page | \${loginpage} | \${browser}'. Below this is a 'Comment' section with a text input field containing 'setup to open login page'. A paragraph of text explains the keyword: 'This keyword is executed before executing any of the test cases or lower level suites. Separate possible arguments with a pipe character like 'My Keyword | arg 1 | arg 2'. Possible pipes in the value must be escaped with a backslash like '\\'. At the bottom, there are 'OK' and 'Cancel' buttons.

Nous avons créé une configuration qui utilise le mot-clé utilisateur Ouvrir la page de connexion avec des arguments `${loginpage}` et `${browser}`.

Cliquez sur OK pour enregistrer la configuration.

Maintenant, nous devons créer le mot-clé défini par l'utilisateur Ouvrir la page de connexion, ce qui se fait comme suit :

Faites un clic droit sur le projet et cliquez sur Nouveau mot clé utilisateur :



Lorsque vous cliquez sur Nouveau mot-clé utilisateur, l'écran suivant s'affiche :

New User Keyword

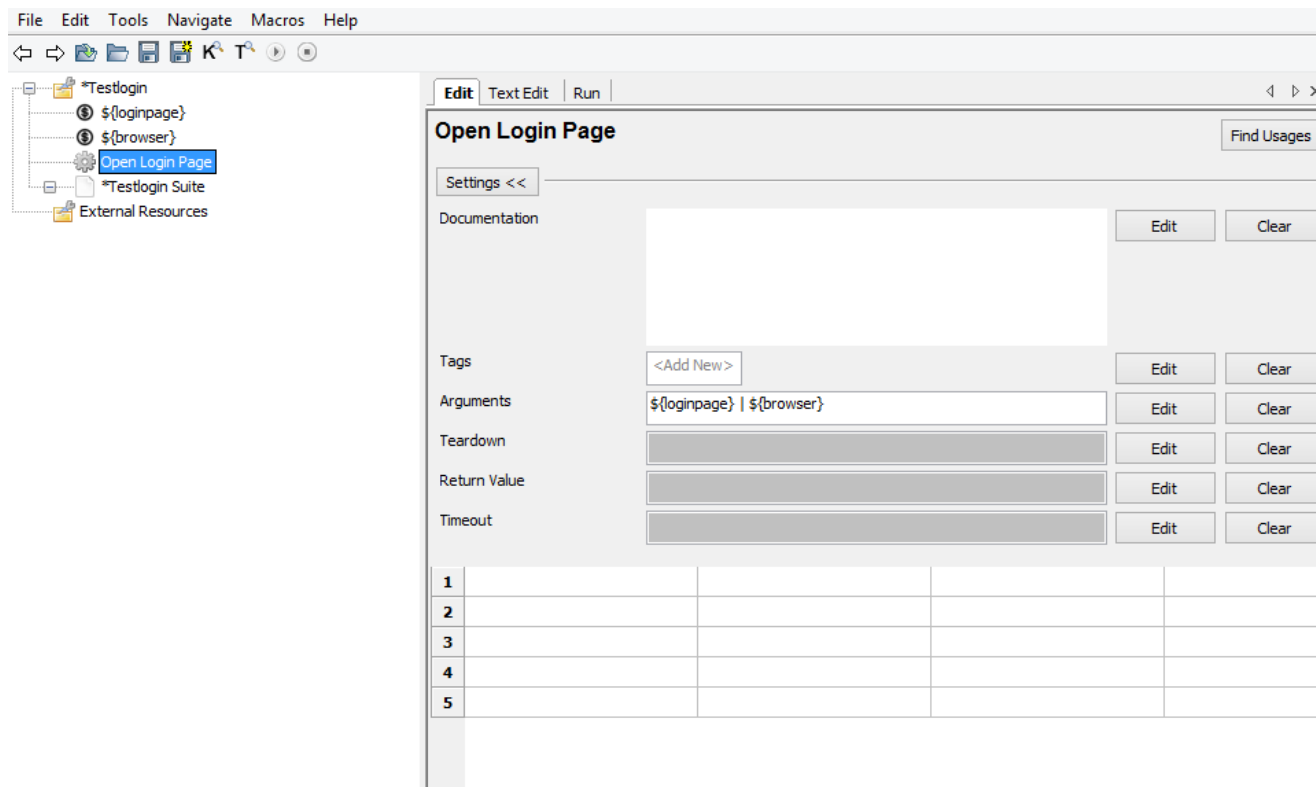
Name: Open Login Page

Arguments: \${loginpage} | \${browser}

Give a name and arguments for the new user keyword.
Specify the arguments separated with a pipe character like '\${arg1} | \${arg2}'.
Default values are given using equal sign and the last argument can be a list variable.
Example: '\${arg1} | \${arg2}=default value | @{rest}'.
Note. You can use variable shortcuts in this field.

OK Cancel

Ici, le mot-clé reçoit 2 arguments – `${loginpage}` et `${browser}`. Cliquez sur OK pour enregistrer le mot-clé de l'utilisateur.



Maintenant, nous devons entrer les mots-clés de la bibliothèque, ce qui ouvrira l'URL.

Open Login Page Find Usages

Settings <<

Documentation Edit Clear

Tags <Add New> Edit Clear

Arguments \${loginpage} | \${browser} Edit Clear

Teardown Edit Clear

Return Value Edit Clear

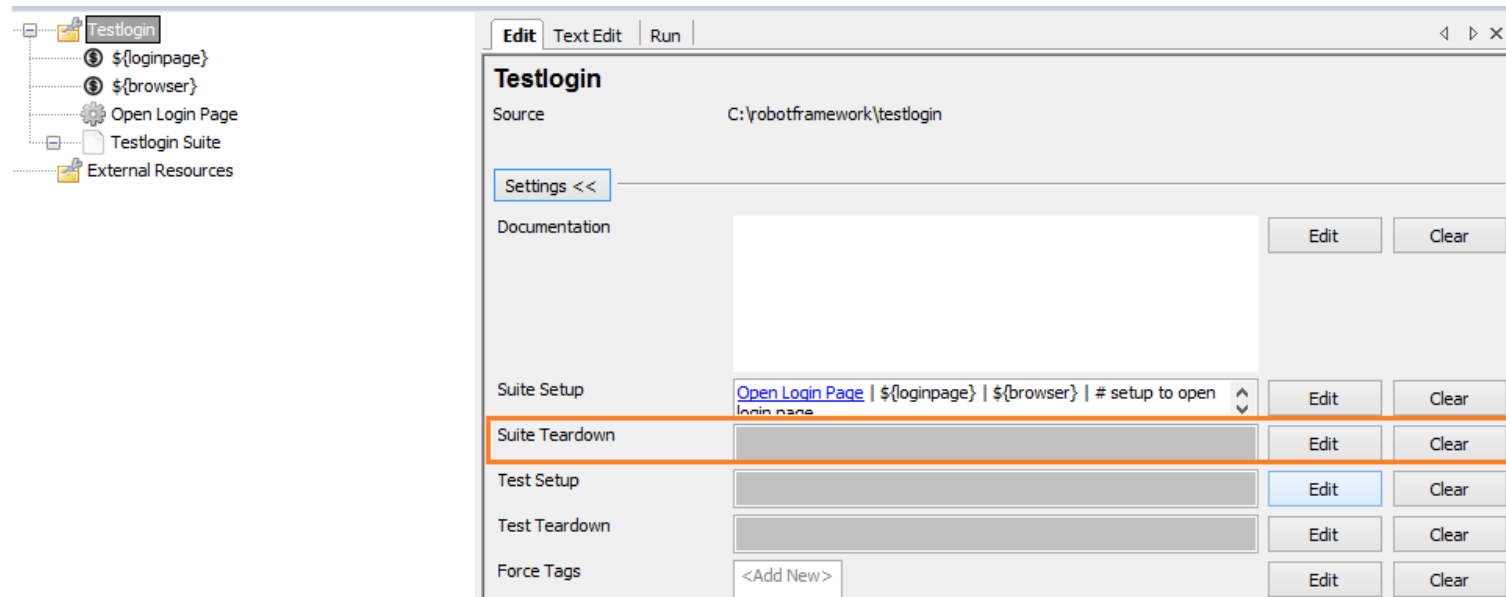
Timeout Edit Clear

1	Open Browser	\${loginpage}	\${browser}
2	Maximize Browser Window		
3	Title Should Be	Login Page	
4			
5			

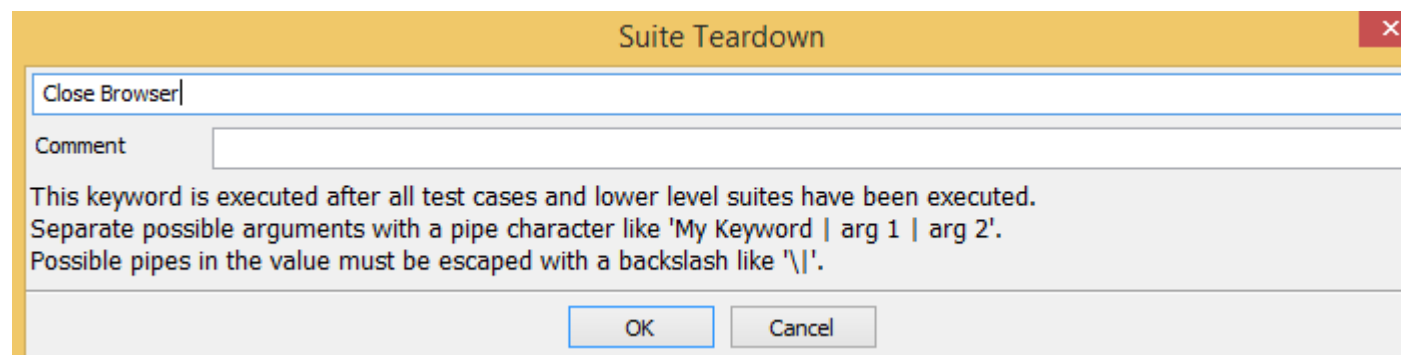
Le mot-clé défini par l'utilisateur Ouvrir la page de connexion contient les détails suivants :

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

Maintenant, nous allons créer Suite Teardown pour la suite.



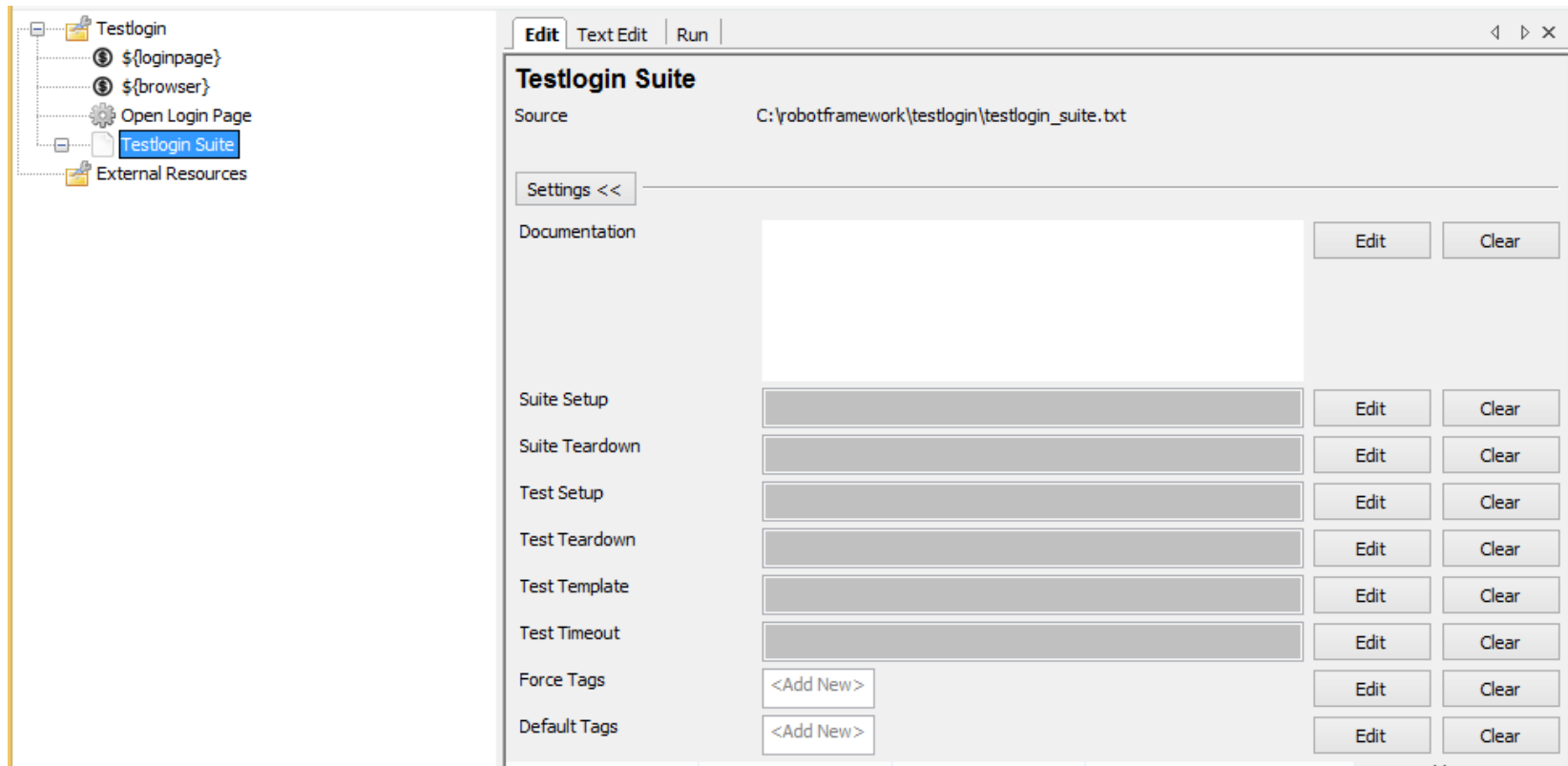
Cliquez sur Modifier pour le démontage de la suite et entrez les détails :



Pour le démontage de la suite, nous utilisons directement le mot-clé de la bibliothèque, qui fermera le navigateur.

Cliquez sur OK pour enregistrer le démontage de la suite.

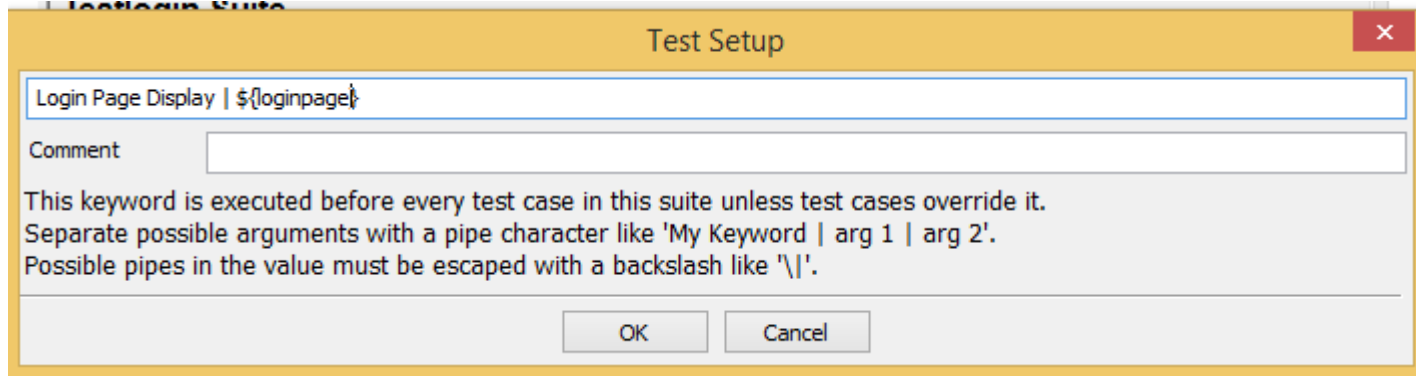
Maintenant, cliquez sur la suite Testlogin que nous avons créée.



Créons maintenant une configuration pour la suite de tests – Test Setup. Cette configuration doit être exécutée

première.

Cliquez sur Modifier pour la configuration du test et entrez les détails.



Test Setup

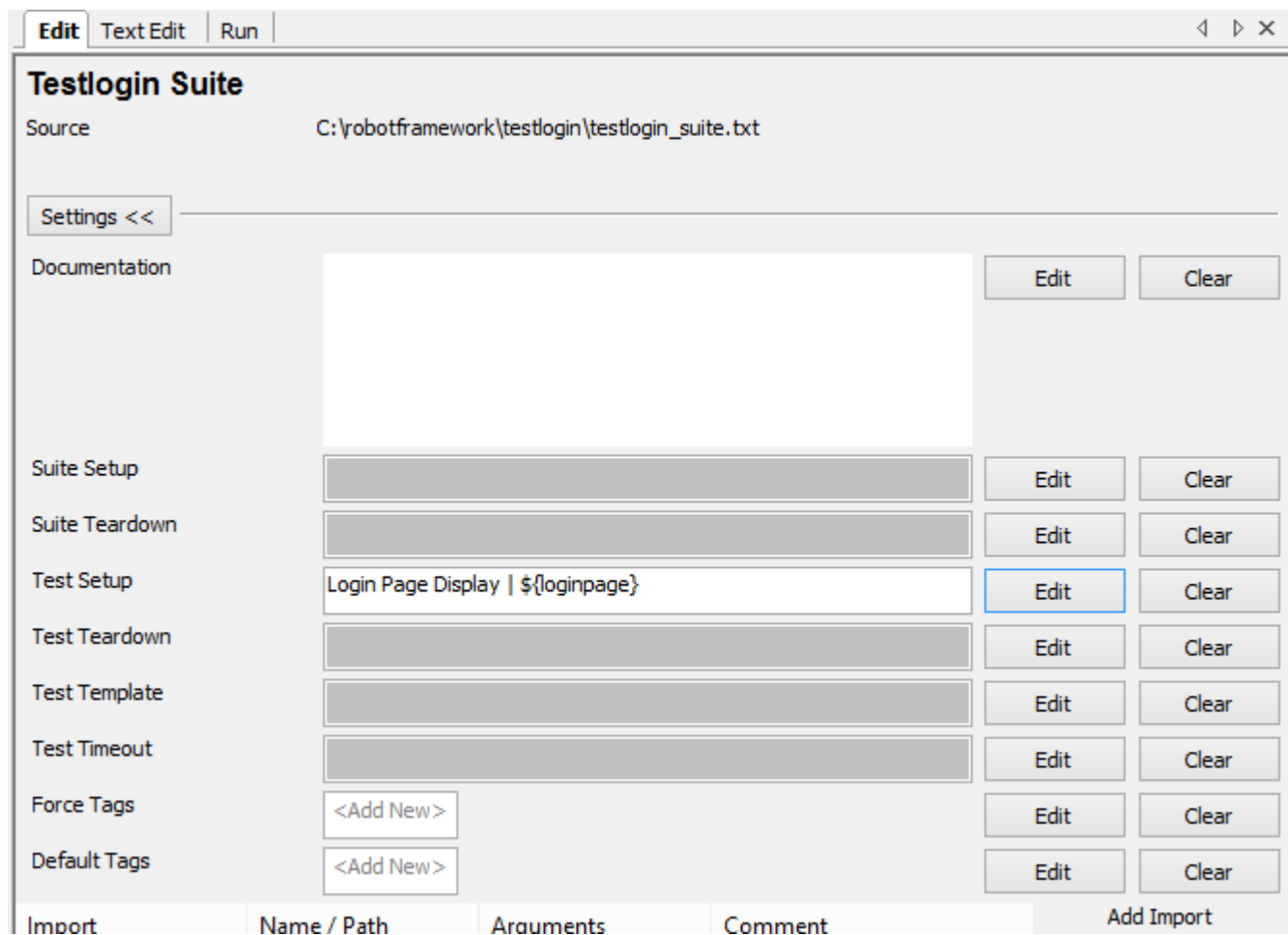
Login Page Display | \${loginpage}

Comment

This keyword is executed before every test case in this suite unless test cases override it.
Separate possible arguments with a pipe character like 'My Keyword | arg 1 | arg 2'.
Possible pipes in the value must be escaped with a backslash like '\\'.
OK Cancel

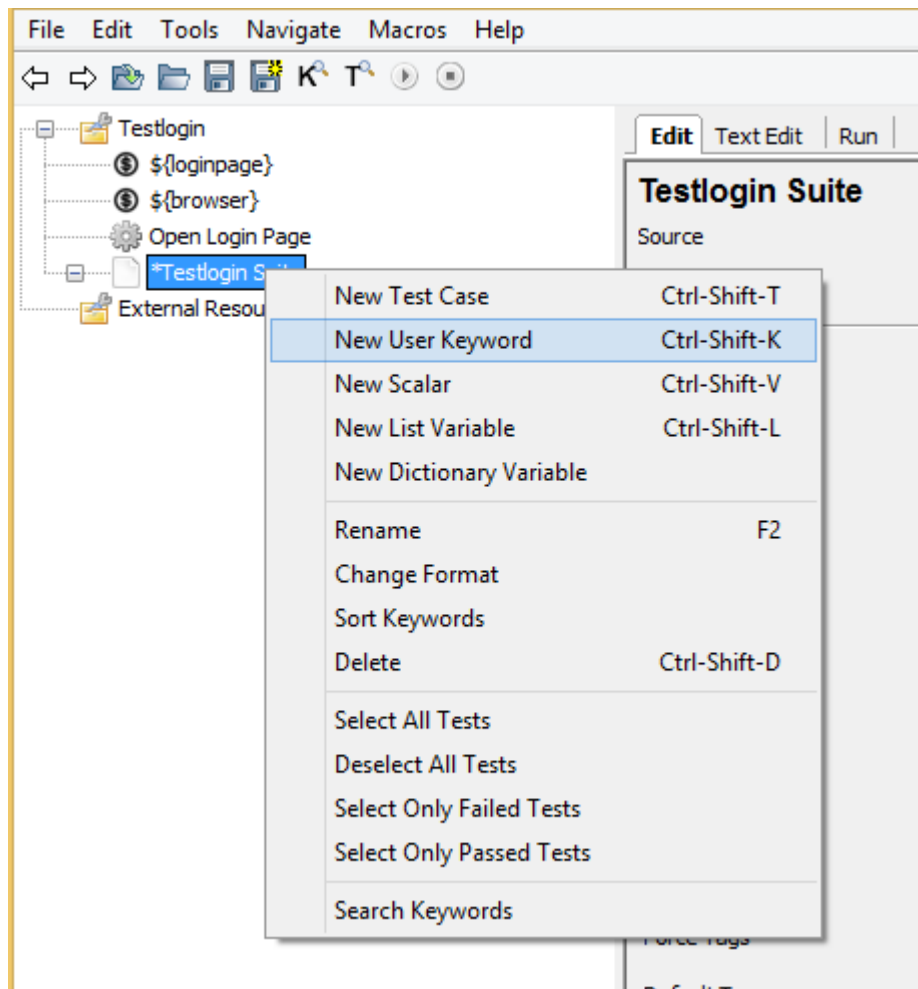
Pour la configuration du test, nous avons créé un mot-clé défini par l'utilisateur appelé Affichage de la page de connexion, qui prendra l'argument comme `${loginpage}` comme dans la capture d'écran ci-dessus.

Cliquez sur OK pour enregistrer la configuration du test.



Maintenant, nous devons créer le mot-clé utilisateur Login Page Display.

Faites un clic droit sur la suite de tests et cliquez sur Nouveau mot-clé utilisateur comme indiqué ci-dessous :



Nouveau mot-clé utilisateur affichera l'écran comme illustré ci-dessous :

New User Keyword

Name Login Page Display

Arguments \${loginpage}

Give a name and arguments for the new user keyword.
Specify the arguments separated with a pipe character like '\${arg1} | \${arg2}'.
Default values are given using equal sign and the last argument can be a list variable.
Example: '\${arg1} | \${arg2}=default value | @-rest'.
Note. You can use variable shortcuts in this field.

OK Cancel

Cliquez sur OK pour enregistrer le mot-clé.

Entrons maintenant le mot-clé dont nous avons besoin pour le mot-clé utilisateur Affichage de la page de connexion.

Login Page Display

Find Usages

Settings <<

Documentation

Edit

Clear

Tags

<Add New>

Edit

Clear

Arguments

`\${loginpage}`

Edit

Clear

Teardown

Edit

Clear

Return Value

Edit

Clear

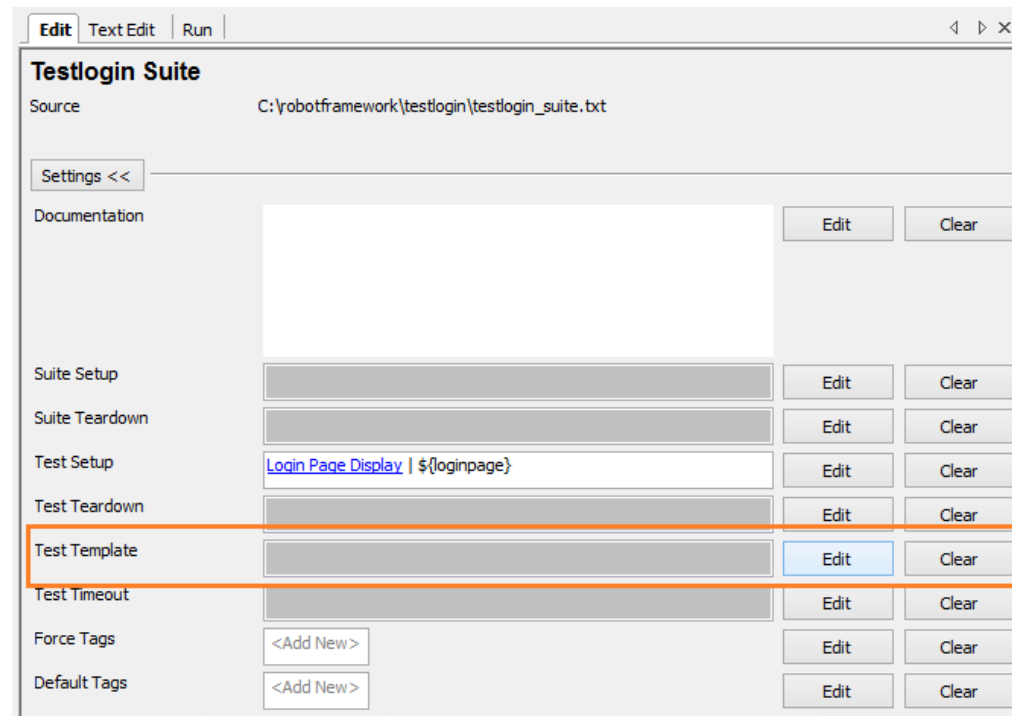
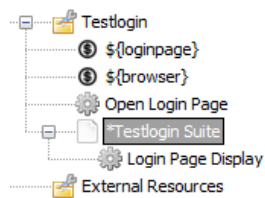
Timeout

Edit

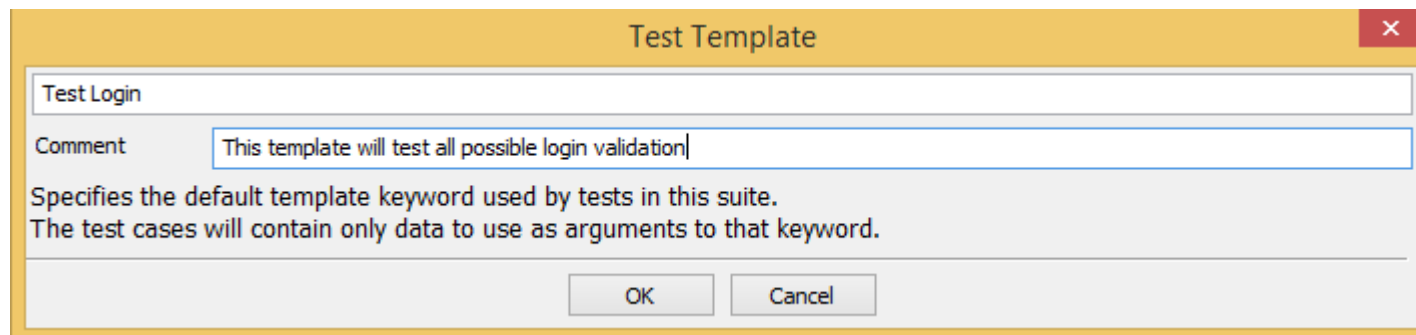
Clear

1	Go To	`\${loginpage}`	
2	Title Should Be	Login Page	
3			
4			
5			

Ici, nous voulons aller à la page de connexion et vérifier si le titre de la page correspond aux valeurs données. Maintenant, nous allons ajouter un modèle à la suite de tests et créer des cas de test basés sur les données. Pour créer un modèle, cliquez sur la suite et sur le côté droit, cliquez sur Modifier pour le modèle de test.



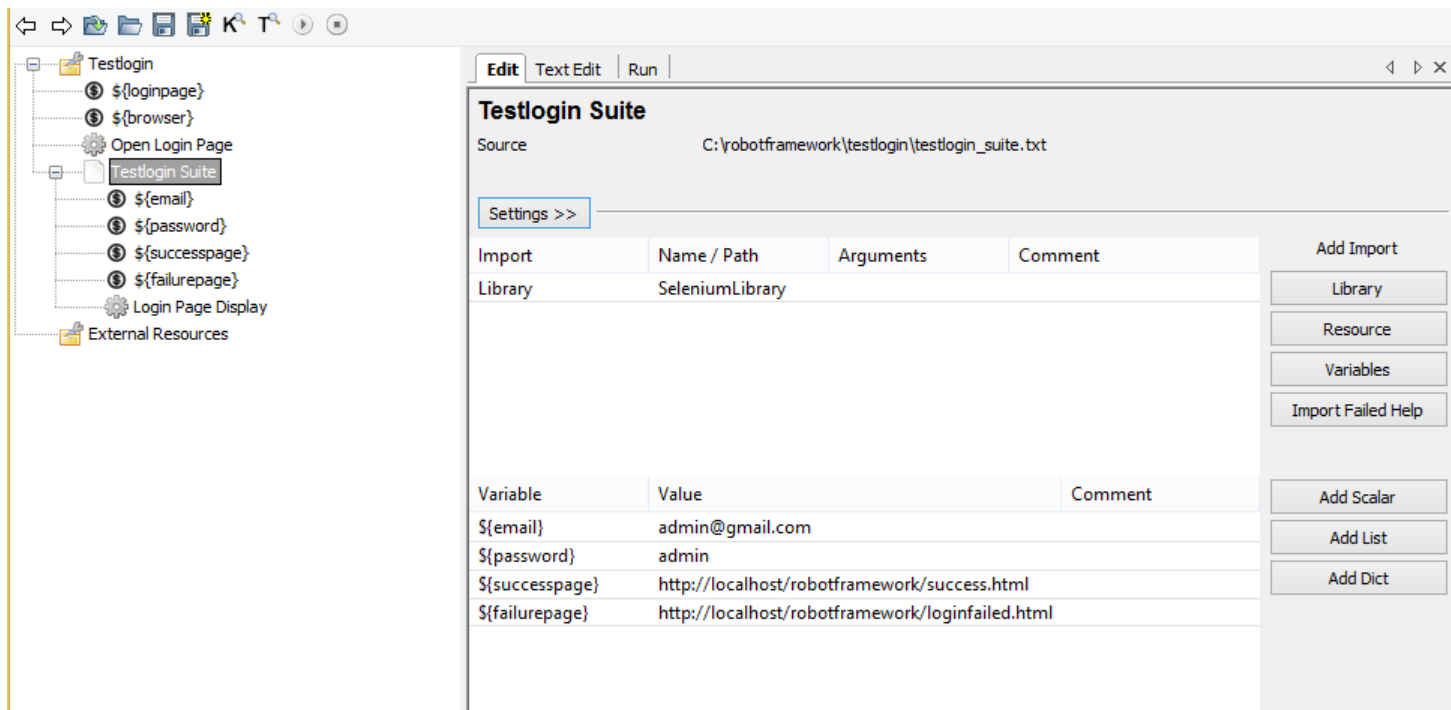
Vous serez dirigé vers l'écran suivant :



La connexion test est à nouveau un mot-clé défini par l'utilisateur. Cliquez sur OK pour enregistrer le modèle.

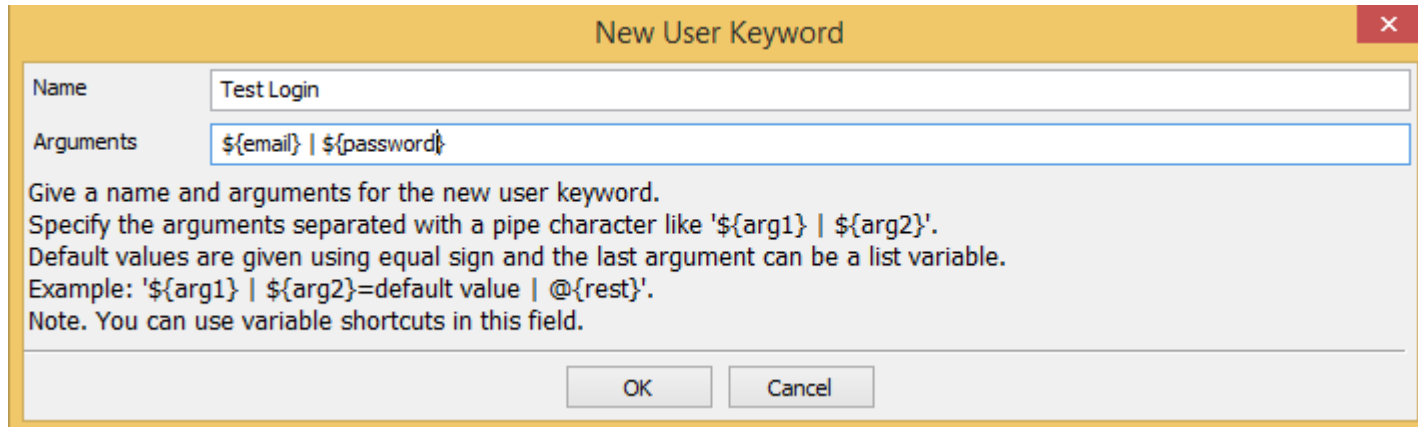
Avant de créer le mot-clé Test Login, nous avons besoin de quelques variables scalaires. Le scalaire les variables contiendront les détails de l'identifiant de l'e-mail, du mot de passe, de la page de réussite, de la page d'échec, etc.

Nous allons créer des variables scalaires pour la suite de tests comme suit :



Nous avons créé des variables scalaires email, password, successpage et failurepage comme indiqué dans la capture d'écran ci-dessus.

Maintenant, nous allons créer un mot-clé défini par l'utilisateur de connexion de test. Faites un clic droit sur la suite de tests et cliquez sur Nouveau mot-clé utilisateur.



New User Keyword

Name: Test Login

Arguments: \${email} | \${password}

Give a name and arguments for the new user keyword.
Specify the arguments separated with a pipe character like '\${arg1} | \${arg2}'.
Default values are given using equal sign and the last argument can be a list variable.
Example: '\${arg1} | \${arg2}=default value | @({rest})'.
Note. You can use variable shortcuts in this field.

OK Cancel

Cliquez sur OK pour enregistrer le mot-clé.

La capture d'écran suivante montre les mots clés saisis pour la connexion test :

Test Login

Find Usages

Settings <<

Documentation

Tags

Arguments

Teardown

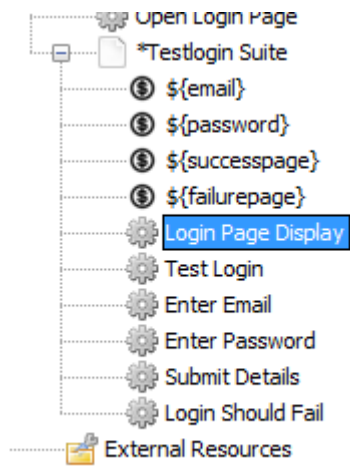
Return Value

Timeout

1	Enter Email	<code>\${email}</code>		
2	Enter Password	<code>\${password}</code>		
3	Submit Details			
4	Login Should Fail			
5				

Entrez l'e-mail, entrez le mot de passe, soumettez les détails et la connexion devrait échouer sont définis par l'utilisateur

Mots-clés, qui sont définis comme suit :



Entrez EmailRobot

Enter Email

Find Usages

Settings <<

Documentation

Edit

Clear

Tags

<Add New>

Edit

Clear

Arguments

`\${email}`

Edit

Clear

Teardown

Edit

Clear

Return Value

Edit

Clear

Timeout

Edit

Clear

1	Input Text	email	`\${email}`	
2				
3				
4				
5				
6				

Entrez le mot de passeRobot

Enter Password

Find Usages

Settings <<

Documentation

Edit Clear

Tags

<Add New>

Edit Clear

Arguments

`\${password}`

Edit Clear

Teardown

Edit Clear

Return Value

Edit Clear

Timeout

Edit Clear

1	Input Text	passwd	`\${password}`	
2				
3				
4				
5				
6				

Soumettre les detailsRobot

Submit Details

Find Usages

Settings <<

Documentation

Edit

Clear

Tags

<Add New>

Edit

Clear

Arguments

Edit

Clear

Teardown

Edit

Clear

Return Value

Edit

Clear

Timeout

Edit

Clear

1	Click Button	btnsubmit	
2			
3			
4			
5			
6			

La connexion devrait échouer

Login Should Fail

Find Usages

Settings <<

Documentation

Tags

Arguments

Teardown

Return Value

Timeout

Edit

Clear

<Add New>

Edit

Clear

Edit

Clear

Edit

Clear

Edit

Clear

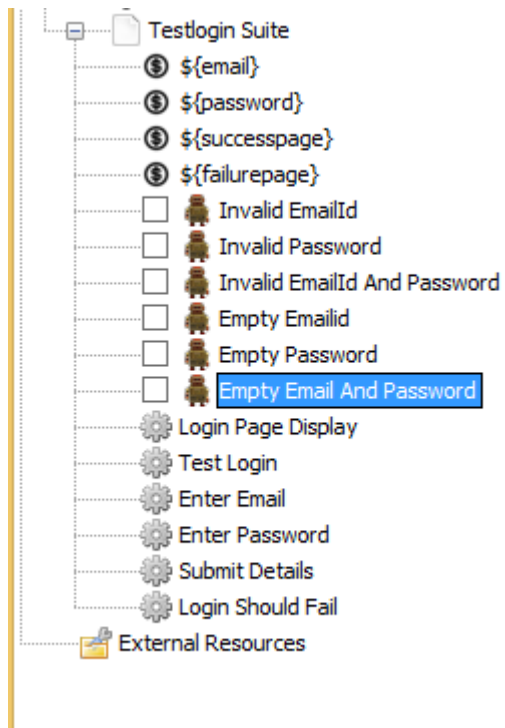
Edit

Clear

1	Location Should Be	\${failurepage}		
2	Title Should Be	Login Failed		
3				
4				
5				
6				
7				

Maintenant, nous allons écrire des cas de test, qui prendront différents identifiants de messagerie et mots de passe pour le modèle créé.

Voici une liste de cas de test :



Identifiant de messagerie invalide Cas de test

	Email	Password		
1	abcd@gmail.com	<code>#{password}</code>		
2				
3				
4				
5				
6				

L'email est passé avec les valeurs abcd@gmail.com et \${password} est le mot de passe stocké dans la variable.

Mot de passe incorrect

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

Identifiant de messagerie et mot de passe invalides

	Email	Password		
1	invalid	invalid		
2				
3				
4				
5				
6				

Identifiant de messagerie vide

	Email	Password		
1	<code>\${EMPTY}</code>	<code>\${password}</code>		
2				
3				
4				
5				
6				

Mot de passe vide

	Email	Password		
1	<code>\${email}</code>	<code>\${EMPTY}</code>		
2				
3				
4				
5				
6				

E-mail et mot de passe vides

	Email	Password		
1	<code>\${EMPTY}</code>	<code>\${EMPTY}</code>		
2				
3				
4				
5				
6				

Maintenant, nous en avons terminé avec les cas de test et pouvons exécuter la même chose.

Accédez à l'onglet Exécuter et cliquez sur Démarrer pour exécuter les scénarios de test.

Execution Profile: **pybot** Report Log ☐ Autosave ☐ Pause on failure ☐ Show message log

▶ Start ◀ Stop ⏸ Pause ▶ Continue ⏮ Next ⏭ Step over

Arguments:

☐ Only run tests with these tags ☐ Skip tests with these tags

elapsed time: 0:00:16 pass: 6 fail: 0

```

command: pybot.bat --argumentfile c:\users\kamat\appdata\local\temp\RIDEsz9smo.d\argfile.txt --listener C:\Python27\lib\site-packa
=====
Testlogin
=====
Testlogin.Testlogin Suite
=====
Invalid EmailId | PASS |
-----
Invalid Fassword | PASS |
-----
Invalid EmailId And Password | PASS |
-----
Empty Emailid | PASS |
-----
Empty Password | PASS |
-----
Empty Email And Password | PASS |
-----
Testlogin.Testlogin Suite | PASS |
6 critical tests, 6 passed, 0 failed
6 tests total, 6 passed, 0 failed
=====
Testlogin | PASS |
6 critical tests, 6 passed, 0 failed
6 tests total, 6 passed, 0 failed
=====
Output: c:\users\appdata\local\temp\RIDEsz9smo.d\output.xml
Log: c:\users\appdata\local\temp\RIDEsz9smo.d\log.html
Report: c:\users\appdata\local\temp\RIDEsz9smo.d\report.html

test finished 20181027 18:06:17

```