Integration with Ranorex

What you'll learn

- Learn Ranorex core concepts
- Learn how creating test automation test cases looks like, at high-level, using Ranorex Studio
- Run the tests and push the test report to Xray
- Validate that the test results are available in Jira
- Learn how to assess the impacts on related user stories in Jira, using Xray

Overview

Ranorex can be seen as a keyword-driven framework for implementing GUI test automation across a broad set of technologies, including desktop, web, and mobile. It's a codeless test automation solutiion. Ranorex provides comprehensive support for test automation as detailed in its core features. Data-driven testing is also supported.

Ranorex provides Ranorex Studio, the main application to implement and organize automated test scripts. Ranorex Studio has multiple components, including a recorder (Ranorex Recorder) and a object /element identifier (Ranorex Spy).

In this article we'll highlight some of Ranorex core concepts and see how to have visilbity of test automation results in Jira, using Xray.

Integrating with Xray is straightforward, using JUnit XML reports that Ranorex can generate. To get the integration done, you just need to get that working.

Ranorex concepts and mapping to Xray

Ranorex Studio provides a complete GUI for implementing automated tests. Therefore, we find some concepts typical in IDEs (e.g. solution, project).

There are some specifics though, related to test automation.

The only concept with a direct mapping to Xray will be Ranorex' Test case which will be abstracted as a Xray Test issue (unstructured/generic).

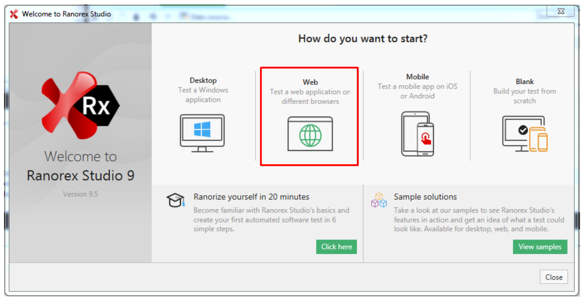
Ranorex concept	Description	Xray concept
Solution	In Ranorex Studio, a solution is the top-level container that contains all other test files. Solutions are organized into one or more projects. Whenever creating a solution, we may identify the type of application we aim to test (e.g. desktop, web, mobile). A solution has always a "test suite" project.	
Project	A tailored place to organize test files. A project can be of one of several types, including "test suite", which offer different capabilities.	
Test suite	The test suite is where you build, organize, and run your tests in Ranorex Studio. A test suite consists primarily of test cases.	Doesn't exist as an entity. Will be visible and part of the definition of each Test issue.
Test case	A test, composed of Modules, which in turn are composed of Actions.	Test issue.

Module	A modularized sequence of actions, that have a certain goal. Can be seen as a grouped sequence of steps in a test case. Modules can be reused between Test cases.	
Action	A step, inside a Module. Can be a mouse/keyboard interaction or a validation.	
Validation	An assertion, a validation.	
Repository	A Repository contains Repository Items (i.e. UI elements) organized in a tree-like structure. UI elements that contain other UI elements are represented as folders in the repository, with app folders acting as top-level elements and rooted folders as children.	
Repository item	A representation of a user interface (UI) element used in a test. Each repository item has a name and is defined by its RanoreX Path (i.e. path).	

The overall result of a given test case will be available on the Test Run that will be created in Xray and associated to the corresponding Test issue.

Prerequisites

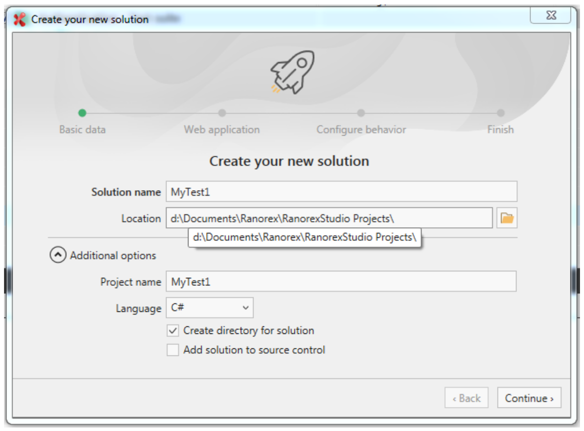
We start by creating a solution, web-based one as we aim to test a web site.


Welcome to Ranorex Studio
Welcome to
Ranorex Studio 9
Version 9.5
How do you want to start?
Desktop
Test a Windows application
Web
Test a web application on different browsers
Mobile
Test a mobile app on iOS or Android
Blank
Build your test from scratch
Ranorize yourself in 20 minutes
Click here
Sample solutions
View samples
Close

Become familiar with Ranorex Studio's basics and create your first automated software test in 6 simple steps.

Take a look at our samples to see Ranorex Studio's features in action and get an idea of what a test could look like. Available for desktop, web, and mobile.

Under "Additional options" we can customise the language of the underlying code that Ranorex will use to support the test automation.



Solution name: MyTest1

Location: d:\Documents\Ranorex\RanorexStudio Projects\

Project name: MyTest1

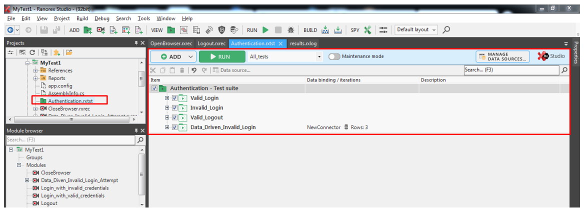
Language: C#

[✓] Create directory for solution

[ ] Add solution to source control

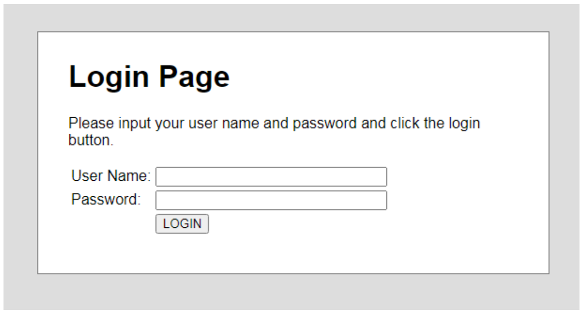
A project of type "test suite" will be created.

It contains one Test Suite, where we can create and manage our test cases. We may rename this Test Suite to have a more meaningful name (e.g. "Authentication").



Implementing automated tests

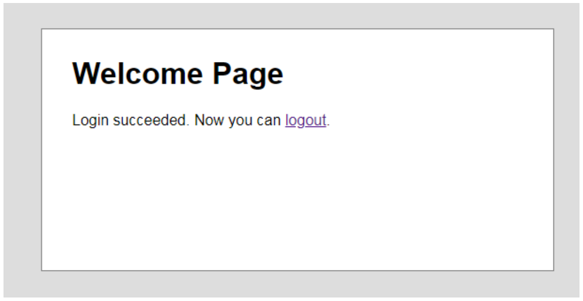
We will implement some automated tests for a dummy web site, providing an authentication mechanism that we aim to check, namely the login and logout features.



Login Page

Please input your user name and password and click the login button.

User Name:
Password:
LOGIN



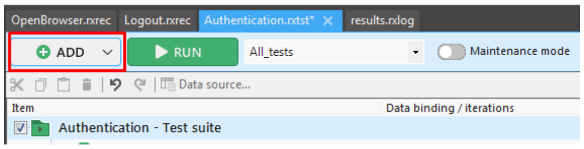
Welcome Page

Login succeeded. Now you can logout.

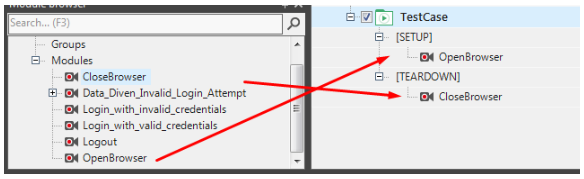
Our tests, part of a test suite, include these scenarios:

- valid login
- invalid login
- valid logout
- invalid login (data-driven scenario)

To add some Test cases to the Test Suite, we can use "Add" button.

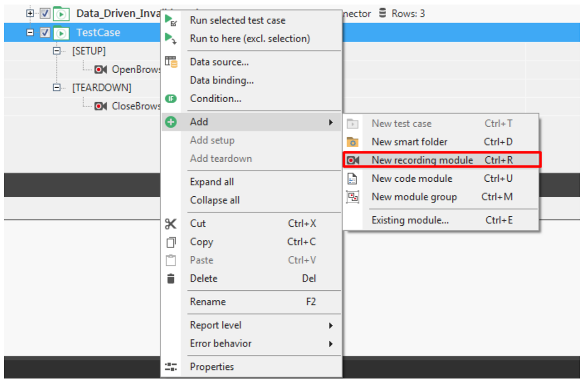


We can rename the Test case (i.e. from TestCase to whatever describes it). We then add a Setup and a Teardown section and add the OpenBrowser and CloseBrowser modules, to each section respectively. Opening the OpenBrowser module, by double-clicking on it, will allow us to set the URL to be used.

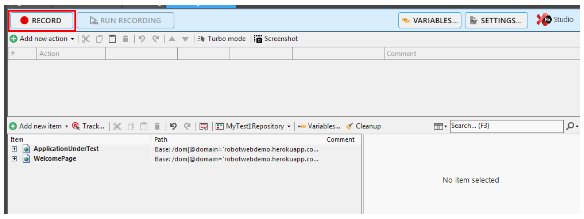


We can then use Ranorex Recorder to create a "module" (i.e. a set of sequential actions and/or validations on UI elements).

This will be used to implement interactions with our web site, without having to code.

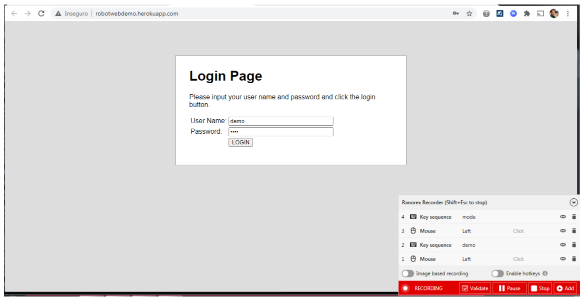


We choose "Record" and then we're redirected to the browser, where we can perform actions which will be recorded.



Item	Path	Comment
ApplicationUnderTest	Base: /dom[@domain='robotwebdemo.herokuapp.co...	
WelcomePage	Base: /dom[@domain='robotwebdemo.herokuapp.co...	

While recording the actions of our module, we can remove some that may be added by accident for example. We can also pause and stop recording.


Inseguro | robotwebdemo.herokuapp.com
Ranorex Recorder (Shift+Esc to stop)
Image based recording
Enable hotkeys
RECORDING
Validate
Pause
Stop
Add

Login Page

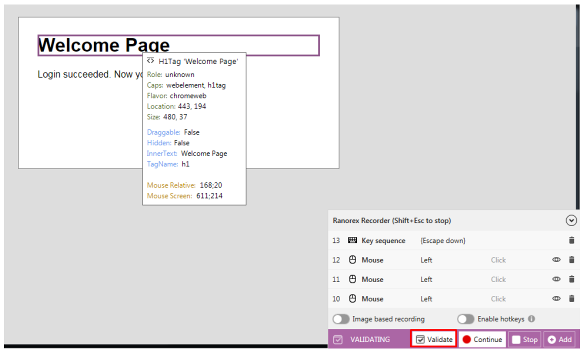
Please input your user name and password and click the login button.

User Name: demo
Password: ••••
LOGIN

4	Key sequence	mode	
3	Mouse	Left	Click
2	Key sequence	demo	
1	Mouse	Left	Click

In order to implement a test, we need to add at least one validation.

For that, we choose "Validate" and move the mouse hover the element we want to validate and click on it.



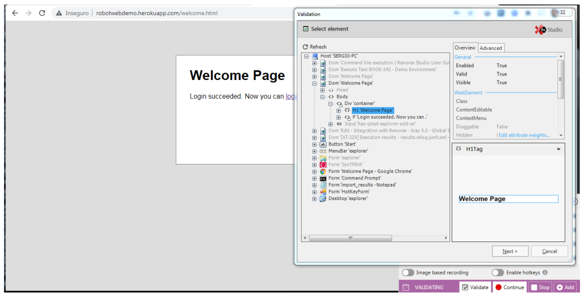
Welcome Page

Login succeeded. Now y

H1Tag 'Welcome Page'
Role: unknown
Caps: webelement, h1tag
Flavor: chromeweb
Location: 443, 194
Size: 480, 37
Draggable: False
Hidden: False
InnerText: Welcome Page
TagName: h1
Mouse Relative: 168;20
Mouse Screen: 611;214

Ranorex Recorder (Shift+Esc to stop)

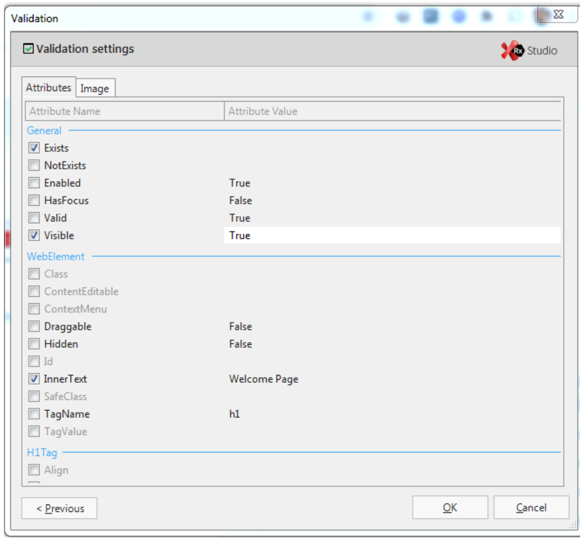
13	Key sequence	{Escape down}		
12	Mouse	Left	Click	
11	Mouse	Left	Click	
10	Mouse	Left	Click	



Welcome Page

Login succeeded. Now you can log

A validation dialog enable us to perform multiple asserts at the same time, on a given element (e.g. element exists, is visible, contains a given text). Each assert/validation will be created as a "Validate" action in our module.

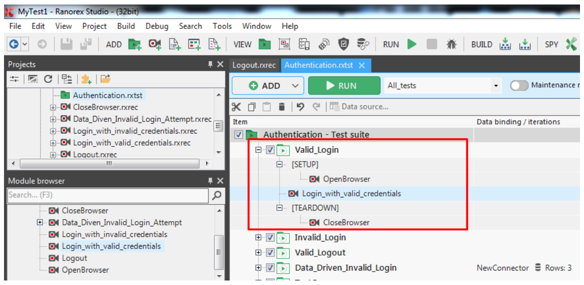


Validation

Validation settings

Attribute Name	Attribute Value
General	
[✓] Exists	
[ ] NotExists	
[ ] Enabled	True
[ ] HasFocus	False
[ ] Valid	True
[✓] Visible	True
WebElement	
[ ] Class	
[ ] ContentEditable	
[ ] ContextMenu	
[ ] Draggable	False
[ ] Hidden	False
[ ] Id	
[✓] InnerText	Welcome Page
[ ] SafeClass	
[ ] TagName	h1
[ ] TagValue	
H1Tag	
[ ] Align	

A possible "Valid Login" test case could be composed of a setup section to open the browser (i.e. using the "OpenBrowser" module), a recorded module where we enter the credentials and validate the welcome page, followed by the close browser instruction (as part of the CloseBrowser module).



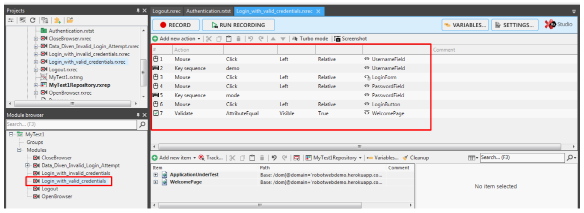
MyTest1 - Ranorex Studio - (32bit)

```
Authentication.rxtst
CloseBrowser.rxrec
Data_Diven_Invalid_Login_Attempt.rxrec
Login_with_invalid_credentials.rxrec
Login_with_valid_credentials.rxrec
Logout.rxrec
```

```
CloseBrowser
Data_Diven_Invalid_Login_Attempt
Login_with_invalid_credentials
Login_with_valid_credentials
Logout
OpenBrowser
```

```
Authentication - Test suite
  Valid_Login
    [SETUP]
      OpenBrowser
    Login_with_valid_credentials
    [TEARDOWN]
      CloseBrowser
  Invalid_Login
  Valid_Logout
  Data_Driven_Invalid_Login          NewConnector   Rows: 3
```

The "Login_with_valid_credentials" module could be composed of these actions, depending on the actions you defined earlier.

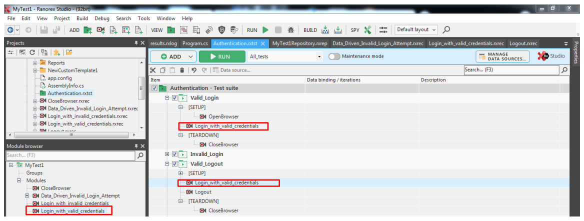


#	Action					Comment
1	Mouse	Click	Left	Relative	UsernameField	
2	Key sequence	demo			UsernameField	
3	Mouse	Click	Left	Relative	LoginForm	
4	Mouse	Click	Left	Relative	PasswordField	
5	Key sequence	mode			PasswordField	
6	Mouse	Click	Left	Relative	LoginButton	
7	Validate	AttributeEqual	Visible	True	WelcomePage	

Item	Path	Comment
ApplicationUnderTest	Base: /dom[@domain='robotwebdemo.herokuapp.co...	
WelcomePage	Base: /dom[@domain='robotwebdemo.herokuapp.co...	

We may decide to implement additional tests, reusing existing modules.

As an example, a test case that checks the valid logout procedure (i.e. "Valid_Logout") can use the "Login_with_valid_credentials" module as the first macro step, before executing the module and its actions that perform the logout and verify its result ((i.e. "Logout" module).



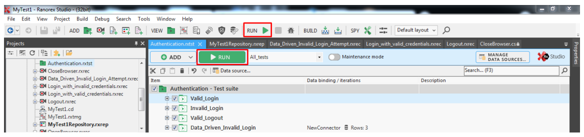
```
Authentication - Test suite
  Valid_Login
    [SETUP]
      OpenBrowser
    Login_with_valid_credentials
    [TEARDOWN]
      CloseBrowser
  Invalid_Login
  Valid_Logout
    [SETUP]
    Login_with_valid_credentials
    Logout
    [TEARDOWN]
      CloseBrowser
```

Running the tests

Before running the tests, you have to build the current project which will produce a executable file. Tests are in fact performed by this executable file and not by Ranorex Studio itself.

Running tests using Ranorex Studio

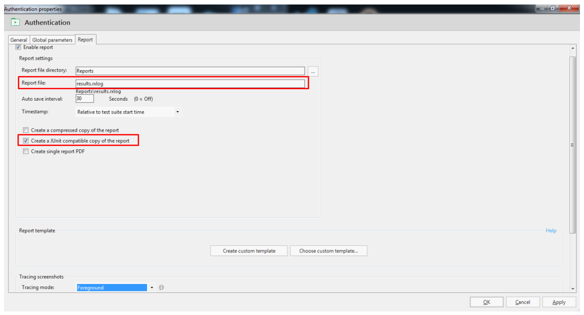
Test cases can be "run" from within the Ranorex Studio UI; a build happens in the background, if needed.



```
Authentication - Test suite
  Valid_Login
  Invalid_Login
  Valid_Logout
  Data_Driven_Invalid_Login    NewConnector    Rows: 3
```

Whenever running, Ranorex Studio uses a run configuration to know which test cases to run; in the previous screenshot it is called "All_tests" and contains all test cases, since they're selected.

In the properties of our test suite we can customize the report file name, the reports directory, and enable the JUnit XML report which Xray can later on process.



Authentication properties

Authentication

General | Global parameters | Report

[✓] Enable report

Report settings

Report file directory: Reports

Report file: results.rxlog

Reports\results.rxlog

Auto save interval: 30 Seconds (0 = Off)

Timestamp: Relative to test suite start time

[ ] Create a compressed copy of the report

[✓] Create a JUnit compatible copy of the report

[ ] Create single report PDF

Report template

Create custom template | Choose custom template...

Tracing screenshots

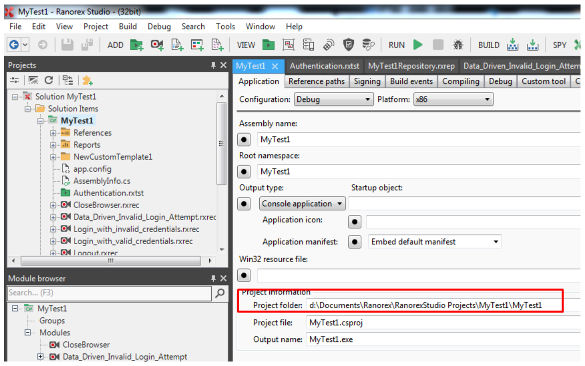
Tracing mode: Foreground

OK | Cancel | Apply

Running tests from the command-line

Tests can be run from the command-line, by calling the executable built by Ranorex Studio.

The execute can be found inside the project folder (obtainable by looking at the project properties), either in the `bin\Debug` or `bin\Release` folder.


MyTest1 - Ranorex Studio - (32bit)
File Edit View Project Build Debug Search Tools Window Help
ADD VIEW RUN BUILD SPY
Projects
Solution MyTest1
Solution Items
MyTest1
References
Reports
NewCustomTemplate1
app.config
AssemblyInfo.cs
Authentication.rxtst
CloseBrowser.rxrec
Data_Driven_Invalid_Login_Attempt.rxrec
Login_with_invalid_credentials.rxrec
Login_with_valid_credentials.rxrec
Logout.rxrec
Module browser
MyTest1
Groups
Modules
CloseBrowser
Data_Driven_Invalid_Login_Attempt
MyTest1 Authentication.rxtst MyTest1Repository.rxrep Data_Driven_Invalid_Login_Attem
Application Reference paths Signing Build events Compiling Debug Custom tool

Configuration: Debug Platform: x86
Assembly name: MyTest1
Root namespace: MyTest1
Output type: Console application
Startup object:
Application icon:
Application manifest: Embed default manifest
Win32 resource file:
Project information
Project folder: d:\Documents\Ranorex\RanorexStudio Projects\MyTest1\MyTest1
Project file: MyTest1.csproj
Output name: MyTest1.exe

To run the tests, we execute the file and pass some arguments to enable the JUnit XML report, customize the report file base file name; (use /help to find available options).

example of a shell script to run the tests

```
MyTest1.exe /junit /reportfile:results
```

In this case, the report will be stored in the current directory and wil be named `results.rxlog.junit.xml`.

Integrating with Xray

In order to have visibility of our test automation results in Jira, we need to generate a JUnit XML report whenever running the tests, that can then be submitted to Xray as shown in the previous section.

To submit the report to Xray, we can use our favourite CI/CD tool or a simple script for that.

Once you have the report file available you can upload it to Xray through a request to the REST API endpoint for JUnit. To do that, follow the first step in the instructions in v1 or v2 (depending on your usage) to obtain the token we will be using in the subsequent requests.

```
curl -H "Content-Type: application/json" -X POST --data '{ "client_id":
"CLIENTID","client_secret": "CLIENTSECRET" }'  https://xray.cloud.getxray.
app/api/v2/authenticate
```

Once you have the token we will use it in the API request with the definition of some common fields on the Test Execution, such as the target project, project version, linked test plan, etc.

```
curl -H "Content-Type: text/xml" -X POST -H "Authorization: Bearer
$token"  --data @"results.rxlog.junit.xml" https://xray.cloud.getxray.app
/api/v2/import/execution/junit?projectKey=XT
```

Sample batch (.bat) script to import results to Xray

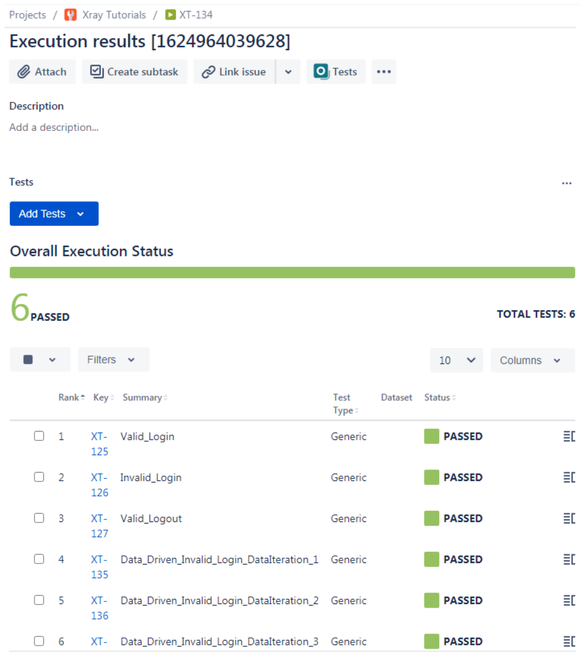
```
@echo off
set client_id="DA2258616A5944198E9BE44A9A000000"
set client_secret="
5bae1aa5b49e5d263781da54ba55cc7deebd7840c68fe2fdfd2a077768000000"
set project_key="XT"
set report_file="Reports\results.rxlog.junit.xml"

set jira_base_url="https://xray.cloud.getxray.app/api/v2"


for /f %%i in ('curl -H "Content-Type: application/json" -X POST --data "{
\"client_id\": \"%client_id%\",\"client_secret\": \"%client_secret%\" }"
https://xray.cloud.getxray.app/api/v2/authenticate') do set token=%%i
rem echo %token%
curl -H "Content-Type: text/xml" -X POST -H "Authorization: Bearer %
token%"  --data @"%report_file%" "%jira_base_url%/import/execution/junit?
projectKey=%project_key%"
```

Sample PowerShell script to import results to Xray

```
try {
 $client_id = 'DA2258616A5944198E9BE44A9A000000'
 $client_secret =
'5bae1aa5b49e5d263781da54ba55cc7deebd7840c68fe2fdfd2a077768000000'
 $project_key = 'XT'
 $report_file = 'results.rxlog.junit.xml'

 $jira_base_url = 'https://xray.cloud.getxray.app/api/v2'
 $json = @"
 {
  "client_id": "$($client_id)", "client_secret": "$($client_secret)"
 }
"@

 $uri = "$($jira_base_url)/authenticate"
 $res = Invoke-WebRequest -Uri $uri -Body $json -Method POST -ContentType
"application/json"
 $token = $res -replace '"',''
 #write-host $token


 $fileContent = Get-Content -Path $report_file -Raw
 $uri = "$($jira_base_url)/import/execution/junit?
projectKey=$($project_key)"
 $res = Invoke-WebRequest -ContentType "text/xml" -Uri $uri -Body
$fileContent -Method POST -Headers @{"Authorization" = "Bearer $token"}
 write-host $res
}
catch {
 write-host $_.Exception.Message
}
```

After submitting the test automation report, a Test Execution will be created in Jira, containing the results for each Test case.

A Test issue will be auto-provisioned, unless it already exists, per each Test Case.



Projects / Xray Tutorials / XT-134

Execution results [1624964039628]

Attach | Create subtask | Link issue | Tests

Description

Add a description...

Tests

Add Tests

Overall Execution Status

6 PASSED

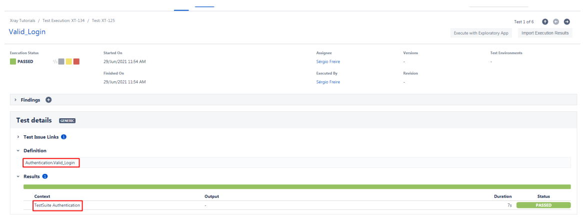
TOTAL TESTS: 6

Filters | 10 | Columns

	Rank	Key	Summary	Test Type	Dataset	Status
☐	1	XT-125	Valid_Login	Generic		🟩 PASSED
☐	2	XT-126	Invalid_Login	Generic		🟩 PASSED
☐	3	XT-127	Valid_Logout	Generic		🟩 PASSED
☐	4	XT-135	Data_Driven_Invalid_Login_DataIteration_1	Generic		🟩 PASSED
☐	5	XT-136	Data_Driven_Invalid_Login_DataIteration_2	Generic		🟩 PASSED
☐	6	XT-	Data_Driven_Invalid_Login_DataIteration_3	Generic		🟩 PASSED

The Test Suite name along with the Test Case name will be used as unique identifier for the Generic Test that will be created.

In the Test Run details of the corresponding Test in Xray, we can see this information along with the overall result. The original Test Suite name appears also as a context.



Xray Tutorials / Test Execution: XT-134 / Test: XT-125

Valid_Login

Test 1 of 6

Execute with Exploratory App | Import Execution Results

Execution Status	Started On	Assignee	Versions	Test Environments
🟩 PASSED	29/Jun/2021 11:54 AM	Sérgio Freire	-	-
	Finished On	**Executed By**	**Revision**	
	29/Jun/2021 11:54 AM	Sérgio Freire	-	

Findings

Test details GENERIC

Test Issue Links 1

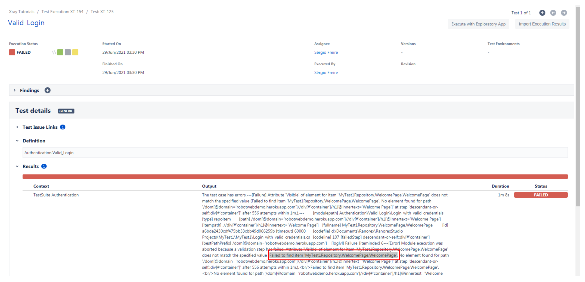
Definition

Authentication.Valid_Login

Results 1

Context	Output	Duration	Status
TestSuite Authentication	-	7s	PASSED

When a test case fails, the corresponding Test Run in Xray will show details about the repository item that failed; its full identifier includes the name of the repository followed by the hierarchical location of the repository item.



Xray Tutorials / Test Execution: XT-154 / Test: XT-125

Valid_Login

Test 1 of 1

Execute with Exploratory App | Import Execution Results

Execution Status	Started On	Assignee	Versions	Test Environments
🟥 FAILED	29/Jun/2021 03:30 PM	Sérgio Freire	-	-
	Finished On	**Executed By**	**Revision**	
	29/Jun/2021 03:30 PM	Sérgio Freire	-	

Findings

Test details GENERIC

Test Issue Links 1

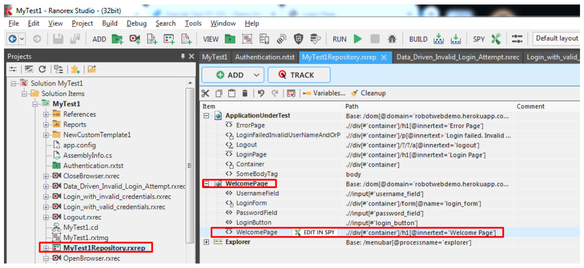
Definition

Authentication.Valid_Login

Results 1

Context	Output	Duration	Status
TestSuite Authentication	The test case has errors.---[Failure] Attribute 'Visible' of element for item 'MyTest1Repository.WelcomePage.WelcomePage' does not match the specified value (Failed to find item 'MyTest1Repository.WelcomePage.WelcomePage'. No element found for path '/dom[@domain='robotwebdemo.herokuapp.com']//div[#'container']/h1[@innertext='Welcome Page']' at step 'descendant-or-self::div[#'container']' after 556 attempts within 1m.).--- [modulepath] Authentication\Valid_Login\Login_with_valid_credentials [type] repoitem [path] /dom[@domain='robotwebdemo.herokuapp.com']//div[#'container']/h1[@innertext='Welcome Page'] [itempath] .//div[#'container']/h1[@innertext='Welcome Page'] [fullname] MyTest1Repository.WelcomePage.WelcomePage [id] a6bde2430cdf475bb33cbb49d066259b [timeout] 60000 [codefile] d:\Documents\Ranorex\RanorexStudio Projects\MyTest1\MyTest1\Login_with_valid_credentials.cs [codeline] 107 [failedStep] descendant-or-self::div[#'container'] [bestPathPrefix] /dom[@domain='robotwebdemo.herokuapp.com'] [loglvl] Failure [itemindex] 6---[Error] Module execution was aborted because a validation step has failed. Attribute 'Visible' of element for item 'MyTest1Repository.WelcomePage.WelcomePage' does not match the specified value (Failed to find item 'MyTest1Repository.WelcomePage.WelcomePage'. No element found for path '/dom[@domain='robotwebdemo.herokuapp.com']//div[#'container']/h1[@innertext='Welcome Page']' at step 'descendant-or-self::div[#'container']' after 556 attempts within 1m.).<br/>Failed to find item 'MyTest1Repository.WelcomePage.WelcomePage'.<br/>No element found for path '/dom[@domain='robotwebdemo.herokuapp.com']//div[#'container']/h1[@innertext='Welcome	1m 8s	FAILED

The failed element was in this case the "Welcome Page" header, in the Welcome Page, identified by "My Test1Repository.WelcomePage.WelcomePage".


MyTest1 - Ranorex Studio - (32bit)
File Edit View Project Build Debug Search Tools Window Help
ADD VIEW RUN BUILD SPY Default layout
Projects
MyTest1 Authentication.rxtst MyTest1Repository.rxrep Data_Driven_Invalid_Login_Attempt.rxrec Login_with_valid_
ADD TRACK
Variables... Cleanup

Solution MyTest1
Solution Items
MyTest1
References
Reports
NewCustomTemplate1
app.config
AssemblyInfo.cs
Authentication.rxtst
CloseBrowser.rxrec
Data_Driven_Invalid_Login_Attempt.rxrec
Login_with_invalid_credentials.rxrec
Login_with_valid_credentials.rxrec
Logout.rxrec
MyTest1.cd
MyTest1.rxtmg
MyTest1Repository.rxrep
OpenBrowser.rxrec

Item	Path	Comment
ApplicationUnderTest	Base: /dom[@domain='robotwebdemo.herokuapp.co...	
ErrorPage	.//div[#'container']/h1[@innertext='Error Page']	
LoginFailedInvalidUserNameAndOrP	.//div[#'container']/p[@innertext>'Login failed. Invalid ...	
Logout	.//div[#'container']/?/?/a[@innertext='logout']	
LoginPage	.//div[#'container']/h1[@innertext='Login Page']	
Container	.//div[#'container']	
SomeBodyTag	body	
WelcomePage	Base: /dom[@domain='robotwebdemo.herokuapp.co...	
UsernameField	.//input[#'username_field']	
LoginForm	.//div[#'container']/form[@name='login_form']	
PasswordField	.//input[#'password_field']	
LoginButton	.//input[#'login_button']	
WelcomePage EDIT IN SPY	.//div[#'container']/h1[@innertext='Welcome Page']	
Explorer	Base: /menubar[@processname='explorer']	

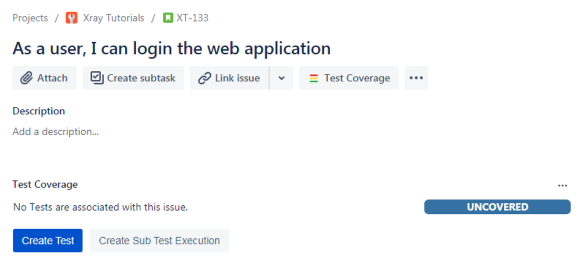
Tips

Seeing the impacts of test automation results on user stories or requirements

After uploading the test automation results, users can link the Test issues to existing user stories or requirements. That will enable users to track coverage and thus assess if user stories are covered by automated test scripts and if based on that, the corresponding user story can be considered to OK or NOK.

> Please remember that coverage is an heuristic but that can still be quite helpful to assess the readiness of user stories, individually or at the release level.

Assuming we have a user story (new or existing), we can then link it to the Tests that correspond to the Test Cases implemented using Ranorex.

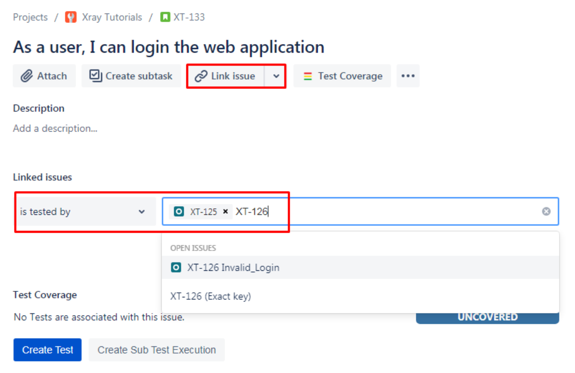


As a user, I can login the web application

Description
Add a description...

Test Coverage
No Tests are associated with this issue.

We can do that right from the user story issue screen, using the "Link issue" action and the "is tested by" issue link . Then, we select the Tests that were auto-provisioned earlier on upon the first import of the test report.



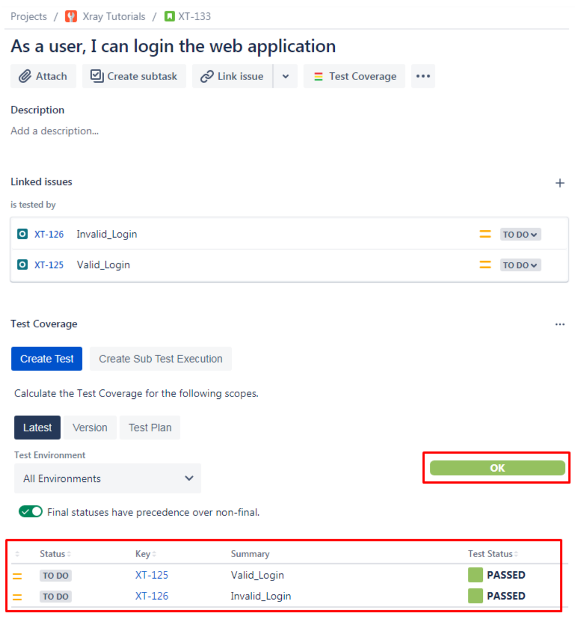
As a user, I can login the web application

Description
Add a description...

Linked issues
is tested by
XT-125 × XT-126
XT-126 Invalid_Login
XT-126 (Exact key)

Test Coverage
No Tests are associated with this issue.

Finally, on the Test Coverage panel we can see the latest test results right from the user story issue screen along with the calculated coverage information for the user story.


Projects / Xray Tutorials / XT-133

As a user, I can login the web application

Description

Add a description...

Linked issues

is tested by

XT-126	Invalid_Login	=	TO DO
XT-125	Valid_Login	=	TO DO

Test Coverage

Calculate the Test Coverage for the following scopes.

Test Environment: All Environments

Final statuses have precedence over non-final.

	Status	Key	Summary	Test Status
=	TO DO	XT-125	Valid_Login	🟩 PASSED
=	TO DO	XT-126	Invalid_Login	🟩 PASSED

Any additional imports of results, will appear automatically reflected on the user story issue screen as the Tests are already linked to the user story.

Run iterations and data-driven tests

Ranorex Studio has support for run iterations and data-driven tests.

These are two different concepts; while run iterations are just a way to run the same test case multiple times by executing the exact same modules and actions, data-driven tests will impact on the action being performed (e.g. for exercising the same test case but with different inputs).

It's possible to have visibility of the corresponding test results in Jira using Xray but some care should be taken.

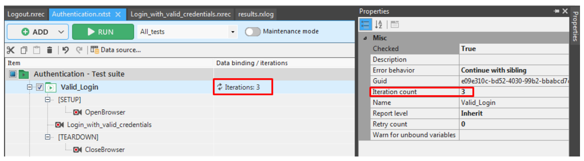
Please note

Due to the way Ranorex Studio reports these results on the JUnit XML report, different Test issues will be created for each run iteration or data row.

We should have this in mind as it hardens management of these Test issues (e.g. number of unrelated Tests, linkage to user stories).

Run iterations

On the properties of a test case, we can configure the number of run iterations (i.e. iteration count).



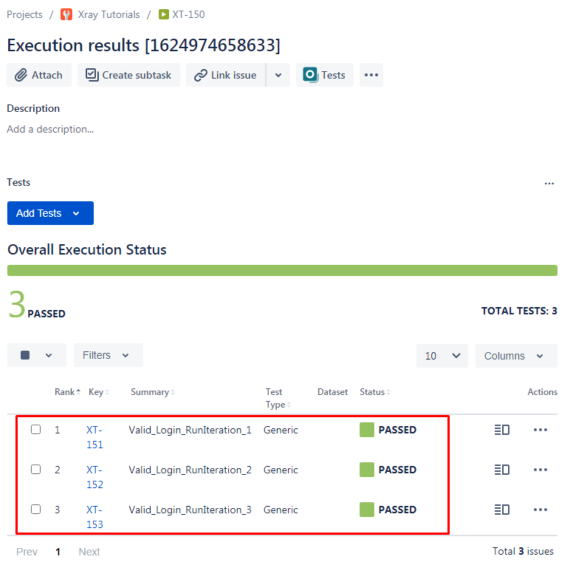
Item	Data binding / iterations
Authentication - Test suite	
Valid_Login	Iterations: 3
[SETUP]	
OpenBrowser	
Login_with_valid_credentials	
[TEARDOWN]	
CloseBrowser	

Properties

Misc

Checked	True
Description	
Error behavior	Continue with sibling
Guid	e09e310c-bd52-4030-99b2-bbabcd7
Iteration count	3
Name	Valid_Login
Report level	Inherit
Retry count	0
Warn for unbound variables	

After running the test case and importing the results to Xray using the JUnit XML report, a Test Execution with 3 Tests is created in Xray.



Projects / Xray Tutorials / XT-150

Execution results [1624974658633]

Description

Add a description...

Tests

Overall Execution Status

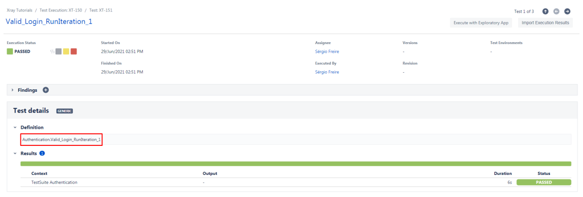
3 PASSED — TOTAL TESTS: 3

Rank	Key	Summary	Test Type	Dataset	Status
1	XT-151	Valid_Login_RunIteration_1	Generic		PASSED
2	XT-152	Valid_Login_RunIteration_2	Generic		PASSED
3	XT-153	Valid_Login_RunIteration_3	Generic		PASSED

Total 3 issues

Due to the way run iterations are reported in the JUnit XML report, each run iteration for our test case is abstracted as a different Test issue, with the run iteration being part of its definition.

In other words, we'll have as many Test issues as the iteration count configured for the test case.



Xray Tutorials / Test Execution: XT-150 / Test: XT-151

Valid_Login_RunIteration_1

Test 1 of 3

Execution Status	Started On	Assignee	Versions	Test Environments
PASSED	29/Jun/2021 02:51 PM	Sérgio Freire	-	-
	Finished On	**Executed By**	**Revision**	
	29/Jun/2021 02:51 PM	Sérgio Freire	-	

Test details GENERIC

Definition

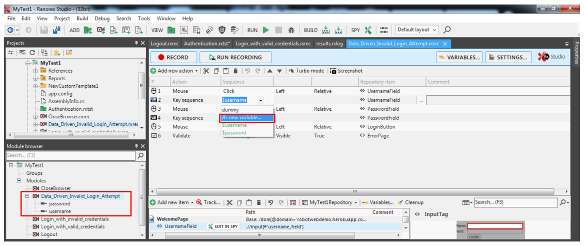
Authentication.Valid_Login_RunIteration_1

Results

Context	Output	Duration	Status
TestSuite Authentication	-	6s	PASSED

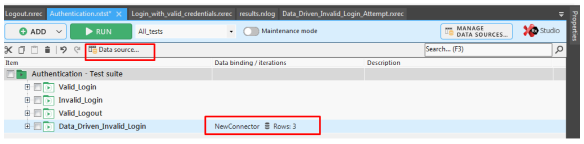
Data-driven tests

To make a data-driven test we need to use some variables in our actions, instead of using hardcoded values/strings.



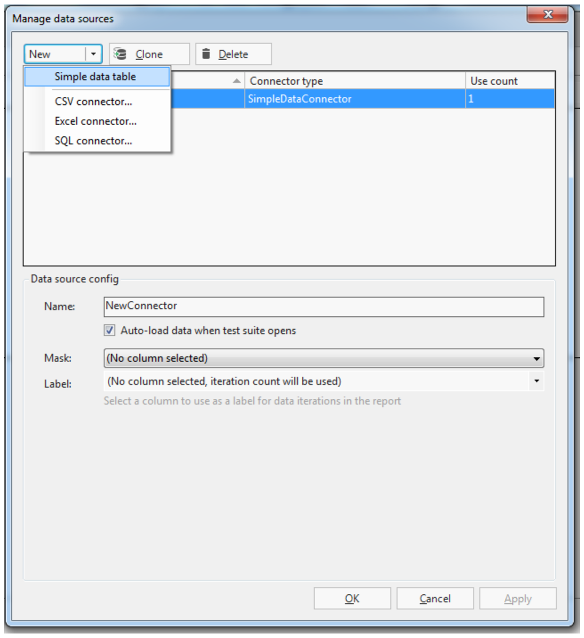
#	Action	Sequence			Repository item
1	Mouse	Click	Left	Relative	UsernameField
2	Key sequence	$username			UsernameField
3	Mouse		Left	Relative	PasswordField
4	Key sequence				PasswordField
5	Mouse		Left	Relative	LoginButton
6	Validate		Visible	True	ErrorPage

Then we need to configure a data-source for the selected test case.



Item	Data binding / iterations	Description
Authentication - Test suite		
Valid_Login		
Invalid_Login		
Valid_Logout		
Data_Driven_Invalid_Login	NewConnector Rows: 3	

There are different types of data-sources; a simple built-in data table can be used to specify several named columns and some rows of data for them.



	Connector type	Use count
	SimpleDataConnector	1

Data source config

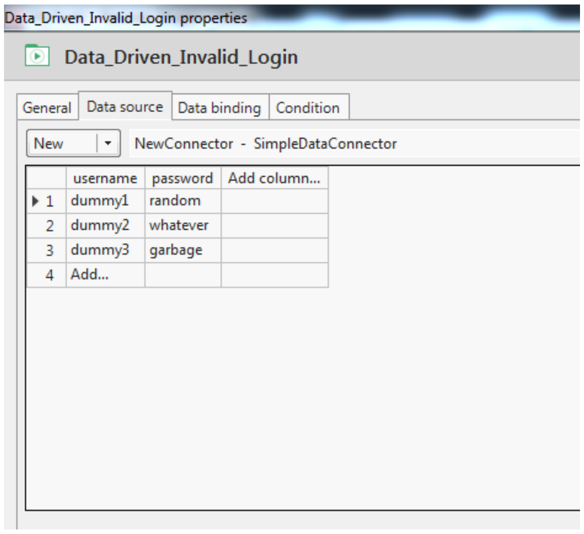
Name: NewConnector

☑ Auto-load data when test suite opens

Mask: (No column selected)

Label: (No column selected, iteration count will be used)

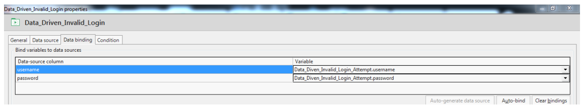
Select a column to use as a label for data iterations in the report


Data_Driven_Invalid_Login properties
Data_Driven_Invalid_Login
General
Data source
Data binding
Condition
New

NewConnector - SimpleDataConnector

	username	password	Add column...
1	dummy1	random	
2	dummy2	whatever	
3	dummy3	garbage	
4	Add...		

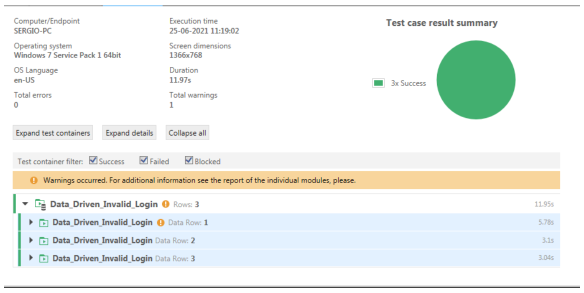
We then need to bind the columns of our data-source to the variables used in the previous module.


Data_Driven_Invalid_Login properties
Data_Driven_Invalid_Login
General
Data source
Data binding
Condition
Auto-generate data source
Auto-bind
Clear bindings

Bind variables to data sources

Data-source column	Variable
username	Data_Diven_Invalid_Login_Attempt.username
password	Data_Diven_Invalid_Login_Attempt.password

We can run the test case in Ranorex Studio and see the results for each data row.



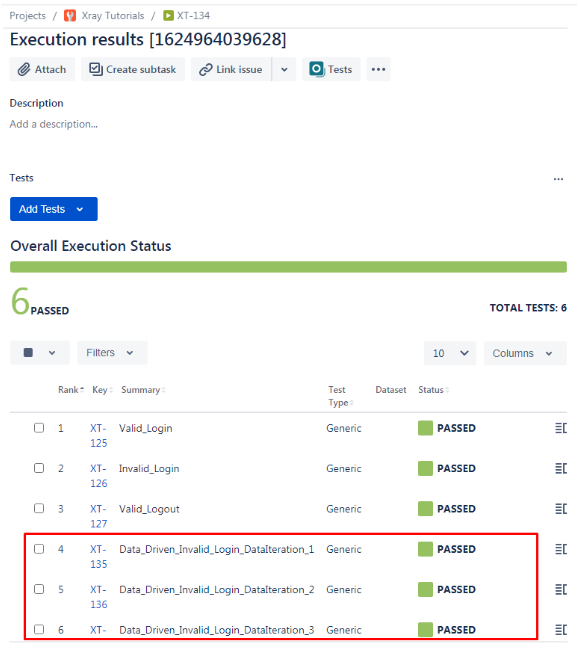
Computer/Endpoint SERGIO-PC	Execution time 25-06-2021 11:19:02
Operating system Windows 7 Service Pack 1 64bit	Screen dimensions 1366x768
OS Language en-US	Duration 11.97s
Total errors 0	Total warnings 1

Test container filter: ☑ Success ☑ Failed ☑ Blocked

Warnings occurred. For additional information see the report of the individual modules, please.

Data_Driven_Invalid_Login Rows: 3	11.95s
Data_Driven_Invalid_Login Data Row: 1	5.78s
Data_Driven_Invalid_Login Data Row: 2	3.1s
Data_Driven_Invalid_Login Data Row: 3	3.04s

After running the test case and importing the results to Xray using the JUnit XML report, a Test Execution having 3 Tests related to our data-source is created in Xray.



Projects / Xray Tutorials / XT-134

Execution results [1624964039628]

Description

Add a description...

Tests

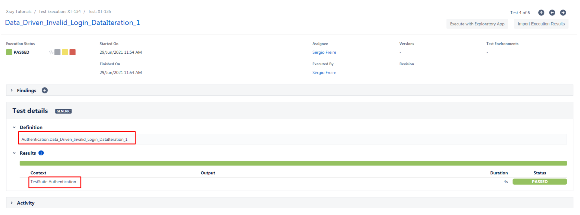
Overall Execution Status

6 PASSED — TOTAL TESTS: 6

Rank	Key	Summary	Test Type	Dataset	Status
1	XT-125	Valid_Login	Generic		PASSED
2	XT-126	Invalid_Login	Generic		PASSED
3	XT-127	Valid_Logout	Generic		PASSED
4	XT-135	Data_Driven_Invalid_Login_DataIteration_1	Generic		PASSED
5	XT-136	Data_Driven_Invalid_Login_DataIteration_2	Generic		PASSED
6	XT-	Data_Driven_Invalid_Login_DataIteration_3	Generic		PASSED

Due to the way run iterations are reported in the JUnit XML report, each run iteration for our test case is abstracted as a different Test issue, with the run iteration being part of its definition.

In other words, we'll have as many Test issues as the iteration count configured for the test case.



Xray Tutorials / Test Execution: XT-134 / Test: XT-135

Data_Driven_Invalid_Login_DataIteration_1

Test 4 of 6

Execution Status	Started On	Assignee	Versions	Test Environments
PASSED	29/Jun/2021 11:54 AM	Sérgio Freire	-	-
	Finished On	**Executed By**	**Revision**	
	29/Jun/2021 11:54 AM	Sérgio Freire	-	

Test details GENERIC

Definition

Authentication.Data_Driven_Invalid_Login_DataIteration_1

Results 1

Context	Output	Duration	Status
TestSuite Authentication	-	4s	PASSED

Ranorex's built-in integration with Jira

Ranorex Studio has a built-in integration with Jira. It can be used, for example, to open bugs, resolve, and reopen them depending on testing results.

The built-in Jira integration is totally independent from the Xray integration described in the current article and may be eventually complementarily.

References

- Ranorex web site
- Ranorex User Guide
- Ranorex vs Selenium WebDriver
- Integrating Ranorex with Jenkins
    - blog post
    - documentation