Squash TF Documentation

Henix

Jan 17, 2019

Contents

1 Sous section overview **3**
- 1.1 An example to begin with . . . 3
  - 1.1.1 What does that TA script? . . . 4
  - 1.1.2 Details . . . 5
- 1.2 An another example . . . 6
  - 1.2.1 What does that TA script? . . . 7
  - 1.2.2 Details . . . 8

2 The suite **11**
- 2.1 What does that TA script? . . . 12
- 2.2 Details . . . 13

3 Indices and tables **15**

A TA script is at the basis of an automation test. In this section, we're going to see the different elements of a TA Script and how it is build.

CHAPTER 1

Sous section overview

1.1 An example to begin with

Contents

- *What does that TA script?*
- *Details*

Every test in Squash Test Automation (Squash TA) is written as a script: the script will coordinate the manipulation of one or several technologies and/or resources in order to test a single functionality. A script is a plain text file (with the extension '.ta') containing lists of instructions. All you have to do is to drop it in the 'tests' directory (or anywhere in its sub hierarchy) of your project.

test - this text should be red

A script is merely an ordered list of instructions. For instance let us consider the script right here:

```
//database test 1

// -- 1 --
SETUP :
// -- 2 --
LOAD queries/sql/select.sql AS select.file
CONVERT select.file TO query.sql AS query
// -- 3 --
# LOAD dbunit/resultsets/mytable_simple_select.xml TO XML DATASET expected.dataset

// -- 4 --
TEST :
// -- 5 --
EXECUTE execute WITH query ON my_database AS raw_result
// -- 6 --
```

(continues on next page)

(continued from previous page)

```
CONVERT raw_result TO dataset.dbunit AS actual.dataset
// -- 7 --
ASSERT actual.dataset IS equal WITH expected.dataset
```

1.1.1 What does that TA script?

All that the script does is the execution of a SQL query and compare the result to the content of a DbUnit dataset. We notice there several elements: some comments (lines beginning with '//'), phases ('SETUP :' and 'TEST :'), instructions (multi-colored lines) and shortcuts (brown lines). Now we will break down and analyze that script, identify its components and see how they work together.

Each step in that example are annotated with a comment numbering them (lines beginning with a double slash '//'). Here are the captions for these corresponding landmarks :

```
// -- 1 --
SETUP :
```

– 1 'SETUP' phase declaration –

The 'SETUP' phase group instructions that will prepare the test. Note that the instructions in that phase could also have been set in the main phase ('TEST'). Differences lie in the handling of test failures: when an instruction fails during the 'SETUP' phase it means that the test itself is wrong, while a failure in the 'TEST' phase means that the System Under Test (SUT) is wrong. Long story short, the 'SETUP' phase sets the test prerequisites up.

```
// -- 2 --
LOAD queries/sql/select.sql AS select.file
CONVERT select.file TO query.sql AS query
```

– 2 Loading a SQL query –

The pair of instructions here will load a file then declare it contains a SQL query, in that order.

```
// -- 3 --
# LOAD dbunit/resultsets/mytable_simple_select.xml TO XML DATASET expected.dataset
```

– 3 Loading a DbUnit dataset –

The standard way to load a DbUnit dataset would require the same steps than above: load the file, then treat it to make it a DbUnit dataset. However, instead of explicitly lay the corresponding instructions, for that precise task Squash TA proposes a shortcut (a macro instruction) that achieves this goal. The line in the script was not syntax colored because it is not a standard instruction (although it looks like one).

```
// -- 4 --
TEST :
```

– 4 'TEST' phase declaration –
The 'TEST' phase groups the main test instructions. The following instructions will interact with the SUT and the resources created during the 'SETUP' phase are still available. If an instruction fails the test will end and status displayed in the test result will be set according to the nature of the error.

```
// -- 5 --
EXECUTE execute WITH query ON my_database AS raw_result
```

– 5 Execution of the query –
We reuse the query created during the setup and execute it against the database. The name of the database here is 'my_database'. The resulting data are gathered and stored in the context under the name supplied at the end of the instruction ('raw_result').

```
// -- 6 --
CONVERT raw_result TO dataset.dbunit AS actual.dataset
```

– 6 Data transformation –
Back to step 3 we readied the (expected) data, formatted as a dbunit dataset. If we are to compare the actual data (the result of the step just above) with the expected data we must first convert the actual data to a suitable type - in this case it must be formatted as a DbUnit dataset.

```
// -- 7 --
ASSERT actual.dataset IS equal WITH expected.dataset
```

– 7 Perform the comparison –
Now we're all set to proceed and test the data against each other. The status of the test will depend on the status of that comparison. If the comparison fails the test will be flagged as failed, while if the comparison is a success the script continues to the next instruction. Here there are no further instructions and the test will terminate with status success in that later case.

1.1.2 Details

So now you should have a glimpse on what a TA script is made of. Beyond the functional goal of that example script (merely check the content of a database) we noticed that a script contains instructions and shortcuts dispatched in

different phases (there is a 'teardown' phase too, optional just like 'setup' is, but it was not featured in that example). Every TA script just looks like that.

When looking closer at an instruction (e.g. step 5) several elements become salient: the tokens of the language (red words) and the name of variable elements (black, blue, pink and yellow words). The token of the language are the spine of a script and never change. The identifiers (name of variable elements) refer to elements sooner or later available in the script, including:

- A file or other physical resource (e.g. step 2 : queries/sql/select.sql)
- The category of an assertion, a command or a converter (e.g. step 5 : EXECUTE **execute** WITH query ON my_database AS raw_result)
- The category of a resource (e.g. step 6 : CONVERT raw_result TO **dataset.dbunit**)
- The name of a target, a repository, an already loaded resource or a resource to be created (e.g. step 5 : EXECUTE execute WITH **query** ON **my_database** AS **raw_result**)

The tokens are the essence of a script and define the semantic of a given instruction, its structure and its format. On the other hand the identifiers are variable and refer to various components of the test engine. Basically the language tokens tell the engine to execute or activate some components and make them interact with each others. The default settings of Squash TA includes the 'commons-components' plugin and some identifiers readily available from fresh start, and later on you will be able to add more components to the engine by adding more plugins.

Simply put, the phase and tokens define the grammar of a script while the identifers are rather listed in a dictionary (and each new plugin brings its dictionary).

1.2 An another example

Contents

- *What does that TA script?*
- *Details*

Add some text

Every test in Squash Test Automation (Squash TA) is written as a script: the script will coordinate the manipulation of one or several technologies and/or resources in order to test a single functionality. A script is a plain text file (with the extension '.ta') containing lists of instructions. All you have to do is to drop it in the 'tests' directory (or anywhere in its sub hierarchy) of your project.

A script is merely an ordered list of instructions. For instance let us consider the script right here:

```
//database test 1

// -- 1 --
SETUP :
// -- 2 --
LOAD queries/sql/select.sql AS select.file
CONVERT select.file TO query.sql AS query
// -- 3 --
# LOAD dbunit/resultsets/mytable_simple_select.xml TO XML DATASET expected.dataset

// -- 4 --
TEST :
```

(continues on next page)

(continued from previous page)

```
// -- 5 --
EXECUTE execute WITH query ON my_database AS raw_result
// -- 6 --
CONVERT raw_result TO dataset.dbunit AS actual.dataset
// -- 7 --
ASSERT actual.dataset IS equal WITH expected.dataset
```

1.2.1 What does that TA script?

All that the script does is the execution of a SQL query and compare the result to the content of a DbUnit dataset. We notice there several elements: some comments (lines beginning with '//'), phases ('SETUP :' and 'TEST :'), instructions (multi-colored lines) and shortcuts (brown lines). Now we will break down and analyze that script, identify its components and see how they work together.

Each step in that example are annotated with a comment numbering them (lines beginning with a double slash '//'). Here are the captions for these corresponding landmarks :

```
// -- 1 --
SETUP :
```

– 1 'SETUP' phase declaration –

The 'SETUP' phase group instructions that will prepare the test. Note that the instructions in that phase could also have been set in the main phase ('TEST'). Differences lie in the handling of test failures: when an instruction fails during the 'SETUP' phase it means that the test itself is wrong, while a failure in the 'TEST' phase means that the System Under Test (SUT) is wrong. Long story short, the 'SETUP' phase sets the test prerequisites up.

```
// -- 2 --
LOAD queries/sql/select.sql AS select.file
CONVERT select.file TO query.sql AS query
```

– 2 Loading a SQL query –

The pair of instructions here will load a file then declare it contains a SQL query, in that order.

```
// -- 3 --
# LOAD dbunit/resultsets/mytable_simple_select.xml TO XML DATASET expected.dataset
```

– 3 Loading a DbUnit dataset –

The standard way to load a DbUnit dataset would require the same steps than above: load the file, then treat it to make it a DbUnit dataset. However, instead of explicitly lay the corresponding instructions, for that precise task Squash TA proposes a shortcut (a macro instruction) that achieves this goal. The line in the script was not syntax colored because it is not a standard instruction (although it looks like one).

```
// -- 4 --
TEST :
```

– 4 'TEST' phase declaration –
The 'TEST' phase groups the main test instructions. The following instructions will interact with the SUT and the resources created during the 'SETUP' phase are still available. If an instruction fails the test will end and status displayed in the test result will be set according to the nature of the error.

```
// -- 5 --
EXECUTE execute WITH query ON my_database AS raw_result
```

– 5 Execution of the query –
We reuse the query created during the setup and execute it against the database. The name of the database here is 'my_database'. The resulting data are gathered and stored in the context under the name supplied at the end of the instruction ('raw_result').

```
// -- 6 --
CONVERT raw_result TO dataset.dbunit AS actual.dataset
```

– 6 Data transformation –
Back to step 3 we readied the (expected) data, formatted as a dbunit dataset. If we are to compare the actual data (the result of the step just above) with the expected data we must first convert the actual data to a suitable type - in this case it must be formatted as a DbUnit dataset.

```
// -- 7 --
ASSERT actual.dataset IS equal WITH expected.dataset
```

– 7 Perform the comparison –
Now we're all set to proceed and test the data against each other. The status of the test will depend on the status of that comparison. If the comparison fails the test will be flagged as failed, while if the comparison is a success the script continues to the next instruction. Here there are no further instructions and the test will terminate with status success in that later case.

1.2.2 Details

So now you should have a glimpse on what a TA script is made of. Beyond the functional goal of that example script (merely check the content of a database) we noticed that a script contains instructions and shortcuts dispatched in

different phases (there is a 'teardown' phase too, optional just like 'setup' is, but it was not featured in that example). Every TA script just looks like that.

When looking closer at an instruction (e.g. step 5) several elements become salient: the tokens of the language (red words) and the name of variable elements (black, blue, pink and yellow words). The token of the language are the spine of a script and never change. The identifiers (name of variable elements) refer to elements sooner or later available in the script, including:

- A file or other physical resource (e.g. step 2 : queries/sql/select.sql)
- The category of an assertion, a command or a converter (e.g. step 5 : EXECUTE **execute** WITH query ON my_database AS raw_result)
- The category of a resource (e.g. step 6 : CONVERT raw_result TO **dataset.dbunit**)
- The name of a target, a repository, an already loaded resource or a resource to be created (e.g. step 5 : EXECUTE execute WITH **query** ON **my_database** AS **raw_result**)

The tokens are the essence of a script and define the semantic of a given instruction, its structure and its format. On the other hand the identifiers are variable and refer to various components of the test engine. Basically the language tokens tell the engine to execute or activate some components and make them interact with each others. The default settings of Squash TA includes the 'commons-components' plugin and some identifiers readily available from fresh start, and later on you will be able to add more components to the engine by adding more plugins.

Simply put, the phase and tokens define the grammar of a script while the identifers are rather listed in a dictionary (and each new plugin brings its dictionary).

Home page _ta_scripting_home_page

Link===== _ta_scripting_home_page

CHAPTER 2

The suite

Contents

- *What does that TA script?*
- *Details*

Every test in Squash Test Automation (Squash TA) is written as a script: the script will coordinate the manipulation of one or several technologies and/or resources in order to test a single functionality. A script is a plain text file (with the extension '.ta') containing lists of instructions. All you have to do is to drop it in the 'tests' directory (or anywhere in its sub hierarchy) of your project.

A script is merely an ordered list of instructions. For instance let us consider the script right here:

```
//database test 1

// -- 1 --
SETUP :
// -- 2 --
LOAD queries/sql/select.sql AS select.file
CONVERT select.file TO query.sql AS query
// -- 3 --
# LOAD dbunit/resultsets/mytable_simple_select.xml TO XML DATASET expected.dataset

// -- 4 --
TEST :
// -- 5 --
EXECUTE execute WITH query ON my_database AS raw_result
// -- 6 --
CONVERT raw_result TO dataset.dbunit AS actual.dataset
// -- 7 --
ASSERT actual.dataset IS equal WITH expected.dataset
```

2.1 What does that TA script?

All that the script does is the execution of a SQL query and compare the result to the content of a DbUnit dataset. We notice there several elements: some comments (lines beginning with '//'), phases ('SETUP :' and 'TEST :'), instructions (multi-colored lines) and shortcuts (brown lines). Now we will break down and analyze that script, identify its components and see how they work together.

Each step in that example are annotated with a comment numbering them (lines beginning with a double slash '//'). Here are the captions for these corresponding landmarks :

```
// -- 1 --
SETUP :
```

– 1 'SETUP' phase declaration –

The 'SETUP' phase group instructions that will prepare the test. Note that the instructions in that phase could also have been set in the main phase ('TEST'). Differences lie in the handling of test failures: when an instruction fails during the 'SETUP' phase it means that the test itself is wrong, while a failure in the 'TEST' phase means that the System Under Test (SUT) is wrong. Long story short, the 'SETUP' phase sets the test prerequisites up.

```
// -- 2 --
LOAD queries/sql/select.sql AS select.file
CONVERT select.file TO query.sql AS query
```

– 2 Loading a SQL query –

The pair of instructions here will load a file then declare it contains a SQL query, in that order.

```
// -- 3 --
# LOAD dbunit/resultsets/mytable_simple_select.xml TO XML DATASET expected.dataset
```

– 3 Loading a DbUnit dataset –

The standard way to load a DbUnit dataset would require the same steps than above: load the file, then treat it to make it a DbUnit dataset. However, instead of explicitly lay the corresponding instructions, for that precise task Squash TA proposes a shortcut (a macro instruction) that achieves this goal. The line in the script was not syntax colored because it is not a standard instruction (although it looks like one).

```
// -- 4 --
TEST :
```

– 4 'TEST' phase declaration –

The 'TEST' phase groups the main test instructions. The following instructions will interact with the SUT and the resources created during the 'SETUP' phase are still available. If an instruction fails the test will end and status displayed in the test result will be set according to the nature of the error.

```
// -- 5 --
EXECUTE execute WITH query ON my_database AS raw_result
```

– 5 Execution of the query –
We reuse the query created during the setup and execute it against the database. The name of the database here is 'my_database'. The resulting data are gathered and stored in the context under the name supplied at the end of the instruction ('raw_result').

```
// -- 6 --
CONVERT raw_result TO dataset.dbunit AS actual.dataset
```

– 6 Data transformation –
Back to step 3 we readied the (expected) data, formatted as a dbunit dataset. If we are to compare the actual data (the result of the step just above) with the expected data we must first convert the actual data to a suitable type - in this case it must be formatted as a DbUnit dataset.

```
// -- 7 --
ASSERT actual.dataset IS equal WITH expected.dataset
```

– 7 Perform the comparison –
Now we're all set to proceed and test the data against each other. The status of the test will depend on the status of that comparison. If the comparison fails the test will be flagged as failed, while if the comparison is a success the script continues to the next instruction. Here there are no further instructions and the test will terminate with status success in that later case.

2.2 Details

So now you should have a glimpse on what a TA script is made of. Beyond the functional goal of that example script (merely check the content of a database) we noticed that a script contains instructions and shortcuts dispatched in different phases (there is a 'teardown' phase too, optional just like 'setup' is, but it was not featured in that example). Every TA script just looks like that.

When looking closer at an instruction (e.g. step 5) several elements become salient: the tokens of the language (red words) and the name of variable elements (black, blue, pink and yellow words). The token of the language are the spine of a script and never change. The identifiers (name of variable elements) refer to elements sooner or later available in the script, including:

- A file or other physical resource (e.g. step 2 : queries/sql/select.sql)
- The category of an assertion, a command or a converter (e.g. step 5 : EXECUTE **execute** WITH query ON my_database AS raw_result)
- The category of a resource (e.g. step 6 : CONVERT raw_result TO **dataset.dbunit**)
- The name of a target, a repository, an already loaded resource or a resource to be created (e.g. step 5 : EXECUTE execute WITH **query** ON **my_database** AS **raw_result**)

The tokens are the essence of a script and define the semantic of a given instruction, its structure and its format. On the other hand the identifiers are variable and refer to various components of the test engine. Basically the language tokens tell the engine to execute or activate some components and make them interact with each others. The default settings of Squash TA includes the 'commons-components' plugin and some identifiers readily available from fresh start, and later on you will be able to add more components to the engine by adding more plugins.

Simply put, the phase and tokens define the grammar of a script while the identifers are rather listed in a dictionary (and each new plugin brings its dictionary).

A TA script is a file containing an amount of instructions and shortcuts that will be interpreted by the engine of Squash-TA to ex
The amount of instructions, shortcuts (macros), and marker phases form the specific language of Squash-TA (DSL) to describe automation tests. It permits to address a heterogeneous panel of tests with a common formalism.

TA scripts use resource components. To handle these resource components and realize tests, instructions need Engine components.

To understand all this concepts, we're going to begin with an example of a TA-script.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search