# SALOMÉ **T**est **F**ramework **M**anagement

Mikaël Marche

April 26, 2006

# Contents

# Chapter 1

# Introduction

Salomé-TMF is a test management framework. Salomé-TMF offers features for creating and executing tests. We use the concept of tests defined in the norm ISO9646. Tests can be manual or automatic, tests are organized in campaigns and are executed with different datasets in different environments. For making test execution fully automatic, we integrate a script language based on Java in our tool, as one of several plugins which extend Salomé-TMF functionalities.

## 1.1 Basic concept definitions

### 1.1.1 Test

A test is the execution of a program or of a sequence of actions, in a given environment. The goal is to check that the tested software fulfills its specifications, by isolating discrepancies between the results obtained and those waited for.

### 1.1.2 Test suite

A test suite is a logical set of tests.

### 1.1.3 Test family

A test family is a set of test suites.

### 1.1.4 Dataset

A dataset is a set of parameters with their values.

### 1.1.5 Test campaign

A test campaign is a set of tests which will be executed with different datasets in different environments.

### 1.1.6 Environment

An environment is a set of elements describing the environment in which tests will be executed:

- Initialization script: a script executed before the beginning of tests, at the launch of a test campaign.
- Restitution script: a script executed at the end of the test campaign.
- Dataset of valued parameters: those parameters and their values can be used by scripts or tests themselves.

### 1.1.7 Execution

A campaign execution links three elements: a test campaign, a dataset, and an environment. A campaign execution is launched and its results are kept as an archive and can be consulted.

## 1.2 Organisation and test description

The way to describe tests is willfully very directed in the Salomé-TMF test management framework, following the concepts of the norm ISO 9646. Test are organized first by family, then by suite, a suite comprising tests as "atomic" elements (Figure 1.1).

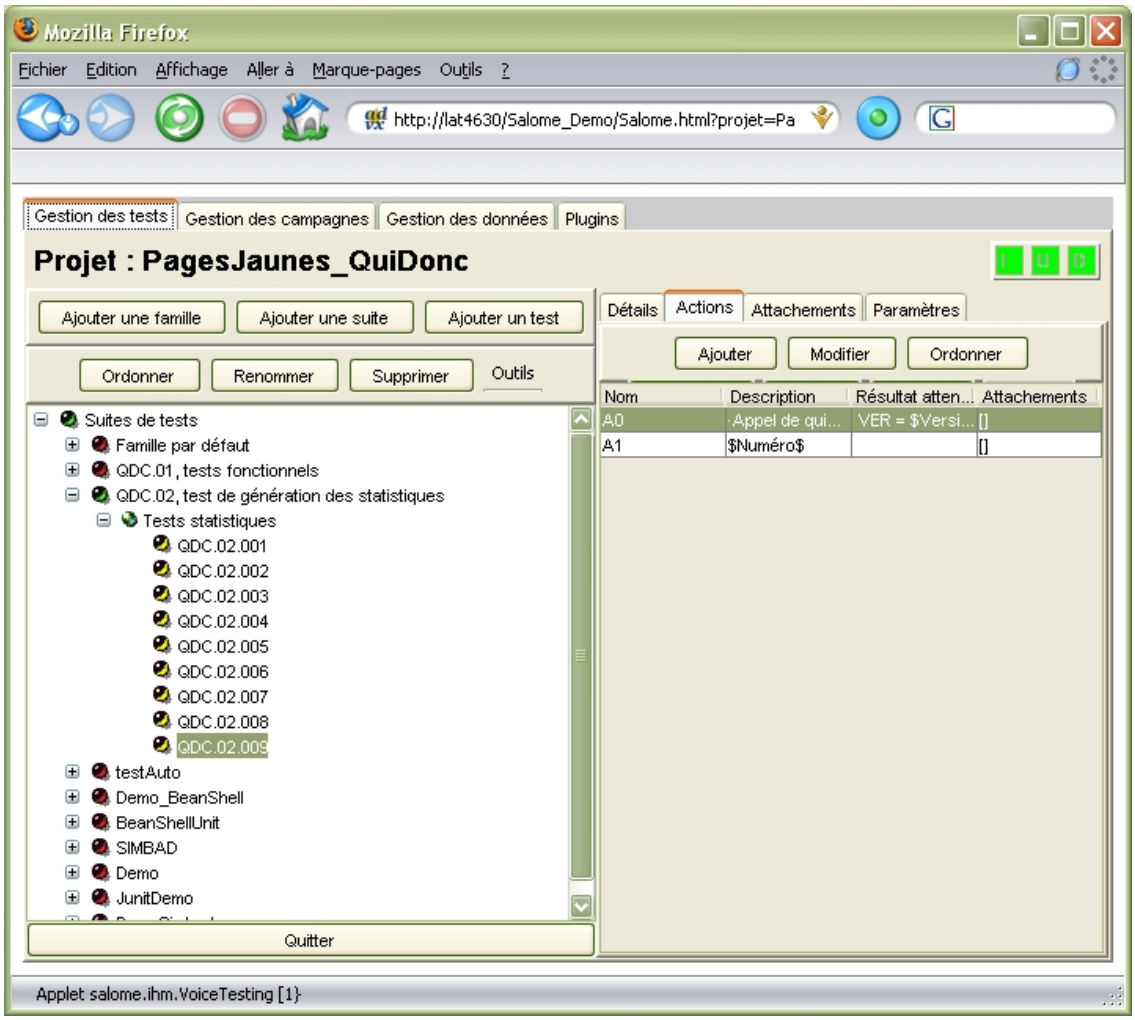


Figure 1.1: Test plan

Tests can be of one of two categories: either automatic or manual. A manual test is made of the description of several steps which need to be executed. For each step, a verification to be made by the tester is included.

An automatic test comprises a script or a test program which will be executed by the Salomé-TMF tool, which will automatically kept the result in the corresponding test campaign execution.

## 1.3 Creating test campaigns

When the set of tests is described in the tool, the user can define test campaigns (Figure 1.2). A test campaign is a set of tests, defined indepently of the test suites and families. A test campaign will be executed in a given test environment, using given datasets.

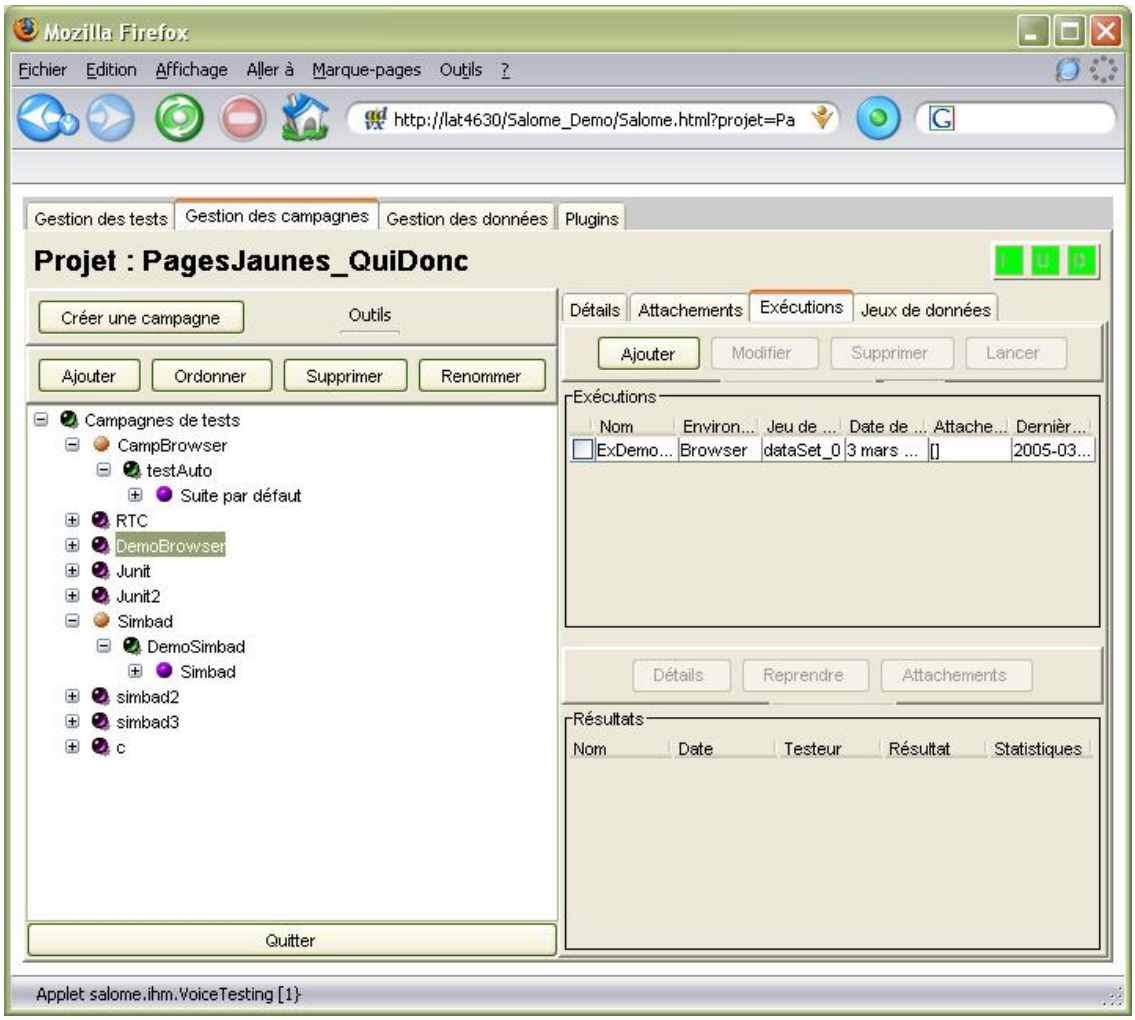


Figure 1.2: Test campaign

## 1.4 Test execution

Test campaigns are defined independently of the datasets and the execution environments. For launching the execution of a test campaign, you need to link a campaign with an execution environment and a dataset.

Those notions of environment and datasets enable to launch the tests for a given campaign on several versions of test environments and datasets in a simple way.

An environment is the target for a test campaign execution. It is made of a description, an initialization script, and a set of valued parameters which can be used in the tests. This dataset must give a value to all the parameters used in at least one test of the campaign. A test campaign execution, or shortly an execution (Figure 1.3) is defined as the linking between those three elements: a campaign, an environment and a dataset. When defined, an execution can be launched one or several times and the test results or verdicts will be attached to the execution for every launch.

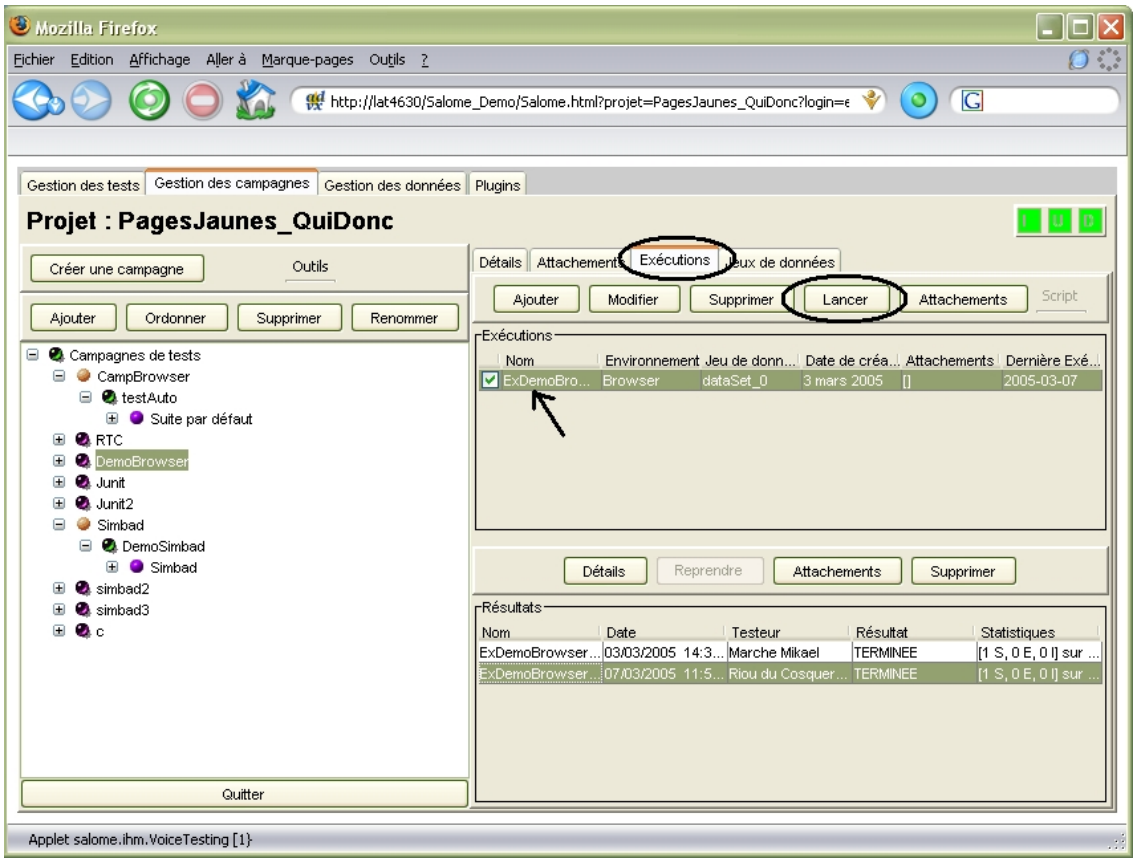


Figure 1.3: test campaign execution

# Chapter 2

# Installation

Before installing the test management framework *Salomé-TMF*, you need to have access to a Java install *JDK 1.4* or more, and to a database server *MySQL*.

## 2.1 Installing the tool Salomé-TMF

Installing Salomé-TMF can be done in two ways:

- An automatic installation using the install program **salome_tmf_install.jar**;
- A manual installation starting from the compressed file **salome_tmf.zip** that contains the tool resources.

We will present in the two following sections an how-to for those two installation methods.

### 2.1.1 Automatic installation

Start the executable file **salome_tmf_install.jar** (enter the command *java -jar salome_tmf_install.jar* in a Linux environment). A first window pops up asking to choose your language for the installation parameters.


Figure 2.1: Language selection

Choose your language (it will be the language kept for the remaining of the installation procedure and the language by default for Salomé-TMF), then press the "Ok" button. The following window welcomes you to the installation.

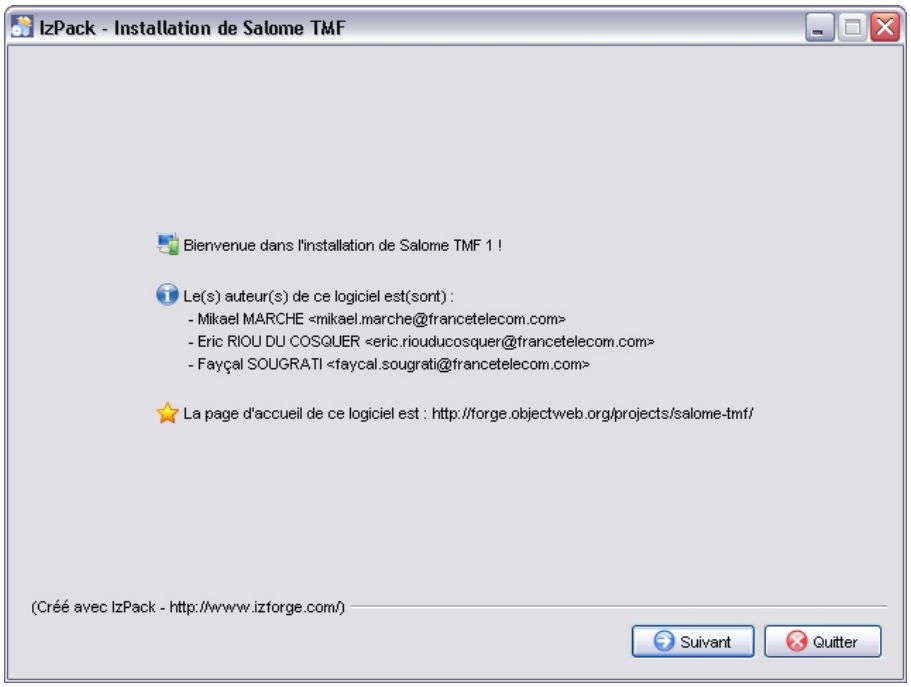


Figure 2.2:

Click the "Next" button for going to the next step. The new window asks your agreement for the GPL license (*GNU General Public License*) under which Salomé-TMF is distributed. If you wish to make a commercial distribution of Salomé-TMF, we will enter under a dual licensing scheme in the same way than the MySQL scheme.

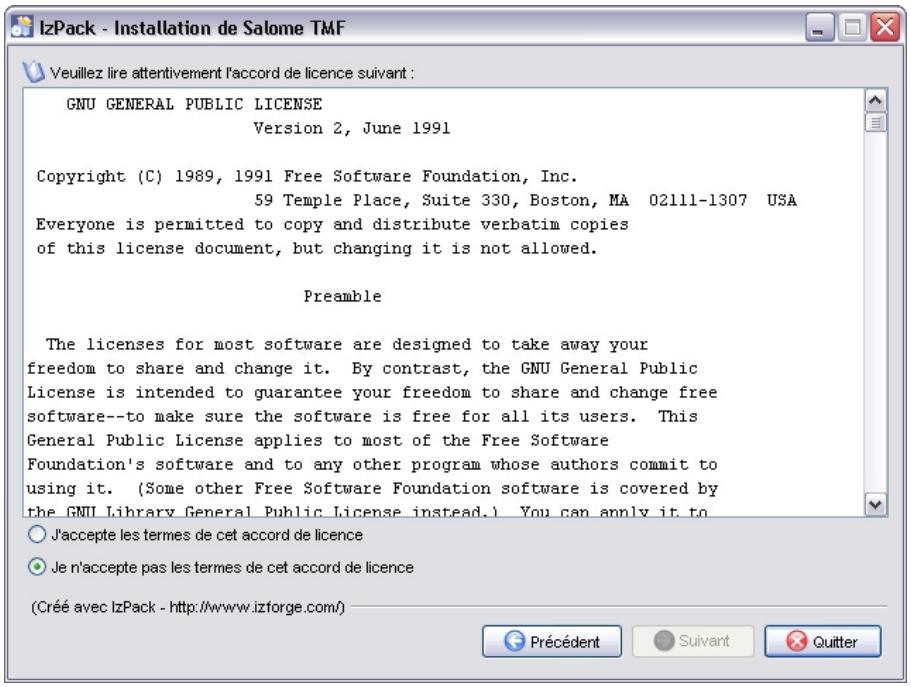


Figure 2.3: GPL license

Before proceeding with the installation procedure, you have to check the option "I agree with the termes of this license agreement", and to click the "Next" button. The next step deals with the installation directory for Salomé-TMF.


Figure 2.4: Choosing the installation directory

Choose the installation directory, then click "Next". A new window proposes optional components for the tool.

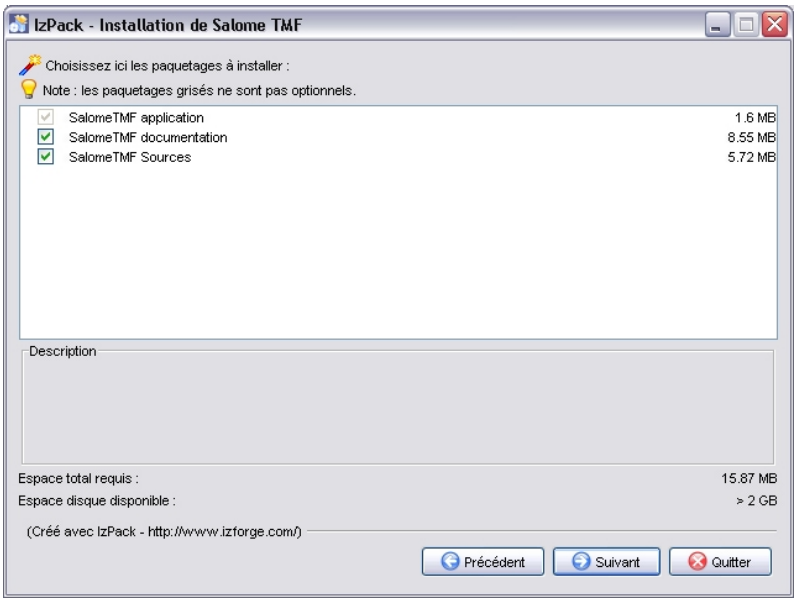


Figure 2.5: Choosing components

After checking the optional components you wish, click "Next".

A window gives then the progress status for the installation procedure, including for the separate components.

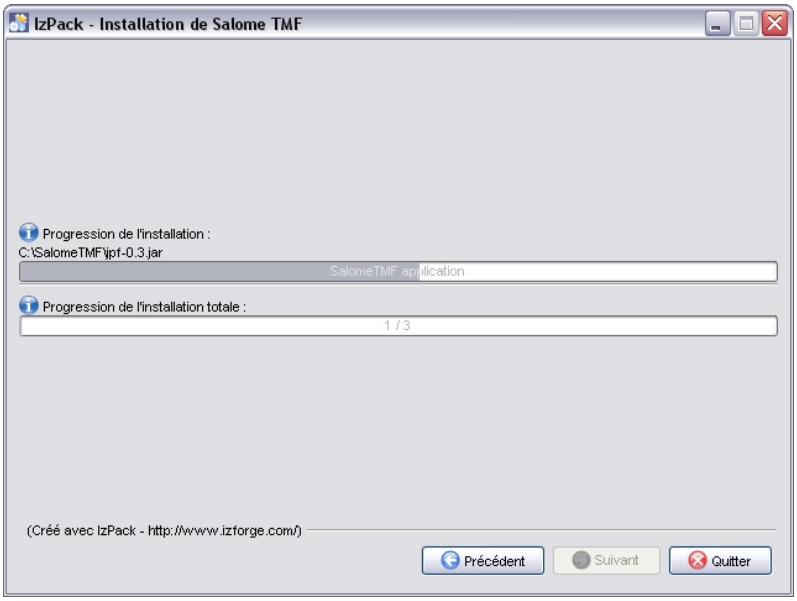


Figure 2.6: Installation progress

When components are installed, click "Next". The following step will install the database of *Salomé TMF*, and will create the configuration file **DB_Connexion.properties** in the directory **cfg**/ put in the installation directory. This file enables to configure the linking of *Salomé TMF* to its *MySQL* database, and to tune some parameters for some features. This configuration file is

created by filling the following form.

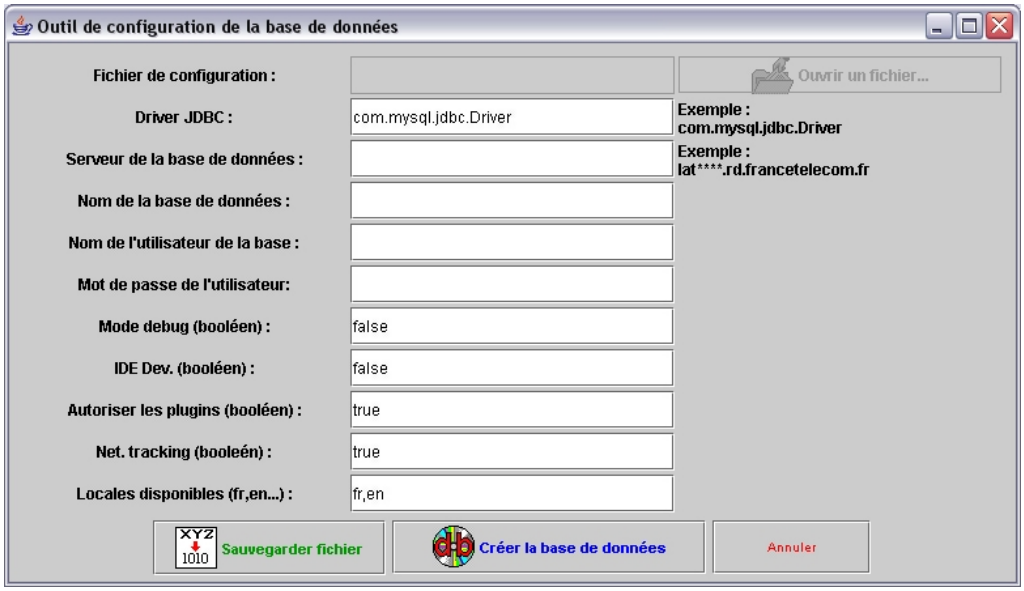


Figure 2.7: Configuration parameters for Salomé-TMF

For configuring Salomé-TMF, you need to fill the following fields:

- **Driver JDBC**: name of the *JDBC* driver for the linking between the Java API and the *MySQL* database. By default, this is *com.mysql.jdbc.Driver*.
- **Data server**: name of the machine (or its IP adress) in which the *MySQL* database will be installed.
- **Database name**.
- **User name**: this is the *MySQL* user which will be created and which will be used by the tool Salomé-TMF for connecting to the database.
- **User password**: this password will be encrypted and saved in the configuration file for connecting to the database.
- **Debug mode**: put this field at *"true"* if you want to have a trace of the application execution in the browser console. By default, the value is *"false"*.
- **IDE Dev**: when you put this field at *"true"*, it implies that Salomé-TMF executes itself over an IDE (Eclipse). If the field is *"false"*, Salomé-TMF will execute inside the browser.
- **Authorizing plugins**: if the field is put to *"false"*, plugin management is not allowed in Salomé-TMF and no plugin will be uploaded.
- **Net Tracking**: the value *"true"* enables to activate the followup of other users' modifications through the three colored buttons in Salomé-TMF interface. If put to *"false"*, this feature is desactivated.
- **Localizations**: it is the list of localizations present in Salomé-TMF (by default, french and english).

When all those fields are defined, click on "Save file". A message pops up for confirming the success of the writing of the configuration file.

Next, click on "Create the database", a new window pops up.

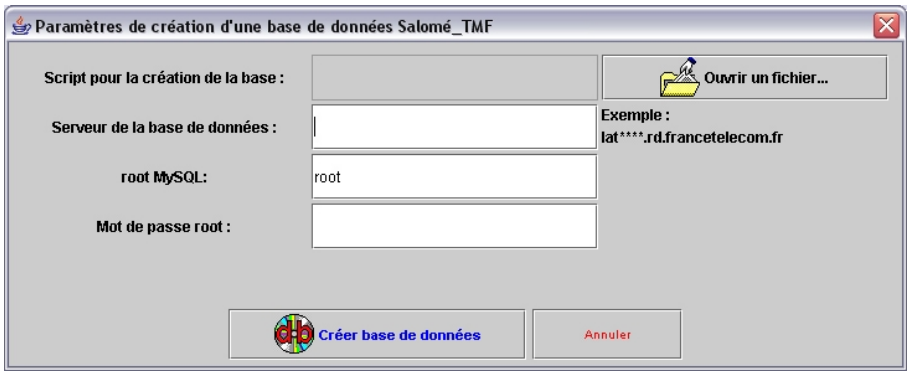


Figure 2.8: Creating Salomé-TMF database

In this window, the user has got to fill the following fields:

- **Script for the database creation:** It is the file containing the creation script for Salomé-TMF database. By default, the proposed script file is: **create_salome_bdd.sql** created in the directory **bdd_model/** of the installation directory, but you can change it here.
- **Database server:** It is the name of the machine where the database is situated. By default, the name entered in the previous window is kept, but this choice is duplicated because in some cases, it is necessary to put here the value *"localhost"* for the database to be created. It is also possible that the database machine will be different than the machine that will hosts the application (local machine or apache server).
- **Root MySQL:** This is the login of the *MySQL* server administrator (by default *"root"*).
- **Password of the *MySQL* administrator.**

When all those fields are filled, you click next on *"Create the database"*. A message confirms then the success of the database creation.

After the database creation, click on "Next" in the installer program. A message says then than the installation procedure has proceeded correctly, you can click "Next" and "Finish".

If there is an error during the database installation, you should refer to the manual installation procedure.

### 2.1.2 Manual installation

After uncompressing the file **salome_tmf-1.zip**, you need to create the database and the *MySQL* user for *Salomé TMF* in the following steps.

**Connection to the MySQL server**

```
mysql -u root -p
```

After entering this command, you need to enter the MySQL administrator (root) password before connecting to the server.

**When connected to the MySQL server, create Salomé-TMF database**

```
mysql> CREATE DATABASE <salomeTMF_bdd>;
```

Where <*salomeTMF_bdd*> is the name of the database.

**Creating the Salomé-TMF user**

```
mysql> GRANT SELECT, INSERT, UPDATE, DELETE, INDEX, ALTER, CREATE, DROP, LOCK
TABLES, CREATE TEMPORARY TABLES, REFERENCES ON <salomeTMF_bdd>.* TO
<salomeTMF_user>@'%' IDENTIFIED BY '<salomeTMF_password>';
```

Where:

- <**salomeTMF_user**> is the user name;
- <**salomeTMF_password**> is the user password.

**Refresh the users' privileges for making sure they are taken into account**

```
mysql> FLUSH PRIVILEGES;
```

**Connecting to the database**

Connect to the database and execute the script for creating the database tables **create_salome_bdd.sql** which is put in the directory **bdd_model/** of the installation directory:

```
mysql> connect <salomeTMF_bdd>; mysql> source
<install_dir>\bdd_model\create_salome_bdd.sql;
```

Where <**install_dir**> is the installation directory for Salomé-TMF. **If you are working under Linux, use "/" as separators**.

When those steps are done, you need to configure the file for connecting to the database, and the password encryption key. For this, execute the JAR file **salome_tmf_tools.jar** which is at the root of the installation directory. The window described in Figure 2.7 pops up. This is the same fields to fill than when configuring the database during the automatic installation (cf. section 2.1.1). The name of the database, the user login and her password must be the same than those entered in the previous *MySQL* fields.

Fill those fields, then click on *"Save file"*. A window asks to choose the directory in which the configuration file will be created. Choose the directory **cfg/** of the installation directory of *Salomé TMF*, then click on *"Validate"*. Installation is finished.

## 2.2 Installing a Salomé-TMF plug-in

Any Salomé-TMF plugin is given as a compressed file **nom_du_plugin.zip**. A utility given as a JAR file (**pluginInstall.jar** put in the plugins directory) enables to install the plugins.

For installing a plug-in, follow those steps:

1. Uncompress the ZIP file in the directory **plugins**/ of the installation directory of *Salomé TMF*.
2. Go to this directory and execute the command

```
 java -jar pluginInstall.jar
nom_du_plugin.zip
```

For some plugins (for example the **Bugzilla** one), other configurations are necessary (see the plugin documentation).

# Chapter 3

# Administration

## 3.1 Salomé-TMF administration

For getting access to Salomé-TMF administration features (Figure 3.1), you need to:

- Go to the Salomé-TMF home page in your browser;
- Select the tab "Admin Salomé_TMF" ;
- Select the admin login and password (by default, the password is "admin");
- Click the button "Admin Salomé_TMF".


Figure 3.1: Connexion to Salomé-TMF administration features

### 3.1.1 Changing the Salomé-TMF admin password

For changing the admin password, select the button "Change password". You will then have to:

- Enter the old password;
- Enter the new password;
- Confirm the new password;
- Validate.

### 3.1.2 Managing projects

For getting access to the projects' management features, select the button "Managing projects".

#### Creating projects

For creating a new project, you need to:

- Click on the "Create" button;
- Select the project admin in the proposed list;
- Define the name and the project description;
- Validate.

### Creating a project from an existing project

You can also create a project by uploading data from an existing project. The process is almost the same than when creating a project from scratch.

- Click on the "Create" button;
- Select the project admin in the proposed list;
- Define the name and the project description;
- Check the box for the option "Copy from an existing project";
- Select the imported project from the proposed list "From the project";
- Select the data that will be imported between:
  - Test suites;
  - Test campaigns;
  - Users;
  - groups.
- Validate.

### Modifying a project

You can modify a project description and/or name:

- Click the "Modify" button;
- Enter the new name and/or description of the project;
- Validate.

### Freezing a project

This feature enables to keep a project in the base and makes it impossible to use it anymore.

- Select a project in the list;
- Click the "Freeze" button;
- Validate.

### Deleting a project

For deleting a project, you need to:

- Select the project to delete in the list;
- Click on the "Delete" button;
- Confirm the deletion.

### 3.1.3 Managing users

For getting access to the users' management, you need to select the "Manage users" button (Figure 3.2), starting from the admin window for Salomé-TMF.

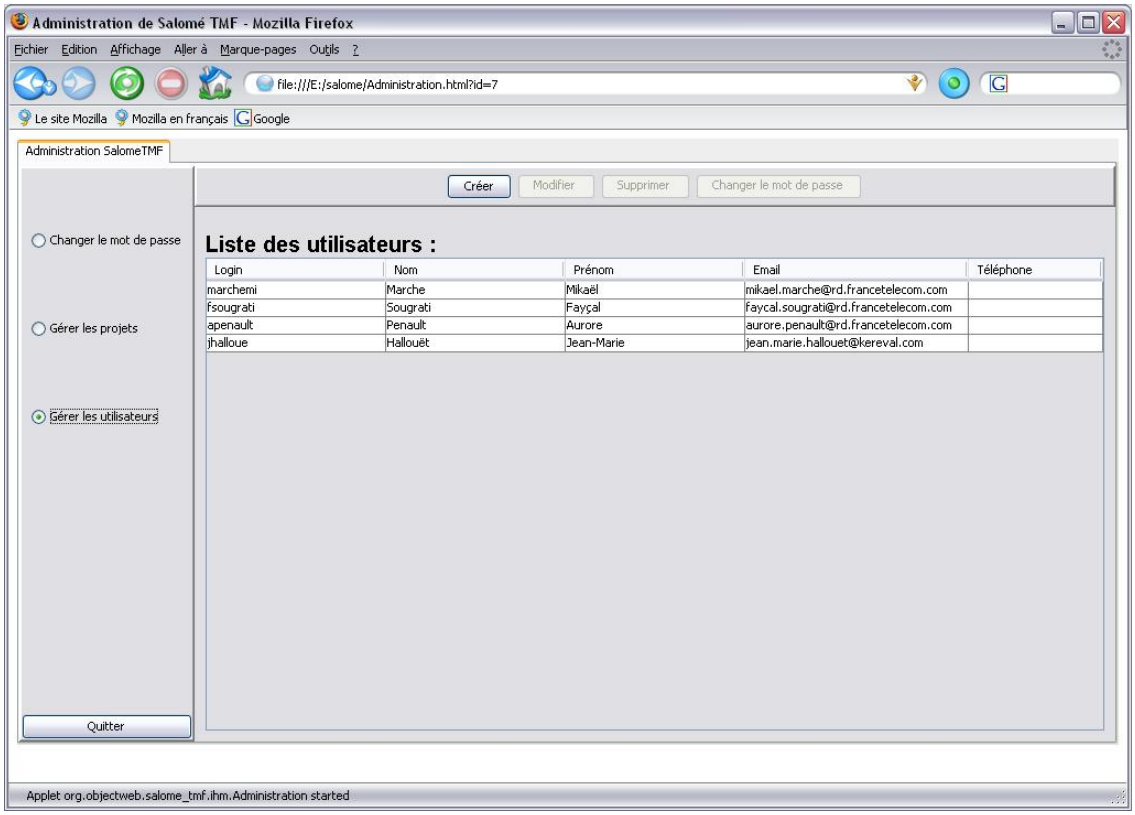


Figure 3.2: Users' management view

#### Creating a user

For creating a user (Figure 3.3), you need to:

- Click on the "Create" button;
- Enter the fields Login, Name, First Name, Email and Password (the Telephone field is optional);
- Validate.


Figure 3.3: Creating a user

### Modifying a user's informations

For modifying the informations for a given user, you need to:

- Select a user in the list;
- Click on "Modify";
- Validate.

### Deleting a user

For deleting a user, you need to:

- Select a user in the list;
- Click on "Delete";
- Confirm the deletion.

### Changing a user's password

For changing a user's password, you need to:

- Select a user in the list;
- Click on "Change password";
- Validate.

## 3.2 Administrating a project

For getting access to the administration features for an existing project (Figure 3.4), you need to:

- Go to the Salomé-TMF home page in your browser;
- Select the tab "Admin a project";
- Select one of the existing projects;
- Select the login and password of the project administrator;
- Click on the button "Admin Project".


Figure 3.4: Administrating a project

### 3.2.1 Creating the users for a project

For managing users for a given project, you need to click the button "Managing users" (Figure 3.5), starting from the project administration window.

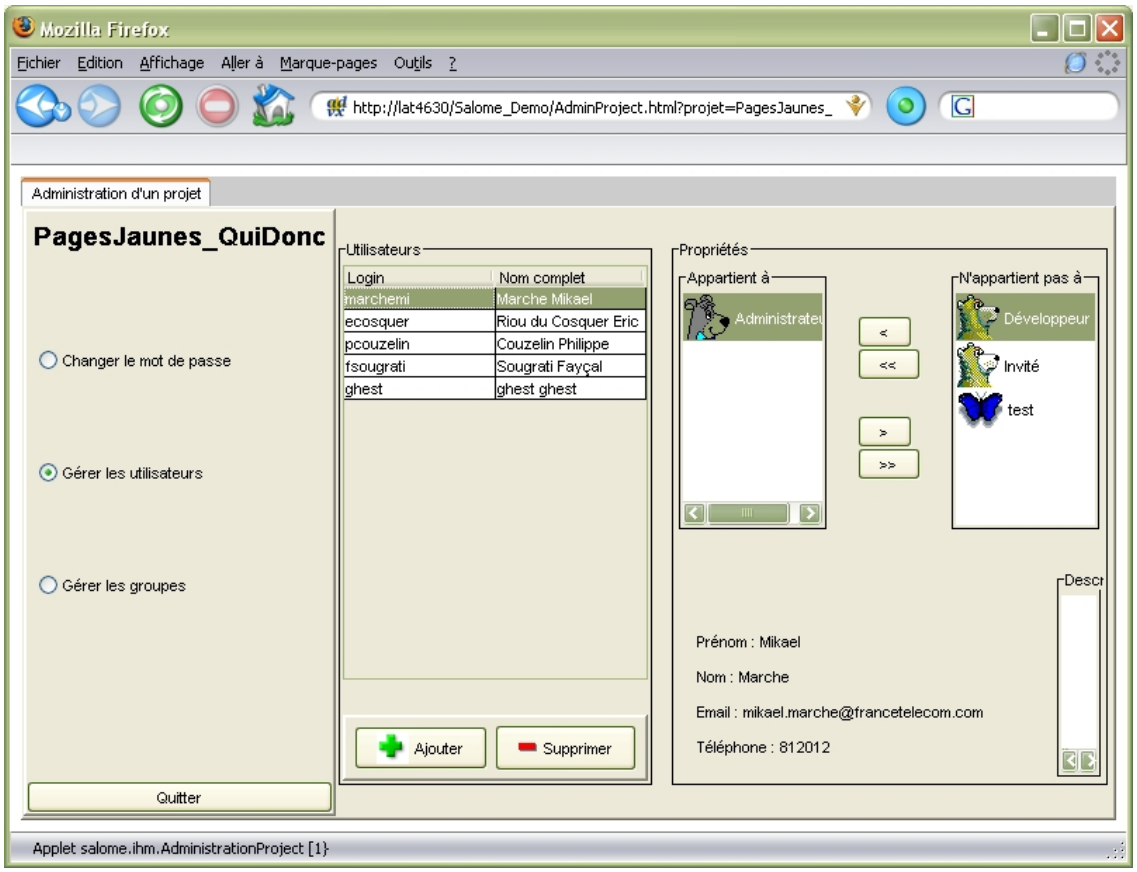


Figure 3.5: Managing a project's users

This window shows the project's users' list. When selecting a user in the list, you get access to her properties. Her contact info and her groups will be published.

### Specifying groups for a user

For changing the groups to which a user belongs, you need to use the buttons for adding and suppressing groups that you can find in the "Properties" part of the window.

### Adding or deleting a user

For adding a user to the project, you need to (Figure 3.6):

- Click on the "Add" button;
- Select the desired user;
- Validate.


Figure 3.6: Adding users in a project

Only the users that have laready been created from the Salomé-TFM administration field can be accessed.

For deleting a user, you need to (Figure 3.5) :

- Select a user in the list;
- Click on the "Delete" button;
- Confirm the deletion.

### 3.2.2 Creating groups

For getting access to the groups' management, you need to select the "Managing groups" button (Figure 3.7), starting from the project administration window.

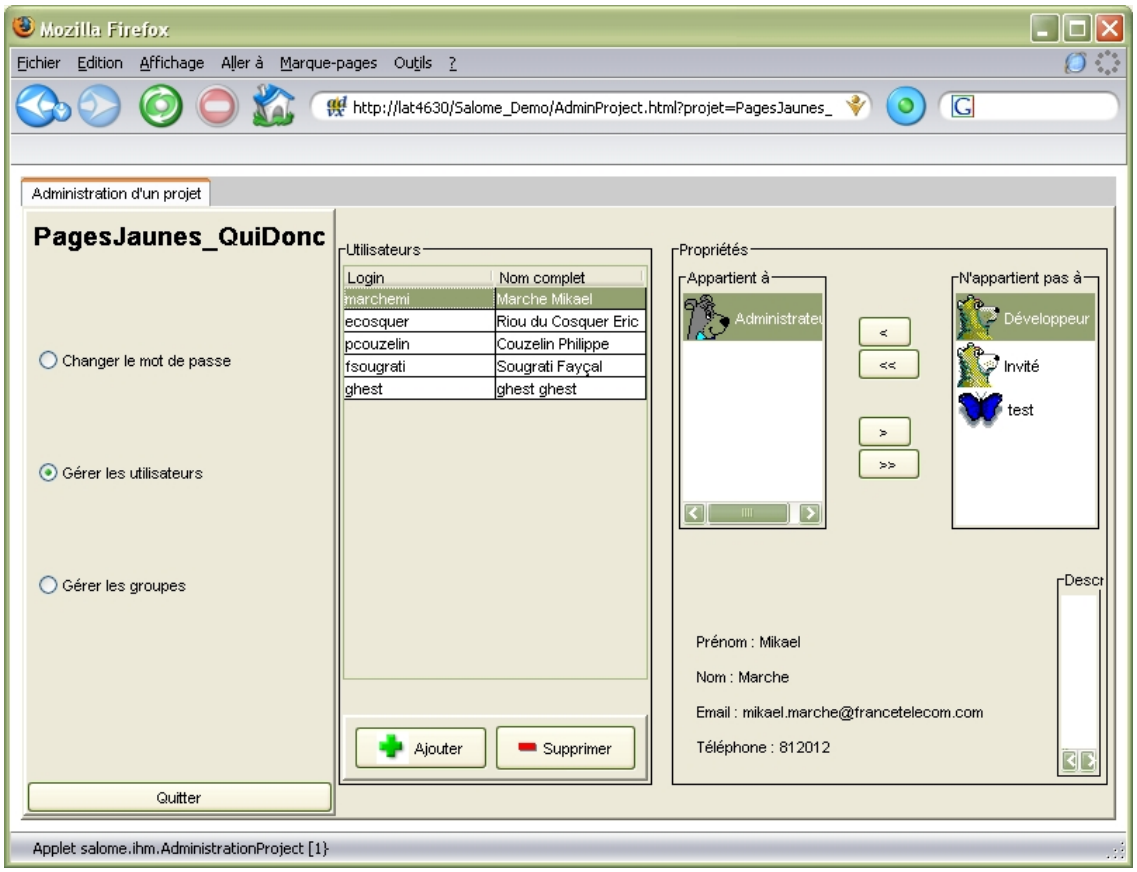


Figure 3.7: Managing a project's groups

This window (Figure 3.7) enables to:

- Look after the current list of groups;
- Look after and modify each group users' list;
- Look after the rights given to predefined groups or specific to the project;
- Modifying the rights given to a group specific to the project;
- Creating a group specific to the project.

### Creating a group specific to the project

For creating a group specific to the project, you need to (Figure 3.8):

- Click on the "New" button;
- Define the group name and the group description;
- Click on "Validate".


Figure 3.8: Adding a group in a project

The new group will be added to the list of existing groups and it will be possible to modify the rights that have been created by default.

### Modifying the rights given to a group specific to the project

For modifying the rights given to a group specific to the project, you need to (Figure 3.9):

- Select the group in the list of existing groups;
- Click on "Modify" ;
- Define the rights between "add/modify/delete" for test suites;
- Define the rights between "add/modify/delete" for test campaigns;
- Validate.

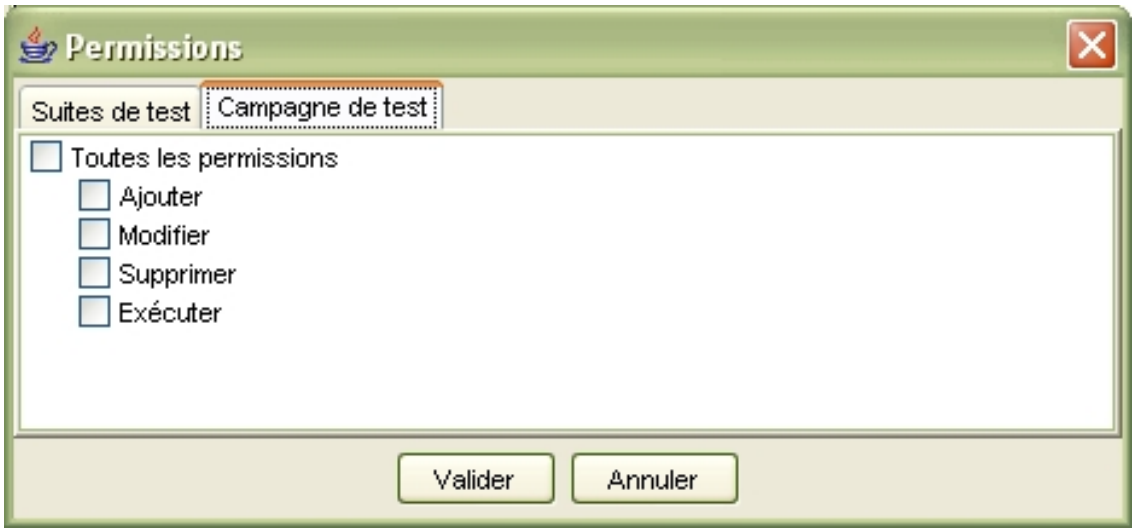


Figure 3.9: Managing a group's rights

# Chapter 4

# Use of Salomé-TMF

## 4.1 Test management

Tests are organized in a tree structure with two levels: family and suite, tests being leaves of the tree.

A family contains suites, and a suite contains tests, whether automatic or manual ones.

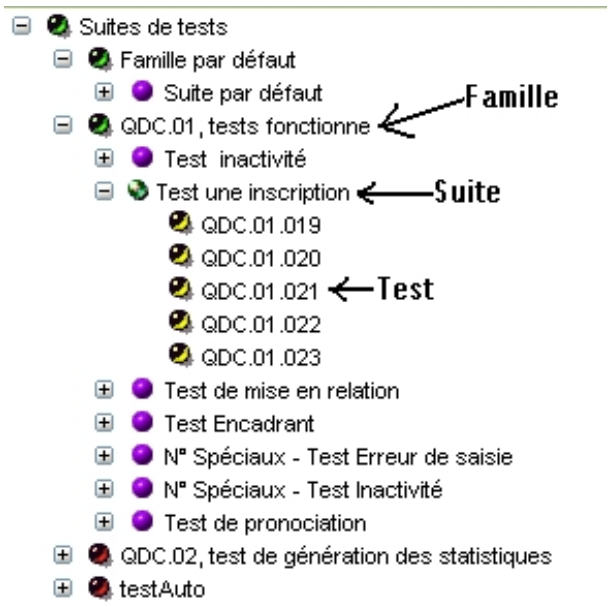


Figure 4.1: Test tree

### 4.1.1 Adding a family

For adding a family (Figure 4.2):

- Click on the button "Add a family".
- Give the name and the description of the new family.
- Validate.


Figure 4.2: Adding a test family

The new family is then created under the root directory "Test suites".

### 4.1.2 Adding a suite

For adding a suite (Figure 4.3):

- Click on "Add a suite".
- Give a name and a description to the new suite.
- Validate.


Figure 4.3: Adding a test suite

The new suite is then created:

- In the family "Default family" if this one was selected, or if a suite or a test in this default family was selected, or if no family was selected.
- In another existing family if this family was selected, or if a suite or a test in this family was selected.

### 4.1.3 Adding a manual test

For creating a manual test (Figure 4.4):

- Select the suite in which the test will be created.
- Click on "Add a test".
- Give the name and the description of the test.
- Select the category "Manual".
- Validate.

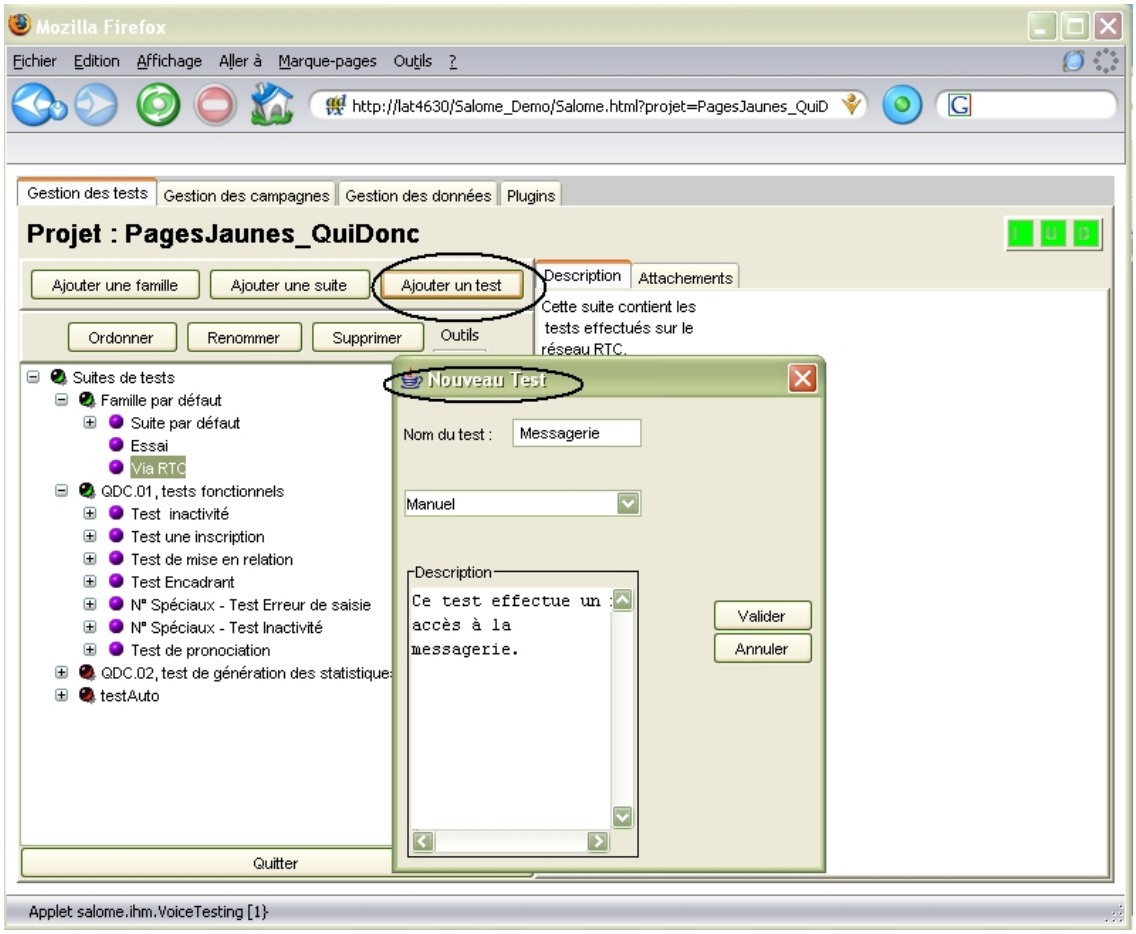


Figure 4.4: Adding a manual test

The test is added to the selected suite. You need to modify it for completing its description.

### 4.1.4 Adding parameters

The window tab "Parameters" (Figure 4.5) enables to view the parameters that can be used by this test and to add new ones (see section 4.4.2 for more details).

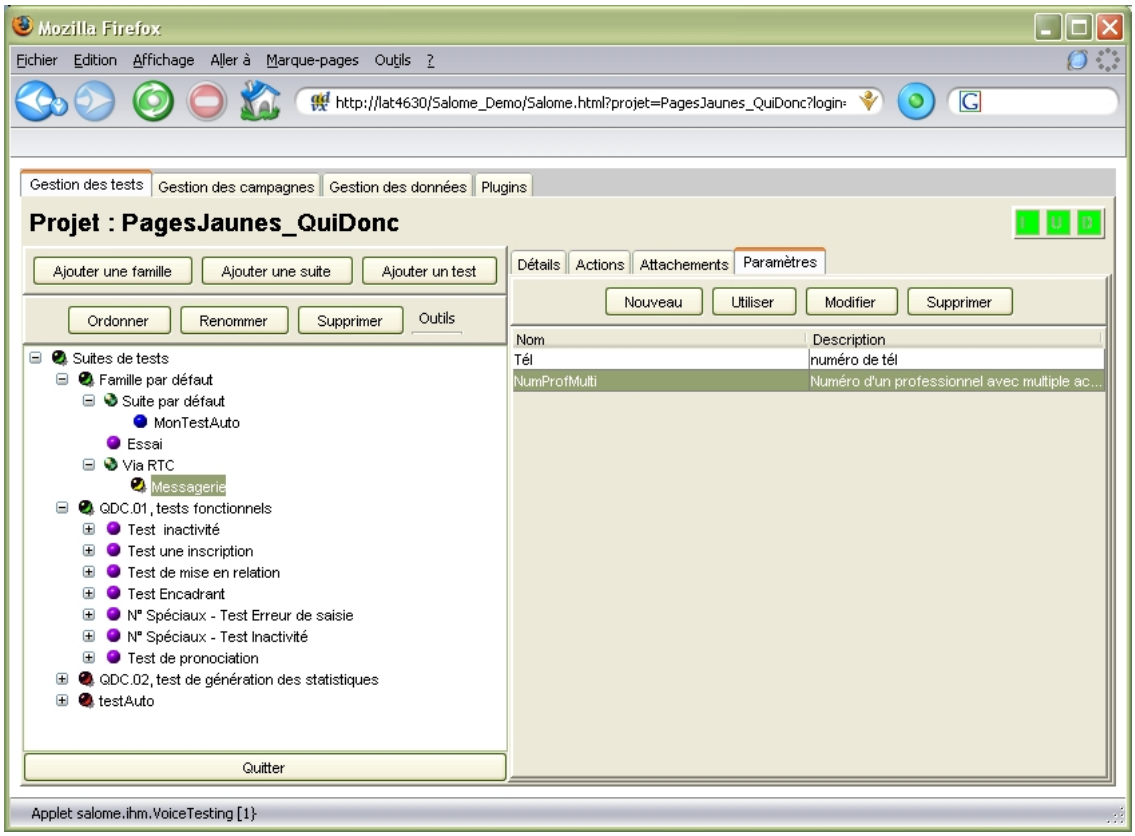


Figure 4.5: Window tab for test parameters

### 4.1.5 Adding test actions

For adding an action to a manual test (Figure 4.6):

- Go to this test.
- Activate the tab "Actions".
- Click on the button "Add".
- Complete the name, the description and the expected result.
- Add if necessary attachments, files or URLs, to which description can be given.
- Validate the creation of the action by clicking on "Validate".
- Renew those steps for creating following actions.

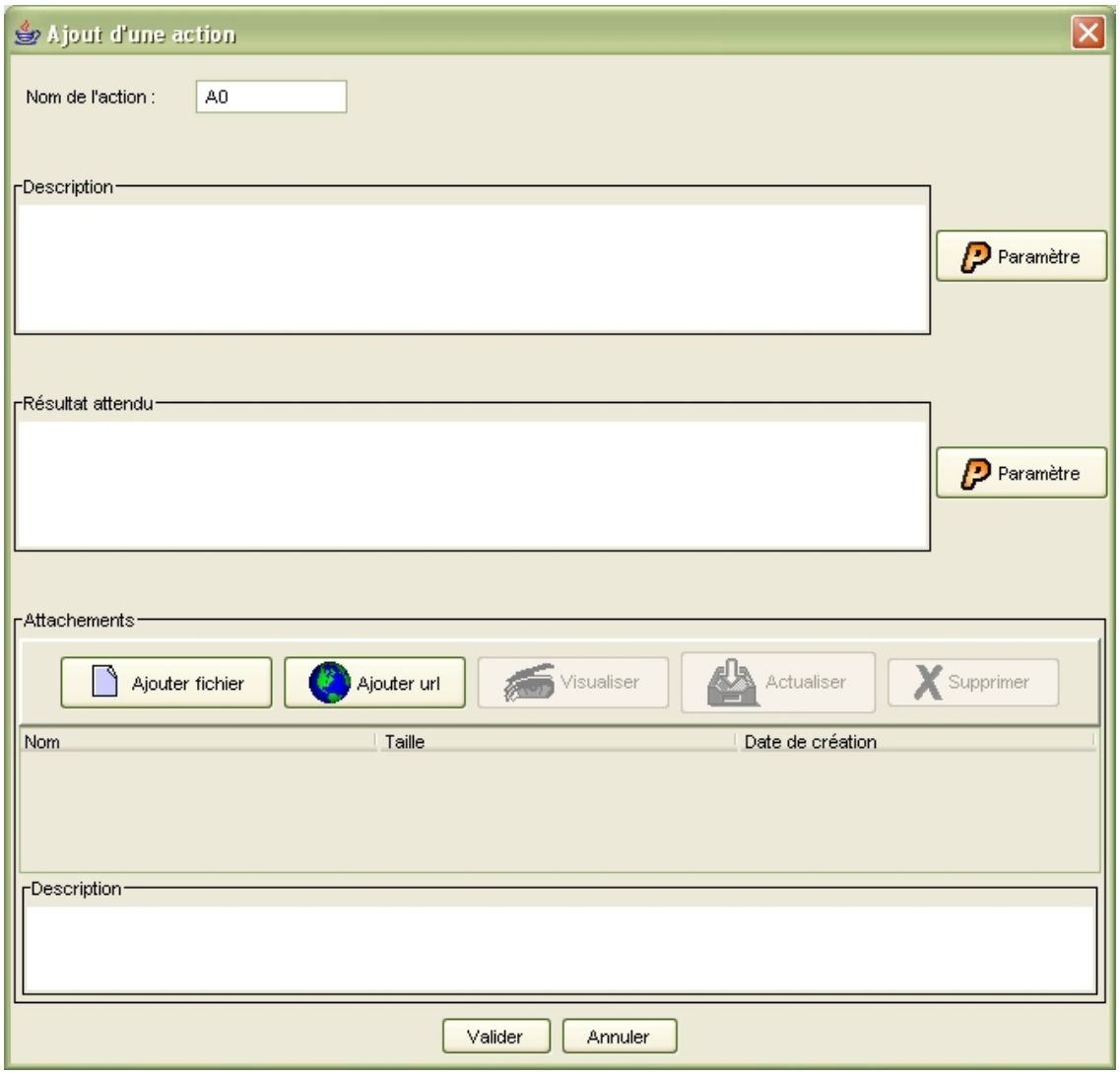


Figure 4.6: Adding a test action

### 4.1.6 Using parameters in actions

For using a parameter in the description of the action or of the expected result (Figure 4.7):

- Click on "Parameter".
- For using an existing parameter, click on "Use".
- For using a new parameter, click on "New".
- Validate.

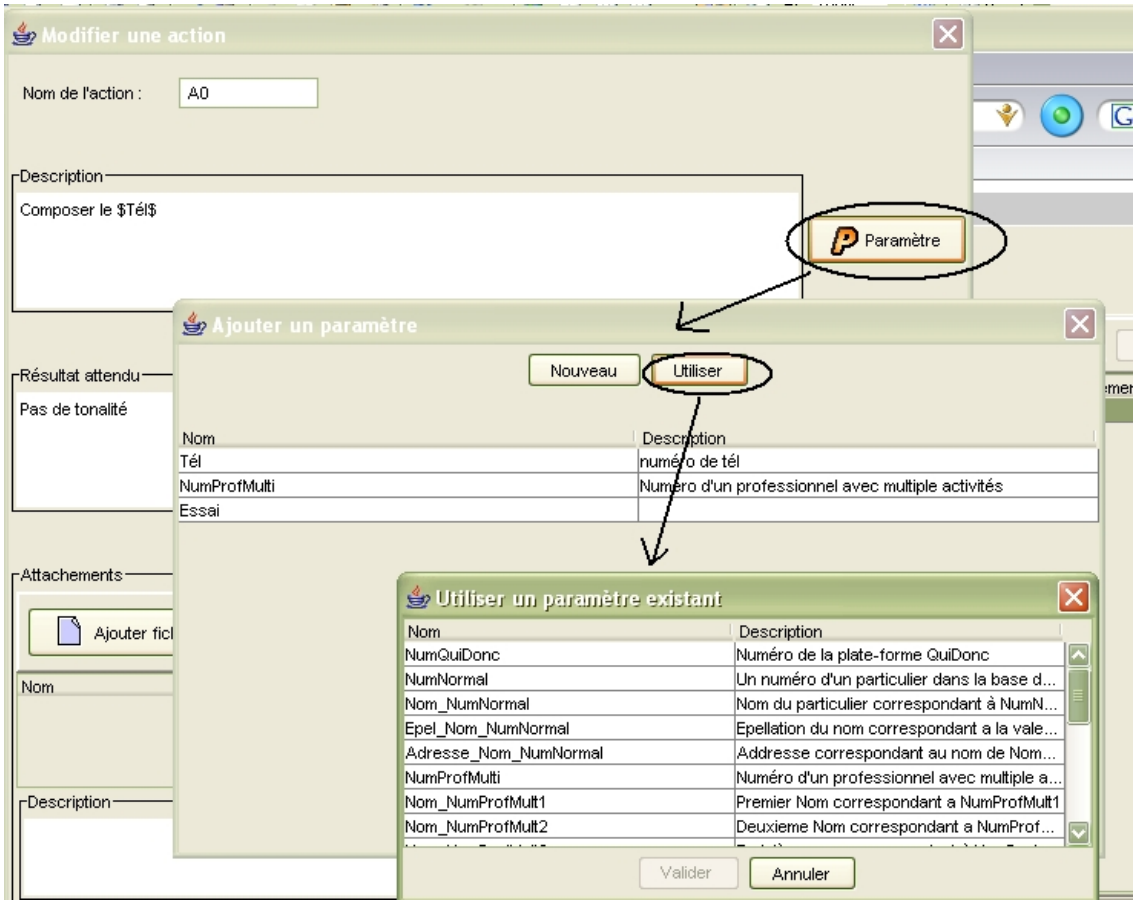


Figure 4.7: Using a parameter

### 4.1.7 Adding attachments to a test

It is possible to add attachments to a test (Figure 4.8). For this:

- Go to this test.
- Activate the tab "Attachments".
- Click on the button "Add file" or "Add URL".
- Enter the file or the URL.
- Validate.

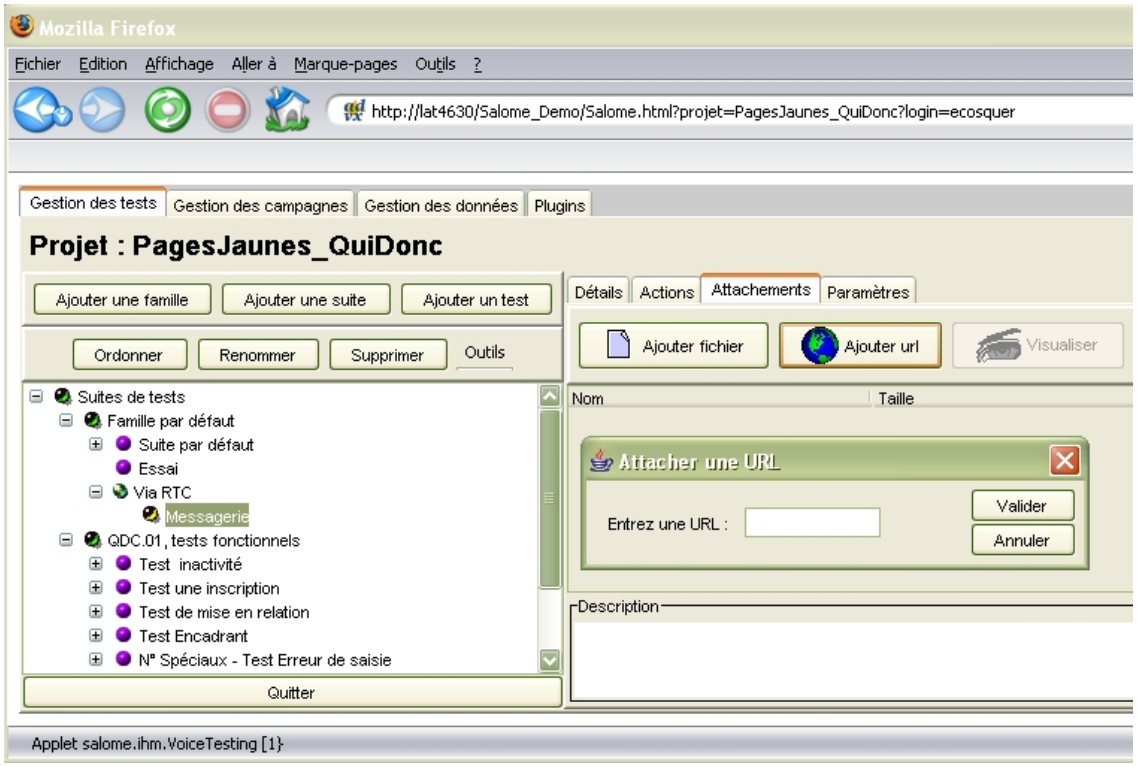


Figure 4.8: Adding an attachment

It is now possible to add a description to each attached element and to visualize them.

## 4.2 Campaign management

From the window tab "Campaign management, several features can be accessed:

- Creation/modification of campaigns.
- Creation of campaign executions.
- Campaign executions launch.
- Results consultation.

### 4.2.1 Campaign creation

The button "Create a campaign" (Figure 4.9) enables to create a new campaign.

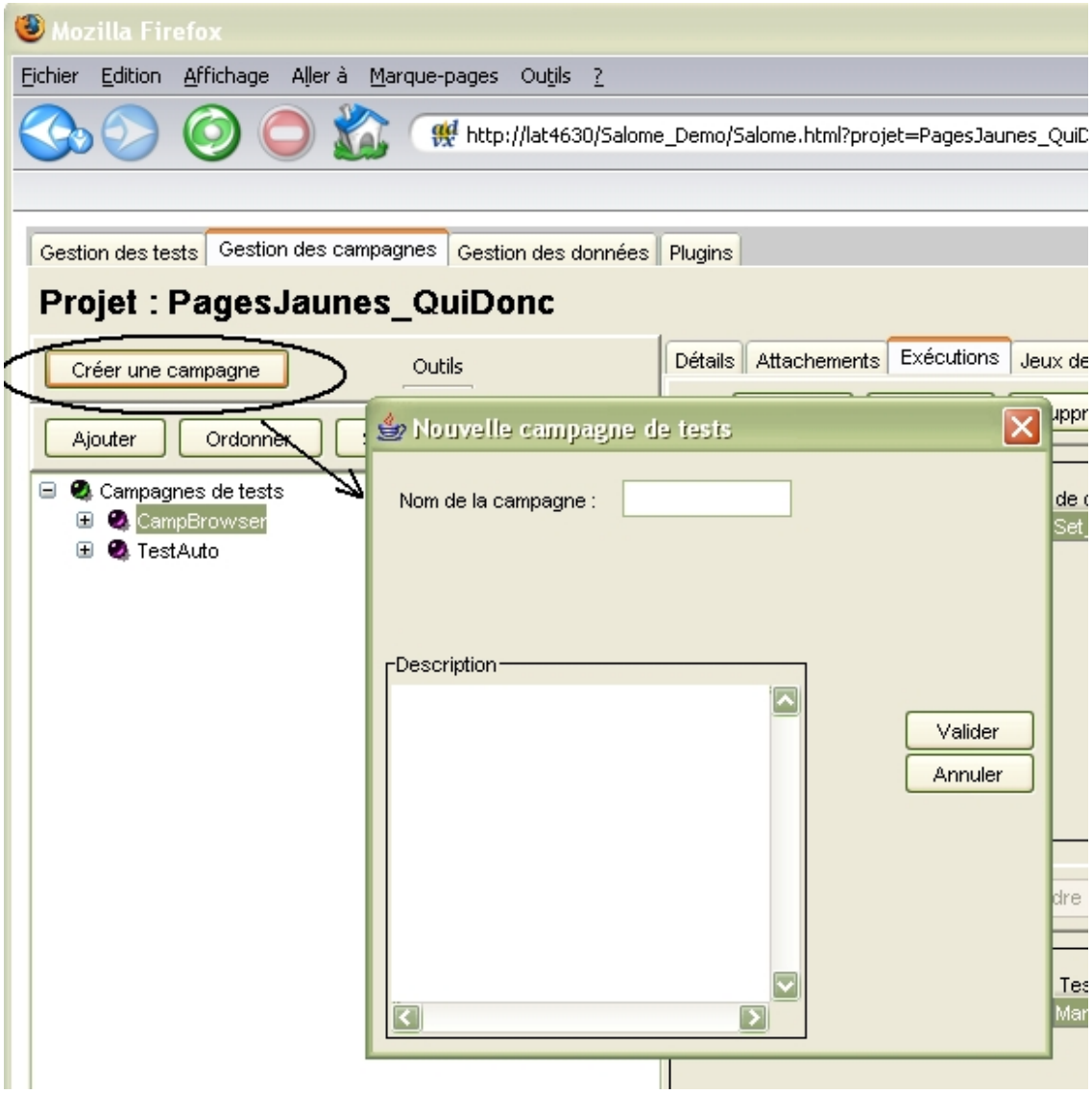


Figure 4.9: Adding a test campaign

### 4.2.2 Adding tests to a campaign

For adding tests to a campaign (Figure 4.10):

1. Select the campaign.
2. Click on the button "Add".
3. A window pops up which enables:
   - To add all the tests member of the test suites of a given family; or,
   - To add all the tests in a given test suite; or,
   - To add a test in particular.

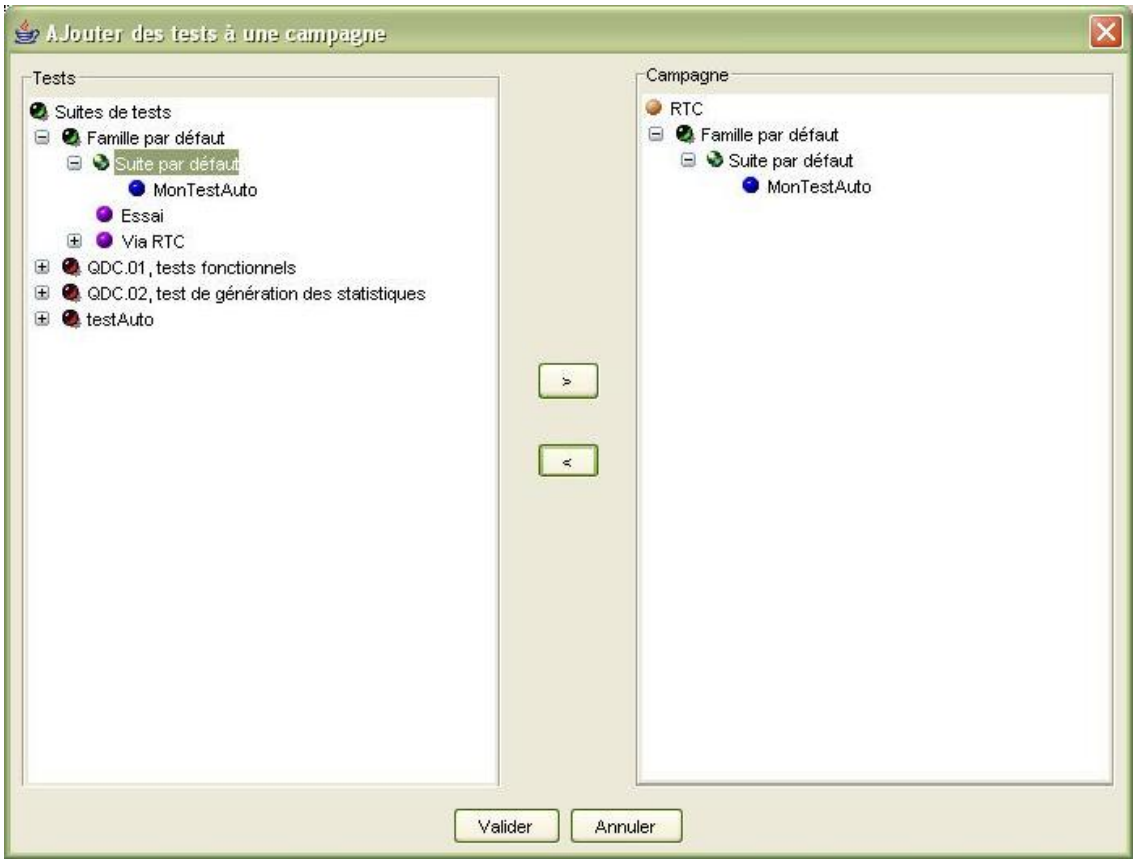


Figure 4.10: Adding tests to a campaign

After validation, the tests can be viewed in the campaign, and organized in their families and suites.

### 4.2.3 Defining a dataset for a campaign

The point is to give values to the parameters used in the tests of the campaign, if there is any. For defining a dataset (Figure 4.11):

1. Select a campaign.
2. Activate the window tab "Dataset". The list of existing datasets appears.
3. Click on the button "Add", a window pops up:
   - Give a name to the dataset.
   - Give a description to the dataset.
   - Give a value for all parameters.
   - Validate.

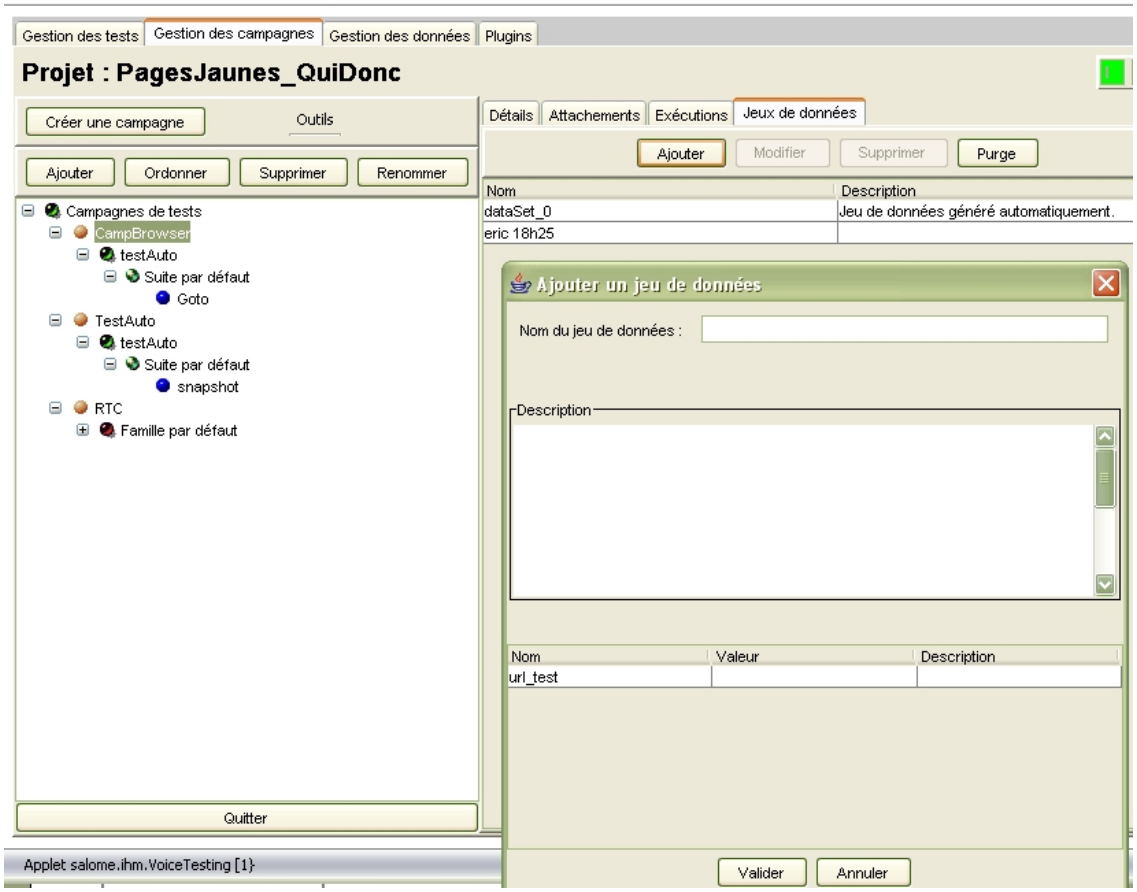


Figure 4.11: Adding a dataset

### 4.2.4 Defining a campaign execution

For launching the execution of tests of a campaign, it is necessary to associate to the campaign an execution which defines an environment and a dataset (Figure 4.12).

For defining an execution for a campaign:

1. Select the campaign.
2. Select the window tab "Execution".
3. Click on the button "Add", a window pops up:
   - Give a name to the execution.
   - Choose an existing dataset or create a new one.
   - Choose an initialization script which will be executed before the tests.
   - Choose a restitution script which will be executed at the end of the campaign execution.
   - Attach one or several files or URLs.
   - Give a description to the execution.
   - Validate.

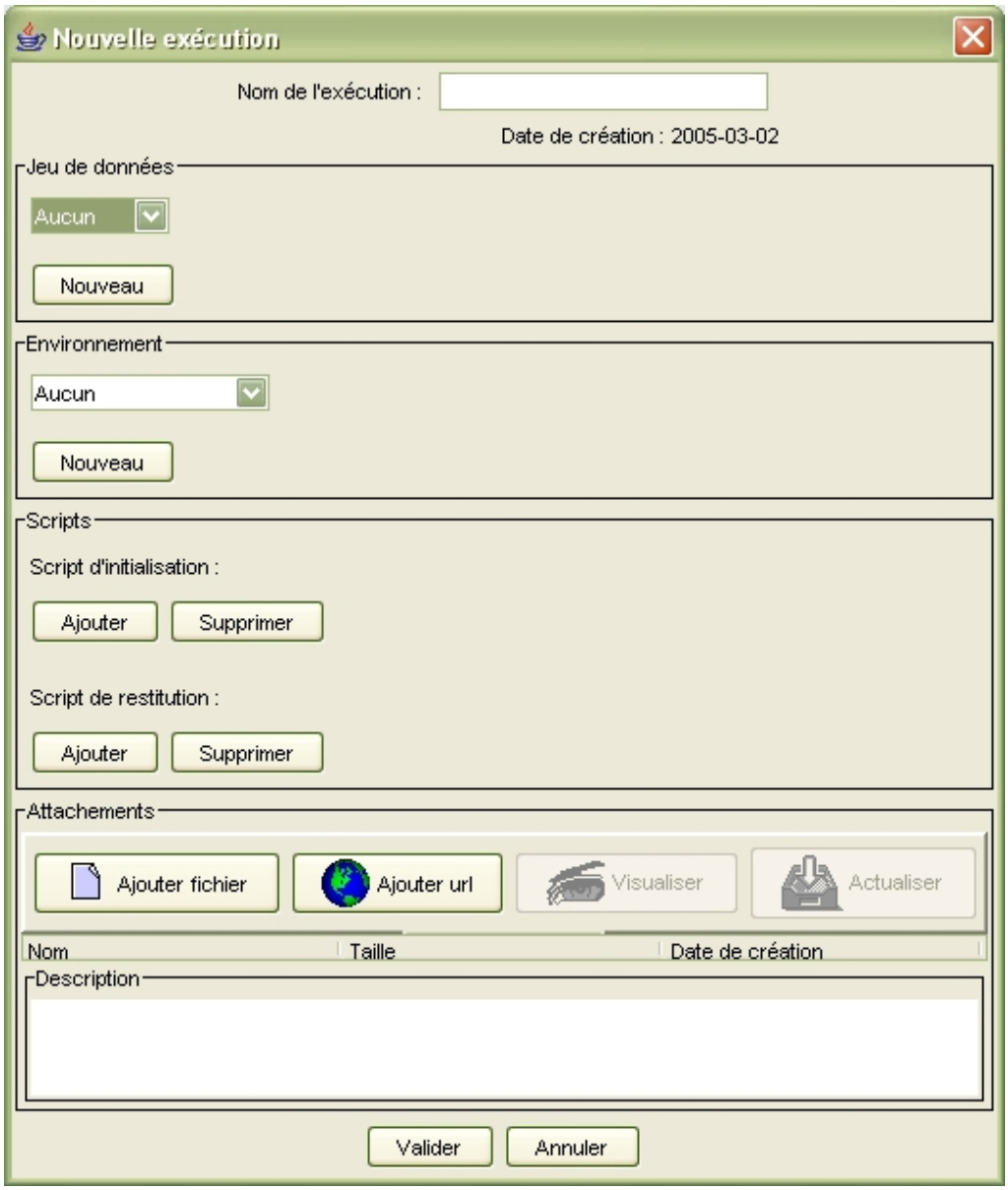


Figure 4.12: Adding an execution to a campaign

### 4.2.5 Managing executions' launch

**Launching an execution**

For launching one (or several) executions (Figure 4.13):

- Select the corresponding campaign.
- Activate the window tab "Execution".
- Select the desired execution(s) in the list of existing executions.
- Click on the button "Launch".

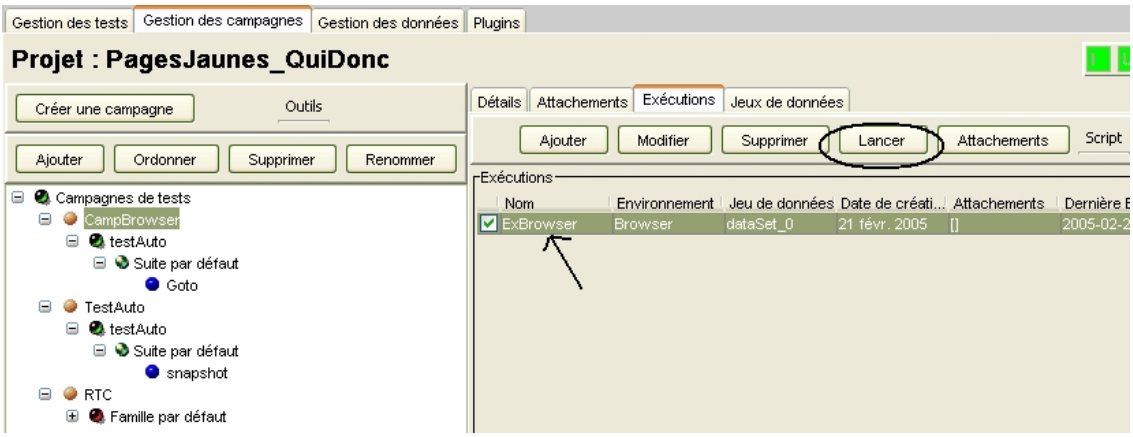


Figure 4.13: Launching an execution

### Resume an execution

When the launch of an execution has been interrupted, it is possible to resume:

- Select the corresponding campaign.
- Activate the tab "Execution".
- Select the desired execution in the list of existing executions. The list of results for its different launches appears.
- Select one of those results.
- Click on "Resume".

### Consulting the results of an execution launch

For consulting the results of an execution (Figure 4.14):

- Select the corresponding campaign.
- Activate the tab "Execution".
- Select the desired execution in the list of existing executions. The list of results for its different launches appears.
- Select one of those results.
- Click on "Details". The results for all tests of the execution appear.

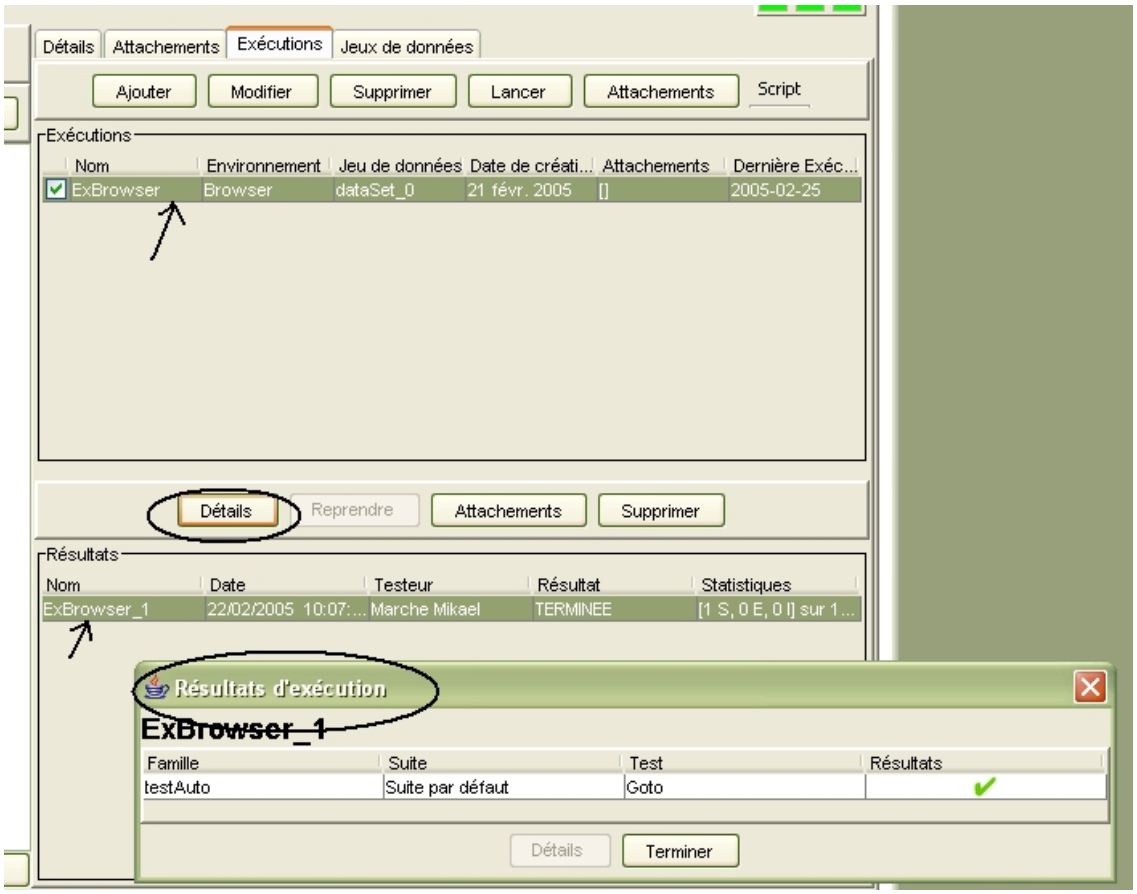


Figure 4.14: Consulting execution results

## 4.3 Managing environments

Environments enable to describe the context in which tests included in a campaign will be executed.

Managing environments is done from the window tab "Data management" (Figure 4.15).

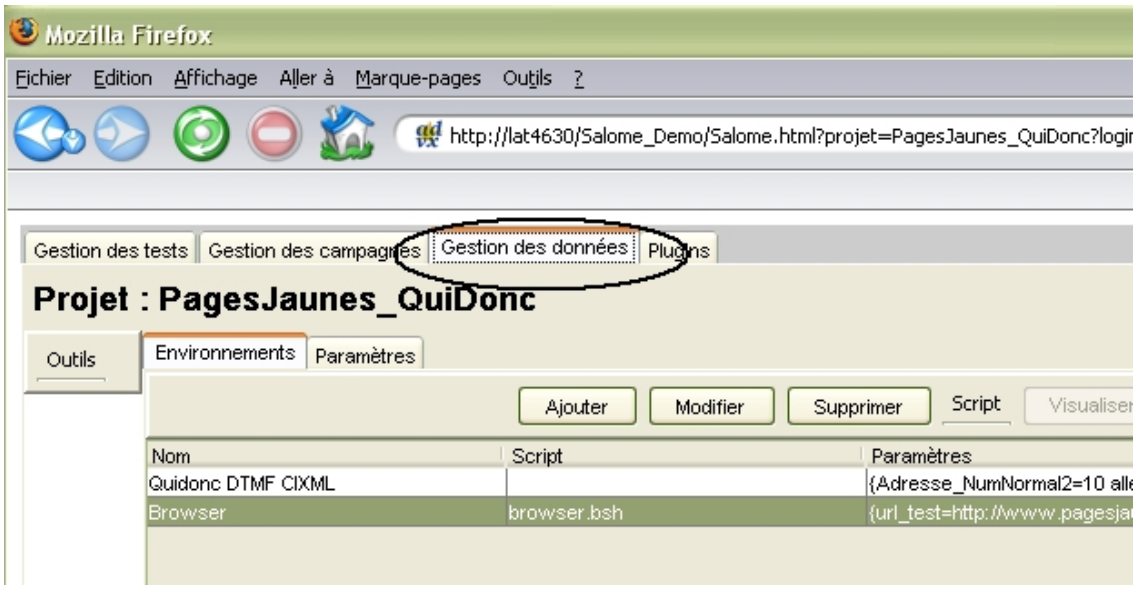


Figure 4.15: Managing a project's environments

### 4.3.1 Adding a new environment

This feature can be accessed by clicking on the button "Add" (Figures 4.15, 4.25). It enables to describe the elements that constitute an environment (Figure 4.16).

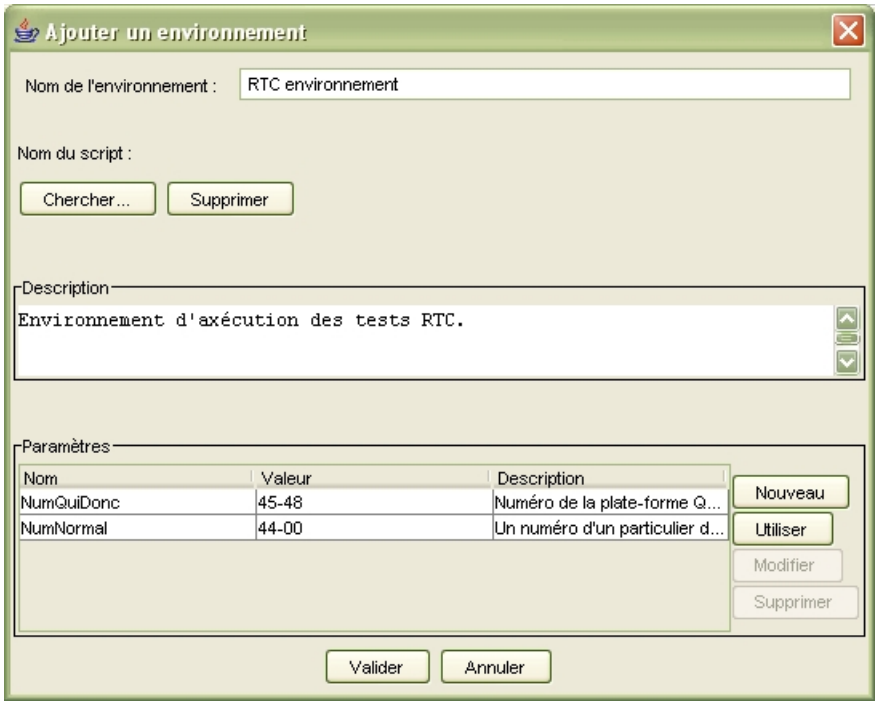


Figure 4.16: Description of an environnment

Fields defining an environment are:

- Environment name.
- Script: the selected script will be executed before the tests belonging to the corresponding campaign.
- Environment description.
- Environment parameters: those parameters, as soon as they are given a value, can be used in tests and in the scripts that can be attached to environments and executions.

### 4.3.2 Modifying an environment

This feature can be accessed by clicking on the button "Modify" (Figure 4.15). It enables to modify an environment, if it has not been previously used in an execution.

### 4.3.3 Deleting an environment

This feature can be accessed by clicking on the button "Delete" (Figure 4.15). t enables to delete the selected environment, if it has not been previously used in an execution.

### 4.3.4 Defining an environment's parameters

It is possible in an environment to define or use parameters of a project. Those * parameters can then be valued for test execution. Thiose features are described in sections 4.4.3 and 4.29.

## 4.4 Parameters management

The global management of parameters for a whole project is done from the window tab "Data management" (Figure 4.17). Nevertheless, as we have seen in section 4.1.4, a parameter can be created from several points.

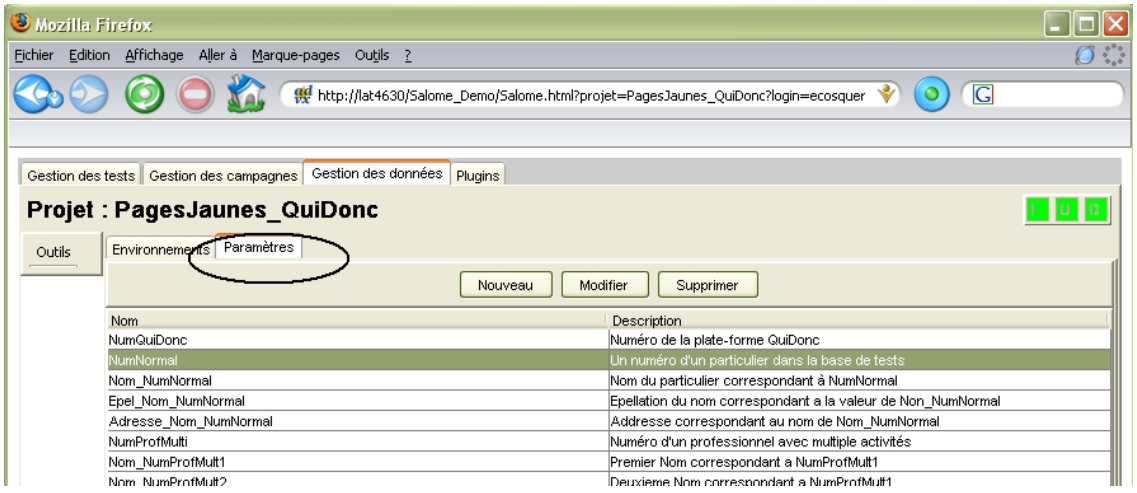


Figure 4.17: Managing a project's parameters

This window tab enables to:

- Consulting parameters defined in the project and used in tests or environments.
- Creating new parameters.
- Modifying existing parameters by changing their description.
- Deleting parameters.

### 4.4.1 Creating a new parameter from the *data management* tab

The whole set of parameters that can be used in a project can be browsed from the window tab "Data management", by clicking on "Parameters". You can also add new parameters by clicking on "New" (Figure 4.18).

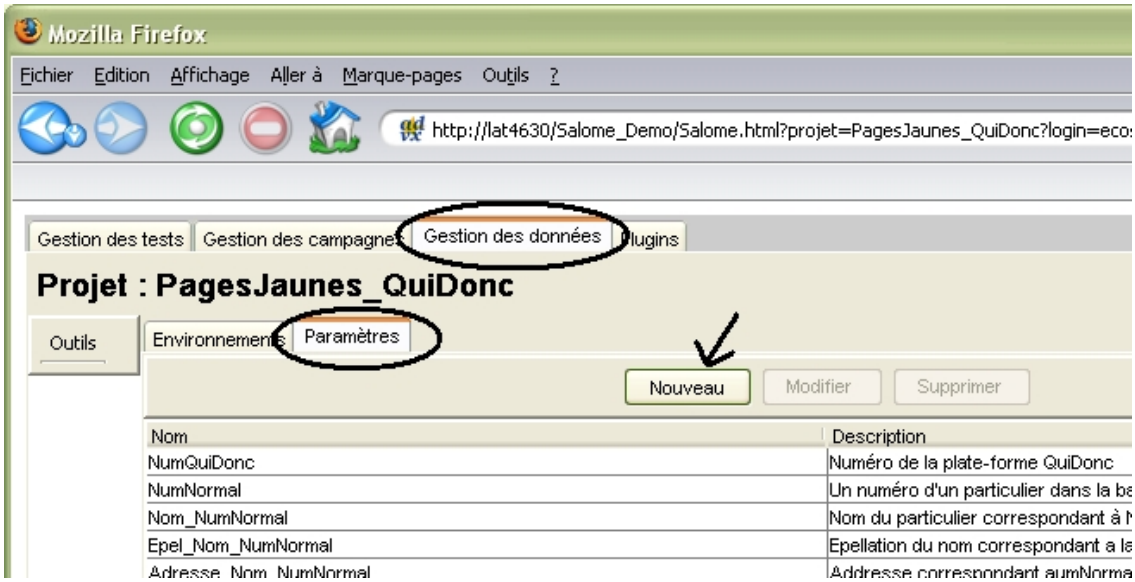


Figure 4.18: Adding parameters to the project

For creating a parameter (Figure : 4.19), it needs to be given a name and a descritpion and then to be validated. The created parameter can then be used by tests and environments.

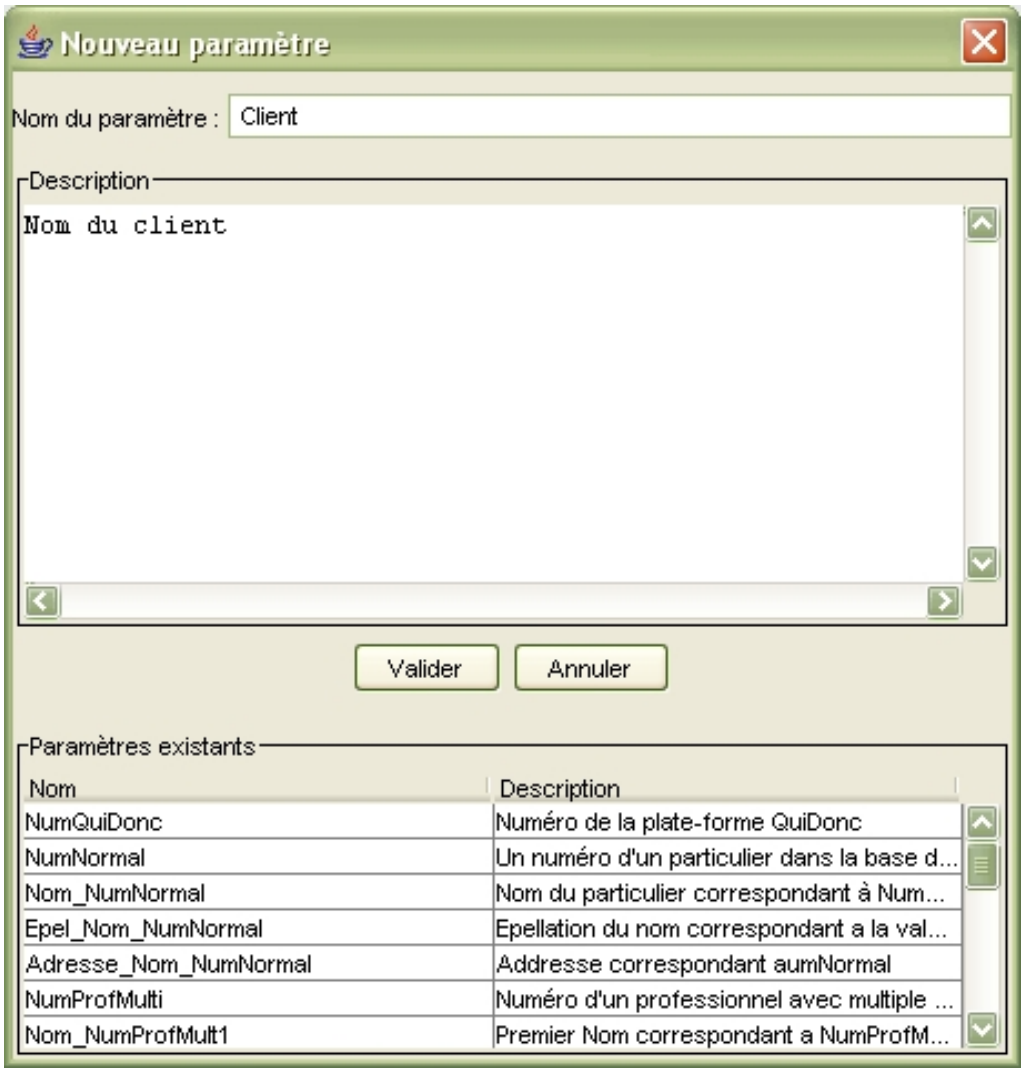


Figure 4.19: Parameter description

### 4.4.2 Creating a parameter from a test action

When describing a manual test action, it is possible to create a new parameter that will be used by the test action. For this, starting from the window for adding a test action (Figure 4.20), click on the button "Parameter".

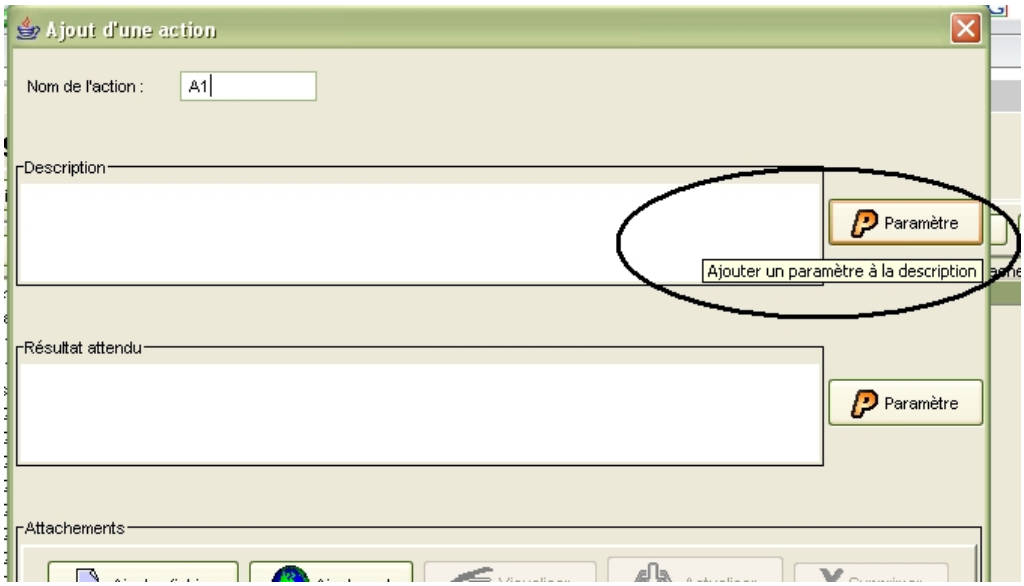


Figure 4.20: Using a parameter from a test action

The window (Figure 4.21) for adding parameters appears, click then on "New". It is also possible to use an existing parameter. For this, select the desired parameter and validate.

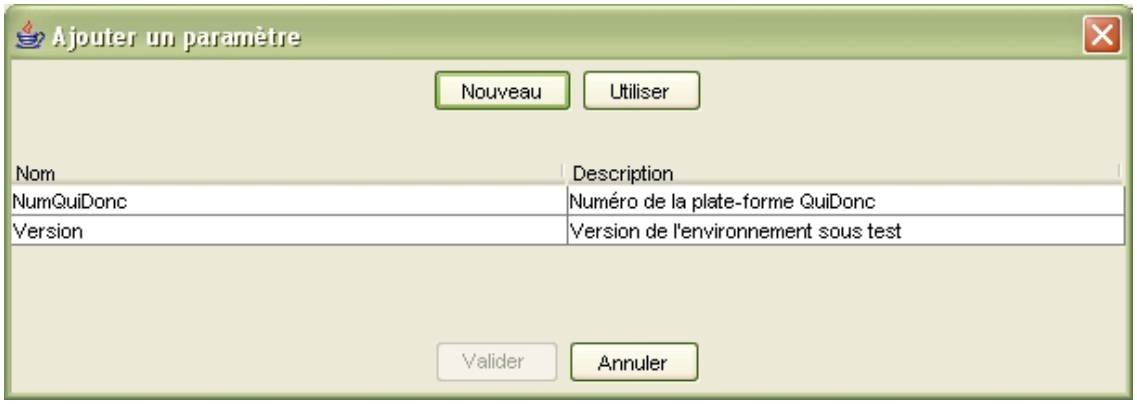


Figure 4.21: View parameters used by a test

For finishing the parameter creation (Figure 4.22), it needs to be named and described before validation.

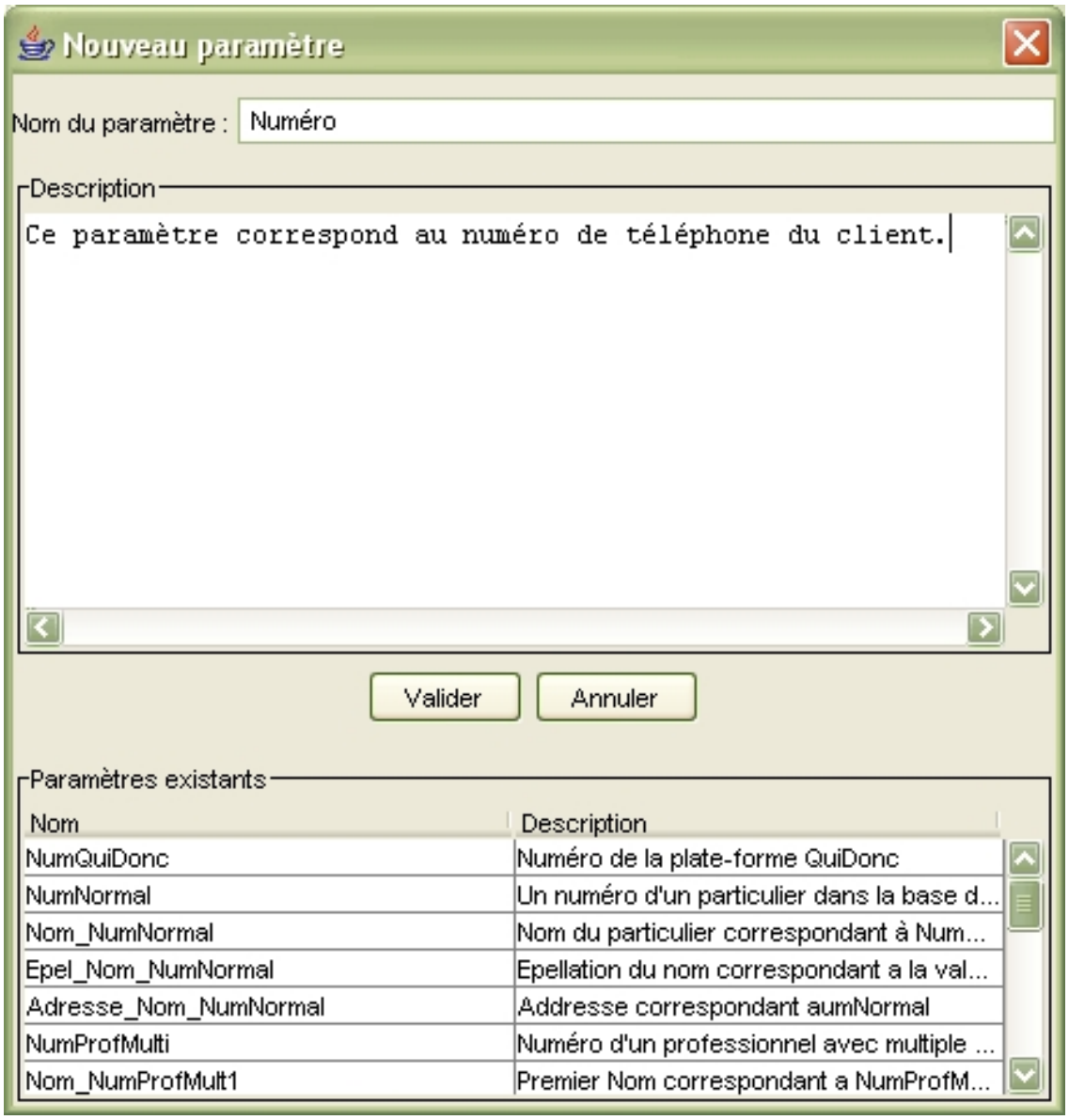


Figure 4.22: Description of parameters used by a test

Next, for using the parameter in the current action (Figure 4.23), select the parameter which has just been created and then validate.


Figure 4.23: Using a parameter

At last, the parameter appears in the action deszcription (Figure 4.24).


Figure 4.24: Parameter in an action

### 4.4.3 Creating a parameter from an environment

Starting from the window tab "Data management" (Figure 4.25), by selecting "Environments" and by clicking on "Add", it is possible to add a new environment in which new parameters can be created.

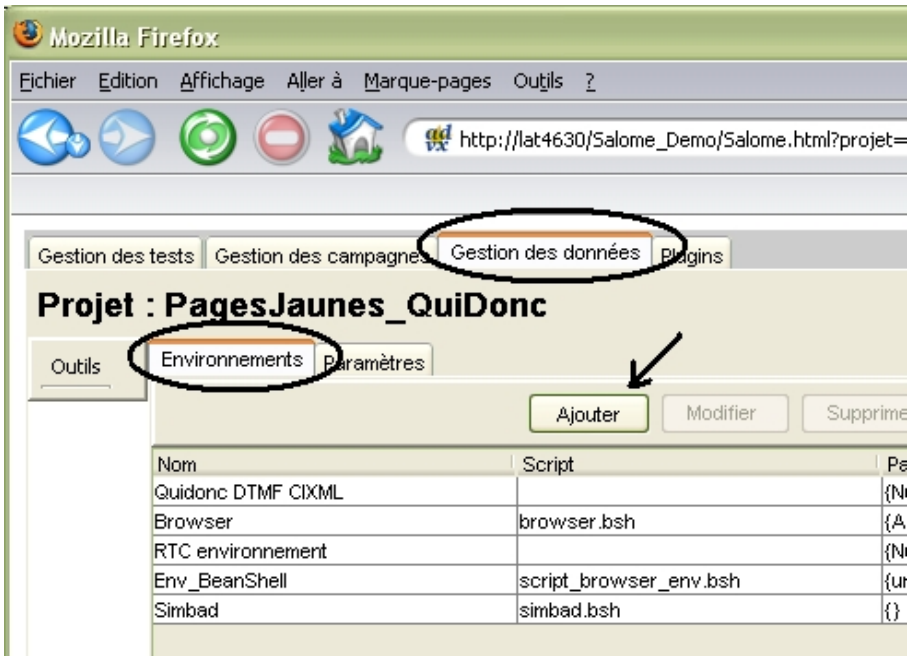


Figure 4.25: Adding an environment

The window for adding an environment appears (Figure 4.26), click then on "New".

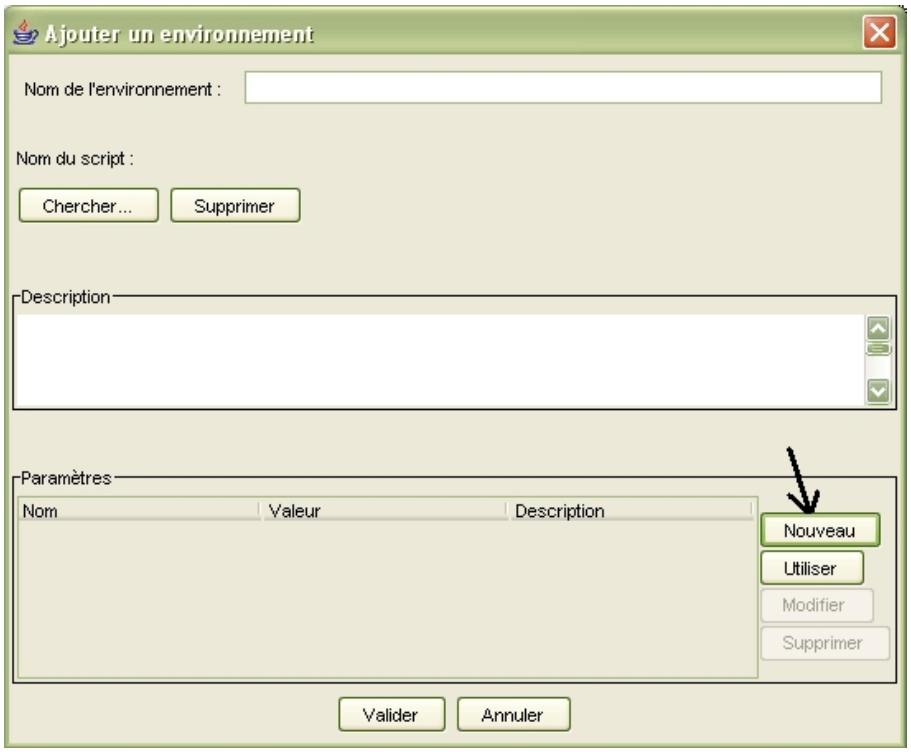


Figure 4.26: Creating a parameter in an environment

A new parameter can be created (Figure 4.27).

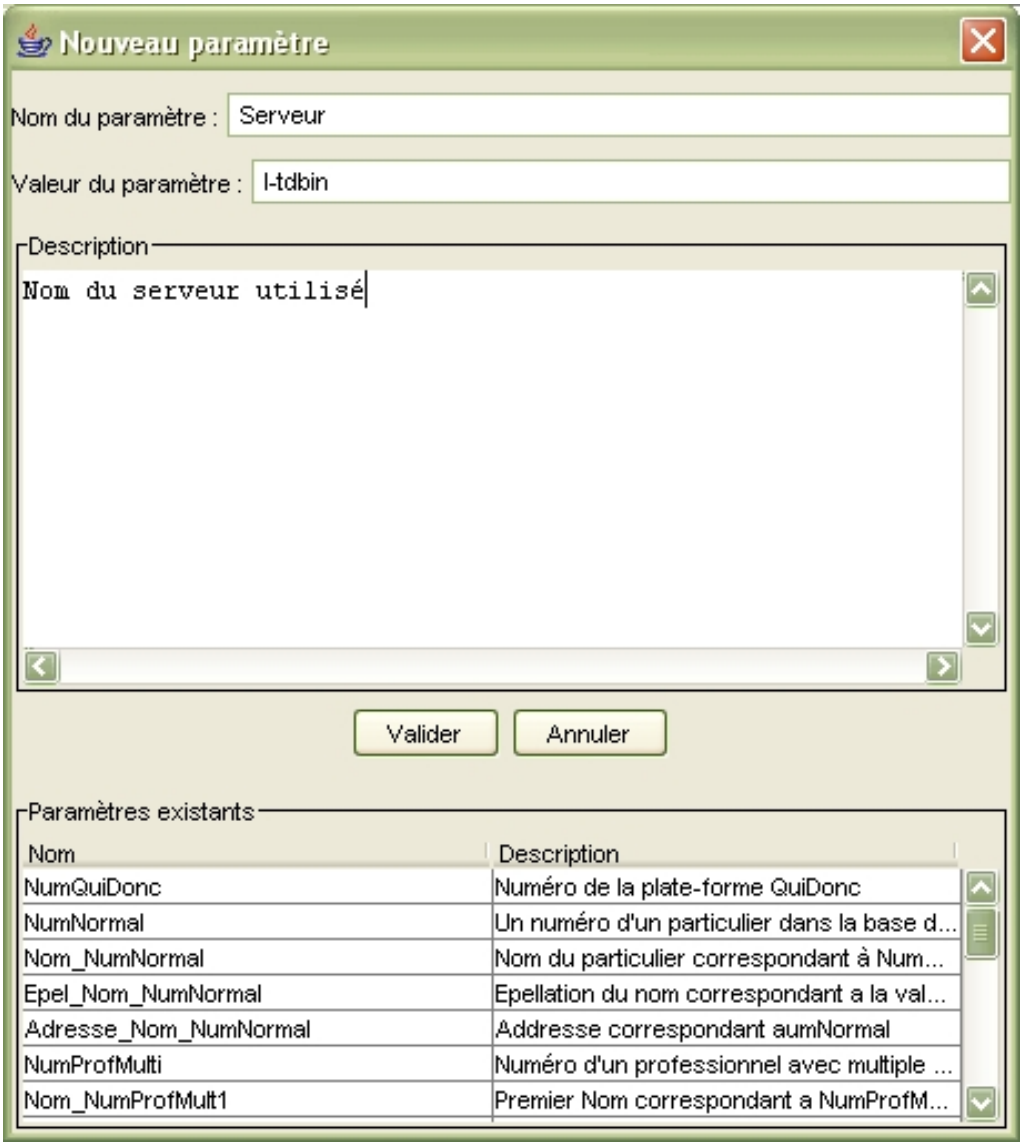


Figure 4.27: Declaring a parameter in an environment

### 4.4.4 Using a parameter in an environment

You can use existing parameters in an environment from the window tab "Data management" by selecting "Parameters":

- Select an environment.
- Click on "Modify", a window pops up (Figure 4.28).

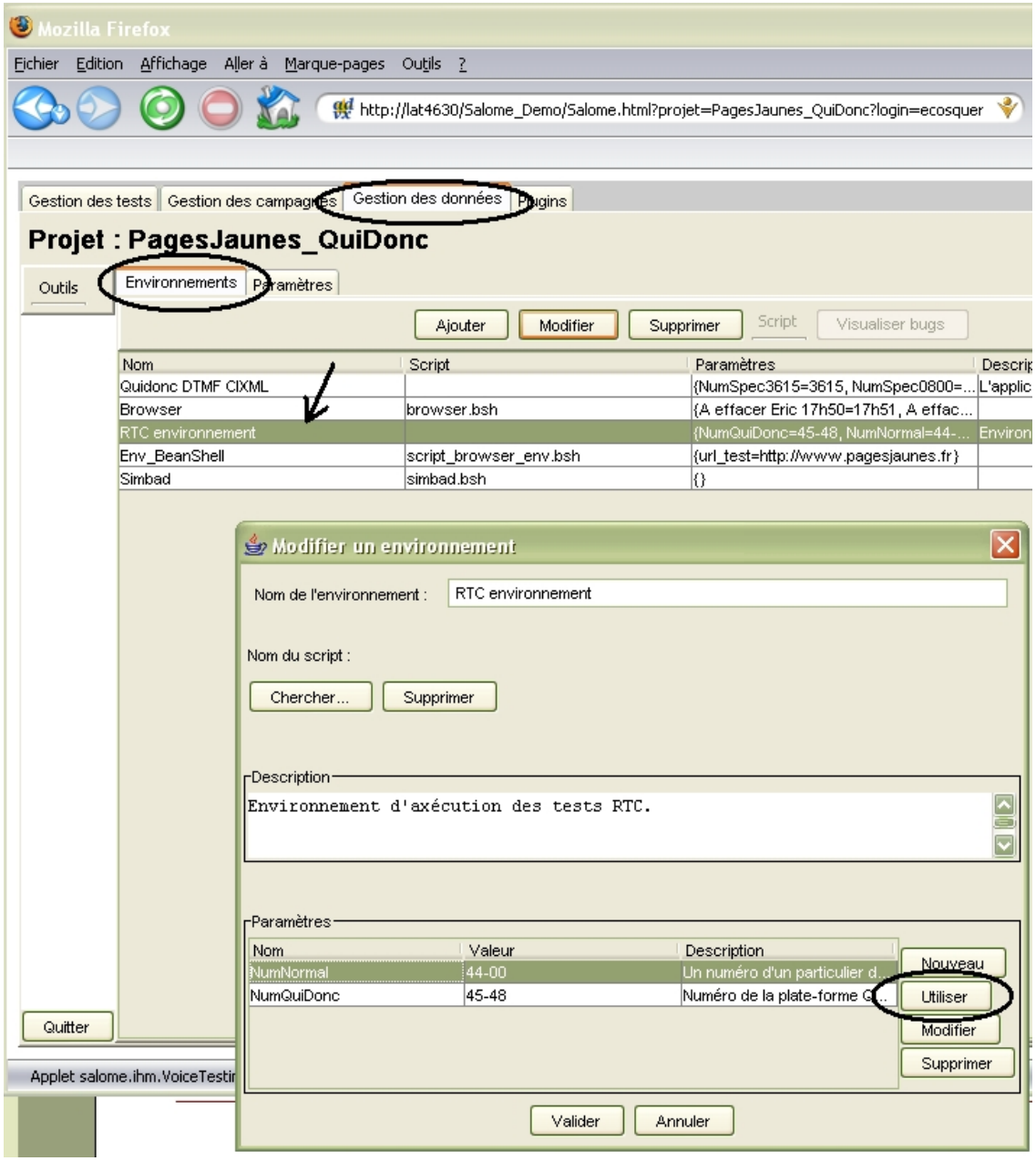


Figure 4.28: Modifying an environment

- Click on "Use", the window "Use an existing parameter" pops up (Figure 4.29).

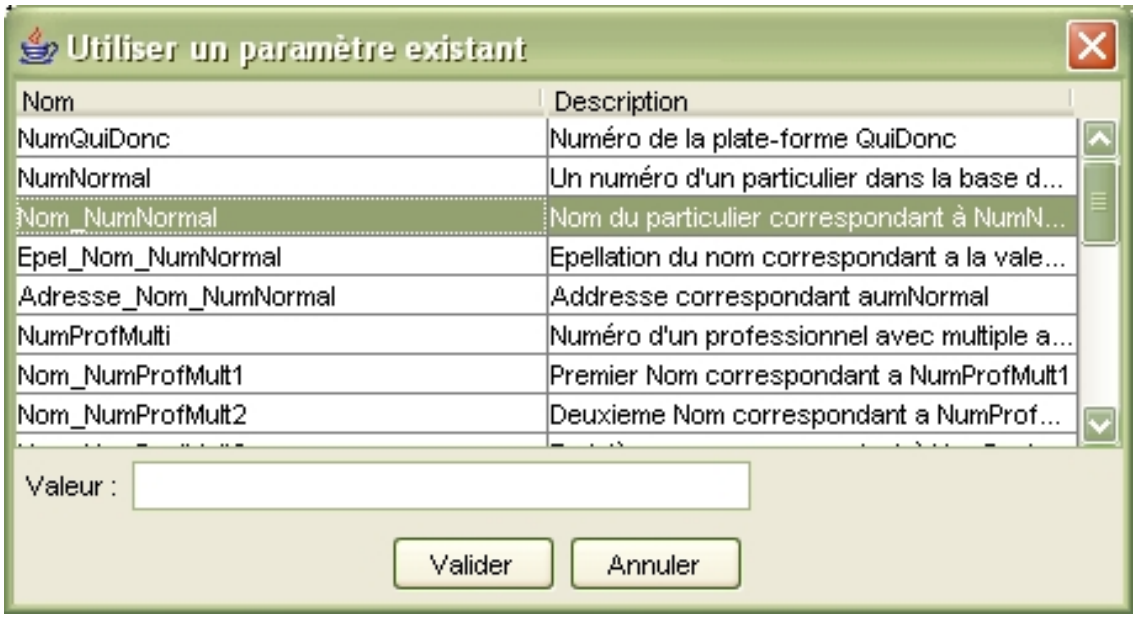


Figure 4.29: Using a parameter in an environment

- Select the desired parameter, and give it a new value.
- Validate.

This valued parameter is integrated to the environment.

### 4.4.5 Giving value to a parameter

A parameter can be valued through a dataset linked to an execution. When a test is launched which uses a parameter, the value for this parameter is:

- The one given to the parameter in the environment related to the corresponding execution, if this parameter is defined in this environment.
- The one given to the parameter in the dataset linked to the execution, if this parameter is not defined in the environment related to the execution.

# Chapter 5

# Automatisation

If your Salomé-TMF installation has got plugins for automatic executions of tests and/or scripts (for environments and test executions), it is possible to create tests and/or automatic scripts that will be managed by the plugin motor.

The plugin architecture that is developed in Salomé is open and enables you to create your own plugins. For more information on this, see the manual on plugin development.

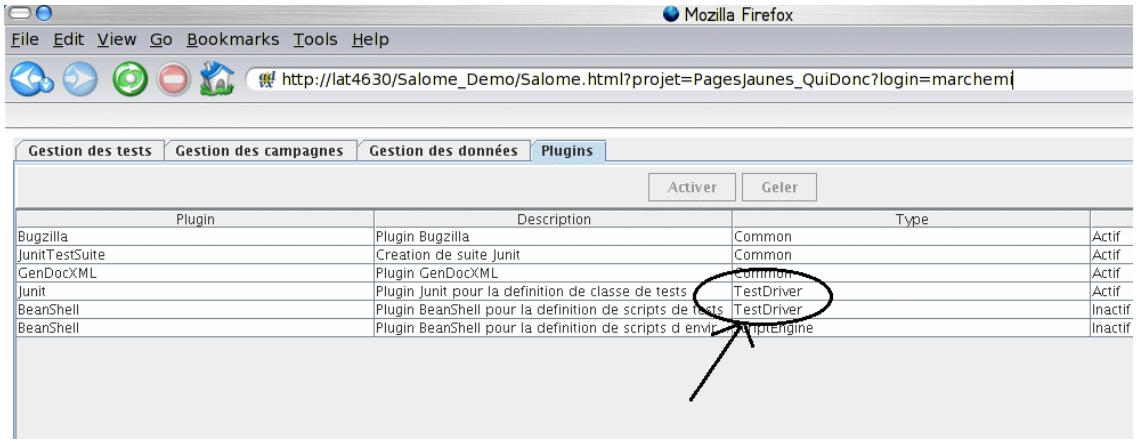


Figure 5.1: Plugins in Salomé-TMF

For looking after the plugins in Salomé-TMF, open the tab "Plugins" in the main window (Figure 5.1). The plugins for automating tests are in the category TestDriver, and those for environment scripts are in the category ScriptEngine.

This chapter describes how to use automatisation plugins in a general framework, but some features can vary from a plugin to another. For more information on a specific plugin, check the corresponding documentation.

## 5.1 Creating an automatic test

The procedure for creating automatic tests is the following one:

- In the test plan tab (Figure 1.1), select the test suite in which the test will be added;

- Click on "Add a test";
- Enter the name and the test description;
- Select the plugin to use (Figure 5.2, possible examples are the JUnit or HTTPUnit plugins);
- Validate.

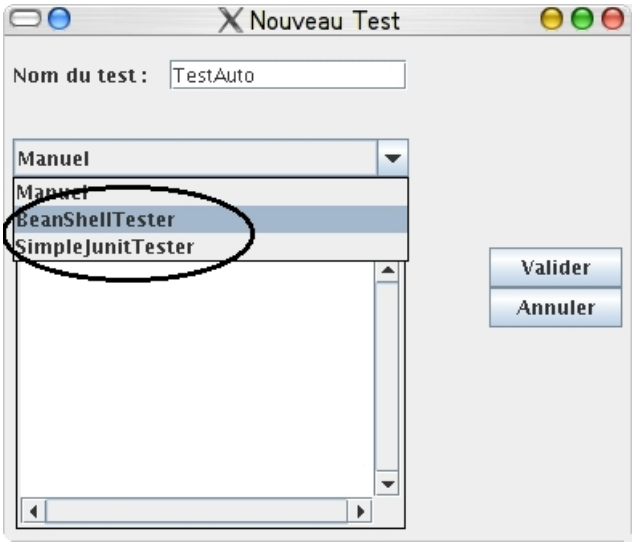


Figure 5.2: Choosing an automatic test driver

The test is added to the selected test suite. You will then define the execution code depending upon the plugin for describing entirely the test. For this, click on "Add" in the tab "Script" of the test (Figure 5.3).

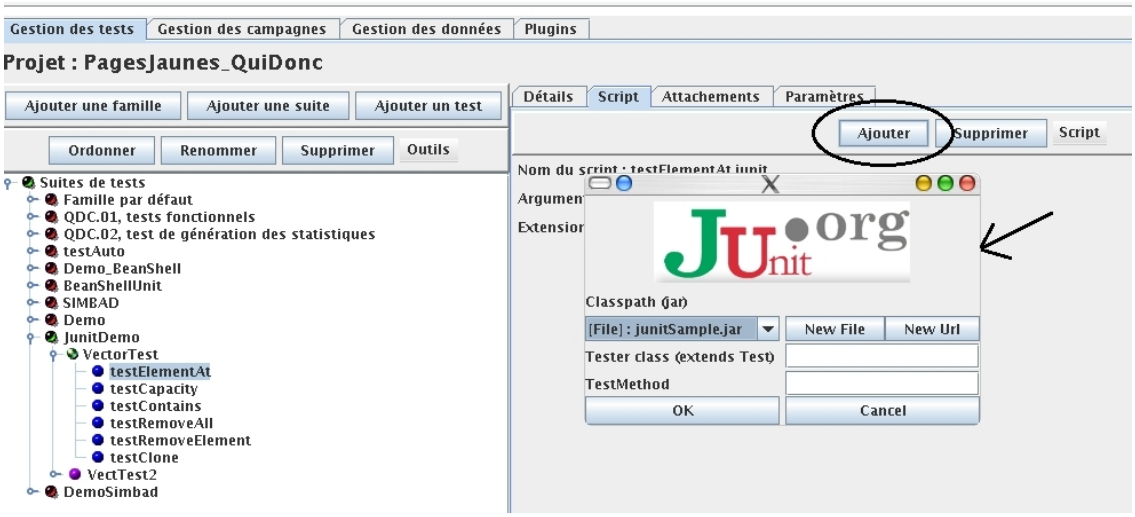


Figure 5.3: Adding a script (code) for an automatic test

Like manual tests, the automatic tests can use parameters (see section 4.1.4) that will be valued at execution time. Please note that the use of parameters depends upon the selected plugin. Look the specific documentation for more informations.

## 5.2 Modifying an automatic test

Modifying an automatic test, except for the script part, is done in the same way than for a manual test (name, description, attachment, parameters). Modifying the test scrip depends upon the plugin and can be done from the script menu in Figure 5.4.

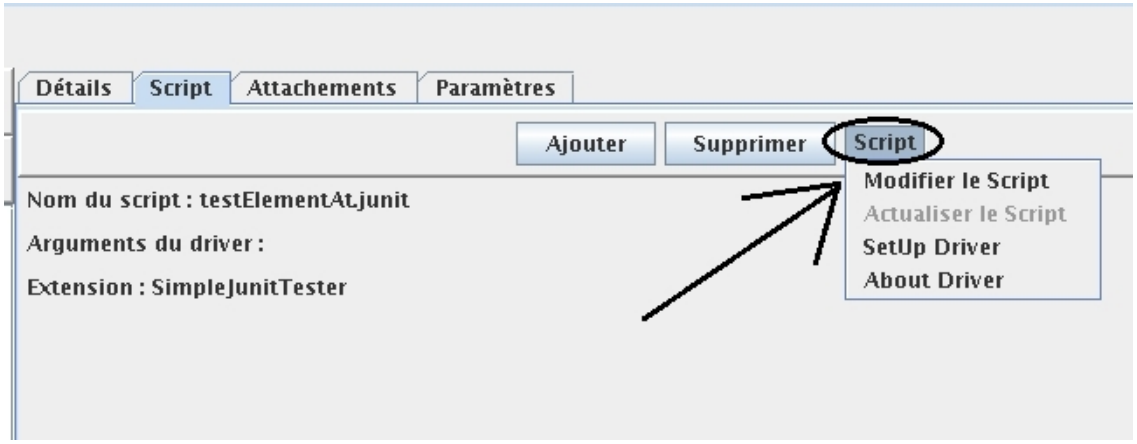


Figure 5.4: Modifying an automatic test script (code)

- The field *Modify the script* calls the modification feature at the plugin level.
- The field *Update the script* updates the script in Salomé-TMF database.
- The field *SetUp Driver* calls the plugin configuration feature.
- The field *About Driver* prints plugin informations.

## 5.3 Using scripts in environments and executions

Like automatic tests, the environments and the executions can use automatic scripts that are built using plugins of the category ScriptEngine. An example of such plugins is the BeanShell plugin.

When executing a test campaign, the environment script is executed first, then the initialization script for the execution, the test scripts, and then the final script for the execution.

The Figure 5.5 shows how to add one or several scripts to an execution.

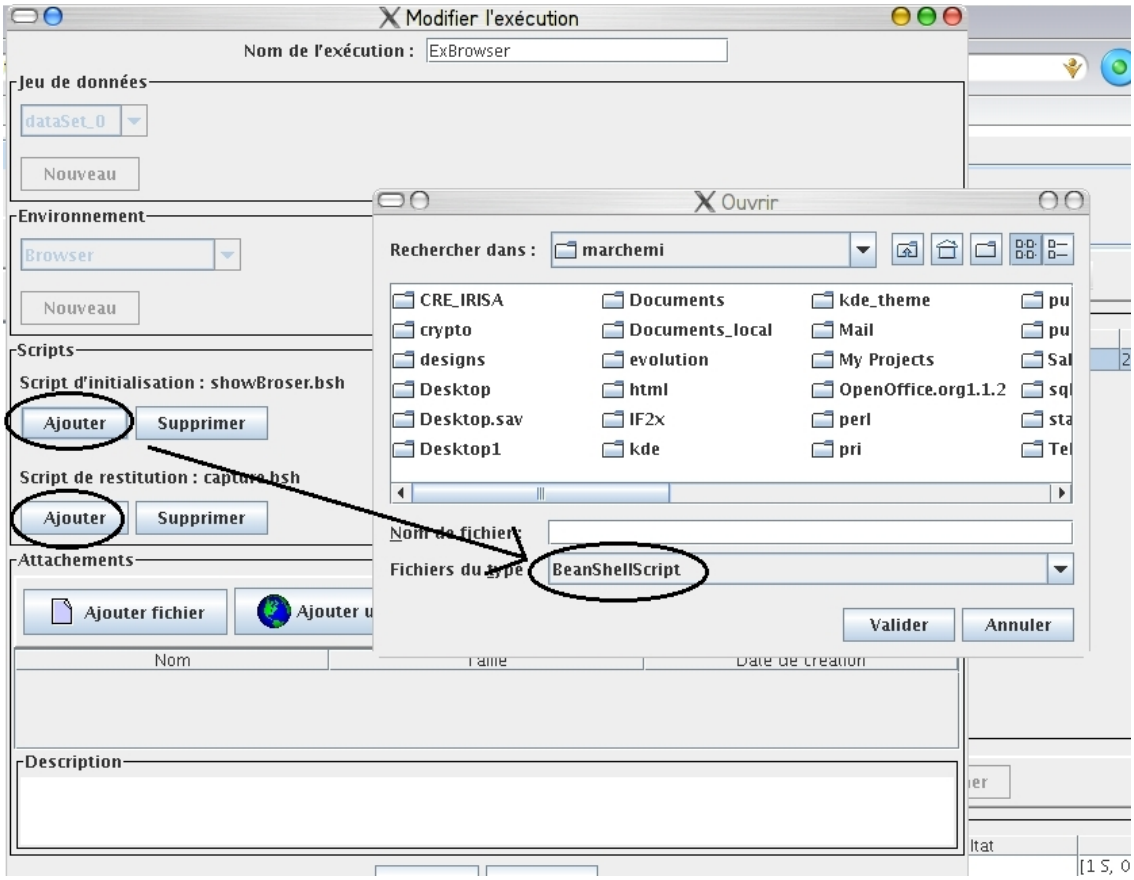


Figure 5.5: Adding scripts to an execution

Modifying execution and/or environments scripts depends on the plugin which is used. It can be done from the "script" menus of the executions and/or environments tabs (Figure 5.6).


Figure 5.6: Modifying an execution script (code)

## 5.4 Execution context

Depending upon the plugins used for describing test scripts, environment scripts and execution scripts, the execution context of the scripts can be unchanged during the complete campaign execution. In that case, the same references will be accessible. This is the case for the BeanShell and JUnit plugins.

In general, the test, execution and environment scripts have got references on the variables described in the following table (*=org.objectweb.salome_tmf.data):

| Name | Class | Description | Disponible |
|---|---|---|---|
| date | *Java.lang.Date* | Execution date | T, Env, Ex |
| time | *Java.lang.Time* | Execution time | T, Env, Ex |
| salome_projectName | *Java.lang.String* | Name of the current project T, Env, Ex | |
| salome_projectObject | **.Project* | Reference to the Salomé-TMF project object | T, Env, Ex |
| salome_debug | *boolean* | False when evaluating a script in the editor | T, Env, Ex |
| salome_ExecResultObject | **.ExecutionResult* | Reference to the current execution results | T, Ex |
| salome_ExecTestResultObject | **.ExecutionTestResult* | Reference to the current test execution result | T |
| salome_CampagneName | *Java.lang.String* | Name of the current campaign | T, Env, Ex |
| salome_CampagneObject | **.Campaign* | Reference to the current campaign | T, Env, Ex |
| salome_environmentName | *Java.lang.String* | Name of the current environment | T, Env, Ex |
| salome_environmentObject | **.Environment* | Reference to the current environment | T, Env, Ex |
| salome_ExecName | *Java.lang.String* | Name of the current execution | T, Env, Ex |
| salome_ExecObject | **.Execution* | Reference to the current execution | T, Env, Ex |
| salome_TestName | *Java.lang.String* | Name of the current test | T |
| salome_TestObject | **.Test* | Reference to the current test | T |
| salome_SuiteTestName | *Java.lang.String* | Name of the current test suite | T |
| salome_SuiteTestObject | **.TestList* | Reference to the current test suite | T |
| salome_FamilyName | *Java.lang.String* | Name of the current family | T |
| salome_FamilyObject | **.Family* | Reference to the current family | T |
| testLog | *Java.lang.String* | Test log that will be added as attachment to the execution | T |
| Verdict | *Java.lang.String* | Test verdict (pass, fail, or unconclusive) | T |
| salome_classloader | *Java.net.URLClassLoader* | BeanShell class loader | T, Env, Ex |

The column *Disponible* indicates whether those variables can be used in test, environment or execution scripts (T for tests, Env for environments, and Ex for executions).

The test, execution and environment scripts have also got access to the parameters valued during executions by the datasets, by using their name. Those values are of the type Java.lang.String.