**Robot Framework**

Nhóm 9

Giảng viên: TS. Trương Anh Hoàng

Môn học: Các vấn đề hiện đại của công nghệ phần mềm.

Mục lục

[1 Mở đầu 3](#_Toc405150120)

[1.1 Giới thiệu 3](#_Toc405150121)

[1.1.1 Tại sao nên dùng Robot Framework 3](#_Toc405150122)

[1.1.2 Kiến trúc bậc cao 3](#_Toc405150123)

[1.1.3 Các thông tin khác 4](#_Toc405150124)

[1.2 Bản quyền và giấy phép 5](#_Toc405150125)

[1.3 Cài đặt 5](#_Toc405150126)

[1.3.1 Giới thiệu 5](#_Toc405150127)

[1.3.2 Yêu cầu 5](#_Toc405150128)

[1.3.3 Cài đặt Robot Framework 5](#_Toc405150129)

[1.3.4 Gỡ cài đặt, nâng cấp 5](#_Toc405150130)

[1.3.5 Một số điểm khác 6](#_Toc405150131)

[1.3.6 Một số tài liệu liên quan 6](#_Toc405150132)

[2 Ca kiểm thử 7](#_Toc405150133)

[2.1 File và thư mục 7](#_Toc405150134)

[2.1.1 Hỗ trợ định dạng file. 7](#_Toc405150135)

[2.1.2 Các quy tắc cho dữ liệu kiểm thử 14](#_Toc405150136)

[2.2 Tạo ca kiểm thử 16](#_Toc405150137)

[2.2.1 Cú pháp 16](#_Toc405150138)

[2.2.2 Sử dụng từ khóa trong ca kiểm thử 17](#_Toc405150139)

[2.2.3 Ca kiểm thử thất bại 17](#_Toc405150140)

[2.3 Sử dụng thư viện test 17](#_Toc405150141)

[2.3.1 Thư viện test trong sử dụng 17](#_Toc405150142)

[2.3.2 Tùy chỉnh tên của thư viện 19](#_Toc405150143)

[2.3.3 Một số thư viện chuẩn 19](#_Toc405150144)

[2.3.4 Thư viện ngoài 19](#_Toc405150145)

[2.4 Biến 20](#_Toc405150146)

[2.4.1 Giới thiệu 20](#_Toc405150147)

[2.4.2 Các kiểu biến 20](#_Toc405150148)

[2.4.3 Cách tạo biến 21](#_Toc405150149)

[2.5 Từ khóa cá nhân (user keyword) 21](#_Toc405150150)

[2.5.1 Cú pháp 21](#_Toc405150151)

[2.5.2 Tham số của từ khóa cá nhân 22](#_Toc405150152)

[2.5.3 Giá trị trả về của từ khóa 23](#_Toc405150153)

[2.6 Các chức năng nâng cao 24](#_Toc405150154)

[2.6.1 Phạm vi từ khóa 24](#_Toc405150155)

[2.6.2 Thời gian chờ (timeouts) 24](#_Toc405150156)

[2.6.3 Vòng lặp FOR 26](#_Toc405150157)

[3 Thực thi ca kiểm thử. 29](#_Toc405150158)

[3.1 Sử dụng cơ bản. 29](#_Toc405150159)

[3.1.1 Bắt đầu thực thi bản kiểm thử. 29](#_Toc405150160)

[3.1.2 Sử dụng tuỳ chọn dòng lệnh 30](#_Toc405150161)

[3.1.3 Kết quả kiểm thử 31](#_Toc405150162)

[3.1.4 Đối số file 32](#_Toc405150163)

[3.1.5 Tạo kịch bản bắt đầu(Creating start-up scripts) 33](#_Toc405150164)

[3.1.6 Vấn đề gỡ lỗi 33](#_Toc405150165)

[3.2 Thực thi kiểm thử 33](#_Toc405150166)

[3.2.1 Dòng chảy thực thi 34](#_Toc405150167)

[3.2.2 Tiếp tục khi ca kiểm thất bại 35](#_Toc405150168)

[3.2.3 Dừng quá trình thực thi kiểm thử 35](#_Toc405150169)

[3.3 Kết quả đầu ra sau xử lý 35](#_Toc405150170)

[3.3.1 Sử dụng công cụ rebot 35](#_Toc405150171)

[3.3.2 Tạo các báo cáo và các log khác nhau 36](#_Toc405150172)

[3.3.3 Kết hợp các kết quả đầu ra 36](#_Toc405150173)

# Mở đầu

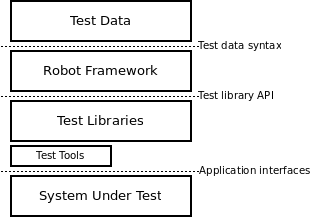
## Giới thiệu

### Tại sao nên dùng Robot Framework

* Là một framework test tự động phục vụ cho việc kiểm thử chấp nhận và kiểm thử có định hướng.
* Là phần mềm mã nguồn mở, được hỗ trợ bởi Nokia Networks.
* Cú pháp dễ sử dụng, phù hợp cho việc tạo ca kiểm thử một cách thống nhất
* Cung cấp khả năng tạo từ khóa tái sử dụng mức cao hơn
* Cung cấp các report, log dưới định dạng HTML
* Là một nền tảng ứng dụng độc lập
* Cung cấp thư viện API đơn giản để tạo các thư viện test dùng cho cả Python và Java
* Cung cấp giao diện dòng lệnh, các tập tin đầu ra dựa trên XML để tích hợp cho việc xây dựng cơ sở hạ tầng hiện có (hệ thống tích hợp liên tục)
* Hỗ trợ Selenium cho Web testing, Java GUI testing, các tiến trình chạy, Telnet, SSH…
* Hỗ trợ việc tạo hướng dữ liệu thử nghiệm (data- driven testing)
* Hỗ trợ các biến, thử nghiệm trong môi trường khác nhau
* Cung cấp các tag để phân loại
* Tích hợp dễ dàng với các source control
* Cung cấp các ca kiểm thử và testsuite
* Kiến trúc dạng modun, hỗ trợ tạo các test đối với cả các ứng dụng có nhiều giao diện khác nhau

### Kiến trúc bậc cao

* Robot Framework là một framework độc lập. Kiến trúc modun của nó được minh họa như hình sau:



Hình Bảng kiến trúc modun của Robot Framework

* Các dự liệu test ở dạng bảng đơn giản, dễ chỉnh sửa. Khi Robot Framework được bắt đầu, nó sẽ xử lý dữ liệu test, tiến hành chạy các ca kiểm thử và tạo các bản ghi, log. Phần chính của framework không quan tâm đến mục tiêu kiểm thử, phần tương tác với nó được xử lý bởi các thư viện kiểm tra. Thư viện có thể sử dụng giao diện ứng dụng trực tiếp hoặc dùng công cụ kiểm tra mức độ thấp

### Các thông tin khác

* Để biết thêm thông tin về Robot Framework cũng như hệ thống sinh thái của nó, truy cập <http://robotframework.org> hoặc Github
* Thảo luận các vấn đề về Robot Framework: <http://groups.google.com/group/robotframework-users>
* Các thông báo, tin tức về Robot Framework: http://groups.google.com/group/robotframework-announce
* Phát triển Robot Framework: http://groups.google.com/group/robotframework-devel
* Theo dõi sự phát triển của Robot Framework:

http://groups.google.com/group/robotframework-commit

## Bản quyền và giấy phép

## Cài đặt

### Giới thiệu

Chạy với Python, Jython (JVM) và IronPython (.NET)

### Yêu cầu

* Cài đặt Python:
  + Hỗ trợ Python 2.5, 2.6, 2.7. Python 3 sẽ được hỗ trợ trong tương lai

Robot Framework 2.0 và 2.1 hỗ trợ Python 2.3, 2.4

* Cài đặt Jthon:
  + Đòi hỏi Java 5 hoặc mới hơn. Phiên bản Jython 2.7 yêu cầu tối thiểu Java 7
  + Robot Framework 2.0 và 2.1 hỗ trợ Jython 2.2
* Cài đặt IronPython:
  + IronPython cho phép chạy RobotFramework trên nền .NET
  + Chỉ IronPython 2.7 được hỗ trợ

### Cài đặt Robot Framework

* Để cài đặt Robot Framework, có thể dùng các cách dưới đây:
  + Cài đặt bằng package manager (vd pip…)
  + Cài đặt từ source (vd Github)
  + Cài đặt từ bản Windows Installer
  + Cài đặt bằng file JAR
  + Cài đặt thủ công

### Gỡ cài đặt, nâng cấp

* Gỡ cài đặt
  + Dùng package manager (vd pip)
  + Dùng Windows installer
  + Thủ công
* Nâng cấp
  + Nên gỡ bỏ phiên bản cũ trước khi nâng cấp
  + Không cần cài đặt lại môi trường

### Một số điểm khác

Robot Framework sử dụng các runner script khác nhau để chạy ca kiểm thử và cho kết quả đầu ra. Các trình thông dịch khác nhau sử dụng các script khác nhau

Vd:

| **Interpreter** | **Test execution** | **Post-processing** |
| --- | --- | --- |
| Python | pybot | rebot |
| Jython | jybot | jyrebot |
| IronPython | ipybot | ipyrebot |

Hình Sử dụng các script khác nhau

Ngoài ra ta còn có thể chạy bằng cách sử dụng lệnh Python -m

| **Entry point** | **Run as module** | **Run as script** |
| --- | --- | --- |
| Test execution | python -m robot.run | python path/robot/run.py |
| Post-processing | python -m robot.rebot | python path/robot/rebot.py |

Hình Thực hiện với tham số -m

### Một số tài liệu liên quan

Một số project demo:

* [Quick Start Guide](http://code.google.com/p/robotframework/wiki/QuickStartGuide) <http://code.google.com/p/robotframework/wiki/QuickStartGuide>
* Robot Framework Demo

https://bitbucket.org/robotframework/robotdemo/wiki/Home

* Web testing Demo

https://bitbucket.org/robotframework/webdemo/wiki/Home

* SwingLibrary Demo

https://github.com/robotframework/SwingLibrary/wiki/SwingLibrary-Demo

* ATDD with Robot Framework

<https://code.google.com/p/atdd-with-robot-framework>

# Ca kiểm thử

Phần này bao gồm cú pháp tổng thể của dữ liệu kiểm thử trong Robot Framework. Các phần sau sẽ giải thích làm thế nào để tạo ra 1 test case, test suites,…

## File và thư mục

Cấu trúc phân cấp để sắp xếp các trường hợp ca kiểm thử:

* Ca kiểm thử được tạo trong các file.
* Ca kiểm thử file tự động tạo ra test suites chứa tất cả các ca kiểm thử trong file này.
* Thư mục test suite được tạo chứa các ca kiểm thử, là 1 test suite con của test suite chính.
* Các thư mục test suite có thể chứa các thư mục test suite khác.
* Thư mục test suite có thể chứa file khởi tạo.

Thêm vào đó là:

* Test libraries chứa các từ khoá ở mức thấp.
* Resource file cùng với biến và các từ khoá của người dùng.
* Variable file sẽ cung cấp cách tạo ra các biến linh hoạt hơn là Resource file.

### Hỗ trợ định dạng file.

Robot Framwork kiểm tra dữ liệu được định dạng trong bảng, bằng cách sử dụng hypertext markup language (HTML), tab-sparated values (TVS), plain text, hoặc reStructuredText (reST). Chi tiết về các định dạng sẽ được mô tả trong phần dưới.

Robot Framwork chọn phân tích các dữ liệu kiểm thử trên các file với phần mở rộng là .html, .xhtml cho HTML, .tsv cho TSV, .txt và đặc biết alf .robot cho những văn bản đơn giản, và .rst và .rest cho reStructuredText.

#### Định dạng HTML.

File HTML hỗ trợ định dạng và các văn bản đơn giản trên bảng. Điều này làm cho nó có thể thêm các thông tin bổ xung vào các file ca kiểm thử và tạo ra các file.

Trong HTML file, dữ liệu kiểm thử được định nghĩa trong các bảng riêng biệt(xem ví dụ dưới). Tất cả mọi thứ bên ngoài đều bị bỏ qua.

| **Setting** | **Value** | **Value** | **Value** |
| --- | --- | --- | --- |
| Library | OperatingSystem |  |  |

| **Variable** | **Value** | **Value** | **Value** |
| --- | --- | --- | --- |
| ${MESSAGE} | Hello, world! |  |  |

| **Test Case** | **Action** | **Argument** | **Argument** |
| --- | --- | --- | --- |
| My Test | [Documentation] | Example test |  |
|  | Log | ${MESSAGE} |  |
|  | My Keyword | /tmp |  |
| Another Test | Should Be Equal | ${MESSAGE} | Hello, world! |

| **Keyword** | **Action** | **Argument** | **Argument** |
| --- | --- | --- | --- |
| My Keyword | [Arguments] | ${path} |  |
|  | Directory Should Exist | ${path} |  |

Hình Sử dụng với định dạng html

##### Chỉnh sửa dữ liệu kiểm thử.

Bạn có thể chỉnh sửa lại dữ liệu kiểm thử trong HTML bằng bất cứ trình soạn thảo nào mà bạn thích, nếu dùng 1 trình soạn thảo có đồ hoạ sẽ tốt hơn.

##### Mã hoá và tổ chức các tài liệu.

Các tài liệu cho HTML(ví dụ &auml;) được hỗ trợ. Ngoài ra, bất kỳ mã hoá nào có thể được sử dụng nếu nó được chỉ định trong các file chứa dữ liệu kiểm thử . Tập tin HTML bình thường phải sử dụng phần tử META như ví dụ dưới:

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

Đối với XML thì như ví dụ này:

<?xml version="1.0" encoding="Big5"?>

Nếu không có mã hoá nào được chỉ định thì Robot Framwork sẽ sử dụng theo mặc định.

#### Định dạng TSV.

Các TSV file có thể được chỉnh sửa trong các chương trình bảng tính ví dụ Microsoft Excel. Chúng cũng dễ dàng để chỉnh sửa bằng cách sử dụng các trình soạn thảo văn bản thông thường.

Định dạng TSV có thể được sử dụng trong Robot Framework’s cũng giống như định dạng HTML. Trong 1 file TSV, tất cả dữ liệu đều ở trong 1 bảng lớn. Bảng dữ liệu kiểm thử được xác định bằng 1 hoặc nhiều dấu (\*), theo sau là tên bảng và 1 dấu (\*) đóng tuỳ chọn. Tất cả mọi thứ trước bảng đầu tiên đều được bỏ qua tương tự như dữ liệu bên ngoài trong định dạng html.

|  |  |  |  |
| --- | --- | --- | --- |
| \*Setting\* | \*Value\* | \*Value\* | \*Value\* |
| Library | OperatingSystem |  |  |
|  |  |  |  |
|  |  |  |  |
| \*Variable\* | \*Value\* | \*Value\* | \*Value\* |
| ${MESSAGE} | Hello, world! |  |  |
|  |  |  |  |
|  |  |  |  |
| \*Test Case\* | \*Action\* | \*Argument\* | \*Argument\* |
| My Test | [Documentation] | Example test |  |
|  | Log | ${MESSAGE} |  |
|  | My Keyword | /tmp |  |
|  |  |  |  |
| Another Test | Should Be Equal | ${MESSAGE} | Hello, world! |
|  |  |  |  |
|  |  |  |  |
| \*Keyword\* | \*Action\* | \*Argument\* | \*Argument\* |
| My Keyword | [Arguments] | ${path} |  |
|  | Directory Should Exist | ${path} |  |

Hình Sử dụng định dạng TSV

##### Chỉnh sửa dữ liệu kiểm thử.

* Bạn có thể tạo và chỉnh sửa TSV file trong bất kỳ 1 chương trình bảng tính nào và lưu với đuôi mở rộng là .tsv.
* Các TSV file tương đối dễ dàng để chỉnh sửa với bất kỳ trình soạn thảo văn bản nào.

##### Mã hoá

TSV file luôn luôn sử dụng mã hoá UTF-8.

##### Định dạng văn bản đơn giản

* Định dạng văn bản đơn giản rất dễ dàng để chỉnh sửa bằng cách sử dụng bất kỳ trình soạn thảo văn bản nào.
* Các bảng dữ liệu kiểm thử phải có 1 hoặc nhiều dấu hoa thị trước tên tương tự như trong các định dạng TSV.

##### Tách các định dạng

Số khoảng trắng được sử dụng có thể là khác nhau nhưng miễn là có ít nhất là 2 khoảng trắng.

\*\*\* Settings \*\*\*

Library OperatingSystem

\*\*\* Variables \*\*\*

${MESSAGE} Hello, world!

\*\*\* Test Cases \*\*\*

My Test

[Documentation] Example test

Log ${MESSAGE}

My Keyword /tmp

Another Test

Should Be Equal ${MESSAGE} Hello, world!

\*\*\* Keywords \*\*\*

My Keyword

[Arguments] ${path}

Directory Should Exist ${path}

##### Kí hiệu | (pipe) và khoảng trắng phân tách trong bản định dạng.

Nếu sử dụng nguyên khoảng trắng để phân cách thì rất khó nhìn và dễ nhầm lẫn. Đây là vấn đề lớn khi từ khoá có nhiều đối số. Trong trường hợp này nên sử dụng “|” sẽ tốt hơn. Ví dụ:

| \*Setting\* | \*Value\* |

| Library | OperatingSystem |

| \*Variable\* | \*Value\* |

| ${MESSAGE} | Hello, world! |

| \*Test Case\* | \*Action\* | \*Argument\* |

| My Test | [Documentation] | Example test |

| | Log | ${MESSAGE} |

| | My Keyword | /tmp |

| Another Test | Should Be Equal | ${MESSAGE} | Hello, world!

| \*Keyword\* |

| My Keyword | [Arguments] | ${path}

| | Directory Should Exist | ${path}

##### Chỉnh sửa và mã hoá

* Một trong những lợi ích lớn nhất của các định dạng văn bản thuần trong HTML và TSV là chỉnh sửa nó bằng các trình soạn thảo văn bản thông thường. Nhiều IDE như: Eclipse, Emacs, Vim và TextMate cũng đã hỗ trợ soạn thảo cho Robot Framework.
* Tương tự như dữ liệu kiểm thử đối với TSV, file văn bản đơn giản luôn sử dụng UTF-8 để mã hoá.

##### Tiện ích mở rộng được công nhận

Bắt đầu từ phiên bản Robot Framework 2.7.6, các tập tin dữ liệu kiểm thử đơn giản có thể sử dụng phần mở rộng là .robot ngoài phần mở rộng là .txt.

#### Định dạng reStructutedText

* reStrucredText(reST) là một cú pháp đánh dấu văn bản đơn giản dễ đọc thường được sử dụng cho các dự án của Python.
* Sử dụng reST với Robot Framework cho phép bạn kết hợn đa dạng về định dạng tài liệu dữ liệu kiểm thử trong 1 văn bản và có thể dễ dàng dùng các trình soạn thảo văn bản đơn giản.
* Khi sử dụng reST file với Robot Framework có 2 cách để xác định dữ liệu kiểm thử. Bạn có thể sử dụng các khối mã(code block) và xác định các ca kiểm thử trong đó sử dụng định dạng văn bản thuần tuý hoặc bạn có thể sử dụng định dạng theo HTML.

##### Sử dụng khối mã

Tài liệu reStructuredText có thể chứa mã trong các khối. Các tài liệu này được biên dịch thành HTML. Các khối mã được bắt đầu bằng cách sử dụng chỉ thị mã, tên của ngôn ngữ lập trình trong khối được đưa ra như là đối số để chỉ thị. Ví dụ sau là có chứa mã Python và Robot Framework:

.. code:: python

def example\_keyword():

print 'Hello, world!'

.. code:: robotframework

\*\*\* Test Cases \*\*\*

Example Test

Example Keyword

Khi Robot Framework phân tích reST thì sẽ tìm những từ khoá code, code-block, sourcecode để nhận biết khối mã.

##### Sử dụng bảng

* Nếu 1 tài liệu restructuredtext không chứa các khối mã với dữ liệu của Robot Framework thì nó sẽ chứa trong các bảng dữ liệu như trong định dạng HTML.
* Robot Framework xác định bảng dữ liệu kiểm thử dựa trên văn bản trong ô đầu tiên và tất cả các nội dung bên ngoài sẽ bị loại bỏ. Ví dụ dưới đây hiển thị cả 2 loại bảng là đơn giản và bảng lưới:

Example

-------

This text is outside tables and thus ignored.

============ ================ ======= =======

Setting Value Value Value

============ ================ ======= =======

Library OperatingSystem

============ ================ ======= =======

============ ================ ======= =======

Variable Value Value Value

============ ================ ======= =======

${MESSAGE} Hello, world!

============ ================ ======= =======

============= ================== ============

Test Case Action Argument Argument

============= ================== ============

My Test [Documentation] Example test

\ Log ${MESSAGE}

\ My Keyword /tmp

\

Another Test Should Be Equal ${MESSAGE} Hello, world!

============= ================== ============

Also this text is outside tables and ignored. Above tables are created

using the simple table syntax and the table below uses the grid table

approach.

+-------------+------------------------+------------+------------+

| Keyword | Action | Argument | Argument |

+-------------+------------------------+------------+------------+

| My Keyword | [Arguments] | ${path} | |

+-------------+------------------------+------------+------------+

| | Directory Should Exist | ${path} | |

+-------------+------------------------+------------+------------+

##### Chỉnh sửa và mã hoá

Dữ liệu kiểm thử trong reStructuredText file cũng có thể chỉnh sửa bằng các trình soạn thảo đơn giản và sử dụng mã hoá UTF-8.

##### Lỗi cú pháp trong file mã nguồn reST

Nếu 1 tài liệu reStructuredText có cú pháp không chính xác(ví dụ 1 bảng bị thay đổi), quá trình phân tích nó sẽ không thành công và Robot Framework sẽ hiển thị lỗi lên màn hình.

##### Bảng dữ liệu kiểm thử

Dữ liệu kiểm thử có cấu trúc trong 4 loại bảng liệt kê dưới đây. Các bảng dữ liệu kiểm thử được xác định bởi ô đầu tiên trong bảng. Cột cuối cùng trong bảng dưới chứa bí danh thay thế cho tên bảng:

| **Tên bảng** | **Tác dụng** | **Bí danh** |
| --- | --- | --- |
| Bảng cài đặt | 1) Import thư viện, các file tài nguyên và file chứa các biến.  2) Xác định siêu dữ liệu cho test suites và ca kiểm thử. | Setting, Settings, Metadata |
| Bảng biến | Xác định các biến có thể được sử dụng ở những nơi khác trong dữ liệu kiểm thử | Variable, Variables |
| Bảng ca kiểm thử | Sinh ca kiểm thử từ các từ khoá. | Test Case, Test Cases |
| Bảng từ khoá | Tạo các từ khoá cho người sử dụng từ các từ khoá cấp thấp hơn. | Keyword, Keywords, User Keyword, User Keywords |

Hình Các bảng dữ liệu trong Robot Framework

### Các quy tắc cho dữ liệu kiểm thử

#### Dữ liệu bỏ qua

Khi Robot Framework phân tích dữ liệu kiểm thử, nó sẽ bỏ qua:

* Tất cả các bảng mà không bắt đầu với 1 tên bảng được công nhận trong ô đầu tiên.
* Tất cả mọi thứ khác trên dòng đầu tiên của 1 bảng ngoài các ô đầu tiên.
* Tất cả các dữ liệu trước bảng đầu tiên.
* Tất cả các hàng trống (các hàng chỉ để dễ đọc hơn).
* Tất cả các ô trống ở cuối hàng.
* Tất cả dấu gạch chéo ngươc đơn(/) khi không được sử dụng để thoát khỏi chương trình.
* Tất cả định dạng trong HTML/reST dữ liệu kiểm thử.

#### Xử lý khoảng trắng

Robot Framework xử lý các khoảng trắng như trong cách xử lý mã nguồn HTML.

#### Thoát

Kí từ để thoát trong Robot Framework dữ liệu kiểm thử là dấu gạch chéo ngược(\) và ngoài ra được chứa trong biến ${EMPTY} và ${SPACE}

#### Thoát với kí tự đặc biệt

Dấu gạch chéo ngược được sử dụng để thoát cùng với kí tự đặc biệt.

| **Kí tự** | **Ý nghĩa** | **Ví dụ** |
| --- | --- | --- |
| \$ | Kí hiệu đô la không bao giờ bắt đầu 1 biến vô hướng. | \${notvar} |
| \@ | Kí hiệu @ không bao giờ bắt đầu 1 biến danh sách. | \@{notvar} |
| \% | Dấu phần trăm, không bao giờ bắt đầu một biến môi trường | \%{notvar} |
| \# | Dấu thăng không bao giờ bắt đầu lời nhận xét | \# not comment |
| \= | Dấu bằng không bao giờ là 1 phần trong cú pháp tham số. | not\=named |
| \| | Kí tự pipe không phải là 1 dấu phân cách theo định dạng ống tách. | | Run | ps \| grep xxx | |
| \\ | Kí tự dấu gạch chéo ngược không bao giờ thoát ra. | c:\\temp, \\${var} |

Hình Các kí tự đặc biệt

#### Hình thành chuỗi exscape

Ngoài ra còn có 1 số các ký tự thoát khác

| **Sequence** | **Ý nghĩa** | **Ví dụ** |
| --- | --- | --- |
| \n | Kí tự xuống dòng. | first line\n2nd line |
| \r | Kí tự về đầu dòng. | text\rmore text |
| \t | Kí tự tab. | text\tmore text |
| \xhh | Kí tự với giá trị hex hh. | null byte: \x00, ä: \xE4 |
| \uhhhh | Kí tự với giá trị hex hhhh. | snowman: \u2603 |
| \Uhhhhhhhh | Kí tự với giá trị hex hhhhhhhh. | love hotel: \U0001f3e9 |

Hình Các kí tự đặc biệt khác

#### Ngăn bỏ qua các ô trống

Không được để các ô được trống, có thể thây thế bằng dấu (\) hoặc ${EMPTY}

| **Test Case** | **Hành động** | **Đối số** | **Đối số** | **Đối số** |
| --- | --- | --- | --- | --- |
| Sử dụng dấu gạch chéo ngược(\) | Không làm gì | Đối số đầu | \ |  |
| Sử dụng ${EMPTY} | Không làm gì | Đối số đầu | ${EMPTY} |  |
| Để trống | Không làm gì |  | Đối số thứ 2 | # lỗi |
| Vòng lặp | :FOR | ${var} | IN | @{VALUES} |
|  |  | Log | ${var} | # lỗi |

Hình Không bỏ qua các ô trống

## Tạo ca kiểm thử

### Cú pháp

* Các ca kiểm thử được tạo ghi tại bảng ca kiểm thử (test case tables)
* Tên ca kiểm thử được ghi tại cột đầu tiên, cột thứ 2 chứa từ khóa, các cột tiếp theo chứa tham số của từ khóa

| **Test Case** | **Action** | **Argument** | **Argument** |
| --- | --- | --- | --- |
| Valid Login | Open Login Page |  |  |
|  | Input Name | demo |  |
|  | Input Password | mode |  |
|  | Submit Credentials |  |  |
|  | Welcome Page Should Be Open |  |  |

Hình Ví dụ ca kiểm thử

### Sử dụng từ khóa trong ca kiểm thử

* Từ khóa với tham số mặc định: Một số từ khóa có tham số với giá trị mặc định. Ca kiểm thử có thể gọi các phiên bản khác nhau của từ khóa đó

VD: Từ khóa Create File có tham số dạng path, content=, encoding=UTF-8. Ta có ca kiểm thử sau:

| **Test Case** | **Action** | **Argument** | **Argument** | **Argument** |
| --- | --- | --- | --- | --- |
| Example | Create File | ${TEMPDIR}/empty.txt |  |  |
|  | Create File | ${TEMPDIR}/utf-8.txt | Hyvä esimerkki |  |
|  | Create File | ${TEMPDIR}/iso-8859-1.txt | Hyvä esimerkki | ISO-8859-1 |

Hình Sử dụng từ khoá trong ca kiểm thử

* Sử dụng từ khóa với tên tham số: Khi 1 từ khóa có nhiều tham số mặc định, ta có thể dễ dàng thay đổi giá trị 1 tham số bất kì bằng việc thêm tên tham số và dấu “=” phía trước giá trị mong muốn

| **Test Case** | **Action** | **Argument** | **Argument** | **Argument** |
| --- | --- | --- | --- | --- |
| Example | Open connection | 10.0.0.42 | port=${PORT} | alias=example |
|  | List files | options=-lh |  |  |
|  | List files | path=/tmp | options=-l |  |

Hình Sử dụng từ khoá với tên tham số

### Ca kiểm thử thất bại

* Ca kiểm thử được gọi là thất bại khi có bất cứ 1 từ khóa nào trong ca kiểm thử không thực hiện được
* Khi ca kiểm thử thất bại, thông báo được đưa ra là thông báo thất bại của từ khóa không thực hiện được

## Sử dụng thư viện test

Thư viện test bao gồm các từ khóa mức thấp – gọi là từ khóa thư viện, tương tác với hệ thống kiểm tra. Tất cả các ca kiểm thử đều sử dụng các từ khóa từ một số thư viện nào đó, thông qua các từ khóa mức cao

### Thư viện test trong sử dụng

#### Thiết lập thư viện để sử dụng

* Thư viện test được import sử dụng các thiết lập cài đặt trong Bảng thiết lập
* Tên thư viện là tên của modun hoặc class đã implement thư viện
* Trong trường hợp thư viện cần tham số, chúng sẽ được liệt kê ở các cột phía sau tên thư viện

| **Setting** | **Value** | **Value** | **Value** |
| --- | --- | --- | --- |
| Library | OperatingSystem |  |  |
| Library | com.company.TestLib |  |  |
| Library | MyLibrary | arg1 | arg2 |
| Library | ${LIBRARY} |  |  |

Hình Import thư viện test

* Có thể import thư viện test vào các file ca kiểm thử, resource file… Khi này, các từ khóa trong thư viện có thể sử dụng được trong file

#### Sử dụng từ khóa Import Library

* Một cách khác để sử dụng thư viện đó chính là sử dụng từ khóa Import Library

| **Test Case** | **Action** | **Argument** | **Argument** | **Argument** |
| --- | --- | --- | --- | --- |
| Example | Do Something |  |  |  |
|  | Import Library | MyLibrary | arg1 | arg2 |
|  | KW From Mylibrary |  |  |  |

Hình Sử dụng từ khoá import thư viện

#### Đường dẫn đến thư viện cần sử dụng

* Cần có phần mở rộng (.py đối với Python hoặc .class đối với java)

| **Setting** | **Value** | **Value** | **Value** |
| --- | --- | --- | --- |
| Library | PythonLib.py |  |  |
| Library | /absolute/path/JavaLib.java |  |  |
| Library | relative/path/PythonDirLib/ | possible | arguments |
| Library | ${RESOURCES}/Example.class |  |  |

Hình Thiết lập đường dẫn đến thư viện

### Tùy chỉnh tên của thư viện

* Tên thư viện được ghi trong test log, phía trước từ khóa. Nếu từ khóa có cùng tên thì chúng phải được gọi với tên của thư viện. Tên thư viện thường là tên modun hoặc tên class implement nó
* Các lí do cần thay đổi tên thư viện:
* Tên thư viện dài, bất tiện.
* Cần sử dụng các biến để import thư viện khác trong các môi trường khác nhau với cùng tên.
* Tên thư viện dễ gây hiểu lầm

| **Setting** | **Value** | **Value** | **Value** |
| --- | --- | --- | --- |
| Library | com.company.TestLib | WITH NAME | TestLib |
| Library | ${LIBRARY} | WITH NAME | MyName |

Hình Tuỳ chỉnh thư viện cần import

| **Setting** | **Value** | **Value** | **Value** | **Value** | **Value** |
| --- | --- | --- | --- | --- | --- |
| Library | SomeLibrary | localhost | 1234 | WITH NAME | LocalLib |
| Library | SomeLibrary | server.domain | 8080 | WITH NAME | RemoteLib |

| **Test Case** | **Action** | **Argument** | **Argument** |
| --- | --- | --- | --- |
| My Test | LocalLib.Some Keyword | some arg | second arg |
|  | RemoteLib.Some Keyword | another arg | whatever |
|  | LocalLib.Another Keyword |  |  |

Hình Tuỳ chỉnh thư viện với tham số

### Một số thư viện chuẩn

* Được sử dụng tự động.
* Một số thư viện phổ biến đó là BuiltIn, Collections, DateTime, Dialogs, OperatingSystem, Process…

### Thư viện ngoài

* Cung câp bởi cộng đồng mã nguồn mở, vd Selenium2Library, SwingLibrary.
* Cơ chế cài đặt khác nhau.

## Biến

### Giới thiệu

* Biến thường được sử dụng trong các bảng kiểm thử và bảng từ khóa.
* Các từ khóa thường không thể định nghĩa biến.
* Robot Framework có 2 kiểu biến là là biến vô hướng (scalar variables) và biến danh sách (list variables).
* Biến vô hướng có dạng ${SCALAR}
* Biến danh sách có dạng @{LIST}
* Tên biến không phân biệt chữ hoa, chữ thường.

### Các kiểu biến

#### Biến vô hướng

* Các kiểu có thể gán cho biến vô hướng có thể là xâu kí tự, đối tượng, danh sách

VD: Hello gán cho ${GREET}, world gán cho ${NAME}. Các ca kiểm thử sau là tương đương:

| **Test Case** | **Action** | **Argument** | **Argument** |
| --- | --- | --- | --- |
| Strings | Log | Hello |  |
|  | Log | Hello, world!! |  |
| Variables | Log | ${GREET} |  |
|  | Log | ${GREET}, ${NAME}!! |  |

Hình Biến vô hướng

#### Biến danh sách

* Biến danh sách là biến trong nó có thể chứa nhiều giá trị.
* Biến danh sách thường chứa các xâu kí tự, ngoài ra nó còn có thể chứa các kiểu dữ liệu khác.

**VD:** ['robot','secret'] gán cho @{USER}. 2 ca kiểm thử sau là tương đương:

| **Test Case** | **Action** | **User Name** | **Password** |
| --- | --- | --- | --- |
| Strings | Login | robot | secret |
| List Variable | Login | @{USER} |  |

Hình Biến danh sách

* Để truy cập giá trị thứ I trong biến danh sách, sử dụng cú pháp @{LIST}[i]

### Cách tạo biến

Các biến được định nghĩa trong bảng biến (Variable tables)

#### Biến vô hướng

* Tên biến đặt trong cột Variable, giá trị đặt trong cột Value đầu tiên trong bảng chứa biến. Ví dụ:

| **Variable** | **Value** | **Value** |
| --- | --- | --- |
| ${NAME} | Robot Framework |  |
| ${VERSION} | 2.0 |  |
| ${ROBOT} | ${NAME} ${VERSION} |  |

Hình Cách tạo biến vô hướng

#### Biến danh sách

- Tên biến đặt trong cột Variable, giá trị đặt trong các cột Value trong bảng chứa biến. Các giá trị của biến có thể đặt trong nhiều dòng.

| **Variable** | **Value** | **Value** | **Value** |
| --- | --- | --- | --- |
| @{NAMES} | Matti | Teppo |  |
| @{NAMES2} | @{NAMES} | Seppo |  |
| @{NOTHING} |  |  |  |
| @{MANY} | one | two | three |
| ... | four | five | six |
| ... | seven |  |  |

Hình Cách tạo biến danh sách

## Từ khóa cá nhân (user keyword)

Ngoài từ khóa sử dụng trong các thư viện, ta có thể thêm mới các từ khóa để sử dụng trong các ca kiểm thử

### Cú pháp

* Các từ khóa được định nghĩa trong bảng từ khóa (keyword tables)
* Từ khóa cá nhân được ghi ở cột 1, từ khóa thư viện được sử dụng ghi ở cột 2, các đối số sử dụng ghi ở các cột tiếp theo

| **Keyword** | **Action** | **Argument** | **Argument** |
| --- | --- | --- | --- |
| Open Login Page | Open Browser | http://host/login.html |  |
|  | Title Should Be | Login Page |  |
|  |  |  |  |
| Title Should Start With | [Arguments] | ${expected} |  |
|  | ${title} = | Get Title |  |
|  | Should Start With | ${title} | ${expected} |

Hình Từ khoá cá nhân

### Tham số của từ khóa cá nhân

* Để định nghĩa tham số cho từ khóa: tên từ khóa ghi tại cột 1, [Arguments] được ghi tại cột 2, các tham số ghi ở các cột tiếp theo

| **Keyword** | **Action** | **Argument** | **Argument** | **Argument** |
| --- | --- | --- | --- | --- |
| One Argument | [Arguments] | ${arg\_name} |  |  |
|  |  |  |  |  |
| Three Arguments | [Arguments] | ${arg1} | ${arg2} | ${arg3} |

Hình Tham số từ khoá cá nhân

* Tham số mặc định: Để chỉ định giá trị mặc định của các tham số, thêm kí tự “=” và giá trị muốn đặt phía sau tham số tương ứng

| **Keyword** | **Action** | **Argument** | **Argument** |
| --- | --- | --- | --- |
| One Argument With Default Value | [Arguments] | ${arg}=default value |  |
|  | [Documentation] | This keyword takes | 0-1 arguments |
|  |  |  |  |
| Two Arguments With Defaults | [Arguments] | ${arg1}=default 1 | ${arg2}=${VARIABLE} |
|  | [Documentation] | This keyword takes | 0-2 arguments |
|  |  |  |  |
| One Required And One With Default | [Arguments] | ${required} | ${optional}=default |
|  | [Documentation] | This keyword takes | 1-2 arguments |

Hình Từ khoá cá nhân với tham số mặc định

* Danh sách tham số: Ta có thể dùng 1 danh sách các biên làm tham số của từ khóa. Để duyệt qua danh sách tham số, ta sử dụng vòng lặp FOR

| **Keyword** | **Action** | **Argument** | **Argument** | **Argument** |
| --- | --- | --- | --- | --- |
| Any Number Of Arguments | [Arguments] | @{varargs} |  |  |
|  |  |  |  |  |
| One Or More Arguments | [Arguments] | ${required} | @{rest} |  |
|  |  |  |  |  |
| Required, Default, Varargs | [Arguments] | ${req} | ${opt}=42 | @{others} |
|  | Log | Required: ${req} |  |  |
|  | Log | Optional: ${opt} |  |  |
|  | Log | Others: |  |  |
|  | : FOR | ${item} | IN | @{others} |
|  |  | Log | ${item} |  |

Hình Từ khoá cá nhân với danh sách tham số

### Giá trị trả về của từ khóa

* Để trả lại giá trị, [Return] ghi vào cột 2, các giá trị trả về ghi vào các cột tiếp theo

| **Keyword** | **Action** | **Argument** | **Argument** | **Argument** |
| --- | --- | --- | --- | --- |
| Return One Value | [Arguments] | ${arg} |  |  |
|  | Do Something | ${arg} |  |  |
|  | ${value} = | Get Some Value |  |  |
|  | [Return] | ${value} |  |  |
|  |  |  |  |  |
| Return Three Values | [Return] | foo | bar | zap |

Hình Từ khoá trả về giá trị

## Các chức năng nâng cao

### Phạm vi từ khóa

Khi 1 từ khóa được định nghĩa nhiều lần tại nhiều vị trí, giá trị của từ khóa được xác định dựa trên độ ưu tiên như sau:

* + Từ khóa được tạo trong file có quyền ưu tiên cao nhất
  + Từ khóa được thêm vào trực tiếp hoặc gián tiếp thông qua các file bổ xung có độ ưu tiên thứ 2
  + Từ khóa trong các thư viện ngoài có độ ưu tiên thứ 3. Nếu từ khóa được sử dụng có cùng tên trong thư viện cơ bản, 1 thông báo sẽ được hiển thị
  + Từ khóa trong các thư viện cơ bản có độ ưu tiên thấp nhất.

### Thời gian chờ (timeouts)

* Thời gian chờ được sử dụng để ngưng ca kiểm thử, từ khóa khi việc thực hiện mất quá nhiều thời gian.
* Thời gian chờ của ca kiểm thử
  + Thời gian chờ của ca kiểm thử có thể được thiết lập qua giá trị Test Timeout trong bảng cài đặt (setting table) hoặc qua [Timeout] trong bảng kiểm thử. Giá trị Test Timeout được thiết đặt cho mọi ca kiểm thử, trong khi [Timeout] chỉ có tác dụng với ca kiểm thử được ghi.
  + Để trống [Timeout] (hoặc ghi NONE) tức là ca kiểm thử đó không giới hạn thời gian chạy
* Thông báo mặc định khi hết thời gian chờ là Test timeout <time> exceeded. Có thể thay thế thông báo bằng cách ghi tại các cột tiếp sau giá trị [Timeout]

| **Setting** | **Value** | **Value** | **Value** |
| --- | --- | --- | --- |
| Test Timeout | 2 minutes |  |  |

| **Test Case** | **Action** | **Argument** | **Argument** | **Argument** |
| --- | --- | --- | --- | --- |
| Default Timeout | [Documentation] | Timeout from the Setting table is used |  |  |
|  | Some Keyword | argument |  |  |
|  |  |  |  |  |
| Override | [Documentation] | Override default, use 10 seconds timeout |  |  |
|  | [Timeout] | 10 |  |  |
|  | Some Keyword | argument |  |  |
|  |  |  |  |  |
| Custom Message | [Documentation] | Override default and use custom message |  |  |
|  | [Timeout] | 1min 10s | This is my custom error. | It continues here. |
|  | Some Keyword | argument |  |  |
|  |  |  |  |  |
| Variables | [Documentation] | It is possible to use variables too |  |  |
|  | [Timeout] | ${TIMEOUT} |  |  |
|  | Some Keyword | argument |  |  |
|  |  |  |  |  |
| No Timeout | [Documentation] | Empty timeout means no timeout even when | Test Timeout has been used |  |
|  | [Timeout] |  |  |  |
|  | Some Keyword | argument |  |  |
|  |  |  |  |  |
| No Timeout 2 | [Documentation] | Empty timeout using NONE, works with | 2.5.6 |  |
|  | [Timeout] | NONE |  |  |
|  | Some Keyword | argument |  |  |

Hình Thời gian chờ với ca kiểm thử

* Thời gian chờ của từ khóa cá nhân:
* Sử dụng [Timeout] trong bảng từ khóa
* Thông báo mặc định khi hết thời gian chờ là Test timeout <time> exceeded. Có thể thay thế thông báo bằng cách ghi tại các cột tiếp sau giá trị [Timeout]

| **Keyword** | **Action** | **Argument** | **Argument** |
| --- | --- | --- | --- |
| Timed Keyword | [Documentation] | Set only the timeout value | and not the custom message. |
|  | [Timeout] | 1 minute 42 seconds |  |
|  | Do Something |  |  |
|  |  |  |  |
| Timed-out Wrapper | [Arguments] | @{args} |  |
|  | [Documentation] | This keyword is a wrapper | that adds a timeout to another keyword. |
|  | [Timeout] | 2 minutes | Original Keyword didn't finish in 2 minutes |
|  | Original Keyword | @{args} |  |

Hình Thời gian chờ với từ khoá

### Vòng lặp FOR

Vòng lặp FOR có dạng: FOR item IN sequence

* Vòng lặp thường
* Cú pháp bắt đầu bằng :FOR tại cột thứ 2, cột tiếp theo là biến chạy, cột thứ ba là IN, các cột tiếp theo là các giá trị cần duyệt

| **Test Case** | **Action** | **Argument** | **Argument** | **Argument** | **Arguments** |
| --- | --- | --- | --- | --- | --- |
| Example 1 | :FOR | ${animal} | IN | cat | dog |
|  |  | Log | ${animal} |  |  |
|  |  | Log | 2nd keyword |  |  |
|  | Log | Outside loop |  |  |  |
|  |  |  |  |  |  |
| Example 2 | :FOR | ${var} | IN | one | two |
|  | ... | ${3} | four | ${last} |  |
|  |  | Log | ${var} |  |  |
| Example | :FOR | ${element} | IN | @{ELEMENTS} |  |
|  |  | Start Element | ${element} |  |  |

Hình Với vòng lặp thường

* Vòng lặp với nhiều biến
* Có thể cùng 1 lúc lấy nhiều biến mỗi lần lặp FOR

| **Test Case** | **Action** | **Argument** | **Argument** | **Argument** | **Arguments** |
| --- | --- | --- | --- | --- | --- |
| Example | :FOR | ${index} | ${english} | ${finnish} | IN |
|  | ... | 1 | cat | kissa |  |
|  | ... | 2 | dog | koira |  |
|  | ... | 3 | horse | hevonen |  |
|  |  | Do X | ${english} |  |  |
|  |  | Y Should Be | ${finnish} | ${index} |  |
|  | :FOR | ${name} | ${id} | IN | @{EMPLOYERS} |
|  |  | Create | ${name} | ${id} |  |

Hình Với vòng lặp nhiều biến

* Lặp trong khoảng
* Thực hiện lặp trong 1 khoảng xác định với cú pháp FOR index IN RANGE limit
* Sau IN RANGE, cột thứ 1 là vị trí bắt đầu, cột thứ 2 là vị trí kết thúc, cột thứ 3 là bước lặp (bỏ trống có nghĩa là bước lặp bằng 1, nếu bước lặp nhỏ hơn 0 thì vòng lặp thực hiện duyệt giảm dần)

| **Test Case** | **Action** | **Argument** | **Argument** | **Arg** | **Arg** | **Arg** |
| --- | --- | --- | --- | --- | --- | --- |
| Only upper limit | [Documentation] | Loops over | values | from 0 | to 9 |  |
|  | :FOR | ${index} | IN RANGE | 10 |  |  |
|  |  | Log | ${index} |  |  |  |
|  |  |  |  |  |  |  |
| Start and end | [Documentation] | Loops over | values | from 1 | to 10 |  |
|  | :FOR | ${index} | IN RANGE | 1 | 11 |  |
|  |  | Log | ${index} |  |  |  |
|  |  |  |  |  |  |  |
| Also step given | [Documentation] | Loops over | values | 5, 15, | and 25 |  |
|  | :FOR | ${index} | IN RANGE | 5 | 26 | 10 |
|  |  | Log | ${index} |  |  |  |
|  |  |  |  |  |  |  |
| Negative step | [Documentation] | Loops over | values | 13, 3, | and -7 |  |
|  | :FOR | ${index} | IN RANGE | 13 | -13 | -10 |
|  |  | Log | ${index} |  |  |  |
|  |  |  |  |  |  |  |
| Arithmetics | [Documentation] | Arithmetics | with | variable |  |  |
|  | :FOR | ${index} | IN RANGE | ${var}+1 |  |  |
|  |  | Log | ${index} |  |  |  |

Hình Với vòng lặp khoảng

* Thoát khỏi vòng lặp
* Từ khóa Exit For Loop và Exit For Loop If có tác dụng thoát khỏi vòng lặp giống như từ khóa break trong 1 số ngôn ngữ lập trình.

| **Test Case** | **Action** | **Argument** | **Argument** | **Argument** | **Argument** |
| --- | --- | --- | --- | --- | --- |
| Exit Example | ${text}= | Set Variable | ${EMPTY} |  |  |
|  | :FOR | ${var} | IN | one | two |
|  |  | Run Keyword If | '${var}' == 'two' | Exit For Loop |  |
|  |  | ${text}= | Set Variable | ${text}${var} |  |
|  | Should Be Equal | ${text} | one |  |  |

Hình Thoát khỏi vòng lặp

* Bỏ qua giá trị lặp
* Từ khóa Continue For Loop và Continue For Loop If có tác dụng chuyển sang lần lặp tiếp thep giống như từ khóa continue trong 1 số ngôn ngữ lập trình.

| **Test Case** | **Action** | **Argument** | **Argument** | **Argument** | **Argument** | **Argument** |
| --- | --- | --- | --- | --- | --- | --- |
| Continue Example | ${text}= | Set Variable | ${EMPTY} |  |  |  |
|  | :FOR | ${var} | IN | one | two | three |
|  |  | Continue For Loop If | '${var}' == 'two' |  |  |  |
|  |  | ${text} = | Set Variable | ${text}${var} |  |  |
|  | Should Be Equal | ${text} | onethree |  |  |  |

Hình Bỏ qua giá trị lặp

# Thực thi ca kiểm thử.

## Sử dụng cơ bản.

Robot Framwork thực thi các ca kiểm thử bằng dòng lệnh, kết quả trả về mặc định là file với định dạng là XML và 1 file html chứa báo cáo và log.

### Bắt đầu thực thi bản kiểm thử.

pybot|jybot|ipybot [options] data\_sources

python|jython|ipy -m robot.run [options] data\_sources

python|jython|ipy path/to/robot/run.py [options] data\_sources

java -jar robotframework.jar [options] data\_sources

* Thực thi kiểm thử mặc định là sử dụng pybot, jybot hoặc ipybot .
* Có thể sử dụng sự đường dẫn đến dữ liệu để kiểm thử. Ngoài ra, tuỳ chọn trên dòng lệnh có thể được sử dụng để thực hiện.

#### Xác định dữ liệu kiểm thử được thực hiện.

* Ca kiểm thử Robot Framwork được tạo ra trong các file và thư mục, và chúng được thực thi(excuted) bằng đường dẫn đến file hoặc thư mục đó.
* Đường dẫn có thể là tuyệt đối hoặc tương đối.

Ví dụ:

pybot test\_cases.html

pybot path/to/my\_tests/

pybot c:\robot\tests.txt

* Robot Framwork có thể cung cấp đường dẫn đến một số tập tin chứa ca kiểm thử hoặc thư mục.

pybot my\_tests.html your\_tests.html

pybot --name Example path/to/tests/pattern\_\*.html

### Sử dụng tuỳ chọn dòng lệnh

Robot Framwork cung cấp 1 số tuỳ chọn dòng lệnh để kiểm soát quá trình thực hiện các ca kiểm thử và kết qua thu được.

#### Sử dụng tuỳ chọn

Khi tuỳ chọn được sử dụng thì chúng phải được đưa ra giữa các script và nguồn dữ liệu. Ví dụ:

pybot -L debug my\_tests.txt

pybot --include smoke --variable HOST:10.0.0.42 path/to/tests/

#### Tuỳ chọn ngắn và dài.

Đối với các tuỳ chọn là 1 chữ cái như thì chỉ cần 1 dấu “-” đặt trước chữ cái đó như là –N, còn tuỳ chọn là các từ thì cần 2 dấu “-” đặt trước từ đó ví dụ: --name.

#### Thiết lập giá trị tuỳ chọn

Hầu hết các tuỳ chọn đều yêu cầu 1 giá trị được đưa ra sau tên tuỳ chọn. Tuỳ chọn ngắn và dài đều chấp nhận giá trị tách ra từ tên tuỳ chọn, như –include tag hoặc –i tag

#### Mẫu đơn giản

Nhiều tuỳ chọn dòng lệnh có đối số như là mô hình đơn giản (simple patterns) theo các quy tắc sau:

* \* là ký tự đại diện cho bất kỳ chuỗi, thậm chí là 1 chuỗi rỗng.
* ? là ký tự đại diện cho bất kỳ chữ cái duy nhất nào.

Ví dụ:

--test Example\* # Matches tests with name starting 'Example', case insensitively.

--include f?? # Matches tests with a tag that starts with 'f' or 'F' and is three characters long.

#### Mẫu tag(tag patterns)

Hầu hết các từ khoá liên quan đến tuỳ chọn chấp nhận đối số như là từ khoá mẫu. Họ có tất cả các đặc tính tương tự như các mẫu đơn giản, nhưng họ cũng hỗ trợ AND, OR và NOT. Có thể sử dụng kết hợp 2 hoặc nhiều thẻ cá nhân hoặc các mẫu với nhau.

Ví dụ:

--include xANDyORz # Matches tests that contain either tags 'x' and 'y', or tag 'z'.

--include xORyNOTz # Matches tests that contain either tag 'x' or 'y', but not tag 'z'.

--include xNOTyANDz # Matches tests that contain tag 'x', but not tags 'y' and 'z'.

### Kết quả kiểm thử

#### Đầu ra dòng lệnh

Đầu ra dễ thầy nhất từ quá trình kiểm thử là kết quả hiển thị trên dòng lệnh và được hiển thị dưới thời gian thực.

Ví dụ:

=============================================================================

Example test suite

=============================================================================

First test :: Possible test documentation | PASS |

-----------------------------------------------------------------------------

Second test | FAIL |

Error message is displayed here

=============================================================================

Example test suite | FAIL |

2 critical tests, 1 passed, 1 failed

2 tests total, 1 passed, 1 failed

=============================================================================

Output: /path/to/output.xml

Report: /path/to/report.html

Log: /path/to/log.html

#### Tạo ra tập tin đầu ra

Các đầu ra dòng lệnh là hạn chế, và các tập tin đầu ra riêng thường cần cho việc kiểm tra các kết quả thử nghiệm. Như ví dụ trên 3 tập tin được tạo ra theo mặc định.

#### Các mã trả về

Tình trạng của hệ thống khi chạy sẽ được thể hiện qua các mã trả về. Các mã trả về quan trọng được thể hiện trong bảng dưới.

| **Mã** | **Giải thích** |
| --- | --- |
| 0 | Tất cả các ca kiểm thử đều tốt. |
| 1-249 | Trả về mã số của các ca kiểm thử quan trọng thất bại. |
| 250 | 250 hoặc nhiều hơn các ca kiểm thử thất bại. |
| 251 | Giúp đỡ hoặc thông tin thêm về phiên bản. |
| 252 | Kiểm tra dữ liệu không hợp lệ hoặc tuỳ chọn dòng lệnh. |
| 253 | Dừng quá trình thực thi bởi người sử dụng. |
| 255 | Lỗi nội bộ bất ngờ. |

#### Lỗi và cảnh báo trong quá trình thực thi

Trong quá trình thực thi có thể có vấn đề bất ngờ như không nhập mật khẩu hoặc 1 tập tin tài nguyên bị khoá. Tuỳ thuộc vào mức độ nghiêm trọng của vấn đề mà được phân loại như các lỗi hoặc các cảnh bào và được viết vào giao diện điều khiển và vào tập tin log. Ví dụ như file log:

|  |  |  |
| --- | --- | --- |
| 20090322 19:58:42.528 | ERROR | Error in file '/home/robot/tests.html' in table 'Setting' in element on row 2: Resource file 'resource.html' does not exist |
| 20090322 19:58:43.931 | WARN | Keyword 'SomeLibrary.Example Keyword' is deprecated. Use keyword `Other Keyword` instead. |

### Đối số file

File tham số cho phép tất cả hoặc một số tuỳ chọn dòng lệnh và đối số vào 1 tập tin bên ngoài, nơi chúng sẽ được đọc.

File tham số được đưa vào cùng với tuỳ chọn --argumentfile (-A)

#### Cú pháp tập tin tham số

File tham số có thể chứa cả tùy chọn dòng lệnh và đường dẫn đến các dữ liệu kiểm thử, một tuỳ chọn hoặc nguồn dữ liệu trên mỗi dòng. Cả hai tùy chọn ngắn và dài đều được hỗ trợ. Ví dụ

--doc This is an example (where "special characters" are ok!)

--metadata X:Value with spaces

--variable VAR:Hello, world!

# This is a comment

path/to/my/tests

#### Sử dụng file đối số

File tham số có thể được sử dụng 1 mình để chứa tất cả các tuỳ chọn và các đường dẫn đến các dữ liệu kiểm thử hoặc cùng với các tuỳ chọn khác và đường dẫn. Các tùy chọn trong các tập tin đối số có thể ghi đè lên các lựa chọn trước đó, và các tùy chọn của nó có thể được ghi đè bởi các tùy chọn sau. Ví dụ:

pybot --argumentfile all\_arguments.txt

pybot --name Example --argumentfile other\_options\_and\_paths.txt

pybot --argumentfile default\_options.txt --name Example my\_tests.html

pybot -A first.txt -A second.txt -A third.txt tests.txt

### Tạo kịch bản bắt đầu(Creating start-up scripts)

Ca kiểm thử thường được thực hiện tự động bởi 1 hệ thống tích hợp hoặc 1 số cơ chế khác. Trong trường hợp như vậy cần có 1 kịch bản để bắt đầu thực hiện kiểm tra và cho kết quả đầu ra sau xử lý bằng cách nào đó.

Trong môi trường UNIX-like, shell scripts cung cấp 1 cơ chế đơn giản nhưng mạnh mẽ để tạo các kịch bản.

Ví dụ đầu là các kiểm thử web tương tự được thực hiện với các trình duyệt khác nhau và kết hợp các kết quả sau đó.

#!/bin/bash

pybot --variable BROWSER:Firefox --name Firefox --log none --report none --output out/fx.xml login

pybot --variable BROWSER:IE --name IE --log none --report none --output out/ie.xml login

rebot --name Login --outputdir out --output login.xml out/fx.xml out/ie.xml

Thực hiện ví dụ trên trên windows

@echo off

call pybot --variable BROWSER:Firefox --name Firefox --log none --report none --output out\fx.xml login

call pybot --variable BROWSER:IE --name IE --log none --report none --output out\ie.xml login

call rebot --name Login --outputdir out --output login.xml out\fx.xml out\ie.xml

### Vấn đề gỡ lỗi

Một ca kiểm thử thất bại vì các hệ thống theo kiểm thử hoạt động không chính xác, trong trường hợp các kiểm thử đã thấy 1 lỗi hoặc bản thân các kiểm thử là lỗi. Các thông báo lỗi được hiển thị trên đầu ra và dòng lệnh trong file báo cáo.

Nếu các tập tin log không cung cấp đủ thông tin nó có thể thực hiện kiểm tra với một mức độ log thấp hơn.

## Thực thi kiểm thử

Phần này mô tả cách cấu trúc test suite được tạo ra từ các dữ liệu, phân tích cú pháp được thực thi, làm thế nào sau khi thất bại, và làm thế nào để ngăn chăn việc thực hiện kiểm tra toàn bộ.

### Dòng chảy thực thi

#### Thực thi suites và kiểm thử

Ca kiểm thử luôn được thực thi trong vòng 1 test suite. 1 test suite được tạo ra từ 1 tập tin có chứa ca kiểm thử.

Mặc định là tất cả các ca kiểm thử sẽ được thực thi từ các test suite cấp cao, nhưng có thể chọn các ca kiểm thử khác nhau bằng sử dụng tuỳ chọn --test, --suite, --include.

#### Thiết lập và teardowns

Thiết lập và teardowns có thể được sử dụng trên test suite, các ca kiểm thử và các cấp độ từ khoá người dùng.

#### Thiết lập suite

Nếu 1 test suite có 1 thiết lập, nó được thực hiện trước khi kiểm thử nó và các con của suite. Nếu thành công thì tiếp tục. Nếu thất bại thì đánh dấu lại.

Thiết lập suite thường được sử dụng để thiết lập môi trường thử nghiệm.

#### Suite teardown

Nếu 1 test suite có 1 teardown, nó được thực thi sau tất cả các ca kiểm thử của nó và con của nó.

Teardowns suite được sử dụng chủ yếu cho việc làm sạch môi trường kiểm thử sau khi thực hiện.

#### Thiết lập kiểm thử

Thiết lập thử nghiệm có thể được thực hiện trước khi các từ khoá của ca kiểm thử. Thường được sử dụng để thiết lập kiểm thử cho các môi trường với trường hợp kiểm thử cụ thể.

#### Kiểm thử teardown

Kiểm thử teardown có thể được thực hiện sau khi các ca kiểm thử đã được thực hiện. Nó thực hiện bất kể tình trạng của ca kiểm thử là thành công hay thất bại.

Được sử dụng chủ yếu để dọn dẹp.

#### Thứ tự thực hiện

Ca kiểm thử trong test suite được thực hiện theo thứ tự như chính được định nghĩa trong file.

Nếu có nhu cầu thực hiện theo thứ tự có thể thêm các tiền tố như 01, 02 vào tập tin, thư mục. Ví dụ:

01\_\_my\_suite.html -> My Suite

02\_\_another\_suite.html -> Another Suite

#### Thực thi thành công

Thông thường các ca kiểm thử, các thiết lập và teardowns được coi là thành công(pass) nếu tất cả các từ khoá được thực thi và không có khoá nào trong số đó thất bại trong quá trình thực thi.

### Tiếp tục khi ca kiểm thất bại

Thông thường các ca kiểm thử được dừng lại ngay khi có 1 từ khoá thất bại. Điều này có nhược điểm mà thường từ khoá tiếp theo sẽ cung cấp thêm thông tin về tình trạng của hệ thống(được sử dụng từ Robot Framwork 2.5 trở lên).

#### Quá trình thực thi thất bại từ các từ khoá

Từ khoá Library báo cáo các lỗi sử dụng các ngoại lệ, và nó có thể sử dụng các ngoại lệ đặc biết để nói với phần core của framwork thực hiện bất chấp vấn đề bị lỗi xảy ra.

Khi kết thúc ca kiểm thử và có 1 hoặc nhiều lỗi xảy ra, các ca kiểm thử được đánh dấu là thất bại và đượ liệt kê trong thông báo lỗi cuối cùng.

Several failures occurred:

1) First error message.

2) Second error message ...

#### Từ khoá Run Keywork And Continue On Failure

Từ khoá Run Keywork And Continue On Failure cho phép chuyển bất kỳ thất bại nào thành thất bại có thể tiếp tục được.

#### Thực hiện tiếp tục trên teardowns tự động

Trong teardowns tất cả các từ khoá trong các cấp luôn được thực thi.

### Dừng quá trình thực thi kiểm thử

Có thể dùng tổ hợp phím Ctrl-C, sử dụng lệnh kill, hoặc có thể sử dụng tuỳ chọn --exitonfailure để dừng khi ca kiểm thử đầu tiên thất bại.

## Kết quả đầu ra sau xử lý

Tập tin đầu ra XML được tạo ra trong quá trình thực thi có thể có được sau xử lý và được sử dụng bằng các công cụ rebot.

### Sử dụng công cụ rebot

**Tóm tắt**

rebot|jyrebot|ipyrebot [options] robot\_outputs

python|jython|ipy -m robot.rebot [options] robot\_outputs

python|jython|ipy path/to/robot/rebot.py [options] robot\_outputs

java -jar robotframework.jar rebot [options] robot\_outputs

Có rebot chạy trên Python nhưng cũng có jpyrebot chạy trên Jython và IronPython tương ứng.

#### Lựa chọn tuỳ chọn và đối số

Cú pháp cơ bản để dùng rebot giống như khi thực thi ca kiểm thử và hầu hết các tuỳ chọn dòng lệnh là giống hệt nhau.

### Tạo các báo cáo và các log khác nhau

Bạn có thể sử dụng repost để tạo ra các báo cáo và các bản ghi được tạo ra tự động trong quá trình thực thi.

rebot output.xml

rebot path/to/output\_file.xml

rebot --include smoke --name Smoke\_Tests c:\results\output.xml

### Kết hợp các kết quả đầu ra

Một tính năng quan trọng trong rebot là khả năng kết hợp các kết quả đầu ra từ các vòng kiểm tra khác nhau. Khả năng này cho phép tạo ra 1 báo cáo tổng thể thừ tất cả các kết quả đầu ra.

rebot output1.xml output2.xml

rebot outputs/\*.xml