RobotFramework

Ursicio Martin Martino

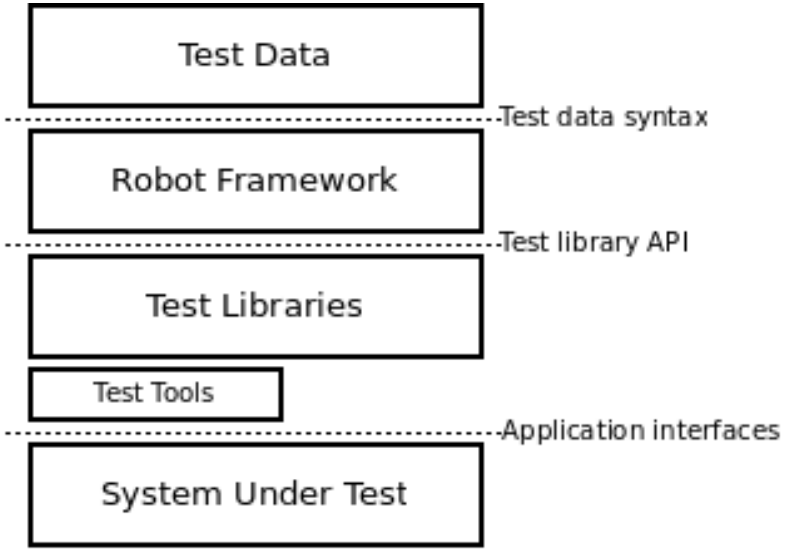
Raul Martin Cabello

Robot Framework

- Robot Framework is a Python-based, extensible keyword-driven test automation framework
  - Provides ability to create reusable higher-level keywords from the existing keywords.
  - Enables easy-to-use tabular syntax for creating test cases in a uniform way.
  - Provides easy-to-read result reports and logs in HTML format.
  - Provides a simple library API for creating customized test libraries which can be implemented natively with either **Python** or Java.
  - Provides a command line interface and XML based output files for integration into existing build infrastructure (continuous integration systems).->
  - Provides support for Selenium for web testing, Java GUI testing, running processes, Telnet, SSH, and so on.
  - Supports creating data-driven test cases and behavior-driven test cases.
  - Provides test-case and test-suite -level setup and teardown.

High-level architecture

- Robot Framework is a generic, application and technology independent framework. It has a highly modular architecture illustrated in the diagram below.



- The test data is in simple, easy-to-edit tabular format. When Robot Framework is started, it processes the test data, executes test cases and generates logs and reports. The core framework does not know anything about the target under test, and the interaction with it is handled by test libraries. Libraries can either use application interfaces directly or use lower level test tools as drivers.

Demo

- The demo application is a very simple calculator implemented with Python (`calculator.py`).

- Test cases
  - The demo contains three different test case files illustrating three different approaches for creating test cases with Robot Framework
  - `keyword_driven.robot`_
    - Example test cases using the *keyword-driven* testing approach.
    - All tests contain a workflow constructed from keywords in `CalculatorLibrary.py`
  - `data_driven.robot`_
    - Example test cases using the *data-driven* testing approach.
    - The *data-driven* style works well when you need to repeat the same workflow multiple times.
  - `gherkin.robot`_
    - Example test case using the *gherkin* syntax.
    - This test has a workflow similar to the *keyword-driven* examples. The difference is that the keywords use higher abstraction level and their arguments are embedded into the keyword names.

Demo

- Test library
  - All test cases interact with the calculator using a custom test library named

  `CalculatorLibrary.py`. In practice the library is just a Python class with methods that create the keywords used by the test cases.

  - Generated library documentation makes it easy to see what keywords the library provides. This documentation is created with Libdoc tool integrated with the framework:

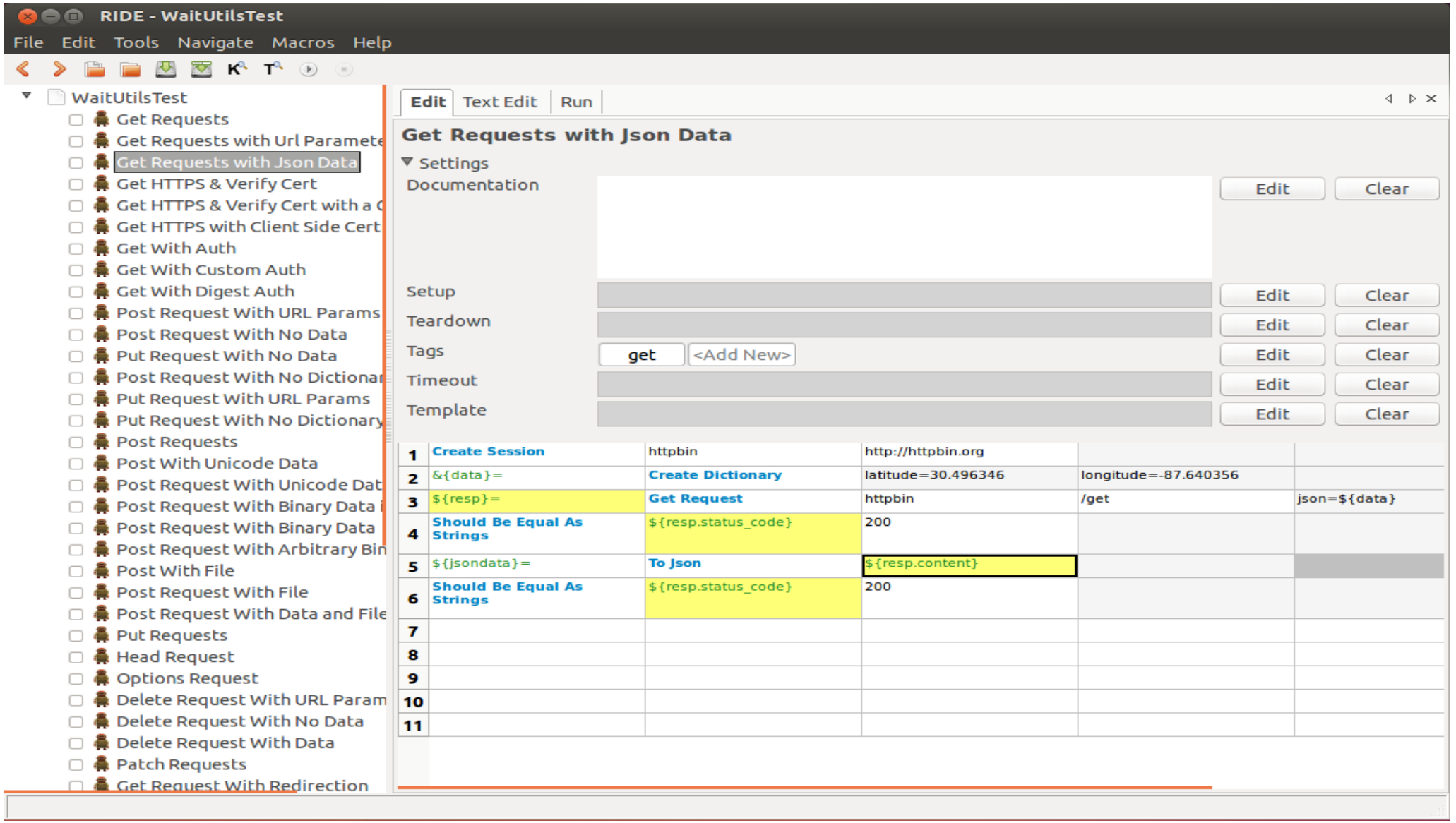
  - firefox CalculatorLibrary.html
- Generated results
  - robot --name Robot --loglevel DEBUG keyword_driven.robot data_driven.robot gherkin.robot

  - firefox log.html

Ride

- Is the integrated development environment (IDE) to implement automated tests for the Robot Framework
- Tree-like structure
- For each testsuite individual testcases can be selected
- Setup and Teardown at Test Suite and Test Case level
- Edition: Tabular and plain text
- Auto completion for keywords
- Possibility to choose test cases to execute
- Nice reports

Ride


RIDE - WaitUtilsTest
File Edit Tools Navigate Macros Help
WaitUtilsTest
Get Requests
Get Requests with Url Paramete
Get Requests with Json Data
Get HTTPS & Verify Cert
Get HTTPS & Verify Cert with a C
Get HTTPS with Client Side Cert
Get With Auth
Get With Custom Auth
Get With Digest Auth
Post Request With URL Params
Post Request With No Data
Put Request With No Data
Post Request With No Dictionar
Put Request With URL Params
Put Request With No Dictionary
Post Requests
Post With Unicode Data
Post Request With Unicode Dat
Post Request With Binary Data i
Post Request With Binary Data
Post Request With Arbitrary Bin
Post With File
Post Request With File
Post Request With Data and File
Put Requests
Head Request
Options Request
Delete Request With URL Param
Delete Request With No Data
Delete Request With Data
Patch Requests
Get Request With Redirection
Edit
Text Edit
Run
Get Requests with Json Data
Settings
Documentation
Edit
Clear
Setup
Teardown
Tags
get
<Add New>
Timeout
Template
1 Create Session httpbin http://httpbin.org
2 &{data}= Create Dictionary latitude=30.496346 longitude=-87.640356
3 ${resp}= Get Request httpbin /get json=${data}
4 Should Be Equal As Strings ${resp.status_code} 200
5 ${jsondata}= To Json ${resp.content}
6 Should Be Equal As Strings ${resp.status_code} 200
7
8
9
10
11