rtcclient Documentation

Release v0.1.dev95

August 17, 2015

Contents

A Python-based Client/API for Rational Team Concert (RTC)

CHAPTER 1

About this library

IBM® Rational Team Concert™, is built on the Jazz platform, allowing application development teams to use one tool to plan across teams, code, run standups, plan sprints, and track work. For more info, please refer to here.

IMPORTANT NOTE: THIS IS NOT AN OFFICIAL Python-based RTC Client.

This library can help you:

- Interacts with an RTC server to retrieve objects which contain the detailed information/configuration, including Project Areas, Team Areas, Workitems, etc
- Creates all kinds of Workitems through self-customized templates or Copies from some existing Workitems
- Add comments to the retrieved Workitems
- Query Workitems using specified filtered rules
- Logs all the activities and messages during your operation

CHAPTER 2

Python & Rational Team Concert versions

The project have been tested against `Rational Team Concert` **5.0.1** and **5.0.2** on Python 2.6, 2.7 and 3.3.

CHAPTER 3

Important Links

Support and bug-reports: https://github.com/dixudx/rtcclient/issues?q=is%3Aopen+sort%3Acomments-desc

Project source code: https://github.com/dixudx/rtcclient

Project documentation:

CHAPTER 4

Installation

To install rtcclient, simply:

```
$ pip install rtcclient
```

CHAPTER 5

Example

RTCClient is intended to map the objects in RTC (e.g. Project Areas, Team Areas, Workitems) into easily managed Python objects:

```
>>> from rtcclient.utils import setup_basic_logging
>>> from rtcclient.client import RTCClient
# you can remove this if you don't need logging
# default logging for console output
>>> setup_basic_logging()
>>> url = "https://your_domain:9443/jazz"
>>> username = "your_username"
>>> password = "your_password"
>>> myclient = RTCClient(url, username, password)
# it will be faster if returned properties is specified
# see in below query example
>>> wk = myclient.getWorkitem(123456) # get a workitem whose id is 123456
# get all workitems
# If both projectarea_id and projectarea_name are None, all the workitems
# in all ProjectAreas will be returned
>>> workitems_list = myclient.getWorkitems(projectarea_id=None,
                                           projectarea_name=None)
>>> myquery = myclient.query # query class
>>> projectarea_name = "your_projectarea_name"
# customize your query string
# below query string means: query all the workitems with title "use case 1"
>>> myquerystr = 'dc:title="use case 1"'
# specify the returned properties: title, id, state, owner
# This is optional. All properties will be returned if not specified
>>> returned_prop = "dc:title,dc:identifier,rtc_cm:state,rtc_cm:ownedBy"
>>> queried_wis = myquery.queryWorkitems(query_str=myquerystr,
                                         projectarea_name=projectarea_name,
                                         returned_properties=returned_prop)
```

CHAPTER 6

Testing

Using a virtualenv is recommended. Setuptools will automatically fetch missing test dependencies.

If you have installed the tox on your system already, you can run the tests using pytest with the following command:

```
virtualenv
source .venv/bin/active
(venv) tox -e py27
(venv) tox -e py33
(venv) tox -e pep8
```