rtcclient Documentation

Release 0.1.dev95

Di Xu

September 18, 2015

Contents

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 1

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: https://readthedocs.org/projects/rtcclient/

CHAPTER 2

User Guide

2.1 Authors

Di Xu <dixudx@users.noreply.github.com> stephenhsu <stephenhsu90@gmail.com>

2.2 Introduction

In this section, some common terminologies are introduced. For more information, please visit Rational Collaborative Lifecycle Management Solution

2.2.1 Project Area

Project Area is, quite simply, an area in the repository where information about the project is stored.

In each of the Collaborative Lifecycle Management (CLM) applications, teams perform their work within the context of a project area. A project area is an area in the repository where information about one or more software projects is stored. **A project area defines the project deliverables, team structure, process, and schedule**. You access all project artifacts, such as iteration plans, work items, requirements, test cases, and files under source control within the context of a project area. Each project area has a process, which governs how members work.

For example, the project area process defines:

- User roles
- Permissions assigned to roles
- Timelines and iterations
- Operation behavior (preconditions and follow-up actions) for Change and Configuration Management and Quality Management
- Work item types and their state transition models (for Change and Configuration Management and Quality Management)

A project area is stored as a top-level or root item in a repository. A project area references project artifacts and stores the relationships between these artifacts. Access to a project area and its artifacts is controlled by access control settings and permissions. A project area cannot be deleted from the repository; however, it can be archived, which places it in an inactive state.

2.2.2 Team Area

You can create a team area to assign users in particular roles for work on a timeline or a particular set of deliverables. You can create a team area within an existing project area or another team area to establish a team hierarchy.

2.2.3 Role

Each project area and each team area can define a set of roles. The defined roles are visible in the area where they're declared and in all child areas. Roles defined in the project area can be assigned to users for the whole project area or they can be assigned in any team area. Roles defined in a team area can similarly be assigned in that team or in any child team. The ordering of roles in this section determines how they will be ordered in other sections of the editor, but it does not affect the process runtime.

2.2.4 Administrator

If you require permissions, contact an administrator. Project administrators can modify and save this project area and its team areas.

2.2.5 PlannedFor

In modern software development, a release is divided into a series of fixed-length development periods, typically ranging from two to six weeks, called iterations. Planning an iteration involves scheduling the work to be done during an iteration and assigning individual work items to members of the team.

Iteration planning takes place in the context of a project area. Each project area has a development line that is divided into development phases or iterations. For each iteration, you can create an iteration plan.

The project *plannedfor* defines a start and end date along with an iteration breakdown.

2.2.6 Workitem

You can use work items to manage and track not only your work, but also the work of your team.

2.2.7 Workitem Type

A workitem type is a classification of work items that has a specific set of attributes. Each predefined process template includes the work item types that allow users to work in that process. For example, the Scrum process includes work item types such as *Epic*, *Story*, *Adoption Item*, *Task*, and *Retrospective*, which support an agile development model. The Formal Project Management process, which supports a more traditional development model, includes workitem types such as *Project Change Request*, *Business Need*, and *Risk*. Some work item types, such as *Defect* and *Task*, are used by multiple processes.

2.2.8 Workitem Type Category

Each work item type belongs to a work item category. Multiple work item types can belong to the same work item category. The work item types of a work item type category share workflow and custom attributes. When you create a work item type, you must associate it with a category. If you intend to define a unique workflow for the new work item type, create a new category and associate it with the work item type. Otherwise, you can associate the work item type with an existing category.

2.3 Workitem Attributes [1]

Attributes identify the information that you want to capture when users create and modify work items. Attributes are similar to fields in records. Work item types include all the built-in attributes that are listed in below Table. Note, however, that not every ready-to-use work item presentation is configured to display all of the built-in attributes in the Rational Team Concert™ Eclipse and web clients. You can customize the attributes that a work item type contains and the presentations that are used to display these attributes. For example, you can customize attributes to add behavior. Such behaviors can include validating an attribute value, or setting an attribute value that is based on other attribute values.

All the attributes of the *rtcclient.workitem.Workitem* can be accessed through **dot notation** and **dictionary**.

2.3.1 Built-in Attributes

Table1. Built-in Attributes

[1] Workitem Customization Overview

Name	Type	ID	Description
Archived	Boolean	archived	Specifies whether the work item is archived.
Comments	Comments	comments	Comments about the work item.
Corrected Estimate	Duration	correctedEstimate	Correction to the original time estimate (as specified by the Estimate attribute) to resolve the work item.
Created By	Contributor	creator	User who created the work item.
Creation Date	Timestamp	created	Date when the work item was created.
Description	Large HTML	description	Detailed description of the work item. For example, the description for a defect might include a list of steps to follow to reproduce the defect. Any descriptions that are longer than 32 KB are truncated, and the entire description is added as an attachment.
Due Date	Timestamp	due	Date by which the resolution of the work item is due.
Estimate	Duration	estimate	Estimated amount of time that it takes to resolve the work item.
Filed Against	Category	filedAgainst	Category that identifies the component or functional area that the work item belongs to. For example, your project might have GUI, Build, and Documentation categories. Each category is associated with a team area; that team is responsible for responding to the work item.
Found In	Deliverable	foundIn	Release in which the issue described in the work item was identified.
Id	Integer	identifier	Identification number that is associated with the work item.
Modified By	Contributor	modifiedBy	User who last modified the work item.
Modified Date	Timestamp	modified	Date when the work item was last modified.
Owned By	Contributor	ownedBy	Owner of the work item.
Planned For	Iteration	plannedFor	Iteration for which the work item is planned.
Priority	Priority	priority	Ranked importance of a work item. For example, *Low*, *Medium*, or *High*.
Project Area	ProjectArea	projectArea	Area in the repository where information about the project is stored.
Resolution	Small String	resolution	How the work item was resolved.
Resolution Date	Timestamp	resolved	Date when the work item was resolved.
Resolved By	Contributor	resolvedBy	User who resolved the work item.

Table2. Built-in Attributes (cont'd)

Name	Type	ID	Description
Restricted Access	UUID	contextId	Scope of access to the work item.
Severity	Severity	severity	Indication of the impact of the work item. For example, *Minor*, *Normal*, *Major*, or *Critical*.
Start Date	Timestamp	startDate	Date when work began on the work item.
Status	Small String	state	Status of the work item. For example, *New*, *In Progress*, or *Resolved*.
Subscribed By	Subscriptions	subscribers	Users who are subscribed to the work item.
Summary	Medium HTML	title	Brief headline that identifies the work item.
Tags	Tag	subject	Tags that are used for organizing and querying on work items.
Time Spent	Duration	timeSpent	Length of time that was spent to resolve the work item.
Type	Type	type	Type of work item. Commonly available types are *Defect*, *Task*, and *Story*.

2.4 Installation

This part of the documentation covers the installation of rtcclient. The first step to using any software package is getting it properly installed.

2.4.1 Distribute & Pip

Installing rtcclient is simple with pip, just run this in your terminal:

```
$ pip install rtcclient
```

or, with easy_install:

```
$ easy_install rtcclient
```

2.4.2 Get from the Source Code

RTCClient is actively developed on GitHub, where the code is always available.

You can either clone the public repository and checkout released tags (e.g. tag 0.1.dev95):

```
$ git clone git://github.com/dixudx/rtcclient.git
$ cd rtcclient
$ git checkout tags/0.1.dev95
```

Once you have a copy of the source, you can embed it in your Python package, or install it into your site-packages easily:

```
$ python setup.py install
```

2.5 Quick Start

Eager to get started? This page gives a good introduction in how to get started with rtcclient.

First, make sure that:

- rtcclient is *installed*
- rtcclient is up-to-date

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

Let's get started with some simple examples.

2.5.1 Setup Logging

You can choose to enable logging during the using of rtcclient. Default logging is for console output. You can also add your own *logging.conf* to store all the logs to your specified files.

```
>>> from rtcclient.utils import setup_basic_logging
# you can remove this if you don't need logging
>>> setup_basic_logging()
```

2.5.2 Add a Connection to the RTC Server

Adding a connection with RTC Server is very simple.

Begin by importing the *RTCClient* module:

```
>>> from rtcclient.client import RTCClient
```

Now, let's input the url, username and password of this to-be-connected RTC Server. For this example,

```
>>> url = "https://your_domain:9443/jazz"
>>> username = "your_username"
>>> password = "your_password"
>>> myclient = RTCClient(url, username, password)
```

2.5.3 Get a Workitem

You can get a workitem by calling `rtcclient.workitem.Workitem.getWorkitem`. The attributes of a workitem can be accessed through **dot notation** and **dictionary**.

Some common attributes are listed in *Built-in Attributes*.

For example,

```
>>> wk = myclient.getWorkitem(123456)
# get a workitem whose id is 123456
# this also works: getting the workitem using the equivalent string
>>> wk2 = myclient.getWorkitem("123456")
# wk equals wk2
>>> wk == wk2
True
>>> wk
<Workitem 123456>
```

```
>>> str(wk)
'141488'
>>> wk.identifier
u'141488'
# access the attributes through dictionary
>>> wk["title"]
u'title demo'
# access the attributes through dot notation
>>> wk.title
u'title demo'
>>> wk.state
u'Closed'
>>> wk.description
u'demo description'
>>> wk.creator
u'tester1@email.com'
>>> wk.created
u'2015-07-16T08:02:30.658Z'
>>> wk.comments
[u'comment test 0', u'add comment test 1', u'add comment test 2']
```

2.5.4 About Returned Properties

You can also customize your preferred properties to be returned by specifying **returned_properties** when the called methods have this optional parameter, which can also **GREATLY IMPROVE** the performance of this client especially when getting or querying lots of workitems.

For the meanings of these attributes, please refer to *Built-in Attributes*.

Important Note: **returned_properties** is an advanced parameter, the returned properties can be found in *instance_obj.field_alias.values()*, e.g. *myworkitem1.field_alias.values()*. If you don't care the performance, just leave it alone with *None*.

```
>>> import pprint
# print the field alias
>>> pprint.pprint(wk2.field_alias, width=1)
{u'affectedByDefect': u'calm:affectedByDefect',
 u'affectsExecutionResult': u'calm:affectsExecutionResult',
 u'affectsPlanItem': u'calm:affectsPlanItem',
 u'apply_step': u'rtc_cm:apply_step',
 u'archived': u'rtc_cm:archived',
 u'blocksTestExecutionRecord': u'calm:blocksTestExecutionRecord',
 u'comments': u'rtc_cm:comments',
 u'contextId': u'rtc_cm:contextId',
 u'correctedEstimate': u'rtc_cm:correctedEstimate',
 u'created': u'dc:created',
 u'creator': u'dc:creator',
 u'description': u'dc:description',
 u'due': u'rtc_cm:due',
 u'elaboratedByArchitectureElement': u'calm:elaboratedByArchitectureElement',
 u'estimate': u'rtc_cm:estimate',
 u'filedAgainst': u'rtc_cm:filedAgainst',
 u'foundIn': u'rtc_cm:foundIn',
 u'identifier': u'dc:identifier',
 u'implementsRequirement': u'calm:implementsRequirement',
 u'modified': u'dc:modified',
 u'modifiedBy': u'rtc_cm:modifiedBy',
```

```
 u'ownedBy': u'rtc_cm:ownedBy',
 u'plannedFor': u'rtc_cm:plannedFor',
 u'priority': u'oslc_cm:priority',
 u'progressTracking': u'rtc_cm:progressTracking',
 u'projectArea': u'rtc_cm:projectArea',
 u'relatedChangeManagement': u'oslc_cm:relatedChangeManagement',
 u'relatedExecutionRecord': u'calm:relatedExecutionRecord',
 u'relatedRequirement': u'calm:relatedRequirement',
 u'relatedTestCase': u'calm:relatedTestCase',
 u'relatedTestPlan': u'calm:relatedTestPlan',
 u'relatedTestScript': u'calm:relatedTestScript',
 u'relatedTestSuite': u'calm:relatedTestSuite',
 u'resolution': u'rtc_cm:resolution',
 u'resolved': u'rtc_cm:resolved',
 u'resolvedBy': u'rtc_cm:resolvedBy',
 u'schedule': u'oslc_pl:schedule',
 u'severity': u'oslc_cm:severity',
 u'startDate': u'rtc_cm:startDate',
 u'state': u'rtc_cm:state',
 u'subject': u'dc:subject',
 u'subscribers': u'rtc_cm:subscribers',
 u'teamArea': u'rtc_cm:teamArea',
 u'testedByTestCase': u'calm:testedByTestCase',
 u'timeSheet': u'rtc_cm:timeSheet',
 u'timeSpent': u'rtc_cm:timeSpent',
 u'title': u'dc:title',
 u'trackedWorkItem': u'oslc_cm:trackedWorkItem',
 u'tracksChanges': u'calm:tracksChanges',
 u'tracksRequirement': u'calm:tracksRequirement',
 u'tracksWorkItem': u'oslc_cm:tracksWorkItem',
 u'type': u'dc:type'}
```

Note: these field aliases may differ due to the type of workitems. But most of the common-used attributes will stay unchanged.

The *returned_properties* is a string **composed by the above values with comma separated**.

It will run faster if *returned_properties* is specified. Because the client will only get/request the attributes you specified.

```
>>> returned_properties = "dc:title,dc:identifier,rtc_cm:state,rtc_cm:ownedBy"
# specify the returned properties: title, identifier, state, owner
# This is optional. All properties will be returned if not specified
>>> wk_rp = myclient.getWorkitem(123456,
                                 returned_properties=returned_properties)
>>> wk_rp.identifier
u'141488'
# access the attributes through dictionary
>>> wk_rp["title"]
# access the attributes through dot notation
u'title demo'
>>> wk_rp.title
u'title demo'
>>> wk_rp.state
u'Closed'
>>> wk_rp.ownedBy
u'tester1@email.com'
```

2.5.5 Add a Comment to a Workitem

After getting the *rtcclient.workitem.Workitem* object, you can add a comment to this workitem by calling `addComment`.

```
>>> mycomment = wk.addComment("add comment test 3")
>>> mycomment
<Comment 3>
>>> mycomment.created
u'2015-08-22T03:55:00.839Z'
>>> mycomment.creator
u'tester1@email.com'
>>> mycomment.description
u'add comment test 3'
>>> str(mycomment)
'3'
```

2.5.6 Get all Workitems

All workitems can be fetched by calling *rtcclient.client.RTCClient.getWorkitems*. It will take a long time to fetch all the workitems in some certain project areas if there are already many existing workitems.

If both *projectarea_id* and *projectarea_name* are None, all the workitems in all project areas will be returned.

```
>>> workitems_list = myclient.getWorkitems(projectarea_id=None,
                                           projectarea_name=None,
                                           returned_properties=returned_properties)
# get all workitems in a specific project area
>>> projectarea_name = "my_projectarea_name"
>>> workitems_list2 = myclient.getWorkitems(projectarea_name=projectarea_name,
                                            returned_properties=returned_properties)
```

2.5.7 Query Workitems

After customizing your query string, all the workitems meet the conditions will be fetched.

```
>>> myquery = myclient.query # query class
>>> projectarea_name = "my_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"'
>>> returned_prop = "dc:title,dc:identifier,rtc_cm:state,rtc_cm:ownedBy"
>>> queried_wis = myquery.queryWorkitems(myquerystr,
                                         projectarea_name=projectarea_name,
                                         returned_properties=returned_prop)
```

More detailed and advanced syntax on querying, please refer to *query syntax*.

2.5.8 Query Workitems by Saved Query

You may have created several customized queries through RTC Web GUI or got some saved queries created by other team members. Using these saved queries

```
>>> myquery = myclient.query # query class
>>> saved_query_url = 'http://test.url:9443/jazz/xxxxxxxx&id=xxxxx'
>>> projectarea_name = "my_projectarea_name"
# get all saved queries
# WARNING: now the RTC server cannot correctly list all the saved queries
#          It seems to be a bug of RTC. Recommend using `runSavedQueryByUrl` to
#          query all the workitems if the query is saved.
>>> allsavedqueries = myquery.getAllSavedQueries(projectarea_name=projectarea_name)
# saved queries created by tester1@email.com
>>> allsavedqueries = myquery.getAllSavedQueries(projectarea_name=projectarea_name,
                                                 creator="tester1@email.com")
# my saved queries
>>> mysavedqueries = myquery.getMySavedQueries(projectarea_name=projectarea_name)
>>> mysavedquery = mysavedqueries[0]
>>> returned_prop = "dc:title,dc:identifier,rtc_cm:state,rtc_cm:ownedBy"
>>> queried_wis = myquery.runSavedQuery(mysavedquery,
                                        returned_properties=returned_prop)
```

2.5.9 Query Workitems by Saved Query Url

You can also query all the workitems directly using your saved query's url.

```
>>> myquery = myclient.query # query class
>>> saved_query_url = 'http://test.url:9443/jazz/xxxxxxxx&id=xxxxx'
>>> returned_prop = "dc:title,dc:identifier,rtc_cm:state,rtc_cm:ownedBy"
>>> queried_wis = myquery.runSavedQueryByUrl(saved_query_url,
                                             returned_properties=returned_prop)
```

2.6 Advanced Usage

This document covers some of rtcclient more advanced features.

2.6.1 Query Syntax [2]

The following section describes the basic query syntax.

Comparison Operators

- = : test for equality of a term,
- != : test for inequality of a term,
- < : test less-than,
- > : test greater-than,
- <= : test less-than or equal,
- >= : test greater-than or equal,
- `in` : test for equality of any of the terms.

Boolean Operators

- `and` : conjunction

[2] Change Management Query Syntax

Query Modifiers

- `/sort` : set the sort order for returned items

BNF

```
query      ::= (term (boolean_op term)*)+ modifiers
term       ::= (identifier operator)? value+ | (identifier "in")? in_val
operator   ::= "=" | "!=" | "<" | ">" | "<=" | ">="
boolean_op ::= "and"
modifiers  ::= sort?
sort       ::= "/sort" "=" identifier
identifier ::= word (":" word)?
in_val     ::= "[" value ("," value)* "]"
value      ::= (integer | string)
word       ::= /any sequence of letters and numbers, starting with a letter/
string     ::= '"' + /any sequence of characters/ + '"'
integer    ::= /any sequence of integers/
```

Notes

1. a word consists of any character with the Unicode class Alpha (alpha-numeric) as well as the characters ".", "-" and "_".
2. a string may include the quote character if preceded by the escape character "", as in "my "quoted" example".

2.6.2 Compose your Query String

Based on the above *query syntax*, it is easy to compose your own query string.

Important Note: For the *identifier* in *query syntax*, please refer to *field alias* and *Built-in Attributes*.

Here are several examples.

Example 1: Query all the defects with tags "bvt" whose state is not "Closed"

Note: here defects' state "default_workflow.state.s1" means "Closed". This may vary in your customized workitem type.

```
>>> query_str = ('dc:type="defect" and '
                 'rtc_cm:state!="default_workflow.state.s1" and '
                 'dc:subject="bvt"')
```

Example 2: Query all the defects which are modified after 18:42:30 on Dec. 02, 2008

Note: here defects' state "default_workflow.state.s1" means "Closed".

```
>>> query_str = 'dc:type="defect" and dc:modified>="12-02-2008T18:42:30"'
```

Example 3: Query all the defects with tags "bvt" or "testautomation"

```
>>> query_str = 'dc:type="defect" and dc:subject in ["bvt", "testautomation"]'
```

Example 4: Query all the defects owned/created/modified by "tester@email.com"

```
>>> user_url = "https://your_domain:9443/jts/users/tester@email.com"
>>> query_str = 'dc:type="defect" and rtc_cm:ownedBy="%s"' % user_url
>>> query_str = 'dc:type="defect" and dc:creator="%s"' % user_url
>>> query_str = 'dc:type="defect" and rtc_cm:modifiedBy="%s"' % user_url
```

Note: please replace *your_domain* with your actual RTC server domain.

Example 5: Query all the defects whose severity are "Critical"

```
>>> projectarea_name="My ProjectArea"
>>> severity = myclient.getSeverity("Critical",
                                    projectarea_name=projectarea_name)
>>> query_str = 'dc:type="defect" and oslc_cm:severity="%s"' % severity.url
```

Example 6: Query all the defects whose priority are "High"

```
>>> projectarea_name="My ProjectArea"
>>> priority = myclient.getPriority("High",
                                    projectarea_name=projectarea_name)
>>> query_str = 'dc:type="defect" and oslc_cm:priority="%s"' % priority.url
```

Example 7: Query all the defects whose FiledAgainst are "FiledAgainstDemo"

```
>>> projectarea_name="My ProjectArea"
>>> filedagainst = myclient.getFiledAgainst("FiledAgainstDemo",
                                            projectarea_name=projectarea_name)
>>> query_str = 'dc:type="defect" and rtc_cm:filedAgainst="%s"' % filedagainst.url
```

CHAPTER 3

API Documentation

3.1 Client

class rtcclient.client.**RTCClient**(*url*, *username*, *password*, *searchpath=None*)

A wrapped class for RTC Client to perform all related operations

Parameters

- **url** – the rtc url (e.g. https://your_domain:9443/jazz)
- **username** – the rtc username
- **password** – the rtc password
- **searchpath** – the folder to store your templates. If *None*, the default search path (/your/site-packages/rtcclient/templates) will be loaded.

Tips: You can also customize your preferred properties to be returned by specified *returned_properties* when the called methods have this optional parameter, which can also GREATLY IMPROVE the performance of this client especially when getting or querying lots of workitems.

Important Note: *returned_properties* is an advanced parameter, the returned properties can be found in *ClassInstance.field_alias.values()*, e.g. *myworkitem1.field_alias.values()*. If you don't care the performance, just leave it alone with *None*.

checkProjectAreaID(*projectarea_id*, *archived=False*)

Check the validity of *rtcclient.project_area.ProjectArea* id

Parameters

- **projectarea_id** – the *rtcclient.project_area.ProjectArea* id
- **(default is False)** (*archived*) – whether the project area is archived

Returns *True* or *False*

Return type bool

checkType(*item_type*, *projectarea_id*)

Check the validity of *rtcclient.workitem.Workitem* type

Parameters

- **item_type** – the type of the workitem (e.g. Story/Defect/Epic)
- **projectarea_id** – the *rtcclient.project_area.ProjectArea* id

Returns *True* or *False*

Return type bool

copyWorkitem(*copied_from*, *title=None*, *description=None*, *prefix=None*)
Create a workitem by copying from an existing one

Parameters

- **copied_from** – the to-be-copied workitem id
- **title** – the new workitem title/summary. If *None*, will copy that from a to-be-copied workitem
- **description** – the new workitem description. If *None*, will copy that from a to-be-copied workitem
- **prefix** – used to add a prefix to the copied title and description

Returns the *rtcclient.workitem.Workitem* object

Return type *rtcclient.workitem.Workitem*

createWorkitem(*item_type*, *title*, *description=None*, *projectarea_id=None*, *projectarea_name=None*, *template=None*, *copied_from=None*, *keep=False*, ***kwargs*)
Create a workitem

Parameters

- **item_type** – the type of the workitem (e.g. task/defect/issue)
- **title** – the title of the new created workitem
- **description** – the description of the new created workitem
- **projectarea_id** – the *rtcclient.project_area.ProjectArea* id
- **projectarea_name** – the project area name
- **template** – The template to render. The template is actually a file, which is usually generated by *rtcclient.template.Templater.getTemplate* and can also be modified by user accordingly.
- **copied_from** – the to-be-copied workitem id
- **keep** – refer to *keep* in *rtcclient.template.Templater.getTemplate*. Only works when *template* is not specified
- ****kwargs** – Optional/mandatory arguments when creating a new workitem. More details, please refer to *kwargs* in *rtcclient.template.Templater.render*

Returns the *rtcclient.workitem.Workitem* object

Return type *rtcclient.workitem.Workitem*

getFiledAgainst(*filedagainst_name*, *projectarea_id=None*, *projectarea_name=None*, *archived=False*)
Get *rtcclient.models.FiledAgainst* object by its name

Parameters

- **filedagainst_name** – the filedagainst name
- **projectarea_id** – the *rtcclient.project_area.ProjectArea* id
- **projectarea_name** – the project area name
- **(default is False)** (*archived*) – whether the filedagainst is archived

Returns the *rtcclient.models.FiledAgainst* object

Return type *rtcclient.models.FiledAgainst*

getFiledAgainsts (*projectarea_id=None*, *projectarea_name=None*, *archived=False*)
Get all *rtcclient.models.FiledAgainst* objects by project area id or name

If both *projectarea_id* and *projectarea_name* are *None*, all the filedagainsts in all project areas will be returned.

If no *rtcclient.models.FiledAgainst* objects are retrieved, *None* is returned.

Parameters

- **projectarea_id** – the *rtcclient.project_area.ProjectArea* id
- **projectarea_name** – the project area name
- **(default is False)** (*archived*) – whether the filedagainsts are archived

Returns a `list` that contains all the *rtcclient.models.FiledAgainst* objects

Return type list

getFoundIn (*foundin_name*, *projectarea_id=None*, *projectarea_name=None*, *archived=False*)
Get *rtcclient.models.FoundIn* object by its name

Parameters

- **foundin_name** – the foundin name
- **projectarea_id** – the *rtcclient.project_area.ProjectArea* id
- **projectarea_name** – the project area name
- **(default is False)** (*archived*) – whether the foundin is archived

Returns the *rtcclient.models.FoundIn* object

Return type *rtcclient.models.FoundIn*

getFoundIns (*projectarea_id=None*, *projectarea_name=None*, *archived=False*)
Get all *rtcclient.models.FoundIn* objects by project area id or name

If both *projectarea_id* and *projectarea_name* are *None*, all the foundins in all project areas will be returned.

If no *rtcclient.models.FoundIn* objects are retrieved, *None* is returned.

Parameters

- **projectarea_id** – the *rtcclient.project_area.ProjectArea* id
- **projectarea_name** – the project area name
- **(default is False)** (*archived*) – whether the foundins are archived

Returns a `list` that contains all the *rtcclient.models.FoundIn* objects

Return type list

getPlannedFor (*plannedfor_name*, *projectarea_id=None*, *projectarea_name=None*, *archived=False*, *returned_properties=None*)
Get *rtcclient.models.PlannedFor* object by its name

Parameters

- **plannedfor_name** – the plannedfor name
- **projectarea_id** – the *rtcclient.project_area.ProjectArea* id

- **projectarea_name** – the project area name
- **(default is False)** (*archived*) – whether the plannedfor is archived
- **returned_properties** – the returned properties that you want. Refer to `rtcclient.client.RTCClient` for more explanations

Returns the `rtcclient.models.PlannedFor` object

Return type *rtcclient.models.PlannedFor*

getPlannedFors(*projectarea_id=None*, *projectarea_name=None*, *archived=False*, *returned_properties=None*)

Get all `rtcclient.models.PlannedFor` objects by project area id or name

If both *projectarea_id* and *projectarea_name* are None, all the plannedfors in all project areas will be returned.

If no `rtcclient.models.PlannedFor` objecs are retrieved, *None* is returned.

Parameters

- **projectarea_id** – the `rtcclient.project_area.ProjectArea` id
- **projectarea_name** – the project area name
- **(default is False)** (*archived*) – whether the plannedfors are archived
- **returned_properties** – the returned properties that you want. Refer to `rtcclient.client.RTCClient` for more explanations

Returns a `list` that contains all the `rtcclient.models.PlannedFor` objects

Return type list

getPriorities(*projectarea_id=None*, *projectarea_name=None*)

Get all `rtcclient.models.Priority` objects by project area id or name

At least either of *projectarea_id* and *projectarea_name* is given.

If no `rtcclient.models.Priority` is retrieved, *None* is returned.

Parameters

- **projectarea_id** – the `rtcclient.project_area.ProjectArea` id
- **projectarea_name** – the project area name

Returns a `list` contains all the `rtcclient.models.Priority` objects

Return type list

getPriority(*priority_name*, *projectarea_id=None*, *projectarea_name=None*)

Get `rtcclient.models.Priority` object by its name

At least either of *projectarea_id* and *projectarea_name* is given

Parameters

- **priority_name** – the priority name
- **projectarea_id** – the `rtcclient.project_area.ProjectArea` id
- **projectarea_name** – the project area name

Returns the `rtcclient.models.Priority` object

Return type *rtcclient.models.Priority*

getProjectArea(*projectarea_name*, *archived=False*, *returned_properties=None*)
Get `rtcclient.project_area.ProjectArea` object by its name

Parameters

- **projectarea_name** – the project area name
- **(default is False)** (*archived*) – whether the project area is archived
- **returned_properties** – the returned properties that you want. Refer to `rtcclient.client.RTCClient` for more explanations

Returns the `rtcclient.project_area.ProjectArea` object

Return type *rtcclient.project_area.ProjectArea*

getProjectAreaByID(*projectarea_id*, *archived=False*, *returned_properties=None*)
Get `rtcclient.project_area.ProjectArea` object by its id

Parameters

- **projectarea_id** – the `rtcclient.project_area.ProjectArea` id
- **(default is False)** (*archived*) – whether the project area is archived
- **returned_properties** – the returned properties that you want. Refer to `rtcclient.client.RTCClient` for more explanations

Returns the `rtcclient.project_area.ProjectArea` object

Return type *rtcclient.project_area.ProjectArea*

getProjectAreaID(*projectarea_name*, *archived=False*)
Get `rtcclient.project_area.ProjectArea` id by its name

Parameters

- **projectarea_name** – the project area name
- **(default is False)** (*archived*) – whether the project area is archived

Returns the `string` object

Return type string

getProjectAreaIDs(*projectarea_name=None*, *archived=False*)
Get all `rtcclient.project_area.ProjectArea` id(s) by project area name

If *projectarea_name* is *None*, all the `rtcclient.project_area.ProjectArea` id(s) will be returned.

Parameters

- **projectarea_name** – the project area name
- **(default is False)** (*archived*) – whether the project area is archived

Returns a `list` that contains all the `ProjectArea` ids

Return type list

getProjectAreas(*archived=False*, *returned_properties=None*)
Get all `rtcclient.project_area.ProjectArea` objects

If no `rtcclient.project_area.ProjectArea` objects are retrieved, *None* is returned.

Parameters

- **(default is False)** (*archived*) – whether the project area is archived

- **returned_properties** – the returned properties that you want. Refer to *rtcclient.client.RTCClient* for more explanations

Returns A `list` that contains all the *rtcclient.project_area.ProjectArea* objects

Return type list

getSeverities(*projectarea_id=None*, *projectarea_name=None*)

Get all *rtcclient.models.Severity* objects by project area id or name

At least either of *projectarea_id* and *projectarea_name* is given

If no *rtcclient.models.Severity* is retrieved, *None* is returned.

Parameters

- **projectarea_id** – the *rtcclient.project_area.ProjectArea* id
- **projectarea_name** – the project area name

Returns a `list` that contains all the *rtcclient.models.Severity* objects

Return type list

getSeverity(*severity_name*, *projectarea_id=None*, *projectarea_name=None*)

Get *rtcclient.models.Severity* object by its name

At least either of *projectarea_id* and *projectarea_name* is given

Parameters

- **severity_name** – the severity name
- **projectarea_id** – the *rtcclient.project_area.ProjectArea* id
- **projectarea_name** – the project area name

Returns the *rtcclient.models.Severity* object

Return type *rtcclient.models.Severity*

getTeamArea(*teamarea_name*, *projectarea_id=None*, *projectarea_name=None*, *archived=False*, *returned_properties=None*)

Get *rtcclient.models.TeamArea* object by its name

If *projectarea_id* or *projectarea_name* is specified, then the matched *rtcclient.models.TeamArea* in that project area will be returned. Otherwise, only return the first found *rtcclient.models.TeamArea* with that name.

Parameters

- **teamarea_name** – the team area name
- **projectarea_id** – the *rtcclient.project_area.ProjectArea* id
- **projectarea_name** – the project area name
- **(default is False)** (*archived*) – whether the team area is archived
- **returned_properties** – the returned properties that you want. Refer to *rtcclient.client.RTCClient* for more explanations

Returns the *rtcclient.models.TeamArea* object

Return type *rtcclient.models.TeamArea*

getTeamAreas(*projectarea_id=None*, *projectarea_name=None*, *archived=False*, *returned_properties=None*)

Get all *rtcclient.models.TeamArea* objects by project area id or name

If both *projectarea_id* and *projectarea_name* are *None*, all team areas in all project areas will be returned.

If no *rtcclient.models.TeamArea* objects are retrieved, *None* is returned.

Parameters

- **projectarea_id** – the *rtcclient.project_area.ProjectArea* id
- **projectarea_name** – the project area name
- **(default is False)** (*archived*) – whether the team areas are archived
- **returned_properties** – the returned properties that you want. Refer to *rtcclient.client.RTCClient* for more explanations

Returns a `list` that contains all the *rtcclient.models.TeamArea* objects

Return type list

getTemplate(*copied_from*, *template_name=None*, *template_folder=None*, *keep=False*, *encoding='UTF-8'*)

Get template from some to-be-copied workitems

More details, please refer to *rtcclient.template.Templater.getTemplate*

getTemplates(*workitems*, *template_folder=None*, *template_names=None*, *keep=False*, *encoding='UTF-8'*)

Get templates from a group of to-be-copied workitems and write them to files named after the names in *template_names* respectively.

More details, please refer to *rtcclient.template.Templater.getTemplates*

getWorkitem(*workitem_id*, *returned_properties=None*)

Get *rtcclient.workitem.Workitem* object by its id/number

Parameters

- **workitem_id** – the workitem id/number (integer or equivalent string)
- **returned_properties** – the returned properties that you want. Refer to *rtcclient.client.RTCClient* for more explanations

Returns the *rtcclient.workitem.Workitem* object

Return type *rtcclient.workitem.Workitem*

getWorkitems(*projectarea_id=None*, *projectarea_name=None*, *returned_properties=None*, *archived=False*)

Get all *rtcclient.workitem.Workitem* objects by project area id or name

If both *projectarea_id* and *projectarea_name* are *None*, all the workitems in all project areas will be returned.

If no *rtcclient.workitem.Workitem* objects are retrieved, *None* is returned.

You can also customize your preferred properties to be returned by specified *returned_properties*

Parameters

- **projectarea_id** – the *rtcclient.project_area.ProjectArea* id
- **projectarea_name** – the project area name

- **returned_properties** – the returned properties that you want. Refer to *rtcclient.client.RTCClient* for more explanations
- **(default is False)** (*archived*) – whether the workitems are archived

Returns a `list` that contains all the *rtcclient.workitem.Workitem* objects

Return type list

listFields(*template*)
List all the attributes to be rendered from the template file

Parameters **template** – The template to render. The template is actually a file, which is usually generated by *rtcclient.template.Templater.getTemplate* and can also be modified by user accordingly.

Returns a `set` that contains all the needed attributes

Return type set

More details, please refer to *rtcclient.template.Templater.listFieldsFromWorkitem*

listFieldsFromWorkitem(*copied_from*, *keep=False*)
List all the attributes to be rendered directly from some to-be-copied workitems

More details, please refer to *rtcclient.template.Templater.listFieldsFromWorkitem*

queryWorkitems(*query_str*, *projectarea_id=None*, *projectarea_name=None*, *returned_properties=None*, *archived=False*)
Query workitems with the query string in a certain project area

At least either of *projectarea_id* and *projectarea_name* is given

Parameters

- **query_str** – a valid query string
- **projectarea_id** – the *rtcclient.project_area.ProjectArea* id
- **projectarea_name** – the project area name
- **returned_properties** – the returned properties that you want. Refer to *rtcclient.client.RTCClient* for more explanations
- **(default is False)** (*archived*) – whether the workitems are archived

Returns a `list` that contains the queried *rtcclient.workitem.Workitem* objects

Return type list

3.2 ProjectArea

class rtcclient.project_area.**ProjectArea**(*url*, *rtc_obj*, *raw_data*)
A wrapped class to perform all the operations in a Project Area

Parameters

- **url** – the project area url
- **rtc_obj** – a reference to the *rtcclient.client.RTCClient* object
- **raw_data** – the raw data (OrderedDict) of the request response

getAdministrator(*email*, *returned_properties=None*)
Get the *rtcclient.models.Administrator* object by the email address

Parameters

- **email** – the email address (e.g. somebody@gmail.com)
- **returned_properties** – the returned properties that you want. Refer to *rtcclient.client.RTCClient* for more explanations

Returns the *rtcclient.models.Administrator* object

Return type *rtcclient.models.Administrator*

getAdministrators(*returned_properties=None*)

Get all the *rtcclient.models.Administrator* objects in this project area

If no Administrators are retrieved, *None* is returned.

Parameters **returned_properties** – the returned properties that you want. Refer to *rtcclient.client.RTCClient* for more explanations

Returns a list that contains all *rtcclient.models.Administrator* objects

Return type list

getItemType(*title*, *returned_properties=None*)

Get the *rtcclient.models.ItemType* object by the title

Parameters

- **title** – the title (e.g. Story/Epic/..)
- **returned_properties** – the returned properties that you want. Refer to *rtcclient.client.RTCClient* for more explanations

Returns the *rtcclient.models.ItemType* object

Return type *rtcclient.models.ItemType*

getItemTypes(*returned_properties=None*)

Get all the *rtcclient.models.ItemType* objects in this project area

If no ItemTypes are retrieved, *None* is returned.

Parameters **returned_properties** – the returned properties that you want. Refer to *rtcclient.client.RTCClient* for more explanations

Returns a list that contains all *rtcclient.models.ItemType* objects

Return type list

getMember(*email*, *returned_properties=None*)

Get the *rtcclient.models.Member* object by the email address

Parameters

- **email** – the email address (e.g. somebody@gmail.com)
- **returned_properties** – the returned properties that you want. Refer to *rtcclient.client.RTCClient* for more explanations

Returns the *rtcclient.models.Member* object

Return type *rtcclient.models.Member*

getMembers(*returned_properties=None*)

Get all the *rtcclient.models.Member* objects in this project area

If no Members are retrieved, *None* is returned.

Parameters **returned_properties** – the returned properties that you want. Refer to *rtcclient.client.RTCClient* for more explanations

Returns a list that contains all *rtcclient.models.Member* objects

Return type list

getRole(*label*)
Get the *rtcclient.models.Role* object by the label name

Parameters **label** – the label name of the role

Returns the *rtcclient.models.Role* object

Return type *rtcclient.models.Role*

getRoles()
Get all *rtcclient.models.Role* objects in this project area

If no Roles are retrieved, *None* is returned.

Returns a list that contains all *rtcclient.models.Role* objects

Return type list

3.3 Workitem

class rtcclient.workitem.**Workitem**(*url*, *rtc_obj*, *workitem_id=None*, *raw_data=None*)
A wrapped class for managing all related resources of the workitem

Parameters

- **url** – the workitem url
- **rtc_obj** – a reference to the *rtcclient.client.RTCClient* object
- **(default is None)** (*workitem_id*) – the id of the workitem, which will be retrieved if not specified
- **raw_data** – the raw data (OrderedDict) of the request response

addComment(*msg=None*)
Add a comment to this workitem

Parameters **msg** – comment message

Returns the *rtcclient.models.Comment* object

Return type *rtcclient.models.Comment*

addSubscriber(*email*)
Add a subscriber to this workitem

If the subscriber has already been added, no more actions will be performed.

Parameters **email** – the subscriber's email

addSubscribers(*emails_list*)
Add subscribers to this workitem

If the subscribers have already been added, no more actions will be performed.

Parameters **emails_list** – a list/tuple/set contains the the subscribers' emails

getAction(*action_name*)
Get the *rtcclient.models.Action* object by its name

Parameters **action_name** – the name/title of the action

Returns the *rtcclient.models.Action* object

Return type *rtcclient.models.Action*

getActions()
Get all *rtcclient.models.Action* objects of this workitem

Returns a list contains all the *rtcclient.models.Action* objects

Return type list

getCommentByID(*comment_id*)
Get the *rtcclient.models.Comment* object by its id

Note: the comment id starts from 0

Parameters **comment_id** – the comment id (integer or equivalent string)

Returns the *rtcclient.models.Comment* object

Return type *rtcclient.models.Comment*

getComments()
Get all *rtcclient.models.Comment* objects in this workitem

Returns a list contains all the *rtcclient.models.Comment* objects

Return type list

getStates()
Get all *rtcclient.models.State* objects of this workitem

Returns a list contains all the *rtcclient.models.State* objects

Return type list

getSubscribers()
Get subscribers of this workitem

Returns a list contains all the *rtcclient.models.Member* objects

Return type list

removeSubscriber(*email*)
Remove a subscriber from this workitem

If the subscriber has not been added, no more actions will be performed.

Parameters **email** – the subscriber's email

removeSubscribers(*emails_list*)
Remove subscribers from this workitem

If the subscribers have not been added, no more actions will be performed.

Parameters **emails_list** – a list/tuple/set contains the the subscribers' emails

3.4 Query

class rtcclient.query.**Query**(*rtc_obj*)

A wrapped class to perform all query-related actions

Parameters **rtc_obj** – a reference to the *rtcclient.client.RTCClient* object

getAllSavedQueries(*projectarea_id=None*, *projectarea_name=None*, *creator=None*, *saved_query_name=None*)

Get all saved queries created by somebody (optional) in a certain project area (optional, either *projectarea_id* or *projectarea_name* is needed if specified)

If *saved_query_name* is specified, only the saved queries match the name will be fetched.

Note: only if *creator* is added as a member, the saved queries can be found. Otherwise None will be returned.

WARNING: now the RTC server cannot correctly list all the saved queries It seems to be a bug of RTC. Recommend using *runSavedQueryByUrl* to query all the workitems if the query is saved.

Note: It will run faster when more attributes are specified.

Parameters

- **projectarea_id** – the *rtcclient.project_area.ProjectArea* id
- **projectarea_name** – the *rtcclient.project_area.ProjectArea* name
- **creator** – the creator email address
- **saved_query_name** – the saved query name

Returns a `list` that contains the saved queried `rtcclient.models.SavedQuery` objects

Return type list

getMySavedQueries(*projectarea_id=None*, *projectarea_name=None*, *saved_query_name=None*)

Get all saved queries created by me in a certain project area (optional, either *projectarea_id* or *projectarea_name* is needed if specified)

Note: only if myself is added as a member, the saved queries can be found. Otherwise None will be returned.

WARNING: now the RTC server cannot correctly list all the saved queries It seems to be a bug of RTC. Recommend using *runSavedQueryByUrl* to query all the workitems if the query is saved.

Parameters

- **projectarea_id** – the *rtcclient.project_area.ProjectArea* id
- **projectarea_name** – the *rtcclient.project_area.ProjectArea* name
- **saved_query_name** – the saved query name

Returns a `list` that contains the saved queried `rtcclient.models.SavedQuery` objects

Return type list

getSavedQueriesByName(*saved_query_name*, *projectarea_id=None*, *projectarea_name=None*, *creator=None*)

Get all saved queries match the name created by somebody (optional) in a certain project area (optional, either *projectarea_id* or *projectarea_name* is needed if specified)

Note: only if *creator* is added as a member, the saved queries can be found. Otherwise None will be returned.

WARNING: now the RTC server cannot correctly list all the saved queries It seems to be a bug of RTC. Recommend using *runSavedQueryByUrl* to query all the workitems if the query is saved.

Parameters

- **saved_query_name** – the saved query name
- **projectarea_id** – the *rtcclient.project_area.ProjectArea* id
- **projectarea_name** – the *rtcclient.project_area.ProjectArea* name
- **creator** – the creator email address

Returns a `list` that contains the saved queried `rtcclient.models.SavedQuery` objects

Return type list

queryWorkitems(*query_str*, *projectarea_id=None*, *projectarea_name=None*, *returned_properties=None*, *archived=False*)

Query workitems with the query string in a certain *rtcclient.project_area.ProjectArea*

At least either of *projectarea_id* and *projectarea_name* is given

Parameters

- **query_str** – a valid query string
- **projectarea_id** – the *rtcclient.project_area.ProjectArea* id
- **projectarea_name** – the *rtcclient.project_area.ProjectArea* name
- **returned_properties** – the returned properties that you want. Refer to *rtcclient.client.RTCClient* for more explanations
- **(default is False)** (*archived*) – whether the *rtcclient.workitem.Workitem* is archived

Returns a `list` that contains the queried *rtcclient.workitem.Workitem* objects

Return type list

runSavedQuery(*saved_query_obj*, *returned_properties=None*)

Query workitems using the `rtcclient.models.SavedQuery` object

Parameters

- **saved_query_obj** – the `rtcclient.models.SavedQuery` object
- **returned_properties** – the returned properties that you want. Refer to *rtcclient.client.RTCClient* for more explanations

Returns a `list` that contains the queried *rtcclient.workitem.Workitem* objects

Return type list

runSavedQueryByUrl(*saved_query_url*, *returned_properties=None*)

Query workitems using the saved query url

Parameters

- **saved_query_url** – the saved query url
- **returned_properties** – the returned properties that you want. Refer to *rtcclient.client.RTCClient* for more explanations

Returns a `list` that contains the queried *rtcclient.workitem.Workitem* objects

Return type list

3.5 Template

class `rtcclient.template.`**Templater**(*rtc_obj*, *searchpath=None*)

A wrapped class used to generate and render templates from some copied workitems

Parameters

- **rtc_obj** – a reference to the *rtcclient.client.RTCClient* object
- **searchpath** – the folder to store your templates. If *None*, the default search path (/your/site-packages/rtcclient/templates) will be loaded automatically.

getTemplate(*copied_from*, *template_name=None*, *template_folder=None*, *keep=False*, *encoding='UTF-8'*)

Get template from some to-be-copied *rtcclient.workitem.Workitem*

The resulting XML document is returned as a `string`, but if *template_name* (a string value) is specified, it is written there instead.

Parameters

- **copied_from** – the to-be-copied *rtcclient.workitem.Workitem* id (integer or equivalent string)
- **template_name** – the template file name
- **template_folder** – the folder to store template file
- **(default is False)** (*keep*) – If *True*, some of below parameters (which may not be included in some customized *rtcclient.workitem.Workitem* type) will remain unchangeable with the to-be-copied *rtcclient.workitem.Workitem*. Otherwise for *False*.
  - teamArea (Team Area)
  - ownedBy (Owned By)
  - plannedFor(Planned For)
  - severity(Severity)
  - priority(Priority)
  - filedAgainst(Filed Against)
- **(default is "UTF-8")** (*encoding*) – coding format

Returns

- a `string` object: if *template_name* is not specified
- write the template to file *template_name*: if *template_name* is specified

getTemplates(*workitems*, *template_folder=None*, *template_names=None*, *keep=False*, *encoding='UTF-8'*)

Get templates from a group of to-be-copied `Workitems` and write them to files named after the names in *template_names* respectively.

Parameters

- **workitems** – a list/tuple/set contains the ids (integer or equivalent string) of some to-be-copied Workitems
- **template_names** – a list/tuple/set contains the template file names for copied Workitems. If *None*, the new template files will be named after the *rtcclient.workitem.Workitem* id with "*.template*" as a postfix
- **template_folder** – refer to *rtcclient.template.Templater.getTemplate*
- **(default is False)** (*keep*) – refer to *rtcclient.template.Templater.getTemplate*
- **(default is "UTF-8")** (*encoding*) – refer to *rtcclient.template.Templater.getTemplate*

listFields(*template*)

List all the attributes to be rendered from the template file

Parameters **template** – The template to render. The template is actually a file, which is usually generated by *rtcclient.template.Templater.getTemplate* and can also be modified by user accordingly.

Returns a set contains all the needed attributes

Return type set

listFieldsFromSource(*template_source*)

List all the attributes to be rendered directly from template source

Parameters **template_source** – the template source (usually represents the template content in string format)

Returns a set contains all the needed attributes

Return type set

listFieldsFromWorkitem(*copied_from*, *keep=False*)

List all the attributes to be rendered directly from some to-be-copied *rtcclient.workitem.Workitem*

Parameters

- **copied_from** – the to-be-copied *rtcclient.workitem.Workitem* id
- **(default is False)** (*keep*) – If *True*, some of below parameters (which will not be included in some customized *rtcclient.workitem.Workitem* type) will remain unchangeable with the to-be-copied *rtcclient.workitem.Workitem*. Otherwise for *False*.
  - teamArea (Team Area)
  - ownedBy (Owned By)
  - plannedFor(Planned For)
  - severity(Severity)
  - priority(Priority)
  - filedAgainst(Filed Against)

Returns a set contains all the needed attributes

Return type set

render(*template*, ***kwargs*)

Renders the template

Parameters

- **template** – The template to render. The template is actually a file, which is usually generated by *rtcclient.template.Templater.getTemplate* and can also be modified by user accordingly.
- **kwargs** – The *kwargs* dict is used to fill the template. These two parameter are mandatory:
  - description
  - title

  Some of below parameters (which may not be included in some customized workitem type) are mandatory if *keep* (parameter in *rtcclient.template.Templater.getTemplate*) is set to *False*; Optional for otherwise.
  - teamArea (Team Area)
  - ownedBy (Owned By)
  - plannedFor(Planned For)
  - severity(Severity)
  - priority(Priority)
  - filedAgainst(Filed Against)

  Actually all these needed keywords/attributes/fields can be retrieved by *rtcclient.template.Templater.listFields*

Returns the `string` object

Return type string

renderFromWorkitem(*copied_from*, *keep=False*, *encoding='UTF-8'*, ***kwargs*)

Render the template directly from some to-be-copied *rtcclient.workitem.Workitem* without saving to a file

Parameters

- **copied_from** – the to-be-copied *rtcclient.workitem.Workitem* id
- **(default is False)** (*keep*) – If *True*, some of the below fields will remain unchangeable with the to-be-copied *rtcclient.workitem.Workitem*. Otherwise for *False*.
  - teamArea (Team Area)
  - ownedBy (Owned By)
  - plannedFor(Planned For)
  - severity(Severity)
  - priority(Priority)
  - filedAgainst(Filed Against)
- **(default is "UTF-8")** (*encoding*) – coding format
- **kwargs** – The *kwargs* dict is used to fill the template. These two parameter are mandatory:
  - description

– title

Some of below parameters (which may not be included in some customized workitem type) are mandatory if *keep* is set to *False*; Optional for otherwise.

– teamArea (Team Area)

– ownedBy (Owned By)

– plannedFor(Planned For)

– severity(Severity)

– priority(Priority)

– filedAgainst(Filed Against)

Actually all these needed keywords/attributes/fields can be retrieved by *rtcclient.template.Templater.listFieldsFromWorkitem*

Returns the `string` object

Return type string

3.6 Models

class `rtcclient.models.`**`Role`**(*url*, *rtc_obj*, *raw_data=None*)
The role in the project area or team area

class `rtcclient.models.`**`Member`**(*url*, *rtc_obj*, *raw_data=None*)
The member in the project area

class `rtcclient.models.`**`Administrator`**(*url*, *rtc_obj*, *raw_data=None*)
The administrator of the project area

class `rtcclient.models.`**`ItemType`**(*url*, *rtc_obj*, *raw_data=None*)
The workitem type

class `rtcclient.models.`**`TeamArea`**(*url*, *rtc_obj*, *raw_data=None*)
The team area

class `rtcclient.models.`**`PlannedFor`**(*url*, *rtc_obj*, *raw_data=None*)
The project plannedfor defines a start and end date along with an iteration breakdown

class `rtcclient.models.`**`FiledAgainst`**(*url*, *rtc_obj*, *raw_data=None*)
Category that identifies the component or functional area that the work item belongs to.

class `rtcclient.models.`**`FoundIn`**(*url*, *rtc_obj*, *raw_data=None*)
Release in which the issue described in the work item was identified.

class `rtcclient.models.`**`Severity`**(*url*, *rtc_obj*, *raw_data=None*)
Indication of the impact of the work item

class `rtcclient.models.`**`Priority`**(*url*, *rtc_obj*, *raw_data=None*)
Ranked importance of a work item

class `rtcclient.models.`**`Action`**(*url*, *rtc_obj*, *raw_data=None*)
The action to change the state of the workitem

class `rtcclient.models.`**`State`**(*url*, *rtc_obj*, *raw_data=None*)
Status of the work item. For example, New, In Progress, or Resolved.

class rtcclient.models.**Comment**(*url*, *rtc_obj*, *raw_data=None*)
Comment about the work item

class **SavedQuery**(*url*, *rtc_obj*, *raw_data=None*)
User saved query

Members

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

A

C

F

G

I

L

M

P

Q

R

S

T

W