JasperServer-Library Documentation

Release 0.0.1

Christophe GRANJOUAN

Sep 27, 2017

Contents

1 QuickStart **1**
- 1.1 Opening a session . . . 1
- 1.2 Searching an existent User or Role . . . 1
- 1.3 Searching Resources . . . 2
- 1.4 Create, Modify or Delete a Content . . . 2
- 1.5 The ReportUnit . . . 3
- 1.6 Resources Synchronization . . . 3

2 Developer Interface **5**
- 2.1 REST Authentification . . . 5
- 2.2 Administration service . . . 5
- 2.3 Resources service . . . 6
- 2.4 Resource service . . . 6
- 2.5 Report service . . . 6
- 2.6 Synchronization . . . 6

3 Indices and tables **7**

Python Module Index **9**

CHAPTER 1

QuickStart

JasperServer Library use REST protocol to communicate with JasperServer for reports or BI. This module defines classes which implement the HTTP methods (PUT, POST, DELETE, GET) and allow use basic services in JasperServer.

First, make sure that the Requests library is installed

```
>>> pip install requests
```

Let's get started with some examples

Opening a session

To make an HTTP request in JasperReport Server, you must be connected to JRS

```
>>> from jasperserver.rest import Client
>>> client = Client('http://localhost:8080/jasperserver', 'jasperadmin', 'jasperadmin
↪')
```

You have a Client object. Now, we can get all REST method from this open session.

Searching an existent User or Role

You can browse for all user

```
>>> from jasperserver.admin import User, Role
>>> users = User(client).search()
>>> users
[{'username': 'anonymousUser', 'fullName': 'anonymousUser', 'enabled': 'true', 'roles
↪': ['ROLE_ANONYMOUS']}, ...
```

Or search by terms

```
>>> users = User(client).search('joe')
>>> users
[{'username': 'joeuser', 'fullName': 'Joe User', 'enabled': 'true', 'roles': ['ROLE_
↪USER']}]
```

and you can do the same with the Role service

```
>>> role = Role(client).search()
>>> role
['ROLE_ADMINISTRATOR', 'ROLE_ANONYMOUS', 'ROLE_USER']
```

Searching Resources

If you want listed all existent resources in JRS in a specified path, you could follow this example :

```
>>> from jasperserver.services import Resources
>>> rs = Resources(client, '/openerp/bases/openerp_demo').search('Product')
>>> rs
[{'wsType': 'jrxml', 'uriString': '/openerp/reports/ProductCategory', 'name':
↪'ProductCategory'}, {'wsType':...
```

For each found resources, you'll obtain his type, uri and name

Create, Modify or Delete a Content

To create or modify you must instanciate a Resource object with the working directory. Keep in mind, you need sufficient permission to send this following requests.

```
>>> from jasperserver.services import Resource
>>> jrxml = Resource(client, '/openerp/bases/reports')
```

And send your query (with or without attached file) to JRS like this :

```
>>> srcfile_path = '/the/local/file/resource/path/'
>>> resource_name = 'myresource'
>>> rtype = 'jrxml'
>>> jrxml.create(resource_name, rtype, path_fileresource=srcfile_path)
```

To modify it (eventually !):

```
>>> jrxml.modify(resource_name, rtype, path_fileresource=srcfile_path)
```

Ah, you don't need it anymore :

```
>>> jrxml.delete(resource_name)
```

If your resource is a reference to another one, you won't be able to delete it.

The ReportUnit

Report Unit is a more complicated resource in wich there a several resource as datasources and jrxml.

So, to create a report unit just modify some informations to add it, as a datasource corresponding to jdbc source and the jrxml resource wich is now in JRS.

```
>>> reportunit = Resource(client, '/openerp/bases/openerp_demo')

>>> rtype = 'reportUnit'
>>> resource_name = 'myreport'
>>> datasource = '/datasources/openerp_demo'
>>> jrxmlsource = '/openerp/bases/reports/myresource'

>>> reportunit.create(resource_name, rtype,  uri_datasource=datasource, uri_
↪jrxmlfile=jrxmlsource)
```

Maybe, you could need run it :

```
>>> from jasperserver.services import Report
>>> report = Report(client, '/openerp/bases/openerp_demo')
>>> report.run('myreport')
```

It will return a binary data stream of a pdf file by default. Just write it in a file. But, you can export the report in XLS :

```
>>> report.run('myreport', output_format='xls')
```

JRS can export report in different output format. Please read the web service documentation of JRS to know all supported format.

Resources Synchronization

This service allow synchronization of reports, and subreports on JRS. Keep in mind, your local path is the master.

To use synchronisation, you could follow this example :

```
>>> # -*- coding: utf-8 -*-
>>> import sys
>>> from jasperserver.rest import Client
>>> from jasperserver.admin import User, Role
>>> from jasperserver.synchronization import SyncResources
>>> from jasperserver.services import Report, Resource, Resources
>>> from jasperserver.exceptions import *
>>>
>>> try:
>>>     client = Client('http://localhost:8080/jasperserver', 'jasperadmin',
↪'jasperadmin')

>>> except JsException:
>>>     print 'Error Authentification FAIL!'
>>> sys.exit(1)
>>>
>>> path_mainjrxml = '/my/local/source/path/mainjrxml/'
>>> path_subjrxml  = '/my/local/source/path/subjrxml/'
>>>
>>> path_js_jrxmlresource = '/my/jrs/mainjrxml/destination/'
```

```
>>> path_js_subjrxmlresource = '/my/jrs/subjrxml/destination/'
>>> path_reportUnit = '/my/jrs/reportunit/destination/'
>>>
>>> sync = SyncResources(client)
>>>
>>> sync.update_subreports(path_subjrxml)
>>> sync.update_mainreports(path_mainjrxml)
```

CHAPTER 2

Developer Interface

This part of the documentation covers all class and methods of the module

REST Authentification

When using web services, the calling application must provide a valid user ID and password to JasperReports Server. The special login service that allows authentication using a POST request to create a session and return a session ID that is used with subsequent requests.

class `jasperserver.rest.`**`Client`**(*url*, *username='jasperadmin'*, *password='jasperadmin'*)
Create a REST connection, with authentification This class implements Login service in JasperServer using the session cookie and the RESTful interface.

Administration service

The web services for administration consists to administers users and roles for searching, editing, deleting and creating. Only administrative users may access these REST services.

class `jasperserver.admin.`**`Role`**(*js_connect*)
The role service allows administrators to view, create, edit, and delete role definitions. However, the role service does not define role membership

`create`(*rolename*)
Create a new role

`delete`(*rolename*)
Delete an existent role, if not found return 404 not found

`modify`(*rolename*)
Modify an existent role

search (*query=''*)
The Search method for the role service returns a list of roles that match the search string. Without query, all roles are listed.

class jasperserver.admin.**User** (*js_connect*)
Manage user inside the JasperServer

create (*name, login, password, roles=['ROLE_USER']*)
Create a new user, if exists it return status 403

delete (*login*)
Delete user with the specied login.

modify (*name, login, password, roles=['ROLE_USER']*)
Modify an existent user, if not found return 404 not found

search (*query=''*)
The GET method for the user service returns descriptors for all users that match the search string

Resources service

This service lets you browse or search the repository in JasperServer. The resources service is a read only service.

Resource service

The resource service supports several HTTP methods to view, create, and modify resources in the repository.

Report service

This service simplifies the API for obtaining report output such as PDF or XLS.

Synchronization

Synchronization allow update (create, modify, delete) all local JRXML files into JRXML Resource and report unit Resource to JasperServer.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

j

jasperserver.admin, 5
jasperserver.rest, 5

Index

C

Client (class in jasperserver.rest), 5
create() (jasperserver.admin.Role method), 5
create() (jasperserver.admin.User method), 6

D

delete() (jasperserver.admin.Role method), 5
delete() (jasperserver.admin.User method), 6

J

jasperserver.admin (module), 4, 5
jasperserver.rest (module), 4, 5

M

modify() (jasperserver.admin.Role method), 5
modify() (jasperserver.admin.User method), 6

R

Role (class in jasperserver.admin), 5

S

search() (jasperserver.admin.Role method), 5
search() (jasperserver.admin.User method), 6

U

User (class in jasperserver.admin), 6