
python-sjsclient Documentation

Release

Spark Jobserver Python Client

December 03, 2018

1	Features	3
2	Library Installation	5
3	Getting started	7
4	Documentation	9
5	Discussion list	11
6	Requirements	13
7	License	15
8	Source code	17
8.1	Client API Reference	17
9	Indices and tables	21

Python bindings to Spark Job Server API.

Features

- Supports Spark Jobserver 0.6.0+

Library Installation

```
$ pip install python-sjsclient
```

Getting started

First create a client instance:

```
>>> from sjsclient import client
>>> sjs = client.Client("http://JOB_SERVER_URL:PORT")
```

Uploading a jar to Spark Jobserver:

```
>>> jar_file_path = os.path.join("path", "to", "jar")
>>> jar_blob = open(jar_file_path, 'rb').read()
>>> app = sjs.apps.create("test_app", jar_blob)
```

Uploading a python egg to Spark Jobserver:

```
>>> from sjsclient import app
>>> egg_file_path = os.path.join("path", "to", "egg")
>>> egg_blob = open(egg_file_path, 'rb').read()
>>> app = sjs.apps.create("test_python_app", egg_blob, app.AppType.PYTHON)
```

Listing available apps:

```
>>> for app in sjs.apps.list():
...     print app.name
...
test_app
my_streaming_app
```

Creating an adhoc job:

```
>>> test_app = sjs.apps.get("test_app")
>>> class_path = "spark.jobserver.VeryShortDoubleJob"
>>> config = {"test_config": "test_config_value"}
>>> job = sjs.jobs.create(test_app, class_path, conf=config)
>>> print("Job Status: ", job.status)
Job Status: STARTED
```

Polling for job status:

```
>>> job = sjs.jobs.create(...)
>>> while job.status != "FINISHED":
>>>     time.sleep(2)
>>>     job = sjs.jobs.get(job.jobId)
```

Getting job config:

```
>>> config = {"test_config": "test_config_value"}
>>> job = sjs.jobs.create(test_app, class_path, conf=config)
>>> job_config = job.get_config()
>>> print("test_config value: ", job_config["test_config"])
test_config_value: test_config_value
```

Listing jobs:

```
>>> for job in sjs.jobs.list():
...     print job.jobId
...
8c5bd52f-6486-44ee-9ac3-a8327ee40494
24b67573-3115-49c7-983c-d0eff0499b71
99c8be9e-a0ec-42dd-8a2c-9a8680bc5051
bb82f712-d4b4-43a4-8e4d-e4bb272e85db
```

Limiting jobs list:

```
>>> for job in sjs.jobs.list(limit=1):
...     print job.jobId
...
8c5bd52f-6486-44ee-9ac3-a8327ee40494
```

Creating a named context:

```
>>> ctx_config = {'num-cpu-cores': '1', 'memory-per-node': '512m'}
>>> ctx = sjs.contexts.create("test_context", ctx_config)
```

Running a job in a named context:

```
>>> test_app = sjs.apps.get("test_app")
>>> test_ctx = sjs.contexts.get("test_context")
>>> config = {"test_config": "test_config_value"}
>>> job = sjs.jobs.create(test_app, class_path, ctx=test_ctx, conf=config)
>>> print("Job Status: ", job.status)
Job Status: STARTED
```

Documentation

<http://python-sjsclient.readthedocs.org>

Discussion list

spark-jobserver google group: <https://groups.google.com/forum/#!forum/spark-jobserver>

Requirements

- Python $\geq$ 2.7.0

License

`python-sjsclient` is offered under the Apache 2 license.

Source code

The latest developer version is available in a github repository: <https://github.com/spark-jobserver/python-sjsclient>

Contents:

Client API Reference

Client

```
class sjsclient.client.Client (endpoint, auth=None)
```

Bases: object

Client for Spark Job Server

App

```
class sjsclient.app.App (manager, attrs=None)
```

Bases: sjsclient.base.Resource

An app is a spark application.

name = None

Name of the App

time = None

App creation time

```
class sjsclient.app.AppManager (client)
```

Bases: sjsclient.base.ResourceManager

Manage App resources.

base_path = 'binaries'

```
create (name, app_binary, app_type='java')
```

Create an app.

Parameters

- **name** – Descriptive name of application
- **app_binary** – Application binary
- **app_type** – App type, for example java or python, default: java

Return type App

delete (*name*)

Delete a specific App.

Parameters **name** – The name of the App to delete.

get (*name*)

Get a specific App.

Parameters **name** – The name of the App to get.

Return type App

list ()

Lists Apps.

resource_class

alias of App

class sjsclient.app.**AppType**

Bases: object

A helper class that contains app types

JAVA = 'java'

PYTHON = 'python'

static get_header (*app_type*)

Job

class sjsclient.job.**Job** (*manager, attrs=None*)

Bases: sjsclient.base.Resource

A Spark job.

classpath = None

Main java class path

context = None

Context name

delete ()

Delete job.

duration = None

Time taken by the job to finish

get_config ()

Get job configuration.

jobId = None

Job ID

result = None

Response from Spark.

status = None

Jobs status

```

class sjsclient.job.JobConfig
    Bases: dict

    A Spark job config dictionary.

class sjsclient.job.JobManager(client)
    Bases: sjsclient.base.ResourceManager

    Manage Job resources.

    base_path = 'jobs'

    create(app, class_path, conf=None, ctx=None, sync=False)
        Create a Spark job.

        Parameters
        • app – Instance of App
        • class_path – Main class path of spark job.
        • conf – Configuration json
        • ctx – Instance of Context
        • sync – Set to True for synchronous job creation

        Return type Job

    delete(job_id)
        Delete a specific Job.

        Parameters job_id – The jobId of the Job to get.

    get(job_id)
        Get a specific Job. This returns more information than create.

        Parameters job_id – The jobId of the Job to get.

        Return type Job

    get_config(job_id)
        Get job configuration.

        Parameters job_id – The jobId of the Job to get.

        Return type JobConfig

    resource_class
        alias of Job

class sjsclient.job.JobStatus
    Bases: object

    A Helper class that contains the job status

    ERROR = 'ERROR'

    FINISHED = 'FINISHED'

    RUNNING = 'RUNNING'

```

Context

```

class sjsclient.context.Context(manager, attrs=None)
    Bases: sjsclient.base.Resource

```

A Spark context.

delete()

Delete context.

class sjsclient.context.**ContextManager**(*client*)

Bases: sjsclient.base.ResourceManager

Manage Context resources.

base_path = 'contexts'

create(*name*, *params=None*)

Create a Spark context.

Parameters

- **name** – Descriptive name of context
- **params** – Dictionary of context parameters

Return type Context

delete(*name*)

Delete a specific Context.

Parameters **name** – The name of the Context to delete.

get(*name*)

Get a specific Context.

Parameters **name** – The name of the Context to get.

Return type Context

resource_class

alias of Context

Indices and tables

- *genindex*
- *modindex*
- *search*