kong

Paul Gessinger

Jul 21, 2020

CONTENTS:

1 API **1**

1.1 State 1

1.2 Models 4

2 Indices and tables **9**

Index **11**

CHAPTER

ONE

API

From python, kong has an API that is very similar to the REPL. You can get an instance like

```
import kong
state = kong.get_instance()

state.ls()
state.create_job(command="sleep 10")
```

kong.**get_instance**() → *kong.state.State*

Create an instance of *kong.state.State*, by reading the default config, and preparing the database. The returned state object can be used for stateful work with the kong database.

Returns state object

1.1 State

class kong.state.**State**(*config:* kong.config.Config, *cwd:* kong.model.folder.Folder)

The state class provides a stateful interface to the kong database. This is modeled closely after the interactive pseudo-shell from kong.repl.Repl, but is pure python. (Actually, kong.repl.Repl is implemented entirely on top of *kong.state.State*, with argument parsing and result printing)

Many functions accept a *JobSpec* parameter. This can be one of:

- A job id (i.e. 1234)
- A job range in the form 1111..9999, which will select job ids within the range **inclusively**
- A path to a folder (i.e. /a/b/c). If this is the case, most methods have a *recursive* argument to instruct collection of jobs from the folder and it's descendants.

__init__(*config:* kong.config.Config, *cwd:* kong.model.folder.Folder) → None

Initializer for the state class. Takes an instance of *kong.config.Config* and a current working directory to start out in.

Parameters

- **config** – The config to initialize with
- **cwd** – Current working directory to start in

cd(*target: Union[str,* kong.model.folder.Folder*]* = *'.'*) → None

Change the current working directory.

Parameters **target** – String path or folder instance to change into.

create_job (**args: Any*, ***kwargs: Any*) → *kong.model.job.Job*
Create a job with the default driver. Passes through any arguments to the driver

Parameters

- **args** – Positional arguments
- **kwargs** – Keyword arguments

Returns The created Job

get_folders (*pattern: str*) → List[*kong.model.folder.Folder*]
Helper method to select jobs from a pattern.

Parameters **pattern** – Glob like pattern to select folders.

Returns List of selected folders

classmethod get_instance () → *kong.state.State*
Create an instance of `kong.state.State`, by reading the default config, and preparing the database. The returned state object can be used for stateful work with the kong database.

Returns state object

get_jobs (*name: Union[str, int,* kong.model.job.Job*]*, *recursive: bool = False*) → List[*kong.model.job.Job*]
Helper method to select jobs from a path, range or instance.

Parameters

- **name** – Identifies one or more jobs
- **recursive** – Select jobs recursively, i.e. follow folders and collect all jobs on the way

Returns A list with all collected jobs.

kill_job (*name: Union[str, int, kong.model.job.Job], recursive: bool = False, confirm: Callable[[str], bool] = <function YES>*) → None
Terminate execution of one or more jobs.

Parameters

- **name** – Path or job id
- **confirm** – Confirmation callback. Defaults to YES
- **recursive** – If *True*, will recursively select jobs for termination. Required if *path* is a n actual path.

ls (*path: str = '.', refresh: bool = False, recursive: bool = False*) → Tuple[List[*kong.model.folder.Folder*], List[*kong.model.job.Job*]]
Lists the current directory content.

Parameters

- **path** – The path to list the content for
- **refresh** – FLag to indicate whether job statuses should be refreshed
- **recursive** – Descend into the folder hierarchy to find all jobs to list

Returns List of folders and list of jobs found

mkdir (*path: str, exist_ok: bool = False, create_parent: bool = False*) → Optional[*kong.model.folder.Folder*]
Make a directory at the given path

Parameters

- **path** – The relative or absolute path to create
- **exist_ok** – If *True*, there will be no error if the folder already exists.
- **create_parent** – Create all parent directories of *path* if they don't exist

Returns The folder if one was created

mv(*source: Union[str, kong.model.job.Job, kong.model.folder.Folder]*, *dest: Union[str, kong.model.folder.Folder]*) → List[Union[*kong.model.job.Job*, *kong.model.folder.Folder*]]

Parameters

- **source** – The object to move, can be a path, job object or folder object
- **dest** – The object to move to. If *source* is a job, this can be a folder. If *source* is a folder, and *dest* is a folder, *source* will be moved *into dest*. If *dest* does not exist, *source* will be renamed to *dest*.

Returns List of moved objects

pushd(*folder: Union[Folder, str]*) → Iterator[None]

Contextmanager to temporarily change the current working directory.

Parameters **folder** – Folder instance or path string to change into

refresh_jobs(*jobs: List[kong.model.job.Job]*) → Sequence[*kong.model.job.Job*]

Refresh a list of jobs and retrieve their current status.

Parameters **jobs** – List of jobs to refresh

Returns Updated job instances

resubmit_job(*name: Union[str, int, kong.model.job.Job]*, *confirm: Callable[[str], bool] = <function YES>*, *recursive: bool = False*, *failed_only: bool = False*) → None

Resubmit one or more jobs. This causes them to run with the same settings as before.

Parameters

- **name** – Path or job id
- **confirm** – Confirmation callback. Defaults to YES
- **recursive** – If *True*, will recursively select jobs for resubmission. Required if *path* is a n actual path.
- **failed_only** – If *True* only select jobs in the FAILED state for resubmission.

rm(*name: Union[str, kong.model.job.Job, kong.model.folder.Folder]*, *recursive: bool = False*, *confirm: Callable[[str], bool] = <function State.<lambda>>*, *threads: Optional[int] = 1*) → bool

Remove jobs or folders.

Parameters

- **name** – A path, job or folder
- **recursive** – Recursively delete from *path*. Needed to remove a directory
- **confirm** – Callback to confirm. Defaults to *True*, i.e. will confirm automatically

Returns Whether the object at *path* was removed or not.

submit_job(*name: Union[str, int, kong.model.job.Job]*, *confirm: Callable[[str], bool] = <function YES>*, *recursive: bool = False*) → None

Submit one or more jobs using the driver it was created with. This will cause it to execute.

Parameters

- **name** – Path or job id

- **confirm** – Confirmation callback. Defaults to YES
- **recursive** – If *True*, will recursively select jobs for submission. Required if *path* is a n actual path.

wait(*jobspecs: Sequence[Union[str, int, kong.model.job.Job]]*, *recursive: bool = False*, *notify: bool = True*, *timeout: Optional[int] = None*, *poll_interval: Optional[int] = None*, *update_interval: Optional[datetime.timedelta] = None*, *progress: bool = False*) → Optional[Iterable[List[*kong.model.job.Job*]]]

Wait for completion of a number of job

Parameters

- **jobspec** – Selector for jobs, string path, job instance or folder instance
- **recursive** – Select jobs recursively
- **notify** – Notify on completion of wait
- **timeout** – Error out after a certain time
- **poll_interval** – How often to poll the driver for updates
- **update_interval** – How often to send update notificationa
- **progress** – If *True*, return progress information as an iterable

Returns An iterable if *progress* is *True*, else *None*.

class kong.config.**Config**(*data: Optional[Dict[str, Any]] = None*)

Class to handle loading the config data from disk.

__init__(*data: Optional[Dict[str, Any]] = None*) → None

Initalize method for the config. Will load the config file from the app directory (OS dependant)

Parameters **data** – Dictionary with pre-loaded data. Will be used as is if provided (optional)

1.2 Models

class kong.model.folder.**Folder**(**args*, ***kwargs*)

Represents a folder in the internal hierarchy for organizing jobs.

Variables

- **folder_id** – The ID of a folder instance
- **name** – Name of this folder instance
- **parent** – Points to the parent instance of this folder. Can be *None* for the root folder
- **created_at** – Timestamp of creation of this folder instance
- **updated_at** – When the instance was last updated.

DoesNotExist

alias of FolderDoesNotExist

add_folder(*name: str*) → *kong.model.folder.Folder*

Add a subfolder to this folder instance.

Parameters **name** – Name of the new folder name (without /)

Returns The new folder instance

static find_by_path(*path: str*, *cwd: Optional[Folder] = None*) → Optional[*kong.model.folder.Folder*]
Retrieve a folder instance by path

Parameters

- **path** – Path to the folder
- **cwd** – Directory to start working from, defaults to root folder

Returns The found folder or *None* if the path doesn't exist

folders_recursive() → Iterable[*kong.model.folder.Folder*]
Recursively find all folders below this one.

Returns All folders in the hierarchy from this folder. (Excludes this folder)

classmethod get_root() → *kong.model.folder.Folder*
Retrieve the root folder (/). There can be only one.

Returns Root folder instance

jobs_recursive() → Iterable[*kong.model.job.Job*]
Recursively get all jobs in this folder and descendants.

Returns Iterable over all jobs found, including jobs directly in this folder.

property path
The path to this folder

Returns The path

subfolder(*name: str*) → Optional[*kong.model.folder.Folder*]
Retrieve a direct subfolder of this folder instance

Parameters **name** – The unqualified name of the subfolder to retrieve.

Returns Folder instance if *name* exists, else *None*

class kong.model.job.**Job**(**args*, ***kwargs*)
Class representing a single job.

Variables

- **job_id** – The ID of a job instance
- **batch_job_id** – The job ID that the driver asigned. For batch systems, this is the internal ID of the batch system.
- **driver** – Holds the driver class used to create the job
- **folder** – The folder in which this job is located
- **command** – The command string the job (will) execute
- **data** – Arbitrary data store that drivers use to persist relevant information
- **status** – Currently synced status of the job. This is only updated if the driver's sync method is used
- **created_at** – When this job was created
- **updated_at** – When this job was last updated
- **cores** – Number of cores the job is supposed to run. Is not necessarily honored by all drivers.

- **memory** – Amount of memory to allocate for the job. Is not necessarily honored by all drivers.

DoesNotExist
alias of JobDoesNotExist

class Status(*value*)
Status enum which lists the various status types The exact meaning might vary from driver to driver.

Variables

- **UNKNOWN** – Catch all status which cannot be mapped
- **CREATED** – Job was created in the database, but not submitted yet
- **SUBMITTED** – Job has been launched via a driver, but might not run yet
- **RUNNING** – The job is currently executing
- **FAILED** – The job terminated abnormally
- **COMPLEETD** – The job completed successfully.

property driver_instance
Get the driver instance of this job. There might not be one (yet)

Returns Driver instance

ensure_driver_instance(*arg: Union[kong.drivers.driver_base.DriverBase, kong.config.Config]*) → None
Makes sure a driver instance is available to this job.

Parameters **arg** – Either a config object or an explicit driver instance.

get_status()
Get the current status of the job. Will synchronize first.

Returns The updated status.

kill()
Kill this job.

property log_dir
Get the log directory of this jobs

Returns The log directory

property output_dir
Get the output directory of this job.

Returns The output directory

remove()
Remove this job using the driver instance attached to the job.

resubmit()
Resubmit this job.

size(*ex: Optional[concurrent.futures._base.Executor] = None*) → int
Retrieve the size of the job output.

Parameters **ex** – An executor like *concurrent.futures.Executor*. If *None*, will execute serially

Returns Job output size in bytes.

stderr()
Convenience context manager to open a read file handle to the job's stderr.

stdout()
Convenience context manager to open a read file handle to the job's stdout.

submit()
Submit this job using the driver instance set on it.

wait(*timeout: Optional[int] = None*)
Wait for completion of this job

Parameters **timeout** – If set to a number, will raise a *TimeoutError* after that time

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

Symbols

__init__() (*kong.config.Config method*), 4
__init__() (*kong.state.State method*), 1

A

add_folder() (*kong.model.folder.Folder method*), 4

C

cd() (*kong.state.State method*), 1
Config (*class in kong.config*), 4
create_job() (*kong.state.State method*), 1

D

DoesNotExist (*kong.model.folder.Folder attribute*), 4
DoesNotExist (*kong.model.job.Job attribute*), 6
driver_instance() (*kong.model.job.Job property*), 6

E

ensure_driver_instance() (*kong.model.job.Job method*), 6

F

find_by_path() (*kong.model.folder.Folder static method*), 4
Folder (*class in kong.model.folder*), 4
folders_recursive() (*kong.model.folder.Folder method*), 5

G

get_folders() (*kong.state.State method*), 2
get_instance() (*in module kong*), 1
get_instance() (*kong.state.State class method*), 2
get_jobs() (*kong.state.State method*), 2
get_root() (*kong.model.folder.Folder class method*), 5
get_status() (*kong.model.job.Job method*), 6

J

Job (*class in kong.model.job*), 5
Job.Status (*class in kong.model.job*), 6
jobs_recursive() (*kong.model.folder.Folder method*), 5

K

kill() (*kong.model.job.Job method*), 6
kill_job() (*kong.state.State method*), 2

L

log_dir() (*kong.model.job.Job property*), 6
ls() (*kong.state.State method*), 2

M

mkdir() (*kong.state.State method*), 2
mv() (*kong.state.State method*), 3

O

output_dir() (*kong.model.job.Job property*), 6

P

path() (*kong.model.folder.Folder property*), 5
pushd() (*kong.state.State method*), 3

R

refresh_jobs() (*kong.state.State method*), 3
remove() (*kong.model.job.Job method*), 6
resubmit() (*kong.model.job.Job method*), 6
resubmit_job() (*kong.state.State method*), 3
rm() (*kong.state.State method*), 3

S

size() (*kong.model.job.Job method*), 6
State (*class in kong.state*), 1
stderr() (*kong.model.job.Job method*), 6
stdout() (*kong.model.job.Job method*), 6
subfolder() (*kong.model.folder.Folder method*), 5
submit() (*kong.model.job.Job method*), 7
submit_job() (*kong.state.State method*), 3

W

wait() (*kong.model.job.Job method*), 7
wait() (*kong.state.State method*), 4