clspy

Release 0.0.5

Connard.Lee

Mar 04, 2022

PACKAGES

- genindex
- modindex
- search

CHAPTER

ONE

CLSPY

1.1 clspy package

1.1.1 Submodules

1.1.2 clspy.cli module

1.1.3 clspy.config module

class Config(*file='config.json'*)

Bases: object

Config is a config manager

Aim to support json/xml/ini/yaml parse and dump operations.

Object can be easily used ad python dict type.

Json: object managed as dict

Ini: object managed as dict

Xml: ElementTree managed object

Yaml: object managed as dict

get(*key=None*)

get config key from parsed contents

Parameters **key** (*<str>, optional*) – key of value or object. Defaults to None.

Returns None

load(*file*)

load config file

Parameters **file** – relative or absolute path to file

Raises **Exception.args** – Exception.args("Not supported")

save(*file*)

dump config contents to file

Parameters **file** (*Path*) – Where to dump file, path would be relative or absolute

Raises **Exception.args** – Exception.args("Not supported")

Returns If config content type is not match <file>'s suffix

Return type False

class ConfigType(*value*)
Bases: enum.Enum

Support types: Json, Yaml, Ini, Xml

CCIni = 3

CCJson = 1

CCXml = 4

CCYaml = 2

class ConfigUnique(**args*, ***kwargs*)
Bases: *clspy.config.Config*

Globally unique Config object

Parameters **metaclass** – Defaults to SingletonMetaclass.

clslog = <RootLogger root (DEBUG)>
pyyaml package is required

1.1.4 clspy.crypto module

class Md5
Bases: object

wheel of md5

file(*filename*)

Parameters **filename** – calc md5 digest from a file

Returns md5 string

same(*f1*, *f2*)

Parameters

- **f1** – first filename or content<string>
- **f2** – second filename or content<string>

Returns md5(f1) == md5(f2)

string(*content*)

Parameters **content** – calc digest from string

Returns md5 string

1.1.5 clspy.db module

class Sql

Bases: object

wheel of SQL operations

Parameters **dburl** – sqlalchemy url parameter

dialect+driver://username:password@host:port/database

password is URL encoded(import urllib.parse)

sqlite://<nohostname>/<path>, where <path> is relative

sqlite:///foo.db, means foo.db in current path

sqlte+pysqlte:///:memory:

sqlite:////absolute/path/to/foo.db, absolute after '///'

sqlite:///C:pathtofoo.db, Windows absolute path

r'sqlite:///C:pathtofoo.db', Windows alternative using raw string

engine = create_engine('sqlite://') Using SQLite :memory:

postgresql://scott:tiger@localhost:5432/mydatabase

postgresql+psycopg2://scott:tiger@localhost/mydatabase

postgresql+pg8000://scott:tiger@localhost/mydatabase

mysql://scott:tiger@localhost/foo

mysql+mysqldb://scott:tiger@localhost/foo

mysql+pymysql://scott:tiger@localhost/foo

oracle://scott:tiger@127.0.0.1:1521/sidname

oracle+cx_oracle://scott:tiger@tnsname

mssql+pyodbc://scott:tiger@mydsn

mssql+pymssql://scott:tiger@hostname:port/dbname

commit()

create(*dburl*)

Create engine

create_table(*table=<class 'sqlalchemy.orm.decl_api.Base'>*)

Create table use MetaData, Column,

More on https://www.osgeo.cn/sqlalchemy/core/type_basics.html

delete(*item=None*)

init(*driver*, *user*, *passwd*, *host*, *port*, *dbname*)

Init database

Parameters

- **driver** (*str*) – pymysql/mysqldb/pyodbc
- **user** (*str*) – username of database connection
- **passwd** (*str*) – pasword of database connection

- **host** (*str*) – database host
- **port** (*str*) – database port
- **dbname** (*str*) – database instance name

insert(*item=None*)
Insert an item

Parameters **item** (*MetaData, optional*) – Basic data structure. Defaults to None.

Raises

- **Exception** – Any exception
- **e** – Any exception

query(*table=None*)
How to filter

result.filter(Table.attr == value)

How to update

result.filter(Table.attr == value).update({'attr': 'new_value'}) filter().delete() filter().all()

1.1.6 clspy.log module

class Logger(**args*, ***kw*)
Bases: *clspy.singleton.SingletonClass*

Gloable logging wrapper

This class inherits a module named loguru, Logger module will use pure logging if loguru module not imported correctly.

property file

property log
get a logger

Parameters **filename** (*Path, optional*) – Log to file. Defaults to None.

Returns <loguru.logger> returns if loguru imported correctly, otherwise <root_logger> returns

Return type logger

1.1.7 clspy.singleton module

class SingletonClass(**args*, ***kw*)
Bases: object

Singleton class wapper Only support **__init__** function without parameters, usage:

class Cls(SingletonClass):

def __init__(self): pass

class SingletonMetaclass
Bases: type

Metaclass implement, usage:

class Cls(metaclass=SingletonMetaclass): pass

clspy_singleton(*cls*, **args*, ***kv*)
Wrapper function to construct a singleton

Returns a singleton class

Return type object

1.1.8 clspy.utils module

dir_copy(*srcpath*, *dstpath*)

is_frozen()

mkdir_p(*absolute_path*)
mkdir -p implement

Usage:

mkdir_p('D:ABC.txt')

mkdir_p('~/A/B/C')

pip_conf_install(*src=None*)

pipguess()

rmdir(*path*)
Warning: all files and directories in path will be deleted.

runpath(*file='/home/docs/checkouts/readthedocs.org/user_builds/clspy/checkouts/latest/docs/../clspy/utils.py'*)

setenv(*permanent=True*, *key=None*, *value=None*)

win_runtime_cp(*src*, *to*)

1.1.9 clspy.version module

0.0.5

1.1.10 Module contents

class Config(*file='config.json'*)
Bases: object

Config is a config manager

Aim to support json/xml/ini/yaml parse and dump operations.

Object can be easily used ad python dict type.

Json: object managed as dict

Ini: object managed as dict

Xml: ElementTree managed object

Yaml: object managed as dict

get(*key=None*)
get config key from parsed contents

Parameters **key** (*<str>, optional*) – key of value or object. Defaults to None.

Returns None

load(*file*)
load config file

Parameters **file** – relative or absolute path to file

Raises **Exception.args** – Exception.args("Not supported")

save(*file*)
dump config contents to file

Parameters **file** (*Path*) – Where to dump file, path would be relative or absolute

Raises **Exception.args** – Exception.args("Not supported")

Returns If config content type is not match <file>'s suffix

Return type False

class ConfigType(*value*)
Bases: enum.Enum

Support types: Json, Yaml, Ini, Xml

CCIni = 3

CCJson = 1

CCXml = 4

CCYaml = 2

class ConfigUnique(**args*, ***kwargs*)
Bases: *clspy.config.Config*

Globally unique Config object

Parameters **metaclass** – Defaults to SingletonMetaclass.

class Logger(**args*, ***kw*)
Bases: *clspy.singleton.SingletonClass*

Gloable logging wrapper

This class inherits a module named loguru, Logger module will use pure logging if loguru module not imported correctly.

property file

property log
get a logger

Parameters **filename** (*Path, optional*) – Log to file. Defaults to None.

Returns <loguru.logger> returns if loguru imported correctly, otherwise <root_logger> returns

Return type logger

class Md5
Bases: object

wheel of md5

file(*filename*)

Parameters **filename** – calc md5 digest from a file

Returns md5 string

`same`(*f1*, *f2*)

Parameters

- **f1** – first filename or content<string>
- **f2** – second filename or content<string>

Returns md5(f1) == md5(f2)

`string`(*content*)

Parameters **content** – calc digest from string

Returns md5 string

`class SingletonClass`(**args*, ***kw*)

Bases: `object`

Singleton class wapper Only support **__init__** function without parameters, usage:

class Cls(SingletonClass):

def __init__(self): pass

`class SingletonMetaclass`

Bases: `type`

Metaclass implement, usage:

class Cls(metaclass=SingletonMetaclass): pass

`class Sql`

Bases: `object`

wheel of SQL operations

Parameters **dburl** – sqlalchemy url parameter

dialect+driver://username:password@host:port/database

password is URL encoded(import urllib.parse)

sqlite://<nohostname>/<path>, where <path> is relative

sqlite:///foo.db, means foo.db in current path

sqlte+pysqlte:///:memory:

sqlite:////absolute/path/to/foo.db, absolute after '///'

sqlite:///C:pathtofoo.db, Windows absolute path

r'sqlite:///C:pathtofoo.db', Windows alternative using raw string

engine = create_engine('sqlite://') Using SQLite :memory:

postgresql://scott:tiger@localhost:5432/mydatabase

postgresql+psycopg2://scott:tiger@localhost/mydatabase

postgresql+pg8000://scott:tiger@localhost/mydatabase

mysql://scott:tiger@localhost/foo

mysql+mysqldb://scott:tiger@localhost/foo

mysql+pymysql://scott:tiger@localhost/foo

oracle://scott:tiger@127.0.0.1:1521/sidname

oracle+cx_oracle://scott:tiger@tnsname

mssql+pyodbc://scott:tiger@mydsn

mssql+pymssql://scott:tiger@hostname:port/dbname

commit()

create(*dburl*)
Create engine

create_table(*table=<class 'sqlalchemy.orm.decl_api.Base'>*)
Create table use MetaData, Column,

More on https://www.osgeo.cn/sqlalchemy/core/type_basics.html

delete(*item=None*)

init(*driver*, *user*, *passwd*, *host*, *port*, *dbname*)
Init database

Parameters

- **driver** (*str*) – pymysql/mysqldb/pyodbc
- **user** (*str*) – username of database connection
- **passwd** (*str*) – pasword of database connection
- **host** (*str*) – database host
- **port** (*str*) – database port
- **dbname** (*str*) – database instance name

insert(*item=None*)
Insert an item

Parameters **item** (*MetaData, optional*) – Basic data structure. Defaults to None.

Raises

- **Exception** – Any exception
- **e** – Any exception

query(*table=None*)
How to filter

result.filter(Table.attr == value)

How to update

result.filter(Table.attr == value).update({'attr': 'new_value'}) filter().delete() filter().all()

clspy_singleton(*cls*, **args*, ***kv*)
Wrapper function to construct a singleton

Returns a singleton class

Return type object

CHAPTER

TWO

ABOUT CLSPY

clspy clspy is a set of fast programming tools that are gradually generated during the python learning process.

CHAPTER

THREE

FAQ OF CLSPY

CHAPTER

FOUR

LICENSE

MIT License

Copyright (c) 2021, Connard.Lee

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

PYTHON MODULE INDEX

C

L

INDEX

R

S

W