DKB Documentation

DKB team

Aug 17, 2018

CONTENTS:

CHAPTER

ONE

PYDKB PACKAGE

Common library for Data Knowledge Base development.

1.1 Subpackages

1.1.1 pyDKB.common package

Common modules.

Submodules

pyDKB.common.Type module

Abstract class for type definitions.

Example

```
>>> myType = Type("Orange", "Apple")
>>> myType.add("Plum")
>>> t = myType.Orange
>>> if t == myType.Orange:
...     print "Orange!"
... elif t == myType.member("Apple"):
...     print "Apple!"
...
Orange!
>>> if not myType.member("Walnut"):
...     print "Wrong type!"
...
Wrong type!
```

class pyDKB.common.Type.**Type**(**args*)

Bases: object

Abstract class for type definitions.

Member names (*str*) are passed to the constructor as positional arguments.

add(*name*)

Add member.

Parameters **name** (*str*) – name of the member to be added

hasMember(*val*)
Check if the member exists (by value).

Parameters **val** (*int*) – member to be checked

Returns True/False

Return type bool

member(*name*)
Check if the member exists (by name).

Parameters **name** (*str*) – name to be checked

Returns member value or False if member does not exist

Return type int, bool

memberName(*val*)
Return string name of the member.

Parameters **val** (*int*) – member to retrieve name for

Returns member name of False if member does not exist

Return type str, bool

pyDKB.common.custom_readline module

pyDKB.common.custom_readline.**custom_readline**(*f*, *newline*)
Custom readline() function.

Custom_readline() separates content from a text file 'f' by delimiter 'newline' to distinct messages. The last line can be incomplete, if the input data flow is interrupted in the middle of data writing.

Keyword arguments: f – file/stream to read newline – custom delimiter

pyDKB.common.exceptions module

Definition of common modules exceptions

exception pyDKB.common.exceptions.**HDFSException**
Bases: exceptions.RuntimeError

Base Exception for HDFS module.

pyDKB.common.hdfs module

Utils to interact with HDFS.

pyDKB.common.hdfs.**check_stderr**(*proc*, *timeout=None*, *max_lines=1*)
Wait till the end of the subprocess and send its STDERR to STDERR.

Output only MAX_LINES of the STDERR to the current STDERR; if MAX_LINES == None, output all the STDERR.

Return value is the subprocess' return code.

pyDKB.common.hdfs.**getfile**(*fname*)
Download file from HDFS.

Return value: file name (without directory)

pyDKB.common.hdfs.**listdir**(*dirname*, *mode='a'*)
List files and/or subdirectories of HDFS directory.

Parameters: dirname – directory to list mode – 'a': list all objects

'f': list files 'd': list subdirectories

pyDKB.common.hdfs.**makedirs**(*dirname*)
Try to create directory (with parents).

pyDKB.common.hdfs.**putfile**(*fname*, *dest*)
Upload file to HDFS.

pyDKB.common.json_utils module

Utils to work with JSON (dict) objects.

pyDKB.common.json_utils.**nestedKeys**(*key*)
Transform STRING with nested keys into LIST.

Parameters:

STRING key – dot-separated list of nested keys. If a key contains dot itself, the key must be put between quotation marks.

pyDKB.common.json_utils.**valueByKey**(*json_data*, *key*)
Return value by a chain (list) of nested keys.

Parameters: DICT json_data – to search in STRING key – dot-separated list of nested keys

1.1.2 pyDKB.dataflow package

Dataflow organization utils.

Subpackages

pyDKB.dataflow.stage package

Stage submodule init file.

class pyDKB.dataflow.stage.**JSONProcessorStage**
Bases: *pyDKB.dataflow.stage.AbstractProcessorStage.AbstractProcessorStage*

JSON2JSON Processor Stage

Input message: JSON Output message: JSON

file_input(*fd*)
Override AbstractProcessorStage.file_input

file_nd_json(*fd*)
Read file as NDJSON file.

Raises ValueError if can't read the first line.

file_true_json(*fd*)
Read file as true JSON file.

class pyDKB.dataflow.stage.**TTLProcessorStage**
Bases: *pyDKB.dataflow.stage.AbstractProcessorStage.AbstractProcessorStage*

TTL2TTL Processor Stage

Input message: TTL Output message: TTL

output(*message*)
Put the (list of) message(s) to the output buffer.

class pyDKB.dataflow.stage.**JSON2TTLProcessorStage**
Bases: *pyDKB.dataflow.stage.processors.JSONProcessorStage*, *pyDKB.dataflow.stage.processors.TTLProcessorStage*

JSON2TTL Processor Stage

Input message: JSON Output message: TTL

input()
Override: Falls back to JSONProcessorStage.input

output(*message*)
Override: Falls back to TTLProcessorStage.output

Submodules

pyDKB.dataflow.stage.AbstractProcessorStage module

Definition of an abstract class for Dataflow Data Processing Stages.

USAGE: ProcessorStage [<options>] [<input files>]

OPTIONS:

-s, --source	{f|s|h} - where to get data from: local (f)iles, (s)tdin, (h)dfs
-i, --input-dir	DIR - base directory for relative input file names (for local and HDFS sources). If <input files> not specified, all files from the directory will be taken as the input.
-d, --dest	{f|s|h} - where to send data to: local (f)iles, (s)tdout, (h)dfs
-o, --output-dir	DIR - base directory for output files (for local and HDFS sources)
--hdfs	• equivalent to "–source h –dest h"
-m, --mode	MODE - MODE: (f)ile = –source f **–dest f (can be** **rewritten with 's'** or 'h') **(s)tream = –source s (can be** rewritten with 'h') –dest s **(m)apreduce = –source s (can be** rewritten with 'h')

–dest s

class pyDKB.dataflow.stage.AbstractProcessorStage.**AbstractProcessorStage**(*description='DKB Dataflow data processing stage.'*)

Bases: *pyDKB.dataflow.stage.AbstractStage.AbstractStage*

Abstract class to implement Processor stages

Processor stage – is a stage for data processing/transfornation.

Class/instance variable description: * Current processing file name:

__current_file_full – full name with path __current_file – file name

- **Iterable object for input data sources (file descriptors)** __input
- **Output messages buffer:** __output_buffer
- Generator object for output file descriptor OR file descriptor (for (s)tream mode)

  __output
- **List of objects to be "stopped"** __stoppable

clear_buffer()
Drop buffered output messages.

defaultArguments()
Default parser configuration.

file_flush()
Flush message buffer into a file.

By default writes to file as to a stream. To be implemented individually if needed.

file_input(*fd*)
Generator for input messages.

By default reads file just as stream. To be implemented individually for other cases.

flush_buffer()
Flush message buffer to the output.

forward()
Send EOPMessage in the streaming output mode.

input()
Generator for input messages.

Returns iterable object. Every iteration returns single input message to be processed.

input_message_class()
Get input message class.

output(*message*)
Put the (list of) message(s) to the output buffer.

output_message_class()
Get output message class.

parseMessage(*input_message*)
Verify and parse input message.

Is called from input() method.

parse_args(*args*)
Parse arguments and set dependant arguments if neeeded.

static process(*stage*, *input_message*)
Transform input_message -> output_message.

To be implemented individually for every stage. Takes the stage as first argument to allow calling output()

from inside the function.

Return value: True – processing successfully finished False – processing failed (skip the input message)

run()
Run process() for every input() message.

stop()
Finalize all the processes and prepare to exit.

stream_flush(*fd=None*)
Flush message buffer as a stream.

stream_input(*fd*)
Generator for input messages.

Read data from STDIN; Split stream into messages; Yield Message object.

pyDKB.dataflow.stage.AbstractStage module

Definition of an abstract class for Dataflow Stages.

class pyDKB.dataflow.stage.AbstractStage.**AbstractStage**(*description='DKB Dataflow stage'*)

Bases: object

Class/instance variable description: * Argument parser (argparse.ArgumentParser)

__parser

- **Parsed arguments (argparse.Namespace)** ARGS

add_argument(**args*, ***kwargs*)
Add specific (not common) arguments.

defaultArguments()
Config argument parser with parameters common for all stages.

parse_args(*args*)
Parse arguments and set dependant arguments if needed.

print_usage(*fd=<open file '<stderr>', mode 'w'>*)
Print usage message.

run()
Run the stage.

pyDKB.dataflow.stage.processors module

Processor stages definitions (with predefined message type).

class pyDKB.dataflow.stage.processors.**JSONProcessorStage**
Bases: *pyDKB.dataflow.stage.AbstractProcessorStage.AbstractProcessorStage*

JSON2JSON Processor Stage

Input message: JSON Output message: JSON

file_input (*fd*)
Override AbstractProcessorStage.file_input

file_nd_json (*fd*)
Read file as NDJSON file.

Raises ValueError if can't read the first line.

file_true_json (*fd*)
Read file as true JSON file.

class pyDKB.dataflow.stage.processors.**TTLProcessorStage**
Bases: *pyDKB.dataflow.stage.AbstractProcessorStage.AbstractProcessorStage*

TTL2TTL Processor Stage

Input message: TTL Output message: TTL

output (*message*)
Put the (list of) message(s) to the output buffer.

class pyDKB.dataflow.stage.processors.**JSON2TTLProcessorStage**
Bases: *pyDKB.dataflow.stage.processors.JSONProcessorStage*, *pyDKB.dataflow.stage.processors.TTLProcessorStage*

JSON2TTL Processor Stage

Input message: JSON Output message: TTL

input ()
Override: Falls back to JSONProcessorStage.input

output (*message*)
Override: Falls back to TTLProcessorStage.output

Submodules

pyDKB.dataflow.cds module

Extended CDSInvenioConnector allowing us to login via Kerberos

pyDKB.dataflow.dkbID module

Utils to generate unique yet meaningful identifier for DKB objects.

pyDKB.dataflow.dkbID.**dkbID** (*json_data*, *data_type*)
Return unique identifier for object of TYPE based on DATA.

pyDKB.dataflow.exceptions module

Definition of DKB Dataflow exceptions

exception pyDKB.dataflow.exceptions.**DataflowException**
Bases: exceptions.Exception

Base Exception for Dataflow modules.

pyDKB.dataflow.messages module

Definition of abstract message class and specific message classes

class pyDKB.dataflow.messages.**AbstractMessage**(*message=None*)
Bases: object

Abstract message

content()
Return message content.

decode(*code*)
Decode original from CODE to TYPE-specific format.

Raises ValueError

decoded = None

encode(*code*)
Encode original message from TYPE-specific format to CODE.

Raises ValueError

encoded = None

classmethod extension()
Return file extension corresponding this message type.

getOriginal()
Return original message.

msg_type = None

native_types = []

classmethod typeName()
Return message type name as string.

exception pyDKB.dataflow.messages.**DecodeUnknownType**(*code*, *cls*)
Bases: exceptions.NotImplementedError

Exception to be thrown when message type is not decodable.

exception pyDKB.dataflow.messages.**EncodeUnknownType**(*code*, *cls*)
Bases: exceptions.NotImplementedError

Exception to be thrown when message type is not encodable.

class pyDKB.dataflow.messages.**JSONMessage**(*message=None*)
Bases: *pyDKB.dataflow.messages.AbstractMessage*

Message in JSON format.

decode(*code=1*)
Decode original data as JSON.

encode(*code=1*)
Encode JSON as CODE.

msg_type = 2

native_types = [<type 'dict'>]

pyDKB.dataflow.messages.**Message**(*msg_type*)
Return class XXXMessage, where XXX is the passed type.

class pyDKB.dataflow.messages.**TTLMessage**(*message=None*)
Bases: *pyDKB.dataflow.messages.AbstractMessage*

Messages in TTL format

Single message = single TTL statement

decode(*code=1*)
Decode original data as TTL.

Currently takes text as it is. TODO: check some formal matter to confirm the string is TTL.

encode(*code=1*)
Encode JSON as CODE.

msg_type = 3

native_types = [<type 'str'>, <type 'unicode'>]

pyDKB.dataflow.types module

Type definitions for library objects.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

I

J

L

M

N

O

P

R

S

T

V