
BioMaj Manager Documentation

Release 1.0.3

Emmanuel Quevillon

November 27, 2016

1	decorators API reference	3
2	links API reference	5
3	manager API reference	7
4	news API reference	9
5	plugins API reference	11
6	utils API reference	13
7	writer API reference	17
8	Indices and tables	19
	Python Module Index	21

Contents:

decorators API reference

Global decorators for BioMAJ Manager

`biomajmanager.decorators.bank_required(func)`

Decorator function that checks a bank name is set

Parameters `func` (*Function*) – Decorated function

Returns Result of function called

Return type `func`

Raises `SystemExit` – If no bank set

`biomajmanager.decorators.deprecated(func)`

This is a decorator which can be used to mark functions as deprecated.

It will result in a warning being emitted when the function is used.

Parameters `func` (*Function*) – Decorated function

Returns Result of function called

Return type `func`

`biomajmanager.decorators.user_granted(func)`

Decorator function that checks a user has enough right to perform action

Parameters `func` (*Function*) – Decorated function

Returns Result of function called

Return type `func`

Raises `SystemExit` – If no owner found either in bank nor in config file

links API reference

manager API reference

Manager.SAVE_BANK_LINE_PATTERN: Print pattern (str), %-20s\t%-30s\t%-20s\t%-20s\t%-20s\n

Manager.simulate: Simulate mode (bool), False

Manager.verbose: Verbose mode (bool), False

news API reference

News.MAX_NEWS: Maximum number of news returned while reading news directory (int), 5

Utilities to create some news for BioMAJ

class `biomajmanager.news.News` (*news_dir=None, config=None, max_news=None*)

Bases: `object`

Class for creating news to be displayed for BioMAJ

__init__ (*news_dir=None, config=None, max_news=None*)

Initiate object building

Parameters

- **news_dir** (*str*) – Path to directory containing templates
- **config** (*configParser*) – Configuration object
- **max_news** (*int*) – Number of news to get when displaying news (default `News.MAX_NEWS`)

Raises

- **SystemExit** – If ‘news_dir’ is not a directory
- **SystemExit** – If ‘NEWS’ section is not defined in `manager.properties`
- **SystemExit** – If ‘news.dir’ is not set in `manager.properties`

__weakref__

list of weak references to the object (if defined)

get_news (*news_dir=None, reverse=True*)

Get the news to be displayed from the specific news.dir directory

Parameters

- **news_dir** (*str*) – Path to news directory
- **reverse** (*bool*) – Reverse list of news files, default True

Returns `news_files`, list of news files found into ‘news’ directory

Return type `list`

Raises

- **SystemExit** – If path ‘news_dir’ does not exist
- **SystemExit** – If ‘news_dir’ not set

class `biomajmanager.news.RSS` (*rss_file=None*, *args, **kwargs)

Bases: `biomajmanager.news.News`

Class for generating RSS feed from news files

__init__ (*rss_file=None*, *args, **kwargs)

Initiate object building

generate_rss (*rss_file=None*, *data=None*)

Generate RSS file from news

Parameters

- **rss_file** (*str*) – Path to file rss.xml
- **data** (*dict* `data['news'] = { ... }`) – Data to create RSS from

Returns Boolean

Return type `bool`

Raises

- **SystemExit** – If ‘news’ key is not found in ‘data’ dict
- **SystemExit** – If rss file cannot be opened

plugins API reference

Plugins mechanism to load user defined method

class `biomajmanager.plugins.BMPlugin`
Base plugin class for BioMAJ manager

get_config()
Get the BioMAJ manager config as object

Returns configparser instance

Return type :class:'configparser'

get_manager()
Get the BioMAJ manager instance

Returns Manager

Return type `biomaj.manager.Manager`

get_name()
Get the name of the plugin. Based on the class name

set_config(config)
Set BioMAJ manager config object

set_manager(manager)
Set BioMAJ manager config object

class `biomajmanager.plugins.Plugins` (*manager=None, name=None*)
Plugin class for BioMAJ Manager

__init__ (*manager=None, name=None*)
Create the plugin object

Parameters

- **manager** (`biomajmanager.manager.Manager`) – Manager instance
- **name** (*String*) – Name of the plugin to load. [DEFAULT: load all plugins]

Raises

- **SystemExit** – If 'manager' arg is not given
- **SystemExit** – If 'PLUGINS' section not found in `manager.properties`
- **SystemExit** – If 'plugins.dir' not set in `manager.properties`
- **SystemExit** – If 'plugins.list' not set in `manager.properties`

- **SystemExit** – If ‘plugins.dir’ does not exist

__weakref__

list of weak references to the object (if defined)

utils API reference

Utils.DATE_FMT: date format pattern (str), %Y-%m-%d %H:%M:%S

Utils.show_warn: show warnings mesasge (bool), True

Utils.show_debug: show debug message (bool), True

Utils.show_verbose: show verbose message (bool), True

Utils.timer_start: Start timer value (float), 0.0

Utils.timer_stop: Stop timer value (float), 0.0

Utilities class for BioMAJ Manager

class `biomajmanager.utils.Utils`

Utility class

`__weakref__`

list of weak references to the object (if defined)

static **`clean_symlinks`** (*path=None, delete=False*)

Search for broken symlinks.

Given a path, it search for all symlinks 'path' directory and remove broken symlinks. If delete is True, remove broken symlink(s), otherwise lists broken symlinks.

Parameters

- **`path`** (*str*) – Path to search symlinks from
- **`delete`** (*bool*) – Wether to delete or not broken symlink(s)

Returns Number of cleaned/deleted link(s)

Return type `int`

Raises

- **`SystemExit`** – If path not found, or not given
- **`SystemExit`** – If remove of symlink(s) failed

static **`elapsed_time`** ()

Get the elapsed time between start and stop timer.

Stop timer call is not required. If not set, it is automatically called as soon as the method is called

Returns Elapsed time

Return type `float`

Raises `SystemExit` – If `Utils.timer_start` is not defined

static `error` (*msg*)

Prints error message on `STDERR` and exits with exit code 1

Parameters `msg` (*str*) – Message to print

Returns Error message

Return type *str*

Raises `SystemExit` –

static `get_broken_links` (*path=None*)

Search for broken symlinks from a particular path.

If path is not given or `None`, then it search from the production directory.

Parameters `path` (*str*) – Path to search broken links from

Returns Number of found broken links

Return type *int*

Raises `SystemExit` – If path does not exist

static `get_deepest_dir` (*path=None, full=False, limit=0*)

Return only one deepest dir from the path

Parameters

- `path` (*str*) – Path
- `full` (*bool*) – Returns complete path or not
- `limit` (*int*) – Limit deepest search to *limit* depth, default 0, no limit

Returns Directory name

Return type *str*

static `get_deepest_dirs` (*path=None, full=False, limit=0*)

Get the last directories from a path

Parameters

- `path` (*str*) – Path to start from
- `full` (*bool*) – Get the full path otherwise the directory only
- `limit` (*int*) – Limit deepest search to *limit* depth, default 0, no limit

Returns List of directories

Return type *list*

Raises `SystemExit` – If ‘path’ not given or does not exist

static `get_files` (*path=None*)

Return the list of file(s) found for a given path

Parameters `path` (*str*) – Path to search from

Returns List of file(s) found

Return type *list*

Raises `SystemExit` – If path does not exist

static `get_now()`

Get current time from `time.time` formatted using `Utils.DATE_FMT`

Returns Current time formatted using `Utils.DATE_FMT`

Return type `time.time`

static `get_subtree(path=None, limit=0)`

Get the subtree structure from a root path

E.g.: File system is `/t/a1/a2/a3`, `get_subtree(path='/t')` -> `/a1/a2/a3` :param path: Root path to get subtree structure from :type path: str :param limit: Limit deepest search to *limit* depth, default 0, no limit :type limit: int :return: List of found subtree :rtype: list

static `ok(msg)`

Prints a [OK] msg

Parameters `msg (str)` – Message to print

Returns Message to print

Return type `str`

static `reset_timer()`

Reset to 0.0 `timer_start()` and `timer_stop()` for a new `elapsed_time()` count

static `start_timer()`

Set current time at function call

static `stop_timer()`

Set current time at function call

static `time2date(otime)`

Convert a timestamp into a datetime object

Parameters `otime (time)` – Timestamp to convert

Returns Formatted time to date

Return type `datetime.datetime`

static `time2datefmt(otime,fmt='%Y-%m-%d %H:%M:%S')`

Converts a timestamp into a date following the format `fmt`, default to `Utils.DATE_FMT`

Parameters

- `otime (time)` – Timestamp to convert
- `fmt (str)` – Date format to follow for conversion

Returns Formatted time to date

Return type `datetime.datetime`

static `uprint(msg,to=<open file '<stdout>', mode 'w'>)`

Redefined print function to support python 2 and 3

Parameters

- `msg (str)` – Message to print
- `to (file)` – File handle

Returns Message to print

Return type `str`

static user ()

Returns the current user running or using the script. Given by `getpass.getuser()`

Returns User name

Return type `str`

static verbose (*msg*)

Prints verbose message. Requires `Manager.verbose` to be `True`

Parameters **msg** (*str*) – Verbose message to print

Returns Verbose message

Return type `str`

static warn (*msg*)

Prints warning message. Required `Utils.show_warn` to be set to `True`

Parameters **msg** (*str*) – Warning message to print

Returns Warning message

Return type `str`

writer API reference

Writer class to be used with Jinja2 templates

class `biomajmanager.writer.Writer` (*template_dir=None, config=None, output=None*)

Writer class for BioMAJ manager to create what's desired as output

__init__ (*template_dir=None, config=None, output=None*)

Create Writer object

Parameters

- **template_dir** (*str*) – Root directory where to find templates
- **config** (*configparser*) – Global configuration file from BiomajConfig
- **output** (*str*) – Output file. Default STDOUT

Raises

- **SystemExit** – If 'template_dir' is not given
- **SystemExit** – If 'MANAGER' section not found in `manager.properties`
- **SystemExit** – If 'template.dir' not set in `manager.properties`

__weakref__

list of weak references to the object (if defined)

write (*template=None, data=None*)

Print template 'data' to stdout using template file 'template'.

'data' arg can be left None, this way method can be used to render file from scratch

Parameters

- **template** (*str*) – Template file name
- **data** (*dict*) – Template data

Returns True, throws on error

Return type `bool`

Raises

- **SystemExit** – If 'template' is None
- **SystemExit** – If 'template' is not found
- **SystemExit** – If 'template' has a syntax error in it
- **SystemExit** – If 'output' file cannot be opened

Indices and tables

- `genindex`
- `modindex`
- `search`

b

`biomajmanager.decorators`, [3](#)
`biomajmanager.news`, [9](#)
`biomajmanager.plugins`, [11](#)
`biomajmanager.utils`, [13](#)
`biomajmanager.writer`, [17](#)

Symbols

`__init__()` (biomajmanager.news.News method), 9
`__init__()` (biomajmanager.news.RSS method), 10
`__init__()` (biomajmanager.plugins.Plugins method), 11
`__init__()` (biomajmanager.writer.Writer method), 17
`__weakref__` (biomajmanager.news.News attribute), 9
`__weakref__` (biomajmanager.plugins.Plugins attribute), 12
`__weakref__` (biomajmanager.utils.Utils attribute), 13
`__weakref__` (biomajmanager.writer.Writer attribute), 17

B

`bank_required()` (in module biomajmanager.decorators), 3
biomajmanager.decorators (module), 3
biomajmanager.news (module), 9
biomajmanager.plugins (module), 11
biomajmanager.utils (module), 13
biomajmanager.writer (module), 17
BMPlugin (class in biomajmanager.plugins), 11

C

`clean_symlinks()` (biomajmanager.utils.Utils static method), 13

D

`deprecated()` (in module biomajmanager.decorators), 3

E

`elapsed_time()` (biomajmanager.utils.Utils static method), 13
`error()` (biomajmanager.utils.Utils static method), 14

G

`generate_rss()` (biomajmanager.news.RSS method), 10
`get_broken_links()` (biomajmanager.utils.Utils static method), 14
`get_config()` (biomajmanager.plugins.BMPlugin method), 11

`get_deepest_dir()` (biomajmanager.utils.Utils static method), 14
`get_deepest_dirs()` (biomajmanager.utils.Utils static method), 14
`get_files()` (biomajmanager.utils.Utils static method), 14
`get_manager()` (biomajmanager.plugins.BMPlugin method), 11
`get_name()` (biomajmanager.plugins.BMPlugin method), 11
`get_news()` (biomajmanager.news.News method), 9
`get_now()` (biomajmanager.utils.Utils static method), 14
`get_subtree()` (biomajmanager.utils.Utils static method), 15

N

News (class in biomajmanager.news), 9

O

`ok()` (biomajmanager.utils.Utils static method), 15

P

Plugins (class in biomajmanager.plugins), 11

R

`reset_timer()` (biomajmanager.utils.Utils static method), 15
RSS (class in biomajmanager.news), 9

S

`set_config()` (biomajmanager.plugins.BMPlugin method), 11
`set_manager()` (biomajmanager.plugins.BMPlugin method), 11
`start_timer()` (biomajmanager.utils.Utils static method), 15
`stop_timer()` (biomajmanager.utils.Utils static method), 15

T

`time2date()` (biomajmanager.utils.Utils static method), 15

`time2datefmt()` (biomajmanager.utils.Utils static method), [15](#)

U

`uprint()` (biomajmanager.utils.Utils static method), [15](#)

`user()` (biomajmanager.utils.Utils static method), [15](#)

`user_granted()` (in module biomajmanager.decorators), [3](#)

Utils (class in biomajmanager.utils), [13](#)

V

`verbose()` (biomajmanager.utils.Utils static method), [16](#)

W

`warn()` (biomajmanager.utils.Utils static method), [16](#)

`write()` (biomajmanager.writer.Writer method), [17](#)

Writer (class in biomajmanager.writer), [17](#)