
Version Manager

Feb 09, 2020

Modules:

1	Version Manager	1
2	Config	3
3	Date	5
4	ErrorCode	7
5	Git	9
6	Logger	11
7	Version	13
8	Version Emailer	17
9	Version File Generator	19
10	Indices and tables	21
	Python Module Index	23
	Index	25

CHAPTER 1

Version Manager

`version_manager.PrintHelp (argv: list, argc: int) → error_code.ErrorCode`

Print a help message on how to use the Version Manager.

Parameters

- **argv** (*list*) – The given arguments.
- **argc** (*int*) – The count of given arguments.

Returns An ErrorCode object telling what the outcome of calling the function was.


```
class config.Config
    Bases: object

    Parse and provide the config.json's data.

    config = None

    static InitConfig() → error_code.ErrorCode
        Read the data from the 'config.json' to initialize the static 'config' class variable, making it a dictionary
        representation of the 'config.json' file contents.

        Returns An error code from the ErrorCode class.

    static GetConfig() → dict
        Return the parsed config.json object as a dict.

        Returns The config file data as a dict.
```



```
class date.Date
```

```
    Bases: object
```

This class provides helpers for date stuff. For example, converting dates to strings and strings to dates.

```
ISO_8601_FORMAT = '%Y_%m_%d-%H_%M_%S'
```

```
GIT_STRING_FORMAT = '%a %b %d %H:%M:%S %Y %z'
```

```
static ConvertDateToString (x: datetime.datetime) → str
```

Convert the given date into a ISO 8601 formatted string.

```
static ConvertStringToDateWithFormat (customFormat: str, x: datetime.datetime) → datetime.datetime
```

Convert the given custom formatted string to a date object.

```
static ConvertGitStringToDate (x: str) → datetime.datetime
```

Convert the given string, which is formatted in the way Git formats dates, into a datetime object.

```
static ConvertStringToDate (x: str) → datetime.datetime
```

Convert the given ISO 8601 formatted string into a date.

```
static Now () → datetime.datetime
```

Get the current date and time information as an object of the datetime class.

```
static NowAsString () → str
```

Convert the current date and time information as an object of the datetime class into a string formatted in the ISO 8601 format.

CHAPTER 4

ErrorCode

```
class error_code.ErrorCode
    Bases: enum.IntEnum

    The error codes for this program.

    OK = 0
    UNKNOWN_COMMAND = 1
    MISSING_ARGUMENT = 2
    TOO_FEW_ARGUMENTS = 3
    FILE_ERROR = 4
    COMMAND_FAILED = 5
    SMTP_ERROR = 6
    UNKNOWN_ERROR = 666
```



```
class git.User
    Bases: object

    The class representation of a Git user.

    name
        The name of the user.

        Type str

    email
        The email of the user.

        Type str

class git.Commit
    Bases: object

    The class representation of a Git commit.

    hash
        The hash of the commit.

        Type str

    author
        The author of the commit.

        Type User

    date
        The timestamp of the commit.

        Type datetime.datetime

    title
        The title of the commit.

        Type str
```

message

The message of the commit.

Type str

```
class logger.Logger
```

```
    Bases: object
```

Provide an API for a logger for the program, which prints out logged messages to the terminal and saves the logged messages to a log file (if enabled in the ‘config.json’ and the file path is set).

```
    LOG_FILE = 'version_manager_{timestamp}.log'
```

```
    LOG_MESSAGE_FORMAT = '[{level}:{timestamp}:{tag}]: {message}'
```

```
    class LogLevel
```

```
        Bases: enum.IntEnum
```

The possible log levels for log messages. The higher the value, the more serious the log message.

```
        DEBUG = 0
```

```
        INFO = 1
```

```
        WARNING = 2
```

```
        ERROR = 3
```

```
    LogLevelToStrings = {<LogLevel.DEBUG: 0>: 'Debug', <LogLevel.INFO: 1>: 'Info', <LogLevel.ERROR: 3>: 'Error'}
```

```
    StringsToLogLevels = {'Debug': <LogLevel.DEBUG: 0>, 'Error': <LogLevel.ERROR: 3>, 'Info': <LogLevel.INFO: 1>}
```

```
    time = None
```

```
    logLevel = 0
```

```
    static Init()
```

```
    static LogToFile(message: str)
```

Log a given message to the log file path specified in the config.json.

Parameters *message* (str) – The message to log to file.

```
    static Error(tag: str, message: str)
```

Log an error message and, if enabled, log the error to a file.

Parameters

- **tag** (*str*) – The tag for the log message.
- **message** (*str*) – The message to log.

static Warning (*tag: str, message: str*)

Log a warning message and, if enabled, log the error to a file.

Parameters

- **tag** (*str*) – The tag for the log message.
- **message** (*str*) – The message to log.

static Info (*tag: str, message: str*)

Log an info message and, if enabled, log the error to a file.

Parameters

- **tag** (*str*) – The tag for the log message.
- **message** (*str*) – The message to log.

static Debug (*tag: str, message: str*)

Log a debug message and, if enabled, log the error to a file.

Parameters

- **tag** (*str*) – The tag for the log message.
- **message** (*str*) – The message to log.


```
class version.Version
    Bases: object
    A representation of the version tag.
    major
        The major version number of the version
        Type int
    minor
        The minor version number of the version
        Type int
    bug
        The bug version number of the version
        Type int
    stage
        The stage of the version
        Type Version.Stage
    stageRev
        The stage revision of the version
        Type int
class Stage
    Bases: enum.IntEnum
    The different stages the program can be set.
    UNKNOWN = -1
    DEVELOPMENT = 0
    RELEASE = 1
```

```
RELEASE_CANDIDATE = 2
ALPHA = 3
BETA = 4

StageStringsToStages = {'alpha': <Stage.ALPHA: 3>, 'beta': <Stage.BETA: 4>, 'dev':

static GetCurrentTag () → str
    Get the current tag.

    Returns The current tag.

static GetPreviousTag () → str
    Get the previous tag.

    Returns The previous tag.

static GetCurrentHash () → str
    Get the hash of the current Git commit.

    Returns The hash of the current commit.

static GetPreviousHash () → str
    Get the hash of the previous Git commit.

    Returns The hash of the previous commit.

static GetCommitsBetweenIds (newer: str, older: str) → list
    Get a list of commits between two Git commits.

    Parameters

    • newer (str) – The newer Git commit id for the comparison.

    • older (str) – The older Git commit id for the comparison.

    Returns A list of commits.

static GenerateVersionFromString (versionString: str)
    Create an instance of the Version class based on the tag string.

    Parameters versionString (str) – The Git tag in human readable format.

    Returns An instance of the Version class.

static PushTagsToOrigin () → error_code.ErrorCode
    Push existing Git tags to ‘origin’.

version.HandleDiffCommand (argv: list, argc: int) → error_code.ErrorCode
version.HandleHashCommand (argv: list, argc: int) → error_code.ErrorCode
version.HandleTagCommand (argv: list, argc: int) → error_code.ErrorCode
version.PrintHelpMessageGet (argv: list, argc: int) → error_code.ErrorCode
version.HandleGetCommand (argv: list, argc: int) → error_code.ErrorCode
version.PrintHelpMessage (argv: list, argc: int) → error_code.ErrorCode
version.HandleCommand (argv: list, argc: int) → error_code.ErrorCode
    Handle a command given to this module

    Parameters

    • argv (list) – The given arguments.

    • argc (int) – The count of given arguments.
```

Returns An ErrorCode object telling what the outcome of calling the function was.


```
class version_emailer.HTMLEmail
```

Bases: object

```
static Send(templateFilePath: str, commits: list) → error_code.ErrorCode
```

Send an HTML email of the given commits in the style of the given template file. Use the email settings (subject, to, from) from the config.json.

Parameters

- **templateFilePath** (*str*) – The path to the email template file.
- **commits** (*list*) – The list of commits to list in the email.

Returns An ErrorCode object telling what the outcome of calling the function was.

```
class version_emailer.TextEmail
```

Bases: object

```
static Send(templateFilePath: str, commits: list) → error_code.ErrorCode
```

Send a text email of the given commits in the style of the given template file. Use the email settings (subject, to, from) from the config.json.

Parameters

- **templateFilePath** (*str*) – The path to the email template file.
- **commits** (*list*) – The list of commits to list in the email.

Returns An ErrorCode object telling what the outcome of calling the function was.

```
version_emailer.HandleCommand(argv: list, argc: int) → error_code.ErrorCode
```

Handle a command given to this module

Parameters

- **argv** (*list*) – The given arguments.
- **argc** (*int*) – The count of given arguments

Returns An ErrorCode object telling what the outcome of calling the function was.

Version File Generator

```
version_file_generator.GenerateVersionFileFromVersion (version:      version.Version,  
                                                         templateFilePath:      str,  
                                                         versionFilePath:      str) →  
                                                         error_code.ErrorCode
```

Generates a version file from a version object.

Parameters

- **version** (*Version*) – An instance of the Version class, which is an object representation of the version tag string.
- **templateFilePath** (*str*) – Path to the template file for a version file.
- **versionFilePath** (*str*) – The path to the version source file which is to be generated.

Returns An ErrorCode object telling what the outcome of calling the function was.

```
version_file_generator.HandleCommand (argv: list, argc: int) → error_code.ErrorCode
```

Handle a command given to this module

Parameters

- **argv** (*list*) – The given arguments.
- **argc** (*int*) – The count of given arguments.

Returns An ErrorCode object telling what the outcome of calling the function was.

CHAPTER 10

Indices and tables

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

c

`config`, [3](#)

d

`date`, [5](#)

e

`error_code`, [7](#)

g

`git`, [9](#)

l

`logger`, [11](#)

v

`version`, [13](#)

`version_emailer`, [17](#)

`version_file_generator`, [19](#)

`version_manager`, [1](#)

A

ALPHA (*version.Version.Stage* attribute), 14
 author (*git.Commit* attribute), 9

B

BETA (*version.Version.Stage* attribute), 14
 bug (*version.Version* attribute), 13

C

COMMAND_FAILED (*error_code.ErrorCode* attribute), 7
 Commit (*class in git*), 9
 Config (*class in config*), 3
 config (*config.Config* attribute), 3
 config (*module*), 3
 ConvertDateToString() (*date.Date* static method), 5
 ConvertGitStringToDate() (*date.Date* static method), 5
 ConvertStringToDate() (*date.Date* static method), 5
 ConvertStringToDateWithFormat() (*date.Date* static method), 5

D

Date (*class in date*), 5
 date (*git.Commit* attribute), 9
 date (*module*), 5
 DEBUG (*logger.Logger.LogLevel* attribute), 11
 Debug() (*logger.Logger* static method), 12
 DEVELOPMENT (*version.Version.Stage* attribute), 13

E

email (*git.User* attribute), 9
 ERROR (*logger.Logger.LogLevel* attribute), 11
 Error() (*logger.Logger* static method), 11
 error_code (*module*), 7
 ErrorCode (*class in error_code*), 7

F

FILE_ERROR (*error_code.ErrorCode* attribute), 7

G

GenerateVersionFileFromVersion() (*in module version_file_generator*), 19
 GenerateVersionFromString() (*version.Version* static method), 14
 GetCommitsBetweenIds() (*version.Version* static method), 14
 GetConfig() (*config.Config* static method), 3
 GetCurrentHash() (*version.Version* static method), 14
 GetCurrentTag() (*version.Version* static method), 14
 GetPreviousHash() (*version.Version* static method), 14
 GetPreviousTag() (*version.Version* static method), 14
 git (*module*), 9
 GIT_STRING_FORMAT (*date.Date* attribute), 5

H

HandleCommand() (*in module version*), 14
 HandleCommand() (*in module version_email*), 17
 HandleCommand() (*in module version_file_generator*), 19
 HandleDiffCommand() (*in module version*), 14
 HandleGetCommand() (*in module version*), 14
 HandleHashCommand() (*in module version*), 14
 HandleTagCommand() (*in module version*), 14
 hash (*git.Commit* attribute), 9
 HTMLEmail (*class in version_email*), 17

I

INFO (*logger.Logger.LogLevel* attribute), 11
 Info() (*logger.Logger* static method), 12
 Init() (*logger.Logger* static method), 11
 InitConfig() (*config.Config* static method), 3
 ISO_8601_FORMAT (*date.Date* attribute), 5

L

LOG_FILE (*logger.Logger* attribute), 11

LOG_MESSAGE_FORMAT (*logger.Logger* attribute), 11
Logger (*class in logger*), 11
logger (*module*), 11
Logger.LogLevel (*class in logger*), 11
logLevel (*logger.Logger* attribute), 11
LogLevelsToStrings (*logger.Logger* attribute), 11
LogToFile() (*logger.Logger* static method), 11

M

major (*version.Version* attribute), 13
message (*git.Commit* attribute), 9
minor (*version.Version* attribute), 13
MISSING_ARGUMENT (*error_code.ErrorCode* attribute), 7

N

name (*git.User* attribute), 9
Now() (*date.Date* static method), 5
NowAsString() (*date.Date* static method), 5

O

OK (*error_code.ErrorCode* attribute), 7

P

PrintHelp() (*in module version_manager*), 1
PrintHelpMessage() (*in module version*), 14
PrintHelpMessageGet() (*in module version*), 14
PushTagsToOrigin() (*version.Version* static method), 14

R

RELEASE (*version.Version.Stage* attribute), 13
RELEASE_CANDIDATE (*version.Version.Stage* attribute), 13

S

Send() (*version_emailer.HTMLEmail* static method), 17
Send() (*version_emailer.TextEmail* static method), 17
SMTP_ERROR (*error_code.ErrorCode* attribute), 7
stage (*version.Version* attribute), 13
stageRev (*version.Version* attribute), 13
StageStringsToStages (*version.Version* attribute), 14
StringsToLogLevels (*logger.Logger* attribute), 11

T

TextEmail (*class in version_emailer*), 17
time (*logger.Logger* attribute), 11
title (*git.Commit* attribute), 9
TOO_FEW_ARGUMENTS (*error_code.ErrorCode* attribute), 7

U

UNKNOWN (*version.Version.Stage* attribute), 13
UNKNOWN_COMMAND (*error_code.ErrorCode* attribute), 7
UNKNOWN_ERROR (*error_code.ErrorCode* attribute), 7
User (*class in git*), 9

V

Version (*class in version*), 13
version (*module*), 13
Version.Stage (*class in version*), 13
version_emailer (*module*), 17
version_file_generator (*module*), 19
version_manager (*module*), 1

W

WARNING (*logger.Logger.LogLevel* attribute), 11
Warning() (*logger.Logger* static method), 12