Version Manager

Feb 09, 2020

Modules:

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.

CHAPTER 2

Config

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.

CHAPTER 3

Date

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

CHAPTER 5

Git

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

CHAPTER 6

Logger

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

LogLevelsToStrings = {<LogLevel.DEBUG: 0>: 'Debug', <LogLevel.INFO: 1>: 'Info', <Log

StringsToLogLevels = {'Debug': <LogLevel.DEBUG: 0>, 'Error': <LogLevel.ERROR: 3>, 'I

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.

CHAPTER 7

Version

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.

CHAPTER 8

Version Emailer

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.

CHAPTER 9

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 generaated.

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

Python Module Index

Index

M

N

O

P

R

S

T

U

V

W