clashcallerbot-reddit Documentation

Release 2.1.3

Jose A. Lerma III

Nov 04, 2018

CONTENTS:

CHAPTER

ONE

CLASHCALLERBOT-REDDIT

1.1 clashcallerbot_database module

Setup the MySQL-compatible database.

If run directly, this module will setup the clashcallerbot database with tables and display their format and contents. Additionally, this module provides a class with various methods for managing the MySQL-compatible database:

- Create database and tables.
- View table data and properties.
- Grant user permissions (if logged into database as root).
- Add rows to tables.
- Delete tables and rows.
- Convert python datetime to MySQL datetime.

class clashcallerbot_database.**ClashCallerDatabase**(*config_file=None*, *root_user=None*)

Bases: object

Implements a class for a ClashCaller Database.

config_file
A configparser object with database.ini file pre-read.

Type configparser.ConfigParser()

root_user
Specifies whether the database will be setup as root user.

Type bool

mysql_connection
A mysql.connector.connect() object.

Type mysql.connect()

cursor
A mysql.connector.connect().cursor() object.

Type mysql.connect()cursor()

close_connections() → None
Close database connections.

Method closes database cursor and connection.

Returns None

static convert_datetime(*dt: <module 'datetime' from '/usr/lib/python3.6/datetime.py'>*) → <module 'datetime' from '/usr/lib/python3.6/datetime.py'>

Converts python datetime to MySQL datetime.

Method converts given python datetime object to MySQL datetime format.

Parameters **dt** – Datetime object in default format.

Returns Datetime object in MySQL format.

create_database() → bool

Create database

Method creates database with database name.

Returns True if successful, False otherwise.

create_table(*tbl_name: str*, *cols: str*) → bool

Create table in database.

Method creates table in database with given name and specifications.

Parameters

- **tbl_name** – Name to give table.
- **cols** – Columns to put in table.

Example

```
>>> tbl_name = 'table'
>>> cols = 'id INT UNSIGNED NOT NULL AUTO_INCREMENT, '
...         'permalink VARCHAR(100), message VARCHAR(100), new_date DATETIME, '
...         'userID VARCHAR(20), PRIMARY KEY(id)'
...
>>> create_table(tbl_name, cols)
```

Returns True if successful, False otherwise.

delete_message(*tid: str*) → bool

Deletes message from message_data table.

Method deletes given table id (row) from message_data table.

Parameters **tid** – Table id from id column of message_data table.

Returns True for success, False otherwise.

describe_table(*tbl_name: str*) → list

Gets description of table.

Method returns a list describing the structure of the given table.

Parameters **tbl_name** – Name of table to describe

Returns List with table description, empty list otherwise.

drop_table(*tbl_name: str*) → bool

Drop table from database.

Function drops given table from given database.

Parameters **tbl_name** – Table to drop.

Returns True if successful, False otherwise.

find_comment_id(*cid: str*) → bool
Check comment_list table for comment id.

Method checks comment_list table for given comment id.

Parameters **cid** – Comment id to search for.

Returns True for success, false otherwise.

get_messages(*time_now: datetime.datetime*) → list
Retrieves list of messages that have expired.

Method returns list of messages whose expiration times are before current datetime.

Parameters **time_now** – Current datetime.

Returns List containing results of query.

get_rows(*tbl_name: str*) → tuple
Fetch table rows.

Method gets rows of given table by order of id in a tuple.

Parameters **tbl_name** – Name of table to get rows from.

Returns Tuple containing each row's data, empty tuple otherwise.

get_tables() → list
Return table list of database.

Method returns a list with the names of the tables.

Returns List of table names.

grant_permissions() → bool
Grants bot user permissions to database.

Method grants bot user permissions to database.

Returns True if successful, False otherwise.

Notes

Only database root user can grant database permissions.

lock_read(*tbl_name: str*) → bool
Locks table for reading.

Method locks a given table for read access.

Parameters **tbl_name** – Name of table to lock.

Returns True if successful, False otherwise.

Notes

- Any previous locks are implicitly released.
- Read locks have lower priority than write locks.

lock_write(*tbl_name: str*) → bool
Locks table for writing.

Method locks a given table for write access.

Parameters **tbl_name** – Name of table to lock.

Returns True if successful, False otherwise.

Notes

- Any previous locks are implicitly released.
- Write locks have higher priority than read locks.

save_comment_id(*cid: str*) → bool
Saves comment id into comment_list table.

Method saves given comment id into the comment_list table.

Parameters **cid** – Comment id to save.

Returns True for success, false otherwise.

save_message(*link: str*, *msg: str*, *exp: <module 'datetime' from '/usr/lib/python3.6/datetime.py'>*, *usr_name: str*) → bool
Saves given comment data into message_data table.

Method saves given inputs in message_date table as a row.

Parameters

- **link** – Comment permalink.
- **msg** – Comment message.
- **exp** – Expiration datetime object.
- **usr_name** – Comment author username.

Returns True for success, false otherwise.

select_database() → bool
Select database for command execution.

Method selects database within MySQL for command execution.

Returns True if successful, False otherwise.

clashcallerbot_database.**main**()

1.2 clashcallerbot_reply module

Checks messages in database and sends PM if expiration time passed.

This module checks messages saved in a MySQL-compatible database and sends a reminder via PM if the expiration time has passed. If so, the message is removed from the database.

clashcallerbot_reply.**get_parent**(*link: str*) → str
Fetch parent comment or submission.

Function gets parent comment of given permalink or submission if top level comment.

Parameters **link** – Permalink to get parent of.

Returns Parent comment link or default string.

clashcallerbot_reply.**main**()

clashcallerbot_reply.**send_reminder**(*link: str*, *msg: str*, *usr: str*) → bool
Sends reminder PM to username.

Function sends given permalink and message to given username.

Parameters

- **link** – Permalink to comment.
- **msg** – Message in comment that was saved.
- **usr** – User to send reminder to.

Returns True if successful, False otherwise.

1.3 clashcallerbot_search module

Searches recent reddit comments for ClashCaller! string and saves to database.

This module uses the Python Reddit API Wrapper (PRAW) to search recent reddit comments for the ClashCaller! string.

If found, the userID, permalink, comment time, message, and expiration time (if any) are parsed. The default, or provided, expiration time is applied, then all the comment data is saved to a MySQL-compatible database.

The comment is replied to, then the userID is PM'ed confirmation.

clashcallerbot_search.**have_replied**(*cid: str*, *bot_name: str*) → bool
Checks if bot user has replied to a comment.

Function checks reply authors for bot user.

Parameters

- **cid** – Comment ID to get replies of.
- **bot_name** – Name of bot to check for.

Returns True if successful, False otherwise.

clashcallerbot_search.**is_recent**(*cmnt: praw.models.reddit.comment.Comment*) → bool
Checks if comment is a recent comment.

Function compares comment time with program start time.

Parameters **cmnt** – praw comment object.

Returns True if comment time is after start time, False otherwise.

clashcallerbot_search.**main**()

clashcallerbot_search.**send_confirmation**(*u_name: str*, *link: str*, *exp: datetime.datetime*) → bool
Send confirmation to reddit user.

Function sends given user confirmation of given expiration time with given link.

Parameters

- **u_name** – username of user.

- **link** – Permalink of comment.
- **exp** – Expiration datetime of call.

Returns True if successful, False otherwise.

clashcallerbot_search.**send_confirmation_reply**(*cid: str*, *link: str*, *exp: datetime.datetime*) → str

Replies to a comment.

Function replies to a given comment ID with a given message.

Parameters

- **cid** – Comment ID to reply to.
- **link** – Permalink of comment.
- **exp** – Expiration datetime of call.

Returns id of new comment if successful, None otherwise

clashcallerbot_search.**send_error_message**(*u_name: str*, *link: str*, *error: str*) → bool

Send error message to reddit user.

Function sends given error to given user.

Parameters

- **u_name** – username of user.
- **link** – Permalink of comment.
- **error** – Error to send to user.

Returns True if successful, False otherwise.

1.4 logging_conf module

Defines logging dictionary.

Module defines dictionary for logging.config.dictConfig()

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

C

l

INDEX