Dennis MUD

Release PreAlpha

Michael D. Reiley

Jun 28, 2020

CONTENTS:

CHAPTER

ONE

INTRODUCTION

CHAPTER

TWO

GENERAL USAGE

CHAPTER

THREE

GAMEPLAY TUTORIAL

CHAPTER

FOUR

COMMAND REFERENCE

CHAPTER

FIVE

DEVELOPING COMMANDS

CHAPTER

SIX

DENNIS MUD API REFERENCE

6.1 Database Format

6.2 DatabaseManager

class database.**DatabaseManager**(*filename*, *log=None*)

The Database Manager

This manager handles interactions with a TinyDB database corresponding to the current game world. After documents are pulled from a table and modified, they need to be upserted for the changes to save.

Variables

- **database** – The TinyDB database instance for the world.
- **rooms** – The table of all rooms in the database.
- **users** – The table of all users in the database.
- **items** – The table of all items in the database.
- **defaults** – The JSON database defaults configuration.

__init__(*filename*, *log=None*)

Database Manager Initializer

Parameters

- **filename** – The relative or absolute filename of the TinyDB database file.
- **log** – Alternative logging facility, if set.

delete_item(*document*)

Delete an item.

Parameters **document** – The item document to delete.

Returns True

delete_room(*document*)

Delete a room.

Parameters **document** – The room document to delete.

Returns True

delete_user(*document*)

Delete a user.

This is not safe to call while the user is online.

Parameters **document** – The user document to delete.

Returns True

item_by_id(*itemid*)
Get an item by its id.

Parameters **itemid** – The id of the item to retrieve from the database.

Returns Item document or None.

login_user(*username*, *passhash*)
Check if a username and password match an existing user, and log them in.

The Database Manager keeps track of which users are online.

Parameters

- **username** – The name of the user to log in.
- **passhash** – The hashed password of the user to log in.

Returns User document if succeeded, None if failed.

logout_user(*username*)
Log out a user.

Parameters **username** – The name of the user to log out.

Returns True.

online(*username*)
Check if a user is online.

Parameters **username** – The name of the user to check.

Returns True if online, False if offline.

room_by_id(*roomid*)
Get a room by its id.

Parameters **roomid** – The id of the room to retrieve from the database.

Returns Room document or None.

upsert_item(*document*)
Update or insert an item.

Parameters **document** – The item document to update or insert.

Returns True

upsert_room(*document*)
Update or insert a room.

Parameters **document** – The room document to update or insert.

Returns True

upsert_user(*document*)
Update or insert a user.

Parameters **document** – The user document to update or insert.

Returns True

user_by_name(*username*)
Get a user by their name.

If there is any chance the user could be logged in, and the record needs to be altered, you should use the equivalent Console method. This method is faster but unsafe for logged in users.

Parameters **username** – The name of the user to retrieve from the database.

Returns User document or None.

user_by_nick(*nickname*)
Get a user by their nickname.

If there is any chance the user could be logged in, and the record needs to be altered, you should use the equivalent Console method. This method is faster but unsafe for logged in users.

Parameters **nickname** – The nickname of the user to retrieve from the database.

Returns User document or None.

6.3 Console

class console.**Console**(*database*, *rname*, *router*, *log=None*)
Each instance of the console corresponds to a single user session. The console handles user-command interaction.

Variables

- **user** – TinyDB user document for the currently logged in user, if any.
- **rname** – The name used by the router for this console.
- **router** – The Router instance, which handles interfacing between the server backend and the user consoles.

__init__(*database*, *rname*, *router*, *log=None*)
Console Initializer

Parameters

- **database** – The DatabaseManager instance to use.
- **rname** – The name used by the router for this console.
- **router** – The Router instance, which handles interfacing between the server backend and the user consoles.
- **log** – Alternative logging facility, if set.

broadcast(*message*)
Broadcast Message

Send a message to all users connected to consoles.

Parameters **message** – The message to send.

Returns True

broadcast_room(*message*)
Broadcast Message to Room

Send a message to all users who are in the same room as the user connected to this console.

Parameters **message** – The message to send.

Returns True

command(*line*, *show_command=True*)
Command Handler

Parse and execute a command line, discerning which arguments are part of the command name and which arguments should be passed to the command.

Parameters

- **line** – The command line to parse.
- **show_command** – Whether or not to echo the command being executed in the console.

Returns Command result or None.

help(*line*)
Help Handler

Retrieve the help for a category or command.

Parameters **line** – The help line to parse.

Returns True if succeeded, False if failed.

msg(*message*, *_nbsp=False*)
Send Message

Send a message to the user connected to this console.

Parameters

- **message** – The message to send.
- **_nbsp** – Will insert non-breakable spaces for formatting on the websocket frontend.

Returns True

usage(*line*)
Usage Handler

Retrieve just the usage string for a command.

Parameters **line** – The help line to parse.

Returns True if succeeded, False if failed.

user_by_name(*username*)
Get a user by their name.

It is necessary to modify user records through the console before updating the database, if they are logged in. Otherwise their database record will be overwritten next time something is changed in their console record. If the user is logged in, return their console record. Otherwise, return their record directly from the database.

Returns Console or Database User Document, or None

user_by_nick(*nickname*)
Get a user by their nickname.

It is necessary to modify user records through the console before updating the database, if they are logged in. Otherwise their database record will be overwritten next time something is changed in their console record. If the user is logged in, return their console record. Otherwise, return their record directly from the database.

Returns Console or Database User Document, or None

6.4 Server/Router

class server.**Router**(*config*, *dbman*)

This class handles interfacing between the server backends and the user command consoles. It manages a lookup table of connected users and their consoles, and handles passing messages between them.

Variables

- **users** – Dictionary of connected users and their consoles, as well as the protocols they are connected by.
- **single_user** – Whether we are running in single-user mode. Hard-coded here to False.
- **telnet_factory** – The active Autobahn telnet server factory.
- **websocket_factory** – The active Autobahn websocket server factory.

__init__(*config*, *dbman*)

Router Initializer

broadcast_all(*msg*)

Broadcast All

Broadcast a message to all logged in users.

Parameters **msg** – Message to send.

Returns True

broadcast_room(*room*, *msg*)

Broadcast Room

Broadcast a message to all logged in users in the given room.

Parameters

- **room** – Room ID.
- **msg** – Message to send.

Returns True

message(*peer*, *msg*, *_nbsp=False*)

Message Peer

Message a user by their internal peer name.

Parameters

- **peer** – Internal peer name.
- **msg** – Message to send.
- **_nbsp** – Will insert non-breakable spaces for formatting on the websocket frontend.

Returns True

register(*peer*, *service*)

Register User

Parameters

- **peer** – Internal peer name.
- **service** – Service type. “telnet” or “websocket”.

Returns True

unregister(*peer*)
Unregister and Logout User

Parameters **peer** – Internal peer name.

Returns True if succeeded, False if no such user.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- search

INDEX

Symbols

B

C

D

H

I

L

M

O

R

U