DiscordDB

Release 0.0.6

| The Cosmos

Feb 13, 2022

CONTENTS

A simple database which uses a Discord channel to store data.

Features

- Supports string and integer data types.
- JSON like data structure.
- Store unlimited data documents without worrying about the disk space.
- Reliably fast accessing of data documents.
- Clean object oriented design.
- Makes easier to store bulk data which are generally not much important.

CHAPTER

ONE

INTRODUCTION

DiscordDB acts like a simple database which uses a discord channel to store data which are generally not much important and you do not care about in an unlimited manner.

1.1 Requirements

discord.py It requires `discord.py` as a main wrapper to post or get data from discord channel.

1.2 Installing

You can install DiscordDB directly from PyPI using PIP and following command in shell or command prompt:

```
python3 -m pip install -U DiscordDB
```

You can also install the latest development version (**maybe unstable/broken**) by using following command:

```
python3 -m pip install -U git+https://github.com/thec0sm0s/DiscordDB.git
```

1.3 Basic Usage

```python
from discordDB import DiscordDB
from discord.ext import commands


LOGS = []
DATABASE_CHANNEL_ID = 399397622743564297


class MyBot(commands.Bot):

    def __init__(self):
        super().__init__(command_prefix="!")
        self.discordDB = DiscordDB(self, DATABASE_CHANNEL_ID)

    @commands.command()
    async def log(self, ctx, *, text):
```

(continues on next page)

(continued from previous page)

```
        data = {
            "name": ctx.author,
            "text": text
        }
        _id = await self.discordDB.set(data)
        LOGS.append(_id)

    @commands.command()
    async def show_logs(self, ctx):
        for _id in LOGS:
            data = await self.discordDB.get(_id)
            await ctx.send(f"Name: {data.name}, Text: {data.text}")


bot = MyBot()
bot.run("TOKEN")
```

CHAPTER

TWO

API REFERENCE

Reference to all of available classes, methods, attributes, etc.

2.1 DiscordDB Client

`class discordDB.`**`DiscordDB`**`(`*bot*, *db_channel_id: int*`)`

The Discord database client.

Parameters

- **`bot`** (*`discord.ext.commands.Bot`*) – An instance of discord.py Client or Bot representing your discord application.
- **`db_channel_id`** (*`int`*) – An integer representing ID of Discord channel you want to be used as database.

`property channel`

A property which returns an instance of `discord.TextChannel` which is being used as database.

`async get``(`*_id: int*`)` → *discordDB.models.types.Data*

A method used to get your saved data from the database channel.

Parameters **`_id`** (*`int`*) – An special integer which was received from the *`discordDB.DiscordDB.set()`* method.

Returns An instance of *`discordDB.models.Data`*, similar to python dictionaries but also supports accessing of its key using . syntax.

Return type *Data*

`async set``(`*data: dict*`)` → int

A method to post and save data to your database channel.

Parameters **`data`** (*`dict`*) – Dictionary representing your raw data.

Returns An special integer which should be saved by the client to get this same data later.

Return type int

2.2 Data Types

`class discordDB.models.Data`

Actually a superset class of python dictionaries, which also supports accessing of its keys using . syntax.

created_at [datetime.datetime] The time this data was created in UTC.

CHAPTER THREE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX