dathost

Release 1.0.1

WardPearce

Oct 03, 2021

CONTENTS

This is a unofficial asynchronous & synchronous wrapper for Dathost's API.

Features:

- Full API coverage.
- Asynchronous & synchronous support.
- Easy to use with an object oriented design.
- 100% test coverage.

CHAPTER

ONE

INSTALL

Pip:

```
pip3 install dathost
```

Git:

```
pip3 install git+https://github.com/WardPearce/dathost.git
```

CHAPTER

TWO

DOCUMENTATION CONTENTS

2.1 Intro

This wrapper has both asynchronous & synchronous support, this intro will cover the basic of both. Lucily for you the API for asynchronous (awaiting) & synchronous (blocking) is identical.

2.1.1 Non-context managers

Awaiting client

```
import dathost

client = dathost.Awaiting(
    email="wardpearce@protonmail.com",
    password="..."
)

# A client should always be closed after being used!
await client.close()
```

Blocking client

```
import dathost

client = dathost.Blocking(
    email="wardpearce@protonmail.com",
    password="..."
)

# A client should always be closed after being used!
client.close()
```

2.1.2 Context managers

Blocking

```
import dathost

with dathost.Blocking(EMAIL, PASSWORD) as client:
    pass
```

Awaiting

```
import dathost

async with dathost.Awaiting(EMAIL, PASSWORD) as client:
    pass
```

2.2 Examples

Here are some simple examples on how to use this wrapper. This is written using the blocking wrapper, but still applies to the awaiting wrapper.

Assume that "client" has been initialized.

2.2.1 Creating a server

```
from dathost.settings import ServerSettings

data, server = client.create_server(
    ServerSettings(
        name="CS: GO server",
        location="sydney",
    ).csgo(
        slots=5,
        game_token="",
        tickrate=128,
        rcon_password=""
    )
)

server.start()
print(data.slots)
```

2.2.2 Creating a match

```
from dathost.settings import MatchSettings

data, match = server.create_match(
    MatchSettings(
        connection_time=60,
    ).team_1(
        [
            "[U:1:116962485]",
            76561198017567105,
            "STEAM_0:1:186064092"
        ]
    ).team_2(
        [
            "[U:1:320762620]",
            "STEAM_0:1:83437164",
            76561198214871324
        ]
    ).spectators(
        [
            "[U:1:320762620]",
            "STEAM_0:1:83437164",
            76561198214871324
        ]
    )
)

# Don't worry about steam IDs, the wrapper
# ensures they're all converted correctly.
```

2.3 API

2.3.1 Awaiting

class dathost.**Awaiting**

await account() → *dathost.models.account.AccountModel*
Gets account details

Returns Holds data on a account.

Return type *AccountModel*

await close() → None
Closes sessions

await create_server(*settings:* dathost.settings.ServerSettings) →
Tuple[*dathost.models.server.ServerModel*, *dathost.server.awaiting.ServerAwaiting*]
Creates a new server.

Parameters **settings** (ServerSettings) – Used to configure server.

Returns

- *ServerModel* – Holds data on server.
- *ServerAwaiting* – Used to interact with the created server.

async for ... in domains() → AsyncGenerator[str, None]
Used to list domains.

Returns List of domains.

Return type list

match(*match_id: str*) → *dathost.match.awaiting.AwaitingMatch*
Used to interact with a match.

Parameters **match_id** (*str*) – Dathost Match ID.

Returns

Return type *AwaitingMatch*

server(*server_id: str*) → *dathost.server.awaiting.ServerAwaiting*
Used for interacting with a server.

Parameters **server_id** (*str*) – Datahost server ID.

Returns Used to interact with the server.

Return type *ServerAwaiting*

async for ... in servers() → AsyncGenerator[Tuple[*dathost.models.server.ServerModel*, *dathost.server.awaiting.ServerAwaiting*], None]
Used to list servers.

Yields *ServerModel* – Holds data on server.

Server

class dathost.server.awaiting.**ServerAwaiting**

backup(*backup_name: str*) → *dathost.server.awaiting.backup.AwaitingBackup*
Used to interact with a backup.

Parameters **backup_name** (*str*) – Name of backup.

Returns

Return type *AwaitingBackup*

async for ... in backups() → AsyncGenerator[Tuple[*dathost.models.backup.BackupModel*, *dathost.server.awaiting.backup.AwaitingBackup*], None]
Used to list backups a server has.

Yields

- *BackupModel* – Holds details on backup.
- *AwaitingBackup* – Used for interacting with a backup.

await console_retrive(*lines: int = 1000*) → list
Used to retrive lines from the console.

Parameters **lines** (*int, optional*) – Amount of lines to retrive, by default 1000

Returns List of strings.

Return type list

Raises ***InvalidConsoleLine*** – Raised when console lines below 1 or above 100000.

await console_send(*line: str*) → None
Used to send a rcon command to console.

Parameters **line** (*str*) – Console command.

await create_match(*match_settings:* dathost.settings.MatchSettings) → Tuple[*dathost.models.match.MatchModel*, *dathost.match.awaiting.AwaitingMatch*]
Creates a match.

Parameters **match_settings** (MatchSettings) – Holds details on the match.

Returns

- *MatchModel* – Holds match details.
- *AwaitingMatch* – Used to interact with a match.

await delete() → None
Used to delete a sever.

await duplicate(*sync: bool = False*) → Tuple[*dathost.models.server.ServerModel*, *dathost.server.awaiting.ServerAwaiting*]
Used to duplicate a server.

Parameters **sync** (*bool*) – Used to force update server cache, by default False

Returns

- *ServerModel* – Holds server data.
- *ServerAwaiting* – Used to interact with server.

file(*pathway: str*) → *dathost.server.awaiting.file.AwaitingFile*
Used to interact with a file on the server.

Parameters **pathway** (*str*) – Pathway of file on server.

Returns

Return type *AwaitingFile*

async for ... in files(*hide_default: bool = False*, *path: Optional[str] = None*, *file_sizes: bool = False*, *deleted_files: bool = False*) → AsyncGenerator[Tuple[*dathost.models.file.FileModel*, *dathost.server.awaiting.file.AwaitingFile*], None]
Used to list files.

Parameters

- **hide_default** (*bool, optional*) – by default False
- **path** (*str, optional*) – Path to use as root, by default None
- **file_sizes** (*bool, optional*) – by default False
- **deleted_files** (*bool, optional*) – Include deleted files in list, by default False

Yields

- *FileModel* – Holds details on a file.
- *AwaitingFile* – Used to interact with a file.

await ftp_reset() → None
Resets the FRP password.

await get() → *dathost.models.server.ServerModel*
Used to get details on server.

Returns Holds data on server.

Return type *ServerModel*

await metrics() → *dathost.models.metrics.MetricsModel*
Used to get server metrics.

Returns Holds details on server metrics.

Return type *MetricsModel*

await reset() → None
Used to reset the server.

await start(*allow_host_reassignment: bool = True*) → None
Used to start the server.

Parameters **allow_host_reassignment** (*bool, optional*) – By default True

await stop() → None
Used to stop the server.

await sync() → None
Used to sync files from server to cache.

await update(*settings:* dathost.settings.ServerSettings) → None
Update servers paramters.

Parameters **settings** (ServerSettings) – Used to configure server.

Match

class dathost.match.awaiting.**AwaitingMatch**

await get() → *dathost.models.match.MatchModel*
Gets details on a match

Returns Holds match details.

Return type *MatchModel*

Backup

class dathost.server.awaiting.backup.**AwaitingBackup**

await restore() → None
Used to restore a backup.

File

class dathost.server.awaiting.file.**AwaitingFile**

await delete() → None
Deletes file.

await dowload() → bytes
Used to download a file into memory.

Returns

Return type bytes

Notes

Its reccomened to use download_iterate for large files.

async for ... in download_iterate() → AsyncGenerator[bytes, None]
Asynchronously downloads data into memory.

Yields *bytes*

await move(*destination: str*) → None
Used for moving a file.

Parameters **destination** (*str*) –

Notes

When called the file_path changes to the given destination.

await save(*local_pathway: str*) → None
Saves file to local pathway.

Parameters **local_pathway** (*str*) – Pathway to save file to.

await unzip(*destination: str*) → None
Used to unzip a file.

Parameters **destination** (*str*) –

await upload(*data: Optional[bytes] = None*) → None
Used for uploading raw bytes.

Parameters **data** (*bytes*) – Data to upload.

await upload_file(*local_pathway: str*) → None
Used to upload a local file.

Parameters **local_pathway** (*str*) – Local file to upload.

2.3.2 Blocking

class dathost.**Blocking**

account() → *dathost.models.account.AccountModel*
Gets account details

Returns Holds data on a account.

Return type *AccountModel*

close() → None
Closes sessions

create_server(*settings:* dathost.settings.ServerSettings) → Tuple[*dathost.models.server.ServerModel*, *dathost.server.blocking.ServerBlocking*]
Creates a new server.

Parameters **settings** (ServerSettings) – Used to configure server.

Returns

- *ServerModel* – Holds data on server.
- *ServerBlocking* – Used to interact with the created server.

for ... in domains() → Generator[str, None, None]
Used to list domains.

Returns List of domains.

Return type list

match(*match_id: str*) → *dathost.match.blocking.BlockingMatch*
Used to interact with a match.

Parameters **match_id** (*str*) – Dathost Match ID.

Returns

Return type *BlockingMatch*

server(*server_id: str*) → *dathost.server.blocking.ServerBlocking*
Used for interacting with a server.

Parameters **server_id** (*str*) – Datahost server ID.

Returns Used to interact with the server.

Return type *ServerBlocking*

for ... in servers() → Generator[Tuple[*dathost.models.server.ServerModel*, *dathost.server.blocking.ServerBlocking*], None, None]
Used to list servers.

Yields

- *ServerModel* – Holds data on server.
- *ServerBlocking* – Used to interact with server.

Server

class dathost.server.blocking.**ServerBlocking**

backup(*backup_name: str*) → *dathost.server.blocking.backup.BlockingBackup*
Used to interact with a backup.

Parameters **backup_name** (*str*) – Name of backup.

Returns

Return type *BlockingBackup*

for ... in backups() → Generator[Tuple[*dathost.models.backup.BackupModel*, *dathost.server.blocking.backup.BlockingBackup*], None, None]
Used to list backups a server has.

Yields

- *BackupModel* – Holds details on backup.
- *Backup* – Used for interacting with a backup.

console_retrive(*lines: int = 1000*) → list
Used to retrive lines from the console.

Parameters **lines** (*int, optional*) – Amount of lines to retrive, by default 1000

Returns List of strings.

Return type list

Raises ***InvalidConsoleLine*** – Raised when console lines below 1 or above 100000.

console_send(*line: str*) → None
Used to send a command to console.

Parameters **line** (*str*) – Console command.

create_match(*match_settings:* dathost.settings.MatchSettings) → Tuple[*dathost.models.match.MatchModel*, *dathost.match.blocking.BlockingMatch*]
Creates a match.

Parameters **match_settings** (MatchSettings) – Holds details on the match.

Returns

- *MatchModel* – Holds match details.
- *BlockingMatch* – Used to interact with a match.

delete() → None
Used to delete a sever.

duplicate(*sync: bool = False*) → Tuple[*dathost.models.server.ServerModel*, *dathost.server.blocking.ServerBlocking*]
Used to duplicate a server.

Parameters **sync** (*bool*) – Used to force update server cache, by default False

Returns

- *ServerModel* – Holds server data.
- *ServerBlocking* – Used to interact with server.

file(*pathway: str*) → *dathost.server.blocking.file.BlockingFile*
Used to interact with a file on the server.

Parameters **pathway** (*str*) – Pathway of file on server.

Returns

Return type *BlockingFile*

for ... in files(*hide_default: bool = False*, *path: Optional[str] = None*, *file_sizes: bool = False*, *deleted_files: bool = False*) → Generator[Tuple[*dathost.models.file.FileModel*, *dathost.server.blocking.file.BlockingFile*], None, None]
Used to list files.

Parameters

- **hide_default** (*bool, optional*) – by default False
- **path** (*str, optional*) – Path to use as root, by default None
- **file_sizes** (*bool, optional*) – by default False
- **deleted_files** (*bool, optional*) – Include deleted files in list, by default False

Yields

- *FileModel* – Holds details on a file.
- *BlockingFile* – Used to interact with a file.

ftp_reset() → None
Resets the FRP password.

get() → *dathost.models.server.ServerModel*
Used to get details on server.

Returns Holds data on server.

Return type *ServerModel*

metrics() → *dathost.models.metrics.MetricsModel*
Used to get server metrics.

Returns Holds details on server metrics.

Return type *MetricsModel*

reset() → None
Used to reset the server.

start(*allow_host_reassignment: bool = True*) → None
Used to stop the server.

Parameters **allow_host_reassignment** (*bool, optional*) – By default True

stop() → None
Used to stop the server.

sync() → None
Used to sync files from server to cache.

update(*settings:* dathost.settings.ServerSettings) → None
Update servers paramters.

Parameters **settings** (ServerSettings) – Used to configure server.

Match

class dathost.match.blocking.**BlockingMatch**

get() → *dathost.models.match.MatchModel*
Gets details on a match

Returns Holds match details.

Return type *MatchModel*

Backup

class dathost.server.blocking.backup.**BlockingBackup**

restore() → None
Used to restore a backup.

File

class dathost.server.blocking.file.**BlockingFile**

delete() → None
Deletes file.

dowload() → bytes
Used to download a file into memory.

Returns

Return type bytes

Notes

Its reccomened to use download_iterate for large files.

download_iterate() → None

Raises ***AwaitingOnly*** – This function is meant only for awaiting code.

move(*destination: str*) → None
Used for moving a file.

Parameters **destination** (*str*) –

Notes

When called the file_path changes to the given destination.

save(*local_pathway: str*) → None
Saves file to local pathway.

Parameters **local_pathway** (*str*) – Pathway to save file to.

unzip(*destination: str*) → None
Used to unzip a file.

Parameters **destination** (*str*) –

upload(*data: bytes*) → None
Used for uploading raw bytes.

Parameters **data** (*bytes*) – Data to upload.

upload_file(*local_pathway: str*) → None
Used to upload a local file.

Parameters **local_pathway** (*str*) – Local file to upload.

2.4 Settings

2.4.1 ServerSettings

class dathost.settings.**ServerSettings**(*name: Optional[str] = None*, *location: Optional[str] = None*, *custom_domain: Optional[str] = None*, *autostop: Optional[bool] = None*, *autostop_minutes: Optional[int] = None*, *mysql: Optional[bool] = None*, *scheduled_commands: Optional[List[str]] = None*, *user_data: Optional[str] = None*, *reboot_on_crash: Optional[bool] = None*, *max_disk_usage_gb: Optional[int] = None*, *manual_sort_order: Optional[int] = None*, *core_dump: Optional[bool] = None*, *prefer_dedicated: Optional[bool] = None*)

csgo(*slots: Optional[int] = None*, *tickrate: Optional[int] = None*, *game_token: Optional[str] = None*, *rcon_password: Optional[str] = None*, *game_mode: Optional[str] = None*, *autoload_configs: Optional[List[str]] = None*, *disable_bots: bool = False*, *workshop_start_map_id: Optional[int] = None*, *csay_plugin: bool = False*, *gotv: bool = False*, *sourcemod: bool = False*, *insecure: bool = False*, *map_group: Optional[str] = None*, *start_map: Optional[str] = None*, *password: Optional[str] = None*, *pure: bool = True*, *admins: Optional[List[Any]] = None*, *plugins: Optional[List[Any]] = None*, *steam_key: Optional[str] = None*, *workshop_id: Optional[int] = None*, *maps_source: Optional[str] = None*) → *dathost.settings.ServerSettings*
Used for configuring a CS: GO server.

Parameters

- **slots** (*int*) –
- **game_token** (*str*) –
- **tickrate** (*int*) –
- **game_mode** (*str, optional*) – by default None
- **autoload_configs** (*List[str], optional*) – by default None

- **disable_bots** (*bool, optional*) – by default False
- **csay_plugin** (*bool, optional*) – by default False
- **gotv** (*bool, optional*) – by default False
- **sourcemod** (*bool, optional*) – by default False
- **insecure** (*bool, optional*) – by default False
- **map_group** (*str, optional*) – by default None
- **start_map** (*str, optional*) – by default None
- **password** (*str, optional*) – by default None
- **pure** (*bool, optional*) – by default True
- **rcon_password** (*str, optional*) – by default None
- **admins** (*List[Any], optional*) – by default None
- **plugins** (*List[Any], optional*) – by default None
- **steam_key** (*str, optional*) – by default None
- **workshop_id** (*int, optional*) – by default None
- **workshop_start_map_id** (*int, optional*) – by default None
- **maps_source** (*int, optional*) – by default None

Raises

- ***MultipleGames*** – Raised when you attempt to create one server with multiple games.
- ***InvalidSlotSize*** – Raised when slot size is below 5 or above 64.
- ***InvalidTickrate*** – Raised when tickrate is invalid.

Returns

Return type *ServerSettings*

teamspeak(*slots: int*) → *dathost.settings.ServerSettings*

Used for configuring a teamspeak server.

Parameters **slots** (*int*) –

Raises

- ***MultipleGames*** – Raised when you attempt to create one server with multiple games.
- ***InvalidSlotSize*** – Raised when slot size is below 5 or above 500.

Returns

Return type *ServerSettings*

tf2(*slots: Optional[int] = None, rcon_password: Optional[str] = None, gotv: bool = False, sourcemod: bool = False, insecure: bool = False, password: Optional[str] = None, admins: Optional[list] = None*) → *dathost.settings.ServerSettings*

Used for configuring a TF2 server.

Parameters

- **rcon_password** (*str*) –
- **slots** (*int*) –

- **gotv** (*bool, optional*) – by default False
- **sourcemod** (*bool, optional*) – by default False
- **insecure** (*bool, optional*) – by default False
- **password** (*str, optional*) – by default None
- **admins** (*list, optional*) – by default None

Raises

- ***MultipleGames*** – Raised when you attempt to create one server with multiple games.
- ***InvalidSlotSize*** – Raised when slot size is below 5 or above 32.

Returns

Return type *ServerSettings*

valheim(*password: Optional[str] = None*, *world_name: Optional[str] = None*, *plus: Optional[bool] = None*, *admins: Optional[List[Any]] = None*) → *dathost.settings.ServerSettings*

Used to configure valheim server.

Parameters

- **password** (*str, optional*) – by default None
- **world_name** (*str, optional*) – by default None
- **plus** (*bool, optional*) – by default None
- **admins** (*List[Any], optional*) – List of SteamIDs in any format, by default None

Returns

Return type *ServerSettings*

Raises ***MultipleGames*** –

2.4.2 MatchSettings

class dathost.settings.**MatchSettings**(*connection_time: int = 300*, *knife_round: bool = False*, *wait_for_spectators: bool = True*, *enable_pause: bool = False*, *enable_ready: bool = False*, *enable_tech_pause: bool = False*, *ready_min_players: int = 1*, *wait_for_coaches: bool = True*, *warmup_time: int = 15*)

playwin(*webhook: Optional[str] = None*) → *dathost.settings.MatchSettings*

Enables playwin AC.

Parameters **webhook** (*str, optional*) – Webhook to push playwin results, by default None

Returns

Return type *MatchSettings*

spectators(*players: list*) → *dathost.settings.MatchSettings*

Spectators

Parameters **players** (*list*) – List of spectator steam IDs, steamID 64, 32 & u are supported.

Returns

Return type *MatchSettings*

team_1(*players: list*, *coach: Optional[Union[str, int]] = None*) → *dathost.settings.MatchSettings*
Team 1 players

Parameters

- **players** (*list*) – List of spectator steam IDs, steamID 64, 32 & u are supported.
- **coach** (*Union[str, int]*) – Steam id of coach, by deafult None

Returns

Return type *MatchSettings*

team_2(*players: list*, *coach: Optional[Union[str, int]] = None*) → *dathost.settings.MatchSettings*
Team 2 players

Parameters

- **players** (*list*) – List of spectator steam IDs, steamID 64, 32 & u are supported.
- **coach** (*Union[str, int]*) – Steam id of coach, by deafult None

Returns

Return type *MatchSettings*

webhook(*match_end: str*, *round_end: str*, *authorization: Optional[str] = None*) → *dathost.settings.MatchSettings*
Used to set webhooks.

Parameters

- **match_end** (*str*) – URL of match end webhook.
- **round_end** (*str*) – URL of round end webhook.
- **authorization** (*str, optional*) – by default None

Returns

Return type *MatchSettings*

2.5 Models

2.5.1 Account

class dathost.models.account.**AccountModel**
Holds infomation around a account.

account_id

Type str

email

Type str

gravatar_url

Type str

credits

Type str

seconds_left

Type int

time_left

Type int

roles

Type str

trial

Type bool

accepted_terms_of_service_version

Type int

subscription_pay_with_credits

Type bool

affiliate

Type bool

first_month_discount_percentage

Type float

confirmed_at

Type datetime

2.5.2 Backup

class dathost.models.backup.**BackupModel**
Holds detail on backups.

backup_name

Type str

timestamp

Type datetime.datetime

2.5.3 File

class dathost.models.file.**FileModel**
Used to hold details on file.

path

Type str

size

Type str, optional

2.5.4 Match

class dathost.models.match.**MatchModel**

Holds match details.

match_id

Type str

server_id

Type str

connect_time

Type int

round_end_webhook

Type str

match_end_webhook

Type str

finished

Type bool

cancel_reason

Type str

rounds_played

Type int

spectators

Type list

team_1

Type *TeamModel*

team_2

Type *TeamModel*

knife_round

Type bool

playwin

Type bool

playwin_webhook

Type str

playwin_result

Type dict

warmup_time

Type int

wait_for_spectators

Type bool

enable_pause

Type bool

enable_ready

Type bool

enable_tech_pause

Type bool

team_1_coach

Type str

team_2_coach

Type str

wait_for_coaches

Type bool

for ... in players() → Generator[*dathost.models.match.MatchPlayerModel*, None, None]
Used to list players.

Yields *PlayerModel* – Holds details on player.

class dathost.models.match.**MatchPlayerModel**
Holds match player details.

steamid

Type str

kills

Type int

deaths

Type int

assists

Type int

kdr

Type float

class dathost.models.match.**TeamModel**
Holds details on team.

score

Type int

players

Type list

2.5.5 Metric

class dathost.models.metrics.**MetricsModel**

for ... in all_time_players() → Generator[*dathost.models.metrics.PlayerModel*, None, None]
Used to list all time players.

Yields *PlayerModel* – Holds details on online players.

for ... in maps() → Generator[*dathost.models.metrics.MapsModel*, None, None]
Used to list all maps what have been played.

Yields *MapsModel* – Holds details on maps.

for ... in players_online() → Generator[*dathost.models.metrics.PlayerModel*, None, None]
Used to list all players online.

Yields *PlayerModel* – Holds details on online players.

for ... in players_online_graph() → Generator[*dathost.models.metrics.PlayersOnlineGraphModel*, None, None]
Used to list all players online graph.

Yields *PlayersOnlineGraphModel* – Holds details on online player times.

class dathost.models.metrics.**MapsModel**
Holds map details

map

Type str

seconds

Type int

class dathost.models.metrics.**PlayerModel**
Holds player details

name

Type str

duration

Type int

score

Type int

class dathost.models.metrics.**PlayersOnlineGraphModel**
Holds player graph details

timestamp

Type str

value

Type str

2.5.6 Server

class dathost.models.server.**ServerModel**

Holds details on server

server_id

Type str

name

Type str

user_data

Type str

match_id

Type str

game

Type str

location

Type str

players_online

Type int

status

Type list

booting

Type bool

server_error

Type str

ip

Type str

raw_ip

Type str

on

Type bool

ports

Type *PortsModel*

confirmed

Type bool

cost_per_hour

Type int

max_cost_per_hour

Type int

month_credits

Type float

month_reset_at

Type datetime

max_cost_per_month

Type float

subscription_cycle_months

Type int

subscription_renewal_failed_attempts

Type int

mysql

Type bool

autostop

Type bool

autostop_minutes

Type int

mysql_username

Type str

mysql_password

Type str

ftp_password

Type str

disk_usage_bytes

Type int

default_file_locations

Type list

custom_domain

Type str

added_voice_server

Type str

duplicate_source_server

Type str

teamspeak

Type *TeamspeakModel*

teamfortress

Type *TeamFortressModel*

valheim

Type *ValheimModel*

csgo

Type CsgoModel

max_disk_usage_gb

Type int

reboot_on_crash

Type bool

core_dump

Type bool

prefer_dedicated

Type bool

for ... in scheduled_commands() → Generator[*dathost.models.server.ScheduledCommandsModel*, None, None]

Lists scheduled commands.

Yields *ScheduledCommandsModel* – Holds data on scheduled commands.

class dathost.models.server.**PortsModel**

Holds details on ports

game

Type int

gotv

Type int

class dathost.models.server.**ScheduledCommandsModel**

Holds details on scheduled commands

name

Type str

action

Type str

command

Type str

run_at

Type str

repeat

Type bool

class dathost.models.server.**TeamspeakModel**

Holds details on teamspeak server

slots

Type int

admin_token

Type str

class dathost.models.server.**TeamFortressModel**
Holds details on tf2.

slots

Type int

rcon

Type str

password

Type str

admins

Type str

gotv

Type bool

sourcemod

Type bool

insecure

Type bool

class dathost.models.server.**ValheimModel**
Holds details on valheim server

admins

Type List[Int]

plus

Type bool

slots

Type int

password

Type str

world_name

Type str

2.6 Exceptions

class dathost.exceptions.**DathostException**
Base exception for dathost.

class dathost.exceptions.**InvalidSlotSize**
Raised when slot size is invalid.

class dathost.exceptions.**MultipleGames**
Raised when you attempt to create one server with multiple games.

class dathost.exceptions.**InvalidTickrate**
Raised when tickrate is invalid.

class dathost.exceptions.**InvalidConsoleLine**
Raised when console line is above 1 or above 100000.

class dathost.exceptions.**AwaitingOnly**
Raised when a coroutine called is awaiting supported only.

class dathost.exceptions.**InvalidSteamID**
Raised when give ID isn't understood.

class dathost.exceptions.**NotFound**
Resource not found.

class dathost.exceptions.**BadRequest**
Path is a directory or Cannot move file into itself.

class dathost.exceptions.**ExceededStorage**
Your disk quota of 30GB per server (excluding base installation) has been exceeded

class dathost.exceptions.**ServerStart**
Failed to start server.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

D

E

F

G

I

K

L

M

N

O

P

R

S

T

U

V

W