kudubot

Release 0.19.0

Jun 05, 2018

CONTENTS

1 kudubot package **3**
- 1.1 Subpackages . . . 3
- 1.2 Submodules . . . 39
- 1.3 kudubot.exceptions module . . . 39
- 1.4 kudubot.main module . . . 39
- 1.5 Module contents . . . 40

2 kudubot **41**

3 Indices and tables **43**

Python Module Index **45**

Index **47**

Contents:

CHAPTER

ONE

KUDUBOT PACKAGE

1.1 Subpackages

1.1.1 kudubot.config package

Submodules

kudubot.config.GlobalConfigHandler module

class kudubot.config.GlobalConfigHandler.**GlobalConfigHandler**(*config_location: str = '/home/gitlab-runner/.kudubot'*)

Bases: `object`

Class that handles the global kudubot configuration files located in $HOME/.kudubot

__init__(*config_location: str = '/home/gitlab-runner/.kudubot'*)
Initializes the ConfigHandler. Determines the config locations using the config_location parameter which defaults to a .kudubot directory in the user's home directory. The configuration may still be invalid once the object is initialized, call validate_config_directory() to make sure that the configuration is correct.

Parameters **config_location** – The location of the config directory

generate_configuration(*delete_old=False*)
Generates a new, empty config location.

Parameters **delete_old** – If set, all old config files that may exist are overwritten

Returns None

load_connections() → List[type]
Loads all connections from the connections configuration file

Returns A list of successfully imported Connection subclasses

load_services() → List[type]
Loads all Services from the services configuration file

Returns A list of successfully imported Service subclasses

logger = <logging.Logger object>
The Logger for this class

validate_config_directory() → bool
Validates the configuration directory. As soon as a discrepancy is detected, the reason is logged and False is returned. If the configuration is valid however, True is returned :return: True if the config is valid, False otherwise

kudubot.config.StandardConfigWriter module

class kudubot.config.StandardConfigWriter.**StandardConfigWriter**(*config_handler: kudubot.config.GlobalConfigHandler.Glo = <kudubot.config.GlobalConfigHandler.G object>*)

Bases: `object`

A config handler class that writes the standard configuration for connections and services into the config.

__init__(*config_handler: kudubot.config.GlobalConfigHandler.GlobalConfigHandler = <kudubot.config.GlobalConfigHandler.GlobalConfigHandler object>*)
Initializes the Standard Config Writer's configuration file locations

Parameters **config_handler** – The GlobalConfigHandler to use. Default to the default one.

write_standard_connection_config()
Writes the standard connection configuration file

Returns None

write_standard_service_config()
Writes the standard service configuration file

Returns None

Module contents

1.1.2 kudubot.connections package

Subpackages

kudubot.connections.cli package

Submodules

kudubot.connections.cli.CliConnection module

class kudubot.connections.cli.CliConnection.**CliConnection**(*services: List[type], config_handler: kudubot.config.GlobalConfigHandler.GlobalCon*

Bases: *kudubot.connections.Connection.Connection*

The CLI kudubot connection class.

static define_identifier() → str
Defines the identifier for this Connection

Returns None

define_user_contact() → kudubot.users.Contact.Contact
Specifies the User Contact, which is of no importance for the CLI Connection

Returns The CLI's user Contact

generate_configuration()
The CLI Connection does not require any configuration

Returns None

listen()
Starts an endless loop that continuously prompts the user for input

Returns None

load_config() → Dict[str, object]
The CLI Connection does not require any configuration

Returns {}

send_audio_message(*receiver: kudubot.users.Contact.Contact*, *audio_file: str*, *caption: str = ''*)
Sending audio files is not supported in the CLI :param receiver: The recipient of the file :param audio_file: The audio file to send :param caption: the caption to send :return: None

send_image_message(*receiver: kudubot.users.Contact.Contact*, *image_file: str*, *caption: str = ''*)
Sending image files is not supported in the CLI

Parameters

- **receiver** – The recipient
- **image_file** – The image file to send
- **caption** – The caption to use

Returns None

send_message(*message: kudubot.entities.Message.Message*)
Prints the message to the console

Parameters **message** – The message to 'send'

Returns None

send_video_message(*receiver: kudubot.users.Contact.Contact*, *video_file: str*, *caption: str = ''*)
Sending videos is not supported in the CLI

Parameters

- **receiver** – The message's recipient
- **video_file** – The video file to send
- **caption** – The caption to send

Returns None

Module contents

kudubot.connections.telegram package

Submodules

kudubot.connections.telegram.TelegramConnection module

class kudubot.connections.telegram.TelegramConnection.**TelegramConnection**(*services: List[type], config_handler: kudubot.config.GlobalCon*

Bases: *kudubot.connections.Connection.Connection*

A Connection class that connects to Telegram

__init__(*services: List[type], config_handler: kudubot.config.GlobalConfigHandler.GlobalConfigHandler*)
Initializes the Telegram Connection. Additionally to the standard Connection, a telegram.Bot object is generated to interface with the Telegram API

Parameters

- **services** – The services to include
- **config_handler** – The GlobalConfigHandler that determines the location of the configuration files

static define_identifier() → str
Defines the Connection's unique identifier

Returns The Connection's identifier

define_user_contact() → kudubot.users.Contact.Contact

Returns The Telegram connection's contact information

generate_configuration()
Generates a new Configuration file for the Telegram Connection

Returns None

listen()
Listens to new Telegram messages Starts an infinite loop that polls for new messages every second, and applies the Connection's Services to these received messages.

Returns None

load_config() → Dict[str, object]
Loads the configuration file for the Telegram Connection A Telegram Connection requires only an api_key attribute in the [credentials] section of the config file

Any configuration errors are raised as new InvalidConfigExceptions

Returns The configuration, as a dictionary

send_audio_message(*receiver: kudubot.users.Contact.Contact*, *audio_file: str*, *caption: str = ''*)
Sends an Audio Message via Telegram

Parameters

- **receiver** – The recipient of the message

- **audio_file** – The audio file to send
- **caption** – The caption to display together with the audio

Returns

send_image_message(*receiver: kudubot.users.Contact.Contact*, *image_file: str*, *caption: str = ''*)
Sends an Image Message via Telegram

Parameters

- **receiver** – The recipient of the message
- **image_file** – The image file to send
- **caption** – The caption to be displayed with the image

Returns

send_message(*message: kudubot.entities.Message.Message*)
Sends a message using the Telegram connection

Parameters **message** – The message to send

Returns None

send_video_message(*receiver: kudubot.users.Contact.Contact*, *video_file: str*, *caption: str = ''*)
Sends a video message via Telegram

Parameters

- **receiver** – The recipient of the message
- **video_file** – The video file to send
- **caption** – The caption to be displayed together with the video

Returns

Module contents

kudubot.connections.whatsapp package

Submodules

kudubot.connections.whatsapp.EchoLayer module

class kudubot.connections.whatsapp.EchoLayer.**EchoLayer**(*connection*)
Bases: yowsup.layers.interface.interface.YowInterfaceLayer

The Yowsup Echo Layer used to communicate with the Whatsapp Servers

__init__(*connection*)
Custom constructor used to pass the WhatsappConnection class to the layer

Parameters **connection** – The Whatsapp connection object that calls this layer

alias_to_jid(*c_alias: str*) → str
Turns an alias into a jid

Parameters **c_alias** – the current alias

Returns the jid

static create_audio_media(*file_path: str*, *url: str*, *to: str*, *ip: str = None*) → yowsup.layers.protocol_media.protocolentities.message_media_downloadable_audio.AudioDo

Creates an Audio Message Entity as a workaround for the missing fromFilePath method in AudioDownloadableMediaMessageProtocolEntity's parent class

Parameters

- **file_path** – the path to the file
- **url** – the whatsapp upload url
- **to** – the receiver
- **ip** – the ip of the receiver

Returns None

Returns the generatd audio entity

do_send_media(*media_type: ('audio', 'image', 'video')*, *file_path: str*, *url: str*, *to: str*, *ip: str = None*, *caption: str = None*)

Sends a media file

Parameters

- **media_type** – The type of media to send
- **file_path** – the path to the file
- **url** – the whatsapp upload url
- **to** – the receiver
- **ip** – the ip of the receiver
- **caption** – the caption to be displayed together with the media file

Returns None

static normalize_jid(*number: str*) → str

Normalizes a jid

Parameters **number** – the number to be receive a normalized jid

Returns the normalized jid

on_message(*message_protocol_entity: yowsup.layers.protocol_messages.protocolentities.message_text.TextMessageProtocol*

Method run when a message is received :param message_protocol_entity: the message received :return: void

on_receipt(*entity: yowsup.layers.protocol_messages.protocolentities.message.MessageProtocolEntity*)

Method run whenever a whatsapp receipt is issued This ensures that a message is marked as 'read'

Parameters **entity** – The receipt entity

Returns None

static on_request_upload_error(*jid: str*, *path: str*, *error_request_upload_iq_protocol_entity: yowsup.layers.protocol_messages.protocolentities.message.MessageProtocolEntity*, *request_upload_iq_protocol_entity: yowsup.layers.protocol_media.protocolentities.iq_requestupload.RequestUploadIqPro*

Method run when a media upload result is negative

Parameters

- **jid** – the jid to receive the media

- **path** – the file path to the media
- **error_request_upload_iq_protocol_entity** – the error result entity
- **request_upload_iq_protocol_entity** – the request entity

Returns None

on_request_upload_result(*jid: str*, *media_type: ('audio', 'image', 'video')*, *file_path: str*, *result_request_upload_iq_protocol_entity: yowsup.layers.protocol_media.protocolentities.iq_requestupload_result.ResultRequestUploadIq request_upload_iq_protocol_entity: yowsup.layers.protocol_media.protocolentities.iq_requestupload.RequestUploadIqProtocolEnti caption: str = None*)

Method run when a media upload result is positive

Parameters

- **jid** – the jid to receive the media
- **media_type** – The media type to send
- **file_path** – the path to the media file
- **result_request_upload_iq_protocol_entity** – the result entity
- **request_upload_iq_protocol_entity** – the request entity
- **caption** – the media caption, if applicable

Returns None

static on_upload_error(*file_path: str*, *jid: str*, *url: str*)

Method run when an upload error occurs

Parameters

- **file_path** – the file path of the file to upload
- **jid** – the jid of the receiver of the file
- **url** – the upload url

Returns None

static on_upload_progress(*file_path: str*, *jid: str*, *url: str*, *progress: float*)

Method that keeps track of the upload process

Parameters

- **file_path** – the file path of the media file
- **jid** – the jid of the receiver
- **url** – the whatsapp upload url
- **progress** – the current progress

:return:None

send_audio_message(*audio_file: str*, *receiver: str*, *caption: str*)

Sends an audio file

Parameters

- **audio_file** – The audio file to send
- **receiver** – The recipient of the file

- **caption** – A caption to be displayed with the audio file

Returns None

send_image_message(*image_file: str*, *receiver: str*, *caption: str*)
Sends an image file

Parameters

- **image_file** – The image file to send
- **receiver** – The recipient of the file
- **caption** – A caption to be displayed with the image

Returns None

send_media(*path: str*, *number: str*, *caption: str*, *media_type: ('audio', 'image', 'video')*)
Sends a media file

Parameters

- **path** – the path to the media file
- **number** – the receiver of the media file
- **caption** – the caption to be shown
- **media_type** – The type of media to send

Returns None

send_receipt(*message_protocol_entity: yowsup.layers.protocol_messages.protocolentities.message.MessageProtocolEntity*
Sends the whatsapp servers that the message was received

Parameters **message_protocol_entity** – the message protocol entity that was received

Returns None

send_text_message(*body: str*, *receiver: str*)
Sends a text message

Parameters

- **body** – The message body
- **receiver** – The recipient address

Returns None

send_video_message(*video_file: str*, *receiver: str*, *caption: str*)
Sends a video file

Parameters

- **video_file** – The video file to send
- **receiver** – The recipient of the file
- **caption** – A caption to be displayed with the video file

Returns None

kudubot.connections.whatsapp.WhatsappConnection module

class kudubot.connections.whatsapp.WhatsappConnection.**WhatsappConnection**(*services: List[type], config_handler: kudubot.config.GlobalCon*

Bases: *kudubot.connections.Connection.Connection*

Class that implements a kudubot connection for the Whatsapp Messaging Service

__init__(*services: List[type], config_handler: kudubot.config.GlobalConfigHandler.GlobalConfigHandler*)
Extends the default Connection constructor to create a yowsup stack

Parameters

- **services** – The services to start
- **config_handler** – The GlobalConfigHandler that determines the location of the configuration files

static define_identifier() → str
Defines a unique identifier for the connection

Returns 'whatsapp'

define_user_contact() → kudubot.users.Contact.Contact

Returns The Whatsapp connection's contact information

generate_configuration()
Generates a new Configuration file for the Whatsapp Connection

Returns None

listen()
Starts the Yowsup listener

Returns None

load_config() → Dict[str, str]
Loads the configuration for the Whatsapp Connection from the config file

Returns The parsed configuration, consisting of a dictionary with a number and pass key

send_audio_message(*receiver: kudubot.users.Contact.Contact*, *audio_file: str*, *caption: str = ''*)
Sends an audio message using the Whatsapp Connection

Parameters

- **receiver** – The receiver of the audio message
- **audio_file** – The audio file to send
- **caption** – An optional caption for the audio message

Returns None

send_image_message(*receiver: kudubot.users.Contact.Contact*, *image_file: str*, *caption: str = ''*)
Sends an image message using the Whatsapp Connection

Parameters

- **receiver** – The receiver of the image message
- **image_file** – The image file to send

- **caption** – An optional caption for the image message

Returns None

send_message(*message: kudubot.entities.Message.Message*)
Sends a Text message using the Whatsapp Connection

Parameters **message** – The message to send

Returns None

send_video_message(*receiver: kudubot.users.Contact.Contact*, *video_file: str*, *caption: str = ''*)
Sends a video message using the Whatsapp Connection

Parameters

- **receiver** – The receiver of the video message
- **video_file** – The video file to send
- **caption** – An optional caption for the video message

Returns None

Module contents

Submodules

kudubot.connections.Connection module

class kudubot.connections.Connection.**Connection**(*services: List[type], config_handler: kudubot.config.GlobalConfigHandler.GlobalConfigHandler*)

Bases: object

Abstract class that provides a unified model for a chat service connection It keeps track of various state variables and provides APIs to send messages and start a listener that reacts on incoming messages using the implemented service modules

__init__(*services: List[type], config_handler: kudubot.config.GlobalConfigHandler.GlobalConfigHandler*)
Initializes the connection object using the specified services Starts the database connection

Parameters

- **services** – The services to use with the connection
- **config_handler** – The GlobalConfigHandler to determine config file locations

apply_services(*message: kudubot.entities.Message.Message*, *break_on_match: bool = True*)
Applies the services to a Message First, the method checks if a service is applicable to a message. Then, if it is applicable, the service will process the message If the break_on_match parameter is set to True, the first match will always end the loop.

Parameters

- **message** – The message to process
- **break_on_match** – Can be set to False to allow more than one result

static define_identifier() → str
Defines the connection's identifier

Returns The identifier for the Connection type

define_user_contact() → kudubot.users.Contact.Contact
Creates a Contact object for the connection by which the connection itself is identified

Returns The connection's user object

generate_configuration()
Generates a new configuration file for this connection.

get_database_connection_copy() → sqlite3.Connection
Creates a new sqlite Connection to the kudubot Connection's database file

Returns The generated sqlite Connection

listen()
Starts listening on the connection in an infinite loop. If the execution of the program has to continue past starting the listener, the listen_in_separate_thread() method should be called instead.

listen_in_separate_thread() → threading.Thread
Runs the listen() method in a separate daemon thread

Returns The listening thread

load_config() → Dict[str, object]
Loads the configuration for the connection. If this fails forsome reason, an InvalidConfigException is raised

Returns A dictionary containing the configuration

send_audio_message(*receiver: kudubot.users.Contact.Contact*, *audio_file: str*, *caption: str = ''*)
Sends an audio message using the connection

Parameters

- **receiver** – The receiver of the message
- **audio_file** – The path to the audio file to send
- **caption** – The caption sent together with the message

send_image_message(*receiver: kudubot.users.Contact.Contact*, *image_file: str*, *caption: str = ''*)
Sends an image message using the connection

Parameters

- **receiver** – The recipient of the image message
- **image_file** – The path to the image file
- **caption** – The caption to be displayed with the image

send_message(*message: kudubot.entities.Message.Message*)
Sends a Message using the connection

Parameters **message** – The message to send

send_video_message(*receiver: kudubot.users.Contact.Contact*, *video_file: str*, *caption: str = ''*)
Sends a video message using the connection

Parameters

- **receiver** – The recipient of the video message
- **video_file** – The path to the video file to be sent
- **caption** – The caption to be displayed with the video

Module contents

1.1.3 kudubot.entities package

Submodules

kudubot.entities.Message module

class kudubot.entities.Message.**Message**(*message_title: str*, *message_body: str*, *receiver: kudubot.users.Contact.Contact*, *sender: kudubot.users.Contact.Contact*, *sender_group: kudubot.users.Contact.Contact = None*, *timestamp: float = -1.0*)

Bases: object

Class that models a message entity.

__init__(*message_title: str*, *message_body: str*, *receiver: kudubot.users.Contact.Contact*, *sender: kudubot.users.Contact.Contact*, *sender_group: kudubot.users.Contact.Contact = None*, *timestamp: float = -1.0*)

Initializes the Message object

Since Messages may also come from chat groups, an optional group argument exists. A group is another contact, just like any other contact. Group messages must define a contact for the group as well as for the individual sender of the message.

A Message object whose group is None is a private message.

Parameters

- **message_title** – The title of the message. Not always applicable, depending on the connection
- **message_body** – The body of the message
- **receiver** – The recipient of the message, which is a Contact object
- **sender** – The sender of the message, which is a Contact object
- **sender_group** – Optionally the group Contact object if this is a message originating from a group
- **timestamp** – The timestamp of the message. If it is not specified, the current UNIX timestamp is used

get_direct_response_contact() → kudubot.users.Contact.Contact

Returns The contact to which one can immediately respond to. I.e. if the message came from a group, this method will return the group contact, private chats however will return the individual sender

to_dict() → Dict[str, str]

Returns The message data as a dictionary which can be used to store a message as a JSON file

kudubot.entities.Message.**from_dict**(*data: Dict[str, str]*) → kudubot.entities.Message.Message

Generates a Message object from a dictionary

Parameters **data** – The dictionary to turn into a Message

Returns The generates Message object

Module contents

1.1.4 kudubot.services package

Subpackages

kudubot.services.internal package

Subpackages

kudubot.services.internal.anime_reminder package

Submodules

kudubot.services.internal.anime_reminder.AnimeReminderService module

class kudubot.services.internal.anime_reminder.AnimeReminderService.**AnimeReminderService**(*co*
Bases: *kudubot.services.HelperService.HelperService*

The Kudubot Service that provides the anime reminder functionality

background_loop()
Starts a new thread which will continuously check for new anime discussion threads on reddit.com/r/anime and notify users if they are subscribed to those shows

Returns None

define_command_name(*language: str*) → str
Defines the command name for the anime-remind functionality

Parameters **language** – The language for which the command is valid

Returns The command name in the specified language

define_help_message(*language: str*) → str
Defines the help message for the service

Parameters **language** – The target language to translate the message to

Returns The help message in the specified language

static define_identifier() → str

Returns The service's identifier

define_language_text() → Dict[str, Dict[str, str]]
Defines the Language strings for the service for the implemented languages

Returns The dictionary used to translate strings

define_syntax_description(*language: str*) → str
Defines the syntax description for the Service

Parameters **language** – The language in which to display the syntax description

Returns The syntax description in the specified language

determine_language(*message: kudubot.entities.Message.Message*) → str
Determines the language used based on the incoming message

Parameters **message** – The message to analyze

Returns The language key to use

handle_message(*message: kudubot.entities.Message.Message*)
Handles a message from the user

Parameters **message** – The message to handle

Returns None

init()
In addition to the normal initialization of a Service, this service initializes its database and starts the background thread

is_applicable_to(*message: kudubot.entities.Message.Message*) → bool
Checks if a message is applicable to this Service

Parameters **message** – The message to check

Returns None

kudubot.services.internal.anime_reminder.database module

kudubot.services.internal.anime_reminder.database.**delete_subscription**(*user: int*, *show_name: str*, *db: sqlite3.Connection*) → bool
Deletes a subscription from the database

Parameters

- **user** – The user ID for this subscription
- **show_name** – The show name of the subscription
- **db** – The database to use

:return True if the subscription existed and is deleted, False if the subscription did not exist beforehand

kudubot.services.internal.anime_reminder.database.**get_subscriptions**(*db: sqlite3.Connection*, *user: int = -1*) → List[Dict[str, str]]
Fetches a list of subscriptions from the database, either for all users or one specific user.

Parameters

- **db** – The database to use
- **user** – The user parameter can limit the query to search for subscriptions for only one user

Returns A list of dictionaries representing the subscriptions

kudubot.services.internal.anime_reminder.database.**initialize_database**(*db: sqlite3.Connection*)
Initializes the Service's database tables

Returns None

kudubot.services.internal.anime_reminder.database.**logger = <logging.Logger object>**
The logger for this module

kudubot.services.internal.anime_reminder.database.**store_subscription**(*user: int*, *show_name: str*, *db: sqlite3.Connection*) → bool

Creates a new subscription if it does not exist yet

Parameters

- **user** – The user ID for this subscription
- **show_name** – The show name of the subscription
- **db** – The database to use

Returns True if a new subscription was stored, False if the subscription already existed

kudubot.services.internal.anime_reminder.database.**store_thread**(*thread: Dict[str, str]*, *db: sqlite3.Connection*)

Stores a reddit discussion thread in the database

Parameters

- **thread** – The thread to store
- **db** – The database to use

Returns None

kudubot.services.internal.anime_reminder.database.**subscription_exists**(*user: int*, *show_name: str*, *db: sqlite3.Connection*) → bool

Checks if a subscription already exists

Parameters

- **user** – The user ID for this subscription
- **show_name** – The show name of the subscription
- **db** – The database to use

Returns True if the subscription already exists, otherwise False

kudubot.services.internal.anime_reminder.database.**thread_exists**(*thread: Dict[str, str]*, *db: sqlite3.Connection*) → bool

Checks if a reddit discussion thread with the given thread parameters already exists

Parameters

- **thread** – The thread to check for

- **db** – The database to check

Returns True if the thread exists, false otherwise

kudubot.services.internal.anime_reminder.scraper module

kudubot.services.internal.anime_reminder.scraper.**logger = <logging.Logger object>**
The logger for this module

kudubot.services.internal.anime_reminder.scraper.**scrape_reddit_discussion_threads**() → List[Dict[str, str]]

Scrapes /u/Holo_of_Yoitsu's post history for anime discussion threads :return: The parsed discussion threads as a list of dictionaries with

descriptive keys

Module contents

kudubot.services.internal.authentication_manager package

Submodules

kudubot.services.internal.authentication_manager.AuthenticationManagerService module

class kudubot.services.internal.authentication_manager.AuthenticationManagerService.**Authent**
Bases: *kudubot.services.HelperService.HelperService*, *kudubot.services.AuthenticatedService.AuthenticatedService*

Lets admins manage admin rights of other users and blacklist them.

define_command_name(*language: str*) → str
Defines the command prefix for the service. By default, '/' followed by the service identifier from self.define_identifier() is used.

Parameters **language** – The language in which to define the command name

Returns The command name in the specified language

define_help_message(*language: str*) → str
Defines the help message of the service in various languages

Parameters **language** – The language to use

Returns The help message in the language

static define_identifier() → str
Defines the identifier for this service

Returns The service's identifier

define_language_text() → Dict[str, Dict[str, str]]
Defines the dictionary used for translating strings using the self.reply_translated() or self.translate() methods

The format is:

term: {lang: value}

Returns The dictionary for use in translating

define_syntax_description(*language: str*) → str
Defines the Syntax description of the Service in various languages

Parameters **language** – The language in which to return the syntax message

Returns The syntax message

determine_language(*message: kudubot.entities.Message.Message*) → str
Determines the language of a message

Parameters **message** – The message to check for the language

Returns The language of the message

handle_message(*message: kudubot.entities.Message.Message*)
Handles an applicable message Checks the mode and handles accordingly

Parameters **message** – The message to handle

is_applicable_to(*message: kudubot.entities.Message.Message*) → bool
Checks if a Message is applicable to this Service

Parameters **message** – The message to check

Returns True if the Message is applicable, False otherwise

Module contents

kudubot.services.internal.echo package

Submodules

kudubot.services.internal.echo.EchoService module

class kudubot.services.internal.echo.EchoService.**EchoService**(*connection*)
Bases: *kudubot.services.BaseService.BaseService*

Service that always replies to a user with the same message body that was received

static define_identifier() → str
Defines the identifier of the service :return: “echo”

handle_message(*message: kudubot.entities.Message.Message*)
Replies with the original message text :param message: The original message :return: None

is_applicable_to(*message: kudubot.entities.Message.Message*) → bool
This service is always applicable :param message: The message to check for applicability for :return: True

Module contents

kudubot.services.internal.jokes package

Submodules

kudubot.services.internal.jokes.JokesService module

class kudubot.services.internal.jokes.JokesService.**JokesService**(*connection*)
Bases: *kudubot.services.HelperService.HelperService*

A Service that tells jokes

define_help_message(*language: str*) → str

Parameters **language** – The language to use

Returns The help message for the specified language

static define_identifier() → str

Returns The service's identifier

define_language_text() → Dict[str, Dict[str, str]]

Returns A dictionary used for translations

define_syntax_description(*language: str*) → str

Parameters **language** – The language to use

Returns The syntax description for this language

determine_language(*message: kudubot.entities.Message.Message*) → str
Determines the language for a message :param message: The message to analyze :return: The language that this message was in

handle_message(*message: kudubot.entities.Message.Message*)
Handles an incoming message :param message: :return:

is_applicable_to(*message: kudubot.entities.Message.Message*)
Checks if a message is applicable to this service :param message: The message to analyze :return: True if applicable, False otherwise

load_english_joke() → str
Fetches an English joke from the internet :return: The joke

load_german_joke() → str
Fetches a German joke from the internet :return: The joke

Module contents

kudubot.services.internal.reminder package

Submodules

kudubot.services.internal.reminder.ReminderService module

class kudubot.services.internal.reminder.ReminderService.**ReminderService**(*connection*)
Bases: *kudubot.services.HelperService.HelperService*

Class that implements a Service for the Kudubot framework that allows users to store reminder message that are then sent at a later time

background_loop()
Perpetually checks for expiring reminders.

Returns None

define_command_name(*language: str*)
Defines the command name for this service :param language: the language in which to get the command name :return: The command name in the specified language

define_help_message(*language: str*) → str
Defines the help message for this service in various languages

Parameters **language** – The language to be used

Returns The help description in the specified language

static define_identifier() → str
Defines the identifier for this service

Returns The Service's identifier

define_language_text() → Dict[str, Dict[str, str]]

Returns A dictionary used to translate any user-facing messages

define_syntax_description(*language: str*) → str
Defines the syntax with which the user can interact with this service

Parameters **language** – The language to use

Returns The syntax description in the specified language

determine_language(*message: kudubot.entities.Message.Message*) → str
Determines the language used in a message

Parameters **message** – The message to analyse

Returns The language that was found

handle_message(*message: kudubot.entities.Message.Message*)
Handles a message received by the Service

Parameters **message** – The message to handle

Returns None

init()
Initializes the database table and starts a background thread that perpetually searches for expired reminders

is_applicable_to(*message: kudubot.entities.Message.Message*) → bool
Checks if the Service is applicable to a message

Parameters **message** – The message to check

Returns True if the Service is applicable, otherwise False

parse_message(*text: str*, *language: str*) → Dict[str, str]
Parses the message and determines the mode of operation

Parameters

- **text** – The text to parse
- **language** – The language in which to parse the text

Returns

A dictionary with at least the key 'status' with three different possible states: - no-match: The message does not match the command syntax - help: A query for the help message.

Will result in the help message being sent to the sender

- store: Command to store a new reminder

parse_time_string(*time_string: str*, *language: str*) → <module 'datetime' from '/usr/lib/python3.5/datetime.py'>
Parses a time string like '1 week' or '2 weeks 1 day' etc. and returns a datetime object with the specified time difference to the current time.

Parameters

- **time_string** – The time string to parse
- **language** – In which language the string should be parsed

Returns The parsed datetime object or None in case the parsing failed

kudubot.services.internal.reminder.database module

kudubot.services.internal.reminder.database.**convert_datetime_to_string**(*to_convert: datetime.datetime*) → str
Converts a datetime object to a string that can be stored in the database

Parameters **to_convert** – The datetime object to convert

Returns The datetime as a string

kudubot.services.internal.reminder.database.**convert_string_to_datetime**(*to_convert: str*) → datetime.datetime
Converts a string of the form '%Y-%m-%d:%H-%M-%S' to a datetime object

Parameters **to_convert** – The string to convert

Returns the resulting datetime object

kudubot.services.internal.reminder.database.**get_next_id**(*database: sqlite3.Connection*)

Fetches the next Reminder ID

Parameters **database** – The database to use

Returns The next highest reminder ID

kudubot.services.internal.reminder.database.**get_unsent_reminders**(*database: sqlite3.Connection*) → List[Dict[str, str]]

Retrieves all unsent reminders from the database

Parameters **database** – The database to use

Returns A list of dictionaries that contain the reminder information

kudubot.services.internal.reminder.database.**initialize_database**(*database: sqlite3.Connection*)

Initializes the Database Table for the reminder service

Parameters **database** – The database connection to use

Returns None

kudubot.services.internal.reminder.database.**logger = <logging.Logger object>**

The logger for this module

kudubot.services.internal.reminder.database.**mark_reminder_sent**(*database: sqlite3.Connection*, *reminder_id: int*)

Marks a reminder as sent

Parameters

- **database** – The database connection to use
- **reminder_id** – The Reminder ID

Returns None

kudubot.services.internal.reminder.database.**store_reminder**(*database: sqlite3.Connection*, *message: str*, *due_time: datetime.datetime*, *sender_id: int*)

Stores a reminder in the database

Parameters

- **database** – The database Connection to use
- **message** – The message text to store
- **due_time** – The time at which the message should be sent
- **sender_id** – The initiator's id in the address book table

Returns None

Module contents

kudubot.services.internal.service_lister package

Submodules

kudubot.services.internal.service_lister.ServiceListerService module

class kudubot.services.internal.service_lister.ServiceListerService.**ServiceListerService**(*co*
Bases: *kudubot.services.HelperService.HelperService*

This Service lists all currently active services

define_command_name(*language: str*) → str
Defines the command prefix for the service. By default, '/' followed by the service identifier from self.define_identifier() is used.

Parameters **language** – The language in which to define the command name

Returns The command name in the specified language

define_help_message(*language: str*) → str
Defines the help message of the service in various languages

Parameters **language** – The language to use

Returns The help message in the language

static define_identifier() → str
Defines the identifier for this service

Returns The service's identifier

define_language_text() → Dict[str, Dict[str, str]]
Defines the dictionary used for translating strings using the self.reply_translated() or self.translate() methods

The format is:

term: {lang: value}

Returns The dictionary for use in translating

define_syntax_description(*language: str*) → str
Defines the Syntax description of the Service in various languages

Parameters **language** – The language in which to return the syntax message

Returns The syntax message

determine_language(*message: kudubot.entities.Message.Message*) → str
Determines the language of a message

Parameters **message** – The message to check for the language

Returns The language of the message

handle_message(*message: kudubot.entities.Message.Message*)
Handles an applicable message. Sends a message containing the identifiers of all active services

Parameters **message** – The message to handle

is_applicable_to(*message: kudubot.entities.Message.Message*) → bool
Checks if a Message is applicable to this Service Checks if the

Parameters **message** – The message to check

Returns True if the Message is applicable, False otherwise

Module contents

kudubot.services.internal.simple_responder package

Submodules

kudubot.services.internal.simple_responder.SimpleResponderService module

class kudubot.services.internal.simple_responder.SimpleResponderService.**SimpleResponderServ**
Bases: *kudubot.services.BaseService.BaseService*

Class that implements a kudubot service that analyzes Strings and responds to them using a relatively simple ruleset

static define_identifier() → str
Defines the Service's unique identifier

Returns The unique identifier

handle_message(*message: kudubot.entities.Message.Message*)
Handles the message, provided this service is applicable to it

Parameters **message** – The message to process

Returns None

is_applicable_to(*message: kudubot.entities.Message.Message*) → bool
Checks if the Service is applicable to a given message

Parameters **message** – The message to check

Returns True if applicable, else False

rules = [({':)': ':) :) :)'}, <function SimpleResponderService.<lambda>>), ({'ping'
These are the rule-based responses for the Simple Responder Service

The consist of a Tuple of a dictionary mapping strings to each other, as well as a lambda expression that return a boolean value. If that lambda expression returns true, the pattern matches and the dictionary value is returned as a response

Module contents

Module contents

Submodules

kudubot.services.AuthenticatedService module

class kudubot.services.AuthenticatedService.**AuthenticatedService**(*connection*)
Bases: *kudubot.services.BaseService.BaseService*

Service that can only be used by admin users

is_applicable_to_authenticated(*message: kudubot.entities.Message.Message*) → bool
Checks if the sender of the message has administrative privileges :param message: The message to check :return: True if the sender is an admin, False otherwise

kudubot.services.BaseService module

class kudubot.services.BaseService.**BaseService**(*connection*)
Bases: object

A class that defines how a chat bot service integrates with a Connection. Includes various helper methods.

__init__(*connection*)
Initializes the Service using a specified Connection

Parameters **connection** – The connection used by this service

static define_identifier() → str
Defines the unique identifier for the service

Returns The Service's identifier.

handle_message(*message: kudubot.entities.Message.Message*)
Handles the message, provided this service is applicable to it

Parameters **message** – The message to process

Returns None

handle_message_with_log(*message: kudubot.entities.Message.Message*)
Wrapper around the handle_message method, which enables logging the message easily for all subclasses

Parameters **message** – The message to handle

Returns None

init()
Helper method that runs after the initialization of the Service object. Can be used for anything, but normal use cases would include initializing a database table or starting a background thread

Returns None

initialize_database_table(*sql: List[str] = []*, *initializer: <built-in function callable> = None*)
Executes the provided SQL queries to create the database table(s).

Parameters

- **sql** – The SQL queries used to create the database tables
- **initializer** – A method that initializes the database connection itself.

Returns None

is_applicable_to(*message: kudubot.entities.Message.Message*) → bool
Checks if the Service is applicable to a given message

Parameters **message** – The message to check

Returns True if applicable, else False

is_applicable_to_with_log(*message: kudubot.entities.Message.Message*) → bool
Wrapper around the is_applicable_to method, which enables logging the message easily for all subclasses

Parameters **message** – The message to analyze

Returns True if the message is applicable, False otherwise

reply(*title: str*, *body: str*, *message: kudubot.entities.Message.Message*)
Provides a helper method that streamlines the process of replying to a message. Very useful for Services that send a reply immediately to cut down on clutter in the code

Parameters

- **title** – The title of the message to send
- **body** – The body of the message to send
- **message** – The message to reply to

Returns None

start_daemon_thread(*target: <built-in function callable>*) → threading.Thread
Starts a daemon/background thread :param target: The target function to execute as a separate thread :return: The thread

kudubot.services.HelperService module

class kudubot.services.HelperService.**HelperService**(*connection*)
Bases: *kudubot.services.MultiLanguageService.MultiLanguageService*

Service extension that allows for the automatic sending of help and syntax messages. Provides support for multiple languages

define_command_name(*language: str*) → str
Defines the command name used to call this Service

Parameters **language** – The language for the command name, for supporting different command names for different languages

Returns The command name for this service. Defaults to a forward slash and the Service's identifier

define_help_message(*language: str*) → str
Defines the help message for the Service

Parameters **language** – The language in which to get the help message in

Returns The help message in the specified language

define_syntax_description(*language: str*) → str
Defines the syntax description for this service.

Parameters **language** – The language in which to get the syntax description in

Returns The syntax description in the specified language

handle_message_helper(*message: kudubot.entities.Message.Message*)
Handles the help message sending. Checks if a message qualifies for a help message and then sends messages accordingly.

Parameters **message** – The message to handle

Returns None

is_applicable_to_helper(*message: kudubot.entities.Message.Message*) → bool
Checks if the message is applicable to the service by checking if the command name is followed by the terms 'help' or 'syntax'.

Parameters **message** – The message to analyze

Returns True if the message is applicable, False otherwise

starts_with_command_keyword(*message: kudubot.entities.Message.Message*, *language: str*, *case_sensitive: bool = False*) → bool

Checks if a message text starts with the command keyword defined by define_command_name()

Parameters

- **message** – The message to check
- **language** – The language for which to check
- **case_sensitive** – Can be set to True to do a case-sensitive check

Returns True if the message starts with the command name, else False

kudubot.services.MultiLanguageService module

class kudubot.services.MultiLanguageService.**MultiLanguageService**(*connection*)

Bases: *kudubot.services.BaseService.BaseService*

Service Extension class that enables support for multiple languages

define_fallback_language() → str

Defines a fallback language in case a language is not implemented for a key.

Returns By default, the language "en" is returned

define_language_text() → Dict[str, Dict[str, str]]

Defines the dictionary with which the text is translated in the translate() method. This should be in the form of a dictionary like this:

{ key: {"language": "text_in_language", ... }, ... }

Keep in mind that every instance of the 'key' value is replaced while translating

Returns The dictionary to create translations with

determine_language(*message: kudubot.entities.Message.Message*) → str

Checks a message object for any language indicators to determine in which language to reply with

Parameters **message** – The message to analyze

Returns The language key

get_stored_language_preference(*contact: kudubot.users.Contact.Contact*) → str

Gets the stored language preference of a contact :param contact: The contact to get the stored language preference for :return: The language

handle_message_multi_language(*message: kudubot.entities.Message.Message*)

Analyzes a message for the language used and stores that language value in the database as a preference of the user. Also implements the /language command which allows a user to view and change the current language

Parameters **message** – The message to analyze

Returns None

is_applicable_to_multi_language(*message: kudubot.entities.Message.Message*)

Checks if a message is applicable for a language list or change :param message: :return:

lang_switch_aliases = {'de': ['de', 'deutsch', 'german'], 'en': ['en', 'english', 'e

Aliases for language names in other languages

lang_switch_command_keywords = ['/language', '/sprache']
List of command triggers that can be used to list or change the language

reply_translated(*title: str*, *body: str*, *message: kudubot.entities.Message.Message*)
Provides a helper method that streamlines the process of replying to a message. Very useful for Services that send a reply immediately to cut down on clutter in the code

In addition to the standard reply method, this method translates the text prior to sending it

Parameters

- **title** – The title of the message to send
- **body** – The body of the message to send
- **message** – The message to reply to

Returns None

supported_languages() → List[str]

Returns A list of languages supported by the Service

translate(*text: str*, *language: str*, *translation_dict: Dict[str, Dict[str, str]] = None*) → str
Translates text using the service's dictionary in the specified language

Parameters

- **text** – The text to translate
- **language** – The language to translate into
- **translation_dict** – Can be specified to determine a custom dictionary

Returns The translated text

kudubot.services.TemplateService module

class kudubot.services.TemplateService.**TemplateService**(*connection*)
Bases: *kudubot.services.HelperService.HelperService*

This is a Template for a Service class. # TODO Change this description

define_command_name(*language: str*) → str
Defines the command prefix for the service. By default, '/' followed by the service identifier from self.define_identifier() is used.

Parameters **language** – The language in which to define the command name

Returns The command name in the specified language

define_help_message(*language: str*) → str
Defines the help message of the service in various languages

Parameters **language** – The language to use

Returns The help message in the language

static define_identifier() → str
Defines the identifier for this service

Returns The service's identifier

define_language_text() → Dict[str, Dict[str, str]]
Defines the dictionary used for translating strings using the self.reply_translated() or self.translate() methods

The format is:

term: {lang: value}

Returns The dictionary for use in translating

define_syntax_description(*language: str*) → str
Defines the Syntax description of the Service in various languages

Parameters **language** – The language in which to return the syntax message

Returns The syntax message

determine_language(*message: kudubot.entities.Message.Message*) → str
Determines the language of a message

Parameters **message** – The message to check for the language

Returns The language of the message

handle_message(*message: kudubot.entities.Message.Message*)
Handles an applicable message # TODO Describe what this Service does

Parameters **message** – The message to handle

is_applicable_to(*message: kudubot.entities.Message.Message*) → bool
Checks if a Message is applicable to this Service # TODO Describe what this Service checks for

Parameters **message** – The message to check

Returns True if the Message is applicable, False otherwise

Module contents

1.1.5 kudubot.tests package

Subpackages

kudubot.tests.config package

Submodules

kudubot.tests.config.test_GlobalConfigHandler module

class kudubot.tests.config.test_GlobalConfigHandler.**UnitTests**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Tests the GlobalConfigHandler class

assure_invalid_config_directory()
Makes sure that the current configuration directory is invalid :return: None

setUp()
Creates a restore point for the class variables of the GlobalConfigHandler and sets these values to ones that make sense for the unit tests :return: None

tearDown()
Restores the class variables and deletes any temporary directories and files :return: None

test_connection_loading()
Tests the loading of connections

Returns None

test_directory_validation()
Tests if the global config validation works correctly and finds errors in the config structure :return: None

test_duplicate_removal()
Tests the duplicate removal method

Returns None

test_generating_new_config()
Tests if the configuration generation works as intended :return: None

test_importing()
Tests the String import handler method

Returns None

test_loading_invalid_classes()

test_service_loading()
Tests the loading of services

Returns None

validate_config_directory()
Validates a configuration directory :return: None

Module contents

kudubot.tests.connections package

Submodules

kudubot.tests.connections.test_Connection module

class kudubot.tests.connections.test_Connection.**UnitTests**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Class that tests the Connection class

setUp()

Returns None

tearDown()

Returns None

test_abstract_methods()
Tests if the methods of the connection class are abstract :return: None

test_daemon_thread_start()
Tests if the daemon thread is started correctly :return: None

test_processing_message()
Tests if the connection correctly processes a message using the services :return: None

Module contents

kudubot.tests.entities package

Submodules

kudubot.tests.entities.test_Message module

class kudubot.tests.entities.test_Message.**UnitTests**(*methodName='runTest'*)
Bases: unittest.case.TestCase

setUp()
Hook method for setting up the test fixture before exercising it.

tearDown()
Hook method for deconstructing the test fixture after testing it.

test_creating_message()

Module contents

kudubot.tests.helpers package

Submodules

kudubot.tests.helpers.DummyConnection module

class kudubot.tests.helpers.DummyConnection.**DummyConnection**(*services: List[type], config_handler: kudubot.config.GlobalConfigHandler.GlobalC*
Bases: *kudubot.connections.Connection.Connection*

A class that implements a Connection for use in unit tests

static define_identifier() → str
Defines the connection's identifier

Returns The identifier for the Connection type

define_user_contact() → kudubot.users.Contact.Contact
Creates a Contact object for the connection by which the connection itself is identified

Returns The connection's user object

generate_configuration()
Generates a new configuration file for this connection.

listen()
Starts listening on the connection in an infinite loop. If the execution of the program has to continue past starting the listener, the listen_in_separate_thread() method should be called instead.

load_config() → Dict[str, object]
Loads the configuration for the connection. If this fails forsome reason, an InvalidConfigException is raised

Returns A dictionary containing the configuration

send_audio_message(*receiver: kudubot.users.Contact.Contact*, *audio_file: str*, *caption: str = ''*)
Sends an audio message using the connection

Parameters

- **receiver** – The receiver of the message
- **audio_file** – The path to the audio file to send
- **caption** – The caption sent together with the message

send_image_message(*receiver: kudubot.users.Contact.Contact*, *image_file: str*, *caption: str = ''*)
Sends an image message using the connection

Parameters

- **receiver** – The recipient of the image message
- **image_file** – The path to the image file
- **caption** – The caption to be displayed with the image

send_message(*message: kudubot.entities.Message.Message*)
Sends a Message using the connection

Parameters **message** – The message to send

send_video_message(*receiver: kudubot.users.Contact.Contact*, *video_file: str*, *caption: str = ''*)
Sends a video message using the connection

Parameters

- **receiver** – The recipient of the video message
- **video_file** – The path to the video file to be sent
- **caption** – The caption to be displayed with the video

kudubot.tests.helpers.DummyService module

class kudubot.tests.helpers.DummyService.**DummyService**(*connection*)
Bases: *kudubot.services.BaseService.BaseService*

A class that implements a Service for use in unit tests

static define_identifier() → str
Defines the unique identifier for the service

Returns The Service's identifier.

static define_requirements() → List[str]

handle_message(*message: kudubot.entities.Message.Message*)
Handles the message, provided this service is applicable to it

Parameters **message** – The message to process

Returns None

is_applicable_to(*message: kudubot.entities.Message.Message*) → bool
Checks if the Service is applicable to a given message

Parameters **message** – The message to check

Returns True if applicable, else False

class kudubot.tests.helpers.DummyService.**DummyServiceWithInvalidDependency**(*connection*)
Bases: *kudubot.tests.helpers.DummyService.DummyService*

A service that has an invalid dependency

static define_requirements() → List[str]

class kudubot.tests.helpers.DummyService.**DummyServiceWithValidDependency**(*connection*)
Bases: *kudubot.tests.helpers.DummyService.DummyService*

A Service that has itself (or DummyService) as its only dependency

static define_requirements() → List[str]

kudubot.tests.helpers.test_config module

kudubot.tests.helpers.test_config.**clean_up_test_environment**()
Deletes the test environment

Returns None

kudubot.tests.helpers.test_config.**generate_test_environment**() → kudubot.config.GlobalConfigHandler.GlobalC
Generates the test environment

Returns The GlobalConfigHandler that points to the configuration

Module contents

kudubot.tests.resources package

Module contents

kudubot.tests.services package

Submodules

kudubot.tests.services.test_Service module

class kudubot.tests.services.test_Service.**UnitTests**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Tests the Service class

setUp()

Returns None

tearDown()

Returns None

test_abstract_methods()
Tests if the methods of the Service class are abstract :return: None

Module contents

kudubot.tests.users package

Submodules

kudubot.tests.users.test_AddressBook module

class kudubot.tests.users.test_AddressBook.**UnitTests**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Class that tests the AddressBook class

setUp()
Changes the class variables for testing and creates the testing directory

Returns None

subtest_adding_contact_to_addressbook(*contact: kudubot.users.Contact.Contact*)
Tests adding a contact to the addressbook

Parameters **contact** – The contact to add

Returns None

subtest_fetching_contacts()
Tests fetching the contact information from the database To be used at the end of the test_contact_operations method

Returns None

subtest_updating_contact()
Tests Updating a contact

Returns None

tearDown()
Restores the class variables and deletes the testing directory

Returns None

test_contact_operations()
Tests various contact operations in the addressbook. Uses subtest methods to make the tests a bit more readable

Returns None

test_invalid_contact_fetches()
Tests if the contact fetch methods return a None object if they fail to find a result

Returns None

kudubot.tests.users.test_Contact module

class kudubot.tests.users.test_Contact.**UnitTests**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Tests the Contact class

setUp()

Returns None

tearDown()

Returns None

test_initialization()

Tests if the Contact class is initialized correctly

Returns None

Module contents

Module contents

1.1.6 kudubot.users package

Submodules

kudubot.users.AddressBook module

class kudubot.users.AddressBook.**AddressBook**(*database: sqlite3.Connection*)

Bases: object

Class that tracks and provides user information in the connection's database.

The address book uses the following database schema:

|id|display_name|address|

__init__(*database: sqlite3.Connection*)

Initializes the address book. Makes sure that the address book's database table exists and has the correct schema

Parameters **database** – The database connection to use

add_or_update_contact(*contact: kudubot.users.Contact.Contact*, *database_override: sqlite3.Connection = None*) → kudubot.users.Contact.Contact

Adds or updates a contact in the address book

Parameters

- **contact** – The contact to insert/update
- **database_override** – Can be specified to use a different database connection, useful for calling this method from a different thread

Returns The contact, possibly with an altered id value (in case the contact was inserted, not updated)

get_contact_for_address(*address: str*, *database_override: sqlite3.Connection = None*) → kudubot.users.Contact.Contact

Generates a Contact object for an address in the address book table.

Parameters

- **address** – The address to look for

- **database_override** – Can be specified to use a different database connection, useful for calling this method from a different thread

Returns The Contact object, or None if no contact was found

get_contact_for_id(*user_id: int*, *database_override: sqlite3.Connection = None*) → kudubot.users.Contact.Contact

Generates a Contact object for a user ID in the address book table

Parameters

- **user_id** – The user's ID
- **database_override** – Can be specified to use a different database connection, useful for calling this method from a different thread

Returns The user as a Contact object

logger = <logging.Logger object>

The Logger for this class

kudubot.users.Authenticator module

class kudubot.users.Authenticator.**Authenticator**(*db: sqlite3.Connection*)

Bases: object

Class that handles the authentication of users Offers the identification of users as either admin or blacklisted

__init__(*db: sqlite3.Connection*)

Initialize self. See help(type(self)) for accurate signature.

blacklist(*contact: kudubot.users.Contact.Contact*, *database_override: sqlite3.Connection = None*)

Blacklists a contact :param contact: The contact to blacklist :param database_override: Provides a different database connection.

Required for use in other threads.

is_admin(*contact: kudubot.users.Contact.Contact*, *database_override: sqlite3.Connection = None*) → bool

Checks if a contact has admin privileges :param contact: The contact to check :param database_override: Provides a different database connection.

Required for use in other threads.

Returns True if the contact has admin privileges else False

is_blacklisted(*contact: kudubot.users.Contact.Contact*, *database_override: sqlite3.Connection = None*) → bool

Checks if a contact is blacklisted :param contact: The contact to check :param database_override: Provides a different database connection.

Required for use in other threads.

Returns True if the contact is blacklisted else False

logger = <logging.Logger object>

The Logger for this class

make_admin(*contact: kudubot.users.Contact.Contact*, *database_override: sqlite3.Connection = None*)

Grants admin privileges to a contact :param contact: The contact to grant admin privileges to :param database_override: Provides a different database connection.

Required for use in other threads.

kudubot.users.Contact module

class kudubot.users.Contact.**Contact**(*database_id: int*, *display_name: str*, *address: str*)
Bases: object

Class that models a contact in the connection's address book database

__init__(*database_id: int*, *display_name: str*, *address: str*)
Initializes the contact object

Parameters

- **database_id** – The contact's ID in the database
- **display_name** – The display name of the contact
- **address** – The contact's address

to_dict() → Dict[str, str]

Returns The contact as a dictionary, which can be used to store the contact in a JSON file

kudubot.users.Contact.**from_dict**(*data: Dict[str, str]*) → kudubot.users.Contact.Contact
Generates a Contact object from a dictionary

Parameters **data** – The data to turn into a Contact object

Returns The generated Contact object

kudubot.users.LanguageSelector module

class kudubot.users.LanguageSelector.**LanguageSelector**(*db: sqlite3.Connection*)
Bases: object

__init__(*db: sqlite3.Connection*)
Initialize self. See help(type(self)) for accurate signature.

get_language_preference(*contact: kudubot.users.Contact.Contact*, *default: str = 'en'*, *db: sqlite3.Connection = None*) → str
Retrieves a language from the user's preferences in the database

Parameters

- **contact** – The user to check the language preference for
- **default** – A default language value used in case no entry was found
- **db** – Optionally defines which database connection to use (necessary for access from other thread)

Returns The language preferred by the user

logger = <logging.Logger object>
The Logger for this class

store_language_preference(*contact: kudubot.users.Contact.Contact*, *language: str*, *user_initiated: bool = False*)
Stores the language preference for a user in the sqlite database :param contact: The user for which to store the preference :param language: The language to store :param user_initiated: Flag that should be set whenever the user

manually requests a language store

Returns None

Module contents

1.2 Submodules

1.3 kudubot.exceptions module

exception kudubot.exceptions.**InvalidConfigException**
Bases: Exception

An Exception that is raised whenever an invalid configuration is detected.

1.4 kudubot.main module

kudubot.main.**initialize_connection**(*identifier: str*, *config_handler: kudubot.config.GlobalConfigHandler.GlobalConfigHandler*) → kudubot.connections.Connection.Connection
Loads the connection for the specified identifier If the connection was not found in the local configuration, the program exits.

Parameters

- **identifier** – The identifier for the Connection
- **config_handler** – The config handler to use to determine file paths etc.

Returns The Connection object

kudubot.main.**initialize_logging**(*quiet: bool*, *verbose: bool*, *debug: bool*, *config_handler: kudubot.config.GlobalConfigHandler.GlobalConfigHandler*, *connection_name: str*)
Initializes the logging levels and files for the program. If neither the verbose or debug flags were provided, the logging level defaults to WARNING. Log files for ERROR, WARNING, DEBUG and INFO are always generated. If the size of a previous log file exceeds 1MB, the file is renamed and a new one is created.

Parameters

- **quiet** – Can be set to disable all logging to the console. Text logs are still done however.
- **verbose** – Flag that determines if the verbose mode is switched on ~ INFO
- **debug** – Flag that determines if the debug mode is on ~ DEBUG
- **config_handler** – The config handler used to determine the logging directory location
- **connection_name** – The name of the connection to log

kudubot.main.**main**()
The Main Method of the Program that starts the Connection Listener in accordance with the command line arguments

kudubot.main.**parse_args**() → argparse.Namespace
Parses the Command Line Arguments using argparse

Returns The parsed arguments

1.5 Module contents

CHAPTER

TWO

KUDUBOT

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

k

kudubot, 40
kudubot.config, 4
kudubot.config.GlobalConfigHandler, 3
kudubot.config.StandardConfigWriter, 4
kudubot.connections, 14
kudubot.connections.cli, 6
kudubot.connections.cli.CliConnection, 4
kudubot.connections.Connection, 12
kudubot.connections.telegram, 7
kudubot.connections.telegram.TelegramConnection, 6
kudubot.connections.whatsapp, 12
kudubot.connections.whatsapp.EchoLayer, 7
kudubot.connections.whatsapp.WhatsappConnection, 11
kudubot.entities, 15
kudubot.entities.Message, 14
kudubot.exceptions, 39
kudubot.main, 39
kudubot.services, 30
kudubot.services.AuthenticatedService, 25
kudubot.services.BaseService, 26
kudubot.services.HelperService, 27
kudubot.services.internal, 25
kudubot.services.internal.anime_reminder, 18
kudubot.services.internal.anime_reminder.AnimeReminderService, 15
kudubot.services.internal.anime_reminder.database, 16
kudubot.services.internal.anime_reminder.scraper, 18
kudubot.services.internal.authentication_manager, 19
kudubot.services.internal.authentication_manager.AuthenticationManagerService, 18
kudubot.services.internal.echo, 20
kudubot.services.internal.echo.EchoService, 19
kudubot.services.internal.jokes, 21
kudubot.services.internal.jokes.JokesService, 20
kudubot.services.internal.reminder, 24
kudubot.services.internal.reminder.database, 22
kudubot.services.internal.reminder.ReminderService, 21
kudubot.services.internal.service_lister, 25
kudubot.services.internal.service_lister.ServiceLis 24
kudubot.services.internal.simple_responder, 25
kudubot.services.internal.simple_responder.SimpleRe 25
kudubot.services.MultiLanguageService, 28
kudubot.services.TemplateService, 29
kudubot.tests, 36
kudubot.tests.config, 31
kudubot.tests.config.test_GlobalConfigHandler, 30
kudubot.tests.connections, 32
kudubot.tests.connections.test_Connection, 31
kudubot.tests.entities, 32
kudubot.tests.entities.test_Message, 32
kudubot.tests.helpers, 34
kudubot.tests.helpers.DummyConnection, 32
kudubot.tests.helpers.DummyService, 33
kudubot.tests.helpers.test_config, 34
kudubot.tests.resources, 34
kudubot.tests.services, 35
kudubot.tests.services.test_Service, 34
kudubot.tests.users, 36
kudubot.tests.users.test_AddressBook, 35
kudubot.tests.users.test_Contact, 35
kudubot.users, 39

kudubot.users.AddressBook, 36
kudubot.users.Authenticator, 37
kudubot.users.Contact, 38
kudubot.users.LanguageSelector, 38

INDEX

Symbols

__init__() (kudubot.config.GlobalConfigHandler.GlobalConfigHandler method), 3
__init__() (kudubot.config.StandardConfigWriter.StandardConfigWriter method), 4
__init__() (kudubot.connections.Connection.Connection method), 12
__init__() (kudubot.connections.telegram.TelegramConnection.TelegramConnection method), 6
__init__() (kudubot.connections.whatsapp.EchoLayer.EchoLayer method), 7
__init__() (kudubot.connections.whatsapp.WhatsappConnection.WhatsappConnection method), 11
__init__() (kudubot.entities.Message.Message method), 14
__init__() (kudubot.services.BaseService.BaseService method), 26
__init__() (kudubot.users.AddressBook.AddressBook method), 36
__init__() (kudubot.users.Authenticator.Authenticator method), 37
__init__() (kudubot.users.Contact.Contact method), 38
__init__() (kudubot.users.LanguageSelector.LanguageSelector method), 38

A

add_or_update_contact() (kudubot.users.AddressBook.AddressBook method), 36
AddressBook (class in kudubot.users.AddressBook), 36
alias_to_jid() (kudubot.connections.whatsapp.EchoLayer.EchoLayer method), 7
AnimeReminderService (class in kudubot.services.internal.anime_reminder.AnimeReminderService), 15
apply_services() (kudubot.connections.Connection.Connection method), 12
assure_invalid_config_directory() (kudubot.tests.config.test_GlobalConfigHandler.UnitTests method), 30
AuthenticatedService (class in kudubot.services.AuthenticatedService), 25
AuthenticationManagerService (class in kudubot.services.internal.authentication_manager.AuthenticationManagerService), 18
Authenticator (class in kudubot.users.Authenticator), 37

B

background_loop() (kudubot.services.internal.anime_reminder.AnimeReminderService method), 15
background_loop() (kudubot.services.internal.reminder.ReminderService.ReminderService method), 21
BaseService (class in kudubot.services.BaseService), 26
blacklist() (kudubot.users.Authenticator.Authenticator method), 37

C

clean_up_test_environment() (in module kudubot.tests.helpers.test_config), 34
CliConnection (class in kudubot.connections.cli.CliConnection), 4
Connection (class in kudubot.connections.Connection), 12
Contact (class in kudubot.users.Contact), 38
convert_datetime_to_string() (in module kudubot.services.internal.reminder.database), 22
convert_string_to_datetime() (in module kudubot.services.internal.reminder.database), 22
create_audio_media() (kudubot.connections.whatsapp.EchoLayer.EchoLayer static method), 7

D

define_command_name() (kudubot.services.HelperService.HelperService method), 27
define_command_name() (kudubot.services.internal.anime_reminder.AnimeReminderService method), 15
define_command_name() (kudubot.services.internal.authentication_manager.Authentication

method), 18
define_command_name() (kudubot.services.internal.reminder.ReminderService.ReminderService method), 21
define_command_name() (kudubot.services.internal.service_lister.ServiceListerService.ServiceListerService method), 24
define_command_name() (kudubot.services.TemplateService.TemplateService method), 29
define_fallback_language() (kudubot.services.MultiLanguageService.MultiLanguageService method), 28
define_help_message() (kudubot.services.HelperService.HelperService method), 27
define_help_message() (kudubot.services.internal.anime_reminder.AnimeReminderService.AnimeReminderService method), 15
define_help_message() (kudubot.services.internal.authentication_manager.AuthenticationManagerService.AuthenticationManagerService method), 18
define_help_message() (kudubot.services.internal.jokes.JokesService.JokesService method), 20
define_help_message() (kudubot.services.internal.reminder.ReminderService.ReminderService method), 21
define_help_message() (kudubot.services.internal.service_lister.ServiceListerService.ServiceListerService method), 24
define_help_message() (kudubot.services.TemplateService.TemplateService method), 29
define_identifier() (kudubot.connections.cli.CliConnection.CliConnection static method), 4
define_identifier() (kudubot.connections.Connection.Connection static method), 12
define_identifier() (kudubot.connections.telegram.TelegramConnection.TelegramConnection static method), 6
define_identifier() (kudubot.connections.whatsapp.WhatsappConnection.WhatsappConnection static method), 11
define_identifier() (kudubot.services.BaseService.BaseService static method), 26
define_identifier() (kudubot.services.internal.anime_reminder.AnimeReminderService.AnimeReminderService static method), 15
define_identifier() (kudubot.services.internal.authentication_manager.AuthenticationManagerService.AuthenticationManagerService static method), 18
define_identifier() (kudubot.services.internal.echo.EchoService.EchoService static method), 19
define_identifier() (kudubot.services.internal.jokes.JokesService.JokesService static method), 20
define_identifier() (kudubot.services.internal.reminder.ReminderService.ReminderService static method), 21
define_identifier() (kudubot.services.internal.service_lister.ServiceListerService.ServiceListerService static method), 24
define_identifier() (kudubot.services.internal.simple_responder.SimpleResponderService.SimpleResponderService static method), 25
define_identifier() (kudubot.services.TemplateService.TemplateService static method), 29
define_identifier() (kudubot.tests.helpers.DummyConnection.DummyConnection static method), 32
define_identifier() (kudubot.tests.helpers.DummyService.DummyService static method), 33
define_language_text() (kudubot.services.internal.anime_reminder.AnimeReminderService.AnimeReminderService method), 15
define_language_text() (kudubot.services.internal.authentication_manager.AuthenticationManagerService.AuthenticationManagerService method), 18
define_language_text() (kudubot.services.internal.jokes.JokesService.JokesService method), 20
define_language_text() (kudubot.services.internal.reminder.ReminderService.ReminderService method), 21
define_language_text() (kudubot.services.internal.service_lister.ServiceListerService.ServiceListerService method), 24
define_language_text() (kudubot.services.MultiLanguageService.MultiLanguageService method), 28
define_language_text() (kudubot.services.TemplateService.TemplateService method), 29
define_required_auths() (kudubot.tests.helpers.DummyService.DummyService static method), 33
define_required_auths() (kudubot.tests.helpers.DummyService.DummyService static method), 34
define_required_auths() (kudubot.tests.helpers.DummyService.DummyService static method), 34
define_service_description() (kudubot.services.ServiceListerService.ServiceListerService method), 24
define_service_description() (kudubot.services.HelperService.HelperService method), 27
define_syntax_description() (kudubot.services.internal.anime_reminder.AnimeReminderService.AnimeReminderService method), 15
define_syntax_description() (kudubot.services.internal.authentication_manager.AuthenticationManagerService.AuthenticationManagerService method), 18
define_syntax_description() (kudubot.services.internal.jokes.JokesService.JokesService method), 20
define_syntax_description() (kudubot.services.internal.reminder.ReminderService.ReminderService method), 21
define_syntax_description() (kudubot.services.internal.service_lister.ServiceListerService.ServiceListerService method), 24
define_syntax_description() (kudubot.services.TemplateService.TemplateService method), 30
define_user_contact() (kudubot.connections.cli.CliConnection.CliConnection method), 4
define_user_contact() (kudubot.connections.Connection.Connection method), 12
define_user_contact() (kudubot.connections.telegram.TelegramConnection.TelegramConnection method), 6
define_user_contact() (kudubot.connections.whatsapp.WhatsappConnection method), 11
define_user_contact() (kudubot.tests.helpers.DummyConnection.DummyConnection method), 32

delete_subscription() (in module kudubot.services.internal.anime_reminder.database), 16
determine_language() (kudubot.services.internal.anime_reminder.AnimeReminder.AnimeReminder method), 15
determine_language() (kudubot.services.internal.authentication_manager.AuthenticationManagerService.AuthenticationManagerService method), 19
determine_language() (kudubot.services.internal.jokes.JokesService.JokesService method), 20
determine_language() (kudubot.services.internal.reminder.ReminderService.ReminderService method), 21
determine_language() (kudubot.services.internal.service_lister.ServiceListerService.ServiceListerService method), 24
determine_language() (kudubot.services.MultiLanguageService.MultiLanguageService method), 28
determine_language() (kudubot.services.TemplateService.TemplateService method), 30
do_send_media() (kudubot.connections.whatsapp.EchoLayer.EchoLayer method), 8
DummyConnection (class in kudubot.tests.helpers.DummyConnection), 32
DummyService (class in kudubot.tests.helpers.DummyService), 33
DummyServiceWithInvalidDependency (class in kudubot.tests.helpers.DummyService), 34
DummyServiceWithValidDependency (class in kudubot.tests.helpers.DummyService), 34

E

EchoLayer (class in kudubot.connections.whatsapp.EchoLayer), 7
EchoService (class in kudubot.services.internal.echo.EchoService), 19

F

from_dict() (in module kudubot.entities.Message), 14
from_dict() (in module kudubot.users.Contact), 38

G

generate_configuration() (kudubot.config.GlobalConfigHandler.GlobalConfigHandler method), 3
generate_configuration() (kudubot.connections.cli.CliConnection.CliConnection method), 4
generate_configuration() (kudubot.connections.Connection.Connection method), 13
generate_configuration() (kudubot.connections.telegram.TelegramConnection.TelegramConnection method), 6
generate_configuration() (kudubot.connections.whatsapp.WhatsappConnection.WhatsappConnection method), 11
generate_configuration() (kudubot.tests.helpers.DummyConnection.DummyConnection method), 32
generate_test_environment() (in module kudubot.tests.helpers.test_config), 34
get_contact_for_address() (kudubot.users.AddressBook.AddressBook method), 36
get_contact_id() (kudubot.users.AddressBook.AddressBook method), 37
get_database_connection_copy() (kudubot.connections.Connection.Connection method), 13
get_direct_response_contact() (kudubot.entities.Message.Message method), 14
get_language_preference() (kudubot.users.LanguageSelector.LanguageSelector method), 38
get_next_id() (in module kudubot.services.internal.reminder.database), 22
get_notified_language_preference() (kudubot.services.MultiLanguageService.MultiLanguageService method), 28
get_subscriptions() (in module kudubot.services.internal.anime_reminder.database), 16
get_unsent_reminders() (in module kudubot.services.internal.reminder.database), 23
GlobalConfigHandler (class in kudubot.config.GlobalConfigHandler), 3

H

handle_message() (kudubot.services.BaseService.BaseService method), 26
handle_message() (kudubot.services.internal.anime_reminder.AnimeReminder method), 16
handle_message() (kudubot.services.internal.authentication_manager.AuthenticationManagerService method), 19
handle_message() (kudubot.services.internal.echo.EchoService.EchoService method), 19
handle_message() (kudubot.services.internal.jokes.JokesService.JokesService method), 20
handle_message() (kudubot.services.internal.reminder.ReminderService.ReminderService method), 21
handle_message() (kudubot.services.internal.service_lister.ServiceListerService method), 24
handle_message() (kudubot.services.internal.simple_responder.SimpleResponder method), 25
handle_message() (kudubot.services.TemplateService.TemplateService method), 30
handle_message() (kudubot.tests.helpers.DummyService.DummyService method), 33
handle_message_helper() (kudubot.services.HelperService.HelperService method), 27
handle_message_multi_language()

(kudubot.services.MultiLanguageService.MultiLanguageService method), 28
handle_message_with_log() (kudubot.services.BaseService.BaseService method), 26
HelperService (class in kudubot.services.HelperService), 27

I

init() (kudubot.services.BaseService.BaseService method), 26
init() (kudubot.services.internal.anime_reminder.AnimeReminderService.AnimeReminderService method), 16
init() (kudubot.services.internal.reminder.ReminderService.ReminderService method), 21
initialize_connection() (in module kudubot.main), 39
initialize_database() (in module kudubot.services.internal.anime_reminder.database), 16
initialize_database() (in module kudubot.services.internal.reminder.database), 23
initialize_database_table() (kudubot.services.BaseService.BaseService method), 26
initialize_logging() (in module kudubot.main), 39
InvalidConfigException, 39
is_admin() (kudubot.users.Authenticator.Authenticator method), 37
is_applicable_to() (kudubot.services.BaseService.BaseService method), 26
is_applicable_to() (kudubot.services.internal.anime_reminder.AnimeReminderService.AnimeReminderService method), 16
is_applicable_to() (kudubot.services.internal.authentication_manager.AuthenticationManagerService.AuthenticationManagerService method), 19
is_applicable_to() (kudubot.services.internal.echo.EchoService.EchoService method), 19
is_applicable_to() (kudubot.services.internal.jokes.JokesService.JokesService method), 20
is_applicable_to() (kudubot.services.internal.reminder.ReminderService.ReminderService method), 21
is_applicable_to() (kudubot.services.internal.service_lister.ServiceListerService.ServiceListerService method), 24
is_applicable_to() (kudubot.services.internal.simple_responder.SimpleResponderService.SimpleResponderService method), 25
is_applicable_to() (kudubot.services.TemplateService.TemplateService method), 30
is_applicable_to() (kudubot.tests.helpers.DummyService.DummyService method), 33
is_applicable_to_authenticated() (kudubot.services.AuthenticatedService.AuthenticatedService method), 26
is_applicable_to_helper() (kudubot.services.HelperService.HelperService method), 27
is_applicable_to_multi_language() (kudubot.services.MultiLanguageService.MultiLanguageService method), 28
is_applicable_to_with_log() (kudubot.services.BaseService.BaseService method), 26
is_blacklisted() (kudubot.users.Authenticator.Authenticator method), 37

J

JokesService (class in kudubot.services.internal.jokes.JokesService), 20

K

kudubot (module), 40
kudubot.config (module), 4
kudubot.config.GlobalConfigHandler (module), 3
kudubot.config.StandardConfigWriter (module), 4
kudubot.connections (module), 14
kudubot.connections.cli (module), 6
kudubot.connections.cli.CliConnection (module), 4
kudubot.connections.Connection (module), 12
kudubot.connections.telegram (module), 7
kudubot.connections.telegram.TelegramConnection (module), 6
kudubot.connections.whatsapp (module), 12
kudubot.connections.whatsapp.EchoLayer (module), 7
kudubot.connections.whatsapp.WhatsappConnection (module), 11
kudubot.entities (module), 15
kudubot.entities.Message (module), 14
kudubot.exceptions (module), 39
kudubot.main (module), 39
kudubot.services (module), 30
kudubot.services.AuthenticatedService (module), 25
kudubot.services.BaseService (module), 26
kudubot.services.HelperService (module), 27
kudubot.services.internal (module), 25
kudubot.services.internal.anime_reminder (module), 18
kudubot.services.internal.anime_reminder.AnimeReminderService (module), 15
kudubot.services.internal.anime_reminder.database (module), 16
kudubot.services.internal.anime_reminder.scraper (module), 18
kudubot.services.internal.authentication_manager (module), 19
kudubot.services.internal.authentication_manager.AuthenticationManagerS (module), 18
kudubot.services.internal.echo (module), 20
kudubot.services.internal.echo.EchoService (module), 19
kudubot.services.internal.jokes (module), 21

kudubot.services.internal.jokes.JokesService (module), 20
kudubot.services.internal.reminder (module), 24
kudubot.services.internal.reminder.database (module), 22
kudubot.services.internal.reminder.ReminderService (module), 21
kudubot.services.internal.service_lister (module), 25
kudubot.services.internal.service_lister.ServiceListerService (module), 24
kudubot.services.internal.simple_responder (module), 25
kudubot.services.internal.simple_responder.SimpleResponderService (module), 25
kudubot.services.MultiLanguageService (module), 28
kudubot.services.TemplateService (module), 29
kudubot.tests (module), 36
kudubot.tests.config (module), 31
kudubot.tests.config.test_GlobalConfigHandler (module), 30
kudubot.tests.connections (module), 32
kudubot.tests.connections.test_Connection (module), 31
kudubot.tests.entities (module), 32
kudubot.tests.entities.test_Message (module), 32
kudubot.tests.helpers (module), 34
kudubot.tests.helpers.DummyConnection (module), 32
kudubot.tests.helpers.DummyService (module), 33
kudubot.tests.helpers.test_config (module), 34
kudubot.tests.resources (module), 34
kudubot.tests.services (module), 35
kudubot.tests.services.test_Service (module), 34
kudubot.tests.users (module), 36
kudubot.tests.users.test_AddressBook (module), 35
kudubot.tests.users.test_Contact (module), 35
kudubot.users (module), 39
kudubot.users.AddressBook (module), 36
kudubot.users.Authenticator (module), 37
kudubot.users.Contact (module), 38
kudubot.users.LanguageSelector (module), 38

L

lang_switch_aliases (kudubot.services.MultiLanguageService.MultiLanguageService attribute), 28
lang_switch_command_keywords (kudubot.services.MultiLanguageService.MultiLanguageService attribute), 28
LanguageSelector (class in kudubot.users.LanguageSelector), 38
listen() (kudubot.connections.cli.CliConnection.CliConnection method), 5
listen() (kudubot.connections.Connection.Connection method), 13
listen() (kudubot.connections.telegram.TelegramConnection.TelegramConnection method), 6
listen() (kudubot.connections.whatsapp.WhatsappConnection.WhatsappConnection method), 11
listen() (kudubot.tests.helpers.DummyConnection.DummyConnection method), 32
listen_in_separate_thread() (kudubot.connections.Connection.Connection method), 13
load_config() (kudubot.connections.cli.CliConnection.CliConnection method), 5
load_config() (kudubot.connections.Connection.Connection method), 13
load_config() (kudubot.connections.telegram.TelegramConnection.TelegramConnection method), 6
load_config() (kudubot.connections.whatsapp.WhatsappConnection.WhatsappConnection method), 11
load_config() (kudubot.tests.helpers.DummyConnection.DummyConnection method), 32
load_connections() (kudubot.config.GlobalConfigHandler.GlobalConfigHandler method), 3
load_english_joke() (kudubot.services.internal.jokes.JokesService.JokesService method), 20
load_german_joke() (kudubot.services.internal.jokes.JokesService.JokesService method), 20
load_services() (kudubot.config.GlobalConfigHandler.GlobalConfigHandler method), 3
logger (in module kudubot.services.internal.anime_reminder.database), 17
logger (in module kudubot.services.internal.anime_reminder.scraper), 18
logger (in module kudubot.services.internal.reminder.database), 23
logger (kudubot.config.GlobalConfigHandler.GlobalConfigHandler attribute), 3
logger (kudubot.users.AddressBook.AddressBook attribute), 37
logger (kudubot.users.Authenticator.Authenticator attribute), 37
logger (kudubot.users.LanguageSelector.LanguageSelector attribute), 38

M

main() (in module kudubot.main), 39
make_admin() (kudubot.users.Authenticator.Authenticator method), 37
mark_reminder_sent() (in module kudubot.services.internal.reminder.database), 23
Message (class in kudubot.entities.Message), 14
MultiLanguageService (class in kudubot.services.MultiLanguageService), 28

N

normalize_jid() (kudubot.connections.whatsapp.EchoLayer.EchoLayer static method), 8

O

on_message() (kudubot.connections.whatsapp.EchoLayer.EchoLayer method), 8
on_receipt() (kudubot.connections.whatsapp.EchoLayer.EchoLayer method), 8
on_request_upload_error() (kudubot.connections.whatsapp.EchoLayer.EchoLayer static method), 8
on_request_upload_result() (kudubot.connections.whatsapp.EchoLayer.EchoLayer method), 9
on_upload_error() (kudubot.connections.whatsapp.EchoLayer.EchoLayer static method), 9
on_upload_progress() (kudubot.connections.whatsapp.EchoLayer.EchoLayer static method), 9

P

parse_args() (in module kudubot.main), 39
parse_message() (kudubot.services.internal.reminder.ReminderService.ReminderService method), 22
parse_time_string() (kudubot.services.internal.reminder.ReminderService.ReminderService method), 22

R

ReminderService (class in kudubot.services.internal.reminder.ReminderService), 21
reply() (kudubot.services.BaseService.BaseService method), 27
reply_translated() (kudubot.services.MultiLanguageService.MultiLanguageService method), 29
rules (kudubot.services.internal.simple_responder.SimpleResponderService.SimpleResponderService attribute), 25

S

scrape_reddit_discussion_threads() (in module kudubot.services.internal.anime_reminder.scraper), 18
send_audio_message() (kudubot.connections.cli.CliConnection.CliConnection method), 5
send_audio_message() (kudubot.connections.Connection.Connection method), 13
send_audio_message() (kudubot.connections.telegram.TelegramConnection.TelegramConnection method), 6
send_audio_message() (kudubot.connections.whatsapp.EchoLayer.EchoLayer method), 9
send_audio_message() (kudubot.connections.whatsapp.WhatsappConnection.WhatsappConnection method), 11
send_audio_message() (kudubot.tests.helpers.DummyConnection.DummyConnection method), 33
send_image_message() (kudubot.connections.cli.CliConnection.CliConnection method), 5
send_image_message() (kudubot.connections.Connection.Connection method), 13
send_image_message() (kudubot.connections.telegram.TelegramConnection method), 7
send_image_message() (kudubot.connections.whatsapp.EchoLayer.EchoLayer method), 10
send_image_message() (kudubot.connections.whatsapp.WhatsappConnection method), 11
send_image_message() (kudubot.tests.helpers.DummyConnection.DummyConnection method), 33
send_media() (kudubot.connections.whatsapp.EchoLayer.EchoLayer method), 10
send_message() (kudubot.connections.cli.CliConnection.CliConnection method), 5
send_message() (kudubot.connections.Connection.Connection method), 13
send_message() (kudubot.connections.telegram.TelegramConnection.TelegramConnection method), 7
send_message() (kudubot.connections.whatsapp.WhatsappConnection.WhatsappConnection method), 12
send_message() (kudubot.tests.helpers.DummyConnection.DummyConnection method), 33
send_receipt() (kudubot.connections.whatsapp.EchoLayer.EchoLayer method), 10
send_text_message() (kudubot.connections.whatsapp.EchoLayer.EchoLayer method), 10
send_video_message() (kudubot.connections.cli.CliConnection.CliConnection method), 5
send_video_message() (kudubot.connections.Connection.Connection method), 13
send_video_message() (kudubot.connections.telegram.TelegramConnection.TelegramConnection method), 7
send_video_message() (kudubot.connections.whatsapp.EchoLayer.EchoLayer method), 10
send_video_message() (kudubot.connections.whatsapp.WhatsappConnection.WhatsappConnection method), 12
send_video_message() (kudubot.tests.helpers.DummyConnection.DummyConnection method), 33
ServiceListerService (class in kudubot.services.internal.service_lister.ServiceListerService), 24
setUp() (kudubot.tests.config.test_GlobalConfigHandler.UnitTests method), 30
setUp() (kudubot.tests.connections.test_Connection.UnitTests method), 31
setUp() (kudubot.tests.entities.test_Message.UnitTests method), 32
setUp() (kudubot.tests.services.test_Service.UnitTests method), 34
setUp() (kudubot.tests.users.test_AddressBook.UnitTests method), 35
setUp() (kudubot.tests.users.test_Contact.UnitTests method), 36
SimpleResponderService (class in kudubot.services.internal.simple_responder.SimpleResponderService), 25

StandardConfigWriter (class in kudubot.config.StandardConfigWriter), 4
start_daemon_thread() (kudubot.services.BaseService.BaseService method), 27
starts_with_command_keyword() (kudubot.services.HelperService.HelperService method), 28
store_language_preference() (kudubot.users.LanguageSelector.LanguageSelector method), 38
store_reminder() (in module kudubot.services.internal.reminder.database), 23
store_subscription() (in module kudubot.services.internal.anime_reminder.database), 17
store_thread() (in module kudubot.services.internal.anime_reminder.database), 17
subscription_exists() (in module kudubot.services.internal.anime_reminder.database), 17
subtest_adding_contact_to_addressbook() (kudubot.tests.users.test_AddressBook.UnitTests method), 35
subtest_fetching_contacts() (kudubot.tests.users.test_AddressBook.UnitTests method), 35
subtest_updating_contact() (kudubot.tests.users.test_AddressBook.UnitTests method), 35
supported_languages() (kudubot.services.MultiLanguageService.MultiLanguageService method), 29

T

tearDown() (kudubot.tests.config.test_GlobalConfigHandler.UnitTests method), 30
tearDown() (kudubot.tests.connections.test_Connection.UnitTests method), 31
tearDown() (kudubot.tests.entities.test_Message.UnitTests method), 32
tearDown() (kudubot.tests.services.test_Service.UnitTests method), 34
tearDown() (kudubot.tests.users.test_AddressBook.UnitTests method), 35
tearDown() (kudubot.tests.users.test_Contact.UnitTests method), 36
TelegramConnection (class in kudubot.connections.telegram.TelegramConnection), 6
TemplateService (class in kudubot.services.TemplateService), 29
test_abstract_methods() (kudubot.tests.connections.test_Connection.UnitTests method), 31
test_abstract_methods() (kudubot.tests.services.test_Service.UnitTests method), 34
test_connection_loading() (kudubot.tests.config.test_GlobalConfigHandler.UnitTests method), 31
test_contact_operations() (kudubot.tests.users.test_AddressBook.UnitTests method), 35
test_creating_message() (kudubot.tests.entities.test_Message.UnitTests method), 32
test_daemon_thread_start() (kudubot.tests.connections.test_Connection.UnitTests method), 31
test_directory_validation() (kudubot.tests.config.test_GlobalConfigHandler.UnitTests method), 31
test_duplicate_removal() (kudubot.tests.config.test_GlobalConfigHandler.UnitTests method), 31
test_generating_new_config() (kudubot.tests.config.test_GlobalConfigHandler.UnitTests method), 31
test_importing() (kudubot.tests.config.test_GlobalConfigHandler.UnitTests method), 31
test_initialization() (kudubot.tests.users.test_Contact.UnitTests method), 36
test_invalid_contact_fetches() (kudubot.tests.users.test_AddressBook.UnitTests method), 35
test_loading_invalid_classes() (kudubot.tests.config.test_GlobalConfigHandler.UnitTests method), 31
test_processing_message() (kudubot.tests.connections.test_Connection.UnitTests method), 31
test_service_loading() (kudubot.tests.config.test_GlobalConfigHandler.UnitTests method), 31
thread_exists() (in module kudubot.services.internal.anime_reminder.database), 17
to_dict() (kudubot.entities.Message.Message method), 14
to_dict() (kudubot.users.Contact.Contact method), 38
translate() (kudubot.services.MultiLanguageService.MultiLanguageService method), 29

U

UnitTests (class in kudubot.tests.config.test_GlobalConfigHandler), 30
UnitTests (class in kudubot.tests.connections.test_Connection), 31
UnitTests (class in kudubot.tests.entities.test_Message), 32
UnitTests (class in kudubot.tests.services.test_Service), 34

UnitTests (class in kudubot.tests.users.test_AddressBook), 35
UnitTests (class in kudubot.tests.users.test_Contact), 35

V

validate_config_directory() (kudubot.config.GlobalConfigHandler.GlobalConfigHandler method), 3
validate_config_directory() (kudubot.tests.config.test_GlobalConfigHandler.UnitTests method), 31

W

WhatsappConnection (class in kudubot.connections.whatsapp.WhatsappConnection), 11
write_standard_connection_config() (kudubot.config.StandardConfigWriter.StandardConfigWriter method), 4
write_standard_service_config() (kudubot.config.StandardConfigWriter.StandardConfigWriter method), 4