

---

**kudubot**  
*Release 0.31.0*

**Hermann Krumrey**

**Sep 13, 2019**



# CONTENTS

<b>1</b>	<b>kudubot package</b>	<b>3</b>
1.1	Subpackages . . . . .	3
1.2	Submodules . . . . .	5
1.3	kudubot.Bot module . . . . .	5
1.4	kudubot.exceptions module . . . . .	7
1.5	kudubot.helper module . . . . .	8
1.6	Module contents . . . . .	8
<b>2</b>	<b>kudubot</b>	<b>9</b>
<b>3</b>	<b>Indices and tables</b>	<b>11</b>
	<b>Python Module Index</b>	<b>13</b>
	<b>Index</b>	<b>15</b>



Contents:



## KUDUBOT PACKAGE

### 1.1 Subpackages

#### 1.1.1 kudubot.db package

##### Subpackages

##### kudubot.db.config package

##### Subpackages

##### kudubot.db.config.impl package

##### Submodules

##### kudubot.db.config.impl.MySqlConfig module

**class** kudubot.db.config.impl.MySqlConfig.**MySqlConfig**(*address: str, port: str, name: str, user: str, password: str*)

Bases: *kudubot.db.config.DbConfig.DbConfig*

Database configuration for MySQL

**\_\_init\_\_**(*address: str, port: str, name: str, user: str, password: str*)

Initializes the database configuration :param address: The database address :param port: The database port :param name: The database name :param user: The database user :param password: The database password

**to\_uri**( ) → str

Turns the configuration into an URI that SQLAlchemy can use :return: The URI

##### kudubot.db.config.impl.SqliteConfig module

**class** kudubot.db.config.impl.SqliteConfig.**SqliteConfig**(*path: str*)

Bases: *kudubot.db.config.DbConfig.DbConfig*

Database configuration for SQLite

**\_\_init\_\_**(*path: str*)

Initializes the database configuration :param path: The path to the SQLite database file

`to_uri ()` → str

Turns the configuration into an URI that SQLAlchemy can use :return: The URI

## Module contents

### Submodules

#### **kudubot.db.config.DbConfig module**

**class** kudubot.db.config.DbConfig.**DbConfig**

Bases: object

Class that defines common methods of database configurations

`to_uri ()` → str

Turns the configuration into an URI that SQLAlchemy can use :return: The URI

## Module contents

### Submodules

#### **kudubot.db.Address module**

**class** kudubot.db.Address.**Address** (\*\*kwargs)

Bases: sqlalchemy.ext.declarative.api.Base

SQLAlchemy table that stores addresses

`__init__` (\*\*kwargs)

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

**address**

The address itself

**id**

The ID of the address

## Module contents

### 1.1.2 kudubot.parsing package

#### Submodules

#### **kudubot.parsing.Command module**

**class** kudubot.parsing.Command.**Command** (keyword: str, arg\_info: List[Tuple[str, Any]])

Bases: object

Class that defines a framework for bot commands

`__init__` (*keyword: str, arg\_info: List[Tuple[str, Any]]*)

Initializes a command :param keyword: The keyword/command argument :param arg\_info: A list of tuples modelling the command arguments.

The first element in the tuple is the name of the argument, the second one specifies the type

`resolve_args` (*args: List[str]*) → Dict[str, Any]

Converts a list of string arguments into a dictionary of arguments converted to their correct types and associated with the appropriate keys. :param args: The arguments to resolve :return: The resolved arguments

`validate` (*keyword: str, args: List[str]*) → bool

Method that evaluates if a given command argument and argument list match the command's parameters :param keyword: The command argument :param args: The rest of the arguments :return:

## kudubot.parsing.CommandParser module

`class` kudubot.parsing.CommandParser.CommandParser

Bases: object

A parser for bot commands

`classmethod` `commands` () → List[kudubot.parsing.Command.Command]

Defines the commands the parser supports :return: The list of commands

`classmethod` `help_text` (*include\_title: bool = False*) → str

Generates a help message for the parser :param include\_title: Whether or not to include a title :return: The help text for this parser

`classmethod` `name` () → str

**Returns** The name of the parser

`parse` (*text: str*) → Tuple[str, Dict[str, Any]]

Parses a command :param text: The text to parse :return: The (command, arguments)

## Module contents

### 1.1.3 kudubot.test package

## Module contents

## 1.2 Submodules

### 1.3 kudubot.Bot module

`class` kudubot.Bot.Bot (*connection: bokkichat.connection.Connection.Connection, location: str, db\_uri: Optional[str] = None*)

Bases: object

The Bot class offers an abstraction layer above bokkichat

`__init__` (*connection: bokkichat.connection.Connection.Connection, location: str, db\_uri: Optional[str] = None*)

Initializes the bot :param connection: The connection the bot should use :param location: The location of config and DB files :param db\_uri: Specifies a custom database URI

**auth\_required** () → Callable

This is a decorator that makes it possible to restrict a user's access to certain commands. To use this, simply decorate an **'on\_'**-method with this decorator. The method will then only be called if the `is_authorized()` method returns True :return: None

**bg\_iteration** (*iteration: int, db\_session: sqlalchemy.orm.session.Session*)

Executes a background iteration. By default this does nothing. This is supposed to be overridden by child classes to implement background functionality :param iteration: The iteration count. Useful for differentiating

between actions that have different repetition rates

**Parameters** **db\_session** – The database session to use

**Returns**

**property** **bg\_pause**

The pause between background iterations :return: The pause in seconds

**classmethod** **create\_config** (*connection\_cls: Type[bokkichat.connection.Connection.Connection], path: str*)

Creates a configuration directory for a bot :param connection\_cls: The connection class for which to generate a config

**Parameters** **path** – The path of the configuration directory

**Returns** None

**classmethod** **extra\_config\_args** () → List[str]

**Returns** A list of additional settings parameters required for this bot. Will be stored in a separate `extras.json` file

**init** ()

Additional init method that gets called at the end of `__init__` :return: None

**is\_authorized** (*address: kudubot.db.Address.Address, args: Dict[str, Any], db\_session: sqlalchemy.orm.session.Session*) → bool

Checks if a user is authorized :param address: The user to check :param args: possible command arguments :param db\_session: The database session to use :return: True if authorized, False otherwise

**classmethod** **load** (*connection\_cls: Type[bokkichat.connection.Connection.Connection], location: str*)

Generates a Bot from the location. :param connection\_cls: The class of the connection :param location: The location of the bot configuration directory :return: The generated bot

**classmethod** **name** () → str

**Returns** The name of the bot

**on\_command** (*parser: kudubot.parsing.CommandParser.CommandParser, command: str, args: Dict[str, Any], sender: kudubot.db.Address.Address, db\_session: sqlalchemy.orm.session.Session*)

Handles text messages that have been parsed as commands. Automatically searches for `'on_X'` methods in the bot and forwards the parameters to those methods if they exist. This mechanism can be used for simple bots that don't need more logic than a simple `if "command" elif "other_command"...` :param parser: The parser containing the command :param command: The command name :param args: The arguments of the command :param sender: The database address of the sender :param db\_session: A valid database session :return: None

**on\_media** (*message:* `bokkichat.entities.message.MediaMessage.MediaMessage`, *sender:* `kudubot.db.Address.Address`, *db\_session:* `sqlalchemy.orm.session.Session`)  
 Handles media messages. Those are by default simply ignored. :param message: The received message :param sender: The database Address object of the sender :param db\_session: A valid database session :return: None

**on\_msg** (*message:* `bokkichat.entities.message.Message.Message`)  
 The callback method is called for every received message. This method automatically delegates message handling to the `on_text` and `on_media` methods. This also creates a database session for those methods to use that will avoid threading issues. :param message: The received message :return: None

**on\_text** (*message:* `bokkichat.entities.message.TextMessage.TextMessage`, *sender:* `kudubot.db.Address.Address`, *db\_session:* `sqlalchemy.orm.session.Session`)  
 Handles text messages that aren't commands. Those are by default simply ignored. :param message: The received message :param sender: The database Address object of the sender :param db\_session: A valid database session :return: None

**parse** (*message:* `bokkichat.entities.message.Message.Message`) → Optional[Tuple[kudubot.parsing.CommandParser.CommandParser, str, Dict[str, Any]]]  
 Parses the received message to check which parser and command are applicable to the message. :param message: The message to parse :return: The resulting parser/command combination

**classmethod parsers** () → List[kudubot.parsing.CommandParser.CommandParser]  
**Returns** A list of parser the bot supports for commands

**pre\_callback** (*message:* `bokkichat.entities.message.Message.Message`) → bool  
 Prepares the callback and decides whether or not the callback is even executed :param message: The message to check :return: True if the execution continues, False otherwise

**run\_in\_bg** ()  
 Method that is started when the bot is started. This executes the `bg_iteration` method every `bg_pause` seconds :return: None

**save\_config** ()  
 Saves the configuration for this bot :return: None

**send\_txt** (*receiver:* `kudubot.db.Address.Address`, *body:* `str`, *title:* `Optional[str] = ""`)  
 Convenience function that allows easier sending of text messages :param receiver: The receiver of the text message :param body: The body of the text message :param title: The (optional) title of the message :return: None

**start** ()  
 Starts the bot using the implemented callback function :return: None

**classmethod unauthorized\_message** () → str  
**Returns** A custom message sent to a user if they tried to access a feature that requires authorization without being authorized

**classmethod version** () → str  
**Returns** The current version of the bot

## 1.4 kudubot.exceptions module

**exception** `kudubot.exceptions.ConfigurationError`

Bases: `Exception`

Exception that gets raised whenever a configuration error was found

**exception** kudubot.exceptions.**ParseError**

Bases: Exception

Exception that gets raised whenever a parsing error occurs

## 1.5 kudubot.helper module

kudubot.helper.**cli\_bot\_start** (*bot\_cls: Type[kudubot.Bot.Bot], connection\_cls: Type[bokkichat.connection.Connection.Connection], sentry\_dsn: Optional[str] = None*)

Implements a standard CLI interface for kudubot implementations :param bot\_cls: The class of the bot to start :param connection\_cls: The connection to use with the bot :param sentry\_dsn: Optional sentry DSN for exception logging :return: None

## 1.6 Module contents

kudubot.**sentry\_dsn** = 'https://cc8787586ec047858ecff38befd67018@sentry.namibsun.net/4'

The sentry DSN for this project

kudubot.**version** = '0.31.0'

The current version of the package

---

CHAPTER  
TWO

---

**KUDUBOT**



## INDICES AND TABLES

- genindex
- modindex
- search



## PYTHON MODULE INDEX

### k

- kudubot, 8
- kudubot.Bot, 5
- kudubot.db, 4
- kudubot.db.Address, 4
- kudubot.db.config, 4
- kudubot.db.config.DbConfig, 4
- kudubot.db.config.impl, 4
- kudubot.db.config.impl.MySqlConfig, 3
- kudubot.db.config.impl.SqliteConfig, 3
- kudubot.exceptions, 7
- kudubot.helper, 8
- kudubot.parsing, 5
- kudubot.parsing.Command, 4
- kudubot.parsing.CommandParser, 5
- kudubot.test, 5



## Symbols

\_\_init\_\_() (kudubot.Bot.Bot method), 5  
 \_\_init\_\_() (kudubot.db.Address.Address method), 4  
 \_\_init\_\_() (kudubot.db.config.impl.MySqlConfig.MySqlConfig method), 3  
 \_\_init\_\_() (kudubot.db.config.impl.SqliteConfig.SqliteConfig method), 3  
 \_\_init\_\_() (kudubot.parsing.Command.Command method), 4

## A

Address (class in kudubot.db.Address), 4  
 address (kudubot.db.Address.Address attribute), 4  
 auth\_required() (kudubot.Bot.Bot method), 5

## B

bg\_iteration() (kudubot.Bot.Bot method), 6  
 bg\_pause() (kudubot.Bot.Bot property), 6  
 Bot (class in kudubot.Bot), 5

## C

cli\_bot\_start() (in module kudubot.helper), 8  
 Command (class in kudubot.parsing.Command), 4  
 CommandParser (class in kudubot.parsing.CommandParser), 5  
 commands() (kudubot.parsing.CommandParser.CommandParser class method), 5  
 ConfigurationError, 7  
 create\_config() (kudubot.Bot.Bot class method), 6

## D

DbConfig (class in kudubot.db.config.DbConfig), 4

## E

extra\_config\_args() (kudubot.Bot.Bot class method), 6

## H

help\_text() (kudubot.parsing.CommandParser.CommandParser class method), 5

## I

id (kudubot.db.Address.Address attribute), 4  
 init() (kudubot.Bot.Bot method), 6  
 is\_authorized() (kudubot.Bot.Bot method), 6

## K

kudubot (module), 8  
 kudubot.Bot (module), 5  
 kudubot.db (module), 4  
 kudubot.db.Address (module), 4  
 kudubot.db.config (module), 4  
 kudubot.db.config.DbConfig (module), 4  
 kudubot.db.config.impl (module), 4  
 kudubot.db.config.impl.MySqlConfig (module), 3  
 kudubot.db.config.impl.SqliteConfig (module), 3  
 kudubot.exceptions (module), 7  
 kudubot.helper (module), 8  
 kudubot.parsing (module), 5  
 kudubot.parsing.Command (module), 4  
 kudubot.parsing.CommandParser (module), 5  
 kudubot.test (module), 5

## L

load() (kudubot.Bot.Bot class method), 6

## M

MySqlConfig (class in kudubot.db.config.impl.MySqlConfig), 3

## N

name() (kudubot.Bot.Bot class method), 6  
 name() (kudubot.parsing.CommandParser.CommandParser class method), 5

## O

on\_command() (kudubot.Bot.Bot method), 6  
 on\_media() (kudubot.Bot.Bot method), 6  
 on\_msg() (kudubot.Bot.Bot method), 7  
 on\_text() (kudubot.Bot.Bot method), 7

## P

`parse()` (*kudubot.Bot.Bot method*), 7  
`parse()` (*kudubot.parsing.CommandParser.CommandParser method*), 5  
`ParseError`, 7  
`parsers()` (*kudubot.Bot.Bot class method*), 7  
`pre_callback()` (*kudubot.Bot.Bot method*), 7

## R

`resolve_args()` (*kudubot.parsing.Command.Command method*), 5  
`run_in_bg()` (*kudubot.Bot.Bot method*), 7

## S

`save_config()` (*kudubot.Bot.Bot method*), 7  
`send_txt()` (*kudubot.Bot.Bot method*), 7  
`sentry_dsn` (*in module kudubot*), 8  
`SqliteConfig` (*class* *in kudubot.db.config.impl.SqliteConfig*), 3  
`start()` (*kudubot.Bot.Bot method*), 7

## T

`to_uri()` (*kudubot.db.config.DbConfig.DbConfig method*), 4  
`to_uri()` (*kudubot.db.config.impl.MySqlConfig.MySqlConfig method*), 3  
`to_uri()` (*kudubot.db.config.impl.SqliteConfig.SqliteConfig method*), 3

## U

`unauthorized_message()` (*kudubot.Bot.Bot class method*), 7

## V

`validate()` (*kudubot.parsing.Command.Command method*), 5  
`version` (*in module kudubot*), 8  
`version()` (*kudubot.Bot.Bot class method*), 7