
bokkichat

Release 0.4.4

Hermann Krumrey

Oct 07, 2019

CONTENTS

1 bokkichat package	3
1.1 Subpackages	3
1.2 Submodules	9
1.3 bokkichat.exceptions module	9
1.4 Module contents	9
2 bokkichat	11
3 Indices and tables	13
Python Module Index	15
Index	17

Contents:

BOKKICHAT PACKAGE

1.1 Subpackages

1.1.1 bokkichat.connection package

Subpackages

bokkichat.connection.impl package

Submodules

bokkichat.connection.impl.CliConnection module

class `bokkichat.connection.impl.CliConnection.CliConnection` (*settings:* `bokkichat.settings.Settings.Settings`)

Bases: `bokkichat.connection.Connection.Connection`

Class that implements a CLI connection which can be used in testing

property address

A CLI connection has no real entities, so a dummy entities is generated. :return: The entities of the connection

close ()

Disconnects the Connection. :return: None

classmethod name () → str

The name of the connection class :return: The connection class name

receive () → List[bokkichat.entities.message.Message.Message]

A CLI Connection receives messages by listening to the input :return: A list of pending Message objects

send (message: bokkichat.entities.message.Message.Message)

Prints a “sent” message :param message: The message to “send” :return: None

classmethod settings_cls () → Type[bokkichat.settings.impl.CliSettings.CliSettings]

The settings class used by this connection :return: The settings class

bokkichat.connection.impl.TelegramBotConnection module

class bokkichat.connection.impl.TelegramBotConnection.**TelegramBotConnection** (*settings:*
bok-
kichat.settings.impl.T

Bases: *bokkichat.connection.Connection.Connection*

Class that implements a Telegram bot connection

__init__ (*settings: bokkichat.settings.impl.TelegramBotSettings.TelegramBotSettings*)

Initializes the connection, with credentials provided by a Settings object. :param settings: The settings for the connection

property address

A connection must be able to specify its own entities :return: The entities of the connection

close ()

Disconnects the Connection. :return: None

loop (*callback: Callable, sleep_time: int = 1*)

Starts a loop that periodically checks for new messages, calling a provided callback function in the process. :param callback: The callback function to call for each

received message. The callback should have the following format:

lambda connection, message: do_stuff()

Parameters **sleep_time** – The time to sleep between loops

Returns None

classmethod name () → str

The name of the connection class :return: The connection class name

receive () → List[bokkichat.entities.message.Message.Message]

Receives all pending messages. :return: A list of pending Message objects

send (*message: bokkichat.entities.message.Message.Message*)

Sends a message. A message may be either a TextMessage or a MediaMessage. :param message: The message to send :return: None

classmethod settings_cls () → Type[bokkichat.settings.impl.TelegramBotSettings.TelegramBotSettings]

The settings class used by this connection :return: The settings class

Module contents

Submodules

bokkichat.connection.Connection module

class bokkichat.connection.Connection.**Connection** (*settings:*
bok-
kichat.settings.Settings.Settings)

Bases: object

Class that defines methods a Connection must implement. A connection is the central class in bokkichat and handles all communications with the chat services.

`__init__` (*settings: bokkichat.settings.Settings.Settings*)

Initializes the connection, with credentials provided by a Settings object. :param settings: The settings for the connection

property address

A connection must be able to specify its own entities :return: The entities of the connection

`close` ()

Disconnects the Connection. :return: None

classmethod from_serialized_settings (*serialized: str*) → bokkichat.connection.Connection.Connection

Generates a Connection using serialized settings :param serialized: The serialized settings :return: The generated connection

loop (*callback: Callable, sleep_time: int = 1*)

Starts a loop that periodically checks for new messages, calling a provided callback function in the process. :param callback: The callback function to call for each

received message. The callback should have the following format:

```
lambda connection, message: do_stuff()
```

Parameters `sleep_time` – The time to sleep between loops

Returns None

classmethod name () → str

The name of the connection class :return: The connection class name

receive () → List[bokkichat.entities.message.Message.Message]

Receives all pending messages. :return: A list of pending Message objects

send (*message: bokkichat.entities.message.Message.Message*)

Sends a message. A message may be either a TextMessage or a MediaMessage. :param message: The message to send :return: None

classmethod settings_cls () → Type[bokkichat.settings.Settings.Settings]

The settings class used by this connection :return: The settings class

Module contents

1.1.2 bokkichat.entities package

Subpackages

bokkichat.entities.message package

Submodules

bokkichat.entities.message.MediaMessage module

```
class bokkichat.entities.message.MediaMessage.MediaMessage (sender:      bok-
                                                         kichat.entities.Address.Address,
                                                         receiver:    bok-
                                                         kichat.entities.Address.Address,
                                                         media_type:  bok-
                                                         kichat.entities.message.MediaType.MediaType,
                                                         data: bytes, caption:
                                                         Optional[str] = ")
```

Bases: *bokkichat.entities.message.Message.Message*

Class that defines an interface for media messages. Each media message has a media type, data and caption.

```
__init__ (sender: bokkichat.entities.Address.Address, receiver: bokkichat.entities.Address.Address,
          media_type: bokkichat.entities.message.MediaType.MediaType, data: bytes, caption: Op-
          tional[str] = ")
```

Initializes the TextMessage object :param sender: The sender of the message :param receiver: The receiver of the message :param media_type: The type of the contained media :param data: The data of the attached media :param caption: The caption attached to the media

```
static is_media () → bool
```

Returns Whether or not the message is a media message

```
make_reply (media_type: Optional[bokkichat.entities.message.MediaType.MediaType] = None,
            data: Optional[bytes] = None, caption: Optional[str] = None) → bok-
            kichat.entities.message.Message.Message
```

Swaps the sender and receiver of the message :param media_type: The type of the contained media :param data: The data of the attached media :param caption: The caption attached to the media :return: The generated reply

bokkichat.entities.message.MediaType module

```
class bokkichat.entities.message.MediaType.MediaType
```

Bases: *enum.Enum*

Enum that specifies the various types of media that can be sent using MediaMessage objects.

```
AUDIO = 2
```

```
IMAGE = 3
```

```
VIDEO = 1
```

bokkichat.entities.message.Message module

```
class bokkichat.entities.message.Message.Message (sender:      bok-
                                                         kichat.entities.Address.Address,
                                                         receiver:    bok-
                                                         kichat.entities.Address.Address)
```

Bases: *object*

Class that defines common attributes for a Message object.

```
__init__ (sender: bokkichat.entities.Address.Address, receiver: bokkichat.entities.Address.Address)
```

Initializes a Message object. :param sender: The sender of the message :param receiver: The receiver of the message

static is_media () → bool

Returns Whether or not the message is a media message

static is_text () → bool

Returns Whether or not the message is a text message

make_reply () → bokkichat.entities.message.Message.Message

Swaps the sender and receiver of the message :return: The generated reply

bokkichat.entities.message.TextMessage module

```
class bokkichat.entities.message.TextMessage.TextMessage (sender:          bok-
                                                         kichat.entities.Address.Address,
                                                         receiver:          bok-
                                                         kichat.entities.Address.Address,
                                                         body: str, title:  Op-
                                                         tional[str] = ")
Bases: bokkichat.entities.message.Message.Message
```

Class that defines an interface for text messages. Each text message has a title and a body. Some chat services don't allow titles for messages, in those cases, the title will be blank.

```
__init__ (sender: bokkichat.entities.Address.Address, receiver: bokkichat.entities.Address.Address,
          body: str, title: Optional[str] = ")
Initializes the TextMessage object :param sender: The sender of the message :param receiver: The receiver
of the message :param body: The message body :param title: The title of the message. Defaults to an empty
string
```

static is_text () → bool

Returns Whether or not the message is a text message

```
make_reply (body: Optional[str] = None, title: Optional[str] = None) → bok-
                                                         kichat.entities.message.Message.Message
```

Swaps the sender and receiver of the message :return: The generated reply

```
split (max_chars: int) → List[str]
```

Splits the body text into multiple chunks below a certain size. Will try to not break up any lines :param max_chars: The chunk size :return: The parts of the message

Module contents

Submodules

bokkichat.entities.Address module

```
class bokkichat.entities.Address.Address (address: str)
```

Bases: object

Class that models an Address.

```
__init__ (address: str)
```

Initializes the entities object :param address: The actual entities to which messages can be sent

Module contents

1.1.3 bokkichat.settings package

Subpackages

bokkichat.settings.impl package

Submodules

bokkichat.settings.impl.CliSettings module

class bokkichat.settings.impl.CliSettings.**CliSettings**

Bases: *bokkichat.settings.Settings.Settings*

Class that defines a Settings object for a CLI connection

classmethod **deserialize** (*_*: *str*) → bokkichat.settings.impl.CliSettings.CliSettings

Deserializes a string and generates a Settings object from it :param *_*: The serialized string :return: The deserialized Settings object

classmethod **prompt** () → bokkichat.settings.Settings.Settings

Prompts the user for input to generate a Settings object :return: The generated settings object

serialize () → *str*

Serializes the settings to a string :return: The serialized Settings object

bokkichat.settings.impl.TelegramBotSettings module

class bokkichat.settings.impl.TelegramBotSettings.**TelegramBotSettings** (*api_key*:
str)

Bases: *bokkichat.settings.Settings.Settings*

Class that defines a Settings object for a Telegram bot connection

__init__ (*api_key*: *str*)

Initializes the Telegram Connection. :param *api_key*: The API key used for authentication

classmethod **deserialize** (*serialized*: *str*) → bokkichat.settings.impl.TelegramBotSettings.TelegramBotSettings

Deserializes a string and generates a Settings object from it :param *serialized*: The serialized string :return: The deserialized Settings object

classmethod **prompt** () → bokkichat.settings.Settings.Settings

Prompts the user for input to generate a Settings object :return: The generated settings object

serialize () → *str*

Serializes the settings to a string :return: The serialized Settings object

Module contents

Submodules

bokkichat.settings.Settings module

class bokkichat.settings.Settings.Settings

Bases: object

Class that defines what methods a Settings class must implement. Settings objects are used to initialize a Connection.

classmethod **deserialize** (*serialized: str*) → bokkichat.settings.Settings.Settings

Deserializes a string and generates a Settings object from it :param serialized: The serialized string :return: The deserialized Settings object

classmethod **prompt** () → bokkichat.settings.Settings.Settings

Prompts the user for input to generate a Settings object :return: The generated settings object

serialize () → str

Serializes the settings to a string :return: The serialized Settings object

static **user_input** (*prompt: str*) → str

Prompts the user for input :param prompt: The text to display with the prompt :return: The resulting response

Module contents

1.1.4 bokkichat.test package

Module contents

1.2 Submodules

1.3 bokkichat.exceptions module

exception bokkichat.exceptions.InvalidMessageData (*message_data: Dict[str, Any]*)

Bases: Exception

Exception that indicates message data that's invalid or otherwise could not be correctly parsed.

__init__ (*message_data: Dict[str, Any]*)

Initializes the Exception :param message_data: The message data that caused the exception to be raised

exception bokkichat.exceptions.InvalidSettings

Bases: Exception

Exception that gets raised whenever there's a problem with the settings. Example: Invalid API key, connection can't be established

1.4 Module contents

BOKKICHAT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

b

- bokkichat, 9
- bokkichat.connection, 5
- bokkichat.connection.Connection, 4
- bokkichat.connection.impl, 4
- bokkichat.connection.impl.CliConnection,
3
- bokkichat.connection.impl.TelegramBotConnection,
4
- bokkichat.entities, 8
- bokkichat.entities.Address, 7
- bokkichat.entities.message, 7
- bokkichat.entities.message.MediaMessage,
6
- bokkichat.entities.message.MediaType, 6
- bokkichat.entities.message.Message, 6
- bokkichat.entities.message.TextMessage,
7
- bokkichat.exceptions, 9
- bokkichat.settings, 9
- bokkichat.settings.impl, 8
- bokkichat.settings.impl.CliSettings, 8
- bokkichat.settings.impl.TelegramBotSettings,
8
- bokkichat.settings.Settings, 9
- bokkichat.test, 9

Symbols

__init__ () (*bokkichat.connection.Connection.Connection* method), 4
__init__ () (*bokkichat.connection.impl.TelegramBotConnection.TelegramBotConnection* method), 4
__init__ () (*bokkichat.entities.Address.Address* method), 7
__init__ () (*bokkichat.entities.message.MediaMessage.MediaMessage* method), 6
__init__ () (*bokkichat.entities.message.Message.Message* method), 6
__init__ () (*bokkichat.entities.message.TextMessage.TextMessage* method), 7
__init__ () (*bokkichat.exceptions.InvalidMessageData* method), 9
__init__ () (*bokkichat.settings.impl.TelegramBotSettings.TelegramBotSettings* method), 8

bokkichat.entities.message.MediaMessage (module), 6
bokkichat.entities.message.MediaType (module), 6
bokkichat.entities.message.Message (module), 6
bokkichat.entities.message.TextMessage (module), 7
bokkichat.exceptions (module), 9
bokkichat.settings (module), 9
bokkichat.settings.impl (module), 8
bokkichat.settings.impl.CliSettings (module), 8
bokkichat.settings.impl.TelegramBotSettings (module), 8
bokkichat.settings.Settings (module), 9
bokkichat.test (module), 9

A

Address (class in *bokkichat.entities.Address*), 7
 address () (*bokkichat.connection.Connection.Connection* property), 5
 address () (*bokkichat.connection.impl.CliConnection.CliConnection* property), 3
 address () (*bokkichat.connection.impl.TelegramBotConnection.TelegramBotConnection* property), 4
 AUDIO (*bokkichat.entities.message.MediaType.MediaType* attribute), 6

B

bokkichat (module), 9
 bokkichat.connection (module), 5
 bokkichat.connection.Connection (module), 4
 bokkichat.connection.impl (module), 4
 bokkichat.connection.impl.CliConnection (module), 3
 bokkichat.connection.impl.TelegramBotConnection (module), 4
 bokkichat.entities (module), 8
 bokkichat.entities.Address (module), 7
 bokkichat.entities.message (module), 7

C

CliConnection (class in *bokkichat.connection.impl.CliConnection*), 3
 CliSettings (class in *bokkichat.settings.impl.CliSettings*), 8
 close () (*bokkichat.connection.Connection.Connection* method), 5
 close () (*bokkichat.connection.impl.CliConnection.CliConnection* method), 3
 close () (*bokkichat.connection.impl.TelegramBotConnection.TelegramBotConnection* method), 4
 Connection (class in *bokkichat.connection.Connection*), 4

D

deserialize () (*bokkichat.settings.impl.CliSettings.CliSettings* class method), 8
 deserialize () (*bokkichat.settings.impl.TelegramBotSettings.TelegramBotSettings* class method), 8
 deserialize () (*bokkichat.settings.Settings.Settings* class method), 9

F

from_serialized_settings() (bokkichat.connection.Connection.Connection class method), 5

I

IMAGE (bokkichat.entities.message.MediaType.MediaType attribute), 6

InvalidMessageData, 9

InvalidSettings, 9

is_media() (bokkichat.entities.message.MediaMessage.MediaMessage static method), 6

is_media() (bokkichat.entities.message.Message.Message static method), 6

is_text() (bokkichat.entities.message.Message.Message static method), 7

is_text() (bokkichat.entities.message.TextMessage.TextMessage static method), 7

L

loop() (bokkichat.connection.Connection.Connection method), 5

loop() (bokkichat.connection.impl.TelegramBotConnection.TelegramBotConnection method), 4

M

make_reply() (bokkichat.entities.message.MediaMessage.MediaMessage method), 6

make_reply() (bokkichat.entities.message.Message.Message method), 7

make_reply() (bokkichat.entities.message.TextMessage.TextMessage method), 7

MediaMessage (class in bokkichat.entities.message.MediaMessage), 6

MediaType (class in bokkichat.entities.message.MediaType), 6

Message (class in bokkichat.entities.message.Message), 6

N

name() (bokkichat.connection.Connection.Connection class method), 5

name() (bokkichat.connection.impl.CliConnection.CliConnection class method), 3

name() (bokkichat.connection.impl.TelegramBotConnection.TelegramBotConnection class method), 4

P

prompt() (bokkichat.settings.impl.CliSettings.CliSettings class method), 8

prompt() (bokkichat.settings.impl.TelegramBotSettings.TelegramBotSettings class method), 8

prompt() (bokkichat.settings.Settings.Settings class method), 9

R

receive() (bokkichat.connection.Connection.Connection method), 5

receive() (bokkichat.connection.impl.CliConnection.CliConnection method), 3

receive() (bokkichat.connection.impl.TelegramBotConnection.TelegramBotConnection method), 4

S

send() (bokkichat.connection.Connection.Connection method), 5

send() (bokkichat.connection.impl.CliConnection.CliConnection method), 3

send() (bokkichat.connection.impl.TelegramBotConnection.TelegramBotConnection method), 4

serialize() (bokkichat.settings.impl.CliSettings.CliSettings method), 8

serialize() (bokkichat.settings.impl.TelegramBotSettings.TelegramBotSettings method), 8

serialize() (bokkichat.settings.Settings.Settings method), 9

Settings (class in bokkichat.settings.Settings), 9

settings_cls() (bokkichat.connection.Connection.Connection class method), 5

settings_cls() (bokkichat.connection.impl.CliConnection.CliConnection class method), 3

settings_cls() (bokkichat.connection.impl.TelegramBotConnection.TelegramBotConnection class method), 4

split() (bokkichat.entities.message.TextMessage.TextMessage method), 7

T

TelegramBotConnection (class in bokkichat.connection.impl.TelegramBotConnection), 4

TelegramBotSettings (class in bokkichat.settings.impl.TelegramBotSettings), 8

TextMessage (class in bokkichat.entities.message.TextMessage), 7

U

user_input() (bokkichat.settings.Settings.Settings static method), 9

V

VIDEO (*bokkachat.entities.message.MediaType.MediaType*
attribute), 6