gramhopper Documentation

Release 1.0.7

Or Bin, Meir Halachmi

Feb 28, 2022

Content

CHAPTER 1

Triggers

1.1 Event Streak Trigger

1.1.1 event_streak

class EventStreakTrigger(*streak_timeout_sec: float*, *event_count: int*, *counting_event_trigger: gramhopper.triggers.basic_triggers.BaseTrigger*, *resetting_event_trigger: Optional[gramhopper.triggers.basic_triggers.BaseTrigger] = None*)

Event-streak trigger. This is used to trigger a rule in a case of a streak of events, for example when a certain phrase is written a few times in a row.

This trigger gets a defined trigger as a "counting" event and optionally another one as a "resetting" event (to stop counting). It also gets the count of events to be considered a streak, and the timeout of the streak (the allowed time period between the first event and the last event in the streak.

__init__(*streak_timeout_sec: float*, *event_count: int*, *counting_event_trigger: gramhopper.triggers.basic_triggers.BaseTrigger*, *resetting_event_trigger: Optional[gramhopper.triggers.basic_triggers.BaseTrigger] = None*)

Constructs the trigger.

Parameters

- **streak_timeout_sec** – The allowed time period between the first event and the last event in the streak
- **event_count** – The count of events to be considered a streak
- **counting_event_trigger** – A trigger to identify an event that counts towards the streak
- **resetting_event_trigger** – A trigger to identify an event that resets the streak

1.2 Filter Triggers

class FilterTriggers
Text-based triggers.

chat = <class 'gramhopper.triggers.filter_triggers._ChatFilterBasedTrigger'>
A user filter trigger. See more in *_ChatFilterBasedTrigger*.

language = <class 'gramhopper.triggers.filter_triggers._LanguageFilterBasedTrigger'>
A user filter trigger. See more in *_LanguageFilterBasedTrigger*.

message_type = <class 'gramhopper.triggers.filter_triggers._MessageTypeFilterBasedTrig
A user filter trigger. See more in *_MessageTypeFilterBasedTrigger*.

user = <class 'gramhopper.triggers.filter_triggers._UserFilterBasedTrigger'>
A user filter trigger. See more in *_UserFilterBasedTrigger*.

1.2.1 filter.user

class _UserFilterBasedTrigger(*nickname: str = None*, *user_id: int = None*, *username: str = None*)
A user filter trigger. This is used to trigger a rule when an incoming message comes from a specific user.

__init__(*nickname: str = None*, *user_id: int = None*, *username: str = None*)
Constructs the trigger. *nickname* can be used if such a nickname is defined in users.json file. Otherwise, one of *user_id* and *username* should be specified.

Parameters

- **message_filter** – The filter to test if the message passes through
- **nickname** – The nickname of the user to pass messages from.
- **user_id** – The Telegram user ID of the user to pass messages from.
- **username** – The Telegram username of the user to pass messages from.

1.2.2 filter.chat

class _ChatFilterBasedTrigger(*chat_id: int = None*, *username: str = None*)
A chat filter trigger. This is used to trigger a rule when an incoming message is in a specific chat.

__init__(*chat_id: int = None*, *username: str = None*)
Constructs the trigger. One and only one of *chat_id* and *username* should be specified.

Parameters

- **chat_id** – The Telegram chat ID of the chat to pass messages from.
- **username** – The Telegram username whose chat to pass messages from.

1.2.3 filter.language

class _LanguageFilterBasedTrigger(*lang: Union[str, List[str]]*)
A language filter trigger. This is used to trigger a rule when an incoming message is in a specific language.

__init__(*lang: Union[str, List[str]]*)
Constructs the trigger.

Parameters **lang** – The language code/s in which to pass messages.

1.2.4 filter.message_type

class _MessageTypeFilterBasedTrigger(*message_type*)
A message type filter trigger. This is used to trigger a rule when an incoming message is of a specific type.

__init__(*message_type*)
Constructs the trigger.

Parameters **message_type** – The message type of which to filter to pass messages, for example: 'photo', 'status_update.left_chat_member' or 'document'. See more in telegram.ext.filters.Filters.

1.3 Text Triggers

class TextTriggers
Text-based triggers.

has_exact_word = <class 'gramhopper.triggers.text_triggers._HasExactWordTrigger'>
A word trigger. See more in *_HasExactWordTrigger*.

has_substring = <class 'gramhopper.triggers.text_triggers._HasSubstringTrigger'>
A substring trigger. See more in *_HasSubstringTrigger*.

regexp = <class 'gramhopper.triggers.text_triggers._RegExpTrigger'>
A regexp-based trigger. See more in *_RegExpTrigger*.

1.3.1 text.has_exact_word

class _HasExactWordTrigger(*word: Union[str, List[str]]*)
Word trigger - the same as substring trigger, but for exact words search. This is used to trigger a rule when a certain word (or one of a list of words) exists in an incoming message.

__init__(*word: Union[str, List[str]]*)
Constructs the trigger.

Parameters **word** – The word/s to search in the message

1.3.2 text.has_substring

class _HasSubstringTrigger(*substring: Union[str, List[str]], exact: bool = False*)
Substring trigger. This is used to trigger a rule when a certain substring (or one of a list of substrings) exists in an incoming message.

__init__(*substring: Union[str, List[str]], exact: bool = False*)
Constructs the trigger.

Parameters

- **substring** – The substring/s to search in the message
- **exact** – Whether the exact substring should appear (as a whole word) in the message

1.3.3 text.regexp

class _RegExpTrigger(*pattern: str*)

Regular-expression-based trigger. This is used to trigger a rule when an incoming message matches the given pattern.

__init__(*pattern: str*)

Constructs the trigger.

Parameters **pattern** – The pattern to test the message's match with

CHAPTER 2

Responses

2.1 Match Responses

There are some matched-based responses, which should normally come after a regexp trigger

class MatchResponses
Regexp-based responses. These responses use the regexp match result from the trigger, as well as the given template, to build the response text.

message = <class 'gramhopper.responses.match_responses._MatchMessageResponse'>
A regexp-based **message** response. See more in *_MatchMessageResponse*.

reply = <class 'gramhopper.responses.match_responses._MatchReplyResponse'>
A regexp-based **reply** response. See more in *_MatchReplyResponse*.

2.1.1 match.message

class _MatchMessageResponse(*template: str*, *parse_mode: Optional[str] = None*)
A regexp-based response in which the response method is a normal message

__init__(*template: str*, *parse_mode: Optional[str] = None*)
Constructs the response.

Parameters

- **template** – The template to use when building the response text
- **parse_mode** – Optional parse mode for the message. Read more in `telegram.ParseMode`.

2.1.2 match.reply

class _MatchReplyResponse(*template: str*, *parse_mode: Optional[str] = None*)
A regexp-based response in which the response method is a reply to the triggering message

__init__ (*template: str*, *parse_mode: Optional[str] = None*)
Constructs the response.

Parameters

- **template** – The template to use when building the response text
- **parse_mode** – Optional parse mode for the message. Read more in telegram.ParseMode.

2.2 Preset Responses

class PresetResponses
Preset responses. These responses use a preset response/s to respond with. If a list of responses is given, one of them will be chosen randomly for each response.

document = <class 'gramhopper.responses.preset_responses._PresetDocumentResponse'>
A preset **document** response. See more in *_PresetDocumentResponse*.

message = <class 'gramhopper.responses.preset_responses._PresetMessageResponse'>
A preset **message** response. See more in *_PresetMessageResponse*.

reply = <class 'gramhopper.responses.preset_responses._PresetReplyResponse'>
A preset **reply** response. See more in *_PresetReplyResponse*.

2.2.1 preset.document

class _PresetDocumentResponse (*preset_response: Union[str, telegram.files.document.Document], parse_mode: Optional[str] = None*)
A preset response in which the response method is a document

__init__ (*preset_response: Union[str, telegram.files.document.Document], parse_mode: Optional[str] = None*)
Constructs the response.

Parameters

- **preset_response** – The preset document URL or document object
- **parse_mode** – Optional parse mode for the message. Read more in telegram.ParseMode.

2.2.2 preset.message

class _PresetMessageResponse (*preset_response: Union[str, List[str]], parse_mode: Optional[str] = None*)
A preset response in which the response method is a normal message

__init__ (*preset_response: Union[str, List[str]], parse_mode: Optional[str] = None*)
Constructs the response.

Parameters

- **preset_response** – The preset response or list of responses
- **parse_mode** – Optional parse mode for the message. Read more in telegram.ParseMode.

2.2.3 preset.reply

class _PresetReplyResponse(*preset_response: Union[str, List[str]], parse_mode: Optional[str] = None*)

A preset response in which the response method is a reply to the triggering message

__init__(*preset_response: Union[str, List[str]], parse_mode: Optional[str] = None*)

Constructs the response.

Parameters

- **preset_response** – The preset response or list of responses
- **parse_mode** – Optional parse mode for the message. Read more in telegram.ParseMode.

Index

Symbols

C

D

E

F

H

L

M

P

R

T

U