python-slacktools Documentation

Austin Pray

Sep 14, 2018

Contents

CHAPTER 1

API Reference

1.1 Authorization

exception slacktools.authorization.**SignatureVersionException**

slacktools.authorization.**verify_signature**(*signing_secret: str*, *request_timestamp: int*, *body: str*, *signature: str*, *current_timestamp: int = None*) → bool

Verifies a signature from X-Slack-Signature and X-Slack-Request-Timestamp

see: https://api.slack.com/docs/verifying-requests-from-slack

also see the implementation in: slackapi/python-slack-events-api. Check out *SlackServer.verify_signature*. History: 2018-09-09

Raises SignatureVersionException if the signature version is something other than v0

1.2 Working With Messages

slacktools.message.**extract_mentions**(*text: str*) → List[str]

Returns a list of slack user ID strings in the order that they appear in the string

```
>>> extract_mentions('waddup <@U5H9UR207> testing <@U8DCV8P6X>')
['U5H9UR207', 'U8DCV8P6X']
>>> extract_mentions('hey <@U5GJR5GF7> test this yooo <@U5GJR5GF7>')
['U5GJR5GF7', 'U5GJR5GF7']
>>> extract_mentions('<@U5GJR5GF7><@U5H9UR207><@U5GJR5GF7>')
['U5GJR5GF7', 'U5H9UR207', 'U5GJR5GF7']
```

slacktools.message.**extract_unique_mentions**(*text: str*) → Set[str]

Returns a set of slack user ID strings

```
>>> extract_unique_mentions('hey <@U5GJR5GF7> test this yooo <@U5GJR5GF7>')
{'U5GJR5GF7'}
>>> extract_unique_mentions('waddup <@U5H9UR207> testing <@U8DCV8P6X>') - {
↪'U8DCV8P6X', 'U5H9UR207'}
set()
```

slacktools.message.**extract_user_id_from_mention**(*m: str*) → Optional[str]

Given a slack mention control sequence extract the

```
>>> extract_user_id_from_mention('<@U5H9UR207>')
'U5H9UR207'
>>> extract_user_id_from_mention('ayy') is None
True
```

slacktools.message.**format_channel_link**(*name: str*, *channel_id: str*)

Formats a channel name and ID as a channel link using slack control sequences https://api.slack.com/docs/message-formatting#linking_to_channels_and_users

```
>>> format_channel_link('general', 'C024BE7LR')
'<#C024BE7LR|general>'
```

slacktools.message.**format_slack_mention**(*slack_id: str*)

Formats a slack user ID as a mention using slack control sequences https://api.slack.com/docs/message-formatting#linking_to_channels_and_users

```
>>> format_slack_mention('U5H9UR207')
'<@U5H9UR207>'
```

slacktools.message.**format_url**(*text*, *url*)

Formats an inline slack hyperlink to a URL https://api.slack.com/docs/message-formatting#linking_to_urls

```
>>> format_url('my website', 'https://austinpray.com')
'<my website|https://austinpray.com>'
```

slacktools.message.**is_user_mention**(*s: str*) → bool

tests if a string is a user mention control sequence

```
>>> is_user_mention('<@UXXXXXXXX>')
True
>>> is_user_mention('<@UXX>')
False
```

slacktools.message.**slack_escape**(*text: str*) → str

Escape slack control sequences in a string https://api.slack.com/docs/message-formatting#how_to_escape_characters

```
>>> slack_escape('Hello & <world> ')
'Hello &amp; &lt;world&gt; '
```

1.3 Sending Messages

slacktools.chat.**reply**(*slack_client*, *message: dict*, *text: str*)

Takes a message from some channel and chat.postMessage some text back with a user mention prepended

slacktools.chat.**send**(*slack_client*, *channel: str*, *text: str*)
chat.postMessage to a channel

slacktools.chat.**send_ephemeral**(*slack_client*, *channel: str*, *user: str*, *text: str*)
chat.postEphemeral to a channel and user

https://api.slack.com/methods/chat.postEphemeral

slacktools.chat.**send_ephemeral_factory**(*slack_client*, *channel: str*, *user: str*)
Curried function for sending ephemeral messages so you don't have to keep passing channel and user in.

slacktools.chat.**send_factory**(*slack_client*, *channel: str*)
Curried function for sending messages so you don't have to keep passing channel in.

1.4 Silly Stuff

class slacktools.arguments.**SlackArgumentParser**(*prog=None*, *usage=None*, *description=None*, *epilog=None*, *parents=[]*, *formatter_class=<class 'argparse.HelpFormatter'>*, *prefix_chars='-'*, *fromfile_prefix_chars=None*, *argument_default=None*, *conflict_handler='error'*, *add_help=True*, *allow_abbrev=True*)
If, for some silly reason, you want your bot to accept argparse.ArgumentParser style arguments you can use this.

error(*message*)
We override this method because ArgumentParser's error() method prints stuff to stdout and then calls exit(1)

Raise SlackArgumentParserException

exception slacktools.arguments.**SlackArgumentParserException**

Python Module Index

S

Index