

---

# **discord-ext-slash**

***Release 0.8.1***

**AbyxDev**

**Dec 19, 2021**



**CONTENTS:**

<b>1</b>	<b>Quickstart</b>	<b>3</b>
1.1	Installation . . . . .	3
1.2	Module . . . . .	3
1.2.1	Example Usage . . . . .	3
1.2.2	Notes . . . . .	4
<b>2</b>	<b>API Reference</b>	<b>5</b>
2.1	Decorators . . . . .	5
2.2	Classes . . . . .	5
2.2.1	The Bot . . . . .	5
2.2.2	Interaction Context . . . . .	6
2.2.3	Slash Commands . . . . .	8
2.2.4	Data Classes . . . . .	10
2.2.5	Miscellaneous . . . . .	11
2.2.6	Partial Objects . . . . .	11
2.3	Enums . . . . .	11
2.4	Events . . . . .	12
<b>3</b>	<b>Indices and tables</b>	<b>13</b>
	<b>Python Module Index</b>	<b>15</b>
	<b>Index</b>	<b>17</b>



Go ahead and begin with the Quickstart. It contains a barebones example to get you started. For a more comprehensive demo of various features, take a look at [demo\\_bot.py](#).



## QUICKSTART

## 1.1 Installation

Install with pip like any other package.

```
pip install -U discord-ext-slash
```

**Note:** This installs as an extension to `discord.py`.

## 1.2 Module

Support slash commands.

### 1.2.1 Example Usage

```
from discord.ext import slash
client = slash.SlashBot(
    # normal arguments to commands.Bot()
    command_prefix='.', description="whatever",
    # special option: modify all global commands to be
    # actually guild commands for this guild instead,
    # for the purposes of testing. Remove this argument
    # or set it to None to make global commands be
    # properly global - note that they take 1 hour to
    # propagate. Useful because commands have to be
    # re-registered if their API definitions are changed.
    debug_guild=staging_guild_id
)

msg_opt = slash.Option(
    # description of option, shown when filling in
    description='Message to send',
    # this means that the slash command will not be invoked
    # if this argument is not specified
    required=True)

@client.slash_cmd() # global slash command
async def repeat( # command name
```

(continues on next page)

(continued from previous page)

```
ctx: slash.Context, # there MUST be one argument annotated with Context
message: msg_opt
):
    """Make the bot repeat what you say""" # description of command
    # respond to the interaction, must be done within 3 seconds
    await ctx.respond(message) # string (or str()able) message

client.run(token)
```

## 1.2.2 Notes

- `Context` emulates `discord.ext.commands.Context`, but only to a certain extent. Notably, `ctx.message` does not exist, because slash commands can be run completely without the involvement of messages. However, channel and author information is still available.
- All descriptions are **required**.
- You must grant the bot `applications.commands` permissions in the OAuth2 section of the developer dashboard.

See the [docs](#).



## API REFERENCE

### 2.1 Decorators

`@discord.ext.slash.cmd`

Decorator that transforms a function into a *Command*.

`@discord.ext.slash.group`

Decorator that transforms a function into a *Group*.

`@discord.ext.slash.permit` (*target*: *Union[discord.role.Role, discord.abc.User, discord.object.Object]*, *perm*: *bool*, *guild\_id*: *Optional[int] = Ellipsis*, *type*: *Optional[discord.ext.slash.ApplicationCommandPermissionType] = None*)

Decorator **on top of a command** that adds a permissions overwrite.

### 2.2 Classes

#### 2.2.1 The Bot

**class** `discord.ext.slash.SlashBot` (*\*args*, *\*\*kwargs*)

A bot that supports slash commands.

Constructor arguments in addition to those provided to `discord.ext.commands.Bot` are as follows:

##### Parameters

- **debug\_guild** (*int*) – While testing your bot, it may be useful to have instant command updates for global commands. Setting this to a guild ID will redirect all global commands to commands specific to that guild. Once in production, set this to `None` or do not set it at all.
- **resolve\_not\_fetch** (*bool*) – If `True` (the default), Discord objects passed in arguments will be resolved from the slash commands API, not retrieved or fetched.
- **fetch\_if\_not\_get** (*bool*) – If `False` (the default), Discord objects passed in arguments will not be fetched from the API if retrieving them from cache fails.

**app\_info**: `discord.AppInfo`

Cached output of `application_info()`. Might not be present until `on_ready()` has fired at least once.

**slash**: `set[Command]`

All top-level *Command* and *Group* objects currently registered **in code**.

**@slash\_cmd** (*\*\*kwargs*)  
Create a *Command* with the decorated coroutine and *\*\*kwargs* and add it to *slash*.

**@slash\_group** (*\*\*kwargs*)  
Create a *Group* with the decorated coroutine and *\*\*kwargs* and add it to *slash*.

**add\_slash** (*func*, *\*\*kwargs*)  
Non-decorator version of *slash\_cmd()*.

**add\_slash\_group** (*func*, *\*\*kwargs*)  
Non-decorator version of *slash\_group()*.

**add\_slash\_cog** (*cog: type*)  
Add all attributes of *cog* that are *Command* or *Group* instances.

**Parameters cog** (*type*) – The cog to read attributes from.

**async application\_info** ()  
Equivalent to *discord.Client.application\_info()*, but caches its output in *app\_info*.

**async register\_commands** (*guild\_id: Optional[int] = None*)  
Update commands on the API.

**Parameters guild\_id** (*int*) – Only update commands specific to this guild.

**async register\_permissions** (*guild\_id: Optional[int] = None*)  
Update command permissions on the API.

**Parameters guild\_id** (*int*) – Only update permissions for this guild. Note: All commands will still be updated, but only permissions related to this guild will be updated.

## 2.2.2 Interaction Context

**class** discord.ext.slash.Context (*\*args*, *\*\*kwargs*)  
Object representing an interaction.

**id**: *int*  
The interaction ID.

**guild**: *Union[discord.Guild, discord.Object]*  
The guild where the interaction took place. Can be an *Object* with just the ID if the client is not in the guild.

**channel**: *Union[discord.TextChannel, discord.Object]*  
The channel where the command was run. Can be an *Object* with just the ID if the client is not in the guild.

**author**: *discord.Member*  
The user who ran the command. If *guild* is an *Object*, a lot of *Member* methods that require the guild will break and should not be relied on.

**command**: *Command*  
The command that was run.

**options**: *Mapping[str, Any]*  
The options passed to the command (including this context). More useful in groups and checks.

**me**: *Optional[discord.Member]*  
The bot, as a *Member* in that context. Can be *None* if the client is not in the guild.

**client**: *SlashBot*  
The bot.

**webhook:** `Optional[discord.Webhook]`

Webhook used for sending followup messages. None until interaction response has been sent

**property bot**

The bot. Alias for `client`.

**async respond**(*content*=", \*", *embed*: `Optional[discord.embeds.Embed]` = `None`, *embeds*: `Optional[Iterable[discord.embeds.Embed]]` = `None`, *allowed\_mentions*: `Optional[discord.mentions.AllowedMentions]` = `None`, *file*: `Optional[discord.file.File]` = `None`, *ephemeral*: `bool` = `False`, *deferred*: `bool` = `False`, *flags*: `Optional[Union[discord.ext.slash.CallbackFlags, int]]` = `None`, *rtype*: `discord.ext.slash.InteractionResponseType` = `<InteractionResponseType.CHANNEL_MESSAGE_WITH_SOURCE: 4>`)

Respond to the interaction. If called again, edits the response.

#### Parameters

- **content** (`str`) – The content of the message.
- **embed** (`discord.Embed`) – Shorthand for `respond(embeds=[embed])`
- **embeds** (`Iterable[discord.Embed]`) – Up to 10 embeds (any more will be silently discarded)
- **allowed\_mentions** (`discord.AllowedMentions`) – Mirrors normal `allowed_mentions` in `send()`
- **file** (`discord.File`) – Mirrors normal `file` in `send()`
- **ephemeral** (`bool`) – Shortcut to setting `flags |= CallbackFlags.EPHEMERAL`. If other flags are present, they are preserved.
- **deferred** (`bool`) – Shortcut to setting `rtype = DEFERRED_CHANNEL_MESSAGE_WITH_SOURCE`. Overrides `rtype` unconditionally if `True`.
- **flags** (`Union[CallbackFlags, int]`) – Message flags, ORed together
- **rtype** (`InteractionResponseType`) – The type of response to send. See that class's documentation.

#### Raises

- **TypeError** – if both `embed` and `embeds` are specified.
- **ValueError** – if sending channel message without content.

**async delete()**

Delete the original interaction response message.

**async send**(*\*args*, *\*\*kwargs*)

Send a message in the channel where the the command was run. Equivalent to `send()` for `Context.channel`.

Only method that works after the interaction token has expired. Only works if client is present there as a bot user too.

`discord.ext.slash.Interaction`

alias of `discord.ext.slash.Context`

### 2.2.3 Slash Commands

**class** discord.ext.slash.**Command**(*coro: Coroutine, \*\*kwargs*)

Represents a slash command.

The following constructor argument does not map to an attribute:

**Parameters** **check** (*Coroutine*) – A coroutine to run before calling the command. If it returns `False` (not `falsy`, `False`), then the command is not run.

The following attributes are set by constructor arguments:

**coro:** **Coroutine**

(Required) Original callback for the command.

**id:** **Optional[int]**

ID of registered command. Can be `None` when not yet registered, or if not a top-level command.

**name:** **str**

Command name. Defaults to `coro.__name__`.

**description:** **str**

Description shown in command list. Default `coro.__doc__`.

**guild\_id:** **Optional[int] = None**

If present, this command only exists in this guild.

**parent:** **Optional[Group] = None**

Parent (sub)command group.

**default\_permission:** **bool = True**

If `False`, this command is disabled by default when the bot is added to a new guild. It must be re-enabled per user or role using permissions.

**Raises**

- **TypeError** – if `coro` has a required argument (other than `self`) without an annotation.
- **ValueError** – if no `description` is specified and `coro` has no docstring.
- **ValueError** – if no arguments to `coro` are annotated with `Context` or a subclass.

The following attributes are *not* set by constructor arguments:

**options:** **Mapping[str, Option]**

Options for this command. Set by inspecting the function annotations.

**permissions:** **CommandPermissionsDict**

Permission overrides for this command. A dict of guild IDs to dicts of: role or user or member objects (partial or real) to boolean enable/disable values to grant/deny permissions.

**default:** **bool = False**

If `True`, invoking the base parent of this command translates into invoking this subcommand. (Not settable in arguments.)

**@check**

Set this command's check to this coroutine.

**property** **qualname**

Fully qualified name of command, including group names.

```
add_perm(target: Union[discord.role.Role, discord.abc.User, discord.object.Object],
          perm: bool, guild_id: Optional[int] = Ellipsis, type: Optional[discord.ext.slash.ApplicationCommandPermissionType] = None)
```

Add a permission override.

#### Parameters

- **target** (`Union[discord.Role, PartialRole, discord.Member, discord.User, PartialMember, discord.Object]`) – The role or user to assign this permission to.
- **perm** (`bool`) – True to grant permission, False to deny it
- **guild\_id** (`Optional[int]`) – The guild ID to set the permission for, or None to apply this to the defaults that all guilds inherit for this command. If specified, overrides `target.guild.id`. Must be specified if `target` is a `Object` or a guildless `User`.
- **type** (`ApplicationCommandPermissionType`) – The type of permission grant this is, `ROLE` or `USER`. Must be specified if `target` is a `Object`.

Generally there are four ways of calling this:

- `add_perm(target, perm)` will infer `guild_id` and `type` from `target.guild.id` and the type of `target` (respectively).
- `add_perm(target, perm, guild_id)` will infer the type, but manually set the guild ID (e.g. with a `User` and not a `Member`).
- `add_perm(discord.Object(id), perm, guild_id, type)` will manually set the guild ID and type since all you have is an ID.
- `add_perm(..., guild_id=None)` will do any of the above but apply the permissions to the defaults that all specific-guild permissions will inherit from, instead of applying to any particular guild.

#### Raises

- **ValueError** – if `type` is unspecified but cannot be inferred.
- **ValueError** – if `guild_id` is unspecified but cannot be inferred.

```
class discord.ext.slash.Group(coro: Coroutine, **kwargs)
```

Represents a group of slash commands. Attributes and constructor arguments are the same as `Command` unless documented below.

**Parameters** `coro` (`Coroutine`) – (Required) Callback invoked when a subcommand of this group is called. (This is not a check! Register a check using `check()`.)

```
slash: Mapping[str, Union[Group, Command]]
```

Subcommands of this group.

```
@slash_cmd(**kwargs)
```

See `SlashBot.slash_cmd()`.

```
@slash_group(**kwargs)
```

See `SlashBot.slash_group()`.

```
add_slash(func, **kwargs)
```

See `SlashBot.add_slash()`.

```
add_slash_group(func, **kwargs)
```

See `SlashBot.add_slash_group()`.

## 2.2.4 Data Classes

```
class discord.ext.slash.Option(description: Union[str, Type[discord.ext.slash.ChoiceEnum]],
                               type: discord.ext.slash.ApplicationCommandOptionType = <ApplicationCommandOptionType.STRING: 3>, **kwargs)
```

An argument to a [Command](#). This must be passed as an annotation to the corresponding argument.

Constructor arguments map directly to attributes, besides the ones below which have different type signatures:

### Parameters

- **description** (`Union[str, Type[ChoiceEnum]]`) – Annotating a parameter with `EnumClass` has the same effect as with `Option(description=EnumClass)`.
- **choices** (`Optional[Iterable[Union[str, Mapping[str, str], Choice]]]`) – Strings are converted into [Choice](#) objects with the same name and value. `dict` objects are passed as kwargs to the [Choice](#) constructor.
- **channel\_types** (`Optional[Iterable[Union[int, discord.ChannelType]]]`) – Pass either the raw integers or the enum values.
- **channel\_type** (`Optional[Union[int, discord.ChannelType]]`) – A shortcut to `channel_types=[channel_type]`.

**description:** `str`

The description of the option, displayed to users.

**type:** `ApplicationCommandOptionType = :attr:`ApplicationCommandOptionType.STRING``

The argument type.

**name:** `Optional[str] = None`

The name of the option, if different from its argument name.

**required:** `bool = False`

If `True`, this option must be specified for a valid command invocation.

**choices:** `Optional[list[Choice]]`

Only these values are allowed for this option.

**channel\_types:** `Optional[set[discord.ChannelType]]`

Sets type to `ApplicationCommandOptionType.CHANNEL`, additionally restricted to a set of specific channel types.

**min\_value:** `Union[int, float, None]`

For numerical options, this is the minimum value allowable. If type is not a numerical type, it is inferred from the type of this argument. Otherwise, this argument is cast to the type corresponding to type.

**max\_value:** `Union[int, float, None]`

Same as `min_value` but maximum. If both `min_value` and `max_value` are specified, *and* type is non-numeric, type is inferred from this argument, not `min_value`.

```
class discord.ext.slash.Choice(name: str, value: str)
```

Represents one choice for an option value.

Constructor arguments map directly to attributes.

**name:** `str`

The description of the choice, displayed to users.

**value:** `str`

The actual value fed into the application.

## 2.2.5 Miscellaneous

```
class discord.ext.slash.SlashWarning
    discord.ext.slash-specific warning type.

discord.ext.slash.CommandPermissionsDict
    alias of Dict[Optional[int], Dict[Tuple[int, discord.ext.slash.
    ApplicationCommandPermissionType], bool]]
```

## 2.2.6 Partial Objects

Objects resolved from the slash commands API often do not contain all the information that discord.py prefers (most notably guild information).

```
class discord.ext.slash.PartialObject(id)
    Bases: discord.object.Object

    Subclasses of this have their .attr:guild as a Object, so their guild-related functionality may break.

    guild: discord.Object

class discord.ext.slash.PartialTextChannel(*, state, guild, data)
    Bases: discord.channel.TextChannel, discord.ext.slash.PartialObject

    A partial TextChannel.

class discord.ext.slash.PartialCategoryChannel(*, state, guild, data)
    Bases: discord.channel.CategoryChannel, discord.ext.slash.PartialObject

    A partial CategoryChannel.

class discord.ext.slash.PartialVoiceChannel(*, state, guild, data)
    Bases: discord.channel.VoiceChannel, discord.ext.slash.PartialObject

    A partial VoiceChannel.

class discord.ext.slash.PartialMember(*, data, guild, state)
    Bases: discord.member.Member, discord.ext.slash.PartialObject

    A partial Member.

class discord.ext.slash.PartialRole(*, guild, state, data)
    Bases: discord.role.Role, discord.ext.slash.PartialObject

    A partial Role.
```

## 2.3 Enums

```
class discord.ext.slash.ApplicationCommandOptionType(value)
    Possible CommandOption types. Default is STRING.

    SUB_COMMAND
        Marks a sub-Command, only used internally.

    SUB_COMMAND_GROUP
        Marks a Group, only used internally.
```

The type signatures of the below attributes mark the type that the argument value is passed as. For example, options of type *STRING* are passed as *str*.

```
STRING: str
INTEGER: int
BOOLEAN: bool
USER: Union[discord.Member, discord.User, PartialMember, discord.Object]
CHANNEL: Union[discord.TextChannel, discord.CategoryChannel, discord.VoiceChannel, PartialChannel]
ROLE: Union[discord.Role, PartialRole, discord.Object]
MENTIONABLE: Union[discord.Member, discord.User, PartialMember, discord.Role, PartialRole]
NUMBER: float
```

**class** discord.ext.slash.**ApplicationCommandPermissionType**(value)  
Possible types of permission grants. For use in `Command.add_perm()` and `permit()`.

**ROLE**  
Specifies that this permission grant is to a role.

**USER**  
Specifies that this permission grant is to a user.

**class** discord.ext.slash.**InteractionResponseType**(value)  
Possible ways to respond to an interaction. For use in `Context.respond()`.

**PONG**  
Only used to ACK a Ping, never valid here. Included only for completeness.

**CHANNEL\_MESSAGE\_WITH\_SOURCE**  
Show user input and send a message. Default.

**DEFERRED\_CHANNEL\_MESSAGE\_WITH\_SOURCE**  
Show user input and display a “waiting for bot” system message. Send a response with this type and edit the response later if you need to do some asynchronous fetch or something.

**class** discord.ext.slash.**CallbackFlags**(value)  
Flags to pass to the flags argument of `Context.respond()`.

**EPHEMERAL**  
Only the user receiving the message can see it

discord.ext.slash.**MessageFlags**  
alias of `discord.ext.slash.CallbackFlags`

**class** discord.ext.slash.**ChoiceEnum**(value)  
Callback parameters annotated with subclasses of this class will use the enums as choices. See the `/numbers` command in the demo bot for an example.

## 2.4 Events

discord.ext.slash.**on\_interaction\_create**(event: dict)  
Triggered by Discord interactions. For internal use.

discord.ext.slash.**on\_slash\_permissions**()  
Triggered immediately after `SlashBot.register_commands()` to give an opportunity to register dynamic permissions in code before pushing to the API. If overriding using `@:meth:discord.Client.event`, you must await `-SlashBot.register_permissions()` at the end of the event handler. See `/stop` in `demo_bot.py` for an example.



## INDICES AND TABLES

- `genindex`
- `search`



## PYTHON MODULE INDEX

### d

`discord.ext.slash`, 3



## INDEX

### A

`add_perm()` (*discord.ext.slash.Command* method), 8  
`add_slash()` (*discord.ext.slash.Group* method), 9  
`add_slash()` (*discord.ext.slash.SlashBot* method), 6  
`add_slash_cog()` (*discord.ext.slash.SlashBot* method), 6  
`add_slash_group()` (*discord.ext.slash.Group* method), 9  
`add_slash_group()` (*discord.ext.slash.SlashBot* method), 6  
`app_info` (*discord.ext.slash.SlashBot* attribute), 5  
`application_info()` (*discord.ext.slash.SlashBot* method), 6  
`ApplicationCommandOptionType` (class in *discord.ext.slash*), 11  
`ApplicationCommandPermissionType` (class in *discord.ext.slash*), 12  
`author` (*discord.ext.slash.Context* attribute), 6

### B

`BOOLEAN` (*discord.ext.slash.ApplicationCommandOptionType* attribute), 12  
`bot()` (*discord.ext.slash.Context* property), 7

### C

`CallbackFlags` (class in *discord.ext.slash*), 12  
`CHANNEL` (*discord.ext.slash.ApplicationCommandOptionType* attribute), 12  
`channel` (*discord.ext.slash.Context* attribute), 6  
`CHANNEL_MESSAGE_WITH_SOURCE` (*discord.ext.slash.InteractionResponseType* attribute), 12  
`channel_types` (*discord.ext.slash.Option* attribute), 10  
`check()` (*discord.ext.slash.Command* method), 8  
`Choice` (class in *discord.ext.slash*), 10  
`ChoiceEnum` (class in *discord.ext.slash*), 12  
`choices` (*discord.ext.slash.Option* attribute), 10  
`client` (*discord.ext.slash.Context* attribute), 6  
`cmd()` (in module *discord.ext.slash*), 5  
`Command` (class in *discord.ext.slash*), 8  
`command` (*discord.ext.slash.Context* attribute), 6

`CommandPermissionsDict` (in module *discord.ext.slash*), 11

`Context` (class in *discord.ext.slash*), 6  
`coro` (*discord.ext.slash.Command* attribute), 8

### D

`default` (*discord.ext.slash.Command* attribute), 8  
`default_permission` (*discord.ext.slash.Command* attribute), 8  
`DEFERRED_CHANNEL_MESSAGE_WITH_SOURCE` (*discord.ext.slash.InteractionResponseType* attribute), 12  
`delete()` (*discord.ext.slash.Context* method), 7  
`description` (*discord.ext.slash.Command* attribute), 8  
`description` (*discord.ext.slash.Option* attribute), 10  
`discord.ext.slash` module, 3

### E

`EPHEMERAL` (*discord.ext.slash.CallbackFlags* attribute), 12

### G

`Group` (class in *discord.ext.slash*), 9  
`group()` (in module *discord.ext.slash*), 5  
`guild` (*discord.ext.slash.Context* attribute), 6  
`guild` (*discord.ext.slash.PartialObject* attribute), 11  
`guild_id` (*discord.ext.slash.Command* attribute), 8

### I

`id` (*discord.ext.slash.Command* attribute), 8  
`id` (*discord.ext.slash.Context* attribute), 6  
`INTEGER` (*discord.ext.slash.ApplicationCommandOptionType* attribute), 12  
`Interaction` (in module *discord.ext.slash*), 7  
`InteractionResponseType` (class in *discord.ext.slash*), 12

### M

`max_value` (*discord.ext.slash.Option* attribute), 10  
`me` (*discord.ext.slash.Context* attribute), 6

MENTIONABLE (*discord.ext.slash.ApplicationCommandOptionType* attribute), 12

MessageFlags (*in module discord.ext.slash*), 12

min\_value (*discord.ext.slash.Option* attribute), 10

module

`discord.ext.slash`, 3

## N

name (*discord.ext.slash.Choice* attribute), 10

name (*discord.ext.slash.Command* attribute), 8

name (*discord.ext.slash.Option* attribute), 10

NUMBER (*discord.ext.slash.ApplicationCommandOptionType* attribute), 12

## O

on\_interaction\_create() (*in module discord.ext.slash*), 12

on\_slash\_permissions() (*in module discord.ext.slash*), 12

Option (*class in discord.ext.slash*), 10

options (*discord.ext.slash.Command* attribute), 8

options (*discord.ext.slash.Context* attribute), 6

## P

parent (*discord.ext.slash.Command* attribute), 8

PartialCategoryChannel (*class in discord.ext.slash*), 11

PartialMember (*class in discord.ext.slash*), 11

PartialObject (*class in discord.ext.slash*), 11

PartialRole (*class in discord.ext.slash*), 11

PartialTextChannel (*class in discord.ext.slash*), 11

PartialVoiceChannel (*class in discord.ext.slash*), 11

permissions (*discord.ext.slash.Command* attribute), 8

permit() (*in module discord.ext.slash*), 5

PONG (*discord.ext.slash.InteractionResponseType* attribute), 12

## Q

qualname() (*discord.ext.slash.Command* property), 8

## R

register\_commands() (*discord.ext.slash.SlashBot* method), 6

register\_permissions() (*discord.ext.slash.SlashBot* method), 6

required (*discord.ext.slash.Option* attribute), 10

respond() (*discord.ext.slash.Context* method), 7

ROLE (*discord.ext.slash.ApplicationCommandOptionType* attribute), 12

ROLE (*discord.ext.slash.ApplicationCommandPermissionType* attribute), 12

send() (*discord.ext.slash.Context* method), 7

slash (*discord.ext.slash.Group* attribute), 9

slash (*discord.ext.slash.SlashBot* attribute), 5

slash\_cmd() (*discord.ext.slash.Group* method), 9

slash\_cmd() (*discord.ext.slash.SlashBot* method), 9

slash\_group() (*discord.ext.slash.Group* method), 9

slash\_group() (*discord.ext.slash.SlashBot* method), 6

SlashBot (*class in discord.ext.slash*), 5

SlashWarning (*class in discord.ext.slash*), 11

STRING (*discord.ext.slash.ApplicationCommandOptionType* attribute), 11

SUB\_COMMAND (*discord.ext.slash.ApplicationCommandOptionType* attribute), 11

SUB\_COMMAND\_GROUP (*discord.ext.slash.ApplicationCommandOptionType* attribute), 11

## T

type (*discord.ext.slash.Option* attribute), 10

## U

USER (*discord.ext.slash.ApplicationCommandOptionType* attribute), 12

USER (*discord.ext.slash.ApplicationCommandPermissionType* attribute), 12

## V

value (*discord.ext.slash.Choice* attribute), 10

## W

webhook (*discord.ext.slash.Context* attribute), 6