dpy-components

Release 0.2.0

sevenc-nanashi

Dec 31, 2021

CONTENTS:

This library lets you send message with components, and receive components' event!

CHAPTER

ONE

API DOCUMENTATION

1.1 Receive components' event

class components.**ComponentsCog**

A cog that receive and handles components' event. You can load this cog as extension with:

```
bot.load_extension("discord.ext.components")
```

1.1.1 events

on_button_click(*com*)

Fires when user clicked button.

Parameters **com** (*components.ButtonResponse*) – A response.

on_menu_select(*com*)

Fires when user selected menu.

Parameters **com** (*components.SelectMenuResponse*) – A response.

1.2 Respond components' event

class components.**ButtonResponse**(*bot*, *data*, *state*)

Represents a button response. Do not initialize this class directly.

async defer_source(*hidden=False*)

ACK an interaction with DeferredChannelMessageWithSource(5). The user sees a loading state.

Parameters **hidden** (*bool*) – Hide interaction response or not.

async defer_update()

ACK an interaction with DeferredUpdateMessage(6). The user doesn't see a loading state.

Parameters **hidden** (*bool*) – Hide interaction response or not.

property fired_by

Return `self.member` or `self.user`.

async send(*content=None*, ***, *embed=None*, *embeds=[]*, *allowed_mentions=None*, *hidden=False*, *tts=False*, *components=[]*)

Responds interaction.

Parameters

- **content...tts** – Same as discord.abc.Messageable.send.
- **hidden** (*bool*) – Hide the message or not.

class components.**SelectMenuResponse**(*bot*, *data*, *state*)

Represents a select menu response. Do not initialize this class directly.

async **defer_source**(*hidden=False*)

ACK an interaction with DeferredChannelMessageWithSource(5). The user sees a loading state.

Parameters **hidden** (*bool*) – Hide interaction response or not.

async **defer_update**()

ACK an interaction with DeferredUpdateMessage(6). The user doesn't see a loading state.

Parameters **hidden** (*bool*) – Hide interaction response or not.

property **fired_by**

Return self.member or self.user.

async **send**(*content=None*, ***, *embed=None*, *embeds=[]*, *allowed_mentions=None*, *hidden=False*, *tts=False*)

Responds interaction.

Parameters

- **content...tts** – Same as discord.abc.Messageable.send.
- **hidden** (*bool*) – Hide the message or not.

property **value**

Return the first value of values or None.

1.3 Send messages with components

async components.**send**(*channel*, *content=None*, ***, *tts=False*, *embed=None*, *embeds=None*, *file=None*, *files=None*, *delete_after=None*, *nonce=None*, *allowed_mentions=None*, *reference=None*, *mention_author=None*, *components=[]*)

Send message with components.

Parameters

- **channel** (*discord.abc.Messageable*) – Channel to send the message.
- **content..mention_author** – Same as discord.abc.Messageable.send.
- **components** (*list, optional*) – Components to attach to the message. Specify 2D list if you want to use multi row components.

Returns Message was sent.

Return type Message

async components.**reply**(*target*, **args*, ***kwargs*)

An utility function for replying message.

1.3.1 Components

class components.**Button**(*label: str*, *custom_id: Optional[str] = None*, *style: Union[int, components.sender.ButtonType] = ButtonType.primary*, *url: Optional[str] = None*, *emoji: Optional[Union[discord.emoji.Emoji, str]] = None*, *enabled: bool = True*, *name: Optional[str] = None*)

Represents a button in component.

Parameters

- **label** (*str*) – Label for the button.
- **custom_id** (*Optional[str]*) – Custom id for the button.
- **style** (*Union[int, ButtonType]*) – Style for the button.
- **url** (*Optional[str]*) – URL for the button.
- **emoji** (*Union[Emoji, str]*) – Emoji for the button.
- **enabled** (*bool*) – Weather button is enabled or disabled.

class components.**SelectMenu**(*custom_id: Optional[str]*, *options: List[components.sender.SelectOption]*, *placeholder: Optional[str] = None*, *min_values: int = 1*, *max_values: int = 1*)

Represents a select menu in component.

Parameters

- **custom_id** (*Optional[str]*) – Custom id for the select menu.
- **options** (*List[SelectOption]*) – Options for the select menu.
- **placeholder** (*Optional[str]*) – Placeholder for the select menu.
- **min_values** (*int*) – Minimum number of items that must be chosen.
- **max_values** (*int*) – Maximum number of items that must be chosen.

class components.**SelectOption**(*label: str*, *value: str*, *description: Optional[str] = None*, *emoji: Optional[Union[discord.emoji.Emoji, str]] = None*, *default: bool = False*)

Represents a option for the select menu.

Parameters

- **label** (*str*) – Label for the option.
- **value** (*str*) – Value for the option.
- **description** (*Optional[str]*) – Description for the option.
- **emoji** (*Union[Emoji, str]*) – Emoji for the option.
- **default** (*bool*) – Weather option is default.

1.3.2 Styles

class components.**ButtonType**
Represents style of button.

primary

primary_cta

blue

blurple
Represents style 1.

secondary

gray

grey
Represents style 2.

success

primary_success

green
Represents style 3.

danger

destructive

red
Represents style 4.

link

url
Represents style 5.

CHAPTER

TWO

EXAMPLES

2.1 Send button and receive

```
import os

from discord.ext import commands, components

bot = commands.Bot("c ")
bot.load_extension("discord.ext.components")


@bot.event
async def on_ready():
    print('We have logged in as {0.user}'.format(bot))


@bot.command()
async def test(ctx, button_label, hidden: bool):
    await components.send(ctx, "Click this", components=[components.Button(button_label,␣
↪custom_id="button1")])
    com = await bot.wait_for("button_click", check=lambda c: c.name == "button1")
    await com.send(f"You clicked {button_label}.", hidden=hidden)

bot.run(os.getenv("token"))
```

2.2 Authorization button

```
import os

import discord
from discord.ext import commands, components

bot = commands.Bot("c ")
bot.load_extension("discord.ext.components")


@bot.event
async def on_ready():
```

(continues on next page)

(continued from previous page)

```
    print('We have logged in as {0.user}'.format(bot))


@bot.command()
async def send_auth(ctx):
    await components.send(ctx, "Click this button to get your member role",␣
↪components=[components.Button("Get member role", custom_id="get_auth_role",␣
↪style=components.ButtonType.green)])


@bot.event
async def on_button_click(com):
    if com.custom_id == "get_auth_role":
        await com.defer_source(hidden=True)
        role = discord.utils.get(com.guild.roles, name="Member")
        if role in com.member.roles:
            await com.send("You already have your member role.")
        else:
            await com.member.add_roles(role)
            await com.send("You got your member role. Enjoy!")

bot.run(os.getenv("token"))
```

2.3 Pagenation with select menu

```
import asyncio
import os

import discord
from discord.ext import commands
from discord.ext import components
bot = commands.Bot("c ")
bot.load_extension("discord.ext.components")

pages = [
    "Done is better than perfect.\n\n--Mark Zuckerberg",
    "The best way to predict the future is to invent it.\n\n--Alan Key",
    "Programs must be written for people to read, and only incidentally for machines to␣
↪execute.\n\n--Hal Alverson"
]


@bot.event
async def on_ready():
    print('We have logged in as {0.user}'.format(bot))


@bot.command()
async def send_page(ctx):
    options = []
```

(continues on next page)

(continued from previous page)

```
    for i, _ in enumerate(pages, 1):
        options.append(components.SelectOption(f"Page {i}", f"pagenation_{i}"))
    msg = await components.send(ctx, "Use select menu for switch page", components=[
        components.SelectMenu("pagenation", options, "Select page...")
    ])
    try:
        while True:
            com = await bot.wait_for("menu_select", check=lambda c: c.message == msg,␣
↪timeout=30)
            page = int(com.value.removeprefix("pagenation_"))
            await com.send(pages[page - 1] + f"\n\n`Page {page}`", hidden=True)
    except asyncio.TimeoutError:
        return


bot.run(os.getenv("discord_bot_token"))
```

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX