dpy-components

リリース *0.2.0*

sevenc-nanashi

2021 年 12 月 31 日

内容：

このライブラリでコンポーネントを送信したり、受け取ったり出来ます！

第 1 章

API ドキュメント

1.1 コンポーネントのイベントを受け取る

class components.**ComponentsCog**

コンポーネントのイベントを受け取り、ハンドルするコグ。エクステンションとして読み込むには以下のようにして下さい：

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

1.1.1 イベント

on_button_click(*com*)

ボタンをクリックしたときに発火するイベント。

パラメータ **com** (*components.ButtonResponse*) -- Gateway からのレスポンス。

on_menu_select(*com*)

Fires when user selected menu.

パラメータ **com** (*components.SelectMenuResponse*) -- Gateway からのレスポンス。

1.2 コンポーネントのイベントに応答する

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

ボタンのイベントです。このクラスを直接初期化しないで下さい。

async defer_source(*hidden=False*)

DeferredChannelMessageWithSource(5) で応答します。``○○が考え中...``が表示されます。

パラメータ **hidden** (*bool*) -- 応答を隠すかどうか。

async defer_update()
DeferredUpdateMessage(6) で応答します。``○○が考え中...``は表示されません。

パラメータ **hidden** (*bool*) -- 応答を隠すかどうか。

property fired_by
self.member または self.user を返します。

async send(*content=None*, ***, *embed=None*, *embeds=[]*, *allowed_mentions=None*, *hidden=False*, *tts=False*, *components=[]*)
インタラクションに反応します。

パラメータ

- **content...tts** -- ``discord.abc.Messageable.send``と同じです。
- **hidden** (*bool*) -- メッセージを隠すかどうか。

class components.**SelectMenuResponse**(*bot*, *data*, *state*)
ドロップダウンのイベントです。このクラスを直接初期化しないで下さい。

async defer_source(*hidden=False*)
DeferredChannelMessageWithSource(5) で応答します。``○○が考え中...``が表示されます。

パラメータ **hidden** (*bool*) -- 応答を隠すかどうか。

async defer_update()
DeferredUpdateMessage(6) で応答します。``○○が考え中...``は表示されません。

パラメータ **hidden** (*bool*) -- 応答を隠すかどうか。

property fired_by
self.member または self.user を返します。

async send(*content=None*, ***, *embed=None*, *embeds=[]*, *allowed_mentions=None*, *hidden=False*, *tts=False*)
インタラクションに反応します。

パラメータ

- **content...tts** -- ``discord.abc.Messageable.send``と同じです。
- **hidden** (*bool*) -- メッセージを隠すかどうか。

property value
Return the first value of values or None.

1.3 メッセージをコンポーネント付きで送信する

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=[]*)

メッセージをコンポーネント付きで送信します。

パラメータ

- **channel** (*discord.abc.Messageable*) -- メッセージを送信するチャンネル。
- **content..mention_author** -- ``discord.abc.Messageable.send``と同じです。
- **components** (*list, optional*) -- メッセージに追加するコンポーネント。複数行にしたい場合は 2 次元配列を指定して下さい。

戻り値 送信したメッセージ。

戻り値の型 Message

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

メッセージへ返信する為の関数。

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.

パラメータ

- **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.

パラメータ

- **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.

パラメータ

- **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 スタイル

class components.**ButtonType**

ボタンのスタイルを表します。

primary

primary_cta

blue

blurple

スタイル``1``を表します。

secondary

gray

grey
スタイル``2``を表します。

success

primary_success

green
スタイル``3``を表します。

danger

destructive

red
スタイル``4``を表します。

link

url
スタイル``5``を表します。

第 2 章

サンプル

2.1 ボタンを送信する

```
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 認証ボタン

```
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():
    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 = []
    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
```

(次のページに続く)

(前のページからの続き)

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

第 3 章

索引

- genindex
- modindex
- search

索引