disputils

Release 0.2.0

May 29, 2021

Contents:

CHAPTER 1

Quickstart guide

Disputils provides different and customizable utilities for discord.py bot developers. This quickstart guide aims to help you understand the basic features of those utilities and get you started as quick as possible.

The library contains two versions of each utility. Which one to use depends on how you use `discord.py`:

- basic `discord.py`, via `discord.Client`
- `discord.py`'s commands extension, via `discord.ext.commands.Bot`

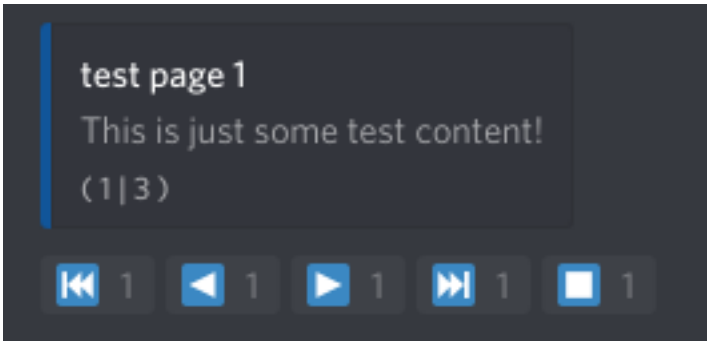
1.1 Installation

You can easily install the latest version from PyPI via pip:

```
pip install disputils
```

1.2 Pagination

The pagination utility allows you to create and control an interactive paginated representation of multiple embeds that users can navigate via reactions.



1.2.1 client version

The simplest way of doing a pagination:

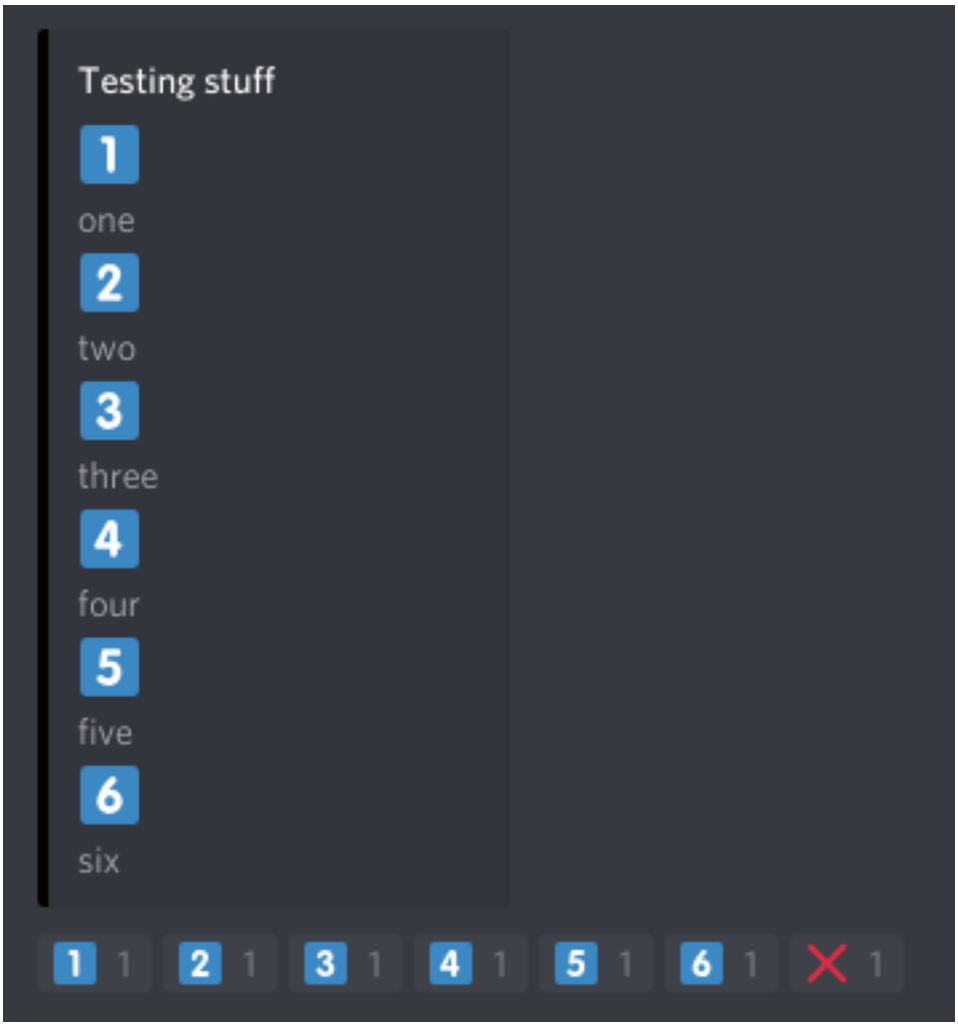
1.2.2 bot version

For the commands extension, we need to slightly modify the last two lines:

1.3 Multiple Choice

This utility gives the power of choice to your users. Let it be polls, interactive menus or something else, you can take user experience to a new level.

Users can interact via automatically generated reactions under the message.



The appearance and behavior is of course customizable.

1.3.1 client version

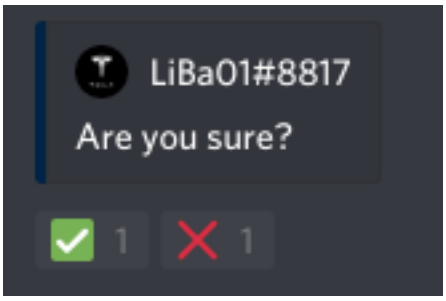
A cool way to realize an interactive multiple choice command, that dynamically changes it's content is this:

1.3.2 bot version

This is what a simple multiple choice command could look like:

1.4 Confirmation

This one is pretty simple and straight-forward. It lets you ask users for confirmation on an action. It's basically a yes-no question.



1.4.1 client version

An interactive confirmation:

1.4.2 bot version

An example confirmation command looks like this:

CHAPTER 2

Code Reference

2.1 Dialog

class disputils.abc.**Dialog**(**args*, ***kwargs*)

Abstract base class defining a general embed dialog interaction.

display(*text: str = None*, *embed: discord.embeds.Embed = None*)

This will edit the dialog message.

Parameters

- **text** – The new text.
- **embed** – The new embed.

Return type None

quit(*text: str = None*)

Quit the dialog.

Parameters **text** (str, optional) – message text to display when dialog is closed

Return type None

update(*text: str*, *color: hex = None*, *hide_author: bool = False*)

This will update the dialog embed.

Parameters

- **text** – The new text.
- **color** – The new embed color.
- **hide_author** – True if you want to hide the embed author (default: False).

Return type None

2.2 Pagination

class disputils.**EmbedPaginator**(*client: discord.client.Client, pages: [<class 'discord.embeds.Embed'>], message: discord.message.Message = None, *, control_emojis: Union[disputils.pagination.ControlEmojis, tuple, list] = ControlEmojis(first='', previous='', next='', last='', close='')*)

Represents an interactive menu containing multiple embeds.

Parameters

- **client** – The `discord.Client` to use.
- **pages** – A list of `discord.Embed` to paginate through.
- **message** – An optional `discord.Message` to edit. Otherwise a new message will be sent.
- **control_emojis** – `ControlEmojis`, *tuple* or *list* containing control emojis to use, otherwise the default will be used. A value of *None* causes a reaction to be left out.

formatted_pages

The embeds with formatted footers to act as pages.

static generate_sub_lists(*origin_list: list*, *max_len: int = 25*) → List[list]

Takes a list of elements and transforms it into a list of sub-lists of those elements with each sublist containing max. `max_len` elements.

This can be used to easily split content for embed-fields across multiple pages.

Note: Discord allows max. 25 fields per Embed (see Embed Limits). Therefore, `max_len` must be set to a value of 25 or less.

Parameters

- **origin_list** (`list`) – total list of elements
- **max_len** (`int`, optional) – maximal length of a sublist

Returns list of sub-lists of elements

Return type `List[list]`

run(*users: List[discord.user.User], channel: discord.channel.TextChannel = None, timeout: int = 100, **kwargs*)

Runs the paginator.

Parameters

- **users** (*`List[discord.User]`*) – A list of `discord.User` that can control the pagination. Passing an empty list will grant access to all users. (Not recommended.)
- **channel** (*`Optional[discord.TextChannel]`*) – The text channel to send the embed to. Must only be specified if *self.message* is *None*.
- **timeout** (*`int`*) – Seconds to wait until stopping to listen for user interaction.
- **kwargs** –
  - text `str`: Text to appear in the pagination message.
  - timeout_msg `str`: Text to appear when pagination times out.

- quit_msg `str`: Text to appear when user quits the dialog.

Returns None

---

class disputils.**BotEmbedPaginator**(*ctx: discord.ext.commands.context.Context, pages: [<class 'discord.embeds.Embed'>], message: discord.message.Message = None, *, control_emojis: Union[disputils.pagination.ControlEmojis, tuple, list] = ControlEmojis(first='', previous='', next='', last='', close='')*)

Same as *EmbedPaginator*, except for the discord.py commands extension.

Parameters

- **ctx** – The `discord.ext.commands.Context` to use.
- **pages** – A list of `discord.Embed` to paginate through.
- **message** – An optional `discord.Message` to edit. Otherwise a new message will be sent.
- **control_emojis** – `ControlEmojis`, *tuple* or *list* containing control emojis to use, otherwise the default will be used. A value of *None* causes a reaction to be left out.

run(*channel: discord.channel.TextChannel = None*, *users: List[discord.user.User] = None*, *timeout: int = 100*, ***kwargs*)

Runs the paginator.

Parameters

- **channel** (*Optional[discord.TextChannel]*) – The text channel to send the embed to. Default is the context channel.
- **users** (*Optional[List[discord.User]]*) – A list of `discord.User` that can control the pagination. Default is the context author. Passing an empty list will grant access to all users. (Not recommended.)
- **timeout** (*int*) – Seconds to wait until stopping to listen for user interaction.
- **kwargs** –
  - text `str`: Text to appear in the pagination message.
  - timeout_msg `str`: Text to appear when pagination times out.
  - quit_msg `str`: Text to appear when user quits the dialog.

Returns None

2.3 MultipleChoice

class disputils.**MultipleChoice**(*client: discord.client.Client, options: List[str], title: str, description: str = '', **kwargs*)

Generate and manage a reaction controlled rich embed multiple choice poll in Discord.

Parameters

- **title** (`str`) – Embed title.
- **description** (`str`) – Embed description.

- **options** (list[str]) – Options to choose from. Each option is going to be a separate embed field.

choice
The option that the user chose.

embed
The generated embed.

run(*users: Union[discord.user.User, List[discord.user.User]] = None, channel: discord.channel.TextChannel = None, **kwargs*) → Tuple[Optional[str], discord.message.Message]
Run the multiple choice dialog.

Parameters

- **users** (list[discord.User]) – Users that can use the reactions (default: *None*). If this is None: Any user can interact.
- **channel** (discord.TextChannel, optional) – The channel to send the message to.
- **kwargs** –
  - message discord.Message
  - timeout int (seconds, default: 60),
  - closable bool (default: True)
  - text str: Text to appear in the message.
  - timeout_msg str: Text to appear when dialog times out.
  - quit_msg str: Text to appear when user quits the dialog.

Returns selected option and used discord.Message

Return type tuple[str, discord.Message]

---

class disputils.**BotMultipleChoice**(*ctx: discord.ext.commands.context.Context, options: list, title: str, description: str = '', **kwargs*)
Same as *MultipleChoice*, except for the discord.py commands extension.

run(*users: Union[discord.user.User, List[discord.user.User]] = None, channel: discord.channel.TextChannel = None, **kwargs*) → Tuple[Optional[str], discord.message.Message]
Run the multiple choice dialog.

Parameters

- **users** (list[discord.User]) – Users that can use the reactions (default: *None*). If this is None: Any user can interact.
- **channel** (discord.TextChannel, optional) – The channel to send the message to.
- **kwargs** –
  - message discord.Message
  - timeout int (seconds, default: 60),
  - closable bool (default: True)
  - text str: Text to appear in the message.
  - timeout_msg str: Text to appear when dialog times out.
  - quit_msg str: Text to appear when user quits the dialog.

Returns selected option and used `discord.Message`

Return type tuple[`str`, `discord.Message`]

2.4 Confirmation

class disputils.**Confirmation**(*client: discord.client.Client*, *color: hex = 0*, *message: discord.message.Message = None*)

Represents a message to let the user confirm a specific action.

confirm(*text: str*, *user: discord.user.User*, *channel: discord.channel.TextChannel = None*, *hide_author: bool = False*, *timeout: int = 20*) → bool

Run the confirmation.

Parameters

- **text** (`str`) – The confirmation text.
- **user** (`discord.User`) – The user who has to confirm.
- **channel** (`discord.TextChannel`, optional) – The channel the message will be sent to. Must only be specified if `self.message` is None.
- **hide_author** (*bool, optional*) – Whether or not the `user` should be set as embed author.
- **timeout** (*int*) – Seconds to wait until stopping to listen for user interaction.

Returns True when it's been confirmed, otherwise False. Will return None when a timeout occurs.

Return type `bool`, optional

confirmed

Whether the user has confirmed the action.

---

class disputils.**BotConfirmation**(*ctx: discord.ext.commands.context.Context*, *color: hex = 0*, *message: discord.message.Message = None*)

confirm(*text: str*, *user: discord.user.User = None*, *channel: discord.channel.TextChannel = None*, *hide_author: bool = False*, *timeout: int = 20*) → bool

Run the confirmation.

Parameters

- **text** (`str`) – The confirmation text.
- **user** (`discord.User`) – The user who has to confirm.
- **channel** (`discord.TextChannel`, optional) – The channel the message will be sent to. Must only be specified if `self.message` is None.
- **hide_author** (*bool, optional*) – Whether or not the `user` should be set as embed author.
- **timeout** (*int*) – Seconds to wait until stopping to listen for user interaction.

Returns True when it's been confirmed, otherwise False. Will return None when a timeout occurs.

Return type `bool`, optional

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Index