promptr

Release 0.2.0

Matt Davis

Sep 11, 2019

CONTENTS

1 Project Info 3

2 Indices and tables 9

Python Module Index 11

Index 13

Release v0.2.0. (*Changelog*)

promptr is a toolkit for building CLI shells, similar to that found on a Cisco router.

CHAPTER

ONE

PROJECT INFO

1.1 API

exception promptr.**AmbiguousCommand**(*cmd*, *args*)
Bases: *promptr.PromptrError*

The promptr command was ambiguous

The input from the user didn't matched multiple states, groups, or commands and so promptr could not choose one to execute.

Parameters

- **cmd** (str) – The attempted command
- **args** (List[str]) – The arguments passed to the command

class promptr.**Argument**(*name*, ***kwargs*)
Bases: object

promptr argument

Base class for promptr arguments. Used by the argument decorator to add an argument to a state, group, or command. Arguments require a name and can take an optional function that provides completions.

Parameters

- **name** (str) – The name of the argument. This will be used to pass the value to the state, group, or command as a kwarg
- **kwargs** (Dict[str, Any]) – A dictionary of keyword arguments.

  completions Either a list of str, or a generator that yields str.

property completions
Return a list of completions that are valid for this *Argument*

Return type List[str]

property name
Get the name of the *Argument*

Return type str

class promptr.**Base**(*name*, *callback*, *params*, *parent*, ***kwargs*)
Bases: object

Base class for most promptr objects

Shouldn't be a need to instantiate this directly but can be sub-classed to add functionality or to extend a current class. Instances of this class can be passed to the decorator as the *cls* keyword in order to customise th behaviour of the library.

Parameters

- **name** (str) – The name that this state, group, or command will be referred by. Used to derive the possible set of completions. Auto generated using the functions name, but can be overriden.
- **callback** (Callable) – The function that the user has written and we've wrapped in this class.
- **params** (List[*Argument*]) – Any *Argument* classes that have been attached to this class.
- **parent** (*Base*) – Link to the parent class. Used for getting back to the *Prompt* instance.
- **kwargs** (Dict[str, Any]) – A dictionary of keyword arguments

optional_prefixes should be a list of str that will be used as additional, optional prefixes when performing command completion.

call(*cmd*, *line_parts*)

Called when this class is activated by the promopt

This method calls the underlying callback after parsing any given arguments. It also adds in an optional keyword argument to the callback function; *called_name* is the completed form of the command that user typed. This allows the callback function to change it's behaviour depending on how it was called.

Parameters

- **cmd** (str) – The command that user entered that has been matched to this class.
- **line_parts** (List[str]) – The rest of the user entered line as list

child_completions(*line_parts*, *last_word*)

completions(*cmd*)

Get all matching completions

Generator that yields str when the *cmd* matches one of this command's *names*.

Parameters **cmd** (str) – The command to match against

Return type Iterator[str]

get_completions(*cmd*)

Get all matching completions

Returns a list of matches and whether they are exact or not.

Parameters **cmd** (str) – The command to match against

Return type List[Tuple[str, bool]]

property hasParams

Return type bool

list_children(*p=<built-in function print>*, *deep=False*, *indent=0*, *curr_indent=0*)

Lists all of the child commands attached to this instance.

Parameters

- **p** (Callable) – The functions used for printing. *default*: print

- **deep** (bool) – Continue traversing into all children.
- **indent** (int) – The amount of indentation to add at each level of depth.
- **curr_indent** (int) – The current level of indentation.

property names
Get the names that this command can be called by

Return type List[str]

class promptr.**Command**(*name*, *callback*, *params*, *parent*, ***kwargs*)
Bases: *promptr.Base*

exception promptr.**CommandNotFound**(*cmd*, *args*)
Bases: *promptr.PromptrError*

The promptr command wasn't found

The input from the user didn't match a known state, group, or command.

Parameters

- **cmd** (str) – The command that wasn't found
- **args** (List[str]) – The arguments passed to the command

exception promptr.**ExitState**
Bases: *promptr.PromptrError*

promptr exit state

Raised when the user has requested to leave the current state. This should be caught and the current state should be popped from the stack, up until the last remaining state at which point the program should exit.

class promptr.**Group**(**args*, ***kwargs*)
Bases: *promptr.Base*

call_child(*line_parts*)

command(**args*, ***kwargs*)
Add a command to this group

Parameters **kwargs** (Dict[str, Any]) – Dictionary of keyword arguments

cls Name of the class to use to create the command *default* *Command*

group(**args*, ***kwargs*)
Add a group to this group

Parameters **kwargs** (Dict[str, Any]) – Dictionary of keyword arguments

cls Name of the class to use to create the group *default* *Group*

state(**args*, ***kwargs*)
Add a state to this group

Parameters **kwargs** (Dict[str, Any]) – Dictionary of keyword arguments

cls Name of the class to use to create the state *default* *State*

exception promptr.**NotEnoughArgs**(*cmd*, *params*)
Bases: *promptr.PromptrError*

Too few parameters passed

The input from the user didn't contain enough parameters for the required number of arguments.

Parameters

- **cmd** (str) – The attempted command
- **params** (List[*Argument*]) – The required parameters

class promptr.**Prompt**(*prompt_fmt='{host}{state}# '*, *state_delim='-'*, *state_paren_l='('*, *state_paren_r=')'*, *extra_completions=None*, ***kwargs*)

Bases: object

argument(**args*, ***kwargs*)

command(**args*, ***kwargs*)

Add a command to this promptr instance

Parameters **kwargs** (Dict[str, Any]) – Dictionary of keyword arguments

cls Name of the class to use to create the command *default* *Command*

property current_prompt

property current_state

get_context(*key*, *default=None*)

group(**args*, ***kwargs*)

Add a group to this promptr instance

Parameters **kwargs** (Dict[str, Any]) – Dictionary of keyword arguments

cls Name of the class to use to create the group *default* *Group*

run_prompt(***kwargs*)

run_prompt_loop(***kwargs*)

run_text(*text*, ***kwargs*)

set_context(*key*, *value*)

state(**args*, ***kwargs*)

Add a state to this promptr instance

Parameters **kwargs** (Dict[str, Any]) – Dictionary of keyword arguments

cls Name of the class to use to create the state *default* *State*

class promptr.**PromptrCompleter**(*prompt*, **args*, ***kwargs*)

Bases: prompt_toolkit.completion.base.Completer

get_completions(*document*, *complete_event*)

This should be a generator that yields Completion instances.

If the generation of completions is something expensive (that takes a lot of time), consider wrapping this *Completer* class in a *ThreadedCompleter*. In that case, the completer algorithm runs in a background thread and completions will be displayed as soon as they arrive.

Parameters

- **document** – Document instance.
- **complete_event** – CompleteEvent instance.

exception promptr.**PromptrError**

Bases: RuntimeError

promptr base error

class promptr.**StackItem**(*state*, *context*)
Bases: tuple

property context
Alias for field number 1

property state
Alias for field number 0

class promptr.**State**(**args*, ***kwargs*)
Bases: *promptr.Group*

get_prompt()

property names
Get the names that this command can be called by

on_exit()

1.2 Changelog

1.2.1 0.2.0 - Auto Context

- **Added:** Automatic context handling for arguments
- **Added:** on_exit handler for states
- **Fixed:** AmbiguousCommand incorrectly thrown in some circumstances

1.2.2 0.1.0 - Initial Release

- **Added:** Ability to create a prompt, commands, groups, and states
- **Added:** Arguments can be added with suggestions
- **Added:** Command completion when run as a prompt loop
- **Known Bug:** Arguments are not validated
- **Known Bug:** Command completion fails on partial commands and after arguments

1.3 License

```
The MIT License (MIT)

Copyright (c) 2019 Matt Davis

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
```

(continues on next page)

(continued from previous page)

```
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

promptr, 3

INDEX

A

AmbiguousCommand, 3
Argument (*class in promptr*), 3
argument() (*promptr.Prompt method*), 6

B

Base (*class in promptr*), 3

C

call() (*promptr.Base method*), 4
call_child() (*promptr.Group method*), 5
child_completions() (*promptr.Base method*), 4
Command (*class in promptr*), 5
command() (*promptr.Group method*), 5
command() (*promptr.Prompt method*), 6
CommandNotFound, 5
completions() (*promptr.Argument property*), 3
completions() (*promptr.Base method*), 4
context() (*promptr.StackItem property*), 7
current_prompt() (*promptr.Prompt property*), 6
current_state() (*promptr.Prompt property*), 6

E

ExitState, 5

G

get_completions() (*promptr.Base method*), 4
get_completions() (*promptr.PromptrCompleter method*), 6
get_context() (*promptr.Prompt method*), 6
get_prompt() (*promptr.State method*), 7
Group (*class in promptr*), 5
group() (*promptr.Group method*), 5
group() (*promptr.Prompt method*), 6

H

hasParams() (*promptr.Base property*), 4

L

list_children() (*promptr.Base method*), 4

N

name() (*promptr.Argument property*), 3
names() (*promptr.Base property*), 5
names() (*promptr.State property*), 7
NotEnoughArgs, 5

O

on_exit() (*promptr.State method*), 7

P

Prompt (*class in promptr*), 6
promptr (*module*), 3
PromptrCompleter (*class in promptr*), 6
PromptrError, 6

R

run_prompt() (*promptr.Prompt method*), 6
run_prompt_loop() (*promptr.Prompt method*), 6
run_text() (*promptr.Prompt method*), 6

S

set_context() (*promptr.Prompt method*), 6
StackItem (*class in promptr*), 6
State (*class in promptr*), 7
state() (*promptr.Group method*), 5
state() (*promptr.Prompt method*), 6
state() (*promptr.StackItem property*), 7