SublimeDSL

Release 0.3.2

August 19, 2015

Contents

CHAPTER 1

Contents

1.1 Keymap

DSL for SublimeText's Key Bindings.

Example:

```
from sublimedsl.keymap import *

Keymap(
    bind('backspace')
        .to('run_macro_file', file='res://Packages/Default/Delete Left Right.sublime-macro')
        .when('setting.auto_match_enabled').any().true()
        .also('preceding_text').regex_contains(r'_$')
        .also('following_text').regex_contains(r'^_'),

    bind('super+k', 'super+shift+up')
        .to('new_pane', move=False),

    common_context=[
        context('selector').equal('text.asciidoc')
    ],
    default_match_all=True

).dump()   # nopep8
```

The above code generates:

```
[{
  "keys": [ "backspace" ],
  "command": "run_macro_file",
  "args": { "file": "res://Packages/Default/Delete Left Right.sublime-macro" },
  "context": [
    { "key": "setting.auto_match_enabled", "operator": "equal", "operand": true, "match_all": false
    { "key": "preceding_text", "operator": "regex_contains", "operand": "_$", "match_all": true },
    { "key": "following_text", "operator": "regex_contains", "operand": "^_", "match_all": true },
    { "key": "selector", "operator": "equal", "operand": "text.asciidoc", "match_all": true }
  ]
}, {
  "keys": [ "super+k", "super+shift+up" ],
  "command": "new_pane",
  "args": { "move": false },
  "context": [
```

```
    { "key": "selector", "operator": "equal", "operand": "text.asciidoc", "match_all": true }
  ]
}]
```

See Key Bindings in the SublimeText documentation.

class `Context` (*key*, *parent=None*)

Bases: `object`

Represents a context's condition for a key binding.

Examples::

```
>>> context('preceding_text').regex_match('^[*_-]+')
'{"key": "preceding_text", "operator": "regex_match", "operand": "^[*_-]+"}'
```

```
>>> context('selector').all().equal('text.asciidoc')
'{"key": "selector", "operator": "equal", "operand": "text.asciidoc", "match_all": true}'
```

```
>>> context('selection_empty').true()
'{"key": "selection_empty", "operator": "equal", "operand": true}'
```

See Structure of a Context in the SublimeText documentation.

All operator methods returns the parent *`Binding`*, or self if parent is `None`.

`equal` (*value*)
Specify that the context's value must be equal to the specified value.

`not_equal` (*value*)
Specify that the context's value must *not* be equal to the specified value.

`regex_match` (*pattern*)
Specify that the context's value must match the pattern (full match).

`not_regex_match` (*pattern*)
Specify that the context's value must *not* match the pattern (full match).

`regex_contains` (*pattern*)
Specify that the context's value must contain the pattern (partial match).

`not_regex_contains` (*pattern*)
Specify that the context's value must *not* contain the pattern (partial match).

Parameters

- **`key`** (*str*) – Name of the context whose value you want to query.
- **`parent`** – An object that should be returned by the operator methods.

`all` ()
Require the test to succeed for all selections.

This method sets attribute `match_all` to `True`.

Returns self (for chaining)

Return type *Context*

`any` ()
Require the test to succeed for at least one selection.

This method sets attribute `match_all` to `False`.

Returns self (for chaining)

Return type *Context*

false()
Specify that the context's value must be `False`.

This is shortcut for `equal(False)`.

true()
Specify that the context's value must be `True`.

This is shortcut for `equal(True)`.

class Binding(**keys*)
Bases: `object`

Represents a single key binding.

See Structure of a Key Binding in the SublimeText documentation.

Parameters ***keys** (*str*) – One or more case-sensitive keys. Modifiers can be specified with the + sign.

also(*key*)
Specify context, i.e. condition that must be met.

Parameters **key** (*str*) – Name of the context whose value you want to query.

Returns

Return type *Context*

and_(*key*)
Specify context, i.e. condition that must be met.

Parameters **key** (*str*) – Name of the context whose value you want to query.

Returns

Return type *Context*

to(*command*, ***args*)
Bind the keys to the specified *command* with some *args*.

Parameters

- **command** (*str*) – Name of the ST command (e.g. `insert_snippet`).
- ****args** – Arguments for the command.

Returns self

Return type *Binding*

when(*key*)
Specify context, i.e. condition that must be met.

Parameters **key** (*str*) – Name of the context whose value you want to query.

Returns

Return type *Context*

class Keymap(**bindings*, *default_match_all=None*, *common_context=[]*)
Bases: `object`

Basically a container for key bindings.

Parameters

- ***bindings** (Binding) – The key bindings to be added to this keymap.
- **default_match_all** (*Optional[bool]*) – The default value of `match_all` to be set when context doesn't specify it. See *Context.any()* and *Context.all()*.
- **common_context** (*List[Context]*) – The context that should be added to all bindings.

dump (*fp=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*, ***kwargs*)

Serialize this keymap as a JSON formatted stream to the *fp*.

Parameters

- **fp** – A `.write()`-supporting file-like object to write the generated JSON to (default is `sys.stdout`).
- ****kwargs** – Options to be passed into `json.dumps()`.

extend (**bindings*)

Append the given bindings to this keymap.

Parameters ***bindings** (Binding) – Bindings to be added.

Returns self

Return type *Keymap*

to_json (***kwargs*)

Parameters ****kwargs** – Options to be passed into `json.dumps()`.

Returns A JSON representing this keymap.

Return type str

bind

alias of *Binding*

context

alias of *Context*

CHAPTER 2

Indices and tables

- genindex
- modindex

S