Systhemer Documentation

Release 1

Javier Pollak & Sheheryar Parvas

Jun 15, 2017

Contents

Systhemer, a system theming utility designed for ease of sharing!

Here resides the API reference for both contibutors to systhemer's code and those willing to contribute to the Program Definition database.

CHAPTER 1

External Links:

- Github repo
- User wiki

CHAPTER 2

Progs Modules:

Progs

Systhemer {Blablabla about what progdefs are or something}

Tutorials:

Adding Programs

Additional programs must be stored under the *Progs* directory. A basic program is a subclass of *ProgDef* (defined under *template.py*) and uses utilities from *common.py* for defining functionality.

For naming, the module name must be the binary name suffixed with '.py', the subclass must be the same name as the binary.

Defining programs is explained in the next section

Example

Let's explain using an example module, for the program *example*.

example.py:

```
from .template import ProgDef # parent class
from .common import RuleTree, Rule, RuleVLen, Section

class example(ProgDef):

    def init(self):
        # implementation

    def get_default_path(self):
        # implementation

    def save(self):
        # implementation
```

Developper references:

Template

Templates for configuring program function

Every program definition must inherit from the `ProgDef` class

class `Progs.template.`**`ProgDef`**`(`**args*, ***kwargs*`)`

Template for program definitions.

Program definitions should inherit from this class and should define settings in the config dictionary (see already written definitions for example implementation).

If a program definition needs special handling, you may override the `set()` method.

`do_save``()`

Save the file and run pre/post-save hooks.

`find_rules``(`*key*, *rules*`)`

Return an array of rule objects that contain *key*.

`gen_diff``()`

Generate and print a diff of the change in config file.

`get_config``()`

Returns the config rule tree.

Can be overridden if necessary

`get_default_path``()`

Return the path to use for configuration.

This method must be overridden.

`get_file_buffer``()`

Check if filebuffer exists. If not, one is created.

`get_file_path``()`

Get file path from Settings. If not found there, get from default path.

`get_name``()`

Returns the name of the program. Class name by default.

Can be overridden if necessary

`get_proper_buffer``(`*initial_buffer*, *rule_obj*`)`

Return rule_objs scope portion of initial_buffer.

`init``(`**args*, ***kwargs*`)`

Define rules for configuration.

This method must set the *self.config* variable, which must be of type *common.ConfigElement*. Usage is defined under the *Common* and *Example* sections.

This method must be overridden.

is_installed()
Check if the program is installed on the target system.

Returns a boolean.

This method must be overridden.

mk_backup()
Save the old contents to a backup file

narrow_buffer(*section_obj*, *initial_buffer*, *recur=False*, *recpos=0*, *recdepth=0*, *excludes=None*)
Return a tuple for the start and end of the scope.

Returns a tuple the start and end positions of the scope within the initial_buffer: (startpos, endpos).

Exclude rule format: (depth, startpos, endpos).

pre_init()
Pre-init defaults. If you absolutely have to override __init__ then you must at least include pre_init in __init__.

save()
Save the file.

This method must be overridden.

set(*key*, *value*, *section*)
Set *value* to *key*.

Can be overridden if there are specific needs.

Common

General utilities and common global variables.

class `Progs.common.`**utils**
Namespace for basic utilities.

static get_home_dir()
Get home directory for the user.

static get_setting(*setting*, *default=None*, *critical=True*, *msg=None*)
Get a setting from self.Settings. NOTE: probably should get moved to common module...

static is_excluded(*exclude_rule*, *check_range*)
Check if the given range is within the exclude rule.

Parameters

- **exclude_rule** (*tuple*) – (depth, startpos, endpos)
- **check_range** (*tuple*) – (startpos, endpos)

Returns

- **1** if range is completely in the exclude range
- **2** if range is partially in the exclude range
- **0** if range is not at all in the exclude range

Python Module Index

Index