weblib Documentation

Release 0

Gregory Petukhov

January 12, 2017

Contents

1 Installation 3

2 Testing 5

3 API 7

3.1 weblib.content . . . 7
3.2 weblib.control . . . 7
3.3 weblib.debug . . . 7
3.4 weblib.encoding . . . 7
3.5 weblib.feed . . . 8
3.6 weblib.files . . . 8
3.7 weblib.html . . . 8
3.8 weblib.http . . . 9
3.9 weblib.lock . . . 9
3.10 weblib.logs . . . 9
3.11 weblib.etree . . . 9
3.12 weblib.metric . . . 10
3.13 weblib.parser . . . 10
3.14 weblib.progress . . . 10
3.15 weblib.pwork . . . 10
3.16 weblib.rex . . . 11
3.17 weblib.russian . . . 11
3.18 weblib.system . . . 11
3.19 weblib.text . . . 11
3.20 weblib.user_agent . . . 12
3.21 weblib.watch . . . 12
3.22 weblib.work . . . 12

4 Indices and tables 13

Python Module Index 15

Weblib provides tools to solve typical tasks in web scraping:

- processing HTML
- handling text encodings
- controling repeating and parallel tasks
- parsing RSS/ATOM feeds
- preparing data for HTTP requests
- working with DOM tree
- working with text and numeral data
- list of common user agents
- cross-platform file locking
- operations with files and directories

CHAPTER 1

Installation

```
pip install -U weblib
```

CHAPTER 2

Testing

Install tox package: *pip install tox* Run the command: *tox*

CHAPTER 3

API

3.1 weblib.content

weblib.content.**find_content_blocks**(*tree*, *min_length=None*)

Iterate over content blocks (russian version)

3.2 weblib.control

weblib.control.**repeat**(*func*, *limit=3*, *args=None*, *kwargs=None*, *fatal_exceptions=()*, *valid_exceptions=()*)

Return value of execution *func* function.

In case of error try to execute *func* maximum *limit* times and then raise latest exception.

Example:

```
def download(url):
    return urllib.urlopen(url).read()

data = repeat(download, 3, args=['http://google.com/'])
```

weblib.control.**sleep**(*lower_limit*, *upper_limit*)

Sleep for random number of seconds in interval between *lower_limit* and *upper_limit*

3.3 weblib.debug

weblib.debug.**memory_usage**(*since=0*, *render=True*, *pid=None*)

Return resident memory usage in bytes.

3.4 weblib.encoding

weblib.encoding.**make_str**(*value*, *encoding='utf-8'*, *errors='strict'*)

Normalize unicode/byte string to byte string.

weblib.encoding.**make_unicode**(*value*, *encoding='utf-8'*, *errors='strict'*)

Normalize unicode/byte string to unicode string.

`weblib.encoding.`**`smart_str`**`(`*value*, *encoding='utf-8'*, *errors='strict'*`)`
Normalize unicode/byte string to byte string.

`weblib.encoding.`**`smart_unicode`**`(`*value*, *encoding='utf-8'*, *errors='strict'*`)`
Normalize unicode/byte string to unicode string.

3.5 weblib.feed

`weblib.feed.`**`parse_entry_tags`**`(`*entry*`)`
Return a list of tag objects of the entry

`weblib.feed.`**`parse_feed`**`(`*grab*, *teaser_size=1000*`)`
Extract details of feed fetched with Grab.

Returns dict with keys: * feed * entries

3.6 weblib.files

Miscellaneous utilities which are helpful sometime.

`weblib.files.`**`clear_directory`**`(`*path*`)`
Delete recursively all directories and files in specified directory.

`weblib.files.`**`unique_file`**`(`*path*`)`
Drop non-unique lines in the file. Return number of unique lines.

`weblib.files.`**`unique_host`**`(`*path*`)`
Filter out urls with duplicated hostnames.

3.7 weblib.html

`weblib.html.`**`decode_entities`**`(`*html*`)`
Convert all HTML entities into their unicode representations.

This functions processes following entities:

- &XXX;
- &#XXX;

Example:

```
>>> print html.decode_entities('&rarr;ABC&nbsp;&#82;&copy;')
→ABC R©
```

`weblib.html.`**`escape`**`(`*html*`)`
Returns the given HTML with ampersands, quotes and angle brackets encoded.

`weblib.html.`**`find_base_url`**`(`*html*`)`
Find url of <base> tag.

`weblib.html.`**`find_refresh_url`**`(`*html*`)`
Find value of redirect url from http-equiv refresh meta tag.

3.8 weblib.http

`weblib.http.`**`encode_cookies`**(*items*, *join=True*, *charset='utf-8'*)

Serialize dict or sequence of two-element items into string suitable for sending in Cookie http header.

`weblib.http.`**`normalize_http_values`**(*items*, *charset='utf-8'*, *ignore_classes=None*)

Accept sequence of (key, value) paris or dict and convert each value into bytestring.

Unicode is converted into bytestring using charset of previous response (or utf-8, if no requests were performed)

None is converted into empty string.

If *ignore_classes* is not None and the value is instance of any classes from the *ignore_classes* then the value is not processed and returned as-is.

`weblib.http.`**`normalize_unicode`**(*value*, *charset='utf-8'*)

Convert unicode into byte-string using detected charset (default or from previous response)

By default, charset from previous response is used to encode unicode into byte-string but you can enforce charset with `charset` option

`weblib.http.`**`smart_urlencode`**(*items*, *charset='utf-8'*)

Convert sequence of items into bytestring which could be submitted in POST or GET request.

It differs from `urllib.urlencode` in that it can process unicode and some special values.

`items` could dict or tuple or list.

3.9 weblib.lock

Provide functions for check if file is locked.

`weblib.lock.`**`assert_lock`**(*fname*)

If file is locked then terminate program else lock file.

`weblib.lock.`**`set_lock`**(*fname*)

Try to lock file and write PID.

Return the status of operation.

3.10 weblib.logs

`weblib.logs.`**`default_logging`**(*grab_log='/tmp/grab.log'*, *level=10*, *mode='a'*, *propagate_network_logger=False*, *network_log='/tmp/grab.network.log'*)

Customize logging output to display all log messages except grab network logs.

Redirect grab network logs into file.

3.11 weblib.etree

Functions to process content of lxml nodes.

`weblib.etree.`**`clean_html`**(*html*, *safe_attrs=('src', 'href')*, *input_encoding=None*, *output_encoding=None*, ***kwargs*)

Fix HTML structure and remove non-allowed attributes from all tags.

weblib.etree.**clone_node**(*elem*)
Create clone of Element node.

The resulted clone is not connected ot original DOM tree.

weblib.etree.**disable_links**(*elem*)
Replace all links with span tags and drop href atrributes.

weblib.etree.**drop_node**(*tree*, *xpath*, *keep_content=False*)
Find sub-node by its xpath and remove it.

weblib.etree.**find_node_number**(*node*, *ignore_spaces=False*, *make_int=True*)
Find number in text content of the *node*.

weblib.etree.**get_node_text**(*node*, *smart=False*, *normalize_space=True*)
Extract text content of the *node* and all its descendants.

In smart mode *get_node_text* insert spaces between <tag><another tag> and also ignores content of the script and style tags.

In non-smart mode this func just return text_content() of node with normalized spaces

weblib.etree.**parse_html**(*html*, *encoding='utf-8'*)
Parse html into ElementTree node.

weblib.etree.**render_html**(*node*, *encoding=None*, *make_unicode=None*)
Render Element node.

weblib.etree.**truncate_html**(*html*, *limit*, *encoding='utf-8'*)
Truncate html data to specified length and then fix broken tags.

weblib.etree.**truncate_tail**(*node*, *xpath*)
Find sub-node by its xpath and remove it and all adjacent nodes following after found node.

3.12 weblib.metric

3.13 weblib.parser

3.14 weblib.progress

3.15 weblib.pwork

weblib.pwork.**make_work**(*callback*, *tasks*, *limit*, *ignore_exceptions=True*, *taskq_size=50*)
Run up to "limit" processes, do tasks and yield results.

Parameters

- **callback** – the function that will process single task
- **tasks** – the sequence or iterator or queue of tasks, each task in turn is sequence of arguments, if task is just signle argument it should be wrapped into list or tuple
- **limit** – the maximum number of processes

3.16 weblib.rex

`weblib.rex.`**`extract_rex_list`**`(`*rex*, *body*`)`
Return found matches.

`weblib.rex.`**`normalize_regexp`**`(`*regexp*, *flags=0*`)`
Accept string or compiled regular expression object.

Compile string into regular expression object.

`weblib.rex.`**`rex`**`(`*body*, *regexp*, *flags=0*, *byte=False*, *default=<object object>*`)`
Search *regexp* expression in *body* text.

`weblib.rex.`**`rex_list`**`(`*body*, *rex*, *flags=0*`)`
Return found matches.

`weblib.rex.`**`rex_text`**`(`*body*, *regexp*, *flags=0*, *default=<object object>*`)`
Search *regexp* expression in *body* text and then strip tags in found result.

`weblib.rex.`**`rex_text_list`**`(`*body*, *rex*, *flags=0*`)`
Return found matches with stripped tags.

3.17 weblib.russian

3.18 weblib.system

3.19 weblib.text

Text parsing and processing utilities.

`weblib.text.`**`drop_space`**`(`*text*`)`
Drop all space-chars in the *text*.

`weblib.text.`**`find_number`**`(`*text*, *ignore_spaces=False*, *make_int=True*, *ignore_chars=None*`)`
Find the number in the *text*.

Parameters

- **`text`** – unicode or byte-string text
- **`ignore_spaces`** – if True then groups of digits delimited by spaces are considered as one number

Raises `DataNotFound` if number was not found.

`weblib.text.`**`normalize_space`**`(`*text*, *replace=' '*`)`
Replace sequence of space-chars with one space char.

Also drop leading and trailing space-chars.

`weblib.text.`**`remove_bom`**`(`*text*`)`
Remove BOM-sequence from the start of byte string.

3.20 weblib.user_agent

3.21 weblib.watch

class `weblib.watch.`**`Watcher`**

this class solves two problems with multithreaded programs in Python, (1) a signal might be delivered to any thread (which is just a malfeature) and (2) if the thread that gets the signal is waiting, the signal is ignored (which is a bug).

The watcher is a concurrent process (not thread) that waits for a signal and the process that contains the threads. See Appendix A of The Little Book of Semaphores. http://greenteapress.com/semaphores/

I have only tested this on Linux. I would expect it to work on the Macintosh and not work on Windows.

3.22 weblib.work

`weblib.work.`**`make_work`**(*callback*, *tasks*, *limit*, *ignore_exceptions=True*, *taskq_size=50*)

Run up to "limit" threads, do tasks and yield results.

Parameters

- **`callback`** – the function that will process single task
- **`tasks`** – the sequence or iterator or queue of tasks, each task in turn is sequence of arguments, if task is just signle argument it should be wrapped into list or tuple
- **`limit`** – the maximum number of threads

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

W

weblib.content, 7
weblib.control, 7
weblib.debug, 7
weblib.encoding, 7
weblib.etree, 9
weblib.feed, 8
weblib.files, 8
weblib.html, 8
weblib.http, 9
weblib.lock, 9
weblib.logs, 9
weblib.metric, 10
weblib.parser, 10
weblib.progress, 10
weblib.pwork, 10
weblib.rex, 11
weblib.russian, 11
weblib.system, 11
weblib.text, 11
weblib.user_agent, 12
weblib.watch, 12
weblib.work, 12

Index

A

assert_lock() (in module weblib.lock), 9

C

clean_html() (in module weblib.etree), 9
clear_directory() (in module weblib.files), 8
clone_node() (in module weblib.etree), 9

D

decode_entities() (in module weblib.html), 8
default_logging() (in module weblib.logs), 9
disable_links() (in module weblib.etree), 10
drop_node() (in module weblib.etree), 10
drop_space() (in module weblib.text), 11

E

encode_cookies() (in module weblib.http), 9
escape() (in module weblib.html), 8
extract_rex_list() (in module weblib.rex), 11

F

find_base_url() (in module weblib.html), 8
find_content_blocks() (in module weblib.content), 7
find_node_number() (in module weblib.etree), 10
find_number() (in module weblib.text), 11
find_refresh_url() (in module weblib.html), 8

G

get_node_text() (in module weblib.etree), 10

M

make_str() (in module weblib.encoding), 7
make_unicode() (in module weblib.encoding), 7
make_work() (in module weblib.pwork), 10
make_work() (in module weblib.work), 12
memory_usage() (in module weblib.debug), 7

N

normalize_http_values() (in module weblib.http), 9
normalize_regexp() (in module weblib.rex), 11
normalize_space() (in module weblib.text), 11
normalize_unicode() (in module weblib.http), 9

P

parse_entry_tags() (in module weblib.feed), 8
parse_feed() (in module weblib.feed), 8
parse_html() (in module weblib.etree), 10

R

remove_bom() (in module weblib.text), 11
render_html() (in module weblib.etree), 10
repeat() (in module weblib.control), 7
rex() (in module weblib.rex), 11
rex_list() (in module weblib.rex), 11
rex_text() (in module weblib.rex), 11
rex_text_list() (in module weblib.rex), 11

S

set_lock() (in module weblib.lock), 9
sleep() (in module weblib.control), 7
smart_str() (in module weblib.encoding), 7
smart_unicode() (in module weblib.encoding), 8
smart_urlencode() (in module weblib.http), 9

T

truncate_html() (in module weblib.etree), 10
truncate_tail() (in module weblib.etree), 10

U

unique_file() (in module weblib.files), 8
unique_host() (in module weblib.files), 8

W

Watcher (class in weblib.watch), 12
weblib.content (module), 7
weblib.control (module), 7
weblib.debug (module), 7
weblib.encoding (module), 7
weblib.etree (module), 9

weblib.feed (module), 8
weblib.files (module), 8
weblib.html (module), 8
weblib.http (module), 9
weblib.lock (module), 9
weblib.logs (module), 9
weblib.metric (module), 10
weblib.parser (module), 10
weblib.progress (module), 10
weblib.pwork (module), 10
weblib.rex (module), 11
weblib.russian (module), 11
weblib.system (module), 11
weblib.text (module), 11
weblib.user_agent (module), 12
weblib.watch (module), 12
weblib.work (module), 12