

---

# **ArchiveTeam Seesaw Kit Documentation**

***Release 0.9***

**ArchiveTeam**

August 02, 2015



<b>1</b>	<b>seesaw Package</b>	<b>3</b>
1.1	seesaw Package . . . . .	3
1.2	config Module . . . . .	3
1.3	event Module . . . . .	4
1.4	externalprocess Module . . . . .	4
1.5	item Module . . . . .	5
1.6	pipeline Module . . . . .	6
1.7	project Module . . . . .	7
1.8	runner Module . . . . .	7
1.9	task Module . . . . .	8
1.10	tracker Module . . . . .	9
1.11	util Module . . . . .	10
1.12	warrior Module . . . . .	10
1.13	web Module . . . . .	12
1.14	web_util Module . . . . .	13
<b>2</b>	<b>Indices and tables</b>	<b>15</b>
	<b>Python Module Index</b>	<b>17</b>



Contents:



---

## seesaw Package

---

### 1.1 seesaw Package

ArchiveTeam seesaw kit

### 1.2 config Module

Configuration value manipulation.

```
class seesaw.config.ConfigInterpolation(s, c)
    Bases: object
```

```
    realize(item)
```

```
class seesaw.config.ConfigValue(name, title='', description='', default=None, editable=True, ad-
                                vanced=True)
    Bases: object
```

Configuration value validator.

The collection methods are useful for providing user configurable settings at run time. For example, when a pipeline file is executed by the warrior, the additional config values are presented in the warrior configuration panel.

```
    check_value(value)
```

```
    collector = None
```

```
    convert_value(value)
```

```
    is_valid()
```

```
    realize(dummy)
```

```
    set_value(value)
```

```
    classmethod start_collecting()
```

```
    classmethod stop_collecting()
```

```
class seesaw.config.NumberConfigValue(*args, **kwargs)
    Bases: seesaw.config.ConfigValue
```

```
    check_value(value)
```

```
    convert_value(value)
```

**class** seesaw.config.**StringConfigValue** (\*args, \*\*kwargs)

Bases: *seesaw.config.ConfigValue*

**check\_value** (value)

seesaw.config.**realize** (v, item=None)

Makes objects contain concrete values from an item.

A silly example:

```
class AddExpression(object):
    def realize(self, item):
        return item['x'] + item['y']

pipeline = Pipeline(ComputeMath(AddExpression()))
```

In the example, we want to compute an addition expression. The values are defined in the Item.

## 1.3 event Module

Actor model.

**class** seesaw.event.**Event**

Bases: object

Lightweight event system.

Example:

```
my_event_system = Event()
my_event_system = my_listener_callback_function
my_event_system(my_event_data)
```

**fire** (\*args, \*\*kwargs)

**getHandlerCount** ()

**handle** (handler)

**unhandle** (handler)

## 1.4 externalprocess Module

Running subprocesses asynchronously.

**class** seesaw.externalprocess.**AsyncPopen** (\*args, \*\*kwargs)

Bases: object

Asynchronous version of subprocess.Popen.

Deprecated.

**classmethod ignore\_sigint** ()

**run** ()

**class** seesaw.externalprocess.**AsyncPopen2** (\*args, \*\*kwargs)

Bases: object

Adapter for the legacy AsyncPopen



**run()**

**stdin**

```
class seesaw.externalprocess.CurlUpload(target, filename, connect_timeout='60',
                                         speed_limit='1', speed_time='900',
                                         max_tries=None)
```

Bases: *seesaw.externalprocess.ExternalProcess*

Upload with Curl process runner.

```
class seesaw.externalprocess.ExternalProcess(name, args, max_tries=1, retry_delay=30,
                                              accept_on_exit_code=None,
                                              retry_on_exit_code=None, env=None)
```

Bases: *seesaw.task.Task*

External subprocess runner.

**enqueue** (*item*)

**handle\_process\_error** (*exit\_code*, *item*)

**handle\_process\_result** (*exit\_code*, *item*)

**on\_subprocess\_end** (*item*, *returncode*)

**on\_subprocess\_stdout** (*pipe*, *item*, *data*)

**process** (*item*)

**stdin\_data** (*item*)

```
class seesaw.externalprocess.RsyncUpload(target, files, target_source_path='./', bwlimit='0',
                                          max_tries=None, extra_args=None)
```

Bases: *seesaw.externalprocess.ExternalProcess*

Upload with Rsync process runner.

**stdin\_data** (*item*)

```
class seesaw.externalprocess.WgetDownload(args, max_tries=1, accept_on_exit_code=None,
                                           retry_on_exit_code=None, env=None,
                                           stdin_data_function=None)
```

Bases: *seesaw.externalprocess.ExternalProcess*

Download with Wget process runner.

**stdin\_data** (*item*)

**seesaw.externalprocess.cleanup()**

## 1.5 item Module

Managing work units.

```
class seesaw.item.Item(pipeline, item_id, item_number, properties=None, keep_data=False, prepare_data_directory=True)
```

Bases: *object*

A thing, or work unit, that needs to be downloaded.

It has properties that are filled by the Task.

An Item behaves like a mutable mapping.

**Note:** State belonging to a item should be stored on the actual item itself. That is, do not store variables onto a Task unless you know what you are doing.

---

**class TaskStatus**

Bases: object

**completed** = 'completed'

**failed** = 'failed'

**running** = 'running'

Item.**cancel**()

Item.**clear\_data\_directory**()

Item.**complete**()

Item.**description**()

Item.**fail**()

Item.**get**(key)

Item.**log\_error**(task, \*args)

Item.**log\_output**(data, full\_line=True)

Item.**prepare\_data\_directory**()

Item.**set\_task\_status**(task, status)

**class seesaw.item.ItemInterpolation(s)**

Bases: object

Formats a string using the percent operator during *realize*().

**realize**(item)

**class seesaw.item.ItemValue(key)**

Bases: object

Get an item's value during *realize*().

**fill**(item, value)

**realize**(item)

## 1.6 pipeline Module

**class seesaw.pipeline.Pipeline(\*tasks)**

Bases: object

The sequence of steps that complete a Task.

Your pipeline will probably be something like this:

- 1.Request an assignment from the tracker.
- 2.Run Wget to download the file.
- 3.Upload the downloaded file with rsync.
- 4.Tell the tracker that the assignment is done.

**add\_task**(task)

```
cancel_items()
enqueue(item)
ui_task_list()
```

## 1.7 project Module

Project information.

```
class seesaw.project.Project(title=None, project_html=None, utc_deadline=None)
    Bases: object
```

Briefly describes a project metadata.

This class defines the title of the project, a short description with an optional project logo and an optional deadline. The information will be shown in the web interface when the project is running.

```
data_for_json()
```

## 1.8 runner Module

Pipeline execution.

```
class seesaw.runner.Runner(stop_file=None, concurrent_items=1, max_items=None,
                           keep_data=False)
    Bases: object
```

Executes and manages the lifetime of Pipeline instances.

```
add_items()
check_stop_file()
is_active()
keep_running()
set_current_pipeline(pipeline)
should_stop()
start()
stop_file_changed()
stop_file_mtime()
stop_gracefully()
```

```
class seesaw.runner.SimpleRunner(pipeline, stop_file=None, concurrent_items=1,
                                  max_items=None, keep_data=False)
    Bases: seesaw.runner.Runner
```

Executes a single class:Pipeline instance.

```
forced_stop()
start()
```

## 1.9 task Module

Managing steps in a work unit.

**class** seesaw.task.**ConditionalTask** (*condition\_function*, *inner\_task*)  
Bases: *seesaw.task.Task*

Runs a task optionally.

**enqueue** (*item*)

**fill\_ui\_task\_list** (*task\_list*)

**class** seesaw.task.**LimitConcurrent** (*concurrency*, *inner\_task*)  
Bases: *seesaw.task.Task*

Restricts the number of tasks of the same type that can be run at once.

**enqueue** (*item*)

**fill\_ui\_task\_list** (*task\_list*)

**class** seesaw.task.**PrintItem**  
Bases: *seesaw.task.SimpleTask*

Output the name of the Item.

**process** (*item*)

**class** seesaw.task.**SetItemKey** (*key*, *value*)  
Bases: *seesaw.task.SimpleTask*

Set a value onto a task.

**process** (*item*)

**class** seesaw.task.**SimpleTask** (*name*)  
Bases: *seesaw.task.Task*

A subclassable *Task* that should do one small thing well.

Example:

```
class MyTask(SimpleTask):
    def process(self, item):
        item['my_message'] = 'hello world!'
```

**enqueue** (*item*)

**process** (*item*)

**class** seesaw.task.**Task** (*name*)  
Bases: *object*

A step in the download process of an Item.

**complete\_item** (*item*)

**fail\_item** (*item*)

**fill\_ui\_task\_list** (*task\_list*)

**start\_item** (*item*)

**task\_cwd** (\*args, \*\*kws)

## 1.10 tracker Module

Contacting the work unit server.

A Tracker refers to the Universal Tracker (<https://github.com/ArchiveTeam/universal-tracker>).

```
class seesaw.tracker.GetItemFromTracker (tracker_url, downloader, version=None)
    Bases: seesaw.tracker.TrackerRequest

    Get a single work unit information from the Tracker.

    data (item)

    process_body (body, item)

class seesaw.tracker.PrepareStatsForTracker (defaults=None, file_groups=None,
                                              id_function=None)
    Bases: seesaw.task.SimpleTask

    Apply statistical values on the item.

    process (item)

class seesaw.tracker.SendDoneToTracker (tracker_url, stats)
    Bases: seesaw.tracker.TrackerRequest

    Inform the Tracker the work unit has been completed.

    data (item)

    process_body (body, item)

class seesaw.tracker.TrackerRequest (name, tracker_url, tracker_command,
                                     may_be_canceled=False)
    Bases: seesaw.task.Task

    Represents a request to a Tracker.

    DEFAULT_RETRY_DELAY = 60

    data (item)

    enqueue (item)

    handle_response (item, response)

    increment_retry_delay (max_delay=300)

    process_body (body, item)

    reset_retry_delay ()

    schedule_retry (item, message='')

    send_request (item)

class seesaw.tracker.UploadWithTracker (tracker_url, downloader, files, version=None,
                                         rsync_target_source_path='.', rsync_bwlimit='0',
                                         rsync_extra_args=[], curl_connect_timeout='60',
                                         curl_speed_limit='1', curl_speed_time='900')
    Bases: seesaw.tracker.TrackerRequest

    Upload work unit results.

    One of the inner task is used depending on the Tracker's response to where to upload:

    •RsyncUpload
```

- `CurlUpload`

**data** (*item*)

**process\_body** (*body, item*)

## 1.11 util Module

Miscellaneous functions.

`seesaw.util.find_executable` (*name, version, paths, version\_arg='-V'*)  
Returns the path of a matching executable.

**See also:**

`test_executable()`

`seesaw.util.test_executable` (*name, version, path, version\_arg='-V'*)  
Try to run an executable and check its version.

`seesaw.util.unique_id_str()`  
Returns a unique string suitable for IDs.

## 1.12 warrior Module

The warrior server.

The warrior phones home to Warrior HQ (<https://github.com/ArchiveTeam/warrior-hq>).

**class** `seesaw.warrior.BandwidthMonitor` (*device*)  
Bases: `object`

Extracts the bandwidth usage from the system stats.

**current\_stats** ()

**devre** = `<_sre.SRE_Pattern object>`

**update** ()

**class** `seesaw.warrior.ConfigManager` (*config\_file*)  
Bases: `object`

Manages the configuration.

**add** (*config\_value*)

**all\_valid** ()

**editable\_values** ()

**load** ()

**remove** (*name*)

**save** ()

**set\_value** (*name, value*)

```
class seesaw.warrior.Warrior (projects_dir, data_dir, warrior_hq_url, real_shutdown=False,  
                             keep_data=False)
```

Bases: object

The warrior god object.

```
class Status
```

Bases: object

```
INVALID_SETTINGS = 'INVALID_SETTINGS'
```

```
NO_PROJECT = 'NO_PROJECT'
```

```
REBOOTING = 'REBOOTING'
```

```
RESTARTING_PROJECT = 'RESTARTING_PROJECT'
```

```
RUNNING_PROJECT = 'RUNNING_PROJECT'
```

```
SHUTTING_DOWN = 'SHUTTING_DOWN'
```

```
STARTING_PROJECT = 'STARTING_PROJECT'
```

```
STOPPING_PROJECT = 'STOPPING_PROJECT'
```

```
SWITCHING_PROJECT = 'SWITCHING_PROJECT'
```

```
UNINITIALIZED = 'UNINITIALIZED'
```

```
Warrior.bandwidth_stats ()
```

```
Warrior.check_project_has_update (*args, **kwargs)
```

```
Warrior.clone_project (project_name, project_path)
```

```
Warrior.collect_install_output (data)
```

```
Warrior.find_lat_lng ()
```

```
Warrior.fire_status ()
```

```
Warrior.forced_reboot ()
```

```
Warrior.forced_stop ()
```

```
Warrior.handle_lat_lng (response)
```

```
Warrior.handle_runner_finish (runner)
```

```
Warrior.install_project (*args, **kwargs)
```

```
Warrior.keep_running ()
```

```
Warrior.load_pipeline (pipeline_path, context)
```

```
Warrior.max_age_reached ()
```

```
Warrior.reboot_gracefully ()
```

```
Warrior.schedule_forced_reboot ()
```

```
Warrior.select_project (*args, **kwargs)
```

```
Warrior.start ()
```

```
Warrior.start_selected_project (*args, **kwargs)
```

```
Warrior.stop_gracefully ()
```

```
Warrior.update_project (*args, **kwargs)
```

```
Warrior.update_warrior_hq(*args, **kwargs)

Warrior.warrior_status()
```

## 1.13 web Module

The warrior web interface.

```
class seesaw.web.ApiHandler(application, request, **kwargs)
    Bases: tornado.web.RequestHandler

    Processes API requests.

    get(command)

    get_template_path()

    initialize(warrior=None, runner=None)

    post(command)

class seesaw.web.IndexHandler(application, request, **kwargs)
    Bases: tornado.web.RequestHandler

    Shows the index.html.

    get()

class seesaw.web.ItemMonitor(item)
    Bases: object

    Pushes item states and information to the client.

    handle_item_cancel(item)

    handle_item_complete(item)

    handle_item_fail(item)

    handle_item_output(item, data)

    handle_item_property(item, key, new_value, old_value)

    handle_item_task_status(item, task, new_status, old_status)

    item_for_broadcast()

    item_status()

class seesaw.web.SeesawConnection(session)
    Bases: sockjs.tornado.conn.SockJSConnection

    A WebSocket server that communicates the state of the warrior.

    classmethod broadcast(event, message)

    classmethod broadcast_bandwidth()

    classmethod broadcast_project_refresh()

    classmethod broadcast_projects()

    classmethod broadcast_timestamp()

    clients = set([])
```



```

emit (event_name, message)
    tornadoio to sockjs adapter.

classmethod handle_broadcast_message (warrior, message)

classmethod handle_finish_item (runner, pipeline, item)

classmethod handle_project_installation_failed (warrior, project, output)

classmethod handle_project_installed (warrior, project, output)

classmethod handle_project_installing (warrior, project)

classmethod handle_project_refresh (warrior, project, runner)

classmethod handle_project_selected (warrior, project)

classmethod handle_projects_loaded (warrior, projects)

classmethod handle_runner_status (runner, status)

classmethod handle_start_item (runner, pipeline, item)

classmethod handle_warrior_status (warrior, new_status)

instance_id = '23216-0.881194'

item_monitors = {}

on_close ()

on_message (message)

on_open (info)

project = None

runner = None

warrior = None

seesaw.web.hash_string (text)
    Generate a digest for broadcast message.

seesaw.web.start_runner_server (project, runner, bind_address='localhost', port_number=8001,
                                http_username=None, http_password=None)
    Starts a web interface for a manually run pipeline.

    Unlike start_warrior_server(), this UI does not contain an configuration or project management panel.

seesaw.web.start_warrior_server (warrior, bind_address='localhost', port_number=8001,
                                http_username=None, http_password=None)
    Starts the warrior web interface.

```

## 1.14 web\_util Module

```

class seesaw.web_util.AuthenticatedApplication (*args, **kwargs)
    Bases: tornado.web.Application

class seesaw.web_util.AuthenticationErrorHandler (application, request, **kwargs)
    Bases: tornado.web.RequestHandler

    initialize (realm='Restricted')

    prepare ()

```



---

## Indices and tables

---

- `genindex`
- `modindex`
- `search`



## S

seesaw.\_\_init\_\_, 3  
seesaw.config, 3  
seesaw.event, 4  
seesaw.externalprocess, 4  
seesaw.item, 5  
seesaw.pipeline, 6  
seesaw.project, 7  
seesaw.runner, 7  
seesaw.task, 8  
seesaw.tracker, 9  
seesaw.util, 10  
seesaw.warrior, 10  
seesaw.web, 12  
seesaw.web\_util, 13



## A

add() (seesaw.warrior.ConfigManager method), 10  
 add\_items() (seesaw.runner.Runner method), 7  
 add\_task() (seesaw.pipeline.Pipeline method), 6  
 all\_valid() (seesaw.warrior.ConfigManager method), 10  
 ApiHandler (class in seesaw.web), 12  
 AsyncPopen (class in seesaw.externalprocess), 4  
 AsyncPopen2 (class in seesaw.externalprocess), 4  
 AuthenticatedApplication (class in seesaw.web\_util), 13  
 AuthenticationErrorHandler (class in seesaw.web\_util), 13

## B

bandwidth\_stats() (seesaw.warrior.Warrior method), 11  
 BandwidthMonitor (class in seesaw.warrior), 10  
 broadcast() (seesaw.web.SeesawConnection class method), 12  
 broadcast\_bandwidth() (seesaw.web.SeesawConnection class method), 12  
 broadcast\_project\_refresh() (seesaw.web.SeesawConnection class method), 12  
 broadcast\_projects() (seesaw.web.SeesawConnection class method), 12  
 broadcast\_timestamp() (seesaw.web.SeesawConnection class method), 12

## C

cancel() (seesaw.item.Item method), 6  
 cancel\_items() (seesaw.pipeline.Pipeline method), 7  
 check\_project\_has\_update() (seesaw.warrior.Warrior method), 11  
 check\_stop\_file() (seesaw.runner.Runner method), 7  
 check\_value() (seesaw.config.ConfigValue method), 3  
 check\_value() (seesaw.config.NumberConfigValue method), 3  
 check\_value() (seesaw.config.StringConfigValue method), 4  
 cleanup() (in module seesaw.externalprocess), 5  
 clear\_data\_directory() (seesaw.item.Item method), 6

clients (seesaw.web.SeesawConnection attribute), 12  
 clone\_project() (seesaw.warrior.Warrior method), 11  
 collect\_install\_output() (seesaw.warrior.Warrior method), 11  
 collector (seesaw.config.ConfigValue attribute), 3  
 complete() (seesaw.item.Item method), 6  
 complete\_item() (seesaw.task.Task method), 8  
 completed (seesaw.item.Item.TaskStatus attribute), 6  
 ConditionalTask (class in seesaw.task), 8  
 ConfigInterpolation (class in seesaw.config), 3  
 ConfigManager (class in seesaw.warrior), 10  
 ConfigValue (class in seesaw.config), 3  
 convert\_value() (seesaw.config.ConfigValue method), 3  
 convert\_value() (seesaw.config.NumberConfigValue method), 3  
 CurlUpload (class in seesaw.externalprocess), 5  
 current\_stats() (seesaw.warrior.BandwidthMonitor method), 10

## D

data() (seesaw.tracker.GetItemFromTracker method), 9  
 data() (seesaw.tracker.SendDoneToTracker method), 9  
 data() (seesaw.tracker.TrackerRequest method), 9  
 data() (seesaw.tracker.UploadWithTracker method), 10  
 data\_for\_json() (seesaw.project.Project method), 7  
 DEFAULT\_RETRY\_DELAY (seesaw.tracker.TrackerRequest attribute), 9  
 description() (seesaw.item.Item method), 6  
 devre (seesaw.warrior.BandwidthMonitor attribute), 10

## E

editable\_values() (seesaw.warrior.ConfigManager method), 10  
 emit() (seesaw.web.SeesawConnection method), 12  
 enqueue() (seesaw.externalprocess.ExternalProcess method), 5  
 enqueue() (seesaw.pipeline.Pipeline method), 7  
 enqueue() (seesaw.task.ConditionalTask method), 8  
 enqueue() (seesaw.task.LimitConcurrent method), 8  
 enqueue() (seesaw.task.SimpleTask method), 8

enqueue() (seesaw.tracker.TrackerRequest method), 9  
 Event (class in seesaw.event), 4  
 ExternalProcess (class in seesaw.externalprocess), 5

## F

fail() (seesaw.item.Item method), 6  
 fail\_item() (seesaw.task.Task method), 8  
 failed (seesaw.item.Item.TaskStatus attribute), 6  
 fill() (seesaw.item.ItemValue method), 6  
 fill\_ui\_task\_list() (seesaw.task.ConditionalTask method), 8  
 fill\_ui\_task\_list() (seesaw.task.LimitConcurrent method), 8  
 fill\_ui\_task\_list() (seesaw.task.Task method), 8  
 find\_executable() (in module seesaw.util), 10  
 find\_lat\_lng() (seesaw.warrior.Warrior method), 11  
 fire() (seesaw.event.Event method), 4  
 fire\_status() (seesaw.warrior.Warrior method), 11  
 forced\_reboot() (seesaw.warrior.Warrior method), 11  
 forced\_stop() (seesaw.runner.SimpleRunner method), 7  
 forced\_stop() (seesaw.warrior.Warrior method), 11

## G

get() (seesaw.item.Item method), 6  
 get() (seesaw.web.ApiHandler method), 12  
 get() (seesaw.web.IndexHandler method), 12  
 get\_template\_path() (seesaw.web.ApiHandler method), 12  
 getHandlerCount() (seesaw.event.Event method), 4  
 GetItemFromTracker (class in seesaw.tracker), 9

## H

handle() (seesaw.event.Event method), 4  
 handle\_broadcast\_message() (seesaw.web.SeesawConnection class method), 13  
 handle\_finish\_item() (seesaw.web.SeesawConnection class method), 13  
 handle\_item\_cancel() (seesaw.web.ItemMonitor method), 12  
 handle\_item\_complete() (seesaw.web.ItemMonitor method), 12  
 handle\_item\_fail() (seesaw.web.ItemMonitor method), 12  
 handle\_item\_output() (seesaw.web.ItemMonitor method), 12  
 handle\_item\_property() (seesaw.web.ItemMonitor method), 12  
 handle\_item\_task\_status() (seesaw.web.ItemMonitor method), 12  
 handle\_lat\_lng() (seesaw.warrior.Warrior method), 11  
 handle\_process\_error() (seesaw.externalprocess.ExternalProcess method), 5

handle\_process\_result() (seesaw.externalprocess.ExternalProcess method), 5  
 handle\_project\_installation\_failed() (seesaw.web.SeesawConnection class method), 13  
 handle\_project\_installed() (seesaw.web.SeesawConnection class method), 13  
 handle\_project\_installing() (seesaw.web.SeesawConnection class method), 13  
 handle\_project\_refresh() (seesaw.web.SeesawConnection class method), 13  
 handle\_project\_selected() (seesaw.web.SeesawConnection class method), 13  
 handle\_projects\_loaded() (seesaw.web.SeesawConnection class method), 13  
 handle\_response() (seesaw.tracker.TrackerRequest method), 9  
 handle\_runner\_finish() (seesaw.warrior.Warrior method), 11  
 handle\_runner\_status() (seesaw.web.SeesawConnection class method), 13  
 handle\_start\_item() (seesaw.web.SeesawConnection class method), 13  
 handle\_warrior\_status() (seesaw.web.SeesawConnection class method), 13  
 hash\_string() (in module seesaw.web), 13

## I

ignore\_sigint() (seesaw.externalprocess.AsyncPopen class method), 4  
 increment\_retry\_delay() (seesaw.tracker.TrackerRequest method), 9  
 IndexHandler (class in seesaw.web), 12  
 initialize() (seesaw.web.ApiHandler method), 12  
 initialize() (seesaw.web\_util.AuthenticationErrorHandler method), 13  
 install\_project() (seesaw.warrior.Warrior method), 11  
 instance\_id (seesaw.web.SeesawConnection attribute), 13  
 INVALID\_SETTINGS (seesaw.warrior.Warrior.Status attribute), 11  
 is\_active() (seesaw.runner.Runner method), 7  
 is\_valid() (seesaw.config.ConfigValue method), 3  
 Item (class in seesaw.item), 5  
 Item.TaskStatus (class in seesaw.item), 6  
 item\_for\_broadcast() (seesaw.web.ItemMonitor method), 12  
 item\_monitors (seesaw.web.SeesawConnection attribute), 13  
 item\_status() (seesaw.web.ItemMonitor method), 12



ItemInterpolation (class in seesaw.item), 6

ItemMonitor (class in seesaw.web), 12

ItemValue (class in seesaw.item), 6

## K

keep\_running() (seesaw.runner.Runner method), 7

keep\_running() (seesaw.warrior.Warrior method), 11

## L

LimitConcurrent (class in seesaw.task), 8

load() (seesaw.warrior.ConfigManager method), 10

load\_pipeline() (seesaw.warrior.Warrior method), 11

log\_error() (seesaw.item.Item method), 6

log\_output() (seesaw.item.Item method), 6

## M

max\_age\_reached() (seesaw.warrior.Warrior method), 11

## N

NO\_PROJECT (seesaw.warrior.Warrior.Status attribute), 11

NumberConfigValue (class in seesaw.config), 3

## O

on\_close() (seesaw.web.SeesawConnection method), 13

on\_message() (seesaw.web.SeesawConnection method), 13

on\_open() (seesaw.web.SeesawConnection method), 13

on\_subprocess\_end() (seesaw.externalprocess.ExternalProcess method), 5

on\_subprocess\_stdout() (seesaw.externalprocess.ExternalProcess method), 5

## P

Pipeline (class in seesaw.pipeline), 6

post() (seesaw.web.ApiHandler method), 12

prepare() (seesaw.web\_util.AuthenticationErrorHandler method), 13

prepare\_data\_directory() (seesaw.item.Item method), 6

PrepareStatsForTracker (class in seesaw.tracker), 9

PrintItem (class in seesaw.task), 8

process() (seesaw.externalprocess.ExternalProcess method), 5

process() (seesaw.task.PrintItem method), 8

process() (seesaw.task.SetItemKey method), 8

process() (seesaw.task.SimpleTask method), 8

process() (seesaw.tracker.PrepareStatsForTracker method), 9

process\_body() (seesaw.tracker.GetItemFromTracker method), 9

process\_body() (seesaw.tracker.SendDoneToTracker method), 9

process\_body() (seesaw.tracker.TrackerRequest method), 9

process\_body() (seesaw.tracker.UploadWithTracker method), 10

Project (class in seesaw.project), 7

project (seesaw.web.SeesawConnection attribute), 13

## R

realize() (in module seesaw.config), 4

realize() (seesaw.config.ConfigInterpolation method), 3

realize() (seesaw.config.ConfigValue method), 3

realize() (seesaw.item.ItemInterpolation method), 6

realize() (seesaw.item.ItemValue method), 6

reboot\_gracefully() (seesaw.warrior.Warrior method), 11

REBOOTING (seesaw.warrior.Warrior.Status attribute), 11

remove() (seesaw.warrior.ConfigManager method), 10

reset\_retry\_delay() (seesaw.tracker.TrackerRequest method), 9

RESTARTING\_PROJECT (seesaw.warrior.Warrior.Status attribute), 11

RsyncUpload (class in seesaw.externalprocess), 5

run() (seesaw.externalprocess.AsyncPopen method), 4

run() (seesaw.externalprocess.AsyncPopen2 method), 4

Runner (class in seesaw.runner), 7

runner (seesaw.web.SeesawConnection attribute), 13

running (seesaw.item.Item.TaskStatus attribute), 6

RUNNING\_PROJECT (seesaw.warrior.Warrior.Status attribute), 11

## S

save() (seesaw.warrior.ConfigManager method), 10

schedule\_forced\_reboot() (seesaw.warrior.Warrior method), 11

schedule\_retry() (seesaw.tracker.TrackerRequest method), 9

seesaw.\_\_init\_\_ (module), 3

seesaw.config (module), 3

seesaw.event (module), 4

seesaw.externalprocess (module), 4

seesaw.item (module), 5

seesaw.pipeline (module), 6

seesaw.project (module), 7

seesaw.runner (module), 7

seesaw.task (module), 8

seesaw.tracker (module), 9

seesaw.util (module), 10

seesaw.warrior (module), 10

seesaw.web (module), 12

seesaw.web\_util (module), 13

SeesawConnection (class in seesaw.web), 12

select\_project() (seesaw.warrior.Warrior method), 11

send\_request() (seesaw.tracker.TrackerRequest method), 9  
 SendDoneToTracker (class in seesaw.tracker), 9  
 set\_current\_pipeline() (seesaw.runner.Runner method), 7  
 set\_task\_status() (seesaw.item.Item method), 6  
 set\_value() (seesaw.config.ConfigValue method), 3  
 set\_value() (seesaw.warrior.ConfigManager method), 10  
 SetItemKey (class in seesaw.task), 8  
 should\_stop() (seesaw.runner.Runner method), 7  
 SHUTTING\_DOWN (seesaw.warrior.Warrior.Status attribute), 11  
 SimpleRunner (class in seesaw.runner), 7  
 SimpleTask (class in seesaw.task), 8  
 start() (seesaw.runner.Runner method), 7  
 start() (seesaw.runner.SimpleRunner method), 7  
 start() (seesaw.warrior.Warrior method), 11  
 start\_collecting() (seesaw.config.ConfigValue class method), 3  
 start\_item() (seesaw.task.Task method), 8  
 start\_runner\_server() (in module seesaw.web), 13  
 start\_selected\_project() (seesaw.warrior.Warrior method), 11  
 start\_warrior\_server() (in module seesaw.web), 13  
 STARTING\_PROJECT (seesaw.warrior.Warrior.Status attribute), 11  
 stdin (seesaw.externalprocess.AsyncPopen2 attribute), 5  
 stdin\_data() (seesaw.externalprocess.ExternalProcess method), 5  
 stdin\_data() (seesaw.externalprocess.RsyncUpload method), 5  
 stdin\_data() (seesaw.externalprocess.WgetDownload method), 5  
 stop\_collecting() (seesaw.config.ConfigValue class method), 3  
 stop\_file\_changed() (seesaw.runner.Runner method), 7  
 stop\_file\_mtime() (seesaw.runner.Runner method), 7  
 stop\_gracefully() (seesaw.runner.Runner method), 7  
 stop\_gracefully() (seesaw.warrior.Warrior method), 11  
 STOPPING\_PROJECT (seesaw.warrior.Warrior.Status attribute), 11  
 StringConfigValue (class in seesaw.config), 3  
 SWITCHING\_PROJECT (seesaw.warrior.Warrior.Status attribute), 11

## T

Task (class in seesaw.task), 8  
 task\_cwd() (seesaw.task.Task method), 8  
 test\_executable() (in module seesaw.util), 10  
 TrackerRequest (class in seesaw.tracker), 9

## U

ui\_task\_list() (seesaw.pipeline.Pipeline method), 7  
 unhandle() (seesaw.event.Event method), 4

UNINITIALIZED (seesaw.warrior.Warrior.Status attribute), 11  
 unique\_id\_str() (in module seesaw.util), 10  
 update() (seesaw.warrior.BandwidthMonitor method), 10  
 update\_project() (seesaw.warrior.Warrior method), 11  
 update\_warrior\_hq() (seesaw.warrior.Warrior method), 11  
 UploadWithTracker (class in seesaw.tracker), 9

## W

Warrior (class in seesaw.warrior), 10  
 warrior (seesaw.web.SeesawConnection attribute), 13  
 Warrior.Status (class in seesaw.warrior), 11  
 warrior\_status() (seesaw.warrior.Warrior method), 12  
 WgetDownload (class in seesaw.externalprocess), 5