
Lawwenda

Release 0.4.219

Author name not set

Sep 13, 2026

CONTENTS

1	License	3
2	Up-To-Date?	5
3	Dependencies	7
4	Introduction	9
4.1	First Steps	9
5	Appendix: Installation	11
6	Command Line Interface Reference	13
7	API Reference	15
7.1	lawwenda package	15
	Python Module Index	81
	Index	83

Lawwenda gives you web-based access to a directory on your filesystem, so users can browse and download your files from any modern browser or WebDAV client. It runs as a WSGI application, the standard for hosting Python web apps, supported by most common web servers.

Key features:

- A clean, browser-based interface for browsing, viewing, and downloading files and folders - much like a desktop file manager.
- Optional write access (upload, rename, move, delete) and optional password protection.
- Rich file metadata - comments, tags, ratings, and geolocation - stored via filesystem extended attributes.
- Flexible “shares”: define one or several, each with its own directory, password, exclude rules, and expiration date.
- WebDAV support for connecting via any compatible file manager.
- A well-documented Python API for building custom plugins.
- Built with security as a core design principle.

LICENSE

Lawwenda is distributed under the terms of the AGPL 3 license. This also affects all included files without a license header (non-source files like images), unless they are explicitly mentioned as third-party content. Read the ‘Dependencies’ section for included third-party stuff.

UP-TO-DATE?

Are you currently reading from another source than the homepage? If you are in doubt whether your package is up-to-date, you should visit the project homepage and check that. You are currently reading the documentation for version 0.4.219.

DEPENDENCIES

Lawwenda makes use of some third-party parts.



Required: **Python 3.7**



Required: **Python package Pillow** `~= 12.3`



Required: **Python package werkzeug** `~= 3.1`



Recommended: **GNU/Linux**



Included: **background image** (License: <https://creativecommons.org/licenses/by-sa/3.0>; from [here](#))



Included: **font ‘Inconsolata’** (for websites; by Raph Levien; License: OFL; from [here](#))



Included: **font ‘Khula’** (for websites; by Erin McLaughlin; License: OFL; from [here](#))



Included: **font ‘Symbola’** (for logo symbol; License: free for use; from [here](#))

INTRODUCTION

Please read how to make Lawwenda ready for the first steps in [Appendix: Installation](#).

4.1 First Steps

In order to make any use of Lawwenda, it is required to create at least one file share. This can be done by executing this on a terminal:

```
lawwenda_cli add_share myshare /path/that/i/want/to/share
```

This assumes `/path/that/i/want/to/share` to be the directory you want to share. `myshare` is the share name and can be chosen freely. It will ask you to choose a password (just leave empty for no password).

Then you need a web server that hosts your Lawwenda installation. If you have skipped this part of [Appendix: Installation](#), you can get one by executing this:

```
lawwenda_cli run_tiny_server
```

This will print the root address of a tiny local web server that hosts your Lawwenda installation for the moment. It will not answer anything useful, but you need it in the following step in order to access your shares.

Warning

Be aware that starting this server potentially makes data available to parties that should not have access (e.g. if 3rd parties have access to your local network services). In any case, for different reasons, do not use this variant for more than some trying, testing, developing!

If your web server provides your Lawwenda installation at `https://example.com/shares/`, you can access your new `myshare` share at `https://example.com/shares/myshare/`. The base address might be anything, and just the share name has to be appended.

There are some optional parameters that can be specified in a `lawwenda_cli add_share` call. They are a large part of Lawwenda's versatility. Get more details by executing:

```
lawwenda_cli add_share -h
```

Once you have Lawwenda running in a web server and you have added at least one share, you can browse to its share url and start using it. Read the `h_ui` for more details about the user interface.

APPENDIX: INSTALLATION

Install Lawwenda via the installation package for your environment, if a suitable one exists for download. This also takes care of installing dependencies and doing preparation (unless mentioned otherwise in the installation procedure). It is possible (and not hard) to use the bare Source Package, but beyond this documentation.

For real usage it is highly recommended to host Lawwenda in a real web server. This makes use of the *wsgi* interface, which is commonly used for hosting Python based web application. You can skip this for now at come back somewhat later if you just want to try it out, and make the *First Steps*.

The exact steps for adding a wsgi application to a web server depend on the daemon that is used. Please read the documentation of your web server.

You will need to specify the location of a wsgi script during configuration. Create this file in a place that is convenient for you and fill it with the content that is printed by this command line:

```
lawwenda_cli generate_wsgi
```

For example, the steps for Apache 2.4 are roughly the following:

- Install and enable *mod_wsgi*.
- Add a snippet like this inside some *VirtualHost* in your Apache configuration:

```
WSGIScriptAlias /shares /srv/lawwenda/lawwenda.wsgi
<Directory /srv/lawwenda>
    Require all granted
</Directory>
```

The */shares* path is the path segment to append to the VirtualHost root url in order to reach Lawwenda. That url might then be *https://example.com/shares*. Note that this url will not return a useful answer. This is okay, the documentation will explain why.

The */srv/lawwenda/lawwenda.wsgi* path (and its parent path in *Directory*) can be chosen freely. This file will be created in the next step. The following assumes the paths as in the snippet.

- Execute this commands on a terminal:

```
mkdir -p /srv/lawwenda
lawwenda_cli generate_wsgi > /srv/lawwenda/lawwenda.wsgi
```

- Adapt as needed to match your security requirements. :)
- Restart Apache.

Note

The web server must be privileged to access the directories that you are going to share.

COMMAND LINE INTERFACE REFERENCE

API REFERENCE

7.1 lawwenda package

User-side API for reading and modifying Lawwenda configurations, creating shares, and more.

For most cases, instantiate `Configuration` and use some of its methods.

7.1.1 Subpackages

lawwenda.apps package

Lawwenda WSGI applications. See submodules.

`lawwenda.apps.pop_path_info(envIRON)`

Pop one segment of path info in environment, i.e. fix `PATH_INFO` and `SCRIPT_NAME`, and return that segment.

Parameters

`environ` (`dict[str, str]`) – The WSGI environ.

Return type

`str` | `None`

`lawwenda.apps.endpoint(method, path_pattern)`

Mark a function to be an endpoint in the web API of its application.

Parameters

- **method** (`str` | `None`) – The http method of this endpoint.
- **path_pattern** (`str` | `None`) – The path pattern of this endpoint.

Return type

`Callable[[Callable], Callable]`

Subpackages

lawwenda.apps.file_manager_app package

A wsgi application that provides a browser based user interface, similar to a desktop file manager.

See `FileManagerApp`.

```
class lawwenda.apps.file_manager_app.FileManagerApp(* (Keyword-only parameters separator (PEP
3102)), root_node, name, password_scrypt,
password_salt, title)
```

Bases: `object`

A wsgi application that provides a browser based user interface, similar to a desktop file manager, for a particular share.

It also includes `lawwenda.apps.http_webdav_app.HttpWebdavApp` for bare http/WebDAV functionality.

One instance is associated to one `lawwenda.config.share.Share`. See also `lawwenda.apps.main_app.MainApp`.

Parameters

- **root_node** (`lawwenda.fs.node.Node`) – The root filesystem node to share.
- **name** (`str`) – The share name.
- **password_script** (`str`) – The share password_script.
- **password_salt** (`str`) – The share password_salt.
- **title** (`str`) – The share title.

property url_internals_name: `str`

The URL internals name.

property root_node: `Node`

The filesystem root node served by this FileManagerApp.

__dispatch_request(`request`)

__dispatch_request_internals(`request`)

__dispatch_request_normal(`request`)

__rules_from_endpoints()

Parameters

endpoints_ (`Endpoints`)

Return type

Sequence[Rule]

__ensure_authorized(`request`)

__authenticate(`username, password`)

Parameters

- **username** (`str`)
- **password** (`str`)

Return type

`bool`

__json_response(`data`)

Parameters

data (*Any*)

Return type

Response

Submodules

lawwenda.apps.file_manager_app.endpoints module

See [Endpoints](#).

class lawwenda.apps.file_manager_app.endpoints.**Endpoints**(*root_node*, *url_internals_name*)

Bases: object

Web API endpoints.

Parameters

- **root_node** ([lawwenda.fs.node.Node](#))
- **url_internals_name** (*str*)

__api_thumbnail(*request*)

__api_upload(*request*)

__api_zip(*request*)

__api_zip_content(*request*, *zip_id*)

__static(*request*, *file_path*)

__help(*request*)

lawwenda.apps.file_manager_app.temp_zip_files module

See [TempZipFiles](#).

class lawwenda.apps.file_manager_app.temp_zip_files.**TempZipFiles**

Bases: object

Handler for temporary zip files.

Allows to create zip files containing some nodes and automatically cleans them up after some time.

This is a static class, potentially used by many applications in parallel.

create_temp_zip(*nodes*)

Create a temporary zip archive from some nodes in memory and return an identifier.

See also [temp_zip_content\(\)](#).

Parameters

nodes (*Iterable*[[lawwenda.fs.node.Node](#)]) – The nodes to include in the zip archive.

Return type

str

temp_zip_content(*zip_id*)

Return the binary content of a zip archive that was created before by [create_temp_zip\(\)](#).

Parameters

zip_id (*str*) – The identifier returned by [create_temp_zip](#).

Return type

bytes | *None*

```
__cleanup_loop()

class _TempZip(nodes)
    Bases: object
    A temporary zip file.

    Parameters
        nodes (Iterable[lawwenda.fs.node.Node])

    property bytes
        The binary representation of this temporary zip archive.

    static _TempZip__is_executable(node)
        Parameters
            node (lawwenda.fs.node.Node)
        Return type
            bool

    _TempZip__put_node(node, zip_file_obj)
        Parameters
            • node (lawwenda.fs.node.Node)
            • zip_file_obj (ZipFile)
        Return type
            None
```

[lawwenda.apps.file_manager_app.template_renderer](#) module

See [TemplateRenderer](#).

```
class lawwenda.apps.file_manager_app.template_renderer.TemplateRenderer(rootname, root_node,
                                                                           url_internals_name)
```

Bases: object

Template renderer for FileManagerApp.

Parameters

- **rootname** (*str*)
- **root_node** ([lawwenda.fs.node.Node](#))
- **url_internals_name** (*str*)

```
render(template, *, root_url, html_head_inner="", path="", head_only=False, **kwargs)
```

Render a template and return a response for it.

Parameters

- **template** (*str*) – The template name.
- **root_url** (*str*) – The root URL.
- **html_head_inner** (*str*) – Additional head html.
- **path** (*str*) – The path.
- **head_only** (*bool*) – Whether this is only a HEAD request.

Return type

Response

render_to_str(*template*, *, *root_url*, *csrf_token*, *html_head_inner*, ***kwargs*)

Render a template and return a string.

Parameters

- **template** (*str*) – The template name.
- **root_url** (*str*) – The root URL.
- **csrf_token** (*str*) – The CSRF token.
- **html_head_inner** (*str*) – Additional head html.

Return type

str

__render_to_str(*template*, ***kwargs*)

Parameters

template (*str*)

Return type

str

class lawwenda.apps.file_manager_app.template_renderer._RenderTemplateValue(*s*)

Bases: object

String representation for usage in template rendering.

Usually an HTML escaped variant of the input string, but with an option to also get the unescaped string.

Parameters

s (*str*)

property unescaped: str

The unescaped string.

lawwenda.apps.http_webdav_app package

A wsgi application that provides an http interface to a filesystem.

See [HttpWebdavApp](#).

class lawwenda.apps.http_webdav_app.HttpWebdavApp(*root_node*)

Bases: object

A wsgi application that provides an http interface to a filesystem.

It provides bare http access, and also WebDAV extensions for a more complete interface. It does not provide an own user interface (or any backend for it beyond http/WebDAV itself) or user authentication.

One instance is associated to one [lawwenda.config.share.Share](#). It is typically owned by an [lawwenda.apps.file_manager_app.FileManagerApp](#) instance.

Parameters

root_node ([lawwenda.fs.node.Node](#))

property root_node: [Node](#)

The filesystem root node served by this HttpWebdavApp.

__rules_from_endpoints()

Parameters

endpoints_ ([Endpoints](#))

Return type
dict[str, *Callable*]

Submodules

lawwenda.apps.http_webdav_app.dav_prop module

WebDAV properties.

`lawwenda.apps.http_webdav_app.dav_prop.register_prop(prop)`

Register a DavProp implementation.

Returns the same type afterward, so it can be used as class decorator.

Parameters
prop (*type*[[DavProp](#)]) – A subclass of DavProp.

Return type
[type](#)[[DavProp](#)]

`lawwenda.apps.http_webdav_app.dav_prop.get_prop_by_name(name)`

Return a DavProp by name or None if no such ones exist.

Parameters
name (*str*) – The name to find, as used in WebDAV.

Return type
[DavProp](#) | None

class `lawwenda.apps.http_webdav_app.dav_prop.DavProp(name, *, is_writable, is_internal=False)`

Bases: object

Base class for WebDAV properties.

Each subclass implements one particular property as used in e.g. PROPFIND requests.

Must be registered with [register_prop\(\)](#).

Parameters

- **name** (*str*)
- **is_writable** (*bool*)
- **is_internal** (*bool*)

property name: **str**
The name of this property.

property is_writable: **bool**
Whether this property can also be modified (in contrast to read-only properties like the file size).

property is_internal: **bool**
Whether this property is only for internal purposes and does not get announced.

get_for_node(*node*)
Return the property value for a node.

Parameters
node ([Node](#)) – The node.

Return type
str | *Element* | None

set_for_node(*node*, *value*)

Set the property value for a node.

Parameters

- **node** ([Node](#)) – The node.
- **value** (*str*)

Return type

None

class lawwenda.apps.http_webdav_app.dav_prop.ResourceTypeDavProp

Bases: [DavProp](#)

The WebDAV resourcetype property.

get_for_node(*node*)

Return the property value for a node.

Parameters

node – The node.

class lawwenda.apps.http_webdav_app.dav_prop.SizeDavProp

Bases: [DavProp](#)

The WebDAV getcontentlength property.

get_for_node(*node*)

Return the property value for a node.

Parameters

node – The node.

class lawwenda.apps.http_webdav_app.dav_prop.MtimeDavProp

Bases: [DavProp](#)

The WebDAV getlastmodified property.

get_for_node(*node*)

Return the property value for a node.

Parameters

node – The node.

class lawwenda.apps.http_webdav_app.dav_prop.CrtimeDavProp

Bases: [DavProp](#)

The WebDAV creationdate property.

get_for_node(*node*)

Return the property value for a node.

Parameters

node – The node.

class lawwenda.apps.http_webdav_app.dav_prop.DisplayNameDavProp

Bases: [DavProp](#)

The WebDAV displayname property.

get_for_node(*node*)

Return the property value for a node.

Parameters

node – The node.

class lawwenda.apps.http_webdav_app.dav_prop.**InfoJsonProp**

Bases: [*DavProp*](#)

Custom WebDAV property {DAV:lawwenda}info_json that provides some basic file information in a simpler way than common ones do.

get_for_node(*node*)

Return the property value for a node.

Parameters

node – The node.

class lawwenda.apps.http_webdav_app.dav_prop.**FullInfoJsonProp**

Bases: [*DavProp*](#)

Custom WebDAV property {DAV:lawwenda}full_info_json that provides file information in a simpler way than common ones do.

get_for_node(*node*)

Return the property value for a node.

Parameters

node – The node.

class lawwenda.apps.http_webdav_app.dav_prop.**KnownTagsProp**

Bases: [*DavProp*](#)

Custom WebDAV property {DAV:lawwenda}known_tags that contains all known tags of the current instance.

get_for_node(*node*)

Return the property value for a node.

Parameters

node – The node.

class lawwenda.apps.http_webdav_app.dav_prop.**CommentDavProp**

Bases: [*DavProp*](#)

Custom WebDAV property {DAV:xattr}xdg.comment that mirrors [*lawwenda.fs.node.Node.comment*](#).

get_for_node(*node*)

Return the property value for a node.

Parameters

node – The node.

set_for_node(*node*, *value*)

Set the property value for a node.

Parameters

node – The node.

class lawwenda.apps.http_webdav_app.dav_prop.**TagsDavProp**

Bases: [*DavProp*](#)

Custom WebDAV property {DAV:xattr}xdg.tags that mirrors [*lawwenda.fs.node.Node.tags*](#).

get_for_node(*node*)

Return the property value for a node.

Parameters

node – The node.

set_for_node(*node*, *value*)

Set the property value for a node.

Parameters

node – The node.

class lawwenda.apps.http_webdav_app.dav_prop.**RatingDavProp**

Bases: [DavProp](#)

Custom WebDAV property {DAV:xattr}baloo.rating that mirrors [lawwenda.fs.node.Node.rating](#).

get_for_node(*node*)

Return the property value for a node.

Parameters

node – The node.

set_for_node(*node*, *value*)

Set the property value for a node.

Parameters

node – The node.

class lawwenda.apps.http_webdav_app.dav_prop.**GeoDavProp**

Bases: [DavProp](#)

Custom WebDAV property {DAV:xattr}pino.geo that mirrors [lawwenda.fs.node.Node.geolocation](#).

get_for_node(*node*)

Return the property value for a node.

Parameters

node – The node.

set_for_node(*node*, *value*)

Set the property value for a node.

Parameters

node – The node.

lawwenda.apps.http_webdav_app.endpoints module

See [Endpoints](#).

class lawwenda.apps.http_webdav_app.endpoints.**Endpoints**(*root_node*)

Bases: object

Web API endpoints.

Parameters

root_node ([lawwenda.fs.node.Node](#))

property allowed_methods: Sequence[str]

All allowed HTTP methods.

__get(*request*, *node*, *, *is_head_request=False*)

Parameters

- **request** (*Request*)
- **node** (*Node*)
- **is_head_request** (*bool*)

__head(*request*, *node*)

Parameters

- **request** (*Request*)
- **node** (*Node*)

__propfind(*request*, *node*)

Parameters

- **request** (*Request*)
- **node** (*Node*)

__proppatch(*request*, *node*)

Parameters

- **request** (*Request*)
- **node** (*Node*)

__options(*request*, *node*)

Parameters

- **request** (*Request*)
- **node** (*Node*)

__put(*request*, *node*)

Parameters

- **request** (*Request*)
- **node** (*Node*)

__delete(*request*, *node*)

Parameters

- **request** (*Request*)
- **node** (*Node*)

__mkcol(*request*, *node*)

Parameters

- **request** (*Request*)
- **node** (*Node*)

__copy(*request*, *node*)

Parameters

- **request** (*Request*)
- **node** (*Node*)

__move(*request*, *node*)

Parameters

- **request** (*Request*)
- **node** (*Node*)

__propfind__args()

Parameters

headers (*dict*[*str*, *str*])

Return type

tuple[*Search* | *None*, *bool*, *str*, *bool*]

__propfind__walk(*node*, *root_path*, *depth*, *props*, *only_names*, *search_config*, *hidden_files_visible*, *sort_column*, *sort_descending*)

Parameters

- **xml_result** (*Element*)
- **node** (*Node*)
- **root_path** (*str*)
- **depth** (*str*)
- **props** (*Sequence*[*DavProp*])
- **only_names** (*bool*)
- **search_config** (*Search* | *None*)
- **hidden_files_visible** (*bool*)
- **sort_column** (*str*)
- **sort_descending** (*bool*)

Return type

None

__copy_move(*request*, *node*, *, *transfer_method*)

Parameters

- **request** (*Request*)
- **node** (*Node*)
- **transfer_method** (*str*)

__xml_multistatus_response(*prop_values*)

Parameters

- **href** (*str*)

- **prop_values** (*dict[str, str | Element]*)

Return type

Element

__parse_xml_propertyupdate()

Parameters

xml_propertyupdate (*<module 'xml.etree.ElementTree' from '/usr/lib/python3.13/xml/etree/ElementTree.py'>*)

Return type

Sequence[tuple[str, str | None]]

__set_node_prop(*prop_name, value*)

Parameters

- **node** (*Node*)
- **prop_name** (*str*)
- **value** (*str*)

Return type

None

lawwenda.apps.main_app package

Lawwenda main wsgi application.

See [MainApp](#).

class lawwenda.apps.main_app.**MainApp**(*configuration*)

Bases: *object*

Lawwenda main wsgi application.

One instance is associated to one “lawwenda.Configuration” and internally hosts one [lawwenda.apps.file_manager_app.FileManagerApp](#) for each share inside that configuration.

Parameters

configuration (*lawwenda.Configuration*) – The Lawwenda configuration.

__get_app_for_share(*share_name*)

Parameters

share_name (*str*)

class _ShareAppInst(*share*)

Bases: *object*

Container for one [lawwenda.apps.file_manager_app.FileManagerApp](#) and its associated [lawwenda.config.share.Share](#).

Parameters

share (*lawwenda.config.share.Share*)

property app: [FileManagerApp](#)

The application.

property is_active: *bool*

Whether this share is still active (not yet removed).

lawwenda.asset package

Various static data and auxiliary project information.

Subpackages

lawwenda.asset.PiMetadataInterpreter namespace

Submodules

lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter module

Interpreting some file metadata.

For basic usage, see [metadata_for_file\(\)](#). For doing searches based on file metadata, see [search_engine\(\)](#).

```
lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.metadata_for_file(file, *,
                                                                           xattr_engine_type=None)
```

Return metadata for a file or directory (incl. facilities to change them).

Parameters

- **file** (*Path* / *str*) – The file or directory to operate on with this instance.
- **xattr_engine_type** (*type[XattrEngine]* / *None*) – For some cases it might be convenient to subclass [FileMetadata.XattrEngine](#) and use this subclass here instead (e.g. a custom xattr backend).

Return type

[FileMetadata](#)

```
lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.search_engine(root_dir, *,
                                                                           xattr_engine_type=None,
                                                                           search_engine_type=None,
                                                                           **kwargs)
```

Return a search engine for a root directory. Use this object for searching by metadata in this directory.

Parameters

- **root_dir** (*Path* / *str*) – The directory to search within.
- **xattr_engine_type** (*type[XattrEngine]* / *None*) – For some cases it might be convenient to subclass [FileMetadata.XattrEngine](#) and use this subclass here instead (e.g. a custom xattr backend).
- **search_engine_type** (*type[SearchEngine]* / *None*) – For some cases it might be convenient to subclass [SearchEngine](#) and use this subclass here instead.

Return type

[SearchEngine](#)

```
class lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadata(file,
                                                                           xattr_engine)
```

Bases: [Generic](#)

Metadata of a particular file (or directory), allowing reading and modifying it.

Do not use directly. See [metadata_for_file\(\)](#).

Parameters

- **file** (*Path* / *str*) – The file (or directory) this metadata is about.

- **xattr_engine** (*TXattrEngine*) – The FileMetadata.XattrEngine to use.

property file: **Path**

The file (or directory) this metadata is about.

property tags: **Sequence[TagAssignment]**

All tags assigned to this file, including ones inherited from ancestors. See also [add_tag\(\)](#) and [remove_tag\(\)](#). See also [tags_without_ancestors](#).

property tags_without_ancestors: **Sequence[TagAssignment]**

All tags assigned to this file, not including ones inherited from ancestors. See also [add_tag\(\)](#) and [remove_tag\(\)](#). See also [tags](#).

add_tag(*tag_name*)

Add a new tag (and return that new assignment). See also [tags](#).

Parameters

tag_name (*str*) – The label of the new tag.

Return type

[TagAssignment](#)

add_tag_by_tag_string(*tag_string*)

Add a new tag by a given tag string (and return that new assignment). See also [FileMetadata.TagAssignment.to_tag_string](#). See also [tags](#).

Parameters

tag_string (*str*) – The tag_string for the new tag.

Return type

[TagAssignment](#)

remove_tag(*tag_name*)

Remove a tag by name and return whether anything was removed. See also [tags](#).

Parameters

tag_name (*str*)

Return type

bool

set_tags_by_tag_strings(*tag_strings*)

Set the tags to a given list. See also [tags](#).

Parameters

tag_strings (*Iterable[str]*) – The tag strings.

Return type

None

property comment: **str**

The comment text. See also [append_comment\(\)](#) (or use its setter for changing it).

append_comment(*comment*)

Append a comment text (i.e. conserving the old one). See also [comment](#).

Parameters

comment (*str*) – The text to append.

Return type

None

RATING_MIN_VALID_VALUE = 1

The minimal rating that can be specified for a file (or directory). See also [RATING_UNDEFINED_VALUE](#)!

RATING_MAX_VALID_VALUE = 10

The maximum rating that can be specified for a file (or directory).

RATING_UNDEFINED_VALUE = 0

This rating value is sometimes used (e.g. in [rating_or_default](#)) when a file (or directory) is not rated at all. Typically, that case would instead be represented by `None`, though! Note: This value is less than [RATING_MIN_VALID_VALUE](#)!

property rating: int | None

The rating, if the associated file has one. Between [RATING_MIN_VALID_VALUE](#) and [RATING_MAX_VALID_VALUE](#), or `:py:code:`None``. This is never [RATING_UNDEFINED_VALUE](#). See also [rating_or_default](#).

property rating_or_default: int

The rating, if the associated file has one, or a default value otherwise. Between [RATING_MIN_VALID_VALUE](#) and [RATING_MAX_VALID_VALUE](#), or [RATING_UNDEFINED_VALUE](#). See also [rating](#).

reset_rating()

Set the [rating](#) back to [RATING_UNDEFINED_VALUE](#).

Return type

`None`

property geolocation: [Geolocation](#) | None

The geolocation.

property _xattr_engine: [TXattrEngine](#)

The xattr engine used by this metadata object.

_tag_assignments_flat()

Return the assigned tags. It will contain only tags assigned directly to this file; see [MetadataInterpreter._tag_assignments](#).

Return type

[_TagAssignments](#)

_tag_assignments()

Return the assigned tags. It may also contain deep tags assigned to parent directories; see [MetadataInterpreter._tag_assignments_flat](#).

Return type

[_TagAssignments](#)

static _rating_sanitize_value(value)

Parameters

value (`str` | `int`)

Return type

`int` | `None`

_list_xattrs()

Return a list of available extended attribute keys. Used internally. Includes cache handling.

Return type

[Sequence](#)[`str`]

_get_xattr(*key*)

Return an extended attribute value by key; if key doesn't exist, it returns "". Used internally. Includes cache handling.

Parameters

key (*str*)

Return type

str

_set_xattr(*key, val*)

Set an extended attribute. Used internally. Includes cache handling.

Parameters

- **key** (*str*)
- **val** (*str*)

Return type

None

_unset_xattr(*key*)

Unset an extended attribute. Used internally. Includes cache handling.

Parameters

key (*str*)

Return type

None

_store_tag_assignments(*assignments*)

Set assigned tags, potentially overwriting the existing ones. Beforehand you can get an instance with `MetadataInterpreter._tag_assignments_flat` and edit it.

Parameters

assignments (`_TagAssignments`) – The new tag assignments.

Return type

None

static _cache_enabled()

Return a context in which a cache is enabled for quicker operations.

Return type

`Generator[None, None, None]`

static _clear_cache()

Used internally for clearing the cache.

Return type

None

_cache_lock = <unlocked `_thread.lock` object>

_cache_refcount = 0

_cache_xattrs: dict[tuple[Path, str], str] | None = None

XATTR_KEY_COMMENT = 'xdg.comment'

XATTR_KEY_GEO = 'pino.geo'

XATTR_KEY_RATING = 'baloo.rating'

XATTR_KEY_TAGS = 'xdg.tags'

class TagAssignment(*tag_name*, *file*)

Bases: object

The assignment of one tag to one file (or directory).

Do not use directly.

Parameters

- **tag_name** (*str*) – The tag name.
- **file** (*Path* | *str*) – The file this assignment is associated with.

property file: *Path*

The file (or directory) this assignment is associated with.

property tag_name: *str*

The tag name.

This is the clean tag name without any encoding and without any special meaning of particular characters. See also [to_tag_string\(\)](#) for its raw on-disk representation.

to_tag_string()

Return an encoded string representation of the tag name that is safe to use e.g. in a URL.

Do not use directly.

It contains no spaces. This is used for on-disk storage of this tag assignment. See also [from_tag_string\(\)](#).

Return type

str

static from_tag_string(*tag_string*, *file*)

Return a TagAssignment from a tag_string (like "baz").

Do not use directly.

See [to_tag_string\(\)](#) and [FileMetadata._TagAssignments.from_tags_string\(\)](#).

Parameters

- **tag_string** (*str*) – A tag string.
- **file** (*Path* | *str*) – The file this assignment are associated with.

Return type

[TagAssignment](#)

class _TagAssignments(*file*)

Bases: object

Storage and management for the tag assignments of a file. Note: This is only a volatile, in-memory object without any persistence included. The owner of such an object must take care of actually writing changes to disk at some point.

Do not use directly.

Do not use directly.

Parameters

- **file** (*Path* | *str*) – The file these assignments are associated with.

property file: Path

The file (or directory) these assignments are associated with. Note: Individual assignments may have own filepaths.

all_assignments(*, *exclude_ancestors=False*)

Return all tag assignments.

Parameters

exclude_ancestors (*bool*) – Whether to exclude assignments which came from ancestors (i.e. ‘parent directories’).

Return type

Sequence[[TagAssignment](#)]

add_assignment(*tag_string*, *file=None*)

Add a tag assignment (and return it).

Parameters

- **tag_string** (*str*) – The tag string of the tag assignment to add.
- **file** (*Path | str | None*) – The file to associate this assignment with. Default: [file](#).

Return type

[TagAssignment](#)

find_assignments_by_tag_name(*tag_name*, *, *exclude_ancestors=False*)

Return any assignments matching a given tag name.

Parameters

- **tag_name** (*str*) – A tag name.
- **exclude_ancestors** (*bool*) – Whether to exclude assignments, which came from parent directories.

Return type

Sequence[[TagAssignment](#)]

remove_assignments_by_tag_name(*tag_name*)

Remove any assignments matching a given tag name and return the number of removed assignments.

Parameters

tag_name (*str*) – A tag name.

Return type

int

is_assignment_from_an_ancestor(*assignment*)

Return whether an assignment is coming from an ancestor, i.e. not directly assigned to this file.

Parameters

assignment ([TagAssignment](#)) – The [TagAssignment](#) to check.

Return type

bool

merge_assignments(*source_assignments*)

Merge all relevant assignments from another instance into this one.

Parameters

source_assignments ([_TagAssignments](#)) – The assignments to merge into this one.

Return type

None

to_tags_string(*, *exclude_ancestors=True*)

Return a string representation (e.g. for on-disk storage). See also [from_tags_string\(\)](#).

Parameters

exclude_ancestors (*bool*) – Whether to exclude assignments, which came from parent directories.

Return type

str

static `from_tags_string(tags_string, file)`Return tag assignments from a tags string. See also `to_tags_string()`.

- A ‘tags string’ is the raw content of the user.xdg.tags xattr for a particular file (or directory).
- A tags string describes a “,”-joint list of tag strings (note: singular), like this: "tag_string_1, tag_string_2, ..., tag_string_N"
- A tag string fully describes one tag assignment to the particular file/dir. It includes the tag name.

Parameters

- **tags_string** (str) – A tags string.
- **file** (Path | str) – The file these assignments are associated with.

Return type`_TagAssignments`**static** `check_is_assignment_from_an_ancestor(file, assignment)`Like `is_assignment_from_an_ancestor()` when no instance is available.**Parameters**

- **file** (Path | str) – The path to assume as the original one.
- **assignment** (TagAssignment) – The TagAssignment to check.

Return type

bool

`_TagAssignments__remove_assignments_by_tag_name(tag_name, *, exclude_ancestors=True)`**Parameters**

- **tag_name** (str)
- **exclude_ancestors** (bool)

Return type

int

class `Geolocation(*, lat, lon, accuracy_meters=None, **kwargs)`

Bases: object

A location on earth.

Parameters

- **lat** (float) – The latitude in degrees.
- **lon** (float) – The longitude in degrees.
- **accuracy_meters** (float | None) – The accuracy in meters.
- **kwargs** – Additional attributes.

`_ATTR_KEY_LATITUDE = 'lat'``_ATTR_KEY_LONGITUDE = 'lon'``_ATTR_KEY_ACCURACY = 'acc'`**property** `attributes: dict[str, Any]`

The dictionary with all attributes.

property `lat: float`

The latitude of this location in degrees.

property `lon: float`

The longitude of this location in degrees.

property accuracy_meters: float

The accuracy of this location in meters.

distance_meters(*to_location*)

Return the point distance (respecting accuracy) between this location and another one in meters.

Parameters

to_location ([Geolocation](#)) – The other geolocation.

Return type

float

distance_meters_interval(*to_location*)

Return the minimum and maximum distance (respecting accuracy) between this location and another one in meters.

Parameters

to_location ([Geolocation](#)) – The other geolocation.

Return type

tuple[float, float]

to_geo_string()

Return a string representation. See [from_geo_string\(\)](#).

Return type

str

static from_geo_string(*geo_string*, ***kwargs*)

Create a geolocation from a geo string (like "lat=1&lon=2"). See [to_geo_string\(\)](#).

Parameters

geo_string (*str*) – A geo string.

Return type

[Geolocation](#)

class XattrEngine(*file*)

Bases: object

The engine for handling extended attributes. For a custom xattr backend, subclass it and see interpreter.

Parameters

file (*Path*)

list_xattrs()

Return a list of available extended attribute keys. Used internally. Without cache handling. You might add an own implementation of it (and some others) in a custom subclass for a custom xattr backend.

Return type

Sequence[str]

get_xattr(*key*)

Get an extended attribute. If key doesn't exist, it returns "". Used internally. Without cache handling. You might add an own implementation of it (and some others) in a custom subclass for a custom xattr backend.

Parameters

key (*str*)

Return type

str

set_xattr(*key*, *val*)

Set an extended attribute. Used internally. Without cache handling. You might add an own implementation of it (and some others) in a custom subclass for a custom xattr backend.

Parameters

- **key** (*str*)
- **val** (*str*)

Return type

None

unset_xattr(*key*)

Unset an extended attribute. Used internally. Without cache handling. You might add an own implementation of it (and some others) in a custom subclass for a custom xattr backend.

Parameters**key** (*str*)**Return type**

None

exception TagDuplicateError(*new_tag_string*, *existing_tag_string*)

Bases: RuntimeError

Exception for tag duplicates occurred.

Parameters

- **new_tag_string** (*str*)
- **existing_tag_string** (*str*)

```
class lawwenda.asset.PiMetadataInterpreter.PiMetadataInterpreter.SearchEngine(root_dir,
                                                                              xattr_engine_type,
                                                                              **_)
```

Bases: object

Search engine. Instances of them allow to execute searches within a directory on the metadata of contained items.

Do not use directly. See [search_engine\(\)](#).

Parameters

- **root_dir** (*Path*) – The root directory to start searches in (as absolute path).
- **xattr_engine_type** (*type[XattrEngine]*) – The [FileMetadata.XattrEngine](#) subclass to use.

property root_dir: Path

The root directory to start searches in.

search(**criteria*)

Execute a search by some criteria and returns all matching items.

Parameters

criteria ([SearchCriterion](#) / *str*) – One or more search criteria. Items will match if they match all the given criteria (but you can logically combine them further inside e.g. using [Or\(\)](#)).

Return type*Sequence*[[ResultItem](#)]**_list_dir**(*path*)

Return a list of subitems of a directory. Used internally. Implement this in a custom search engine.

Parameters**path** (*Path*)**Return type***Sequence*[*str*]

_is_dir(*path*)

Return whether a given path points to a directory. Used internally. Implement this in a custom search engine.

Parameters

path (*Path*)

Return type

bool

_metadata(*file*)

Return metadata for a file. Used internally.

Parameters

file (*Path* | *str*)

Return type

[FileMetadata](#)

class ResultItem(*file*, *summary*)

Bases: object

One file or directory as part of a search result.

Used internally. See `SearchEngine.search`.

Parameters

- **file** (*Path* | *str*)
- **summary** (*str*)

property file: *Path*

The file (or directory) that matched the search criteria.

property summary: *str*

A summary text that might give additional details about *how* *file* matched the search criteria.

class SearchCriterion

Bases: ABC

Abstract base class for the implementation of search criteria (those with program code deciding if a file matches a certain query). Also includes business logic for (de)serialization of query strings.

class SerializationFormat(**values*)

Bases: Enum

Various serialization formats for parameters of search criteria. See `define_param()`.

JSON = 1

Allows to serialize arbitrary objects to JSON.

SUBCRITERION = 2

Allows to serialize a [SearchEngine.SearchCriterion](#) to a compactly embedded query string.

URL_ENCODED = 3

Serialize a string to a (reduced) urlencoded format.

static define_param(*arg_name*, *, *index=None*, *ctor_arg_list=False*,
format=SerializationFormat.JSON)

Used for decorating SearchCriterion subclasses for specifying which arguments to (de)serialize and how.

Parameters

- **arg_name** (*str*) – The argument name as it is stored inside the instance (and as the constructor expects it).
- **index** (*int* / *None*) – If specified, then it's an index of a positional argument within the resulting query string. Otherwise, this parameter will be passed via keyword.
- **ctor_arg_list** (*bool*) – Whether the parameter is expected to be a list, which is to be expanded within the query string and passed to the constructor as positional arguments.
- **format** ([SerializationFormat](#)) – The serialization format. One of `SearchEngine.SearchCriterion.SERIALIZE_FORMAT_*`.

static _encode_part(*s*)

Encode a value string in some reduces variant of urlencoding, avoiding conflicts with the control characters used in a query string. Used internally.

Parameters

s (*str*) – The part.

Return type

str

static _serialize_value(*v, format*)

Serialize an arbitrary value to some serialization format. Used internally.

Parameters

- **v** (*Any*) – The value as string, or arbitrary object, or whatever 'format' allows.
- **format** ([SerializationFormat](#)) – The serialization format.

Return type

str

static _deserialize_value(*s, format*)

Deserialize back what `SearchEngine.SearchCriterion._serialize_value` returns. Used internally.

Parameters

- **s** (*str*) – The string to deserialize.
- **format** ([SerializationFormat](#)) – The serialization format.

Return type

Any

static from_query_string(*query_string*)

Return a `SearchCriterion` by interpreting a query string. See also [SearchEngine.SearchCriterion.to_query_string\(\)](#).

Parameters

query_string (*str*) – A query string specifying the search criterion.

Return type

[SearchCriterion](#)

to_query_string()

Return a query string representation for this search criterion. See also [SearchEngine.SearchCriterion.from_query_string\(\)](#).

Return type

str

abstractmethod is_matching(*metadata*)

Check whether the given file (or dir) is matching the search criterion. Used internally. Implement this in a subclass with own decision logic.

Returning a string, a list of strings (even an empty one) or `True` is considered a positive answer, `False` or `None` is considered a negative answer.

Parameters

metadata ([FileMetadata](#)) – Metadata of the file to check. Get its path by *metadata.file*.

Return type

`str | Iterable[str] | bool | None`

static Not(*inner_criterion*)

A search criterion which inverts an inner one. Sample query string: *Not?{SomeOther?...}*.

Parameters

inner_criterion ([SearchCriterion](#)) – The criterion to invert.

Return type

[SearchCriterion](#)

static Or(**inner_criteria*)

A search criterion which matches, iff at least one of its inner ones matches. Sample query string: *Or?{SomeOther?...}&{YetAnother?...}&...*

Parameters

inner_criteria ([SearchCriterion](#)) – The criteria to combine.

Return type

[SearchCriterion](#)

static And(**inner_criteria*)

A search criterion which matches, iff all its inner ones match. Sample query string: *And?{SomeOther?...}&{YetAnother?...}&...*

Parameters

inner_criteria ([SearchCriterion](#)) – The criteria to combine.

Return type

[SearchCriterion](#)

static Tag(*tag_name, is_re=False*)

A search criterion which matches, iff an item has a certain tag assigned. Sample query string: *Tag?some_tag* or *Tag?some_tag&is_re=true*.

Parameters

- **tag_name** (*str*) – The tag name to search for.
- **is_re** (*bool*) – Whether ‘tag_name’ is to be interpreted as regular expression.

Return type

[SearchCriterion](#)

static Comment(*comment, is_re=False*)

A search criterion which matches, iff an item has a certain comment. Sample query string: *Comment?foo.*&is_re=true*.

Parameters

- **comment** (*str*) – The comment substring to search for (as a regular expression, it applies to the entire comment).
- **is_re** (*bool*) – Whether ‘comment’ is to be interpreted as regular expression.

Return type

[SearchCriterion](#)

static Rating(*min=None, max=None*)

A search criterion which matches, iff an item is in a certain rating range. Note that each item implicitly has some default rating, if it never got one assigned explicitly. Sample query string: *Rating?min=2&max=3*. See also [FileMetadata.RATING_MIN_VALID_VALUE](#), [FileMetadata.RATING_MAX_VALID_VALUE](#) and [FileMetadata.RATING_UNDEFINED_VALUE](#).

Parameters

- **min** (*int | None*) – The minimum rating to search for (default: unlimited).
- **max** (*int | None*) – The maximum rating to search for (default: unlimited).

Return type

[SearchCriterion](#)

static Geo(*center=None, lat=None, lon=None, distance_meters=5000, distance_tolerance=None*)

A search criterion which matches, iff an item is within a certain distance of a certain geographic range. Sample query string: *Geo?lat=12&lon=34&distance_meters=5000*.

You must specify either center, or both of lat and lon!

Parameters

- **center** ([Geolocation](#) | *None*) – The geographic center location.
- **lat** (*float* | *None*) – The latitude of the geographic center location as number.
- **lon** (*float* | *None*) – The longitude of the geographic center location as number.
- **distance_meters** (*float*) – The maximum distance in meters.
- **distance_tolerance** (*int* | *None*) – How to deal with position inaccuracies. *None*: center-to-center, -1: worst case, 1: best case.

Return type

[SearchCriterion](#)

static XattrValue(*value, *, key, key_is_re=None, value_is_re=None*)

A search criterion which matches, iff a certain item's xattr has a certain value. You should typically not need it. Sample query string: *XattrValue?some%20comment&key=comment*.

Parameters

- **value** – The value to search for (as string).
- **key** – The xattr key to look into (as string; *None* for all).
- **key_is_re** – Whether key is to be interpreted as regular expression (default: no).
- **value_is_re** – Whether value is to be interpreted as regular expression (default: no).

Return type

[SearchCriterion](#)

static _is_matching_answer(*value*)

Interpret the answer of an [is_matching\(\)](#) call and return whether it is considered a positive answer and a list of comments (for positive answers).

Parameters

value (*str* | *Iterable[str]* | *bool* | *None*) – The value to interpret.

Return type

tuple[bool, Sequence[str]]

_abc_impl = *<_abc._abc_data object>*

class _SearchCriterionNot(*inner_criterion*)

Bases: [SearchCriterion](#)

See [SearchEngine.SearchCriterion.Not\(\)](#).

Parameters

inner_criterion ([SearchEngine.SearchCriterion](#) | *str*)

is_matching(*metadata*)

Check whether the given file (or dir) is matching the search criterion. Used internally. Implement this in a subclass with own decision logic.

Returning a string, a list of strings (even an empty one) or *True* is considered a positive answer, *False* or *None* is considered a negative answer.

Parameters

metadata – Metadata of the file to check. Get its path by *metadata.file*.

_abc_impl = *<_abc._abc_data object>*

_pmi_serialization_info = *[('inner_criterion', 0, SerializationFormat.SUBCRITERION, False)]*

```
class _SearchCriterionOr(*inner_criteria)
```

Bases: [SearchCriterion](#)

See [SearchEngine.SearchCriterion.Or\(\)](#).

Parameters

inner_criteria ([SearchEngine.SearchCriterion](#) / *str*)

is_matching(*metadata*)

Check whether the given file (or dir) is matching the search criterion. Used internally. Implement this in a subclass with own decision logic.

Returning a string, a list of strings (even an empty one) or True is considered a positive answer, False or None is considered a negative answer.

Parameters

metadata – Metadata of the file to check. Get its path by *metadata.file*.

_abc_impl = <_abc._abc_data object>

_pmi_serialization_info = [('inner_criteria', 0, SerializationFormat.SUBCRITERION, True)]

```
class _SearchCriterionAnd(*inner_criteria)
```

Bases: [SearchCriterion](#)

See [SearchEngine.SearchCriterion.And\(\)](#).

Parameters

inner_criteria ([SearchEngine.SearchCriterion](#) / *str*)

is_matching(*metadata*)

Check whether the given file (or dir) is matching the search criterion. Used internally. Implement this in a subclass with own decision logic.

Returning a string, a list of strings (even an empty one) or True is considered a positive answer, False or None is considered a negative answer.

Parameters

metadata – Metadata of the file to check. Get its path by *metadata.file*.

_abc_impl = <_abc._abc_data object>

_pmi_serialization_info = [('inner_criteria', 0, SerializationFormat.SUBCRITERION, True)]

```
class _SearchCriterionTag(tag_name, is_re=None)
```

Bases: [SearchCriterion](#)

See [SearchEngine.SearchCriterion.Tag\(\)](#).

Parameters

- **tag_name** (*str*)
- **is_re** (*bool* / *None*)

is_matching(*metadata*)

Check whether the given file (or dir) is matching the search criterion. Used internally. Implement this in a subclass with own decision logic.

Returning a string, a list of strings (even an empty one) or True is considered a positive answer, False or None is considered a negative answer.

Parameters

metadata – Metadata of the file to check. Get its path by *metadata.file*.

_abc_impl = <_abc._abc_data object>

_pmi_serialization_info = [('tag_name', 0, SerializationFormat.URL_ENCODED, False), ('is_re', None, SerializationFormat.JSON, False)]

class _SearchCriterionComment(*comment, is_re=None*)

Bases: [SearchCriterion](#)

See [SearchEngine.SearchCriterion.Comment\(\)](#).

Parameters

- **comment** (*str*)
- **is_re** (*bool | None*)

is_matching(*metadata*)

Check whether the given file (or dir) is matching the search criterion. Used internally. Implement this in a subclass with own decision logic.

Returning a string, a list of strings (even an empty one) or True is considered a positive answer, False or None is considered a negative answer.

Parameters

metadata – Metadata of the file to check. Get its path by *metadata.file*.

_abc_impl = <_abc._abc_data object>

_pmi_serialization_info = [('comment', 0, SerializationFormat.URL_ENCODED, False), ('is_re', None, SerializationFormat.JSON, False)]

class _SearchCriterionRating(*min=None, max=None*)

Bases: [SearchCriterion](#)

See [SearchEngine.SearchCriterion.Rating\(\)](#).

Parameters

- **min** (*int | None*)
- **max** (*int | None*)

is_matching(*metadata*)

Check whether the given file (or dir) is matching the search criterion. Used internally. Implement this in a subclass with own decision logic.

Returning a string, a list of strings (even an empty one) or True is considered a positive answer, False or None is considered a negative answer.

Parameters

metadata – Metadata of the file to check. Get its path by *metadata.file*.

_abc_impl = <_abc._abc_data object>

_pmi_serialization_info = [('min', None, SerializationFormat.JSON, False), ('max', None, SerializationFormat.JSON, False)]

```
class _SearchCriterionGeo(center=None, lat=None, lon=None, distance_meters=5000,  
                          distance_tolerance=None)
```

Bases: [SearchCriterion](#)

See [SearchEngine.SearchCriterion.Geo\(\)](#).

Parameters

- **center** ([Geolocation](#) | *None*)
- **lat** (*float* | *None*)
- **lon** (*float* | *None*)
- **distance_meters** (*float*)
- **distance_tolerance** (*int* | *None*)

is_matching(*metadata*)

Check whether the given file (or dir) is matching the search criterion. Used internally. Implement this in a subclass with own decision logic.

Returning a string, a list of strings (even an empty one) or True is considered a positive answer, False or None is considered a negative answer.

Parameters

metadata – Metadata of the file to check. Get its path by *metadata.file*.

```
_abc_impl = <_abc._abc_data object>
```

```
_pmi_serialization_info = [('lat', None, SerializationFormat.JSON, False),  
                           ('lon', None, SerializationFormat.JSON, False), ('distance_meters', None,  
                           SerializationFormat.JSON, False), ('distance_tolerance', None,  
                           SerializationFormat.JSON, False)]
```

```
class _SearchCriterionXattrValue(key, value, key_is_re=None, value_is_re=None)
```

Bases: [SearchCriterion](#)

See [SearchEngine.SearchCriterion.XattrValue\(\)](#).

Parameters

- **key** (*str* | *None*)
- **value** (*str*)
- **key_is_re** (*bool* | *None*)
- **value_is_re** (*bool* | *None*)

is_matching(*metadata*)

Check whether the given file (or dir) is matching the search criterion. Used internally. Implement this in a subclass with own decision logic.

Returning a string, a list of strings (even an empty one) or True is considered a positive answer, False or None is considered a negative answer.

Parameters

metadata – Metadata of the file to check. Get its path by *metadata.file*.

```
_abc_impl = <_abc._abc_data object>
```

```
_pmi_serialization_info = [('key', None, SerializationFormat.URL_ENCODED,
False), ('value', 0, SerializationFormat.URL_ENCODED, False), ('key_is_re',
None, SerializationFormat.JSON, False), ('value_is_re', None,
SerializationFormat.JSON, False)]
```

```
exception MetadataFormatError
```

```
Bases: ValueError
```

```
class lawwenda.asset.PiMetadataInterpreter.pimetadataainterpreter._CommandLineInterface
```

```
Bases: object
```

```
Allows usage of the most features from command line.
```

```
comment__get(metadata)
```

```
Parameters
```

```
    metadata (FileMetadata)
```

```
comment__set(metadata, new_comment)
```

```
Parameters
```

- metadata (FileMetadata)
- new_comment (str)

```
comment__append(metadata, new_comment)
```

```
Parameters
```

- metadata (FileMetadata)
- new_comment (str)

```
geo__get(metadata)
```

```
Parameters
```

```
    metadata (FileMetadata)
```

```
geo__get_raw(metadata)
```

```
Parameters
```

```
    metadata (FileMetadata)
```

```
geo__modify(metadata, key, value)
```

```
Parameters
```

- metadata (FileMetadata)
- key (str)
- value (str)

```
geo__set(metadata, new_geo_string)
```

```
Parameters
```

- metadata (FileMetadata)
- new_geo_string (str)

`rating__get(metadata)`

Parameters

`metadata` (`FileMetadata`)

`rating__set(metadata, new_rating)`

Parameters

- `metadata` (`FileMetadata`)
- `new_rating` (`str`)

`rating__reset(metadata)`

Parameters

`metadata` (`FileMetadata`)

`rating__get_minimum(_)`

`rating__get_maximum(_)`

`rating__get_default(_)`

`tags__get(metadata)`

Parameters

`metadata` (`FileMetadata`)

`tags__add(metadata, by_tag_name=None, by_tag_string=None)`

Parameters

- `metadata` (`FileMetadata`)
- `by_tag_name` (`str` | `None`)
- `by_tag_string` (`str` | `None`)

`tags__drop(metadata, tag_name)`

Parameters

- `metadata` (`FileMetadata`)
- `tag_name` (`str`)

`misc__measure_geolocations_distance(_, geo_string_1, geo_string_2)`

Parameters

- `geo_string_1` (`str`)
- `geo_string_2` (`str`)

`misc__search(metadata, query_string)`

Parameters

`metadata` (`FileMetadata`)

`static _friendly_quote(s, encode_whitespace=False)`

`static _friendly_quote_as_file_url(s)`

`_exec(args)`

main()

class lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter._MyTestCase(*methodName='runTest'*)

Bases: `TestCase`

Create an instance of the class that will use the named test method when executed. Raises a `ValueError` if the instance does not have a method with the specified name.

not_existing_file = `'/proc///this/dummy/does/not/exist'`

_ustorage = `{}`

class MyMetadataEngine(*path*)

Bases: `XattrEngine`

list_xattrs()

Return a list of available extended attribute keys. Used internally. Without cache handling. You might add an own implementation of it (and some others) in a custom subclass for a custom xattr backend.

get_xattr(*key*)

Get an extended attribute. If key doesn't exist, it returns `''`. Used internally. Without cache handling. You might add an own implementation of it (and some others) in a custom subclass for a custom xattr backend.

set_xattr(*key, val*)

Set an extended attribute. Used internally. Without cache handling. You might add an own implementation of it (and some others) in a custom subclass for a custom xattr backend.

unset_xattr(*key*)

Unset an extended attribute. Used internally. Without cache handling. You might add an own implementation of it (and some others) in a custom subclass for a custom xattr backend.

class MySearchEngine(*root_dir, xattr_engine_type, **_*)

Bases: `SearchEngine`

Do not use directly. See `search_engine()`.

Parameters

- **root_dir** (*Path*) – The root directory to start searches in (as absolute path).
- **xattr_engine_type** (*type* [`XattrEngine`]) – The `FileMetadata.XattrEngine` subclass to use.

_list_dir(*path*)

Return a list of subitems of a directory. Used internally. Implement this in a custom search engine.

_is_dir(*path*)

Return whether a given path points to a directory. Used internally. Implement this in a custom search engine.

static metadata_for_file(*path*)

static search_engine(*root_dir*)

test_parse_and_generate_tag_string_to_from_tag_assignments()

test_create_tag_assignment_s_and_tag_assignments_s_from_scratch_and_assign_them()

test_tags_non_alphanumeric_characters()

```
test_high_level_tags()
test_high_level_comment()
test_high_level_rating()
test_high_level_geo()
test_interpreter_cache_xattr()
test_search_via_query_string_by_tags_comment_rating_geo()
test_search_via_search_criteria_by_combined()
test_search_convert_query_string_search_criterion()
_classSetupFailed = False
_class_cleanups = []
test_search_by_xattr()
test_search_query_string_empty()
test_search_tags_control_character()
test_cli_basic()
```

Submodules

lawwenda.asset.data module

Data resources.

`lawwenda.asset.data.find_data_file(name, search_dirs=None)`

Return the absolute path of a Lawwenda data file.

Can return None if no such file exists.

Parameters

- **name** (*str*) – The name of the file to find.
- **search_dirs** (*Iterable[str] | None*) – List of directories to look into. If not specified (as it usually should be), it looks at some usual places.

Return type

Path | None

`lawwenda.asset.data.readme_pdf(preferred_culture)`

Return the absolute path to the Lawwenda README file.

Parameters

preferred_culture (*str*) – The preferred culture.

Return type

Path

lawwenda.asset.project_info module**lawwenda.builtin package****Subpackages****lawwenda.builtin.fs_backends package****Submodules****lawwenda.builtin.fs_backends.local module**

Local filesystem backend.

class lawwenda.builtin.fs_backends.local.**Backend**(*root_path*)

Bases: [Backend](#)

A filesystem backend that resembles a particular subtree of your real local filesystem.

Do not use. See [lawwenda.fs.factory.filesystem\(\)](#).

Parameters

root_path (*Path* | *str*)

__path_to_system_path(*path*)

Parameters

path (*Path* | *str*)

Return type

Path

system_path(*handle*, *, *writable*)

Try to return an absolute path in the local filesystem for a node (in a handle).

This is optional and the default implementation returns `None`!

Parameters

- **handle** – The read or write handle to a node.
- **writable** – Whether you might do any write accesses to the result path.

child_names(*handle*)

Return all child names for a node (in a handle).

Parameters

handle – The read handle to a node.

is_hidden(*handle*)

Return whether a node (in a handle) is hidden.

Parameters

handle – The read handle to a node.

is_dir(*handle*)

Return whether a node (in a handle) is a directory.

This is also `True` for link nodes (see [is_link\(\)](#)) that point to a directory!

Parameters

handle – The read handle to a node.

is_file(*handle*)

Return whether a node (in a handle) is a regular file.

This is also `True` for link nodes (see [`is\_link\(\)`](#)) that point to a file!

Parameters

handle – The read handle to a node.

is_link(*handle*)

Return whether a node (in a handle) is a link. If this is a resolvable link, some of the other `is_` flags are `True` as well.

Resolving links is always done internally by the filesystem implementation. It is usually not required to know the link target in order to use the node.

Parameters

handle – The read handle to a node.

exists(*handle*)

Return whether a node (in a handle) points to something that actually exists.

This is also `True` for links that have no valid target.

Parameters

handle – The read handle to a node.

size(*handle*)

Return the size of a node (in a handle) in bytes.

Parameters

handle – The read handle to a node.

mtime(*handle*)

Return the ‘last modified’ time of a node (in a handle).

Parameters

handle – The read handle to a node.

comment(*handle*)

Return the comment assigned to a node (in a handle).

Parameters

handle – The read handle to a node.

rating(*handle*)

Return the rating assigned to a node (in a handle). Minimal valid value is 1, maximal is 10. 0 or `None` is considered “unrated”.

Parameters

handle – The read handle to a node.

tags(*handle*)

Return the tags that are assigned to a node (in a handle).

Parameters

handle – The read handle to a node.

geolocation(*handle*)

Return the geographic location associated to a node (in a handle).

Parameters

handle – The read handle to a node.

delete(*handle*)

Delete a node (in a handle).

Parameters

handle – The write handle to a node.

mkdir(*handle, exist_ok*)

Make a node (in a handle) an existing directory (but fail if the parent does not exist yet).

Parameters

- **handle** – The write handle to a node.
- **exist_ok** – Whether an already existing directory is considered no failure.

copy_to(*source, destination*)

Copy a node (in a handle) to another node (in a handle).

After copying, the destination node has similar characteristics as the source node, i.e. either is a file with the same content, or is a directory with the same subitems as the source.

This transfer conserves metadata like mtime and xattrs.

Parameters

- **source** – The read handle to the source node.
- **destination** – The write handle to the destination node.

move_to(*source, destination*)

Move a node (in a handle) to another node (in a handle).

This is similar to [copy_to\(\)](#); see there for more details.

Parameters

- **source** – The write handle to the source node.
- **destination** – The write handle to the destination node.

set_comment(*handle, comment*)

Set the comment for a node (in a handle).

Parameters

- **handle** – The write handle to a node.
- **comment** – The new comment.

set_geolocation(*handle, geolocation*)

Set the geographic location for a node (in a handle).

Parameters

- **handle** – The write handle to a node.
- **geolocation** – The new geographic location.

set_rating(*handle, rating*)

Set the rating for a node (in a handle).

Parameters

- **handle** – The write handle to a node.
- **rating** – The new rating (between 1 and 10 or None).

set_tags(*handle*, *tags*)

Set the tags for a node (in a handle).

Parameters

- **handle** – The write handle to a node.
- **tags** – The new tags.

read_file(*handle*)

Return a file-like object for reading content of a node (in a handle).

The caller must ensure that it gets closed after usage, usually by means of the Python `with` keyword.

Parameters

handle – The read handle to a node.

write_file(*handle*, *content*)

Write content to a node (in a handle).

This will overwrite its original content.

Parameters

- **handle** – The write handle to a node.
- **content** – The binary content to write to the node (can be either a byte array or an IO stream!).

known_tags()

Return a list of known tags of this filesystem, ordered by relevance. This list is used for additional features like UI auto completion.

_abc_impl = <_abc._abc_data object>

lawwenda.builtin.previewers package

Submodules

lawwenda.builtin.previewers.common_images module

Preview functionality for common image formats.

class lawwenda.builtin.previewers.common_images.**Previewer**

Bases: [Previewer](#)

Preview functionality for common image formats.

is_thumbnailable(*iar*)

Return if this previewer is able to generate a thumbnail for a file.

Note that this decision has to be made with very few information, e.g. without knowing the file content, for performance reasons.

See also [thumbnail\(\)](#).

Parameters

iar – An object that describes the file.

thumbnail(*node*)

Return a thumbnail for a file in PNG format.

See also [is_thumbnailable\(\)](#).

Parameters

node – The file to generate a thumbnail for.

is_previewable(iar)

Return if this previewer is able to generate an html preview snippet for a file.

Note that this decision has to be made with very few information, e.g. without knowing the file content, for performance reasons.

See also [preview_html\(\)](#).

Parameters

iar – An object that describes the file.

preview_html(node)

Return an html snippet that shows a preview of a file.

This is larger and richer in terms of flexibility than thumbnails, and is typically used by the file details panel.

See also [is_previewable\(\)](#).

Parameters

node – The file to preview.

_abc_impl = <_abc._abc_data object>

lawwenda.builtin.previewers.common_videos module

Preview functionality for common video formats.

class lawwenda.builtin.previewers.common_videos.**Previewer**

Bases: [Previewer](#)

Preview functionality for common video formats.

is_thumbnailable(iar)

Return if this previewer is able to generate a thumbnail for a file.

Note that this decision has to be made with very few information, e.g. without knowing the file content, for performance reasons.

See also [thumbnail\(\)](#).

Parameters

iar – An object that describes the file.

__ffmpeg_thumb(stdin=-3)**Parameters**

- **inputpath** (*str*)
- **stdin** (*Any*)

Return type

bytes

thumbnail(node)

Return a thumbnail for a file in PNG format.

See also [is_thumbnailable\(\)](#).

Parameters

node – The file to generate a thumbnail for.

is_previewable(*iar*)

Return if this previewer is able to generate an html preview snippet for a file.

Note that this decision has to be made with very few information, e.g. without knowing the file content, for performance reasons.

See also [preview_html\(\)](#).

Parameters

iar – An object that describes the file.

preview_html(*node*)

Return an html snippet that shows a preview of a file.

This is larger and richer in terms of flexibility than thumbnails, and is typically used by the file details panel.

See also [is_previewable\(\)](#).

Parameters

node – The file to preview.

_abc_impl = <_abc._abc_data object>

lawwenda.config package

User-side API for reading and modifying Lawwenda configurations, creating shares, and more.

For most cases, instantiate [Configuration](#) and use some of its methods.

class lawwenda.config.**Configuration**(*config_dir=None*)

Bases: object

Holds a Lawwenda configuration. A configuration typically holds some shares. Each running Lawwenda instance relates to a configuration, providing access to these shares as configured.

Parameters

config_dir (*Path* | *str* | *None*) – The path of the configuration directory. Will be created on demand if it does not exist. Defaults to a location that is usual for your operating system.

property config_dir: Path

The configuration directory.

peek_share_cache_tag(*name*)

Return an object that can be used later for detecting changes on the share configuration (by comparison against a later result).

Parameters

name (*str*) – The share name.

Return type

object | None

all_shares()

Return all shares that are currently part of this configuration.

Return type

[Sequence](#)[[Share](#)]

share_by_name(*name*)

Return the share by a name (or None if it does not exist).

Parameters

name (*str*)

Return type

Share | None

add_share(*path*, *, *name*, *password*, *title*=None, *readonly*=True, *hide_by_patterns*=(), *hide_by_tags*=(), *include_by_patterns*=None, *include_by_tags*=None, *exclude_by_patterns*=(), *exclude_by_tags*=(), *exclude_hidden*=False, *active_until*=None)

Add a new share.

Parameters

- **path** (*str*) – The directory to share. See [lawwenda.config.share.Share.path](#).
- **name** (*str*) – The unique name of the new share. See [lawwenda.config.share.Share.name](#).
- **password** (*str* | None) – The password to protect the share with.
- **title** (*str* | None) – The share title. See [lawwenda.config.share.Share.title](#).
- **readonly** (*bool*) – If to share in a read-only way. See [lawwenda.config.share.Share.readonly](#).
- **hide_by_patterns** (*Iterable[str]*) – See [lawwenda.config.share.Share.hide_by_patterns](#).
- **hide_by_tags** (*Iterable[str]*) – See [lawwenda.config.share.Share.hide_by_tags](#).
- **include_by_patterns** (*Iterable[str]* | None) – See [lawwenda.config.share.Share.include_by_patterns](#).
- **include_by_tags** (*Iterable[str]* | None) – See [lawwenda.config.share.Share.include_by_tags](#).
- **exclude_by_patterns** (*Iterable[str]*) – See [lawwenda.config.share.Share.exclude_by_patterns](#).
- **exclude_by_tags** (*Iterable[str]*) – See [lawwenda.config.share.Share.exclude_by_tags](#).
- **exclude_hidden** (*bool*) – See [lawwenda.config.share.Share.exclude_hidden](#).
- **active_until** (*datetime* | None) – The optional expiration time of the share. See [lawwenda.config.share.Share.active_until](#).

Return type

Share

remove_share(*name*)

Remove a share.

Parameters

name (*str*) – The name of the share to remove.

Return type

None

start_tiny_server()

Start a tiny local server for this configuration.

Such a server can be used for trying, development, testing, and so on, but is not recommended for real usage.

It will automatically find a free port and will return a control object that e.g. provides the full url and a method to stop it.

Return type
`TinyServer`

generate_wsgi()

Generates a wsgi script for hosting Lawwenda in a web server.

Read the ‘Installation’ section of the documentation for more details about what to do with it.

Return type
`str`

__verify_valid_name()

Parameters
name (`str`)

Return type
`None`

Submodules

lawwenda.config.share module

User-side API for reading and modifying Lawwenda configurations, creating shares, and more.

For most cases, instantiate `lawwenda.config.Configuration` and use some of its methods.

```
class lawwenda.config.share.Share(path, *, configuration, name, title, cache_tag, readonly=True,  
                                hide_by_patterns=(), hide_by_tags=(), include_by_patterns=None,  
                                include_by_tags=None, exclude_by_patterns=(), exclude_by_tags=(),  
                                exclude_hidden=False, password_scrypt=None, password_salt=None,  
                                active_until_timestamp=None)
```

Bases: `object`

A directory path together with some parameters (e.g. for access control) for sharing via Lawwenda. Read the documentation for more about shares.

Do not use. Instantiated by the Lawwenda infrastructure.

Parameters

- **path** (`Path` | `str`)
- **configuration** (`lawwenda.config.Configuration`)
- **name** (`str`)
- **title** (`str`)
- **cache_tag** (`object`)
- **readonly** (`bool`)
- **hide_by_patterns** (`Iterable[str]`)
- **hide_by_tags** (`Iterable[str]`)
- **include_by_patterns** (`Iterable[str]` | `None`)
- **include_by_tags** (`Iterable[str]` | `None`)
- **exclude_by_patterns** (`Iterable[str]`)

- **exclude_by_tags** (*Iterable[str]*)
- **exclude_hidden** (*bool*)
- **password_scrypt** (*str | None*)
- **password_salt** (*str | None*)
- **active_until_timestamp** (*float | None*)

property path: **str**

The path of the share's root directory.

property name: **str**

The share name.

This usually makes the last part of the url to this share. Is unique in the containing *configuration*.

property configuration: *lawwenda.config.Configuration*

The configuration that contains this share.

property title: **str**

The share title.

This is an arbitrary text shown in the window title. Should not contain line breaks and should be short.

property readonly: **bool**

If this share is restricted to only read actions (no removal, copying, uploading, editing, ... of the files and directories in *path*).

property hide_by_patterns: **Sequence[str]**

A list of regular expressions of paths for hiding.

A file or directory will be hidden if its path matches at least one of them. Those paths are always relative to *path*, always start with a '/', but never end with a one (unless it is the root path).

Note that hiding is not a security feature unless *exclude_hidden* is set.

property hide_by_tags: **Sequence[str]**

A list of tags for hiding files and directories.

A file or directory will be hidden if it is tagged with at least one of them.

Note that hiding is not a security feature unless *exclude_hidden* is set.

property include_by_patterns: **Sequence[str] | None**

A list of regular expressions of paths for including explicitly.

Those paths are always relative to *path*, always start with a '/', but never end with a one (unless it is the root path).

If this is specified, the share will switch from blacklist to whitelist. Everything that is not considered as included is implicitly considered as excluded.

property include_by_tags: **Sequence[str] | None**

A list of tags for including files and directories.

If this is specified, the share will switch from blacklist to whitelist. Everything that is not considered as included is implicitly considered as excluded.

property exclude_by_patterns: Sequence[str]

A list of regular expressions of paths for excluding.

A file or directory will be excluded if its path matches at least one of them. Those paths are always relative to [path](#), always start with a '/', but never end with a one (unless it is the root path).

Exclusions are enforced on backend side and not just a presentation aspect. There is no way for a client to work around that (unless there is a software bug).

property exclude_by_tags: Sequence[str]

A list of tags for excluding files and directories.

Exclusions are enforced on backend side and not just a presentation aspect. There is no way for a client to work around that (unless there is a software bug).

property exclude_hidden: bool

If to consider 'hidden' flags of files or directories as exclusions.

Exclusions are enforced on backend side and not just a presentation aspect. There is no way for a client to work around that (unless there is a software bug).

property password_scrypt: str | None

The scrypt hash of the password for this share.

Empty or None for disabled password protection. See also [password_salt](#).

property password_salt: str | None

The hash salt of the password for this share, if password protected.

See also [password_scrypt](#).

property active_until: datetime | None

The expiration time of this share, or None for infinite.

property active_until_timestamp: float | None

Same as [active_until](#), but as Unix timestamp.

property is_active: bool

If this share is currently active (e.g. not yet expired; see [active_until](#)).

property is_expired: bool

If this share is expired.

_to_dict()**lawwenda.fs package**

Lawwenda abstract filesystem.

Lawwenda usually shows a subtree of the local filesystem, but could also work with something completely different. This module implements the basic infrastructure for that.

See [lawwenda.fs.factory.filesystem\(\)](#).

lawwenda.fs.sanitize_path(path)

Sanitize slashes in a path and returns a path in the form /foo/bar. See [lawwenda.fs.node.Node.path](#).

Parameters

path (*str*) – The input path.

Return type

str

exception `lawwenda.fs.CircularTraversalError`Bases: `OSError`

Raised when traversing a tree that has a circle (i.e. that is not really a tree), usually by an ‘unfortunate’ link.

Subpackages**lawwenda.fs.previewer package**

Preview functionality.

Used by different parts of the Lawwenda user interface.

class `lawwenda.fs.previewer.Previewer`Bases: `ABC`

Base class for previewers.

Subclasses implement thumbnail and preview functionality for particular file formats.

thumbsize = `(300, 300)`**thumbwidth** = `300`**thumbheight** = `300`**abstractmethod** `is_thumbnailable(iar)`

Return if this previewer is able to generate a thumbnail for a file.

Note that this decision has to be made with very few information, e.g. without knowing the file content, for performance reasons.

See also `thumbnail()`.

Parameters

iar (`IsAbleRequest`) – An object that describes the file.

Return type

`bool`

abstractmethod `thumbnail(node)`

Return a thumbnail for a file in PNG format.

See also `is_thumbnailable()`.

Parameters

node (`lawwenda.fs.node.Node`) – The file to generate a thumbnail for.

Return type

`bytes`

abstractmethod `is_previewable(iar)`

Return if this previewer is able to generate an html preview snippet for a file.

Note that this decision has to be made with very few information, e.g. without knowing the file content, for performance reasons.

See also `preview_html()`.

Parameters

iar (`IsAbleRequest`) – An object that describes the file.

Return type

bool

abstractmethod preview_html(*node*)

Return an html snippet that shows a preview of a file.

This is larger and richer in terms of flexibility than thumbnails, and is typically used by the file details panel.

See also [*is_previewable\(\)*](#).

Parameters

node ([*lawwenda.fs.node.Node*](#)) – The file to preview.

Return type

str

_image_scale_to_thumbnail_size(*imgdata*)**Parameters**

imgdata (*BinaryIO*)

Return type*BytesIO*

_abc_impl = <_abc._abc_data object>

class lawwenda.fs.previewer.IsAbleRequest(*name, mimetype*)

Bases: object

Information about a particular file, used for asking a [*Previewer*](#) if it can provide some functionality for that file.

Instances of this class are used as an argument to some [*Previewer*](#) methods.

The available infos are not very rich for performance reasons.

Parameters

- **name** (*str*)
- **mimetype** (*str*)

name: str

mimetype: str

lawwenda.fs.search package

File search functionality.

Used by the Lawwenda user interface's 'Search' feature.

[*lawwenda.fs.search._query_by_tree_traversal\(*node, decide_func*\)*](#)

Parameters

- **node** ([*lawwenda.fs.node.Node*](#))
- **decide_func** (*Callable[[[*lawwenda.fs.node.Node*](#)], bool]*)

Return type

t.Iterable[[*lawwenda.fs.node.Node*](#)]

`lawwenda.fs.search._termstring_to_termlist(term)`

Parameters

term (*str*)

Return type

List[*str*]

class `lawwenda.fs.search.Search`

Bases: `ABC`

Base class for different search behaviors.

abstractmethod `query(node)`

Return a list of nodes that match the criteria of this search, starting from a given root node.

Parameters

node (`lawwenda.fs.node.Node`) – The root node. Query result nodes are usually inside that subtree.

Return type

t.Sequence[`lawwenda.fs.node.Node`]

`_abc_impl = <_abc._abc_data object>`

class `lawwenda.fs.search.DeeplySearch(term)`

Bases: `Search`

Deep searches look for search terms in some places, like file names or comments, but also in fulltext indexes or - as a fallback - directly in the content of each file.

This is a rather versatile search behavior, but is not precisely defined, and can be impractically slow if not fulltext index is available.

Parameters

term (*str*)

query(*node*)

Return a list of nodes that match the criteria of this search, starting from a given root node.

Parameters

node – The root node. Query result nodes are usually inside that subtree.

__try_get_index(*node*)

Parameters

node (`lawwenda.fs.node.Node`)

Return type

object

__query_indexed(*idxdata*)

Parameters

- **node** (`lawwenda.fs.node.Node`)
- **idxdata** (*object*)

Return type

t.Iterable[`lawwenda.fs.node.Node`]

__query_unindexed(*node*)

Parameters

node ([lawwenda.fs.node.Node](#))

Return type

`t.Iterable[lawwenda.fs.node.Node]`

__decide_unindexed(*node*)

Parameters

node ([lawwenda.fs.node.Node](#))

Return type

`bool`

__decide_unindexed_by_metadata(*node*)

Parameters

node ([lawwenda.fs.node.Node](#))

Return type

`bool`

__decide_unindexed_by_content(*node*)

Parameters

node ([lawwenda.fs.node.Node](#))

Return type

`bool`

_abc_impl = `<_abc._abc_data object>`

class [lawwenda.fs.search.ByNameSearch](#)(*term*)

Bases: [Search](#)

Searches for search terms in file names.

See also [lawwenda.fs.node.Node.name](#).

Parameters

term (`str`)

query(*node*)

Return a list of nodes that match the criteria of this search, starting from a given root node.

Parameters

node – The root node. Query result nodes are usually inside that subtree.

__decide(*node*)

Parameters

node ([lawwenda.fs.node.Node](#))

Return type

`bool`

_abc_impl = `<_abc._abc_data object>`

class lawwenda.fs.search.ByTagSearch(*term*)

Bases: [Search](#)

Searches for files that have particular tags.

See also [lawwenda.fs.node.Node.tags](#).

Parameters

term (*str*)

query(*node*)

Return a list of nodes that match the criteria of this search, starting from a given root node.

Parameters

node – The root node. Query result nodes are usually inside that subtree.

__decide(*node*)

Parameters

node ([lawwenda.fs.node.Node](#))

Return type

bool

_abc_impl = <_abc._abc_data object>

class lawwenda.fs.search.ByGeoSearch(*position*, *, *radius*)

Bases: [Search](#)

Searches for files that are associated with a particular geographic region.

See also [lawwenda.fs.node.Node.geolocation](#).

Parameters

- **position** (*str*)
- **radius** (*str* | *float*)

query(*node*)

Return a list of nodes that match the criteria of this search, starting from a given root node.

Parameters

node – The root node. Query result nodes are usually inside that subtree.

__decide(*node*)

Parameters

node ([lawwenda.fs.node.Node](#))

Return type

bool

_abc_impl = <_abc._abc_data object>

lawwenda.fs.search.**create_search**(*mode*, ***kwargs*)

Return a Search object with a particular configuration.

Parameters

- **mode** (*str*) – The search mode string.
- **kwargs** – Additional search arguments, specific to the chosen mode.

Return type[Search](#)**Submodules****lawwenda.fs.backend module**

Lawwenda filesystem backend foundation.

class lawwenda.fs.backend.Backend

Bases: ABC

Base class for filesystem backend implementations.

Subclasses implement different kinds of filesystems, e.g. [lawwenda.builtin.fs_backends.local.Backend](#).

DO NOT USE! The interface of this class is relevant for implementing custom filesystems only! See [lawwenda.fs.node.Node](#) instead. Otherwise, things will break, even in security related ways!

Do not use. See [lawwenda.fs.factory.filesystem\(\)](#).

class ReadHandle(path)

Bases: object

Read and write handles are just a stupid container that hold one [lawwenda.fs.node.Node](#).

This looks stupid at first, because you could just use this node directly instead. The added value of handles are a central mechanism for access control, which would be a bit less obvious and more scattered in code without this indirection.

Of course, it cannot avoid a way around it in code. An attacker that can change the code has won anyway. It just simplifies writing correct code that hopefully does not provide ways around it for the client.

See [Backend.read_handle\(\)](#) and [Backend.write_handle\(\)](#).

Parameters

path (*str*)

class WriteHandle(path)

Bases: [ReadHandle](#)

See [Backend.ReadHandle](#).

Parameters

path (*str*)

read_handle(node)

Return a read handle for a node.

Such handles are needed for executing read actions on that node.

For write handles, see [write_handle\(\)](#).

Note: These methods implements access control (typically by putting wrappers with more restrictive implementations around it; see e.g. [lawwenda.fs.wrappers.ReadOnlyWrappedBackend](#)).

Parameters

node (*Node*) – The node to read from later.

Return type

[ReadHandle](#)

write_handle(*node*)

Return a write handle for a node.

Such handles are needed for executing write actions on that node.

For read handles, see [read_handle\(\)](#).

Note: These methods implements access control (typically by putting wrappers with more restrictive implementations around it; see e.g. [lawwenda.fs.wrappers.ReadOnlyWrappedBackend](#)).

Parameters

node ([Node](#)) – The node to write from later.

Return type

[WriteHandle](#)

system_path(*handle*, *, *writable*)

Try to return an absolute path in the local filesystem for a node (in a handle).

This is optional and the default implementation returns `None`!

Parameters

- **handle** ([ReadHandle](#) / [WriteHandle](#)) – The read or write handle to a node.
- **writable** (*bool*) – Whether you might do any write accesses to the result path.

Return type

Path | `None`

abstractmethod child_names(*handle*)

Return all child names for a node (in a handle).

Parameters

handle ([ReadHandle](#)) – The read handle to a node.

Return type

Iterable[*str*]

abstractmethod is_hidden(*handle*)

Return whether a node (in a handle) is hidden.

Parameters

handle ([ReadHandle](#)) – The read handle to a node.

Return type

bool

abstractmethod is_dir(*handle*)

Return whether a node (in a handle) is a directory.

This is also `True` for link nodes (see [is_link\(\)](#)) that point to a directory!

Parameters

handle ([ReadHandle](#)) – The read handle to a node.

Return type

bool

abstractmethod is_file(*handle*)

Return whether a node (in a handle) is a regular file.

This is also `True` for link nodes (see [is_link\(\)](#)) that point to a file!

Parameters

handle ([ReadHandle](#)) – The read handle to a node.

Return type

bool

abstractmethod is_link(handle)

Return whether a node (in a handle) is a link. If this is a resolvable link, some of the other `is_` flags are True as well.

Resolving links is always done internally by the filesystem implementation. It is usually not required to know the link target in order to use the node.

Parameters

handle ([ReadHandle](#)) – The read handle to a node.

Return type

bool

abstractmethod exists(handle)

Return whether a node (in a handle) points to something that actually exists.

This is also True for links that have no valid target.

Parameters

handle ([ReadHandle](#)) – The read handle to a node.

Return type

bool

abstractmethod size(handle)

Return the size of a node (in a handle) in bytes.

Parameters

handle ([ReadHandle](#)) – The read handle to a node.

Return type

int

mimetype(handle)

Return the mimetype of a node (in a handle).

Parameters

handle ([ReadHandle](#)) – The read handle to a node.

Return type

str

abstractmethod mtime(handle)

Return the ‘last modified’ time of a node (in a handle).

Parameters

handle ([ReadHandle](#)) – The read handle to a node.

Return type

datetime

has_thumbnail(handle)

Return whether there could be a thumbnail available for a node (in a handle).

See also [thumbnail\(\)](#).

There might be cases when it returns True but the thumbnail generation will fail.

Parameters**handle** ([ReadHandle](#)) – The read handle to a node.**Return type**

bool

has_preview(*handle*)

Return whether there could be an HTML preview snippet available for a node (in a handle).

See also [preview_html\(\)](#).

There might be cases when it returns True but the preview generation will fail.

Parameters**handle** ([ReadHandle](#)) – The read handle to a node.**Return type**

bool

preview_html(*handle*)

Return an HTML snippet that shows a preview of a node (in a handle).

This is larger and richer in terms of flexibility than thumbnails, and is typically used by the file details panel.

See also [has_preview\(\)](#).**Parameters****handle** ([ReadHandle](#)) – The read handle to a node.**Return type**

str

thumbnail(*handle*)

Return a thumbnail for a node (in a handle) in PNG format.

See also [has_thumbnail\(\)](#).**Parameters****handle** ([ReadHandle](#)) – The read handle to a node.**Return type**

bytes

abstractmethod comment(*handle*)

Return the comment assigned to a node (in a handle).

Parameters**handle** ([ReadHandle](#)) – The read handle to a node.**Return type**

str

abstractmethod rating(*handle*)

Return the rating assigned to a node (in a handle). Minimal valid value is 1, maximal is 10. 0 or None is considered “unrated”.

Parameters**handle** ([ReadHandle](#)) – The read handle to a node.**Return type**

int | None

abstractmethod tags(*handle*)

Return the tags that are assigned to a node (in a handle).

Parameters

handle ([ReadHandle](#)) – The read handle to a node.

Return type

Iterable[str]

abstractmethod geolocation(*handle*)

Return the geographic location associated to a node (in a handle).

Parameters

handle ([ReadHandle](#)) – The read handle to a node.

Return type

[Geolocation](#) | str | None

abstractmethod delete(*handle*)

Delete a node (in a handle).

Parameters

handle ([WriteHandle](#)) – The write handle to a node.

Return type

None

abstractmethod mkdir(*handle*, *exist_ok*)

Make a node (in a handle) an existing directory (but fail if the parent does not exist yet).

Parameters

- **handle** ([WriteHandle](#)) – The write handle to a node.
- **exist_ok** (*bool*) – Whether an already existing directory is considered no failure.

Return type

None

abstractmethod copy_to(*source*, *destination*)

Copy a node (in a handle) to another node (in a handle).

After copying, the destination node has similar characteristics as the source node, i.e. either is a file with the same content, or is a directory with the same subitems as the source.

This transfer conserves metadata like mtime and xattrs.

Parameters

- **source** ([ReadHandle](#)) – The read handle to the source node.
- **destination** ([WriteHandle](#)) – The write handle to the destination node.

Return type

None

abstractmethod move_to(*source*, *destination*)

Move a node (in a handle) to another node (in a handle).

This is similar to [copy_to\(\)](#); see there for more details.

Parameters

- **source** ([WriteHandle](#)) – The write handle to the source node.

- **destination** ([WriteHandle](#)) – The write handle to the destination node.

Return type

None

abstractmethod `set_comment(handle, comment)`

Set the comment for a node (in a handle).

Parameters

- **handle** ([WriteHandle](#)) – The write handle to a node.
- **comment** (*str*) – The new comment.

Return type

None

abstractmethod `set_geolocation(handle, geolocation)`

Set the geographic location for a node (in a handle).

Parameters

- **handle** ([WriteHandle](#)) – The write handle to a node.
- **geolocation** ([Geolocation](#) / *None*) – The new geographic location.

Return type

None

abstractmethod `set_rating(handle, rating)`

Set the rating for a node (in a handle).

Parameters

- **handle** ([WriteHandle](#)) – The write handle to a node.
- **rating** (*int* / *None*) – The new rating (between 1 and 10 or *None*).

Return type

None

abstractmethod `set_tags(handle, tags)`

Set the tags for a node (in a handle).

Parameters

- **handle** ([WriteHandle](#)) – The write handle to a node.
- **tags** (*Sequence[str]*) – The new tags.

Return type

None

abstractmethod `read_file(handle)`

Return a file-like object for reading content of a node (in a handle).

The caller must ensure that it gets closed after usage, usually by means of the Python `with` keyword.**Parameters**

- handle** ([ReadHandle](#)) – The read handle to a node.

Return type*BinaryIO*

abstractmethod `write_file(handle, content)`

Write content to a node (in a handle).

This will overwrite its original content.

Parameters

- **handle** ([WriteHandle](#)) – The write handle to a node.
- **content** (*bytes* | *BinaryIO*) – The binary content to write to the node (can be either a byte array or an IO stream!).

Return type

None

abstractmethod `known_tags()`

Return a list of known tags of this filesystem, ordered by relevance. This list is used for additional features like UI auto completion.

Return type

Iterable[*str*]

_node_by_path(path)

Return a node by a path.

Used internally by implementations in order to get access to node functionality (e.g. in deep copy operations).

Parameters

path (*str*) – The path.

Return type

[Node](#)

`_abc_impl = <_abc._abc_data object>`

lawwenda.fs.data module

Data structures for Lawwenda filesystems.

lawwenda.fs.factory module

Creating Lawwenda filesystems.

`lawwenda.fs.factory.filesystem(root_path, *, readonly=False, hide_by_patterns=(), hide_by_tags=(), include_by_patterns=None, include_by_tags=None, exclude_by_patterns=(), exclude_by_tags=(), exclude_hidden=False)`

Create a filesystem resembling a particular subtree of your real local filesystem, with some configuration for access control and more, and return its root node.

Parameters

- **root_path** (*str*) – The path from your real local filesystem to consider as the root directory.
- **readonly** (*bool*) – See [lawwenda.config.share.Share.readonly](#).
- **hide_by_patterns** (*Iterable*[*str*]) – See [lawwenda.config.share.Share.hide_by_patterns](#).
- **hide_by_tags** (*Iterable*[*str*]) – See [lawwenda.config.share.Share.hide_by_tags](#).

- **include_by_patterns** (*Iterable[str] | None*) – See [lawwenda.config.share.Share.include_by_patterns](#).
- **include_by_tags** (*Iterable[str] | None*) – See [lawwenda.config.share.Share.include_by_tags](#).
- **exclude_by_patterns** (*Iterable[str]*) – See [lawwenda.config.share.Share.exclude_by_patterns](#).
- **exclude_by_tags** (*Iterable[str]*) – See [lawwenda.config.share.Share.exclude_by_tags](#).
- **exclude_hidden** (*bool*) – See [lawwenda.config.share.Share.exclude_hidden](#).

Return type[Node](#)

```
lawwenda.fs.factory._configured_backend(filesystem, *, readonly=False, hide_by_patterns=(),
                                         hide_by_tags=(), include_by_patterns=None,
                                         include_by_tags=None, exclude_by_patterns=(),
                                         exclude_by_tags=(), exclude_hidden=False)
```

Decorates a filesystem with some access control and more.

Parameters

- **filesystem** ([Backend](#)) – The filesystem to decorate.
- **readonly** (*bool*) – See [lawwenda.config.share.Share.readonly](#).
- **hide_by_patterns** (*Iterable[str]*) – See [lawwenda.config.share.Share.hide_by_patterns](#).
- **hide_by_tags** (*Iterable[str]*) – See [lawwenda.config.share.Share.hide_by_tags](#).
- **include_by_patterns** (*Iterable[str] | None*) – See [lawwenda.config.share.Share.include_by_patterns](#).
- **include_by_tags** (*Iterable[str] | None*) – See [lawwenda.config.share.Share.include_by_tags](#).
- **exclude_by_patterns** (*Iterable[str]*) – See [lawwenda.config.share.Share.exclude_by_patterns](#).
- **exclude_by_tags** (*Iterable[str]*) – See [lawwenda.config.share.Share.exclude_by_tags](#).
- **exclude_hidden** (*bool*) – See [lawwenda.config.share.Share.exclude_hidden](#).

Return type[Backend](#)

```
lawwenda.fs.factory._any(predicates)
```

Return a predicate that returns True iff at least one of the input predicates return True.

Parameters

predicates (*Iterable[Callable[[[Node](#)], bool]]*) – The input predicates.

Return type*Callable[[[Node](#)], bool]*

```
lawwenda.fs.factory._not(predicate)
```

Return a predicate that inverts the input predicate.

Parameters

predicate (*Callable*[[[Node](#)], *bool*]) – The input predicate.

Return type

Callable[[[Node](#)], *bool*]

`lawwenda.fs.factory._descending(predicate)`

Return a predicate that returns True iff the input predicate returns True for the input node and/or some nodes in its subtree (if it is a directory).

Parameters

predicate (*Callable*[[[Node](#)], *bool*]) – The input predicate.

Return type

Callable[[[Node](#)], *bool*]

`lawwenda.fs.factory._path_matches_regexp(regex)`

Return a predicate that returns True for input nodes whose `node.Node.path` fully matches the given regular expression.

Parameters

regex (*str*) – The regular expression string.

Return type

Callable[[[Node](#)], *bool*]

`lawwenda.fs.factory._has_tag(tag)`

Return a predicate that returns True for input nodes whose `node.Node.tags` contain the given tag.

Parameters

tag (*str*) – The tag to look for.

Return type

Callable[[[Node](#)], *bool*]

lawwenda.fs.node module

Lawwenda filesystem nodes.

class `lawwenda.fs.node.Node`(*path*, *, *backend*)

Bases: `object`

A filesystem node. It represents a particular filesystem item (a file, directory, link, ...) in a particular filesystem, by some path. This may be an item that does not even exist (but maybe e.g. is going to be created).

This interface provides all filesystem operations. All parts of code in Lawwenda and its plugins must use this interface.

In order to get a filesystem node, start with a root node (see `lawwenda.fs.factory.filesystem()`). Then use attributes and methods like `child_nodes` or `child_by_name()`.

Do not use.

Parameters

- **path** (*str*)
- **backend** (`lawwenda.fs.backend.Backend`)

property path: str

The path of this node (like `"/foo/bar"` or `"/"`).

This is similar to a Unix filesystem path, i.e. path segments are separated by `"/"`.

This path will always be considered as relative to the root node of the filesystem (see [backend](#)) it is part of. It is not relative to `/` of your real filesystem (unless you have explicitly chosen a filesystem that resembles your entire real filesystem).

In detail, a path has the following pattern:

- with a slash in the beginning
- without a slash at the end (exception: root path)
- without double slashes
- with `..` and `.` resolved
- root path: `/`

property backend: Backend

The filesystem backend (a.k.a. “the filesystem”) this node is part of.

property name: str

The file name of the node (empty string for the root node).

This is the last segment of [path](#).

property dirpath: str

The directory path of the node.

This is all but the last segment of [path](#). Same as [path](#) of [parent_node](#) (but `"/"` for the root node).

property is_writable: bool

Whether the node is writable.

property is_hidden: bool

Whether the node is hidden.

property is_dir: bool

Whether the node is a directory.

This is also True for link nodes (see [is_link\(\)](#)) that point to a directory!

property is_file: bool

Whether the node is a regular file.

This is also True for link nodes (see [is_link\(\)](#)) that point to a directory!

property is_link: bool

Whether the node is a link. If this is a resolvable link, some of the other `is_` flags are True as well.

Resolving links is always done internally by the filesystem implementation. It is usually not required to know the link target in order to use the node.

property exists: bool

Whether a node points to something that actually exists.

This can e.g. be False for nodes coming e.g. from [child_by_name\(\)](#).

property size: int

The size of this node in bytes.

property mimetype: str

The mimetype of this node.

property mtime: datetime

The 'last modified' time of this node.

property mtime_ts: float

Same as [mtime](#), but as Unix timestamp.

property icon_name: str | None

The recommended icon name for this node.

property has_thumbnail: bool

Whether there could be a thumbnail available for this node.

See also [thumbnail\(\)](#).

There might be cases when it returns True but the thumbnail generation will fail.

property has_preview: bool

Whether there could be a html preview snippet available for this node.

See also [preview_html](#).

There might be cases when it returns True but the preview generation will fail.

property comment: str

The node comment text.

property rating: int | None

The node rating. Between 1 and 10 or None.

property tags: Sequence[str]

The tags assigned to this node. See also [all_tags](#).

property all_tags: Sequence[str]

The tags assigned to this node or any ancestor. See also [tags](#).

property tags_string: str

The tags assigned to this node, encoded in one string.

property geolocation: [Geolocation](#) | None

The geographic location associated to this node.

property geolocation_string: str

The geographic location associated to this node as geo string.

property basics_as_dict

Basic node data as dict.

This is solely used internally for serialization.

property full_as_dict

Complete node data as dict.

This is solely used internally for serialization.

property preview_html: str

An HTML snippet that shows a preview of this node.

This is larger and richer in terms of flexibility than thumbnails, and is typically used by the file details panel.

See also [has_preview](#).

property child_nodes: Sequence[Node]

The list of child nodes, i.e. nodes for all files and subdirectories inside this node.

This only makes sense on directory nodes and will be empty otherwise.

property parent_node: Node | None

The parent node.

This is None for the root node.

traverse_dir(*, raise_on_circle, param_path="")

Return paths from this node and all descendants (i.e. the subtree).

Parameters

- **raise_on_circle** (*bool*) – If to raise an exception when there is a circle (due to links), instead of silently skipping and continuing.
- **param_path** (*str*) – Prefix for all output paths.

Return type

Iterable[Tuple[Node, str]]

system_path(*, writable)

Try to return an absolute path in the local filesystem for this node.

You should usually not need this method, you should avoid to use it as good as you can, and when you use it, you must be very careful! This is due to the following reasons:

- It allows you to get a path from a filesystem that is shared in a read-only way, but then accidentally make write accesses to it. Such a bug would obviously lead to a disastrous security hole. Avoid that by taking care of the `writable` argument!
- Some `:py:class`Filesystem`s` might not support that at all and return `None`.
- The method signature could change in later versions due to new access control features.

Parameters

writable (*bool*) – Whether you might do any write accesses to the result path. Never set to `False` without understanding the security implications mentioned above.

Return type

Path | None

child_by_name(name)

Return a child node by name.

This will not fail for names that do not exist yet, but return a node that could be used for creating it.

Parameters

name (*str*) – The file name of the child.

Return type

[Node](#)

child_by_path(*path*)

Return a child node by path.

This will not fail for names that do not exist yet, but return a node that could be used for creating it.

Parameters

path (*str*) – The relative file path of the child.

Return type

Node

read_file()

Return a file-like object for reading content of this node.

The caller must ensure that it gets closed after usage, usually by means of the Python `with` keyword.

Return type

BinaryIO

thumbnail()

The thumbnail for this node in PNG format.

See also [*has_thumbnail*](#).

Return type

bytes

delete()

Delete this node.

Return type

None

mkdir(*, exist_ok=True)

Make this node an existing directory (but fail if the parent does not exist yet).

Parameters

exist_ok (*bool*) – Whether an already existing directory is considered no failure.

Return type

None

copy_to(*newpath*)

Copy this node to a destination path.

After copying, the destination has similar characteristics as this node, i.e. either is a file with the same content, or is a directory with the same subitems.

Parameters

newpath (*str*) – The destination path.

Return type

None

move_to(*newpath*)

Move this node to a destination path.

After moving, the destination has similar characteristics as this node had, i.e. either is a file with the same content, or is a directory with the same subitems.

Parameters

newpath (*str*) – The destination path.

Return type

None

set_comment(*comment*)

Set the comment for this node.

Parameters**comment** (*str* / *None*) – The new comment.**Return type**

None

set_geolocation(*geolocation*)

Set the geographic location for this node.

Parameters**geolocation** (*Geolocation* / *str* / *None*) – The new geographic location.**Return type**

None

set_rating(*rating*)Set the rating for this node. Minimal valid value is 1, maximal is 10. 0 or *None* is considered “unrated”.**Parameters****rating** (*int* / *None*) – The new rating.**Return type**

None

set_tags(*tags*)

Set the tags for this node.

Parameters**tags** (*Iterable[str]*) – The new tags.**Return type**

None

write_file(*content*)

Write content to this node.

This will overwrite its original content.

Parameters**content** (*bytes* / *BinaryIO*) – The binary content to write to the node.**Return type**

None

known_tags()

Return a list of known tags of this filesystem, ordered by relevance. This list is used for additional features like UI auto completion.

Return type*Sequence*[*str*]

lawwenda.fs.wrappers module

Backend wrappers.

Used for filtering nodes from existing filesystems, influence access control or other things.

class lawwenda.fs.wrappers._**WrappedBackend**(*inner*)

Bases: *Backend*

Base class for backend wrappers. They provide custom implementations for some methods, but forward other calls to their inner backend.

Do not use. See *lawwenda.fs.factory.filesystem()*.

Parameters

inner (*Backend*)

_eval_predicate(*predicate*, *path*)

Parameters

- **predicate** (*Callable*[[*Node*], *bool*])
- **path** (*str*)

Return type

bool

_abc_impl = <_abc._abc_data object>

class lawwenda.fs.wrappers.**HidingNodesWrappedBackend**(*inner*, *predicate*)

Bases: *_WrappedBackend*

Backend wrapper that marks some nodes as hidden by a predicate (on top of the nodes that are already hidden by the filesystem implementation itself, e.g. file names that start with a dot).

Note: See user can see and access hidden files if he decides so. For access control purposes, just might want to exclude them instead!

Parameters

- **inner** (*Backend*) – The inner filesystem.
- **predicate** (*Callable*[[*Node*], *bool*]) – The predicate that defines what nodes to hide.

is_hidden(*handle*)

Return whether a node (in a handle) is hidden.

Parameters

handle – The read handle to a node.

_abc_impl = <_abc._abc_data object>

class lawwenda.fs.wrappers.**ExcludingNodesWrappedBackend**(*inner*, *predicate*)

Bases: *_WrappedBackend*

Backend wrapper that forcefully excludes some nodes.

Parameters

- **inner** (*Backend*) – The inner filesystem.
- **predicate** (*Callable*[[*Node*], *bool*]) – The predicate that defines what nodes to exclude.

child_names(*handle*)

Return all child names for a node (in a handle).

Parameters

handle – The read handle to a node.

read_handle(*node*)

Return a read handle for a node.

Such handles are needed for executing read actions on that node.

For write handles, see [write_handle\(\)](#).

Note: These methods implements access control (typically by putting wrappers with more restrictive implementations around it; see e.g. [lawwenda.fs.wrappers.ReadOnlyWrappedBackend](#)).

Parameters

node – The node to read from later.

write_handle(*node*)

Return a write handle for a node.

Such handles are needed for executing write actions on that node.

For read handles, see [read_handle\(\)](#).

Note: These methods implements access control (typically by putting wrappers with more restrictive implementations around it; see e.g. [lawwenda.fs.wrappers.ReadOnlyWrappedBackend](#)).

Parameters

node – The node to write from later.

__is_excluded(*node*)**Parameters**

node ([Node](#))

Return type

bool

_abc_impl = <**_abc._abc_data** object>

class [lawwenda.fs.wrappers.ReadOnlyWrappedBackend](#)(*inner*)

Bases: [_WrappedBackend](#)

Backend wrapper that blocks all write accesses, making it a readonly filesystem.

Do not use. See [lawwenda.fs.factory.filesystem\(\)](#).

Parameters

inner ([Backend](#))

write_handle(*node*)

Return a write handle for a node.

Such handles are needed for executing write actions on that node.

For read handles, see [read_handle\(\)](#).

Note: These methods implements access control (typically by putting wrappers with more restrictive implementations around it; see e.g. [lawwenda.fs.wrappers.ReadOnlyWrappedBackend](#)).

Parameters

node – The node to write from later.

```
_abc_impl = <_abc._abc_data object>
```

lawwenda.utils package

Submodules

lawwenda.utils.tiny_server module

Tiny local server for trying, development, testing.

`lawwenda.utils.tiny_server.start_tiny_server(configuration)`

Start a tiny local server for a given configuration.

Such a server can be used for trying, development, testing, and so on, but is not recommended for real usage.

It will automatically find a free port and will return a control object that e.g. provides the full url and a method to stop it.

Parameters

configuration ([Configuration](#)) – The configuration to run with.

Return type

[TinyServer](#)

`lawwenda.utils.tiny_server.running_tiny_servers()`

Return the servers started by [start_tiny_server\(\)](#) that are currently running.

Return type

[Sequence](#)[[TinyServer](#)]

`lawwenda.utils.tiny_server._start_tiny_server_for_app(app)`

Start a tiny local server for a given wsgi application.

Parameters

app ([Callable](#)) – The wsgi application to run.

Return type

[TinyServer](#)

class `lawwenda.utils.tiny_server._TinyServerThread(wsgi_server)`

Bases: [Thread](#)

This constructor should always be called with keyword arguments. Arguments are:

group should be `None`; reserved for future extension when a [ThreadGroup](#) class is implemented.

target is the callable object to be invoked by the `run()` method. Defaults to `None`, meaning nothing is called.

name is the thread name. By default, a unique name is constructed of the form “Thread-N” where N is a small decimal number.

args is a list or tuple of arguments for the target invocation. Defaults to `()`.

kwargs is a dictionary of keyword arguments for the target invocation. Defaults to `{}`.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (`Thread.__init__()`) before doing anything else to the thread.

run()

Method representing the thread’s activity.

You may override this method in a subclass. The standard `run()` method invokes the callable object passed to the object’s constructor as the target argument, if any, with sequential and keyword arguments taken from the `args` and `kwargs` arguments, respectively.

```
class lawwenda.utils.tiny_server.TinyServer(wsgi_server, server_thread)
```

Bases: object

A tiny server.

```
property url: str
```

The url of this running server.

```
shutdown()
```

Stop this server.

Return type

None

```
wait_stopped()
```

Wait until this server stopped.

Return type

None

7.1.2 Submodules

7.1.3 lawwenda.lawwenda_cli module

The Lawwenda CLI.

```
lawwenda.lawwenda_cli.main()
```

CLI main routine.

```
lawwenda.lawwenda_cli.parser(*, only_documentation=True)
```

Return an `argparse.ArgumentParser`.

Parameters

only_documentation (*bool*) – Whether to include only stuff that is relevant for documentation.

Return type

ArgumentParser

```
class lawwenda.lawwenda_cli.Commands(config, **_)
```

Bases: object

CLI commands.

```
console(**_)
```

Return type

None

```
run_tiny_server(**_)
```

Return type

None

```
list_shares(**_)
```

Return type

str

```
describe_share(*, name, **_)
```

Parameters

name (*str*)

Return type

str

add_share(**name*, *path*, *title*, *active_until*, *hide_by_pattern*, *hide_by_tag*, *include_by_pattern*, *include_by_tag*, *exclude_by_pattern*, *exclude_by_tag*, *exclude_hidden*, *readonly*, **_)

Parameters

- **name** (*str*)
- **path** (*str*)
- **title** (*str*)
- **active_until** (*str*)
- **hide_by_pattern** (*Sequence[str] | None*)
- **hide_by_tag** (*Sequence[str] | None*)
- **include_by_pattern** (*Sequence[str] | None*)
- **include_by_tag** (*Sequence[str] | None*)
- **exclude_by_pattern** (*Sequence[str] | None*)
- **exclude_by_tag** (*Sequence[str] | None*)
- **exclude_hidden** (*bool*)
- **readonly** (*bool*)

Return type

None

remove_share(**name*, **_)

Parameters**name** (*str*)**Return type**

None

generate_wsgi(**_)

Return type

str

__quickstart(*config_path*)

Parameters**config_path** (*Path*)**Return type**

str

__quickstart__format_command(*txt*)

Parameters**txt** (*str*)**Return type**

str

PYTHON MODULE INDEX

|
lawwenda, 15
lawwenda.apps, 15
lawwenda.apps.file_manager_app, 15
lawwenda.apps.file_manager_app.endpoints, 17
lawwenda.apps.file_manager_app.temp_zip_files,
17
lawwenda.apps.file_manager_app.template_renderer,
18
lawwenda.apps.http_webdav_app, 19
lawwenda.apps.http_webdav_app.dav_prop, 20
lawwenda.apps.http_webdav_app.endpoints, 23
lawwenda.apps.main_app, 26
lawwenda.asset, 27
lawwenda.asset.data, 46
lawwenda.asset.PiMetadataInterpreter, 27
lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter,
27
lawwenda.asset.project_info, 47
lawwenda.builtin, 47
lawwenda.builtin.fs_backends, 47
lawwenda.builtin.fs_backends.local, 47
lawwenda.builtin.previewers, 50
lawwenda.builtin.previewers.common_images, 50
lawwenda.builtin.previewers.common_videos, 51
lawwenda.config, 52
lawwenda.config.share, 54
lawwenda.fs, 56
lawwenda.fs.backend, 62
lawwenda.fs.data, 68
lawwenda.fs.factory, 68
lawwenda.fs.node, 70
lawwenda.fs.previewer, 57
lawwenda.fs.search, 58
lawwenda.fs.wrappers, 76
lawwenda.lawwenda_cli, 79
lawwenda.utils, 78
lawwenda.utils.tiny_server, 78

INDEX

Symbols

`__api_thumbnail()` (lawwenda.apps.file_manager_app.endpoints.Endpoints method), 17
`__api_upload()` (lawwenda.apps.file_manager_app.endpoints.Endpoints method), 17
`__api_zip()` (lawwenda.apps.file_manager_app.endpoints.Endpoints method), 17
`__api_zip_content()` (lawwenda.apps.main_app.MainApp method),
`__authenticcate()` (lawwenda.apps.file_manager_app.FileManagerApp method), 17
`__cleanup_loop()` (lawwenda.apps.file_manager_app.temp_zip_files.TempZipFiles._TempZip static method), 18
`__copy_move()` (lawwenda.apps.http_webdav_app.endpoints.Endpoints method), 25
`__decide()` (lawwenda.fs.search.ByGeoSearch method), 61
`__decide_unindexed()` (lawwenda.fs.search.DeeplySearch method), 60
`__decide_unindexed_by_content()` (lawwenda.fs.search.DeeplySearch method), 60
`__decide_unindexed_by_metadata()` (lawwenda.fs.search.DeeplySearch method), 60
`__delete()` (lawwenda.apps.http_webdav_app.endpoints.Endpoints method), 24
`__dispatch_request()` (lawwenda.apps.file_manager_app.FileManagerApp method), 16
`__dispatch_request_internals()` (lawwenda.apps.file_manager_app.FileManagerApp method), 16
`__dispatch_request_normal()` (lawwenda.apps.file_manager_app.FileManagerApp method), 16
`__ensure_authorized()` (lawwenda.apps.file_manager_app.FileManagerApp method), 16
`__ffmpeg_thumb()` (lawwenda.builtin.previewers.common_videos.PreviewVideos._PreviewVideos method), 16
`__get()` (lawwenda.apps.http_webdav_app.endpoints.Endpoints method), 23
`__get_app_for_share()` (lawwenda.apps.main_app.MainApp method),
`__remove_assignments_by_tag_name()` (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.PiMetadata.Geolocation attribute), 33
`__tempzip_is_executable()` (lawwenda.apps.file_manager_app.temp_zip_files.TempZipFiles._TempZip static method), 18
`__tempzip_put_node()` (lawwenda.apps.file_manager_app.temp_zip_files.TempZipFiles._TempZip static method), 18
`__tinyserverthread` (class in lawwenda.uts.tiny_server), 78
`__wrappedbackend` (class in lawwenda.fs.wrappers), 76
`_ATTR_KEY_ACCURACY` (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.PiMetadata.Geolocation attribute), 33
`_ATTR_KEY_LATITUDE` (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.PiMetadata.Geolocation attribute), 33
`_ATTR_KEY_LONGITUDE` (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.PiMetadata.Geolocation attribute), 33
`_CommandLineInterface` (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), 43
`_MyTestCase` (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), 45
`_MyTestCase.MyMetadataEngine` (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), 45
`_MyTestCase.MySearchEngine` (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), 45
`_RenderTemplateValue` (class in lawwenda.apps.file_manager_app.template_renderer), 19
`_TagAssignments__remove_assignments_by_tag_name()` (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.PiMetadata._TagAssignments method), 33
`_TempZip__is_executable()` (lawwenda.apps.file_manager_app.temp_zip_files.TempZipFiles._TempZip static method), 18
`_TempZip__put_node()` (lawwenda.apps.file_manager_app.temp_zip_files.TempZipFiles._TempZip static method), 18
`_TinyServerThread` (class in lawwenda.uts.tiny_server), 78
`_WrappedBackend` (class in lawwenda.fs.wrappers), 76
`__api_thumbnail()` (lawwenda.apps.file_manager_app.endpoints.Endpoints method), 17
`__api_upload()` (lawwenda.apps.file_manager_app.endpoints.Endpoints method), 17
`__api_zip()` (lawwenda.apps.file_manager_app.endpoints.Endpoints method), 17
`__api_zip_content()` (lawwenda.apps.main_app.MainApp method),
`__authenticcate()` (lawwenda.apps.file_manager_app.FileManagerApp method), 17
`__cleanup_loop()` (lawwenda.apps.file_manager_app.temp_zip_files.TempZipFiles._TempZip static method), 18
`__copy_move()` (lawwenda.apps.http_webdav_app.endpoints.Endpoints method), 25
`__decide()` (lawwenda.fs.search.ByGeoSearch method), 61
`__decide_unindexed()` (lawwenda.fs.search.DeeplySearch method), 60
`__decide_unindexed_by_content()` (lawwenda.fs.search.DeeplySearch method), 60
`__decide_unindexed_by_metadata()` (lawwenda.fs.search.DeeplySearch method), 60
`__delete()` (lawwenda.apps.http_webdav_app.endpoints.Endpoints method), 24
`__dispatch_request()` (lawwenda.apps.file_manager_app.FileManagerApp method), 16
`__dispatch_request_internals()` (lawwenda.apps.file_manager_app.FileManagerApp method), 16
`__dispatch_request_normal()` (lawwenda.apps.file_manager_app.FileManagerApp method), 16
`__ensure_authorized()` (lawwenda.apps.file_manager_app.FileManagerApp method), 16
`__ffmpeg_thumb()` (lawwenda.builtin.previewers.common_videos.PreviewVideos._PreviewVideos method), 16
`__get()` (lawwenda.apps.http_webdav_app.endpoints.Endpoints method), 23
`__get_app_for_share()` (lawwenda.apps.main_app.MainApp method),
`__remove_assignments_by_tag_name()` (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.PiMetadata._TagAssignments method), 33
`__tempzip_is_executable()` (lawwenda.apps.file_manager_app.temp_zip_files.TempZipFiles._TempZip static method), 18
`__tempzip_put_node()` (lawwenda.apps.file_manager_app.temp_zip_files.TempZipFiles._TempZip static method), 18
`__tinyserverthread` (class in lawwenda.uts.tiny_server), 78
`__wrappedbackend` (class in lawwenda.fs.wrappers), 76
`_ATTR_KEY_ACCURACY` (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.PiMetadata.Geolocation attribute), 33
`_ATTR_KEY_LATITUDE` (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.PiMetadata.Geolocation attribute), 33
`_ATTR_KEY_LONGITUDE` (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.PiMetadata.Geolocation attribute), 33
`_CommandLineInterface` (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), 43
`_MyTestCase` (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), 45
`_MyTestCase.MyMetadataEngine` (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), 45
`_MyTestCase.MySearchEngine` (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), 45
`_RenderTemplateValue` (class in lawwenda.apps.file_manager_app.template_renderer), 19
`_TagAssignments__remove_assignments_by_tag_name()` (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.PiMetadata._TagAssignments method), 33
`_TempZip__is_executable()` (lawwenda.apps.file_manager_app.temp_zip_files.TempZipFiles._TempZip static method), 18
`_TempZip__put_node()` (lawwenda.apps.file_manager_app.temp_zip_files.TempZipFiles._TempZip static method), 18
`_TinyServerThread` (class in lawwenda.uts.tiny_server), 78
`_WrappedBackend` (class in lawwenda.fs.wrappers), 76

```

26                                __verify_valid_name()
__head() (lawwenda.apps.http_webdav_app.endpoints.Endpoints (lawwenda.config.Configuration method),
method), 24                                54
__help() (lawwenda.apps.file_manager_app.endpoints.Endpoints __multistatus_response()
method), 17                                (lawwenda.apps.http_webdav_app.endpoints.Endpoints
__is_excluded() (lawwenda.fs.wrappers.ExcludingNodesWrappedBackend), 25
method), 77                                _abc_impl (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.S
__json_response() (lawwenda.apps.file_manager_app.FileManagerApp attribute), 39
method), 16                                _abc_impl (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.S
__mkcol() (lawwenda.apps.http_webdav_app.endpoints.Endpoints attribute), 40
method), 24                                _abc_impl (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.S
__move() (lawwenda.apps.http_webdav_app.endpoints.Endpoints attribute), 41
method), 25                                _abc_impl (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.S
__options() (lawwenda.apps.http_webdav_app.endpoints.Endpoints attribute), 42
method), 24                                _abc_impl (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.S
__parse_xml_propertyupdate()                                attribute), 39
(lawwenda.apps.http_webdav_app.endpoints.Endpoints _abc_impl (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.S
method), 26                                attribute), 40
__path_to_system_path()                                _abc_impl (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.S
(lawwenda.builtin.fs_backends.local.Backend attribute), 41
method), 47                                _abc_impl (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.S
__propfind() (lawwenda.apps.http_webdav_app.endpoints.Endpoints attribute), 41
method), 24                                _abc_impl (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.S
__propfind__args() (lawwenda.apps.http_webdav_app.endpoints.Endpoints attribute), 42
method), 25                                _abc_impl (lawwenda.builtin.fs_backends.local.Backend
__propfind__walk() (lawwenda.apps.http_webdav_app.endpoints.Endpoints attribute), 50
method), 25                                _abc_impl (lawwenda.builtin.previewers.common_images.Previewer
__proppatch() (lawwenda.apps.http_webdav_app.endpoints.Endpoints attribute), 51
method), 24                                _abc_impl (lawwenda.builtin.previewers.common_videos.Previewer
__put() (lawwenda.apps.http_webdav_app.endpoints.Endpoints attribute), 52
method), 24                                _abc_impl (lawwenda.fs.backend.Backend attribute), 68
__query_indexed() (lawwenda.fs.search.DeeplySearch _abc_impl (lawwenda.fs.previewer.Previewer attribute),
method), 59                                58
__query_unindexed()                                _abc_impl (lawwenda.fs.search.ByGeoSearch attribute),
(lawwenda.fs.search.DeeplySearch method), 59                                61
__quickstart() (lawwenda.lawwenda_cli.Commands _abc_impl (lawwenda.fs.search.ByNameSearch at-
method), 80                                tribute), 60
__quickstart__format_command()                                _abc_impl (lawwenda.fs.search.ByTagSearch attribute),
(lawwenda.lawwenda_cli.Commands method),                                61
80                                _abc_impl (lawwenda.fs.search.DeeplySearch at-
__render_to_str() (lawwenda.apps.file_manager_app.template_renderer.TemplateRenderer
method), 19                                _abc_impl (lawwenda.fs.search.Search attribute), 59
__rules_from_endpoints()                                _abc_impl (lawwenda.fs.wrappers.ExcludingNodesWrappedBackend
(lawwenda.apps.file_manager_app.FileManagerApp attribute), 77
method), 16                                _abc_impl (lawwenda.fs.wrappers.HidingNodesWrappedBackend
__rules_from_endpoints()                                attribute), 76
(lawwenda.apps.http_webdav_app.HttpWebdavApp _abc_impl (lawwenda.fs.wrappers.ReadOnlyWrappedBackend
method), 19                                attribute), 77
__set_node_prop() (lawwenda.apps.http_webdav_app.endpoints.Endpoints _abc_impl (lawwenda.fs.wrappers._WrappedBackend
method), 26                                attribute), 76
__static() (lawwenda.apps.file_manager_app.endpoints.Endpoints _abc_impl (in module lawwenda.fs.factory), 69
method), 17                                _cache_enabled() (lawwenda.asset.PiMetadataInterpreter.pimetadaintert
__try_get_index() (lawwenda.fs.search.DeeplySearch static method), 30
method), 59                                _cache_lock (lawwenda.asset.PiMetadataInterpreter.pimetadaintert

```

[illegible]

A

accuracy_meters (lawwenda.asset.PiMetadataInterpreter.pimetadainter)

[property](#)), 33
[active_until](#) ([lawwenda.config.share.Share](#) property), 56
[active_until_timestamp](#) ([lawwenda.config.share.Share](#) property), 56
[add_assignment\(\)](#) ([lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.FileMetadata._TagAssignments](#) method), 32
[add_share\(\)](#) ([lawwenda.config.Configuration](#) method), 53
[add_share\(\)](#) ([lawwenda.lawwenda_cli.Commands](#) method), 80
[add_tag\(\)](#) ([lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.FileMetadata._TagAssignments](#) method), 28
[add_tag_by_tag_string\(\)](#) ([lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.FileMetadata._TagAssignments](#) method), 28
[all_assignments\(\)](#) ([lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.FileMetadata._TagAssignments](#) method), 32
[all_shares\(\)](#) ([lawwenda.config.Configuration](#) method), 52
[all_tags](#) ([lawwenda.fs.node.Node](#) property), 72
[allowed_methods](#) ([lawwenda.apps.http_webdav_app.endpoint.Endpoint](#) property), 23
[And\(\)](#) ([lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.FileMetadata._TagAssignments](#) static method), 38
[app](#) ([lawwenda.apps.main_app.MainApp._ShareAppInst](#) property), 26
[append_comment\(\)](#) ([lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.FileMetadata](#) method), 28
[attributes](#) ([lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.FileMetadata](#) property), 33

B

[Backend](#) (class in [lawwenda.builtin.fs_backends.local](#)), 47
[Backend](#) (class in [lawwenda.fs.backend](#)), 62
[backend](#) ([lawwenda.fs.node.Node](#) property), 71
[Backend.ReadHandle](#) (class in [lawwenda.fs.backend](#)), 62
[Backend.WriteHandle](#) (class in [lawwenda.fs.backend](#)), 62
[basics_as_dict](#) ([lawwenda.fs.node.Node](#) property), 72
[ByGeoSearch](#) (class in [lawwenda.fs.search](#)), 61
[ByNameSearch](#) (class in [lawwenda.fs.search](#)), 60
[ByTagSearch](#) (class in [lawwenda.fs.search](#)), 60
[bytes](#) ([lawwenda.apps.file_manager_app.temp_zip_files.TempZipFiles](#) property), 18

C

[check_is_assignment_from_an_ancestor\(\)](#) ([lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.FileMetadata._TagAssignments](#) static method), 33
[child_by_name\(\)](#) ([lawwenda.fs.node.Node](#) method), 73
[child_by_path\(\)](#) ([lawwenda.fs.node.Node](#) method), 73
[child_names\(\)](#) ([lawwenda.builtin.fs_backends.local.Backend](#) method), 47
[child_names\(\)](#) ([lawwenda.fs.backend.Backend](#) method), 63
[child_names\(\)](#) ([lawwenda.fs.wrappers.ExcludingNodesWrappedBackend](#) method), 63
[child_nodes](#) ([lawwenda.fs.node.Node](#) property), 73
[CircularTraversalError](#), 56
[Commands](#) (class in [lawwenda.lawwenda_cli](#)), 79
[comment](#) ([lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.FileMetadata](#) property), 28
[comment\(\)](#) ([lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.FileMetadata](#) method), 28
[Comment\(\)](#) ([lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.FileMetadata](#) static method), 38
[comment\(\)](#) ([lawwenda.builtin.fs_backends.local.Backend](#) method), 48
[comment\(\)](#) ([lawwenda.fs.backend.Backend](#) method), 63
[comment__append\(\)](#) ([lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.FileMetadata](#) method), 43
[comment__get\(\)](#) ([lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.FileMetadata](#) method), 43
[comment__set\(\)](#) ([lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.FileMetadata](#) method), 43
[CommentDavProp](#) (class in [lawwenda.apps.http_webdav_app.dav_prop](#)), 22
[config_dir](#) ([lawwenda.config.Configuration](#) property), 52
[Configuration](#) (class in [lawwenda.config](#)), 52
[Configuration](#) ([lawwenda.config.Configuration](#) property), 55
[console\(\)](#) ([lawwenda.lawwenda_cli.Commands](#) method), 79
[copy_to\(\)](#) ([lawwenda.builtin.fs_backends.local.Backend](#) method), 49
[copy_to\(\)](#) ([lawwenda.fs.backend.Backend](#) method), 66
[copy_to\(\)](#) ([lawwenda.fs.node.Node](#) method), 74
[create_search\(\)](#) (in module [lawwenda.fs.search](#)), 61
[create_temp_zip\(\)](#) ([lawwenda.apps.file_manager_app.temp_zip_files.TempZipFiles](#) method), 17
[CrttimeDavProp](#) (class in [lawwenda.apps.http_webdav_app.dav_prop](#)), 21

D

[DavProp](#) (class in [lawwenda.apps.http_webdav_app.dav_prop](#)), 20
[DeeplySearch](#) (class in [lawwenda.fs.search](#)), 59
[define_param\(\)](#) ([lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.FileMetadata._TagAssignments](#) static method), 36
[delete\(\)](#) ([lawwenda.builtin.fs_backends.local.Backend](#) method), 48
[delete\(\)](#) ([lawwenda.fs.backend.Backend](#) method), 66

delete() (lawwenda.fs.node.Node method), 74
 describe_share() (lawwenda.lawwenda_cli.Commands method), 79
 dirpath (lawwenda.fs.node.Node property), 71
 DisplayNameDavProp (class in lawwenda.apps.http_webdav_app.dav_prop), 21
 distance_meters() (lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.SearchEngine.ResultItem method), 34
 distance_meters_interval() (lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.SearchEngine.ResultItem method), 34
E
 endpoint() (in module lawwenda.apps), 15
 Endpoints (class in lawwenda.apps.file_manager_app.endpoints), 17
 Endpoints (class in lawwenda.apps.http_webdav_app.endpoints), 23
 exclude_by_patterns (lawwenda.config.share.Share property), 55
 exclude_by_tags (lawwenda.config.share.Share property), 56
 exclude_hidden (lawwenda.config.share.Share property), 56
 ExcludingNodesWrappedBackend (class in lawwenda.fs.wrappers), 76
 exists (lawwenda.fs.node.Node property), 71
 exists() (lawwenda.builtin.fs_backends.local.Backend method), 48
 exists() (lawwenda.fs.backend.Backend method), 64
F
 file (lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.FileMetadata property), 28
 file (lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.FileMetadata.TagAssignments property), 31
 file (lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.FileMetadata.TagAssignment property), 31
 file (lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.SearchEngine.ResultItem property), 36
 FileManagerApp (class in lawwenda.apps.file_manager_app), 15
 FileMetadata (class in lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter), 27
 FileMetadata.TagAssignments (class in lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter), 31
 FileMetadata.Geolocation (class in lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter), 33
 FileMetadata.TagAssignment (class in lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter), 31
 FileMetadata.TagDuplicateError, 35
 FileMetadata.XattrEngine (class in lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter), 34
 filesystem() (in module lawwenda.fs.factory), 68
 find_assignments_by_tag_name() (lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.SearchEngine.ResultItem method), 32
 find_data_file() (in module lawwenda.asset.data), 46
 FileMetadata.Geolocation
 from_geo_string() (lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.SearchEngine.ResultItem static method), 34
 from_query_string() (lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.SearchEngine.ResultItem static method), 37
 from_tag_string() (lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.SearchEngine.ResultItem static method), 31
 from_tags_string() (lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.SearchEngine.ResultItem static method), 33
 full_as_dict (lawwenda.fs.node.Node property), 72
 FullInfoJsonProp (class in lawwenda.apps.http_webdav_app.dav_prop), 22
G
 generate_wsgi() (lawwenda.config.Configuration method), 54
 generate_wsgi() (lawwenda.lawwenda_cli.Commands method), 80
 Geo() (lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.SearchEngine.ResultItem static method), 38
 geo__get() (lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.SearchEngine.ResultItem method), 43
 geo__get_raw() (lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.SearchEngine.ResultItem method), 43
 geo__modify() (lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.SearchEngine.ResultItem method), 43
 geo__set() (lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.SearchEngine.ResultItem method), 43
 GeoDavProp (class in lawwenda.apps.http_webdav_app.dav_prop), 23
 geolocation (lawwenda.asset.PiMetadataInterpreter.pimetadatainterpreter.SearchEngine.ResultItem property), 29
 geolocation (lawwenda.fs.node.Node property), 72
 geolocation() (lawwenda.builtin.fs_backends.local.Backend method), 48
 geolocation() (lawwenda.fs.backend.Backend method), 66
 geolocation_string (lawwenda.fs.node.Node property), 72
 get_for_node() (lawwenda.apps.http_webdav_app.dav_prop.CommentData method), 22

[get_for_node\(\)](#) (lawwenda.apps.http_webdav_app.dav_prop.InfoProp (class in lawwenda.apps.http_webdav_app.dav_prop), method), 21
[get_for_node\(\)](#) (lawwenda.apps.http_webdav_app.dav_prop.DavProp (class in lawwenda.apps.http_webdav_app.dav_prop), method), 20
[get_for_node\(\)](#) (lawwenda.apps.http_webdav_app.dav_prop.DisplayNameProp (class in lawwenda.apps.http_webdav_app.dav_prop), method), 21
[get_for_node\(\)](#) (lawwenda.apps.http_webdav_app.dav_prop.IsFavoriteProp (class in lawwenda.apps.http_webdav_app.dav_prop), method), 22
[get_for_node\(\)](#) (lawwenda.apps.http_webdav_app.dav_prop.GeoDataProp (class in lawwenda.apps.http_webdav_app.dav_prop), method), 23
[get_for_node\(\)](#) (lawwenda.apps.http_webdav_app.dav_prop.IsInfoProp (class in lawwenda.apps.http_webdav_app.dav_prop), method), 22
[get_for_node\(\)](#) (lawwenda.apps.http_webdav_app.dav_prop.IsKnownByUrlProp (class in lawwenda.apps.http_webdav_app.dav_prop), method), 22
[get_for_node\(\)](#) (lawwenda.apps.http_webdav_app.dav_prop.IsMileDataProp (class in lawwenda.apps.http_webdav_app.dav_prop), method), 21
[get_for_node\(\)](#) (lawwenda.apps.http_webdav_app.dav_prop.RatingDataProp (class in lawwenda.apps.http_webdav_app.dav_prop), method), 23
[get_for_node\(\)](#) (lawwenda.apps.http_webdav_app.dav_prop.IsKiddedTypeDataProp (class in lawwenda.apps.http_webdav_app.dav_prop), method), 21
[get_for_node\(\)](#) (lawwenda.apps.http_webdav_app.dav_prop.SizeDataProp (class in lawwenda.apps.http_webdav_app.dav_prop), method), 21
[get_for_node\(\)](#) (lawwenda.apps.http_webdav_app.dav_prop.TagsDataProp (class in lawwenda.apps.http_webdav_app.dav_prop), method), 22
[get_prop_by_name\(\)](#) (in module lawwenda.apps.http_webdav_app.dav_prop), method), 20
[get_xattr\(\)](#) (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadataEngine (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), method), 45
[get_xattr\(\)](#) (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadataEngine (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), method), 34

H

[has_preview](#) (lawwenda.fs.node.Node property), 72
[has_preview\(\)](#) (lawwenda.fs.backend.Backend (class in lawwenda.fs.backend), method), 65
[has_thumbnail](#) (lawwenda.fs.node.Node property), 72
[has_thumbnail\(\)](#) (lawwenda.fs.backend.Backend (class in lawwenda.fs.backend), method), 64
[hide_by_patterns](#) (lawwenda.config.share.Share property), 55
[hide_by_tags](#) (lawwenda.config.share.Share property), 55
[HidingNodesWrappedBackend](#) (class in lawwenda.fs.wrappers), 76
[HttpWebdavApp](#) (class in lawwenda.apps.http_webdav_app), 19

I

[icon_name](#) (lawwenda.fs.node.Node property), 72
[include_by_patterns](#) (lawwenda.config.share.Share property), 55
[include_by_tags](#) (lawwenda.config.share.Share property), 55
[is_active](#) (lawwenda.apps.main_app.MainApp._ShareAppInst (class in lawwenda.apps.main_app), method), 56
[is_active](#) (lawwenda.config.share.Share property), 56
[is_ancestor](#) (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadataEngine (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), method), 32
[is_dir](#) (lawwenda.fs.node.Node property), 71
[is_dir\(\)](#) (lawwenda.builtin.fs_backends.local.Backend (class in lawwenda.builtin.fs_backends), method), 47
[is_expired](#) (lawwenda.config.share.Share property), 56
[is_file](#) (lawwenda.fs.node.Node property), 71
[is_file\(\)](#) (lawwenda.builtin.fs_backends.local.Backend (class in lawwenda.builtin.fs_backends), method), 47
[is_file\(\)](#) (lawwenda.fs.backend.Backend (class in lawwenda.fs.backend), method), 63
[is_hidden](#) (lawwenda.fs.node.Node property), 71
[is_hidden\(\)](#) (lawwenda.builtin.fs_backends.local.Backend (class in lawwenda.builtin.fs_backends), method), 47
[is_hidden\(\)](#) (lawwenda.fs.backend.Backend (class in lawwenda.fs.backend), method), 63
[is_hidden\(\)](#) (lawwenda.fs.wrappers.HidingNodesWrappedBackend (class in lawwenda.fs.wrappers), method), 76
[is_internal](#) (lawwenda.apps.http_webdav_app.dav_prop.DavProp (class in lawwenda.apps.http_webdav_app.dav_prop), property), 20
[is_link](#) (lawwenda.builtin.fs_backends.local.Backend (class in lawwenda.builtin.fs_backends), method), 47
[is_link\(\)](#) (lawwenda.fs.backend.Backend (class in lawwenda.fs.backend), method), 64
[is_matching](#) (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadataEngine (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), method), 40
[is_matching\(\)](#) (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadataEngine (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), method), 41
[is_matching\(\)](#) (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadataEngine (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), method), 42
[is_matching\(\)](#) (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadataEngine (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), method), 39
[is_matching\(\)](#) (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadataEngine (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), method), 40
[is_matching\(\)](#) (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadataEngine (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), method), 41
[is_matching\(\)](#) (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadataEngine (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), method), 40
[is_matching\(\)](#) (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadataEngine (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), method), 42
[is_matching\(\)](#) (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadataEngine (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), method), 37
[is_previewable](#) (lawwenda.builtin.previewers.common_images.Previewer (class in lawwenda.builtin.previewers.common_images), method), 51
[is_previewable](#) (lawwenda.builtin.previewers.common_videos.Previewer (class in lawwenda.builtin.previewers.common_videos), method), 51
[is_previewable\(\)](#) (lawwenda.fs.preview.Previewer (class in lawwenda.fs.preview), method), 51

method), 57
 is_thumbnailable() (lawwenda.builtin.previewers.common_images.PiMetadataInterpreter.pimetadainterpreter.
 method), 50
 is_thumbnailable() (lawwenda.builtin.previewers.common_videos.PiMetadataInterpreter.pimetadainterpreter.
 method), 51
 is_thumbnailable() (lawwenda.fs.previewer.Previewer method), 57
 is_writable (lawwenda.apps.http_webdav_app.dav_prop.DavProp property), 20
 is_writable (lawwenda.fs.node.Node property), 71
 IsAbleRequest (class in lawwenda.fs.previewer), 58

J

JSON (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.SearchEngine.SearchCriteria.SearchCriteriaFormat
 attribute), 36

K

known_tags() (lawwenda.builtin.fs_backends.local.Backend method), 50
 known_tags() (lawwenda.fs.backend.Backend method), 68
 known_tags() (lawwenda.fs.node.Node method), 75
 KnownTagsProp (class in lawwenda.apps.http_webdav_app.dav_prop), 22

L

lat (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadata.GeoLocation property), 33
 lawwenda
 module, 15
 lawwenda.apps
 module, 15
 lawwenda.apps.file_manager_app
 module, 15
 lawwenda.apps.file_manager_app.endpoints
 module, 17
 lawwenda.apps.file_manager_app.temp_zip_files
 module, 17
 lawwenda.apps.file_manager_app.template_renderer
 module, 18
 lawwenda.apps.http_webdav_app
 module, 19
 lawwenda.apps.http_webdav_app.dav_prop
 module, 20
 lawwenda.apps.http_webdav_app.endpoints
 module, 23
 lawwenda.apps.main_app
 module, 26
 lawwenda.asset
 module, 27
 lawwenda.asset.data
 module, 46
 lawwenda.asset.PiMetadataInterpreter

module, 27
 lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter
 module, 27
 lawwenda.asset.project_info
 module, 47
 lawwenda.builtin
 module, 47
 lawwenda.builtin.fs_backends
 module, 47
 lawwenda.builtin.fs_backends.local
 module, 47
 lawwenda.builtin.previewers
 module, 50
 lawwenda.builtin.previewers.common_images
 module, 50
 lawwenda.builtin.previewers.common_videos
 module, 51
 lawwenda.config
 module, 52
 lawwenda.config.share
 module, 54
 lawwenda.fs
 module, 56
 lawwenda.fs.backend
 module, 62
 lawwenda.fs.data
 module, 68
 lawwenda.fs.factory
 module, 68
 lawwenda.fs.node
 module, 70
 lawwenda.fs.previewer
 module, 57
 lawwenda.fs.search
 module, 58
 lawwenda.fs.wrappers
 module, 76
 lawwenda.lawwenda_cli
 module, 79
 lawwenda.utils
 module, 78
 lawwenda.utils.tiny_server
 module, 78
 list_shares() (lawwenda.lawwenda_cli.Commands
 method), 79
 list_xattrs() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.
 method), 45
 list_xattrs() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.
 method), 34
 lon (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMeta
 property), 33

M

main() (in module lawwenda.lawwenda_cli), 79

[main\(\)](#) ([lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter](#) [50](#) [CommandLineInterface](#) [method](#)), [44](#)
[MainApp](#) (class in [lawwenda.apps.main_app](#)), [26](#)
[MainApp._ShareAppInst](#) (class in [lawwenda.apps.main_app](#)), [26](#)
[merge_assignments\(\)](#) ([lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter](#) [50](#) [CommandLineInterface](#) [method](#)), [32](#)
[metadata_for_file\(\)](#) (in module [lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter](#) [27](#) [lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter](#) [50](#) [CommandLineInterface](#) [method](#)), [45](#)
[mimetype](#) ([lawwenda.fs.node.Node](#) property), [71](#)
[mimetype](#) ([lawwenda.fs.previewer.IsAbleRequest](#) attribute), [58](#)
[mimetype\(\)](#) ([lawwenda.fs.backend.Backend](#) method), [64](#)
[misc__measure_geolocations_distance\(\)](#) ([lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter](#) [50](#) [CommandLineInterface](#) [method](#)), [44](#)
[misc__search\(\)](#) ([lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter](#) [50](#) [CommandLineInterface](#) [method](#)), [44](#)
[mkdir\(\)](#) ([lawwenda.builtin.fs_backends.local.Backend](#) method), [49](#)
[mkdir\(\)](#) ([lawwenda.fs.backend.Backend](#) method), [66](#)
[mkdir\(\)](#) ([lawwenda.fs.node.Node](#) method), [74](#)
[module](#)
 [lawwenda](#), [15](#)
 [lawwenda.apps](#), [15](#)
 [lawwenda.apps.file_manager_app](#), [15](#)
 [lawwenda.apps.file_manager_app.endpoints](#), [17](#)
 [lawwenda.apps.file_manager_app.temp_zip_file](#), [17](#)
 [lawwenda.apps.file_manager_app.template_render](#), [18](#)
 [lawwenda.apps.http_webdav_app](#), [19](#)
 [lawwenda.apps.http_webdav_app.dav_prop](#), [20](#)
 [lawwenda.apps.http_webdav_app.endpoints](#), [23](#)
 [lawwenda.apps.main_app](#), [26](#)
 [lawwenda.asset](#), [27](#)
 [lawwenda.asset.data](#), [46](#)
 [lawwenda.asset.PiMetadataInterpreter](#), [27](#)
 [lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter](#), [27](#)
 [lawwenda.asset.project_info](#), [47](#)
 [lawwenda.builtin](#), [47](#)
 [lawwenda.builtin.fs_backends](#), [47](#)
 [lawwenda.builtin.fs_backends.local](#), [47](#)
 [lawwenda.builtin.previewers](#), [50](#)
 [lawwenda.builtin.previewers.common_images](#), [51](#)
 [lawwenda.config](#), [52](#)
 [lawwenda.config.share](#), [54](#)
 [lawwenda.fs](#), [56](#)
 [lawwenda.fs.backend](#), [64](#)
 [lawwenda.fs.data](#), [68](#)
 [lawwenda.fs.factory](#), [68](#)
 [lawwenda.fs.node](#), [70](#)
 [lawwenda.fs.previewer](#), [57](#)
 [lawwenda.fs.search](#), [58](#)
 [lawwenda.fs.wrappers](#), [76](#)
 [lawwenda.lawwenda_cli](#), [79](#)
 [lawwenda.utils](#), [78](#)
 [lawwenda.utils.tiny_server](#), [78](#)
 [move_to\(\)](#) ([lawwenda.builtin.fs_backends.local.Backend](#) method), [49](#)
 [move_to\(\)](#) ([lawwenda.fs.backend.Backend](#) method), [66](#)
 [move_to\(\)](#) ([lawwenda.fs.node.Node](#) method), [74](#)
 [mtime](#) ([lawwenda.fs.node.Node](#) property), [72](#)
 [mtime\(\)](#) ([lawwenda.fs.backend.Backend](#) method), [64](#)
 [mtime_ts](#) ([lawwenda.fs.node.Node](#) property), [72](#)
 [MtimeDavProp](#) (class in [lawwenda.apps.http_webdav_app.dav_prop](#)), [21](#)

N
 [name](#) ([lawwenda.apps.http_webdav_app.dav_prop.DavProp](#) property), [20](#)
 [name](#) ([lawwenda.config.share.Share](#) property), [55](#)
 [name](#) ([lawwenda.fs.node.Node](#) property), [71](#)
 [name](#) ([lawwenda.fs.previewer.IsAbleRequest](#) attribute), [58](#)
 [Node](#) (class in [lawwenda.fs.node](#)), [70](#)
 [Not\(\)](#) ([lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.Search](#) static method), [38](#)
 [not_existing_file](#) ([lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter](#) attribute), [45](#)

O
 [Or\(\)](#) ([lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.Search](#) static method), [38](#)

P
 [parent_node](#) ([lawwenda.fs.node.Node](#) property), [73](#)
 [parser\(\)](#) (in module [lawwenda.lawwenda_cli](#)), [79](#)
 [password_salt](#) ([lawwenda.config.share.Share](#) property), [56](#)
 [password_encrypt](#) ([lawwenda.config.share.Share](#) property), [56](#)
 [path](#) ([lawwenda.config.share.Share](#) property), [55](#)
 [path](#) ([lawwenda.fs.node.Node](#) property), [70](#)

peek_share_cache_tag() (lawwenda.config.Configuration method), 52
 pop_path_info() (in module lawwenda.apps), 15
 preview_html (lawwenda.fs.node.Node property), 72
 preview_html() (lawwenda.builtin.previewers.common_images.Previewer method), 51
 preview_html() (lawwenda.builtin.previewers.common_videos.Previewer method), 52
 preview_html() (lawwenda.fs.backend.Backend method), 65
 preview_html() (lawwenda.fs.previewer.Previewer method), 58
 Previewer (class in lawwenda.builtin.previewers.common_images), 50
 Previewer (class in lawwenda.builtin.previewers.common_videos), 51
 Previewer (class in lawwenda.fs.previewer), 57
Q
 query() (lawwenda.fs.search.ByGeoSearch method), 61
 query() (lawwenda.fs.search.ByNameSearch method), 60
 query() (lawwenda.fs.search.ByTagSearch method), 61
 query() (lawwenda.fs.search.DeeplySearch method), 59
 query() (lawwenda.fs.search.Search method), 59
R
 rating (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadata property), 29
 rating (lawwenda.fs.node.Node property), 72
 Rating() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.SearchEngine.SearchCriterion static method), 38
 rating() (lawwenda.builtin.fs_backends.local.Backend method), 48
 rating() (lawwenda.fs.backend.Backend method), 65
 rating_get() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.CommandLineInterface method), 43
 rating_get_default() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.CommandLineInterface method), 44
 rating_get_maximum() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.CommandLineInterface method), 44
 rating_get_minimum() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.CommandLineInterface method), 44
 rating_reset() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.CommandLineInterface method), 44
 rating_set() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.CommandLineInterface method), 44
 RATING_MAX_VALID_VALUE (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadata attribute), 29
 RATING_MIN_VALID_VALUE (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadata attribute), 28
 rating_or_default (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadata property), 29
 RATING_UNDEFINED_VALUE (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadata attribute), 29
 RatingDavProp (class in lawwenda.apps.http_webdav_app.dav_prop), 23
 read_file() (lawwenda.builtin.fs_backends.local.Backend method), 50
 read_file() (lawwenda.fs.backend.Backend method), 67
 read_file() (lawwenda.fs.node.Node method), 74
 read_handle() (lawwenda.fs.backend.Backend method), 62
 read_handle() (lawwenda.fs.wrappers.ExcludingNodesWrappedBackend method), 77
 readme_pdf() (in module lawwenda.asset.data), 46
 readonly (lawwenda.config.share.Share property), 55
 ReadOnlyWrappedBackend (class in lawwenda.fs.wrappers), 77
 register_prop() (in module lawwenda.apps.http_webdav_app.dav_prop), 20
 remove_assignments_by_tag_name() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadata method), 32
 remove_share() (lawwenda.config.Configuration method), 53
 remove_share() (lawwenda.lawwenda_cli.Commands method), 80
 remove_tag() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.CommandLineInterface method), 28
 render() (lawwenda.apps.file_manager_app.template_renderer.TemplateRenderer method), 18
 render_to_str() (lawwenda.apps.file_manager_app.template_renderer.TemplateRenderer method), 18
 reset_rating() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.CommandLineInterface method), 29
 ResourceTypeDavProp (class in lawwenda.apps.http_webdav_app.dav_prop), 21
 root_dir (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.CommandLineInterface property), 35
 root_node (lawwenda.apps.file_manager_app.FileManagerApp property), 16
 root_node (lawwenda.apps.http_webdav_app.HttpWebdavApp property), 19
 run() (lawwenda.utils.tiny_server._TinyServerThread method), 78
 run_tiny_server() (lawwenda.lawwenda_cli.Commands method), 78

method), 79

running_tiny_servers() (in module lawwenda.utils.tiny_server), 78

S

sanitize_path() (in module lawwenda.fs), 56

Search (class in lawwenda.fs.search), 59

search() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.SearchEngine method), 35

search_engine() (in module lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), 27

search_engine() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.MyTestCase static method), 45

SearchEngine (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), 35

SearchEngine._SearchCriterionAnd (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), 40

SearchEngine._SearchCriterionComment (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), 41

SearchEngine._SearchCriterionGeo (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), 41

SearchEngine._SearchCriterionNot (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), 39

SearchEngine._SearchCriterionOr (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), 39

SearchEngine._SearchCriterionRating (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), 41

SearchEngine._SearchCriterionTag (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), 40

SearchEngine._SearchCriterionXattrValue (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), 42

SearchEngine.MetadataFormatError, 43

SearchEngine.ResultItem (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), 36

SearchEngine.SearchCriterion (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), 36

SearchEngine.SearchCriterion.SerializationFormat (class in lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter), 36

set_comment() (lawwenda.builtin.fs_backends.local.Backend method), 49

set_comment() (lawwenda.fs.backend.Backend method), 67

set_comment() (lawwenda.fs.node.Node method), 75

set_for_node() (lawwenda.apps.http_webdav_app.dav_prop.CommentDavProp method), 22

set_for_node() (lawwenda.apps.http_webdav_app.dav_prop.DavProp method), 20

set_for_node() (lawwenda.apps.http_webdav_app.dav_prop.GeoDavProp method), 23

set_for_node() (lawwenda.apps.http_webdav_app.dav_prop.RatingDavProp method), 23

set_for_node() (lawwenda.apps.http_webdav_app.dav_prop.TagsDavProp method), 23

set_geolocation() (lawwenda.builtin.fs_backends.local.Backend method), 49

set_geolocation() (lawwenda.fs.backend.Backend method), 67

set_geolocation() (lawwenda.fs.node.Node method), 75

set_rating() (lawwenda.builtin.fs_backends.local.Backend method), 49

set_rating() (lawwenda.fs.backend.Backend method), 67

set_rating() (lawwenda.fs.node.Node method), 75

set_tags() (lawwenda.builtin.fs_backends.local.Backend method), 49

set_tags() (lawwenda.fs.backend.Backend method), 67

set_tags() (lawwenda.fs.node.Node method), 75

set_tags_by_tag_strings() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.File method), 28

set_xattr() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.File method), 45

set_xattr() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.File method), 34

Share (class in lawwenda.config.share), 54

share_by_name() (lawwenda.config.Configuration method), 52

shutdown() (lawwenda.utils.tiny_server.TinyServer method), 79

size (lawwenda.fs.node.Node property), 71

size() (lawwenda.builtin.fs_backends.local.Backend method), 48

size() (lawwenda.fs.backend.Backend method), 64

SizeDavProp (class in lawwenda.apps.http_webdav_app.dav_prop), 21

start_tiny_server() (in module lawwenda.utils.tiny_server), 78

start_tiny_server() (lawwenda.config.Configuration method), 53

SUBCRITERION (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter attribute), 36

summary (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.PiMetadataInterpreter property), 36

system_path() (lawwenda.builtin.fs_backends.local.Backend method), 46

system_path() (lawwenda.fs.backend.Backend method), 63

system_path() (lawwenda.fs.node.Node method), 73

T

Tag() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.PiMetadataInterpreter static method), 38

tag_name (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.PiMetadata.TagAssignment property), 31

tags (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.PiMetadata property), 28

tags (lawwenda.fs.node.Node property), 72

tags() (lawwenda.builtin.fs_backends.local.Backend method), 48

tags() (lawwenda.fs.backend.Backend method), 65

tags__add() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.PiMetadata method), 44

tags__drop() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.PiMetadata method), 44

tags__get() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.PiMetadata method), 44

tags_string (lawwenda.fs.node.Node property), 72

tags_without_ancestors (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.PiMetadata property), 28

TagsDavProp (class in lawwenda.apps.http_webdav_app.dav_prop), 22

temp_zip_content() (lawwenda.apps.file_manager_app.temp_zip_files.TempZipFiles method), 17

TemplateRenderer (class in lawwenda.apps.file_manager_app.template_renderer), 18

TempZipFiles (class in lawwenda.apps.file_manager_app.temp_zip_files), 17

TempZipFiles._TempZip (class in lawwenda.apps.file_manager_app.temp_zip_files), 18

test_cli_basic() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.MyTestCase method), 46

test_create_tag_assignment_s_and_tag_assignments_s_from_scratch_and_assign_them() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.MyTestCase method), 45

test_high_level_comment() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.MyTestCase method), 46

test_high_level_geo() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.MyTestCase method), 46

test_high_level_tags() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.MyTestCase method), 46

test_high_level_tags() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.MyTestCase method), 45

test_interpreter_cache_xattr() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.MyTestCase method), 46

test_parse_and_generate_tag_string_to_from_tag_assignments (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.MyTestCase method), 45

test_search_by_xattr() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.MyTestCase method), 46

test_search_convert_query_string_search_criterion() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.MyTestCase method), 46

test_search_query_string_empty() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.MyTestCase method), 46

test_search_tags_control_character() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.MyTestCase method), 46

test_search_via_query_string_by_tags_comment_rating_geo() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.MyTestCase method), 46

test_search_via_search_criteria_by_combined() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.MyTestCase method), 46

test_tags_non_alphanumeric_characters() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.MyTestCase method), 45

thumbheight (lawwenda.fs.previewer.Previewer attribute), 57

thumbnail() (lawwenda.builtin.previewers.common_images.Previewer method), 50

thumbnail() (lawwenda.builtin.previewers.common_videos.Previewer method), 51

thumbnail() (lawwenda.fs.backend.Backend method), 65

thumbnail() (lawwenda.fs.node.Node method), 74

thumbnail() (lawwenda.fs.previewer.Previewer method), 57

thumbsize (lawwenda.fs.previewer.Previewer attribute), 57

thumbwidth (lawwenda.fs.previewer.Previewer attribute), 57

TinyServer (class in lawwenda.utils.tiny_server), 79

title (lawwenda.config.share.Share property), 55

to_geo_string() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.MyTestCase method), 34

to_query_string() (lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.MyTestCase method), 37

`to_tag_string()` (*lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadata.TagAssignment method*), 31

`to_tags_string()` (*lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadata._TagAssignments method*), 32

`traverse_dir()` (*lawwenda.fs.node.Node method*), 73

U

`unesaped` (*lawwenda.apps.file_manager_app.template_renderer._RenderTemplateValue property*), 19

`unset_xattr()` (*lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter._MyTestCase.MyMetadataEngine method*), 45

`unset_xattr()` (*lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadata.XattrEngine method*), 35

`url` (*lawwenda.utils.tiny_server.TinyServer property*), 79

`URL_ENCODED` (*lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.SearchEngine.SearchCriterion.SerializationFormat attribute*), 36

`url_internals_name` (*lawwenda.apps.file_manager_app.FileManagerApp property*), 16

W

`wait_stopped()` (*lawwenda.utils.tiny_server.TinyServer method*), 79

`write_file()` (*lawwenda.builtin.fs_backends.local.Backend method*), 50

`write_file()` (*lawwenda.fs.backend.Backend method*), 67

`write_file()` (*lawwenda.fs.node.Node method*), 75

`write_handle()` (*lawwenda.fs.backend.Backend method*), 62

`write_handle()` (*lawwenda.fs.wrappers.ExcludingNodesWrappedBackend method*), 77

`write_handle()` (*lawwenda.fs.wrappers.ReadOnlyWrappedBackend method*), 77

X

`XATTR_KEY_COMMENT` (*lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadata attribute*), 30

`XATTR_KEY_GEO` (*lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadata attribute*), 30

`XATTR_KEY_RATING` (*lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadata attribute*), 30

`XATTR_KEY_TAGS` (*lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.FileMetadata attribute*), 31

`XattrValue()` (*lawwenda.asset.PiMetadataInterpreter.pimetadainterpreter.SearchEngine.SearchCriterion static method*), 39