lawwenda - Plugin API Documentation

Release 0.3.157

Josef Hahn

Apr 14, 2021

CONTENTS

Lawwenda has a plugin interface which allows to add functionality to the core from outside.

TODO not yet available

CHAPTER

ONE

API REFERENCE

1.1 lawwenda package

1.1.1 Subpackages

lawwenda.fs package

Submodules

lawwenda.fs.data module

Data structures for Lawwenda filesystems.

lawwenda.fs.decorators module

Filesystem decorators.

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

class lawwenda.fs.decorators.**AbstractFilesystemDecorator**(*inner*)

Bases: *lawwenda.fs.Filesystem*

TODO.

Parameters **inner** (lawwenda.fs.Filesystem) –

_eval_predicate(*pred*, *node*)

child_nodes(*handle*)

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

Parameters **handle** – The read handle to a node.

class lawwenda.fs.decorators.**ExcludeNodesFilesystemDecorator**(*inner*, *pred*)

Bases: *lawwenda.fs.decorators.AbstractFilesystemDecorator*

Decorator for filesystems that forcefully excludes some nodes.

Parameters

- **inner** (lawwenda.fs.Filesystem) –
- **pred** (*Callable[[Filesystem.Node], bool]*) –

child_nodes(*handle*)
Return all child nodes for a node (in a handle).

Parameters **handle** – The read handle to a node.

get_readhandle(*node*)
Return a read handle for a node.

Such handles are needed for executing read actions on that node. See also `Filesystem.ReadHandle`.

Parameters **node** – The node to read from later.

get_writehandle(*node*)
Return a write handle for a node.

Such handles are needed for executing write actions on that node. See also `Filesystem.WriteHandle`.

Parameters **node** – The node to write from later.

class lawwenda.fs.decorators.**HideNodesFilesystemDecorator**(*inner*, *pred*)
Bases: *lawwenda.fs.decorators.AbstractFilesystemDecorator*

Decorator for filesystems that marks some nodes.

Parameters

- **inner** (lawwenda.fs.Filesystem) –
- **pred** (*Callable[[Filesystem.Node], bool]*) –

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

Parameters **handle** – The read handle to a node.

class lawwenda.fs.decorators.**ReadOnlyFilesystemDecorator**(*inner*)
Bases: *lawwenda.fs.decorators.AbstractFilesystemDecorator*

Decorator for filesystems that blocks all write accesses.

Parameters **inner** (lawwenda.fs.Filesystem) –

get_writehandle(*node*)
Return a write handle for a node.

Such handles are needed for executing write actions on that node. See also `Filesystem.WriteHandle`.

Parameters **node** – The node to write from later.

lawwenda.fs.factory module

Creating Lawwenda filesystems.

class lawwenda.fs.factory.**_PredicateFactory**
Bases: object

Used by *create_filesystem()* for translating its arguments into filter predicates.

static excludehidden(*node*)
Return *True* for a hidden node.

static false(*node*)
Return *False* for each node.

static p_descending(*pred*)
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 **pred** – The input predicate.

static p_not(*pred*)
Return a predicate that inverts the input predicate.

Parameters **pred** – The input predicate.

static p_or(**preds*)
Return a predicate that returns *True* iff at least one of the input predicates return *True*.

Parameters **preds** – The input predicates.

static p_regexp(*restr*)
Return a predicate that returns *True* for input nodes whose `Filesystem.Node.path` matches the given regular expression.

Parameters **restr** – The regular expression string.

static p_tag(*tag*)
Return a predicate that returns *True* for input nodes whose `Filesystem.Node.tags` contain the given tag.

Parameters **tag** – The tag to look for.

lawwenda.fs.factory.**create_filesystem**(*rootpath*, ***kwargs*)
Create a `Filesystem` resembling a particular subtree of your real local filesystem, with some configuration for access control and more.

See *decorate_filesystem()* for details about the parameters.

Parameters **rootpath** (*str*) – The path from your real local filesystem to consider as the root directory.

Return type *lawwenda.fs.Filesystem*

lawwenda.fs.factory.**decorate_filesystem**(*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** (`lawwenda.fs.Filesystem`) – The filesystem to decorate.
- **readonly** (*bool*) – Whether it should block write accesses.
- **hide_by_patterns** (*Iterable[str]*) – TODO.
- **hide_by_tags** (*Iterable[str]*) – TODO.
- **include_by_patterns** (*Optional[Iterable[str]]*) – TODO.
- **include_by_tags** (*Optional[Iterable[str]]*) – TODO.
- **exclude_by_patterns** (*Iterable[str]*) – TODO.
- **exclude_by_tags** (*Iterable[str]*) – TODO.
- **exclude_hidden** (*bool*) – TODO.

Return type *lawwenda.fs.Filesystem*

lawwenda.fs.local module

Local filesystem implementation.

class lawwenda.fs.local.**LocalFilesystem**(*rootpath*)
Bases: *lawwenda.fs.Filesystem*

A `Filesystem` implementation that resembles a particular subtree of your real local filesystem.

Parameters **rootpath** (*str*) –

__path_to_fullpath(*path*)

Parameters **path** (*str*) –

Return type str

add_tag(*handle*, *tag*)
Add a tag to a node (in a handle).

Parameters

- **handle** – The write handle to a node.
- **tag** – The tag to add.

child_nodes(*handle*)
Return all child nodes for 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.

copy_to(*srchandle*, *desthandle*)
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.

Parameters

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

delete(*handle*)
Delete a node (in a handle).

Parameters **handle** – The write handle to a node.

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

This can e.g. be *False* for nodes coming from `node_by_path()`.

Parameters **handle** – The read handle to a node.

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

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

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.

known_tags()
TODO.

mkdir(*handle*)
Make a node (in a handle) an existing directory.

Parameters **handle** – The write handle to a node.

move_to(*srchandle*, *desthandle*)
Move a node (in a handle) to another node (in a handle).

This is similar to *copy_to()*; see there for more details.

Parameters

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

mtime(*handle*)
Return the 'last modified' time of 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).

Parameters **handle** – The read handle to a node.

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.

remove_tag(*handle*, *tag*)
Remove a tag from a node (in a handle).

Parameters

- **handle** – The write handle to a node.
- **tag** – The tag to remove.

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_geo(*handle*, *geo*)
Set the geographic location for a node (in a handle).

Parameters

- **handle** – The write handle to a node.
- **geo** – The new geographic location (as encoded string).

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.

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

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.

try_get_fullpath(*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.

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.

Module contents

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.

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.

class lawwenda.fs.**Filesystem**
Bases: object

Base class for a filesystem source implementation.

Subclasses implement different kinds of filesystems, e.g. *lawwenda.fs.local.LocalFilesystem*.

You should not use this interface directly for much more than getting its *rootnode*. It provides a friendlier and equally powerful interface. The interface of this class is relevant for implementing custom filesystems only.

This class is not intended to be instantiated directly. You will get instances by some other api methods.

class Node(*path*, ***, *filesystem*)
Bases: object

TODO.

Parameters

- **path** (*str*) –
- **filesystem** (Filesystem) –

property _filesystem
Return type *Filesystem*
Return type *Filesystem*
Return type *Filesystem*
Return type *Filesystem*

add_tag(*tag*)
Add a tag to this node.
Parameters **tag** (*str*) – The tag to add.
Return type None

property basics_as_dict
Basic node data as dict.

This is solely used internally for serialization.

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 Optional[*lawwenda.fs.Filesystem.Node*]

property child_nodes
The list of child nodes, i.e. nodes for all files and sub-directories inside this node.

This only makes sense on directory nodes and will be empty otherwise.
Return type t.List["Filesystem.Node"]
Return type t.List["Filesystem.Node"]

Return type t.List["Filesystem.Node"]
Return type t.List["Filesystem.Node"]

property comment
The node comment text.
Return type str
Return type str
Return type str
Return type str

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

delete()
Delete this node.
Return type None

property dirpath
The directory path of the node.

This is all but the last segment of *path*. Same as *path* of *parent_node*.
Return type str
Return type str
Return type str
Return type str

property exists
Whether a node points to something that actually exists.

This can e.g. be *False* for nodes coming from *child_by_name()*.
Return type bool
Return type bool
Return type bool
Return type bool

property full_as_dict
Complete node data as dict.

This is solely used internally for serialization.

property geo
The geographic location associated to this node, encoded in a string.
Return type str
Return type str
Return type str
Return type str

property geo_obj
The geographic location associated to this node.
Return type lawwenda.fs.data.GeoLocation
Return type lawwenda.fs.data.GeoLocation
Return type lawwenda.fs.data.GeoLocation
Return type lawwenda.fs.data.GeoLocation

property has_preview
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.
Return type bool
Return type bool
Return type bool
Return type bool

property has_thumbnail
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.
Return type bool
Return type bool
Return type bool
Return type bool

property icon_name
The recommended icon name for this node.
Return type t.Optional[str]
Return type t.Optional[str]
Return type t.Optional[str]
Return type t.Optional[str]

property is_dir
Whether the node is a directory.

This is also *True* for link nodes (see *is_link()*) that point to a directory!
Return type bool
Return type bool
Return type bool
Return type bool

property is_file
Whether the node is a regular file.

This is also *True* for link nodes (see *is_link()*) that point to a directory!
Return type bool
Return type bool
Return type bool
Return type bool

property is_hidden
Whether the node is hidden.
Return type bool
Return type bool
Return type bool
Return type bool

property is_link
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.
Return type bool

Return type bool
Return type bool
Return type bool

property is_writable
Whether the node is writable.
Return type bool
Return type bool
Return type bool
Return type bool

property mimetype
The mimetype of this node.
Return type str
Return type str
Return type str
Return type str

mkdir()
Make this node an existing directory.
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

property mtime
The 'last modified' time of this node.
Return type datetime.datetime
Return type datetime.datetime
Return type datetime.datetime
Return type datetime.datetime

property mtime_ts
Same as *mtime*, but as Unix timestamp.
Return type float
Return type float
Return type float
Return type float

property name
The file name of the node.

This is the last segment of *path*.
Return type str
Return type str
Return type str
Return type str

property parent_node
The parent node.

This is *None* for the root node.
Return type *Filesystem.Node*
Return type *Filesystem.Node*

Return type *Filesystem.Node*
Return type *Filesystem.Node*

property path
The path of this node.

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* it is part of. It is not relative to '/' of your real filesystem (unless you have actually set up a *Filesystem* that resembles your entire real filesystem).
Return type str
Return type str
Return type str
Return type str

property preview_html
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*.
Return type str
Return type str
Return type str
Return type str

property rating
The node rating.
Return type int
Return type int
Return type int
Return type int

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

remove_tag(*tag*)
Remove a tag from this node.
Parameters **tag** (*str*) – The tag to remove.
Return type None

set_comment(*comment*)
Set the comment for this node.
Parameters **comment** (*str*) – The new comment.
Return type None

set_geo(*geo*)
Set the geographic location for this node.
Parameters **geo** (*str*) – The new geographic location (as encoded string).
Return type None

set_rating(*rating*)
Set the rating for this node.
Parameters **rating** (*int*) – The new rating.
Return type None

property size
The size of this node in bytes.
Return type int
Return type int
Return type int
Return type int

property tags
The tags assigned to this node.
Return type t.List[str]
Return type t.List[str]
Return type t.List[str]
Return type t.List[str]

property tagstring
The tags assigned to this node, encoded in one string.
Return type str
Return type str
Return type str
Return type str

thumbnail()
The thumbnail for this node in PNG format.

See also *has_thumbnail*.
Return type bytes

traverse_dir(*, *raise_on_circle*, *param_path=''*)
Return paths from this node and all descendants (i.e. the sub-tree).
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[*lawwenda.fs.Filesystem.Node*, str]]

try_get_fullpath(*, *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 Optional[str]

write_file(*content*)
Write content to this node.

This will overwrite its original content.
Parameters **content** (*Union[bytes, BinaryIO]*) – The binary content to write to the node.
Return type None

class ReadHandle(*node*)
Bases: object

Read and write handles are just a stupid container that hold one *Filesystem.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 *Filesystem.get_readhandle()* and *Filesystem.get_writehandle()*.

Parameters **node** (Filesystem.Node) –

class WriteHandle(*node*)
Bases: *lawwenda.fs.Filesystem.ReadHandle*

See *Filesystem.ReadHandle*.

Parameters **node** (Filesystem.Node) –

add_tag(*handle*, *tag*)
Add a tag to a node (in a handle).

Parameters

- **handle** (lawwenda.fs.Filesystem.WriteHandle) – The write handle to a node.
- **tag** (*str*) – The tag to add.

Return type None

child_nodes(*handle*)
Return all child nodes for a node (in a handle).

Parameters **handle** (lawwenda.fs.Filesystem.ReadHandle) – The read handle to a node.

Return type List[*lawwenda.fs.Filesystem.Node*]

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

Parameters **handle** (lawwenda.fs.Filesystem.ReadHandle) – The read handle to a node.

Return type str

copy_to(*srchandle*, *desthandle*)
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.

Parameters

- **srchandle** (lawwenda.fs.Filesystem.ReadHandle) – The read handle to the source node.
- **desthandle** (lawwenda.fs.Filesystem.WriteHandle) – The write handle to the destination node.

Return type None

delete(*handle*)
Delete a node (in a handle).

Parameters **handle** (lawwenda.fs.Filesystem.WriteHandle) – The write handle to a node.

Return type None

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

This can e.g. be *False* for nodes coming from *node_by_path()*.

Parameters **handle** (lawwenda.fs.Filesystem.ReadHandle) – The read handle to a node.

Return type bool

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

Parameters **handle** (lawwenda.fs.Filesystem.ReadHandle) – The read handle to a node.

Return type Union[lawwenda._aux.PiMetadataInterpreter.pimetadatainterpreter.GeoLocation, Dict[str, Any]]

get_readhandle(*node*)
Return a read handle for a node.

Such handles are needed for executing read actions on that node. See also *Filesystem.ReadHandle*.

Parameters **node** (lawwenda.fs.Filesystem.Node) – The node to read from later.

Return type *lawwenda.fs.Filesystem.ReadHandle*

get_writehandle(*node*)
Return a write handle for a node.

Such handles are needed for executing write actions on that node. See also *Filesystem.WriteHandle*.

Parameters **node** (lawwenda.fs.Filesystem.Node) – The node to write from later.

Return type *lawwenda.fs.Filesystem.WriteHandle*

has_preview(*handle*)
Return whether there could be a 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** (lawwenda.fs.Filesystem.ReadHandle) – The read handle to a node.

Return type bool

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** (lawwenda.fs.Filesystem.ReadHandle) – The read handle to a node.

Return type bool

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** (lawwenda.fs.Filesystem.ReadHandle) – The read handle to a node.

Return type bool

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** (lawwenda.fs.Filesystem.ReadHandle) – The read handle to a node.

Return type bool

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

Parameters **handle** (lawwenda.fs.Filesystem.ReadHandle) – The read handle to a node.

Return type bool

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** (lawwenda.fs.Filesystem.ReadHandle) – The read handle to a node.

Return type bool

known_tags()
TODO.

Return type List[str]

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

Parameters **handle** (lawwenda.fs.Filesystem.ReadHandle) – The read handle to a node.

Return type str

mkdir(*handle*)
Make a node (in a handle) an existing directory.

Parameters **handle** (lawwenda.fs.Filesystem.WriteHandle) – The write handle to a node.

Return type None

move_to(*srchandle*, *desthandle*)
Move a node (in a handle) to another node (in a handle).

This is similar to *copy_to()*; see there for more details.

Parameters

- **srchandle** (lawwenda.fs.Filesystem.WriteHandle) – The write handle to the source node.
- **desthandle** (lawwenda.fs.Filesystem.WriteHandle) – The write handle to the destination node.

Return type None

mtime (*handle*)

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

Parameters **handle** (lawwenda.fs.Filesystem.ReadHandle) – The read handle to a node.

Return type datetime.datetime

node_by_path (*path*)

Return a node by a given path.

It will not fail, even if there is no such file or access would be denied.

Parameters **path** (*str*) – The path of the node to return, relative to the filesystem's root node.

Return type *lawwenda.fs.Filesystem.Node*

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** (lawwenda.fs.Filesystem.ReadHandle) – The read handle to a node.

Return type str

rating (*handle*)

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

Parameters **handle** (lawwenda.fs.Filesystem.ReadHandle) – The read handle to a node.

Return type int

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** (lawwenda.fs.Filesystem.ReadHandle) – The read handle to a node.

Return type BinaryIO

remove_tag (*handle*, *tag*)

Remove a tag from a node (in a handle).

Parameters

- **handle** (lawwenda.fs.Filesystem.WriteHandle) – The write handle to a node.

- **tag** (*str*) – The tag to remove.

Return type None

property rootnode
The root node of this filesystem.

Return type *Filesystem.Node*

Return type *Filesystem.Node*

Return type *Filesystem.Node*

Return type *Filesystem.Node*

static sanitize_abspath(*path*)
Sanitize slashes in a path and returns a path in the form */foo/bar*.

Precisely:

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

Parameters **path** (*str*) – The input path.

Return type str

set_comment(*handle*, *comment*)
Set the comment for a node (in a handle).

Parameters

- **handle** (lawwenda.fs.Filesystem.WriteHandle) – The write handle to a node.
- **comment** (*str*) – The new comment.

Return type None

set_geo(*handle*, *geo*)
Set the geographic location for a node (in a handle).

Parameters

- **handle** (lawwenda.fs.Filesystem.WriteHandle) – The write handle to a node.
- **geo** (*str*) – The new geographic location (as encoded string).

Return type None

set_rating(*handle*, *rating*)
Set the rating for a node (in a handle).

Parameters

- **handle** (lawwenda.fs.Filesystem.WriteHandle) – The write handle to a node.
- **rating** (*int*) – The new rating.

Return type None

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

Parameters **handle** (lawwenda.fs.Filesystem.ReadHandle) – The read handle to a node.

Return type int

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

Parameters **handle** (lawwenda.fs.Filesystem.ReadHandle) – The read handle to a node.

Return type List[str]

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

See also *has_thumbnail()*.

Parameters **handle** (lawwenda.fs.Filesystem.ReadHandle) – The read handle to a node.

Return type bytes

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

Return type Optional[str]

write_file(*handle*, *content*)
Write content to a node (in a handle).

This will overwrite its original content.

Parameters

- **handle** (lawwenda.fs.Filesystem.WriteHandle) – The write handle to a node.
- **content** (*Union[bytes, BinaryIO]*) – The binary content to write to the node.

Return type None

lawwenda.previewer package

Submodules

lawwenda.previewer.commonimages module

Preview functionality for common image formats.

class lawwenda.previewer.commonimages.**CommonImagesPreviewer**
Bases: *lawwenda.previewer.Previewer*

Preview functionality for common image formats.

_abc_impl = <_abc_data object>

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.

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.

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.

thumbnail(*node*)
Return a thumbnail for a file in PNG format.

See also *is_thumbnailable()*.

Parameters **node** – The file to generate a thumbnail for.

lawwenda.previewer.commonvideos module

Preview functionality for common video formats.

class lawwenda.previewer.commonvideos.**CommonVideosPreviewer**
Bases: *lawwenda.previewer.Previewer*

Preview functionality for common video formats.

__ffmpeg_thumb(*stdin=- 3*)

Parameters

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

Return type bytes

_abc_impl = <_abc_data object>

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.

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.

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.

thumbnail(*node*)
Return a thumbnail for a file in PNG format.

See also *is_thumbnailable()*.

Parameters **node** – The file to generate a thumbnail for.

Module contents

Preview functionality.

Used by different parts of the Lawwenda user interface.

class lawwenda.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*) –

Return type None

mimetype: str

name: str

class lawwenda.previewer.**Previewer**
Bases: abc.ABC

Base class for previewers.

Subclasses implement thumbnail and preview functionality for particular file formats.

_abc_impl = <_abc_data object>

_image_scale_to_thumbnail_size(*imgdata*)

Parameters **imgdata** (*BinaryIO*) –

Return type _io.BytesIO

abstract 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** (lawwenda.previewer.IsAbleRequest) – An object that describes the file.

Return type bool

abstract 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** (lawwenda.previewer.IsAbleRequest) – An object that describes the file.

Return type bool

abstract 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.Filesystem.Node) – The file to preview.

Return type str

thumbheight = 300

abstract thumbnail(*node*)
Return a thumbnail for a file in PNG format.

See also *is_thumbnailable()*.

Parameters **node** (lawwenda.fs.Filesystem.Node) – The file to generate a thumbnail for.

Return type bytes

thumbsize = (300, 300)

thumbwidth = 300

lawwenda.search package

Module contents

File search functionality.

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

class lawwenda.search.**ByGeoSearch**(*position*, *, *radius*)
Bases: *lawwenda.search.Search*

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

See also *lawwenda.fs.Filesystem.Node.geo*.

Parameters

- **position** (*str*) –
- **radius** (*Union[str, float]*) –

__decide(*node*)

Parameters **node** (*FilesystemNode*) –

Return type bool

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.

class lawwenda.search.**ByNameSearch**(*term*)
Bases: *lawwenda.search.Search*

Searches for search terms in file names.

See also *lawwenda.fs.Filesystem.Node.name*.

Parameters **term** (*str*) –

__decide(*node*)

Parameters **node** (*FilesystemNode*) –

Return type bool

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.

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

Bases: *lawwenda.search.Search*

Searches for files that have particular tags.

See also *lawwenda.fs.Filesystem.Node.tags*.

Parameters **term** (*str*) –

__decide(*node*)

Parameters **node** (*FilesystemNode*) –

Return type bool

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.

class lawwenda.search.**DeeplySearch**(*term*)

Bases: *lawwenda.search.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*) –

__decide_unindexed(*node*)

Parameters **node** (*FilesystemNode*) –

Return type bool

__decide_unindexed_by_content(*node*)

Parameters **node** (*FilesystemNode*) –

Return type bool

__decide_unindexed_by_metadata(*node*)

Parameters **node** (*FilesystemNode*) –

Return type bool

__query_indexed(*idxdata*)

Parameters

- **node** (*FilesystemNode*) –
- **idxdata** (*object*) –

Return type FilesystemNodes

__query_unindexed(*node*)

Parameters **node** (*FilesystemNode*) –

Return type FilesystemNodes

__try_get_index(*node*)

Parameters **node** (*FilesystemNode*) –

Return type object

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.

class lawwenda.search.**Search**

Bases: object

Base class for different search behaviors.

query(*node*)

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

Parameters **node** (*FilesystemNode*) – The root node. Query result nodes are usually inside that subtree.

Return type FilesystemNodes

lawwenda.search.**_query_by_tree_traversal**(*node*, *decidefct*)

Parameters

- **node** (*FilesystemNode*) –
- **decidefct** (*Callable[[FilesystemNode], bool]*) –

Return type FilesystemNodes

lawwenda.search.**_termstring_to_termlist**(*term*)

Parameters **term** (*str*) –

Return type List[str]

lawwenda.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 *lawwenda.search.Search*

lawwenda.sharehttpapp package

Submodules

lawwenda.sharehttpapp.davprop module

WebDAV properties.

class lawwenda.sharehttpapp.davprop.**CommentDavProp**
Bases: *lawwenda.sharehttpapp.davprop.DavProp*

Custom WebDAV property *{DAV:xattr}xdg.comment* that mirrors *lawwenda.fs.Filesystem.Node.comment*.

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

Parameters **fsnode** – The node to inquire.

class lawwenda.sharehttpapp.davprop.**DavProp**(*davname*, *, *is_writable*)
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

- **davname** (*str*) –
- **is_writable** (*bool*) –

property davname
The name of this property.

Return type str

Return type str

Return type str

Return type str

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

Parameters **fsnode** (lawwenda.fs.Filesystem.Node) – The node to inquire.

Return type Optional[str]

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

Return type bool

Return type bool

Return type bool

Return type bool

class lawwenda.sharehttpapp.davprop.**GeoDavProp**
Bases: *lawwenda.sharehttpapp.davprop.DavProp*

Custom WebDAV property *{DAV:xattr}pino.geo* that mirrors *lawwenda.fs.Filesystem.Node.geo*.

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

Parameters **fsnode** – The node to inquire.

class lawwenda.sharehttpapp.davprop.**MtimeDavProp**
Bases: *lawwenda.sharehttpapp.davprop.DavProp*

The WebDAV *getlastmodified* property.

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

Parameters **fsnode** – The node to inquire.

class lawwenda.sharehttpapp.davprop.**RatingDavProp**
Bases: *lawwenda.sharehttpapp.davprop.DavProp*

Custom WebDAV property *{DAV:xattr}baloo.rating* that mirrors *lawwenda.fs.Filesystem.Node.rating*.

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

Parameters **fsnode** – The node to inquire.

class lawwenda.sharehttpapp.davprop.**ResourceTypeDavProp**
Bases: *lawwenda.sharehttpapp.davprop.DavProp*

The WebDAV *resourcetype* property.

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

Parameters **fsnode** – The node to inquire.

class lawwenda.sharehttpapp.davprop.**SizeDavProp**
Bases: *lawwenda.sharehttpapp.davprop.DavProp*

The WebDAV *getcontentlength* property.

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

Parameters **fsnode** – The node to inquire.

class lawwenda.sharehttpapp.davprop.**TagsDavProp**
Bases: *lawwenda.sharehttpapp.davprop.DavProp*

Custom WebDAV property *{DAV:xattr}xdg.tags* that mirrors *lawwenda.fs.Filesystem.Node.tagstring*.

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

Parameters **fsnode** – The node to inquire.

lawwenda.sharehttpapp.davprop.**get_prop_by_davname**(*davname*)
Return a DavProp by name or *None* if no such ones exist.

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

Return type Optional[*lawwenda.sharehttpapp.davprop.DavProp*]

lawwenda.sharehttpapp.davprop.**register_prop**(*prop*)
Register a DavProp implementation.

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

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

Return type Type[*lawwenda.sharehttpapp.davprop.DavProp*]

Module contents

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

See *ShareHttpApp*.

class lawwenda.sharehttpapp.**ShareHttpApp**(*filesystem*)
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 user authentication.

Parameters **filesystem** (lawwenda.fs.Filesystem) –

property __allowed_methods

Return type t.List[str]

Return type t.List[str]

Return type t.List[str]

Return type t.List[str]

_method_copy(*request*, *fsnode*, *, *xfermethod='copy_to'*)

Parameters

- **request** (lawwenda.comm.Request) –
- **fsnode** (lawwenda.fs.Filesystem.Node) –
- **xfermethod** (*str*) –

_method_delete(*request*, *fsnode*)

Parameters

- **request** (lawwenda.comm.Request) –
- **fsnode** (lawwenda.fs.Filesystem.Node) –

_method_get(*request*, *fsnode*, *is_head_request=False*)

Parameters

- **request** (lawwenda.comm.Request) –
- **fsnode** (lawwenda.fs.Filesystem.Node) –
- **is_head_request** (*bool*) –

_method_head(*request*, *fsnode*)

Parameters

- **request** (lawwenda.comm.Request) –
- **fsnode** (lawwenda.fs.Filesystem.Node) –

_method_mkcol(*request*, *fsnode*)

Parameters

- **request** (lawwenda.comm.Request) –
- **fsnode** (lawwenda.fs.Filesystem.Node) –

_method_move(*request*, *fsnode*)

Parameters

- **request** (lawwenda.comm.Request) –
- **fsnode** (lawwenda.fs.Filesystem.Node) –

_method_options(*request*, *fsnode*)

Parameters

- **request** (lawwenda.comm.Request) –
- **fsnode** (lawwenda.fs.Filesystem.Node) –

_method_propfind(*request*, *fsnode*)

Parameters

- **request** (lawwenda.comm.Request) –
- **fsnode** (lawwenda.fs.Filesystem.Node) –

_method_proppatch(*request*, *fsnode*)

Parameters

- **request** (lawwenda.comm.Request) –
- **fsnode** (lawwenda.fs.Filesystem.Node) –

_method_put(*request*, *fsnode*)

Parameters

- **request** (lawwenda.comm.Request) –
- **fsnode** (lawwenda.fs.Filesystem.Node) –

property filesystem

The filesystem used by this ShareHttpApp.

Return type *lawwenda.fs.Filesystem*

Return type *lawwenda.fs.Filesystem*

Return type *lawwenda.fs.Filesystem*

Return type *lawwenda.fs.Filesystem*

1.1.2 Submodules

1.1.3 lawwenda.comm module

Internal communication helpers.

class lawwenda.comm.**JsonedResponse**(*data*, ***kwargs*)
Bases: werkzeug.wrappers.response.Response

A werkzeug response that serializes response data to json.

Parameters **data** (*Optional[Any]*) – Response data. Can be anything that *json* is able to serialize.

class lawwenda.comm.**Request**(*environ*, *populate_request=True*, *shallow=False*)
Bases: werkzeug.wrappers.request.Request, werkzeug.wrappers.json.JSONMixin

A werkzeug request that is able to interpret json data in the response body.

1.1.4 lawwenda.datafiles module

Paths to some files and directories of Lawwenda.

lawwenda.datafiles.**find_data_file**(*fname*, *searchdirs=None*)
Return the absolute path of a Lawwenda data file.

Can return *None* if no such file exists.

Parameters

- **fname** (*str*) – The name of the file to find.
- **searchdirs** (*Optional[Iterable[str]]*) – List of directories to look into. If not specified (as it usually should be), it looks at some usual places.

Return type Optional[str]

1.1.5 lawwenda.devserver module

Tiny local server for trying, development, testing.

class lawwenda.devserver.**_DevServerInfo**(*svr*, *svrthread*)
Bases: object

shutdown()
Stop this server.

Return type None

property url
The url of this running server.

Return type str

Return type str

Return type str

Return type str

wait_stopped()
Wait until this server stopped.

Return type None

class lawwenda.devserver.**_DevServerThread**(*svr*)
Bases: threading.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 the argument tuple 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.

lawwenda.devserver.**_run_dev_server_for_app**(*app*)
Start a tiny local server for a given wsgi application.

Parameters **app** (*Callable*) – The wsgi application to run.

Return type *lawwenda.devserver._DevServerInfo*

lawwenda.devserver.**get_running_dev_servers**()
Return the servers started by *run_dev_server()* that are currently running.

Return type Iterable[*lawwenda.devserver._DevServerInfo*]

lawwenda.devserver.**run_dev_server**(*cfg*)
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 contains the full url, and more.

Parameters **cfg** (lawwenda.Configuration) – The configuration to run with.

Return type *lawwenda.devserver._DevServerInfo*

1.1.6 lawwenda.fmapp module

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

See *FmApp*.

class lawwenda.fmapp.**FmApp**(*share*)
Bases: object

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

Parameters **share** (lawwenda.Share) – The share to provide by this application.

URL_INTERNALS_NAME = '.~__lawwenda__int~'

__auth(*username*, *password*)

Parameters

- **username** (*str*) –
- **password** (*str*) –

Return type bool

__dispatch_request(*request*)

__dispatch_request_internals(*request*)

__dispatch_request_normal(*request*)

__ensure_authed(*request*)

__on_api_copymove(*request*, *action*)

__render_template(*template*, ***, *html_head_inner=''*, *path=''*, *headonly=False*, *url_internals_name='.~__lawwenda__int~'*, ***kwargs*)

Parameters

- **template** (*str*) –
- **html_head_inner** (*str*) –
- **path** (*str*) –
- **headonly** (*bool*) –
- **url_internals_name** (*str*) –

Return type werkzeug.wrappers.response.Response

__render_template_text(*template*, ***, *csrftoken*, *html_head_inner*, ***kwargs*)

Parameters

- **template** (*str*) –
- **csrftoken** (*str*) –
- **html_head_inner** (*str*) –

Return type str

__render_template_text_raw(*template*, ***kwargs*)

Parameters **template** (*str*) –

Return type str

_on_api_copy(*request*)

_on_api_delete(*request*)

_on_api_details(*request*)

_on_api_dir(*request*)

_on_api_known_tags(*_*)

_on_api_mkdir(*request*)

_on_api_move(*request*)

_on_api_rename(*request*)

_on_api_set_comment(*request*)

_on_api_set_geo(*request*)

_on_api_set_rating(*request*)

_on_api_tag_entries(*request*)

_on_api_thumbnail(*request*)

_on_api_untag_entries(*request*)

_on_api_upload(*request*)

_on_api_zip(*request*)

_on_api_zip_download(*request*, *zipid*)

_on_help(*_*)

_on_static(*_*, *filepath*)

property filesystem
The filesystem used by this FmApp.

Return type *lawwenda.fs.Filesystem*

Return type *lawwenda.fs.Filesystem*

Return type *lawwenda.fs.Filesystem*

Return type *lawwenda.fs.Filesystem*

class lawwenda.fmapp.**_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.

property unescaped
The unescaped string.

class lawwenda.fmapp.**_TempZips**
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.

class _TempZip(*nodes*)
Bases: object

A temporary zip file.

Parameters **nodes** (*List[lawwenda.fs.Filesystem.Node]*) –

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

_TempZip__putnode(*node*, *zipf*)
Parameters
- **node** (lawwenda.fs.Filesystem.Node) –
- **zipf** (*zipfile.ZipFile*) –

Return type None

property bytes
The binary representation of this temporary zip archive.

classmethod _TempZips__cleanup_loop()

_cleanup_thread = None

_lock = <unlocked _thread.lock object>

_zips = {}

classmethod create_tempzip(*owner*, *nodes*)
Create a temporary zip archive from some nodes in memory and return an identifier.

See also *get_tempzip()*.

Parameters

- **owner** (*object*) – Request owner. Can be an arbitrary object (but must be equal for all calls that belong together).
- **nodes** (*List[lawwenda.fs.Filesystem.Node]*) – The nodes to include in the zip archive.

Return type str

classmethod get_tempzip(*owner*, *zipid*)
Return the binary content of a zip archive that was created before by *create_tempzip()*.

Parameters

- **owner** (*object*) – Request owner. Can be an arbitrary object (but must be equal for all calls that belong together).
- **zipid** (*str*) – The identifier returned by *create_tempzip*.

Return type Optional[bytes]

1.1.7 lawwenda.lawwenda_cli module

The Lawwenda command line interface.

This tool is used by the end user for creating and managing shares and more. It can be called with some command line arguments for a particular action, or without parameters (which will open an interactive Python prompt for further actions).

Do not use this module in code. See *lawwenda.Configuration* instead.

lawwenda.lawwenda_cli.**_fmt_cmd**(*txt*)

Parameters **txt** (*str*) –

Return type str

lawwenda.lawwenda_cli.**_quickstart**(*cfgpath*)

Parameters **cfgpath** (*str*) –

Return type str

lawwenda.lawwenda_cli.**add_share**(*, *cfg*, *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*)
Add a share.

Only used internally by the cli. See *lawwenda.Configuration.add_share()*.

Parameters

- **cfg** (lawwenda.Configuration) –
- **name** (*str*) –
- **path** (*str*) –
- **title** (*str*) –
- **active_until** (*str*) –
- **hide_by_pattern** (*List[str]*) –
- **hide_by_tag** (*List[str]*) –
- **include_by_pattern** (*List[str]*) –
- **include_by_tag** (*List[str]*) –
- **exclude_by_pattern** (*List[str]*) –
- **exclude_by_tag** (*List[str]*) –
- **exclude_hidden** (*bool*) –
- **readonly** (*bool*) –

Return type None

lawwenda.lawwenda_cli.**console**(*, *cfg*)

Open an interactive Python console.

The user will get a small help text and is then able to configure Lawwenda by typing Python code.

Parameters **cfg** (lawwenda.Configuration) – The initial value for the *cfg* variable inside the console.

Return type None

lawwenda.lawwenda_cli.**describe_share**(*, *cfg*, *name*)

Return a textual share description (i.e. how it is configured) for a share.

Only used internally by the cli. See *lawwenda.Configuration*.

Parameters

- **cfg** (lawwenda.Configuration) –
- **name** (*str*) –

Return type str

lawwenda.lawwenda_cli.**generate_wsgi**(*, *cfg*)

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

Only used internally by the cli. See *lawwenda.Configuration.generate_wsgi()*.

Parameters **cfg** (lawwenda.Configuration) –

Return type str

lawwenda.lawwenda_cli.**list_shares**(*, *cfg*)

Return a list of share names as string, one line for each share.

Only used internally by the cli. See *lawwenda.Configuration*.

Parameters **cfg** (lawwenda.Configuration) –

Return type str

lawwenda.lawwenda_cli.**main**()
Parse command line arguments and call the right function with its arguments.

lawwenda.lawwenda_cli.**remove_share**(*, *cfg*, *name*)
Remove a share.

Only used internally by the cli. See *lawwenda.Configuration.remove_share()*.

Parameters

- **cfg** (lawwenda.Configuration) –
- **name** (*str*) –

Return type None

lawwenda.lawwenda_cli.**run_dev_server**(*, *cfg*)
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, will print its url, and blocks.

Only used internally by the cli. See *lawwenda.Configuration*.

Parameters **cfg** (lawwenda.Configuration) –

Return type None

1.1.8 lawwenda.mainapp module

TODO.

class lawwenda.mainapp.**MainApp**(*, *cfgpath=None*)
Bases: object

TODO.

Parameters **cfgpath** (*Optional[str]*) – TODO.

class **_ShareAppInst**(*share*)
Bases: object

Container for one *lawwenda.fmapp.FmApp* and its associated *lawwenda.Share*.

Parameters **share** (lawwenda.Share) –

property **app**
The application.
Return type *lawwenda.fmapp.FmApp*
Return type *lawwenda.fmapp.FmApp*
Return type *lawwenda.fmapp.FmApp*
Return type *lawwenda.fmapp.FmApp*

property **is_active**
Whether this share is still active (not yet removed).
Return type bool
Return type bool
Return type bool
Return type bool

__get_app_for_share(*sharename*)

Parameters **sharename** (*str*) –

1.1.9 Module contents

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

For most cases, instantiate *Configuration* and use some if its methods.

class lawwenda.**Configuration**(*cfgpath=None*)

Bases: object

TODO.

Parameters **cfgpath** (*Optional[str]*) – 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.

__verify_valid_name()

Parameters **name** (*str*) –

Return type 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 *Share.path*.
- **name** (*str*) – The unique name of the new share. See *Share.name*.
- **password** (*Optional[str]*) – The password to protect the share with.
- **title** (*Optional[str]*) – The share title. See *Share.title*.
- **readonly** (*bool*) – If to share in a read-only way. See *Share.readonly*.
- **hide_by_patterns** (*Iterable[str]*) – See *Share.hide_by_patterns*.
- **hide_by_tags** (*Iterable[str]*) – See *Share.hide_by_tags*.
- **include_by_patterns** (*Optional[Iterable[str]]*) – See *Share.include_by_patterns*.
- **include_by_tags** (*Optional[Iterable[str]]*) – See *Share.include_by_tags*.
- **exclude_by_patterns** (*Iterable[str]*) – See *Share.exclude_by_patterns*.
- **exclude_by_tags** (*Iterable[str]*) – See *Share.exclude_by_tags*.
- **exclude_hidden** (*bool*) – See *Share.exclude_hidden*.
- **active_until** (*Optional[datetime.datetime]*) – The optional expiration time of the share. See *Share.active_until*.

Return type *lawwenda.Share*

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

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

Parameters **name** (*str*) –

Return type Optional[*lawwenda.Share*]

get_shares()
Return all shares that are currently part of this configuration.

Return type List[*lawwenda.Share*]

property path
The path of the configuration directory.

Return type str

Return type str

Return type str

Return type str

peek_share_cache_tag(*name*)
TODO.

Parameters **name** (*str*) – The share name.

Return type Optional[object]

remove_share(*name*)
Remove a share.

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

Return type None

run_dev_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 contains the full url, and more.

Return type *lawwenda.devserver._DevServerInfo*

class lawwenda.**Share**(*path*, ***, *configuration*, *name*, *title*, *cachetag*, *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.

This class is not intended to be instantiated directly. You will get instances by some other api methods.

Parameters

- **path** (*str*) –
- **configuration** (Configuration) –
- **name** (*str*) –
- **title** (*str*) –

- **cachetag** (*object*) –
- **readonly** (*bool*) –
- **hide_by_patterns** (*Iterable[str]*) –
- **hide_by_tags** (*Iterable[str]*) –
- **include_by_patterns** (*Optional[Iterable[str]]*) –
- **include_by_tags** (*Optional[Iterable[str]]*) –
- **exclude_by_patterns** (*Iterable[str]*) –
- **exclude_by_tags** (*Iterable[str]*) –
- **exclude_hidden** (*bool*) –
- **password_scrypt** (*Optional[str]*) –
- **password_salt** (*Optional[str]*) –
- **active_until_timestamp** (*Optional[float]*) –

_to_dict()

property active_until

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

Return type t.Optional[datetime.datetime]

Return type t.Optional[datetime.datetime]

Return type t.Optional[datetime.datetime]

Return type t.Optional[datetime.datetime]

property active_until_timestamp

Same as *active_until*, but as Unix timestamp.

Return type t.Optional[float]

Return type t.Optional[float]

Return type t.Optional[float]

Return type t.Optional[float]

property configuration

The configuration that contains this share.

Return type *Configuration*

Return type *Configuration*

Return type *Configuration*

Return type *Configuration*

property exclude_by_patterns

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).

Return type t.Iterable[str]

Return type t.Iterable[str]

Return type t.Iterable[str]

Return type t.Iterable[str]

property exclude_by_tags
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).

Return type t.Iterable[str]

Return type t.Iterable[str]

Return type t.Iterable[str]

Return type t.Iterable[str]

property exclude_hidden
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).

Return type bool

Return type bool

Return type bool

Return type bool

property hide_by_patterns
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.

Return type t.Iterable[str]

Return type t.Iterable[str]

Return type t.Iterable[str]

Return type t.Iterable[str]

property hide_by_tags
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.

Return type t.Iterable[str]

Return type t.Iterable[str]

Return type t.Iterable[str]

Return type t.Iterable[str]

property include_by_patterns
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.

Return type t.Optional[t.Iterable[str]]

Return type t.Optional[t.Iterable[str]]

Return type t.Optional[t.Iterable[str]]

Return type t.Optional[t.Iterable[str]]

property include_by_tags

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.

Return type t.Optional[t.Iterable[str]]

Return type t.Optional[t.Iterable[str]]

Return type t.Optional[t.Iterable[str]]

Return type t.Optional[t.Iterable[str]]

property is_active

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

Return type bool

Return type bool

Return type bool

Return type bool

property is_expired

If this share is expired.

Return type bool

Return type bool

Return type bool

Return type bool

property name

The share name.

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

Return type str

Return type str

Return type str

Return type str

property password_salt

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

See also *password_scrypt*.

Return type t.Optional[str]

Return type t.Optional[str]

Return type t.Optional[str]

Return type t.Optional[str]

property password_scrypt
The scrypt hash of the password for this share.

Empty or *None* for disabled password protection. See also *password_salt*.

Return type t.Optional[str]

Return type t.Optional[str]

Return type t.Optional[str]

Return type t.Optional[str]

property path
The path of the share's root directory.

Return type str

Return type str

Return type str

Return type str

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

Return type bool

Return type bool

Return type bool

Return type bool

property title
The share title.

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

Return type str

Return type str

Return type str

Return type str

PYTHON MODULE INDEX

l

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

P

Q

R

S

T

U

W