python-alist-api

Kai Peng (pkemb)

2023 年 08 月 06 日

Contents:

Alist python api。仅支持 Alist v2。

CHAPTER 1

API Reference

1.1 alist

1.1.1 Module contents

class alist.**AlistClient**(*base_url*, *password=None*, *authorization=None*, *ssl_verify=True*, *cert=None*)

基类：object

Python wrapper for the Alist API.

public: ***alist.public.AlistPublic***

admin: ***alist.admin.AlistAdmin***

login(*password=None*, *authorization=None*)

使用 password 或 authorization 登录。

参数

- **password** -- 密码
- **authorization** -- 授权码

返回

登录成功返回 Ture，登录失败触发异常。

is_login()

是否登录

返回

如果已登录，返回 True；否则返回 Flase。

static decode_response(*response*)

解析服务器的响应。

返回

将 JSON 解析为字典，如果不是 JSON 则返回原始字符串。

Raises

如果响应包含 HTTP 错误则触发 requests.HTTPError。

get_request_dict(*method*, *endpoint*, ***kwargs*)

获取 requests 请求参数。

参数

- **method** -- 请求方法，GET、POST 或 DELETE。
- **endpoint** -- URL 端点。
- **kwargs** -- 其他可选参数。

返回

requests 请求参数。

get_api_url(*endpoint*)

返回指定端点的 api url，不包含主机和端口。

参数

endpoint -- 服务端点。

返回

api url

get_endpoint_url(*endpoint*)

返回指定端点的完整 URL，包含主机和端口。

参数

endpoint -- 服务端点。

返回

完整的 URL。

get(*endpoint*, ***kwargs*)

发送 HTTP GET 请求到端点。

参数

endpoint -- 发送请求的端点。

返回

服务器返回的数据。

post(*endpoint*, ***kwargs*)

发送 HTTP POST 请求到端点。

参数

endpoint -- 发送请求的端点。

返回

服务器返回的数据。

delete(*endpoint*, ***kwargs*)

发送 HTTP DELETE 请求到端点。

参数

endpoint -- 发送请求的端点。

返回

服务器返回的数据。

1.1.2 Submodules

alist.setting

class alist.setting.**AlistSetting**(***kwargs*)

基类：dict

描述 Alist 设置，包含 key、value、description、type、group、access、values、version 等信息。必须设置 key。对象初始化之后，只能修改 value，其他属性不能修改。

get_value()

获取设置值

set_value(*new*)

根据类型，对新值 new 进行判断，保存。

参数

new -- 如果 type == 'bool'，则新值 new 只能是'true' 或'false'。如果 type == 'select'，则新值 new 必须是 values 指定的值。其他 type 无限制。

返回

新值 new。

class alist.setting.**AlistAaminSettings**(*alist*, *admin*, *endpoint*)

基类：object

Alist 管理员设置

每个管理员设置均有单独的 API 来获取和设置。请参考*admin setting*。

settings_ro = ['version']

只读设置

settings_rw = ['title', 'logo', 'favicon', 'icon color', 'announcement', 'text types', 'audio types', 'video types', 'hide files', 'music cover', 'site beian', 'global readme url', 'pdf viewer url', 'autoplay video', 'autoplay audio', 'customize head', 'customize body', 'home emoji', 'animation', 'artplayer whitelist', 'artplayer autoSize', 'load type', 'default page size', 'password', 'd_proxy types', 'check parent folder', 'check down link', 'WebDAV username', 'WebDAV password', 'Visitor WebDAV username', 'Visitor WebDAV password', 'ocr api', 'enable search', 'Aria2 RPC url', 'Aria2 RPC secret']

可修改的设置

get(*group=None*)

获取管理员设置。

参数

group -- 指定设置组。如果留空，返回所有设置。

返回

管理员设置。

frontend()

获取前端设置

backend()

获取后端设置

other()

获取其他设置

get_setting(*key*) → *AlistSetting*

获取指定设置。

参数

key -- 设置的键值

save(*settings: list*)

保存设置

参数

settings -- AlistSetting 列表

返回

保存成功返回 True

delete()

删除设置。未实现。

class alist.setting.**AlistPublicSettings**(*alist*, *public*, *endpoint*)

基类：object

每个公开设置均有单独的 API 来获取。请参考*public setting*。

settings = ['version', 'title', 'logo', 'favicon', 'icon color', 'announcement', 'text types', 'audio types', 'video types', 'hide files', 'music cover', 'site beian', 'global readme url', 'pdf viewer url', 'autoplay video', 'autoplay audio', 'home emoji', 'animation', 'check down link', 'artplayer whitelistartplayer autoSize', 'load type', 'default page size', 'enable search', 'no cors', 'no upload']

公开设置的键值

get(*group=None*)

获取公开的设置

返回

公开的设置

get_setting(*key*)

获取指定设置。

参数

key -- 设置的键值

alist.account

class alist.account.**AlistAccount**(**kwargs*)

基类：dict

描述 Alist 账户信息。不同账户需要的信息各不相同。更详细的信息请参考`AlistAdminDrivers`。

class alist.account.**AlistAdminAccount**(*alist*, *endpoint: str*)

基类：object

/api/admin/account API 的实现。创建、删除、修改账号。

create_Onedrive(*name*, *zone*, *internal_type*, *client_id*, *client_secret*, *redirect_uri*, *refresh_token*, *proxy=False*, *webdav_proxy=False*, *webdav_direct=False*, **kwargs*)

创建一个 Onedrive 账号。

参数

- **name** -- 账号名字。即虚拟地址。
- **zone** -- 区域。可以是 global、cn、us 或 de。

- **internal_type** -- `onedrive` 或 `sharepoint`。
- **client_id** -- client id
- **client_secret** -- client secret
- **redirect_uri** -- 重定向 URI
- **refresh_token** -- 刷新 token
- **proxy** -- 是否开启代码。默认为 False。
- **webdav_proxy** -- 默认为 False。
- **webdav_direct** -- 默认为 False。
- **down_proxy_url** -- 可选参数，默认为 None。
- **extract_folder** -- 可选参数，默认为 None。可以填 `front` 或 `back`。
- **site_id** -- 可选参数。sharepoint 站点 ID。默认为 None。
- **root_folder** -- 可选参数。根目录路径。默认为 None。
- **order_by** -- 可选参数。排序依据。默认为 None。可以填 `name`、`size` 、`lastModifiedDateTime`。
- **order_direction** -- 可选参数。排序方向。默认为 None。可以填 `asc` 或 `desc`。

返回

创建成功返回 True。否则触发异常。

create_Native(*name*, *root_folder*, *webdav_direct=False*, ***kwargs*)

创建一个本地账号。

```
client.admin.account.create_Native('/native_tmp', '/tmp')
```

参数

- **name** -- 账号名字。即虚拟地址。
- **root_folder** -- 根目录路径。
- **webdav_direct** -- 默认为 False。
- **down_proxy_url** -- 可选参数，默认为 None。
- **extract_folder** -- 可选参数，默认为 None。可以填 `front` 或 `back`。
- **order_by** -- 可选参数。排序依据。默认为 None。可以填 `name`、`size` 、`lastModifiedDateTime`。
- **order_direction** -- 可选参数。排序方向。默认为 None。可以填 `asc` 或 `desc`。

返回

创建成功返回 True。否则触发异常。

create_Alist(*name*, *site_url*, *access_token*, *proxy=False*, *webdav_proxy=False*, *webdav_direct=False*, ***kwargs*)

创建一个 Alist 账号

```
client.admin.account.create_Alist('/another_alist', 'http://alist.xxxx.com',
↪'xxxxxx')
```

参数

- **name** -- 账号名字。即虚拟地址。
- **site_url** -- Alist 网站的 URL。
- **access_token** -- 访问网站的 token。
- **proxy** -- 是否开启代码。默认为 False。
- **webdav_proxy** -- 默认为 False。
- **webdav_direct** -- 默认为 False。
- **down_proxy_url** -- 可选参数，默认为 None。
- **extract_folder** -- 可选参数，默认为 None。可以填 `front` 或 `back`。
- **root_folder** -- 可选参数。根目录路径。

返回

创建成功返回 True。否则触发异常。

delete(*name*)

删除一个账户。

```
client.admin.account.delete('/another_alist')
```

参数

name -- 账号名字。即虚拟地址。

返回

删除成功返回 True。否则触发异常。

save(*account: AlistAccount*)

保存修改后的账户。

```
# 首先获取已有的账号。
account = client.admin.accounts.get_account('/native_tmp')
# 将 root_folder 修改为 /var
account['root_folder'] = '/var'
```

(续下页)

(接上页)

```
# 保存
client.admin.account.save(account)
```

参数

account -- 修改后的账号。

返回

修改成功返回 True。否则触发异常。

class alist.account.**AlistAdminAccounts**(*alist*, *endpoint*)

基类：object

alist 账号列表。获取账号。

accounts = []

get() → list

获取账号列表。

```
accounts = client.admin.accounts.get()
# or
accounts = client.admin.accounts()
```

返回

账号列表。*AlistAccount* 组成的列表。

get_account(*id_or_name*) → *AlistAccount*

根据账号 ID 或名字获取账号。

参数

id_or_name -- 账号 ID 或名字。

返回

账号。

alist.admin

class alist.admin.**AlistAdmin**(*alist*)

基类：object

'/api/admin' 相关的 API

login()

登录。不建议直接使用此接口。

```
result = client.admin.login()
```

返回

登录成功返回 Ture，登录失败触发异常。

clear_cache()

清理所有的缓存数据。

```
client.admin.clear_cache()
```

返回

清理成功返回 True，清理失败触发异常。

link(*path*)

返回真实的链接，且携带头，只提供给中转程序使用。

```
link = client.admin.link('/path/to/file')
```

参数

path -- 文件路径。

返回

真实的链接。

files(*path*, *names*)

删除指定路径下的若干个文件和文件夹。

```
# 删除文件 '/path/file' 和文件夹 '/path/dir'。
result = client.admin.files('/path', ['file', 'dir'])
```

参数

- **path** -- 文件所在路径。
- **names** (*list*) -- 文件名和文件夹列表。

返回

删除成功返回 True。

mkdir(*path*)

创建文件夹。

```
client.admin.mkdir('/path/to/new-dir')
```

参数

path -- 新文件夹的路径

返回

创建成功放回 True。创建失败触发异常。

rename(*path*, *name*)

重命名文件或文件名

```
# 将文件 '/path/to/old-name' 重命名为 '/path/to/new-name'
client.admin.rename('/path/to/old-name', 'new-name')
```

参数

- **path** -- 旧文件名，完整路径
- **name** -- 新文件名，不带路径

返回

重命名成功返回 True

move(*src_dir*, *dst_dir*, *names*)

移动文件和文件夹。

```
# 将文件 '/path/to/old/file' 移动到 '/path/to/new/file'
# 将文件 '/path/to/old/dir' 移动到 '/path/to/new/dir'

client.admin.move('/path/to/old', '/path/to/new', ['file', 'dir'])
```

参数

- **src_dir** -- 源文件夹
- **dst_dir** -- 目的文件夹
- **names** (*list*) -- 文件/文件夹列表

返回

移动成功返回 True

copy(*src_dir*, *dst_dir*, *names*)

复制文件和文件夹。

```
# 将文件 '/path/to/old/file' 复制到 '/path/to/new/file'
# 将文件 '/path/to/old/dir' 复制到 '/path/to/new/dir'

client.admin.copy('/path/to/old', '/path/to/new', ['file', 'dir'])
```

参数

- **src_dir** -- 源文件夹
- **dst_dir** -- 目的文件夹
- **names** (*list*) -- 文件/文件夹列表

返回

复制成功返回 True

folder(*path*)

获取指定路径下的所有文件夹。

```
result = client.admin.folder('/path')
```

参数

path -- 指定路径。

返回

文件夹列表。

refresh(*path*)

刷新指定路径。

```
client.admin.refresh('/path')
```

参数

path -- 刷新的路径。

返回

刷新成功返回 True，刷新失败触发异常。

admin setting

class alist.admin.**AlistAdmin**

setting_version()

获取 version

```
version = client.admin.setting_version()
```

setting_title(*new=None*)

获取或更新 title。

参数

new -- 如果不为 None，则更新 title

```
# 更新标题为'new title'
client.admin.setting_title('new title')
```

类似的 API 还有：

- setting_logo(new = None)
- setting_favicon(new = None)
- setting_icon_color(new = None)
- setting_announcement(new = None)
- setting_text_types(new = None)
- setting_audio_types(new = None)
- setting_video_types(new = None)
- setting_hide_files(new = None)
- setting_music_cover(new = None)
- setting_site_beian(new = None)
- setting_global_readme_url(new = None)
- setting_pdf_viewer_url(new = None)
- setting_autoplay_video(new = None)
- setting_autoplay_audio(new = None)
- setting_customize_head(new = None)
- setting_customize_body(new = None)
- setting_home_emoji(new = None)
- setting_animation(new = None)
- setting_artplayer_whitelist(new = None)
- setting_artplayer_autoSize(new = None)
- setting_load_type(new = None)
- setting_default_page_size(new = None)
- setting_password(new = None)
- setting_d_proxy_types(new = None)
- setting_check_parent_folder(new = None)
- setting_check_down_link(new = None)
- setting_WebDAV_username(new = None)

- setting_WebDAV_password(new = None)
- setting_Visitor_WebDAV_username(new = None)
- setting_Visitor_WebDAV_password(new = None)
- setting_ocr_api(new = None)
- setting_enable_search(new = None)
- setting_Aria2_RPC_url(new = None)
- setting_Aria2_RPC_secret(new = None)

alist.driver

class alist.driver.**AlistDriverAttribute**(**attr*)

基类：dict

驱动属性。驱动包含 name、label、type、default、values、required、description 等字段。所有字段初始化之后无法修改。

get_name()

获取驱动属性的名字

is_required()

属性是否必须提供。返回 True 表示必须提供。

class alist.driver.**AlistDriver**(*name*, *attrs*)

基类：object

描述 Alist 驱动。一个驱动包含若干个属性。

get_attr(*name*) → *AlistDriverAttribute*

获取驱动属性。

参数

name -- 属性的名字

get_name()

获取驱动的名字

get_required()

获取驱动必须提供的属性

class alist.driver.**AlistAdminDrivers**(*alist*, *endpoint*)

基类：object

驱动列表。api /api/admin/drivers 的实现。

drivers = []

get()

获取所有驱动的列表，包含驱动必须提供的属性。

返回

驱动列表

get_driver(*name*) → *AlistDriver*

获取指定名字的驱动。

参数

name -- 驱动的名字。

alist.meta

class alist.meta.**AlistMeta**(**kwargs*)

基类：dict

描述 Alist meta 信息。meta 包含 “path“、password、hide、only_shows、upload、readme 和 id 字段。其中 hide 和 only_shows 是列表。

class alist.meta.**AlistAdminMeta**(*alist*, *endpoint: str*)

基类：object

/api/admin/meta 相关 API 的实现。

create(*path*, *password=None*, *hide=None*, *only_shows=None*, *upload=False*, *readme=None*)

创建 meta

参数

- **path** -- 路径
- **password** -- 访问密码
- **hide** (*list*) -- 隐藏文件列表
- **only_shows** (*list*) -- 允许显示的文件列表
- **upload** -- 允许游客上传
- **readme** -- readme url

```
>>> alist.admin.meta.create('/path', password='123', hide=['README.md'])
True
```

delete(*path*)

删除 meta

参数

path -- 路径

```
>>> alist.admin.meta.delete('/path')
True
```

save(*meta:* AlistMeta)

修改 meta 的设置并保存

参数

meta (AlistMeta) -- meta 信息

```
>>> meta = client.admin.metas.get_meta('/path')
>>> meta['password'] = '789'
>>> meta['upload'] = True
>>> client.admin.meta.save(meta)
True
```

class alist.meta.**AlistAdminMetas**(*alist*, *endpoint*)

基类：object

API /api/admin/metas 的实现。

metas = []

get()

获取 meta 列表

```
>>> client.admin.metas.get()
[{'path': '/path', 'password': '789', 'hide': 'README.md', 'only_shows': '',
↪'upload': True, 'readme': '', 'id': 1}]
```

get_meta(*id_or_path*) → *AlistMeta*

获取指定 meta

参数

id_or_path -- meta id 或者是 meta path

```
>>> client.admin.metas.get_meta('/path')
{'path': '/path', 'password': '789', 'hide': 'README.md', 'only_shows': '',
↪'upload': True, 'readme': '', 'id': 1}
```

alist.public

class alist.public.**AlistPublic**(*alist*)

基类：object

path(*path*, *page_num=1*, *page_size=30*, *password=None*)

获取指定路径 path 下的文件和文件夹列表。

参数

- **path** -- 路径
- **page_num** -- 默认为 1
- **page_size** -- 默认为 30
- **password** -- 路径的访问密码。

返回

文件或目录属性，以及文件列表。

```
files = client.public.path('/path')
print(files['files'])
```

preview(*path*)

获取文件的预览 URL。

参数

path -- 文件路径

返回

文件的预览 URL。

search(*path*, *keyword*)

搜索文件。需要开启设置 enable search。默认是关闭的。

参数

- **path** -- 在此路径下搜索。
- **keyword** -- 搜索关键字

返回

搜索的文件列表。

upload(*files*, *path*, *password=None*)

上传文件到指定路径。如果没有登录，则需要开启允许游客上传。参考 meta。

参数

- **files**(*list*) -- 文件列表

- **path** -- 上传的路径
- **password** -- 访问密码

返回

上传结果。True 表示成功，False 表示失败。

```
client.public.upload(['/path/to/local_file'], '/path/to/upload/path')
```

public setting

class alist.public.**AlistPublic**

setting_version()

获取 version。

```
version = client.public.setting_version()
```

类似的 API 还有：

- setting_title()
- setting_logo()
- setting_favicon()
- setting_icon_color()
- setting_announcement()
- setting_text_types()
- setting_audio_types()
- setting_video_types()
- setting_hide_files()
- setting_music_cover()
- setting_site_beian()
- setting_global_readme_url()
- setting_pdf_viewer_url()
- setting_autoplay_video()
- setting_autoplay_audio()
- setting_home_emoji()
- setting_animation()

- setting_check_down_link()
- setting_artplayer_whitelist()
- setting_artplayer_autoSize()
- setting_load_type()
- setting_default_page_size()
- setting_enable_search()
- setting_no_cors()
- setting_no_upload()

alist.utils

alist.utils.**get_timestamp**()

alist.utils.**calc_authorization**(*password*)

根据密码计算出 authorization :param password: 登录密码:returns: authorization

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python 模块索引

a

索引

F

G

I

L

M

O

P

R

S

U

模块