Universal Downloader

Release 2.0.3

Iceflower S

Sep 05, 2022

CONTENTS

CHAPTER

ONE

USER GUIDE

1.1 Installation

1.1.1 System Requirements

Python is required with version 3.8. or higher, it can be downloaded at https://www.python.org/downloads/.

During the Windows installation you should make sure that the `PATH` / environment variable is set and `pip` is installed.

Under Linux it should be ensured that `pip` is installed, if this is not done within the standard installation.

1.1.2 Installation

The installation is preferably done over the terminal.

Pip

Installation with pip.

```
pip install unidown
```

1.2 Using UniDown

The program is a terminal program, so it runs from the terminal.

Calling with:

```
unidown
```

Furthermore, there are additional arguments:

-h, --help
: show this help message and exit

-v, --version
: show program's version number and exit

--list-plugins
: show plugin list and exit

-p name, **--plugin** name
: plugin to execute

-o option [option ...], **--option** option [option ...]
: options passed to the plugin, e.g. *-o username=South American coati -o password=Nasua Nasua*

--logfile path
: log filepath relativ to the main dir (default: ./unidown.log)

-l {DEBUG,INFO,WARNING,ERROR,CRITICAL}, **--log** {DEBUG,INFO,WARNING,ERROR,CRITICAL}
: set the logging level (default: INFO)

CHAPTER

TWO

CHANGELOG

All notable changes to this project will be documented in this file.

The format is based on Keep a Changelog and this project adheres to Semantic Versioning.

Versions before 2.1.0 are not documented.

2.1 2.1.0

2.1.1 Breaking Changes

- Removed `__version__` in `__init__.py`
- Renamed `PluginException` to `PluginError`
- Renamed `PluginState` enums to uppercase
- Moved `get_plugins()` to module level out of `APlugin`

CHAPTER

THREE

WRITING PLUGINS

3.1 Structure

3.1.1 Variables

_INFO
information about the plugin, must be set everytime

_SAVESTATE_CLS
must be set if a custom SaveState format is in use

_simul_downloads
adjust it to a low value to reduce the load on the target server

_options
options, passed by the command line, `delay` will be set to 0 in absence of a value, set it to a higher value to reduce the load on the target server

_unit
unit displayed while downloading

3.1.2 Methods

_create_last_update_time
returns the update time of the complete data e.g. the newest item in the collection If for some reasons its not easily collectable or not available or want to check the links every time, return the current time.

_create_download_data
returns a LinkItemDict, with links and their update time

_load_default_options
override if you need your own default options

load_savestate
override if you have your own custom savestate

update_savestate
override if you have your own custom savestate

3.1.3 LinkItemDict

The LinkItemDict is an essential part of unidown. It is a normal dictionary with some special function. The key is the link as a string. The value is a LinkItem.

3.1.4 LinkItem

LinkItem has two essential values `name` and `time`, the used name is at the same time the file given at downloading.

3.2 Hello World

We will use the test plugin, to show how to create a plugin for unidown.

3.2.1 Basics

First of all we subclass from *unidown.plugin.a_plugin.APlugin*, to have all essential variables and functions for free.

```
from unidown.plugin import APlugin

class Plugin(APlugin):
```

Next part is to add basic information about the plugin like name, version and server host.

```
class Plugin(APlugin):
    _INFO = PluginInfo('test', '0.1.0', 'raw.githubusercontent.com')
```

Next we will hook into the options loading to set default options if nothing was passed.

```
class Plugin(APlugin):
    _INFO = PluginInfo('test', '0.1.0', 'raw.githubusercontent.com')

def _load_default_options(self):
    super(Plugin, self)._load_default_options()
    if 'username' not in self._options:
        self._options['username'] = ''
```

Now as the username is in the options for sure we can just get it safely.

```
class Plugin(APlugin):
    _INFO = PluginInfo('test', '0.1.0', 'raw.githubusercontent.com')

def __init__(self, settings: Settings, options: Dict[str, Any] = None):
    super().__init__(settings, options)
    self._username: str = self._options['username']

def _load_default_options(self):
    super(Plugin, self)._load_default_options()
    if 'username' not in self._options:
        self._options['username'] = ''
```

To get an overall update time for the complete data set, create the `_create_last_update_time` method. If the time generated by this function is older compared to the savestate it will run the individual link generating, if not it will skip the rest.

```
class Plugin(APlugin):
    _INFO = PluginInfo('test', '0.1.0', 'raw.githubusercontent.com')

def __init__(self, settings: Settings, options: Dict[str, Any] = None):
    super().__init__(settings, options)
    self._username: str = self._options['username']

def _create_last_update_time(self) -> datetime:
    self.download_as_file('/IceflowRE/unidown/main/tests/last_update_time.txt', self._
↪temp_dir, 'last_update_time.txt')
    with self._temp_dir.joinpath('last_update_time.txt').open(encoding='utf8') as reader:
        return datetime.strptime(reader.read(), LinkItem.TIME_FORMAT)

def _load_default_options(self):
    super(Plugin, self)._load_default_options()
    if 'username' not in self._options:
        self._options['username'] = ''
```

So if the generate update time is newer, the application continues to `_create_download_links`. In the example we just download a json with preconfigured values and create the `LinkItemDict` from it.

```
class Plugin(APlugin):
    _INFO = PluginInfo('test', '0.1.0', 'raw.githubusercontent.com')

def __init__(self, settings: Settings, options: Dict[str, Any] = None):
    super().__init__(settings, options)
    self._username: str = self._options['username']

def _create_last_update_time(self) -> datetime:
    self.download_as_file('/IceflowRE/unidown/main/tests/last_update_time.txt', self._
↪temp_dir, 'last_update_time.txt')
    with self._temp_dir.joinpath('last_update_time.txt').open(encoding='utf8') as reader:
        return datetime.strptime(reader.read(), LinkItem.TIME_FORMAT)

def _create_download_data(self) -> LinkItemDict:
    self.download_as_file('/IceflowRE/unidown/main/tests/item_dict.json', self._temp_dir,
↪ 'item_dict.json')
    with self._temp_dir.joinpath('item_dict.json').open(encoding='utf8') as reader:
        data = json.loads(reader.read())
    result = LinkItemDict()
    for link, item in data.items():
        result[link] = LinkItem(item['name'], datetime.strptime(item['time'], LinkItem.
↪TIME_FORMAT))
    return result

def _load_default_options(self):
    super(Plugin, self)._load_default_options()
    if 'username' not in self._options:
        self._options['username'] = ''
```

Final step includes that we create a python package out of the generate file.

```
unidown_test/
├── __init__.py
├── plugin.py  «the plugin file»
└── setup.py
```

The most important part is the entry point, otherwise unidown cannot recognize it.

`'unidown.plugin': "test = unidown_test.plugin:Plugin"` Where as `test` is the name of the plugin and after that the import path the plugin class.

Listing 1: setup.py

```python
from setuptools import find_packages, setup

setup(
    name="unidown_test",
    version="0.1.0",
    description="Test plugin for unidown.",
    author="Iceflower S",
    author_email="iceflower@gmx.de",
    license='MIT',
    packages=find_packages(include=['unidown_test', 'unidown_test.*']),
    python_requires='>=3.7',
    install_requires=[
        'unidown',
    ],
    entry_points={
        'unidown.plugin': "test = unidown_test.plugin:Plugin"
    },
)
```

3.2.2 Advanced

The advanced part will show how to use a custom savestate.

First of all we have to create the custom savestate. It is required to subclass from *unidown.plugin.savestate.SaveState*.

```python
from unidown.plugin.savestate import SaveState

class MySaveState(SaveState):
```

In our example we want to permanently store a username.

```python
class MySaveState(SaveState):

    def __init__(self, plugin_info: PluginInfo, last_update: datetime, link_items:␣
↪LinkItemDict, username: str = ''):
        super().__init__(plugin_info, last_update, link_items)
        self.username: str = username
```

Furthermore to get it loaded from a custom json file we have to override `from_json`, the same goes with saving with `to_json`.

```
class MySaveState(SaveState):

    def __init__(self, plugin_info: PluginInfo, last_update: datetime, link_items:␣
↪LinkItemDict, username: str = ''):
        super().__init__(plugin_info, last_update, link_items)
        self.username: str = username

    @classmethod
    def from_json(cls, data: dict) -> SaveState:
        savestate = super(MySaveState, cls).from_json(data)
        if 'username' in data:
            savestate.username = data['username']
        return savestate

    def to_json(self) -> dict:
        data = super().to_json()
        data['username'] = self.username
        return data
```

Additionally it's required to provide an own `upgrade` method, in case the format changes and as the super method may erase your own set variables while upgrading.

```
class MySaveState(SaveState):

    def __init__(self, plugin_info: PluginInfo, last_update: datetime, link_items:␣
↪LinkItemDict, username: str = ''):
        super().__init__(plugin_info, last_update, link_items)
        self.username: str = username

    @classmethod
    def from_json(cls, data: dict) -> SaveState:
        savestate = super(MySaveState, cls).from_json(data)
        if 'username' in data:
            savestate.username = data['username']
        return savestate

    def to_json(self) -> dict:
        data = super().to_json()
        data['username'] = self.username
            return data

    def upgrade(self) -> SaveState:
        new_savestate = super(MySaveState, self).upgrade()
        new_savestate.username = self.username
        return new_savestate
```

The next step is to register your own custom savestate in the plugin, so it will be used, by setting `_SAVESTATE_CLS`.

```
from unidown_test.savestate import MySaveState

class Plugin(APlugin):
    _INFO = PluginInfo('test', '0.1.0', 'raw.githubusercontent.com')
    _SAVESTATE_CLS = MySaveState
```

To get the username loaded into your plugin, after the savestate was loaded override `load_savestate`.

```
from unidown_test.savestate import MySaveState

class Plugin(APlugin):
    _INFO = PluginInfo('test', '0.1.0', 'raw.githubusercontent.com')
    _SAVESTATE_CLS = MySaveState

# ... all other stuff in between

def load_savestate(self):
    super(Plugin, self).load_savestate()
    # do not override set username by options
    if self._username == '':
        self._username = self.savestate.username
```

The last step is to hook into updating the savestate. To get the current username saved into a new savestate.

```
from unidown_test.savestate import MySaveState

class Plugin(APlugin):
    _INFO = PluginInfo('test', '0.1.0', 'raw.githubusercontent.com')
    _SAVESTATE_CLS = MySaveState

# ... all other stuff in between

def load_savestate(self):
    super(Plugin, self).load_savestate()
    # do not override set username by options
    if self._username == '':
        self._username = self.savestate.username

def update_savestate(self, new_items: LinkItemDict):
    super(Plugin, self).update_savestate(new_items)
    self._savestate.username = self._username
```

CHAPTER

FOUR

DEVELOPMENT

4.1 Continuous Integration

4.1.1 Github Action Secrets

PYPI_TOKEN
: PYPI token for the project.

CODACY_PROJECT_TOKEN
: Codacy token for the project.

4.2 Functionality

1. Get plugin from the given name
2. Get the last overall update time
3. Load the savestate
4. Compare last update time with the one from the savestate
5. Get the download links
6. Compare received links and their times with the savestate
7. Clean up names, to eliminate duplicated
8. Download new and newer links
9. Check downloaded data
10. Update savestate
11. Save new savestate to file

CHAPTER

FIVE

INTERNALS

Reference material.

5.1 unidown

Modules

5.1.1 unidown.core

Core utility.

Modules

unidown.core.manager

Manager of the whole program, contains the most important functions as well as the download routine.

Functions

check_update()	Check for app updates and print/log them.
download_from_plugin(plugin)	Download routine.
get_options(options)	Convert the option list to a dictionary where the key is the option and the value is the related option.
init_logging(settings)	Initialize the _downloader.
run(settings, plugin_name, raw_options)	Run a plugin so use the download routine and clean up after.
shutdown()	Close and exit important things.

check_update()

Check for app updates and print/log them.

Return type
None

download_from_plugin(*plugin*)

Download routine.

1. Get plugin from the given name
2. Get the last overall update time
3. Load the savestate
4. Compare last update time with the one from the savestate
5. Get the download links
6. Compare received links and their times with the savestate
7. Clean up names, to eliminate duplicated
8. Download new and newer links
9. Check downloaded data
10. Update savestate
11. Save new savestate to file

Parameters
plugin (*APlugin*) – Plugin.

Return type
None

get_options(*options*)

Convert the option list to a dictionary where the key is the option and the value is the related option. Is called in the init.

Parameters
options (list[list[str]]) – Options given to the plugin.

Return type
dict[str, Any]

Returns
Dictionary which contains the option key and values.

init_logging(*settings*)

Initialize the _downloader.

Parameters
settings (*Settings*) – Settings.

Return type
None

run(*settings*, *plugin_name*, *raw_options*)

Run a plugin so use the download routine and clean up after.

Parameters

- **settings** (*Settings*) – Settings to use.
- **plugin_name** (str) – Name of plugin.
- **raw_options** (list[list[str]]) – Parameters which will be sent to the plugin initialization.

Return type
PluginState

Returns
Ending state.

shutdown()

Close and exit important things.

Return type
None

unidown.core.plugin_state

Classes

PluginState(value)	State of a plugin, after it ended or was not found.

class PluginState(*value*)

Bases: IntEnum

State of a plugin, after it ended or was not found.

END_SUCCESS = 0

successfully end

LOAD_CRASH = 3

Exception was raised while loading/ initializing.

NOT_FOUND = 4

Plugin was not found.

RUN_CRASH = 2

Exception was raised but not *PluginError*.

RUN_FAIL = 1

PluginError was raised.

unidown.core.settings

Classes

Settings([root_dir, log_file, log_level])	Settings.

class Settings(*root_dir=None*, *log_file=None*, *log_level='INFO'*)

Bases: object

Settings.

Parameters

- **root_dir** (Optional[Path]) – root dir
- **log_file** (Optional[Path]) – log file
- **log_level** (str) –

check_dirs()

Check the directories if they exist.

Raises
FileExistsError – if a file exists but is not a directory

Return type
None

mkdir()

Create all base directories.

Return type
None

_cores

Number of cores to be used.

_disable_tqdm

Disable console progress bar.

_download_dir: Path

Download main path, here are the sub folders for every plugin.

_log_file: Path

Log file of the program.

_log_level

Log level.

_root_dir: Path

Root path.

_savestate_dir: Path

Savestates main path, here are the sub folders for every plugin.

_temp_dir: Path

Temporary main path, here are the sub folders for every plugin.

property cores: int

Plain getter.

Return type
int

property disable_tqdm: bool

Plain getter.

Return type
bool

property download_dir: Path

Plain getter.

Return type
Path

property log_file: Path

Plain getter.

Return type
Path

property log_level: str

Plain getter.

Return type
str

property root_dir: Path

Plain getter.

Return type
Path

property savestate_dir: Path

Plain getter.

Return type
Path

property temp_dir: Path

Plain getter.

Return type
Path

unidown.core.updater

Things needed for checking for updates.

Functions

check_for_app_updates()	Check for updates.
get_newest_app_version()	Download the version tag from remote.

check_for_app_updates()

Check for updates.

Return type
bool

Returns
Is update available.

get_newest_app_version()

Download the version tag from remote.

Return type
Version

Returns
Version from remote.

5.1.2 unidown.main

Entry into the program.

Functions

main([argv])	Entry point into the program.

Classes

PluginListAction(option_strings, dest, **kwargs)	Lists all plugins which are available.

class PluginListAction(*option_strings*, *dest*, ***kwargs*)

Bases: Action

Lists all plugins which are available. Extension for ArgumentParser.

Parameters

- **option_strings** (Sequence[str]) –
- **dest** (str) –

main(*argv=None*)

Entry point into the program. Gets the arguments from the console and proceed them with ArgumentParser. Returns if its success successful 0 else 1.

Parameters
argv (Optional[list[str]]) –

Return type
None

5.1.3 unidown.meta

Metadata about the application.

Module Attributes

NAME	name of this program
LONG_NAME	long name of this program
VERSION	version in PEP440 format
PYPI_JSON_URL	url to the unidown pypi json
AUTHOR	author
AUTHOR_GITHUB	author github link
AUTHOR_EMAIL	author email
DESCRIPTION	short description
PROJECT_URL	project url

AUTHOR = 'Iceflower S'

author

AUTHOR_EMAIL = 'iceflower@gmx.de'

author email

AUTHOR_GITHUB = 'https://github.com/IceflowRE'

author github link

DESCRIPTION = 'Universal downloader, a modular extensible downloader who manage progress and updates.'

short description

LONG_NAME = 'Universal Downloader'

long name of this program

NAME = 'unidown'

name of this program

PROJECT_URL = 'https://github.com/IceflowRE/unidown'

project url

PYPI_JSON_URL = 'https://pypi.org/pypi/unidown/json'

url to the unidown pypi json

VERSION = '2.0.3'

version in PEP440 format

5.1.4 unidown.plugin

exception PluginError(*msg=''*)

Bases: Exception

Base class for exceptions in a plugin. If catching this, it is implicit that the plugin is unable to work further.

Parameters
msg (str) – message

msg: str

Exception message.

class APlugin(*settings*, *options=None*)

Bases: ABC

Abstract class of a plugin. Provides all needed variables and methods.

Parameters

- **options** (Optional[dict[str, Any]]) – Parameters which can include optional parameters.
- **settings** (*Settings*) –

Raises
PluginError – Can not create default plugin paths.

_SAVESTATE_CLS

alias of *SaveState*

abstract _create_download_data()

Get the download links in a specific format.

Note: Has to be implemented inside Plugins.

Raises
NotImplementedError – Abstract method.

Return type
LinkItemDict

abstract _create_last_update_time()

Get the newest update time from the referencing data.

Note: Has to be implemented inside Plugin.

Raises
NotImplementedError – Abstract method.

Return type
datetime

_load_default_options()

Load default options if they were not passed at creation.

Return type
None

check_download(*link_item_dict*, *folder*, *log=False*)

Check if the download of the given dict was successful. No proving if the content of the file is correct too.

Parameters

- **link_item_dict** (*LinkItemDict*) – Items to check.
- **folder** (Path) – Folder where the downloads are saved.
- **log** (bool) – Log lost items.

Return type
tuple[*LinkItemDict*, *LinkItemDict*]

Returns
Succeed.

clean_up()

Clean up for a module. Is deleting *_temp_dir*.

Return type
None

download(*link_items*, *folder*, *desc*, *unit*)

Download the given LinkItem dict from the plugins host, to the given path. Proceeded with multiple connections. *_simul_downloads*. After *check_download()* is recommended.

This function don't use an internal *link_item_dict*, *delay* or *folder* directly set in options or instance vars, because it can be used aside of the normal download routine inside the plugin itself for own things. As of this it still needs access to the logger, so a staticmethod is not possible.

Warning: The parameters may change in future versions. (e.g. change order and accept another host)

Parameters

- **link_items** (*LinkItemDict*) – Data which gets downloaded.
- **folder** (Path) – Target download folder.
- **desc** (str) – Description of the progressbar.
- **unit** (str) – Unit of the download, shown in the progressbar.

Return type
None

download_as_file(*url*, *target_file*, *delay=0*)

Download the given url to the given target folder.

Parameters

- **url** (str) – Link.
- **target_file** (Path) – Target file.
- **delay** (float) – Delay after each download. Delay is in seconds.

Return type
str

Returns
Url.

Raises
HTTPError – Connection had an error.

load_savestate()
Load the save of the plugin.

Raises
- ***PluginError*** – Broken savestate json.
- ***PluginError*** – Different savestate versions.
- ***PluginError*** – Different plugin versions.
- ***PluginError*** – Different plugin names.
- ***PluginError*** – Could not parse the json.

Return type
None

save_savestate()
Save meta data about the downloaded things and the plugin to a file.

Return type
None

update_download_data()
Update the download links. Calls *_create_download_data()*.

Return type
None

update_last_update()
Call this to update the latest update time. Calls *_create_last_update_time()*.

Return type
None

update_savestate(*new_items*)
Update savestate.

Parameters
new_items (*LinkItemDict*) – New items.

Return type
None

_INFO: *PluginInfo* = <unidown.plugin.plugin_info.PluginInfo object>
Meta information about the plugin.

_abc_impl = <_abc._abc_data object>

_disable_tqdm: bool
If the tqdm progressbar should be disabled.

_download_data: *LinkItemDict*
Referencing data.

_download_dir: *Path*

General download path of the plugin.

_downloader: *HTTPSConnectionPool*

Downloader which will download the data.

_last_update: *datetime*

Latest update time of the referencing data.

_log: *Logger*

Use this for logging.

_options: *dict[str, Any]*

Options which the plugin uses internally, should be used for the given options at initialization.

_savestate: *SaveState*

Savestate of the plugin.

_savestate_file: *Path*

File which contains the latest savestate of the plugin.

_simul_downloads: *int*

Number of simultaneous downloads.

_temp_dir: *Path*

Path where the plugin can place all temporary data.

_unit: *str*

The unit which will be displayed in the progress bar.

property download_data: *LinkItemDict*

Plain getter.

Return type
LinkItemDict

property download_dir: *Path*

Plain getter.

Return type
Path

property host: *str*

Plain getter.

Return type
str

property info: *PluginInfo*

Plain getter.

Return type
PluginInfo

property last_update: *datetime*

Plain getter.

Return type
datetime

property log: Logger

Plain getter.

Return type
Logger

property name: str

Plain getter.

Return type
str

property options: dict[str, Any]

Plain getter.

Return type
dict[str, Any]

property savestate: *SaveState*

Plain getter.

Return type
SaveState

property simul_downloads: int

Plain getter.

Return type
int

property temp_dir: Path

Plain getter.

Return type
Path

property unit: str

Plain getter.

Return type
str

property version: Version

Plain getter.

Return type
Version

class LinkItem(*name*, *time*)

Bases: object

Item which represents the data, who need to be downloaded. Has a name and an update time.

Parameters

- **name** (str) – name
- **time** (datetime) – update time

Raises

- **ValueError** – name cannot be empty or None

- **ValueError** – time cannot be empty or None

classmethod from_json(*data*)

Construct from json dict.

Parameters
data (dict) – Json data as dict.

Return type
LinkItem

Returns
LinkItem.

Raises
ValueError – Missing parameter.

to_json()

Create json data.

Return type
dict

Returns
Json dictionary.

TIME_FORMAT: str = '%Y%m%dT%H%M%S.%fZ'

Time format to use.

_name: str

Name of the item.

_time: datetime

Time of the item.

property name: str

Plain getter.

Return type
str

property time: datetime

Plain getter.

Return type
datetime

class LinkItemDict

Bases: dict

LinkItem dictionary, acts as a wrapper for special methods and functions.

actualize(*new_data*, *log=None*)

Actualize dictionary like ~dict.update does. If a logger is passed it will log updated items, **not** new one.

Parameters

- **new_data** (*LinkItemDict*) – Data used for updating.
- **log** (Optional[Logger]) – Logger.

Return type
None

clean_up_names()

Rename duplicated names with an additional _r.

Return type
None

static get_new_items(*old_data*, *new_data*, *disable_tqdm=False*)

Get the new items which are not existing or are newer as in the old data set.

Parameters

- **old_data** (*LinkItemDict*) – Old data.
- **new_data** (*LinkItemDict*) – New data.
- **disable_tqdm** (bool) – Disable tqdm progressbar.

Return type
LinkItemDict

Returns
New and updated link items.

class PluginInfo(*name*, *version*, *host*)

Bases: object

Information about the module. This information will be saved into the save files as well.

Parameters

- **name** (str) – Name of the plugin.
- **version** (str) – Version, PEP440 conform.
- **host** (str) – Host url of the main data.

Raises

- **ValueError** – Name is empty.
- **ValueError** – Host is empty.
- **InvalidVersion** – Version is not PEP440 conform.

classmethod from_json(*data*)

Construct from json dict.

Parameters
data (dict) – Json data as dict.

Return type
PluginInfo

Returns
Plugin info.

Raises

- **ValueError** – Name is missing.
- **ValueError** – Version is missing.
- **ValueError** – Host is missing.

to_json()

Create json data.

Return type
dict

Returns
Json dictionary.

host: str

Host url of the main data.

name: str

Name of the plugin.

version: Version

Plugin version.

class SaveState(*plugin_info*, *last_update*, *link_items*, *version=<Version('1')>*)

Bases: object

Savestate of a plugin.

Parameters

- **version** (Version) – Savestate version.
- **plugin_info** (*PluginInfo*) – Plugin info.
- **last_update** (datetime) – Last udpate time of the referenced data.
- **link_items** (*LinkItemDict*) – Data.
- **version** – Savestate version.

classmethod from_json(*data*)

Parameters
data (dict) – Json data as dict.

Return type
SaveState

Returns
SaveState.

Raises

- **ValueError** – Version of SaveState does not exist or is empty.
- **InvalidVersion** – Version is not PEP440 conform.

to_json()

Create json data.

Return type
dict

Returns
Json dictionary.

upgrade()

Upgrading current savestate to the latest savestate version.

Return type
SaveState

Returns
Upgraded savestate.

TIME_FORMAT: str = '%Y%m%dT%H%M%S.%fZ'

Time format to use.

_DEFAULT_VERSION: Version = <Version('1')>

Default savestate version.

last_update: datetime

Newest update time.

link_items: *LinkItemDict*

Data.

plugin_info: *PluginInfo*

Plugin info.

version: Version

Savestate version.

get_plugins()

Get all available plugins for unidown.

Return type
dict[str, EntryPoint]

Returns
Plugin name list.

Modules

unidown.plugin.a_plugin	
unidown.plugin.exceptions	Default exceptions of plugins.
unidown.plugin.link_item	
unidown.plugin.link_item_dict	
unidown.plugin.plugin_info	
unidown.plugin.savestate	

unidown.plugin.a_plugin

Functions

get_plugins()	Get all available plugins for unidown.

Classes

APlugin(settings[, options])	Abstract class of a plugin.

class APlugin(*settings*, *options=None*)

Bases: ABC

Abstract class of a plugin. Provides all needed variables and methods.

Parameters

- **options** (Optional[dict[str, Any]]) – Parameters which can include optional parameters.
- **settings** (*Settings*) –

Raises

PluginError – Can not create default plugin paths.

_SAVESTATE_CLS

Savestate class to use.

Parameters

- **plugin_info** (*PluginInfo*) –
- **last_update** (datetime) –
- **link_items** (*LinkItemDict*) –
- **version** (Version) –

alias of *SaveState*

abstract _create_download_data()

Get the download links in a specific format.

Note: Has to be implemented inside Plugins.

Raises

NotImplementedError – Abstract method.

Return type

LinkItemDict

abstract _create_last_update_time()

Get the newest update time from the referencing data.

Note: Has to be implemented inside Plugin.

Raises
NotImplementedError – Abstract method.

Return type
datetime

_load_default_options()

Load default options if they were not passed at creation.

Return type
None

check_download(*link_item_dict*, *folder*, *log=False*)

Check if the download of the given dict was successful. No proving if the content of the file is correct too.

Parameters

- **link_item_dict** (*LinkItemDict*) – Items to check.
- **folder** (Path) – Folder where the downloads are saved.
- **log** (bool) – Log lost items.

Return type
tuple[*LinkItemDict*, *LinkItemDict*]

Returns
Succeed.

clean_up()

Clean up for a module. Is deleting *_temp_dir*.

Return type
None

download(*link_items*, *folder*, *desc*, *unit*)

Download the given LinkItem dict from the plugins host, to the given path. Proceeded with multiple connections. *_simul_downloads*. After *check_download()* is recommended.

This function don't use an internal *link_item_dict*, *delay* or *folder* directly set in options or instance vars, because it can be used aside of the normal download routine inside the plugin itself for own things. As of this it still needs access to the logger, so a staticmethod is not possible.

> **Warning:** The parameters may change in future versions. (e.g. change order and accept another host)

Parameters

- **link_items** (*LinkItemDict*) – Data which gets downloaded.
- **folder** (Path) – Target download folder.
- **desc** (str) – Description of the progressbar.
- **unit** (str) – Unit of the download, shown in the progressbar.

Return type
None

download_as_file(*url*, *target_file*, *delay=0*)

Download the given url to the given target folder.

Parameters

- **url** (str) – Link.
- **target_file** (Path) – Target file.
- **delay** (float) – Delay after each download. Delay is in seconds.

Return type
str

Returns
Url.

Raises
HTTPError – Connection had an error.

load_savestate()

Load the save of the plugin.

Raises

- ***PluginError*** – Broken savestate json.
- ***PluginError*** – Different savestate versions.
- ***PluginError*** – Different plugin versions.
- ***PluginError*** – Different plugin names.
- ***PluginError*** – Could not parse the json.

Return type
None

save_savestate()

Save meta data about the downloaded things and the plugin to a file.

Return type
None

update_download_data()

Update the download links. Calls *_create_download_data()*.

Return type
None

update_last_update()

Call this to update the latest update time. Calls *_create_last_update_time()*.

Return type
None

update_savestate(*new_items*)

Update savestate.

Parameters
new_items (*LinkItemDict*) – New items.

Return type
None

_INFO: *PluginInfo* = <unidown.plugin.plugin_info.PluginInfo object>

Meta information about the plugin.

_abc_impl = <_abc._abc_data object>

_disable_tqdm: bool

If the tqdm progressbar should be disabled.

_download_data: *LinkItemDict*

Referencing data.

_download_dir: Path

General download path of the plugin.

_downloader: HTTPSConnectionPool

Downloader which will download the data.

_last_update: datetime

Latest update time of the referencing data.

_log: Logger

Use this for logging.

_options: dict[str, Any]

Options which the plugin uses internally, should be used for the given options at initialization.

_savestate: *SaveState*

Savestate of the plugin.

_savestate_file: Path

File which contains the latest savestate of the plugin.

_simul_downloads: int

Number of simultaneous downloads.

_temp_dir: Path

Path where the plugin can place all temporary data.

_unit: str

The unit which will be displayed in the progress bar.

property download_data: *LinkItemDict*

Plain getter.

Return type
LinkItemDict

property download_dir: Path

Plain getter.

Return type
Path

property host: str

Plain getter.

Return type
str

property info: *PluginInfo*

Plain getter.

Return type
PluginInfo

property last_update: datetime

Plain getter.

Return type
datetime

property log: Logger

Plain getter.

Return type
Logger

property name: str

Plain getter.

Return type
str

property options: dict[str, Any]

Plain getter.

Return type
dict[str, Any]

property savestate: *SaveState*

Plain getter.

Return type
SaveState

property simul_downloads: int

Plain getter.

Return type
int

property temp_dir: Path

Plain getter.

Return type
Path

property unit: str

Plain getter.

Return type
str

property version: Version

Plain getter.

Return type
Version

get_plugins()

Get all available plugins for unidown.

Return type
dict[str, EntryPoint]

Returns
Plugin name list.

unidown.plugin.exceptions

Default exceptions of plugins.

Exceptions

PluginError([msg])	Base class for exceptions in a plugin.

exception PluginError(*msg=''*)

Bases: Exception

Base class for exceptions in a plugin. If catching this, it is implicit that the plugin is unable to work further.

Parameters
msg (str) – message

msg: str

Exception message.

unidown.plugin.link_item

Classes

LinkItem(name, time)	Item which represents the data, who need to be downloaded.

class LinkItem(*name*, *time*)

Bases: object

Item which represents the data, who need to be downloaded. Has a name and an update time.

Parameters

- **name** (str) – name
- **time** (datetime) – update time

Raises

- **ValueError** – name cannot be empty or None
- **ValueError** – time cannot be empty or None

classmethod from_json(*data*)

Construct from json dict.

Parameters
data (dict) – Json data as dict.

Return type
LinkItem

Returns
LinkItem.

Raises
ValueError – Missing parameter.

to_json()

Create json data.

Return type
dict

Returns
Json dictionary.

TIME_FORMAT: str = '%Y%m%dT%H%M%S.%fZ'

Time format to use.

_name: str

Name of the item.

_time: datetime

Time of the item.

property name: str

Plain getter.

Return type
str

property time: datetime

Plain getter.

Return type
datetime

unidown.plugin.link_item_dict

Classes

LinkItemDict	LinkItem dictionary, acts as a wrapper for special methods and functions.

class LinkItemDict

Bases: dict

LinkItem dictionary, acts as a wrapper for special methods and functions.

actualize(*new_data*, *log=None*)

Actualize dictionary like ~dict.update does. If a logger is passed it will log updated items, **not** new one.

Parameters

- **new_data** (*LinkItemDict*) – Data used for updating.
- **log** (Optional[Logger]) – Logger.

Return type
None

clean_up_names()

Rename duplicated names with an additional _r.

Return type
None

static get_new_items(*old_data*, *new_data*, *disable_tqdm=False*)

Get the new items which are not existing or are newer as in the old data set.

Parameters

- **old_data** (*LinkItemDict*) – Old data.
- **new_data** (*LinkItemDict*) – New data.
- **disable_tqdm** (bool) – Disable tqdm progressbar.

Return type
LinkItemDict

Returns
New and updated link items.

unidown.plugin.plugin_info

Classes

PluginInfo(name, version, host)	Information about the module.

class PluginInfo(*name*, *version*, *host*)

Bases: object

Information about the module. This information will be saved into the save files as well.

Parameters

- **name** (str) – Name of the plugin.
- **version** (str) – Version, PEP440 conform.
- **host** (str) – Host url of the main data.

Raises

- **ValueError** – Name is empty.
- **ValueError** – Host is empty.
- **InvalidVersion** – Version is not PEP440 conform.

classmethod from_json(*data*)

Construct from json dict.

Parameters
data (dict) – Json data as dict.

Return type
PluginInfo

Returns
Plugin info.

Raises

- **ValueError** – Name is missing.
- **ValueError** – Version is missing.
- **ValueError** – Host is missing.

to_json()

Create json data.

Return type
dict

Returns
Json dictionary.

host: str

Host url of the main data.

name: str

Name of the plugin.

version: Version

Plugin version.

unidown.plugin.savestate

Classes

SaveState(plugin_info, last_update, link_items)	Savestate of a plugin.

class SaveState(*plugin_info*, *last_update*, *link_items*, *version=<Version('1')>*)

Bases: object

Savestate of a plugin.

Parameters

- **version** (Version) – Savestate version.
- **plugin_info** (*PluginInfo*) – Plugin info.
- **last_update** (datetime) – Last udpate time of the referenced data.
- **link_items** (*LinkItemDict*) – Data.
- **version** – Savestate version.

classmethod from_json(*data*)

Parameters
data (dict) – Json data as dict.

Return type
SaveState

Returns
SaveState.

Raises

- **ValueError** – Version of SaveState does not exist or is empty.
- **InvalidVersion** – Version is not PEP440 conform.

to_json()

Create json data.

Return type
dict

Returns
Json dictionary.

upgrade()

Upgrading current savestate to the latest savestate version.

Return type
SaveState

Returns
Upgraded savestate.

TIME_FORMAT: str = '%Y%m%dT%H%M%S.%fZ'

Time format to use.

_DEFAULT_VERSION: Version = <Version('1')>

Default savestate version.

last_update: datetime

Newest update time.

link_items: *LinkItemDict*

Data.

plugin_info: *PluginInfo*

Plugin info.

version: Version

Savestate version.

5.1.5 unidown.tools

Different tools.

Functions

print_plugin_list(plugins)	Print all registered plugins and checks if they can be loaded or not.
unlink_dir_rec(path)	Delete a folder recursive.

print_plugin_list(*plugins*)

Print all registered plugins and checks if they can be loaded or not.

Parameters
plugins (dict[str, EntryPoint]) – Plugins name to entrypoint.

Return type
None

unlink_dir_rec(*path*)

Delete a folder recursive.

Parameters
path (Path) – Folder to delete.

Return type
None

- modindex

PYTHON MODULE INDEX

u

INDEX

Symbols

A

C

T

U

V