
Clay Documentation

Release 1.0a1

Andrew Dunai

Aug 09, 2018

Contents:

1	app.py	1
2	appsettings.py	3
3	gp.py	5
4	player.py	11
5	songlist.py	13
6	playbar.py	15
7	mylibrary.py	17
8	myplaylists.py	19
9	playerqueue.py	21
10	settings.py	23
11	settings.py	25
12	page	27
13	notifications.py	29
14	hotkeys.py	31
15	eventhook.py	33
16	meta.py	35
	Python Module Index	37

CHAPTER 1

app.py

Application settings manager.

class `clay.settings._Settings`

Settings management class.

`__init__()`

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

`__weakref__`

list of weak references to the object (if defined)

`_commit_edits (config)`

Write config to file.

This method is supposed to be called only from `__exit__()`.

`_ensure_directories()`

Create config dir, config file & cache dir if they do not exist yet.

`_load_cache()`

Load cached files.

`_load_config()`

Read config from file.

`edit()`

Return `_SettingsEditor` context manager to edit config.

Settings are saved to file once the returned context manager exists.

Example usage:

```
from clay.settings import settings

with settings.edit() as config:
    config['foo']['bar'] = 'baz'
```

get (*key*, **sections*)

Return their configuration key in a specified section By default it looks in `play_settings`.

get_cached_file_path (*filename*)

Get full path to cached file.

get_default_config_section (**sections*)

Always get a section from the default/system configuration. You would use this whenever you need to loop through all the values in a section. In the user config they might be incomplete.

get_is_file_cached (*filename*)

Return `True` if *filename* is present in cache.

get_section (**sections*)

Get a section from the user configuration file if it can find it, else load it from the system config

save_file_to_cache (*filename*, *content*)

Save content into file in cache.

class `clay.settings._SettingsEditor` (*original_config*, *commit_callback*)

Thread-safe settings editor context manager.

For example see `edit()`.

__init__ (*original_config*, *commit_callback*)

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

__weakref__

list of weak references to the object (if defined)

Google Play Music integration via gmusicapi.

class `clay.gp.Artist` (*artist_id, name*)

Model that represents an artist.

__init__ (*artist_id, name*)

x.__init__(...) initializes *x*; see `help(type(x))` for signature

__weakref__

 list of weak references to the object (if defined)

classmethod from_data (*data, many=False*)

 Construct and return one or many *Artist* instances from Google Play Music API response.

id

 Artist ID.

class `clay.gp.LikedSongs`

A local model that represents the songs that a user liked and displays them as a faux playlist.

This mirrors the “liked songs” generated playlist feature of the Google Play Music apps.

__init__ ()

x.__init__(...) initializes *x*; see `help(type(x))` for signature

__weakref__

 list of weak references to the object (if defined)

add_liked_song (*song*)

 Add a liked song to the list.

remove_liked_song (*song*)

 Remove a liked song from the list

tracks

 Get a sorted list of liked tracks.

class `clay.gp.Playlist` (*playlist_id, name, tracks*)

Model that represents remotely stored (Google Play Music) playlist.

__init__ (*playlist_id, name, tracks*)
x.__init__(...) initializes x; see help(type(x)) for signature

__weakref__
list of weak references to the object (if defined)

classmethod from_data (*data, many=False*)
Construct and return one or many *Playlist* instances from Google Play Music API response.

id
Playlist ID.

class `clay.gp.SearchResults` (*tracks, artists*)
Model that represents search results including artists & tracks.

__init__ (*tracks, artists*)
x.__init__(...) initializes x; see help(type(x)) for signature

__weakref__
list of weak references to the object (if defined)

classmethod from_data (*data*)
Construct and return *SearchResults* instance from raw data.

get_artists ()
Return found artists.

get_tracks ()
Return found tracks.

class `clay.gp.Station` (*station_id, name*)
Model that represents specific station on Google Play Music.

__init__ (*station_id, name*)
x.__init__(...) initializes x; see help(type(x)) for signature

__weakref__
list of weak references to the object (if defined)

classmethod from_data (*data, many=False*)
Construct and return one or many *Station* instances from Google Play Music API response.

get_tracks ()
Return a list of tracks in this station.

id
Station ID.

load_tracks ()
Fetch tracks related to this station and populate it with *Track* instances.

load_tracks_async (***kwargs*)
Inner function.

class `clay.gp.Track` (*source, data*)
Model that represents single track from Google Play Music.

__eq__ (*other*)
x.__eq__(y) <=> x==y

__init__ (*source, data*)
x.__init__(...) initializes x; see help(type(x)) for signature

__repr__ ()
x.__str__() <==> str(x)

__str__ () <==> str(x)

__weakref__
list of weak references to the object (if defined)

add_to_my_library ()
Add a track to my library.

add_to_my_library_async (**kwargs)
Inner function.

create_station (**kwargs)
Inner function.

create_station_async (**kwargs)
Inner function.

filename
Return a filename for this track.

classmethod from_data (data, source, many=False)
Construct and return one or many [Track](#) instances from Google Play Music API response.

get_artist_art_filename (**kwargs)
Inner function.

get_url (callback)
Gets playable stream URL for this track.
“callback” is called with “(url, error)” args after URL is fetched.
Keep in mind this URL is valid for a limited time.

id
Return ID for this track.

rate_song (rating)
Rate the song either 0 (no thumb), 1 (down thumb) or 5 (up thumb).

remove_from_my_library ()
Remove a track from my library.

remove_from_my_library_async (**kwargs)
Inner function.

class clay.gp._GP
Interface to gmusicapi.Mobileclient. Implements asynchronous API calls, caching and some other perks.
Singleton.

__init__ ()
x.__init__(...) initializes x; see help(type(x)) for signature

__weakref__
list of weak references to the object (if defined)

_make_call_proxy (func)
Return a function that wraps fn and logs args & return values.

add_to_my_library (track)
Add a track to my library.

get_all_tracks (***kwargs*)
Inner function.

get_all_tracks_async (***kwargs*)
Inner function.

get_all_user_playlist_contents (***kwargs*)
Inner function.

get_all_user_playlist_contents_async (***kwargs*)
Inner function.

get_all_user_station_contents (***kwargs*)
Inner function.

get_all_user_station_contents_async (***kwargs*)
Inner function.

get_authtoken ()
Return currently active auth token.

get_cached_tracks_map ()
Return a dictionary of tracks where keys are strings with track IDs and values are *Track* instances.

get_stream_url (*stream_id*)
Returns playable stream URL of track by id.

get_stream_url_async (***kwargs*)
Inner function.

get_track_by_id (*any_id*)
Return track by id or store_id.

invalidate_caches ()
Clear cached tracks & playlists & stations.

is_authenticated
Return True if user is authenticated on Google Play Music, false otherwise.

is_subscribed
Return True if user is subscribed on Google Play Music, false otherwise.

login (***kwargs*)
Inner function.

login_async (***kwargs*)
Inner function.

remove_from_my_library (*track*)
Remove a track from my library.

search (*query*)
Find tracks and return an instance of *SearchResults*.

search_async (***kwargs*)
Inner function.

use_authtoken (***kwargs*)
Inner function.

use_authtoken_async (***kwargs*)
Inner function.

`clay.gp.asynchronous` (*func*)

Decorates a function to become asynchronous.

Once called, runs original function in a new Thread.

Must be called with a 'callback' argument that will be called once thread with original function finishes. Receives two args: result and error.

- "result" contains function return value or None if there was an exception.
- "error" contains None or Exception if there was one.

`clay.gp.synchronized` (*func*)

Decorates a function to become thread-safe by preventing it from being executed multiple times before previous calls end.

Lock is acquired on entrance and is released on return or Exception.

CHAPTER 4

player.py

CHAPTER 5

songlist.py

CHAPTER 6

playbar.py

CHAPTER 7

mylibrary.py

CHAPTER 8

myplaylists.py

CHAPTER 9

playerqueue.py

CHAPTER 10

settings.py

CHAPTER 11

settings.py

Generic page classes.

class `clay.pages.page.AbstractPage`

Represents app page.

__weakref__

list of weak references to the object (if defined)

activate()

Notify page that it is activated.

key

Return page key (`int`), used for hotkeys.

name

Return page name.

Notification widgets.

class `clay.notifications._Notification` (*area, notification_id, message*)

Single notification widget. Can be updated or closed.

`__init__` (*area, notification_id, message*)

Parameters

- **widget_list** – iterable of flow or box widgets
- **dividechars** – number of blank characters between columns
- **focus_column** – index into `widget_list` of column in focus, if `None` the first selectable widget will be chosen.
- **min_width** – minimum width for each column which is not calling `widget.pack()` in `widget_list`.
- **box_columns** – a list of column indexes containing box widgets whose height is set to the maximum of the rows required by columns not listed in `box_columns`.

`widget_list` may also contain tuples such as:

(*given_width, widget*) make this column *given_width* screen columns wide, where *given_width* is an int

('pack', *widget*) call `pack()` to calculate the width of this column

('weight', *weight, widget*) give this column a relative *weight* (number) to calculate its width from the screen columns remaining

Widgets not in a tuple are the same as ('weight', 1, *widget*)

If the Columns widget is treated as a box widget then all children are treated as box widgets, and `box_columns` is ignored.

If the Columns widget is treated as a flow widget then the rows are calculated as the largest `rows()` returned from all columns except the ones listed in `box_columns`. The box widgets in `box_columns` will be displayed with this calculated number of rows, filling the full height.

_set_text (*message*)
Set contents for this notification.

close ()
Close notification.

id
Notification ID.

is_alive
Return True if notification is currently visible.

update (*message*)
Update notification message.

class `clay.notifications._NotificationArea`
Notification area widget.

__init__ ()

Parameters

- **widget_list** (*iterable*) – child widgets
- **focus_item** (*Widget or int*) – child widget that gets the focus initially. Chooses the first selectable widget if unset.

widget_list may also contain tuples such as:

(**given_height**, **widget**) always treat *widget* as a box widget and give it *given_height* rows, where *given_height* is an int

(**'pack'**, **widget**) allow *widget* to calculate its own height by calling its `rows()` method, ie. treat it as a flow widget.

(**'weight'**, **weight**, **widget**) if the pile is treated as a box widget then treat *widget* as a box widget with a height based on its relative weight value, otherwise treat the same as (**'pack'**, *widget*).

Widgets not in a tuple are the same as (**'weight'**, 1, *widget*)

Note: If the Pile is treated as a box widget there must be at least one **'weight'** tuple in *widget_list*.

append_notification (*notification*)
Append an existing notification (that was probably closed).

close_all ()
Close all notifications.

close_newest ()
Close newest notification

notify (*message*)
Create new notification with message.

set_app (*app*)
Set app instance.

Required for proper screen redraws when new notifications are created asynchronously.

Hotkeys management. Requires “gi” package and “Gtk” & “Keybinder” modules.

class `clay.hotkeys._HotkeyManager`

Manages configs. Runs Gtk main loop in a thread.

`__init__()`

`x.__init__(...)` initializes x; see `help(type(x))` for signature

`__weakref__`

list of weak references to the object (if defined)

`_parse_hotkeys()`

Reads out the configuration file and parse them into a hotkeys for urwid.

`_parse_x_hotkeys()`

Reads out them configuration file and parses them into hotkeys readable by GTK.

static `_to_gtk_modifier(key)`

Translates the modifies to the way that GTK likes them.

fire_hook (*key, operation*)

Fire hook by name.

initialize ()

Unbind previous hotkeys, re-read config & bind new hotkeys.

keypress (*name, caller, super_, size, key*)

Process the pressed key by looking it up in the configuration file

`clay.hotkeys.report_error(exc)`

Print an error message to the debug screen

CHAPTER 15

eventhook.py

Events implemetation for signal handling.

```
class clay.eventhook.EventHook
    Event that can have handlers attached.

    __iadd__(handler)
        Add event handler.

    __init__()
        x.__init__(...) initializes x; see help(type(x)) for signature

    __isub__(handler)
        Remove event handler.

    __weakref__
        list of weak references to the object (if defined)

fire (*args, **kwargs)
    Execute all handlers.
```


CHAPTER 16

meta.py

Predefined values.

C

- `clay.eventhook`, 33
- `clay.gp`, 5
- `clay.hotkeys`, 31
- `clay.meta`, 35
- `clay.notifications`, 29
- `clay.pages.page`, 27
- `clay.settings`, 3

Symbols

_GP (class in clay.gp), 7
 _HotkeyManager (class in clay.hotkeys), 31
 _Notification (class in clay.notifications), 29
 _NotificationArea (class in clay.notifications), 30
 _Settings (class in clay.settings), 3
 _SettingsEditor (class in clay.settings), 4
 __eq__() (clay.gp.Track method), 6
 __iadd__() (clay.eventhook.EventHook method), 33
 __init__() (clay.eventhook.EventHook method), 33
 __init__() (clay.gp.Artist method), 5
 __init__() (clay.gp.LikedSongs method), 5
 __init__() (clay.gp.Playlist method), 5
 __init__() (clay.gp.SearchResults method), 6
 __init__() (clay.gp.Station method), 6
 __init__() (clay.gp.Track method), 6
 __init__() (clay.gp._GP method), 7
 __init__() (clay.hotkeys._HotkeyManager method), 31
 __init__() (clay.notifications._Notification method), 29
 __init__() (clay.notifications._NotificationArea method), 30
 __init__() (clay.settings._Settings method), 3
 __init__() (clay.settings._SettingsEditor method), 4
 __isub__() (clay.eventhook.EventHook method), 33
 __repr__() (clay.gp.Track method), 6
 __str__() (clay.gp.Track method), 7
 __weakref__ (clay.eventhook.EventHook attribute), 33
 __weakref__ (clay.gp.Artist attribute), 5
 __weakref__ (clay.gp.LikedSongs attribute), 5
 __weakref__ (clay.gp.Playlist attribute), 6
 __weakref__ (clay.gp.SearchResults attribute), 6
 __weakref__ (clay.gp.Station attribute), 6
 __weakref__ (clay.gp.Track attribute), 7
 __weakref__ (clay.gp._GP attribute), 7
 __weakref__ (clay.hotkeys._HotkeyManager attribute), 31
 __weakref__ (clay.pages.page.AbstractPage attribute), 27
 __weakref__ (clay.settings._Settings attribute), 3
 __weakref__ (clay.settings._SettingsEditor attribute), 4

_commit_edits() (clay.settings._Settings method), 3
 _ensure_directories() (clay.settings._Settings method), 3
 _load_cache() (clay.settings._Settings method), 3
 _load_config() (clay.settings._Settings method), 3
 _make_call_proxy() (clay.gp._GP method), 7
 _parse_hotkeys() (clay.hotkeys._HotkeyManager method), 31
 _parse_x_hotkeys() (clay.hotkeys._HotkeyManager method), 31
 _set_text() (clay.notifications._Notification method), 29
 _to_gtk_modifier() (clay.hotkeys._HotkeyManager static method), 31

A

AbstractPage (class in clay.pages.page), 27
 activate() (clay.pages.page.AbstractPage method), 27
 add_liked_song() (clay.gp.LikedSongs method), 5
 add_to_my_library() (clay.gp._GP method), 7
 add_to_my_library() (clay.gp.Track method), 7
 add_to_my_library_async() (clay.gp.Track method), 7
 append_notification() (clay.notifications._NotificationArea method), 30
 Artist (class in clay.gp), 5
 asynchronous() (in module clay.gp), 8

C

clay.eventhook (module), 33
 clay.gp (module), 5
 clay.hotkeys (module), 31
 clay.meta (module), 35
 clay.notifications (module), 29
 clay.pages.page (module), 27
 clay.settings (module), 3
 close() (clay.notifications._Notification method), 30
 close_all() (clay.notifications._NotificationArea method), 30
 close_newest() (clay.notifications._NotificationArea method), 30
 create_station() (clay.gp.Track method), 7

create_station_async() (clay.gp.Track method), 7

E

edit() (clay.settings._Settings method), 3

EventHook (class in clay.eventhook), 33

F

filename (clay.gp.Track attribute), 7

fire() (clay.eventhook.EventHook method), 33

fire_hook() (clay.hotkeys._HotkeyManager method), 31

from_data() (clay.gp.Artist class method), 5

from_data() (clay.gp.Playlist class method), 6

from_data() (clay.gp.SearchResults class method), 6

from_data() (clay.gp.Station class method), 6

from_data() (clay.gp.Track class method), 7

G

get() (clay.settings._Settings method), 3

get_all_tracks() (clay.gp._GP method), 7

get_all_tracks_async() (clay.gp._GP method), 8

get_all_user_playlist_contents() (clay.gp._GP method), 8

get_all_user_playlist_contents_async() (clay.gp._GP method), 8

get_all_user_station_contents() (clay.gp._GP method), 8

get_all_user_station_contents_async() (clay.gp._GP method), 8

get_artist_art_filename() (clay.gp.Track method), 7

get_artists() (clay.gp.SearchResults method), 6

get_authtoken() (clay.gp._GP method), 8

get_cached_file_path() (clay.settings._Settings method), 4

get_cached_tracks_map() (clay.gp._GP method), 8

get_default_config_section() (clay.settings._Settings method), 4

get_is_file_cached() (clay.settings._Settings method), 4

get_section() (clay.settings._Settings method), 4

get_stream_url() (clay.gp._GP method), 8

get_stream_url_async() (clay.gp._GP method), 8

get_track_by_id() (clay.gp._GP method), 8

get_tracks() (clay.gp.SearchResults method), 6

get_tracks() (clay.gp.Station method), 6

get_url() (clay.gp.Track method), 7

I

id (clay.gp.Artist attribute), 5

id (clay.gp.Playlist attribute), 6

id (clay.gp.Station attribute), 6

id (clay.gp.Track attribute), 7

id (clay.notifications._Notification attribute), 30

initialize() (clay.hotkeys._HotkeyManager method), 31

invalidate_caches() (clay.gp._GP method), 8

is_alive (clay.notifications._Notification attribute), 30

is_authenticated (clay.gp._GP attribute), 8

is_subscribed (clay.gp._GP attribute), 8

K

key (clay.pages.page.AbstractPage attribute), 27

keypress() (clay.hotkeys._HotkeyManager method), 31

L

LikedSongs (class in clay.gp), 5

load_tracks() (clay.gp.Station method), 6

load_tracks_async() (clay.gp.Station method), 6

login() (clay.gp._GP method), 8

login_async() (clay.gp._GP method), 8

N

name (clay.pages.page.AbstractPage attribute), 27

notify() (clay.notifications._NotificationArea method), 30

P

Playlist (class in clay.gp), 5

R

rate_song() (clay.gp.Track method), 7

remove_from_my_library() (clay.gp._GP method), 8

remove_from_my_library() (clay.gp.Track method), 7

remove_from_my_library_async() (clay.gp.Track method), 7

remove_liked_song() (clay.gp.LikedSongs method), 5

report_error() (in module clay.hotkeys), 31

S

save_file_to_cache() (clay.settings._Settings method), 4

search() (clay.gp._GP method), 8

search_async() (clay.gp._GP method), 8

SearchResults (class in clay.gp), 6

set_app() (clay.notifications._NotificationArea method), 30

Station (class in clay.gp), 6

synchronized() (in module clay.gp), 9

T

Track (class in clay.gp), 6

tracks (clay.gp.LikedSongs attribute), 5

U

update() (clay.notifications._Notification method), 30

use_authtoken() (clay.gp._GP method), 8

use_authtoken_async() (clay.gp._GP method), 8