# Weverse

MujyKun

## **CONTENTS:**

1	WeverseClient	1
2	Clients2.1WeverseClientSync2.2WeverseClientAsync	<b>5</b> 5
3	Models         3.1 Community          3.2 Notification          3.3 Photo          3.4 Artist          3.5 Comment          3.6 Post          3.7 Tab          3.8 Media	11 11 12 14 15 17 19 21 22
4	Model Creation	25
5	Exceptions 5.1 Invalid Token	27 27 27 27
6	Get Account Token	29
7	Asynchronous Usage	31
8	Synchronous Usage	33
9	Indices and tables	35
Рy	thon Module Index	37
In	dex	39

## WEVERSECLIENT

## class Weverse.WeverseClient(\*\*kwargs)

Abstract & Parent Client for connecting to Weverse and creating the internal cache.

Do not create an object directly from this class. Instead, create a Weverse. WeverseClientSync or Weverse. WeverseClientAsync object since those are concrete.

#### **Parameters**

- **verbose** (*bool*) Whether to print out verbose messages.
- web\_session An aiohttp or requests client session.
- **token** The account token to connect to the Weverse API. In order to find your token, please refer to *Get Account Token*

#### verbose

Whether to print out verbose messages.

Type bool

#### web\_session

An aiohttp or requests client session.

#### token

The account token to connect to the Weverse API. In order to find your token, please refer to *Get Account Token* 

#### user\_notifications

Most recent notifications of the account connected.

Type list

## headers

API Header

Type dict

#### api\_url

URL to connect to the API

Type str

## api\_communities\_url

Endpoint for communities

Type str

## api\_notifications\_url

Endpoint for user notifications

#### Type str

#### api\_new\_notifications\_url

Endpoint for checking new user notifications

Type str

#### api\_all\_artist\_posts\_url

Endpoint for getting the Artist Feed from a community

Type str

#### api\_artist\_to\_fans

part of the endpoint (NOT FULL) for viewing the tofans posts.

Type str

#### api\_all\_communities\_info\_url

Endpoint for information about ALL communities and ALL idols.

Type str

#### cache loaded

Whether the Internal Weverse Cache is fully loaded. This will change for a split moment when grabbing a new post.

Type bool

#### new\_media

We do not store ALL old media objects as cache, so only when there are new media, we store it

**Type** list(*Media*)

#### user\_endpoint

User info endpoint.

Type str

#### all\_posts

All posts in cache where the dict of Post ID is the key, and the value being the Post Object

**Type** dict(*Post*)

## all\_artists

All artists in cache where the dict of Artist ID is the key, and the value being the Artist Object

**Type** dict(*Artist*)

#### all\_comments

All comments in cache where the dict of Comment ID is the key, and the value being the Comment Object

**Type** dict(*Comment*)

#### all\_notifications

All notifications in cache where the dict of Notification ID is the key, and the value being the Notification Object

**Type** dict(*Notification*)

#### all\_photos

All photos in cache where the dict of Photo ID is the key, and the value being the Photo Object

Type dict(Photo)

#### all\_communities

All communities in cache where the dict of Community ID is the key, and the value being the Community Object

**Type** dict(*Community*)

#### all\_media

All media in cache where the dict of Media ID is the key, and the value being the Media Object

Type dict(Media)

#### all\_tabs

All tabs in cache where the dict of Tab ID is the key, and the value being the Tab Object

Type dict(Tab)

#### **check\_status**(*status*, *url*) → bool

Confirm the status of a URL

#### **Parameters**

- status Status code of url connection
- **url** Link that we connected to.

**Returns** True if the connection was a success.

Raises Invalid Token if there was an invalid token.

#### static determine\_notification\_type( $notification\_body$ ) $\rightarrow$ str

Determine the post type based on the notification body.

Since notifications do not differentiate between Posts and Comments, this is for that purpose.

**Parameters notification\_body** – The body of the notification.

**Returns** A string with either "comment", "media", or "post"

## $\texttt{get\_artist\_by\_id}(\textit{artist\_id}) \rightarrow \textit{Weverse.models.artist.Artist}$

Get artist by their ID.

Parameters artist\_id - The artist's ID

**Returns** *Artist* or NoneType

## $\texttt{get\_comment\_by\_id}(comment\_id) \rightarrow \textit{Weverse.models.comment.Comment}$

Get a comment by the ID :param comment\_id: Comment ID :returns: Comment or NoneType

## $\texttt{get\_community\_by\_id}(community\_id) \rightarrow \textit{Weverse.models.community}. \textit{Community}$

Get a community by the ID.

**Parameters community\_id** – Community ID

**Returns** Community or NoneType

## $\texttt{get\_media\_by\_id}(media\_id) \rightarrow Weverse.models.media.Media$

Get Media by the ID

Parameters media\_id - Media ID

**Returns** *Media* or NoneType

## $\texttt{get\_notification\_by\_id}(notification\_id) \rightarrow \textit{Weverse.models.notification.Notification}$

Get a notification by the ID

Parameters notification\_id - Notification ID

**Returns** *Notification* or NoneType

```
get\_photo\_by\_id(photo\_id) \rightarrow Weverse.models.photo.Photo
Get a photo by the ID
```

**Parameters photo\_id** – Photo ID

Returns Photo or NoneType

 $get_post_by_id(post_id) \rightarrow Weverse.models.post.Post$ Get a post by the ID

Parameters post\_id - Post ID

Returns Post or NoneType

 $\texttt{get\_tab\_by\_id}(tab\_id) \rightarrow \textit{Weverse.models.tab.Tab}$  Get tab by their ID.

Parameters tab\_id - The tab ID

**Returns** *Tab* or NoneType

static process\_community\_artists\_and\_tabs(community, response\_text\_as\_dict)

Process the community artists and tabs and add them to their respective communities.

#### **Parameters**

- community Community object
- response\_text\_as\_dict Response text of connection to endpoint but as a dict.

## **TWO**

## **CLIENTS**

## 2.1 WeverseClientSync

## class Weverse.WeverseClientSync(\*\*kwargs)

Synchronous Weverse Client that Inherits from Weverse Client.

**Parameters kwargs** – Same as *WeverseClient*.

Attributes are the same as WeverseClient.

#### check\_new\_user\_notifications()

Checks if there is a new user notification, updates the cache, and returns if there was.

**Returns** (bool) Whether there is a new notification.

## check\_token\_works()

Check if a token is invalid.

**Returns** (bool) True if the token works.

#### create\_communities()

Get and Create the communities the logged in user has access to.

#### create\_community\_artists\_and\_tabs()

Create the community artists and tabs and add them to their respective communities.

 $create\_post(community: Weverse.models.community.Community, post\_id) \rightarrow Weverse.models.post.Post$  Create a post and update the cache with it. This is meant for an individual post.

#### **Parameters**

- **community** *Community* the post was created under.
- **post\_id** The id of the post we are needing to fetch.

**create\_posts**(*community:* Weverse.models.community.Community, *next\_page\_id:* Optional[int] = None) Paginate through a community's posts and add it to object cache.

#### **Parameters**

- **community** *Community* the posts exist under.
- next\_page\_id ([OPTIONAL]) Next Page ID (Weverse paginates posts).

#### fetch\_artist\_comments(community\_id, post\_id)

Fetches the artist comments on a post.

#### **Parameters**

• **community\_id** – Community ID the post is on.

• **post\_id** – Post ID to fetch the artist comments of.

**Returns** List[Comment]

## fetch\_comment\_body(community\_id, comment\_id)

Fetches a comment from its ID.

#### **Parameters**

- **community\_id** The ID of the community the comment belongs to.
- comment\_id The ID of the comment to fetch.

**Returns** (str) Body of the comment.

#### fetch\_media(community\_id, media\_id)

Receive media object based on media id.

#### **Parameters**

- **community\_id** The ID of the community the media belongs to.
- **media\_id** The ID of the media to fetch.

Returns Media or NoneType

#### get\_user\_notifications()

Get a list of updated user notification objects.

**Returns** List[Notification]

#### start()

Creates internal cache.

This is the main process that should be run.

```
Raises Weverse.error.PageNotFound
Raises Weverse.error.InvalidToken
```

Raises Weverse.error.BeingRateLimited

**translate**(*post\_or\_comment\_id*, *is\_post=False*, *is\_comment=False*, *p\_obj=None*, *community\_id=None*)

Translates a post or comment, must set post or comment to True.

### **Parameters**

- post\_or\_comment\_id A post or comment ID.
- is\_post ([OPTIONAL]) If we passed in a post.
- is\_comment ([OPTIONAL]) If we passed in a comment
- **p\_obj** ([OPTIONAL]) The object we are looking to translate
- $community\_id([\mathit{OPTIONAL}])$  The community id the post/comment was made under.

Returns (str) Translated message or NoneType

#### update\_cache\_from\_notification()

Grab a new post based from a notification and add it to cache.

6 Chapter 2. Clients

## 2.2 WeverseClientAsync

Asynchronous Weverse Client that Inherits from Weverse Client.

#### **Parameters**

- **loop** Asyncio Event Loop
- **kwargs** Same as Weverse.

## loop

Asyncio Event Loop

Attributes are the same as :ref:`WeverseClient`.

#### async check\_new\_user\_notifications() $\rightarrow$ bool

Checks if there is a new user notification, updates the cache, and returns if there was.

This is a coroutine and must be awaited.

**Returns** (bool) Whether there is a new notification.

### async check\_token\_works() $\rightarrow$ bool

Check if a token is invalid.

This is a coroutine and must be awaited.

**Returns** (bool) True if the token works.

#### async create\_communities()

Get and Create the communities the logged in user has access to.

This is a coroutine and must be awaited.

#### async create\_community\_artists\_and\_tabs()

Create the community artists and tabs and add them to their respective communities.

This is a coroutine and must be awaited.

# **async create\_post**(community: Weverse.models.community.Community, $post\_id$ ) $\rightarrow$ Weverse.models.post.Post

Create a post and update the cache with it. This is meant for an individual post.

This is a coroutine and must be awaited.

## **Parameters**

- **community** *Community* the post was created under.
- **post\_id** The id of the post we are needing to fetch.

# 

Paginate through a community's posts and add it to object cache.

This is a coroutine and must be awaited.

#### **Parameters**

- **community** *Community* the posts exist under.
- next\_page\_id ([OPTIONAL]) Next Page ID (Weverse paginates posts).

#### async fetch\_artist\_comments(community\_id, post\_id)

Fetches the artist comments on a post.

This is a coroutine and must be awaited.

#### **Parameters**

- **community\_id** Community ID the post is on.
- **post\_id** Post ID to fetch the artist comments of.

**Returns** List[Comment]

## async fetch\_comment\_body(community\_id, comment\_id)

Fetches a comment from its ID.

This is a coroutine and must be awaited.

#### **Parameters**

- **community\_id** The ID of the community the comment belongs to.
- **comment\_id** The ID of the comment to fetch.

**Returns** (str) Body of the comment.

#### async fetch\_media(community\_id, media\_id)

Receive media object based on media id.

This is a coroutine and must be awaited.

#### **Parameters**

- **community\_id** The ID of the community the media belongs to.
- **media\_id** The ID of the media to fetch.

**Returns** *Media* or NoneType

#### async get\_user\_notifications()

Get a list of updated user notification objects.

This is a coroutine and must be awaited.

**Returns** List[*Notification*]

## async start()

Creates internal cache.

This is the main process that should be run.

This is a coroutine and must be awaited.

```
Raises Weverse.error.PageNotFound
Raises Weverse.error.InvalidToken
```

Raises Weverse.error.BeingRateLimited

**async** translate(post\_or\_comment\_id, is\_post=False, is\_comment=False, p\_obj=None, community\_id=None)

Translates a post or comment, must set post or comment to True.

This is a coroutine and must be awaited.

## **Parameters**

• post\_or\_comment\_id - A post or comment ID.

8 Chapter 2. Clients

- is\_post ([OPTIONAL]) If we passed in a post.
- **is\_comment** ([OPTIONAL]) If we passed in a comment
- **p\_obj** ([OPTIONAL]) The object we are looking to translate
- $community\_id([\textit{OPTIONAL}])$  The community id the post/comment was made under.

**Returns** (str) Translated message or NoneType

## async update\_cache\_from\_notification()

Grab a new post based from a notification and add it to cache.

This is a coroutine and must be awaited.

10 Chapter 2. Clients

## THREE

## **MODELS**

## 3.1 Community

## class Weverse.models.Community(\*\*kwargs)

A Comment object that represents a Weverse Comment that belongs to an Artist.

It is not suggested to create a Comment manually, but rather through the following method:  $Weverse.objects.create\_comment\_objects$ 

The information retrieved on a Comment is directly from the Weverse API and altered to fit this class.

#### **Parameters**

- id (int) The Community ID.
- name (str) The Community Name.
- **description** (str) Description of the Community.
- member\_count (int) Amount of members in the community.
- home\_banner (str) Direct Image URL to the home banner.
- icon (str) Direct Image URL to the Icon.
- Banner (str) Direct Image URL to the Banner.
- **full\_name** (str) Full Name of the Community.
- **fc\_member** (*bool*) If a special membership is required to join.
- **show\_member\_count** (*bool*) If the member count is visible.

id

The Community ID.

Type int

name

The Community Name.

Type str

### description

Description of the Community.

Type str

## member\_count

Amount of members in the community.

Type int

#### home\_banner

Direct Image URL to the home banner.

Type str

icon

Direct Image URL to the Icon.

Type str

#### Banner

Direct Image URL to the Banner.

Type str

#### full\_name

Full Name of the Community.

Type str

#### fc member

If a special membership is required to join.

Type bool

#### show\_member\_count

If the member count is visible.

Type bool

#### artists

List of artists the community has.

**Type** List[*Artist*]

#### tabs

The Tabs the community has.

**Type** List[*Tab*]

## 3.2 Notification

#### class Weverse.models.Notification(\*\*kwargs)

A Media object that represents a Weverse Media Post.

It is not suggested to create a Notification object manually, but rather through the following method: Weverse. objects.create\_notification\_objects

The information retrieved on a Notification is directly from the Weverse API and altered to fit this class.

#### **Parameters**

- id(int) The id of the notification.
- message(str) The message of the notification.
- **bold\_element** (*str*) The bolded element in the notification.
- **community\_id** (*int*) The community id associated with the notification.
- $community\_name(str)$  The community name associated with the notification.
- **contents\_type** (str) The type of post it is.

12 Chapter 3. Models

- **contents\_id** (*int*) The id of the content post.
- **notified\_at** The time the notification was triggered.
- **icon\_image\_url** (*str*) Icon image url of the notification.
- **thumbnail\_image\_url** (*str*) Thumbnail url of the notification.
- **artist\_id** (*int*) The ID of the Artist that released the content.
- **is\_membership\_content** (*bool*) If the content is exclusive to members.
- **is\_web\_only** (*bool*) Whether the notification is only available directly on the website.
- platform (str) The platform of the notification.

#### id

The id of the notification.

Type int

#### message

The message of the notification.

Type str

#### bold\_element

The bolded element in the notification.

Type str

#### community\_id

The community id associated with the notification.

Type int

## community\_name

The community name associated with the notification.

Type str

#### contents\_type

The type of post it is.

Type str

#### contents\_id

The id of the content post.

Type int

#### notified\_at

The time the notification was triggered.

## icon\_image\_url

Icon image url of the notification.

Type str

#### thumbnail\_image\_url

Thumbnail url of the notification.

Type str

## artist\_id

The ID of the Artist that released the content.

Type int

3.2. Notification 13

#### is\_membership\_content

If the content is exclusive to members.

Type bool

#### is\_web\_only

Whether the notification is only available directly on the website.

Type bool

#### platform

The platform of the notification.

Type str

## 3.3 Photo

## class Weverse.models.Photo(\*\*kwargs)

A Photo object that represents a Weverse Photo that belongs to media or a post.

It is not suggested to create a Photo manually, but rather through the following method: Weverse.objects. create\_photo\_objects

The information retrieved on a Photo is directly from the Weverse API and altered to fit this class.

#### **Parameters**

- **id** (*int*) The ID of the photo.
- **content\_index** (*int*) Index the photo is in from a bundle of photos.
- **thumbnail\_img\_url** (*str*) The thumbnail image link.
- **thumbnail\_img\_width** (*str*) The original image width.
- **thumbnail\_img\_height** (*str*) The thumbnail image height.
- original\_img\_url (str) The original image link.
- original\_img\_width (str) The original image width.
- original\_img\_height (str) The original image height.
- **file\_name** (*str*) File name of the photo.

id

The ID of the photo.

Type int

#### content\_index

Index the photo is in from a bundle of photos.

Type int

## thumbnail\_img\_url

The thumbnail image link.

Type str

#### thumbnail\_img\_width

The original image width.

Type str

14 Chapter 3. Models

#### thumbnail\_img\_height

The thumbnail image height.

Type str

#### original\_img\_url

The original image link.

Type str

#### original\_img\_width

The original image width.

Type str

#### original\_img\_height

The original image height.

Type str

## file\_name

File name of the photo.

Type str

#### post

The Post Object the photo belongs to.

Type Post

## 3.4 Artist

### class Weverse.models.Artist(\*\*kwargs)

An Artist object that represents a Weverse Artist that belongs in a community.

It is not suggested to create an Artist manually, but rather through the following method: Weverse.objects. create\_artist\_objects

The information retrieved on an Artist is directly from the Weverse API and altered to fit this class.

### **Parameters**

- id (int) The Artist ID.
- **community\_user\_id** (int) Artist's ID in the community.
- name (str) The Primary Artist Name.
- list\_name (list) A list of names for the Artist.
- **is\_online** (bool) Whether the Artist is currently online
- **profile\_nick\_name** (str) Artist nickname.
- **profile\_img\_path** (str) Image URL for the Artist's profile.
- **is\_birthday** (bool) Whether it is the Artist's birthday.
- **group\_name** (str) The group name the Artist is associated with.
- max\_comment\_count (int) The maximum amount of comments this Artist can post.
- community\_id (int) The ID of the community this Artist object was selected from.
- is\_enabled (bool) If the Artist account is enabled.

3.4. Artist 15

- has\_new\_to\_fans (bool) If the Artist has a new post for fans.
- has\_new\_private\_to\_fans (bool) If the Artist has a new private post for fans.
- to\_fan\_last\_id (int) The latest tofan post ID.
- to\_fan\_last\_created\_at When the artist's last to fan post was created.
- **to\_fan\_last\_expire\_in** When the artist's last to fan post expires.
- birthday\_img\_url (str) A direct image url to the artist's birthday image.
- **community** (Community) The community the Artist is in.
- **posts** (list) A list of posts the Artist has.

#### id

The Artist ID.

Type int

#### community\_user\_id

Artist's ID in the community.

Type int

#### name

The Primary Artist Name.

Type str

#### list\_name

A list of names for the Artist.

Type list

## is\_online

Whether the Artist is currently online

Type bool

## profile\_nick\_name

Artist nickname.

Type str

#### profile\_img\_path

Image URL for the Artist's profile.

Type str

#### is\_birthday

Whether it is the Artist's birthday.

Type bool

## group\_name

The group name the Artist is associated with.

Type str

#### max\_comment\_count

The maximum amount of comments this Artist can post.

Type int

## community\_id

The ID of the community this Artist object was selected from.

16 Chapter 3. Models

#### Type int

#### is\_enabled

If the Artist account is enabled.

Type bool

#### has\_new\_to\_fans

If the Artist has a new post for fans.

Type bool

#### has\_new\_private\_to\_fans

If the Artist has a new private post for fans.

Type bool

#### to\_fan\_last\_id

The latest tofan post ID.

Type int

#### to\_fan\_last\_created\_at

When the artist's last tofan post was created.

#### to\_fan\_last\_expire\_in

When the artist's last tofan post expires.

#### birthday\_img\_url

A direct image url to the artist's birthday image.

Type str

## community

The community the Artist is in.

**Type** Community

#### posts

A list of posts the Artist has.

Type list

## 3.5 Comment

### class Weverse.models.Comment(\*\*kwargs)

A Comment object that represents a Weverse Comment that belongs to an Artist.

It is not suggested to create a Comment manually, but rather through the following method: Weverse.objects. create\_comment\_objects

The information retrieved on a Comment is directly from the Weverse API and altered to fit this class.

#### **Parameters**

- id (int) The ID of the comment.
- **body** (str) The comment content AKA the body of the message.
- **comment\_count** (int) Amount of comments inside of this comment (replies).
- like\_count (int) Amount of likes on the comment.
- has\_my\_like (bool) Whether the client has liked the comment.

3.5. Comment 17

- is\_blind (bool) NOT SURE WHAT THIS IS
- **post\_id** (int) The Post ID that the comment was created under.
- **created\_at** The time the comment was created.
- updated\_at The time the comment was updated.

id

The ID of the comment.

Type int

body

The comment content AKA the body of the message.

Type str

## comment\_count

Amount of comments inside of this comment (replies).

Type int

#### like\_count

Amount of likes on the comment.

Type int

## has\_my\_like

Whether the client has liked the comment.

Type bool

## is\_blind

NOT SURE WHAT THIS IS

Type bool

#### post\_id

The Post ID that the comment was created under.

Type int

#### created\_at

The time the comment was created.

## updated\_at

The time the comment was updated.

## post

The Post Object the comment belongs to.

Type Post

18 Chapter 3. Models

## 3.6 Post

```
class Weverse.models.Post(**kwargs)
```

A Post object that represents a Weverse Post.

It is not suggested to create a Post manually, but rather through the following method: Weverse.objects.create\_post\_objects

The information retrieved on a Post is directly from the Weverse API and altered to fit this class.

#### **Parameters**

- **id** (*int*) The ID of the post.
- **community\_tab\_id** (*int*) The tab the post is under.
- **type** (*str*) The type of Post.
- **body** (*str*) Body Message on the Post.
- comment\_count (int) Current amount of comments on the Post
- like\_count (int) Current amount of likes on the Post
- max\_comment\_count (int) Maximum amount of comments that can be on the Post
- has\_my\_like (bool) If the client user has the post liked.
- has\_my\_bookmark (bool) If the client user has the post bookmarked.
- **created\_at** When the post was created
- updated\_at When the post was last modified.
- **is\_locked** (*boo1*) Whether the post is locked.
- **is\_blind** (*bool*) Whether the post is visible?? Unknown
- **is\_active** (*bool*) Whether the post is active.
- **is\_private** (*bool*) Whether the post is private.
- **photos** (List[*Photo*]) A list of photos under the post.
- **is\_hot\_trending\_post** (*bool*) If the post is trending.
- **is\_limit\_comment** (*bool*) If the comments are limited.
- artist\_comments (List[Comment]) The Artist comments under the post.
- **community\_artist\_id** (*int*) The Community Artist ID that made the post.
- artist\_id (int) The ID of the Artist that made the post.

id

The ID of the post.

Type int

## community\_tab\_id

The tab the post is under.

Type int

#### type

The type of Post.

Type str

3.6. Post 19

#### body

Body Message on the Post.

Type str

#### comment\_count

Current amount of comments on the Post

Type int

#### like\_count

Current amount of likes on the Post

Type int

#### max\_comment\_count

Maximum amount of comments that can be on the Post

Type int

## has\_my\_like

If the client user has the post liked.

Type bool

## has\_my\_bookmark

If the client user has the post bookmarked.

Type bool

#### created\_at

When the post was created

### updated\_at

When the post was last modified.

#### is\_locked

Whether the post is locked.

Type bool

## is\_blind

Whether the post is visible?? Unknown

Type bool

## is\_active

Whether the post is active.

Type bool

## is\_private

Whether the post is private.

Type bool

## photos

A list of photos under the post.

**Type** List[*Photo*]

## is\_hot\_trending\_post

If the post is trending.

Type bool

20 Chapter 3. Models

## is\_limit\_comment

If the comments are limited.

Type bool

#### artist\_comments

The Artist comments under the post.

**Type** List[Comment]

#### community\_artist\_id

The Community Artist ID that made the post.

Type int

#### artist\_id

The ID of the Artist that made the post.

Type int

## artist

The Artist Object the post belongs to.

Type Artist

## 3.7 Tab

class Weverse.models.Tab(tab\_id=None, name=None)

A Post object that represents a Weverse Post.

It is not suggested to create a Post manually, but rather through the following method: Weverse.objects.create\_post\_objects

The information retrieved on a Post is directly from the Weverse API and altered to fit this class.

#### **Parameters**

- tab\_id ([Optional] int) The ID of the Tab.
- name ([Optional] str) The Tab name.

id

The ID of the Tab.

Type int

name

The Tab name.

Type str

3.7. Tab 21

## 3.8 Media

#### class Weverse.models.Media(\*\*kwargs)

A Media object that represents a Weverse Media Post.

It is not suggested to create a Media object manually, but rather through the following method: Weverse. objects.create\_media\_object

The information retrieved on Media is directly from the Weverse API and altered to fit this class.

#### **Parameters**

- **id** (*int*) ID of the Media post.
- **community\_id** (*int*) ID of the Community the media post was made in.
- **body** (*str*) The media content AKA the body of the message.
- type(str) The type of media post it is.
- **thumbnail\_path** (*str*) The (url??) of the thumbnail.
- **title** (*str*) The title of the media post.
- **level** The level of access the media post is categorized under.
- video\_link (str) The video link supplied under the media post.
- **youtube\_id** (*str*) The youtube video ID.

id

ID of the Media post.

Type int

## community\_id

ID of the Community the media post was made in.

Type int

#### body

The media content AKA the body of the message.

Type str

#### type

The type of media post it is.

Type str

## thumbnail\_path

The (url??) of the thumbnail.

Type str

## title

The title of the media post.

Type str

#### level

The level of access the media post is categorized under.

#### video\_link

The video link supplied under the media post.

Type str

22 Chapter 3. Models

## youtube\_id

The youtube video ID.

Type str

3.8. Media 23

24 Chapter 3. Models

## **FOUR**

## MODEL CREATION

Weverse.objects.create\_artist\_objects( $current\_artists: list$ )  $\rightarrow$  list

Creates artist objects based on a list of information sent in and returns the objects.

**Parameters** current\_artists – Artist information received from endpoint.

**Returns** List[*Artist*]

 $\texttt{Weverse.objects.create\_comment\_objects}(\textit{current\_comments}: \textit{list}) \rightarrow \textit{list}$ 

Creates & Returns comment objects based on a list of comments

 $\label{lem:parameters} \textbf{Parameters} \ \textbf{-comment} \ \textbf{-comment} \ \textbf{information} \ \textbf{from} \ \textbf{endpoint}.$ 

**Returns** List[Comment]

Weverse.objects.create\_community\_objects( $current\_communities: list$ )  $\rightarrow$  dict

Creates community objects based on a list of information sent in and returns the objects.

Parameters current\_communities - Community information received from endpoint.

**Returns** dict{community id: *Community*}

Weverse.objects.create\_media\_object( $media\_info: dict$ )  $\rightarrow Weverse.models.media.Media$ Creates and returns a media object

**Parameters media\_info** – media information from endpoint.

Returns Media

Weverse.objects.create\_notification\_objects( $current\_notifications: list$ )  $\rightarrow$  list Creates notification objects based on a list of information sent in and returns the objects.

**Parameters current\_notifications** – Notification information received from endpoint.

**Returns** List[Notification]

Weverse.objects.create\_photo\_objects( $current\_photos: list$ )  $\rightarrow$  list

Creates & Returns photo objects based on a list of photos

**Parameters current\_photos** – photo information from endpoint.

**Returns** List[*Photo*]

Weverse.objects.create\_post\_objects(current\_posts: list, community:

Weverse.models.community.Community, new=False)  $\rightarrow$  list

Creates post objects based on a list of posts sent in and the community and returns the objects.

#### **Parameters**

- **current\_posts** Post information received from endpoint.
- **community** *Community* that the post belongs in.

• **new** – bool Whether or not the post is new.

**Returns** List[*Post*]

 $\texttt{Weverse.objects.create\_tab\_objects}(\textit{current\_tabs: list}) \rightarrow \textit{list}$ 

Creates tab objects based on a list of information sent in and returns the objects.

**Parameters** current\_tabs – Tab information received from endpoint.

**Returns** List[*Tab*]

**FIVE** 

## **EXCEPTIONS**

## 5.1 Invalid Token

## exception Weverse.InvalidToken

An Exception Raised When an Invalid Token was Supplied.

## 5.2 Page Not Found

## exception Weverse.PageNotFound(url)

An Exception Raised When a link was not found.

**Parameters url** (str) – The link that was not found.

## 5.3 Being Rate Limited

## exception Weverse.BeingRateLimited

An Exception Raised When Weverse Is Being Rate-Limited.

SIX

## **GET ACCOUNT TOKEN**

Your account token is needed (Will need to be updated about every 6 months iirc).

In order to get your account token, go to https://www.weverse.io/ and Inspect Element (F12).

Then go to the *Network* tab and filter by *XHR*.

Then refresh your page (F5) and look for *info* or *me* under *XHR*.

Under Headers, scroll to the bottom and view the request headers.

You want to copy everything past authorization: Bearer.

For example, you may see (This is just an example):

authorization: Bearer ABCDEFGHIJKLMNOPQRSTUVWXYZ

Then ABCDEFGHIJKLMNOPQRSTUVWXYZ would be your auth token for Weverse.

It is suggested to have the auth token as an environment variable.

## **SEVEN**

## **ASYNCHRONOUS USAGE**

## **EIGHT**

## SYNCHRONOUS USAGE

```
# Synchronous
import requests
from Weverse.weversesync import WeverseClientSync
from Weverse.error import InvalidToken

token = "fake_token" # REQUIRED
# It is advised to pass in your own web session as it is not closed in Weverse
web_session = requests.Session() # A session is created by default
weverse_client = WeverseClientSync(authorization=token, verbose=True)
try:
    weverse_client.start() # creates all the cache needed for your account.
except InvalidToken:
    print("Invalid Token")
```

## **NINE**

## **INDICES AND TABLES**

- genindex
- modindex
- search

## **PYTHON MODULE INDEX**

W

Weverse.objects, 25

38 Python Module Index

## **INDEX**

A	C
all_artists (Weverse.WeverseClient attribute), 2	cache_loaded (Weverse.WeverseClient attribute), 2
all_comments (Weverse.WeverseClient attribute), 2	<pre>check_new_user_notifications() (Wev-</pre>
all_communities (Weverse.WeverseClient attribute), 2	erse.WeverseClientAsync method), 7
all_media (Weverse. Weverse Client attribute), 3	check_new_user_notifications() (Wev-
all_notifications (Weverse.WeverseClient attribute),	erse.WeverseClientSync method), 5
2	<pre>check_status() (Weverse.WeverseClient method), 3</pre>
all_photos (Weverse.WeverseClient attribute), 2	check_token_works() (Weverse.WeverseClientAsync
all_posts (Weverse.WeverseClient attribute), 2	method), 7
all_tabs (Weverse.WeverseClient attribute), 3	check_token_works() (Weverse.WeverseClientSync
api_all_artist_posts_url (Weverse.WeverseClient	method), 5
attribute), 2	Comment (class in Weverse.models), 17
api_all_communities_info_url (Wev- erse.WeverseClient attribute), 2	comment_count (Weverse.models.Comment attribute), 18
api_artist_to_fans (Weverse.WeverseClient at-	comment_count (Weverse.models.Post attribute), 20
tribute), 2	Community (class in Weverse.models), 11
api_communities_url (Weverse.WeverseClient at-	community (Weverse.models.Artist attribute), 17
tribute), 1	community_artist_id(Weverse.models.Post attribute),
api_new_notifications_url(Weverse.WeverseClient	21
attribute), 2	community_id (Weverse.models.Artist attribute), 16
api_notifications_url (Weverse.WeverseClient at-	community_id (Weverse.models.Media attribute), 22
tribute), 1	community_id (Weverse.models.Notification attribute),
api_url (Weverse.WeverseClient attribute), 1	13
Artist (class in Weverse.models), 15	community_name (Weverse.models.Notification at-
artist (Weverse.models.Post attribute), 21	tribute), 13
artist_comments (Weverse.models.Post attribute), 21 artist_id (Weverse.models.Notification attribute), 13	community_tab_id (Weverse.models.Post attribute), 19 community_user_id (Weverse.models.Artist attribute),
artist_id(Weverse.models.Post attribute), 21	16
artists (Weverse.models.Community attribute), 12	content_index (Weverse.models.Photo attribute), 14
D	contents_id(Weverse.models.Notification attribute), 13
В	contents_type (Weverse.models.Notification attribute),
Banner (Weverse.models.Community attribute), 12	13
BeingRateLimited, 27	create_artist_objects() (in module Wev-
birthday_img_url (Weverse.models.Artist attribute),	erse.objects), 25
17	create_comment_objects() (in module Wev-
body (Weverse.models.Comment attribute), 18	erse.objects), 25
body (Weverse.models.Media attribute), 22	create_communities() (Weverse.WeverseClientAsync
body (Weverse.models.Post attribute), 19	<pre>method), 7 create_communities() (Weverse.WeverseClientSync</pre>
bold_element (Weverse.models.Notification attribute), 13	method), 5
	create_community_artists_and_tabs() (Wev-

<pre>create_community_artists_and_tabs() (Wev- erse.WeverseClientSync method), 5</pre>	<pre>get_notification_by_id() (Weverse.WeverseClient</pre>
<pre>create_community_objects() (in module Wev- erse.objects), 25</pre>	<pre>get_photo_by_id() (Weverse.WeverseClient method), 3</pre>
<pre>create_media_object() (in module Weverse.objects),</pre>	<pre>get_post_by_id() (Weverse.WeverseClient method), 4 get_tab_by_id() (Weverse.WeverseClient method), 4</pre>
<pre>create_notification_objects() (in module Wev- erse.objects), 25</pre>	<pre>get_user_notifications()</pre>
<pre>create_photo_objects() (in module Wev- erse.objects), 25</pre>	<pre>get_user_notifications()</pre>
<pre>create_post() (Weverse.WeverseClientAsync method),</pre>	group_name (Weverse.models.Artist attribute), 16
<pre>create_post() (Weverse.WeverseClientSync method), 5</pre>	Н
create_post_objects() (in module Weverse.objects), 25	has_my_bookmark (Weverse.models.Post attribute), 20 has_my_like (Weverse.models.Comment attribute), 18
<pre>create_posts()</pre>	has_my_like (Weverse.models.Post attribute), 20 has_new_private_to_fans (Weverse.models.Artist at-
<pre>create_posts() (Weverse.WeverseClientSync method),</pre>	tribute), 17
5 create_tab_objects() (in module Weverse.objects),	has_new_to_fans (Weverse.models.Artist attribute), 17 headers (Weverse.WeverseClient attribute), 1
26	home_banner (Weverse.models.Community attribute), 12
created_at (Weverse.models.Comment attribute), 18 created_at (Weverse.models.Post attribute), 20	1
D	icon (Weverse.models.Community attribute), 12
description (Weverse.models.Community attribute), 11	icon_image_url (Weverse.models.Notification at-
determine_notification_type() (Wev-	tribute), 13 id (Weverse.models.Artist attribute), 16
erse.WeverseClient static method), 3	id (Weverse.models.Comment attribute), 18
	id (Weverse.models.Community attribute), 11
F	id (Weverse.models.Media attribute), 22
fc_member (Weverse.models.Community attribute), 12	id (Weverse.models.Notification attribute), 13
<pre>fetch_artist_comments() (Wev-</pre>	id (Weverse.models.Photo attribute), 14
erse.WeverseClientAsync method), 7	id (Weverse.models.Post attribute), 19
<pre>fetch_artist_comments()</pre>	id (Weverse.models.Tab attribute), 21
erse. Weverse Client Sync method), 5	InvalidToken, 27
<pre>fetch_comment_body() (Weverse.WeverseClientAsync</pre>	is_active (Weverse.models.Post attribute), 20
method), 8	is_birthday (Weverse.models.Artist attribute), 16
fetch_comment_body() (Weverse.WeverseClientSync	is_blind (Weverse.models.Comment attribute), 18
method), 6	is_blind (Weverse.models.Post attribute), 20
<pre>fetch_media() (Weverse.WeverseClientAsync method),</pre>	<pre>is_enabled (Weverse.models.Artist attribute), 17 is_hot_trending_post (Weverse.models.Post at-</pre>
<pre>fetch_media() (Weverse.WeverseClientSync method), 6</pre>	tribute), 20
file_name (Weverse.models.Photo attribute), 15	<pre>is_limit_comment (Weverse.models.Post attribute), 20</pre>
full_name (Weverse.models.Community attribute), 12	is_locked (Weverse.models.Post attribute), 20
G	<pre>is_membership_content (Weverse.models.Notification</pre>
<pre>get_artist_by_id() (Weverse.WeverseClient method),</pre>	is_online (Weverse.models.Artist attribute), 16
3	is_private (Weverse.models.Post attribute), 20
<pre>get_comment_by_id()</pre>	is_web_only (Weverse.models.Notification attribute), 14
<pre>get_community_by_id() (Weverse.WeverseClient</pre>	L
method), 3	level (Weverse.models.Media attribute), 22
<pre>get_media_by_id() (Weverse.WeverseClient method),</pre>	like_count (Weverse.models.Comment attribute), 18

40 Index

<pre>list_name (Weverse.models.Artist attribute), 16</pre>	Т		
100p (Weverse. Weverse Client Async attribute), 7	Tab (class in Weverse.models), 21		
N //	tabs (Weverse.models.Community attribute), 12		
M	thumbnail_image_url (Weverse.models.Notification		
<pre>max_comment_count (Weverse.models.Artist attribute), 16</pre>	<pre>attribute), 13 thumbnail_img_height (Weverse.models.Photo at-</pre>		
<pre>max_comment_count (Weverse.models.Post attribute),</pre>	tribute), 14		
20	thumbnail_img_url (Weverse.models.Photo attribute).		
Media (class in Weverse.models), 22	14		
member_count (Weverse.models.Community attribute), 11	thumbnail_img_width (Weverse.models.Photo at- tribute), 14		
message (Weverse.models.Notification attribute), 13	thumbnail_path (Weverse.models.Media attribute), 22		
module	title (Weverse.models.Media attribute), 22		
Weverse.objects, 25	to_fan_last_created_at (Weverse.models.Artist at- tribute), 17		
N	to_fan_last_expire_in (Weverse.models.Artist		
name (Weverse.models.Artist attribute), 16	attribute), 17		
name (Weverse.models.Community attribute), 11	to_fan_last_id (Weverse.models.Artist attribute), 17		
name (Weverse.models.Tab attribute), 21	token (Weverse. WeverseClient attribute), 1		
new_media (Weverse. Weverse Client attribute), 2	translate() (Weverse. Weverse Client Async method), 8		
Notification (class in Weverse.models), 12	translate() (Weverse. Weverse Client Sync method), 6 type (Weverse. models. Media attribute), 22		
notified_at (Weverse.models.Notification attribute), 13	type (Weverse.models.Post attribute), 19		
0	cype (weverse.models.rosi diritolie), 19		
	U		
original_img_height (Weverse.models.Photo attribute), 15	<pre>update_cache_from_notification() (Wev-</pre>		
original_img_url (Weverse.models.Photo attribute),	erse.WeverseClientAsync method), 9		
15	update_cache_from_notification() (Wev-		
original_img_width (Weverse.models.Photo at-	erse.WeverseClientSync method), 6		
tribute), 15	updated_at (Weverse.models.Comment attribute), 18		
	updated_at (Weverse.models.Post attribute), 20		
P	<pre>user_endpoint (Weverse.WeverseClient attribute), 2</pre>		
PageNotFound, 27	user_notifications (Weverse.WeverseClient at-		
Photo (class in Weverse.models), 14	tribute), 1		
photos (Weverse.models.Post attribute), 20	V		
platform (Weverse.models.Notification attribute), 14	•		
Post (class in Weverse.models), 19	verbose (Weverse. Weverse Client attribute), 1		
post (Weverse.models.Comment attribute), 18	video_link (Weverse.models.Media attribute), 22		
post (Weverse.models.Photo attribute), 15	W		
post_id (Weverse.models.Comment attribute), 18			
posts (Weverse.models.Artist attribute), 17	web_session (Weverse. Weverse Client attribute), 1		
process_community_artists_and_tabs() (Wev- erse.WeverseClient static method), 4	Weverse.objects module, 25		
profile_img_path (Weverse.models.Artist attribute),	WeverseClient (class in Weverse), 1		
16	WeverseClientAsync (class in Weverse), 7		
<pre>profile_nick_name (Weverse.models.Artist attribute),</pre>	WeverseClientSync (class in Weverse), 5		
16			
0	Υ		
S	youtube_id (Weverse.models.Media attribute), 23		
${\tt show\_member\_count} \ \ (\textit{Weverse.models.Community} \ \ \textit{at-}$	-		
tribute), 12			
<pre>start() (Weverse.WeverseClientAsync method), 8</pre>			
<pre>start() (Weverse. WeverseClientSync method), 6</pre>			

Index 41