otaku-info-web

Release 0.1.0

Hermann Krumrey

Mar 25, 2020

CONTENTS

Contents:

CHAPTER

ONE

OTAKU_INFO_WEB

1.1 otaku_info_web package

1.1.1 Subpackages

otaku_info_web.background package

Submodules

otaku_info_web.background.anilist module

otaku_info_web.background.anilist.**fetch_anilist_data**()
Retrieves all entries on the anilists of all users that provided an anilist username :return: None

otaku_info_web.background.anilist.**update_media_entries**(*anilist_data: Dict[otaku_info_web.db.ServiceUsername.ServiceUs Dict[otaku_info_web.utils.enums.MediaType, List[otaku_info_web.utils.anilist.AnilistItem.AnilistIt* → Dict[Tuple[int, otaku_info_web.utils.enums.MediaType], otaku_info_web.db.MediaId.MediaId]
Updates the media entries and anilist IDs :param anilist_data: :return:

otaku_info_web.background.anilist.**update_media_id**(*new_data: otaku_info_web.utils.anilist.AnilistItem.AnilistItem, media_item: otaku_info_web.db.MediaItem.MediaItem, existing: Optional[otaku_info_web.db.MediaId.MediaId]*) → otaku_info_web.db.MediaId.MediaId
Updates/Creates a MediaId database entry based on anilist data :param new_data: The anilist data to use :param media_item: The media item associated with the ID :param existing: The existing database entry. If None, will be created :return: The updated/created MediaId object

otaku_info_web.background.anilist.**update_media_item**(*new_data: otaku_info_web.utils.anilist.AnilistItem.AnilistItem, existing: Optional[otaku_info_web.db.MediaItem.MediaItem]*) → otaku_info_web.db.MediaItem.MediaItem
Updates or creates MediaItem database entries based on anilist data :param new_data: The new anilist data

:param existing: The existing database entry. If None, will be created :return: The updated/created MediaItem object

otaku_info_web.background.anilist.**update_media_lists**(*anilist_data: Dict[otaku_info_web.db.ServiceUsername.ServiceUserr Dict[otaku_info_web.utils.enums.MediaType, List[otaku_info_web.utils.anilist.AnilistItem.AnilistItem*

Updates the database for anilist user lists. This includes custom anilist lists. :param anilist_data: The anilist data to enter into the database :return: None

otaku_info_web.background.anilist.**update_media_user_entries**(*anilist_data: Dict[otaku_info_web.db.ServiceUsername.Se Dict[otaku_info_web.utils.enums.MediaType, List[otaku_info_web.utils.anilist.AnilistItem.A media_ids: Dict[Tuple[int, otaku_info_web.utils.enums.MediaType], otaku_info_web.db.MediaId.MediaId]*)

Updates the individual users' current state for media items in thei ranilist account. :param anilist_data: The anilist data to enter into the database :param media_ids: The anilist media IDs of the previously added media items :return: None

otaku_info_web.background.anilist.**update_media_user_state**(*new_data: otaku_info_web.utils.anilist.AnilistItem.AnilistIte media_id: otaku_info_web.db.MediaId.MediaId, user: puffotter.flask.db.User.User, existing: Optional[otaku_info_web.db.MediaUserState.Medi* → otaku_info_web.db.MediaUserState.MediaUserS

Updates or creates a MediaUserState entry in the database :param new_data: The new anilist data :param media_id: The media ID of the anilist media item :param user: The user associated with the data :param existing: The existing database entry. If None, will be created :return: The updated/created MediaUserState object

otaku_info_web.background.manga_chapters module

otaku_info_web.background.manga_chapters.**update_manga_chapter_guesses**()

Updates the manga chapter guesses :return: None

Module contents

otaku_info_web.background.**bg_tasks: Dict[str, Tuple[int, Callable]] = {'anilist_update':**
A dictionary containing background tasks for the flask application

otaku_info_web.db package

Submodules

otaku_info_web.db.MangaChapterGuess module

class otaku_info_web.db.MangaChapterGuess.**MangaChapterGuess**(**args*, ***kwargs*)
Bases: puffotter.flask.db.ModelMixin.ModelMixin, sqlalchemy.ext.declarative.api.Model

Database model that keeps track of manga chapter guesses.

__init__(**args*, ***kwargs*)
Initializes the Model :param args: The constructor arguments :param kwargs: The constructor keyword arguments

guess
The actual guess for the most current chapter of the manga series

id

last_update
Timestamp from when the guess was last updated

media_id
The media ID referenced by this manga chapter guess

media_id_id
The ID of the media ID referenced by this manga chapter guess

update()
Updates the manga chapter guess (if the latest guess is older than an hour) :return: None

otaku_info_web.db.MediaId module

class otaku_info_web.db.MediaId.**MediaId**(**args*, ***kwargs*)
Bases: puffotter.flask.db.ModelMixin.ModelMixin, sqlalchemy.ext.declarative.api.Model

Database model for media IDs. These are used to map media items to their corresponding external IDS on external sites.

__init__(**args*, ***kwargs*)
Initializes the Model :param args: The constructor arguments :param kwargs: The constructor keyword arguments

id

manga_chapter_guesses

media_item
The media item referenced by this ID

media_item_id
The ID of the media item referenced by this ID

media_user_states

service
The service for which this object represents an ID

service_id
The ID of the media item on the external service

property service_url

Returns The URL to the series for the given service

otaku_info_web.db.MediaItem module

class otaku_info_web.db.MediaItem.**MediaItem**(**args*, ***kwargs*)
Bases: puffotter.flask.db.ModelMixin.ModelMixin, sqlalchemy.ext.declarative.api.Model

Database model for media items. These model a generic, site-agnostic representation of a series.

__init__(**args*, ***kwargs*)
Initializes the Model :param args: The constructor arguments :param kwargs: The constructor keyword arguments

cover_url
An URL to a cover image of the media item

english_title
The English title of the media item

id

latest_release
The latest release chapter/episode for this media item

media_ids

media_subtype
The subtype (for example, TV short, movie oneshot etc)

media_type
The media type of the list item

releasing_state
The current releasing state of the media item

romaji_title
The Japanese title of the media item written in Romaji

property title

Returns The default title for the media item.

otaku_info_web.db.MediaList module

class otaku_info_web.db.MediaList.**MediaList**(**args*, ***kwargs*)

Bases: puffotter.flask.db.ModelMixin.ModelMixin, sqlalchemy.ext.declarative.api.Model

Database model for user-specific media lists.

__init__(**args*, ***kwargs*)

Initializes the Model :param args: The constructor arguments :param kwargs: The constructor keyword arguments

id

media_list_items

media_type

The media type for this list

name

The name of this list

service

The service for which this list applies to

user

The user associated with this list

user_id

The ID of the user associated with this list

otaku_info_web.db.MediaListItem module

class otaku_info_web.db.MediaListItem.**MediaListItem**(**args*, ***kwargs*)

Bases: puffotter.flask.db.ModelMixin.ModelMixin, sqlalchemy.ext.declarative.api.Model

Database model for media list items. This model maps MediaLists and MediaUserStates

__init__(**args*, ***kwargs*)

Initializes the Model :param args: The constructor arguments :param kwargs: The constructor keyword arguments

id

media_list

The media list this list item is a part of

media_list_id

The ID of the media list this list item is a part of

media_user_state

The media user state this list item references

media_user_state_id

The ID of the media user state this list item references

otaku_info_web.db.MediaUserState module

class otaku_info_web.db.MediaUserState.**MediaUserState**(**args*, ***kwargs*)
Bases: puffotter.flask.db.ModelMixin.ModelMixin, sqlalchemy.ext.declarative.api.Model

Database model that keeps track of a user's entries on external services for a media item

__init__(**args*, ***kwargs*)
Initializes the Model :param args: The constructor arguments :param kwargs: The constructor keyword arguments

consuming_state
The current consuming state of the user for this media item

id

media_id
The media ID referenced by this user state

media_id_id
The ID of the media ID referenced by this user state

media_list_items

progress
The user's current progress consuming the media item

score
The user's score for the references media item

user
The user associated with this user state

user_id
The ID of the user associated with this user state

otaku_info_web.db.ServiceUsername module

class otaku_info_web.db.ServiceUsername.**ServiceUsername**(**args*, ***kwargs*)
Bases: puffotter.flask.db.ModelMixin.ModelMixin, sqlalchemy.ext.declarative.api.Model

Database model that stores an external service username for a user

__init__(**args*, ***kwargs*)
Initializes the Model :param args: The constructor arguments :param kwargs: The constructor keyword arguments

id

service
The external service this item is a username for

user
The user associated with this service username

user_id
The ID of the user associated with this service username

username
The service username

Module contents

otaku_info_web.db.**models**: **List[sqlalchemy.ext.declarative.api.Model] = [<class 'otaku_info**
The database models of the application

otaku_info_web.routes package

Submodules

otaku_info_web.routes.external_service module

otaku_info_web.routes.external_service.**define_blueprint**(*blueprint_name: str*) → flask.blueprints.Blueprint
Defines the blueprint for this route :param blueprint_name: The name of the blueprint :return: The blueprint

otaku_info_web.routes.manga module

otaku_info_web.routes.manga.**define_blueprint**(*blueprint_name: str*) → flask.blueprints.Blueprint
Defines the blueprint for this route :param blueprint_name: The name of the blueprint :return: The blueprint

Module contents

otaku_info_web.routes.**blueprint_generators**: **List[Tuple[Callable[str, flask.blueprints.Blue**
Defines the functions used to create the various blueprints as well as their names

otaku_info_web.test package

Subpackages

otaku_info_web.test.db package

Submodules

otaku_info_web.test.db.TestApiKey module

class otaku_info_web.test.db.TestApiKey.**TestApiKey**(*methodName='runTest'*)
Bases: otaku_info_web.test.TestFramework._TestFramework

Class that tests the ApiKey database model

test_equality()
Tests checking equality for model objects :return: None

test_expiration()
Tests if the expiration of API keys works correctly :return: None

test_hashing()
Tests using the model objects as keys in a dictionary :return: None

test_json_representation()
Tests the JSON representation of the model :return: None

test_repr()
Tests the __repr__ method of the model class :return: None

test_string_representation()
Tests the string representation of the model :return: None

test_verifying_key()
Tests verifying an api key :return: None

otaku_info_web.test.db.TestUser module

class otaku_info_web.test.db.TestUser.**TestUser**(*methodName='runTest'*)
Bases: otaku_info_web.test.TestFramework._TestFramework

Class that tests the User database model

test_equality()
Tests checking equality for model objects :return: None

test_flask_properties()
Tests if the flask_login properties work as expected :return: None

test_hashing()
Tests using the model objects as keys in a dictionary :return: None

test_json_representation()
Tests the JSON representation of the model :return: None

test_repr()
Tests the __repr__ method of the model class :return: None

test_string_representation()
Tests the string representation of the model :return: None

test_verifying_password()
Tests verifying the password of a user :return: None

Module contents

otaku_info_web.test.misc package

Submodules

otaku_info_web.test.misc.TestApiCalls module

class otaku_info_web.test.misc.TestApiCalls.**TestConfig**(*methodName='runTest'*)
Bases: otaku_info_web.test.TestFramework._TestFramework

Class that tests varios API calls

test_expired_api_key()
Tests using an expired API key :return: None

test_non_base64_header()
Tests using a header that's not base64 encoded :return: None

test_random_exception()
Tests that the API routes catch any Exceptions without issue :return: None

test_unauthorized_call()
Tests and unauthorized API call :return: None

test_using_non_json_data()
Tests sending the data as something that's not JSON :return: None

otaku_info_web.test.misc.TestConfig module

class otaku_info_web.test.misc.TestConfig.**TestConfig**(*methodName='runTest'*)
Bases: otaku_info_web.test.TestFramework._TestFramework

Class that tests the config class

test_db_config()
Tests the database configuration :return: None

test_version()
Tests if the version is fetched correctly :return: None

otaku_info_web.test.misc.TestErrorHandling module

class otaku_info_web.test.misc.TestErrorHandling.**TestErrorHandling**(*methodName='runTest'*)
Bases: otaku_info_web.test.TestFramework._TestFramework

Class that tests the flask error handling

test_404()
Tests if a 404 error is handled correctly :return: None

test_exception()
Tests if unexpected exceptions are caught correctly :return: None

otaku_info_web.test.misc.TestServer module

class otaku_info_web.test.misc.TestServer.**TestServer**(*methodName='runTest'*)
Bases: otaku_info_web.test.TestFramework._TestFramework

Class that tests starting the server

test_starting_server()
Tests starting the server :return: None

Module contents

otaku_info_web.test.routes package

Subpackages

otaku_info_web.test.routes.api package

Submodules

otaku_info_web.test.routes.api.TestApiKeyRoute module

class otaku_info_web.test.routes.api.TestApiKeyRoute.**TestApiKeyRoute**(*methodName='runTest'*)
Bases: otaku_info_web.test.TestFramework._TestFramework

Class that tests API-key related features

test_requesting_api_key()
Tests requesting an API key :return: None

test_requesting_invalid_api_keys()
Tests requesting API keys with invalid data :return: None

test_revoking_api_key()
Tests revoking an API key :return: None

test_unsuccessfully_revoking_api_key()
Tests unsuccessfully revoking an API key :return: None

Module contents

Submodules

otaku_info_web.test.routes.TestForgotRoute module

class otaku_info_web.test.routes.TestForgotRoute.**TestForgotRoute**(*methodName='runTest'*)
Bases: otaku_info_web.test.TestFramework._TestFramework

Class that tests password reset features

test_invalid_recaptcha()
Tests that invalid ReCaptcha responses are handled correctly :return: None

test_page_get()
Tests getting the page :return: None

test_resetting_password()
Tests successfully resetting a password :return: None

test_unsuccessfully_resetting_password()
Tests unsuccessfully resetting a password :return: None

otaku_info_web.test.routes.TestLoginRoute module

class otaku_info_web.test.routes.TestLoginRoute.**TestLoginRoute**(*methodName='runTest'*)
Bases: otaku_info_web.test.TestFramework._TestFramework

Class that tests log-in features

test_invalid_login_attempts()
Tests trying to log in with invalid credentials etc :return: None

test_logging_in_and_out()
Tests logging in successfully, then once more, then logging out :return: None

test_page_get()
Tests getting the page :return: None

otaku_info_web.test.routes.TestProfileRoute module

class otaku_info_web.test.routes.TestProfileRoute.**TestProfileRoute**(*methodName='runTest'*)
Bases: otaku_info_web.test.TestFramework._TestFramework

Class that tests profile features

test_changing_password()
Tests changing a password :return: None

test_page_get()
Tests getting the page :return: None

test_unsuccessful_password_change()
Tests unsuccessfully changing a password :return: None

test_unsuccessful_user_delete()
Tests unsuccessfully deleting a user :return: None

test_user_delete()
Tests deleting a user :return: None

otaku_info_web.test.routes.TestRegisterRoute module

class otaku_info_web.test.routes.TestRegisterRoute.**TestRegisterRoute**(*methodName='runTest'*)
Bases: otaku_info_web.test.TestFramework._TestFramework

Class that tests registration features

test_confirming()
Tests confirming a user :return: None

test_invalid_confirm()
Tests invalid confirmations :return: None

test_invalid_recaptcha()
Tests that invalid ReCaptcha responses are handled correctly :return: None

test_invalid_registrations()
Tests registering using invalid parameters :return: None

test_page_get()
Tests getting the page :return: None

test_registering_user()
Tests registering a new user :return: None

otaku_info_web.test.routes.TestStaticRoutes module

class otaku_info_web.test.routes.TestStaticRoutes.**TestStaticRoutes**(*methodName='runTest'*)
Bases: otaku_info_web.test.TestFramework._TestFramework

Class that tests static pages

test_get_about()
Tests getting the about page :return: None

test_get_index()
Tests getting the index page :return: None

test_get_privacy()
Tests getting the privacy page :return: None

Module contents

Submodules

otaku_info_web.test.TestFramework module

Module contents

otaku_info_web.utils package

Subpackages

otaku_info_web.utils.anilist package

Submodules

otaku_info_web.utils.anilist.AnilistItem module

class otaku_info_web.utils.anilist.AnilistItem.**AnilistItem**(*anilist_id: int, media_type: otaku_info_web.utils.enums.MediaType, media_subtype: otaku_info_web.utils.enums.MediaSubType, english_title: Optional[str], romaji_title: str, cover_url: str, chapters: Optional[int], episodes: Optional[int], releasing_state: otaku_info_web.utils.enums.ReleasingState, score: Optional[int], progress: Optional[int], consuming_state: otaku_info_web.utils.enums.ConsumingState, list_name: str*)

Bases: object

Class that models an anilist list item for a user Represents the information fetched using anilist's API

__init__(*anilist_id: int, media_type: otaku_info_web.utils.enums.MediaType, media_subtype: otaku_info_web.utils.enums.MediaSubType, english_title: Optional[str], romaji_title: str, cover_url: str, chapters: Optional[int], episodes: Optional[int], releasing_state: otaku_info_web.utils.enums.ReleasingState, score: Optional[int], progress: Optional[int], consuming_state: otaku_info_web.utils.enums.ConsumingState, list_name: str*)

Initializes the AnilistItem object :param anilist_id: The anilist ID of the series :param media_type: The media type of the series :param media_subtype: The media subtype of the series :param english_title: The English title of the series :param romaji_title: The Japanes title of the series written in romaji :param cover_url: URL to a cover image for the series :param chapters: The total amount of known manga chapters :param episodes: The total amount of known anime episodes :param releasing_state: The current releasing state of the series :param score: The user's score for the series :param progress: The user's progress for the series :param consuming_state: The user's consumption state for the series :param list_name: Which of the user's lists this entry belongs to

property latest_release

Returns The latest release. Chapters for manga, episodes for anime

otaku_info_web.utils.anilist.api module

otaku_info_web.utils.anilist.api.**guess_latest_manga_chapter**(*anilist_id: int*) → Optional[int]

Guesses the latest chapter number based on anilist user activity :param anilist_id: The anilist ID to check :return: The latest chapter number

otaku_info_web.utils.anilist.api.**load_anilist**(*username: str, media_type: otaku_info_web.utils.enums.MediaType*) → List[otaku_info_web.utils.anilist.AnilistItem.AnilistItem]

Loads the anilist for a user :param username: The username :param media_type: The media type, either MANGA or ANIME :return: The anilist list items for the user and media type

Module contents

otaku_info_web.utils.manga_updates package

Submodules

otaku_info_web.utils.manga_updates.MangaUpdate module

class otaku_info_web.utils.manga_updates.MangaUpdate.**MangaUpdate**(*list_item: otaku_info_web.db.MediaListItem.Me*
chapter_guess: Optional[otaku_info_web.db.MangaChap

Bases: object

Class that encapsulates important data to display for manga updates

__init__(*list_item: otaku_info_web.db.MediaListItem.MediaListItem, chapter_guess: Optional[otaku_info_web.db.MangaChapterGuess.MangaChapterGuess]*)

Initializes the MangaUpdate object :param list_item: The list item to display :param chapter_guess: The corresponding chapter guess

otaku_info_web.utils.manga_updates.generator module

otaku_info_web.utils.manga_updates.generator.**prepare_manga_updates**(*user: puffotter.flask.db.User.User*, *service: str*, *media_list: str*, *include_complete: bool*, *only_updates: bool*) → List[otaku_info_web.utils.manga_u

Prepares easily understandable objects to display for manga updates :param user: The user requesting the manga updates :param service: The service for which to fetch the updates :param media_list: The media list for which to fetch the updates :param include_complete: Whether or not to include completed series :param only_updates: Whether or not to only retrieve entries with at least

one new chapter

Returns A list of MangaUpdate objects, sorted by score

Module contents

Submodules

otaku_info_web.utils.enums module

class otaku_info_web.utils.enums.**ConsumingState**
Bases: enum.Enum

Class that defines the possible consuming states for a user and media item

COMPLETED = 'completed'

CURRENT = 'current'

DROPPED = 'dropped'

PAUSED = 'paused'

PLANNING = 'planning'

REPEATING = 'repeating'

class otaku_info_web.utils.enums.**ListService**
Bases: enum.Enum

Class that defines available list services

ANILIST = 'anilist'

MYANIMELIST = 'myanimelist'

class otaku_info_web.utils.enums.**MediaSubType**
Bases: enum.Enum

Class that models a media subtype for media items

MANGA = 'manga'

MOVIE = 'movie'

MUSIC = 'music'

NOVEL = 'novel'

ONA = 'ona'

ONE_SHOT = 'one_shot'

OVA = 'ova'

SPECIAL = 'special'

TV = 'tv'

TV_SHORT = 'tv_short'

UNKNOWN = 'unknown'

class otaku_info_web.utils.enums.**MediaType**
Bases: enum.Enum

Class that models a media type for media items

ANIME = 'anime'

MANGA = 'manga'

class otaku_info_web.utils.enums.**ReleasingState**
Bases: enum.Enum

Class that defines possible releasing states

CANCELLED = 'cancelled'

FINISHED = 'finished'

NOT_YET_RELEASED = 'not_yet_released'

RELEASING = 'releasing'

UNKNOWN = 'unknown'

Module contents

1.1.2 Submodules

1.1.3 otaku_info_web.Config module

class otaku_info_web.Config.**Config**
Bases: puffotter.flask.Config.Config

Configuration for the flask application

DB_MODE = None

DB_URI = None

FLASK_SECRET = None

LOGGING_PATH = None

RECAPTCHA_SECRET_KEY = None

RECAPTCHA_SITE_KEY = None

SENTRY_DSN = None

SMTP_ADDRESS = None

SMTP_HOST = None

SMTP_PASSWORD = None

SMTP_PORT = None

TESTING = None

VERSION = None

1.1.4 otaku_info_web.main module

otaku_info_web.main.**main**()
Starts the flask application :return: None

1.1.5 otaku_info_web.template_extras module

otaku_info_web.template_extras.**profile_extras**() → Dict[str, Any]
Makes sure that the profile page displays service usernames :return: The variables to forward to the template

1.1.6 Module contents

otaku_info_web.**root_path: str = '/usr/local/lib/python3.6/dist-packages/otaku_info_web-0.1**
The root path of the application

otaku_info_web.**sentry_dsn = 'https://f899b0c46d324f37b83527a3994afd8d@sentry.namibsun.net/1**
The sentry DSN used for error logging

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

O

INDEX

Symbols

A

B

C

D

E

F

G

I

L

M

N

O

P

R

S

T

U

V