

---

# **otaku-info**

*Release 0.3.2*

**Hermann Krumrey**

**Aug 26, 2020**



# CONTENTS

<b>1</b>	<b>otaku_info</b>	<b>3</b>
1.1	otaku_info package . . . . .	3
<b>2</b>	<b>Indices and tables</b>	<b>37</b>
	<b>Python Module Index</b>	<b>39</b>
	<b>Index</b>	<b>41</b>



Contents:



## OTAKU\_INFO

### 1.1 otaku\_info package

#### 1.1.1 Subpackages

##### otaku\_info.background package

###### Submodules

##### otaku\_info.background.anilist module

otaku\_info.background.anilist.**update\_anilist\_data** (*usernames:* *Optional[List[otaku\_info.db.ServiceUsername.ServiceUsername] = None*)

Retrieves all entries on the anilists of all users that provided an anilist username :param usernames: Can be used to override the usernames to use :return: None

##### otaku\_info.background.ln\_releases module

otaku\_info.background.ln\_releases.**create\_media\_item\_from\_reddit\_ln\_release** (*ln\_release:* *otaku\_info.external.entities.LightNovelRelease = Optional[otaku\_info.db.M*)

Creates a media item based on a reddit ln release Fills any missing entries in the database :param ln\_release: The ln release :return: The media item

otaku\_info.background.ln\_releases.**update\_ln\_releases** ()

Updates the light novel releases :return: None

### otaku\_info.background.manga\_chapter\_guesses module

otaku\_info.background.manga\_chapter\_guesses.update\_manga\_chapter\_guesses ()  
Updates the manga chapter guesses :return: None

### otaku\_info.background.mangadex module

otaku\_info.background.mangadex.update\_mangadex\_data (start: int = 1, end: Optional[int]  
= None, refresh: bool = False)  
Goes through mangadex IDs sequentially and stores ID mappings for these entries if found. Stops once 100  
consecutive entries didn't return any results :param start: Optionally specifies a starting index :param end:  
Optionally specifies an ending index :param refresh: If true, will update existing mangadex info :return: None

### otaku\_info.background.notifications module

otaku\_info.background.notifications.handle\_notification (media\_user\_state:  
otaku\_info.db.MediaUserState.MediaUserState,  
settings:  
otaku\_info.db.NotificationSetting.NotificationSetting)  
Handles a single notification :param media\_user\_state: The user state for which to notify :param settings: The  
notification settings :return: None

otaku\_info.background.notifications.send\_new\_update\_notifications ()  
Sends out telegram notifications for media updates :return: None

### Module contents

otaku\_info.background.bg\_tasks: Dict[str, Tuple[int, Callable]] = {'anilist\_update': (300, ...)  
A dictionary containing background tasks for the flask application

### otaku\_info.db package

#### Submodules

#### otaku\_info.db.LnRelease module

**class** otaku\_info.db.LnRelease.LnRelease (\*args, \*\*kwargs)  
Bases: otaku\_info.db.ModelMixin.ModelMixin, sqlalchemy.ext.declarative.api.  
Model  
Database model that keeps track of light novel releases  
**\_\_init\_\_** (\*args, \*\*kwargs)  
Initializes the Model :param args: The constructor arguments :param kwargs: The constructor keyword  
arguments  
**digital**  
Whether this is a digital release  
**get\_ids** () → List[otaku\_info.db.MediaId.MediaId]  
Returns Any related Media IDs



**id****property identifier\_tuple****Returns** A tuple that uniquely identifies this database entry**media\_item**

The media item referenced by this release

**media\_item\_id**

The ID of the media item referenced by this release

**physical**

Whether this is a physical release

**publisher**

The publisher

**purchase\_link**

Link to a store page

**property release\_date****Returns** The release date as a datetime object**release\_date\_string**

The release date as a ISO-8601 string

**series\_name**

The series name

**update** (*new\_data*: `otaku_info.db.LnRelease.LnRelease`)

Updates the data in this record based on another object :param new\_data: The object from which to use the new values :return: None

**volume**

The volume identifier

**property volume\_number****Returns** The volume number as an integer

## otaku\_info.db.MangaChapterGuess module

**class** `otaku_info.db.MangaChapterGuess.MangaChapterGuess` (*\*args*, *\*\*kwargs*)**Bases:** `otaku_info.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****property identifier\_tuple****Returns** A tuple that uniquely identifies this database entry

**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** (*new\_data*: `otaku_info.db.MangaChapterGuess.MangaChapterGuess`)

Updates the data in this record based on another object :param new\_data: The object from which to use the new values :return: None

**update\_guess** ()

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

**otaku\_info.db.MediaId module**

**class** `otaku_info.db.MediaId.MediaId` (\*args, \*\*kwargs)

Bases: `otaku_info.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

**chapter\_guess**

Chapter Guess for this media ID (Only applicable if this is a manga title)

**id**

**property identifier\_tuple**

**Returns** A tuple that uniquely identifies this database entry

**media\_item**

The media item referenced by this ID

**media\_item\_id**

The ID of the media item referenced by this ID

**media\_subtype**

The media subtype of the list item

**media\_type**

The media type of the list item

**media\_user\_states**

Media user states associated with this media ID

**service**

The service for which this object represents an ID

**property service\_icon**

**Returns** The path to the service's icon file

**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

**update** (*new\_data*: otaku\_info.db.MediaId.MediaId)

Updates the data in this record based on another object :param new\_data: The object from which to use the new values :return: None

**otaku\_info.db.MediaItem module**

**class** otaku\_info.db.MediaItem.**MediaItem** (\*args, \*\*kwargs)

Bases: `otaku_info.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****property identifier\_tuple**

**Returns** A tuple that uniquely identifies this database entry

**latest\_release**

The latest release chapter/episode for this media item

**latest\_volume\_release**

The latest volume for this media item

**ln\_releases**

Light novel releases associated with this Media item

**property media\_id\_mapping**

**Returns** A dictionary mapping list services to IDs for this media item

**media\_ids**

Media IDs associated with this Media item

**media\_subtype**

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

**media\_type**

The media type of the list item

**property own\_url**

**Returns** The URL to the item's page on the otaku-info site

**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.

**update** (*new\_data*: `otaku_info.db.MediaItem.MediaItem`)

Updates the data in this record based on another object :param new\_data: The object from which to use the new values :return: None

### otaku\_info.db.MediaList module

**class** `otaku_info.db.MediaList.MediaList` (\*args, \*\*kwargs)

Bases: `otaku_info.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**

**property identifier\_tuple**

**Returns** A tuple that uniquely identifies this database entry

**media\_list\_items**

Media List Items that are a part of this media list

**media\_type**

The media type for this list

**name**

The name of this list

**service**

The service for which this list applies to

**update** (*new\_data*: `otaku_info.db.MediaList.MediaList`)

Updates the data in this record based on another object :param new\_data: The object from which to use the new values :return: None

**user**

The user associated with this list

**user\_id**

The ID of the user associated with this list

### otaku\_info.db.MediaListItem module

**class** `otaku_info.db.MediaListItem.MediaListItem` (\*args, \*\*kwargs)

Bases: `otaku_info.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****property identifier\_tuple****Returns** A tuple that uniquely identifies this database entry**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

**update** (*new\_data*: [otaku\\_info.db.MediaListItem.MediaListItem](#))

Updates the data in this record based on another object :param new\_data: The object from which to use the new values :return: None

### otaku\_info.db.MediaNotification module

**class** [otaku\\_info.db.MediaNotification.MediaNotification](#) (\*args, \*\*kwargs)**Bases:** [otaku\\_info.db.ModelMixin.ModelMixin](#), [sqlalchemy.ext.declarative.api.Model](#)

Database model that stores a media notification for a user

**\_\_init\_\_** (\*args, \*\*kwargs)

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

**id****property identifier\_tuple****Returns** A tuple that uniquely identifies this database entry**last\_update**

The last update value sent to the user

**media\_user\_state**

The media user state this notification references

**media\_user\_state\_id**

The ID of the media user state this notification references

**update** (*new\_data*: [otaku\\_info.db.MediaNotification.MediaNotification](#))

Updates the data in this record based on another object :param new\_data: The object from which to use the new values :return: None

## otaku\_info.db.MediaUserState module

**class** `otaku_info.db.MediaUserState.MediaUserState` (\*args, \*\*kwargs)  
Bases: `otaku_info.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**

**property identifier\_tuple**  
**Returns** A tuple that uniquely identifies this database entry

**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**

**media\_notification**  
Notification object for this user state

**progress**  
The user's current progress consuming the media item

**score**  
The user's score for the references media item

**update** (*new\_data*: `otaku_info.db.MediaUserState.MediaUserState`)  
Updates the data in this record based on another object :param new\_data: The object from which to use the new values :return: None

**user**  
The user associated with this user state

**user\_id**  
The ID of the user associated with this user state

**volume\_progress**  
The user's current 'volume' progress.

**otaku\_info.db.ModelMixin module****class** otaku\_info.db.ModelMixin.**ModelMixin**

Bases: puffotter.flask.db.ModelMixin.ModelMixin

Class that define methods that greatly ease working with existing database entries

**property identifier\_tuple****Returns** A tuple that's unique to this database entry**update** (*new\_data*: otaku\_info.db.ModelMixin.ModelMixin)

Updates the data in this record based on another object :param new\_data: The object from which to use the new values :return: None

**otaku\_info.db.NotificationSetting module****class** otaku\_info.db.NotificationSetting.**NotificationSetting** (\*args, \*\*kwargs)

Bases: otaku\_info.db.ModelMixin.ModelMixin, sqlalchemy.ext.declarative.api.Model

Database model that stores notification settings for a user

**\_\_init\_\_** (\*args, \*\*kwargs)

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

**id****property identifier\_tuple****Returns** A tuple that uniquely identifies this database entry**minimum\_score**

The minimum score for notification items

**notification\_type**

The notification type

**update** (*new\_data*: otaku\_info.db.NotificationSetting.NotificationSetting)

Updates the data in this record based on another object :param new\_data: The object from which to use the new values :return: None

**user**

The user associated with this notification setting

**user\_id**

The ID of the user associated with this notification setting

**value**

Whether or not the notification is active or not

## otaku\_info.db.ServiceUsername module

**class** otaku\_info.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**

**property identifier\_tuple**

**Returns** A tuple that uniquely identifies this database entry

**service**

The external service this item is a username for

**update** (new\_data: otaku\_info.db.ServiceUsername.ServiceUsername)

Updates the data in this record based on another object :param new\_data: The object from which to use the new values :return: None

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

## otaku\_info.external package

### Subpackages

## otaku\_info.external.entities package

### Submodules



**otaku\_info.external.entities.AnilistItem module**

```
class otaku_info.external.entities.AnilistItem.AnilistItem(_id: int, service:
    otaku_info.enums.ListService,
    extra_ids:
    Dict[otaku_info.enums.ListService,
    str], media_type:
    otaku_info.enums.MediaType,
    media_subtype:
    otaku_info.enums.MediaSubType,
    english_title: Optional[str],
    romaji_title: str,
    cover_url: str, chapters: Optional[int],
    volumes: Optional[int], episodes:
    Optional[int], releasing_state:
    otaku_info.enums.ReleasingState,
    relations:
    Dict[Tuple[otaku_info.enums.MediaType,
    int],
    otaku_info.enums.MediaRelationType])
```

Bases: *otaku\_info.external.entities.AnimeListItem.AnimeListItem*

Class that models a general anilist list item Represents the information fetched using anilist's API

```
classmethod from_query(media_type: otaku_info.enums.MediaType, data: Dict[str, Any]) →
    otaku_info.external.entities.AnilistItem.AnilistItem
```

Generates an AnilistItem from a dictionary generated by an API query :param media\_type: The media type of the item :param data: The data to use :return: The generated AnilistItem

```
property myanimelist_id
```

**Returns** The myanimelist ID

**otaku\_info.external.entities.AnilistUserItem module**

```
class otaku_info.external.entities.AnilistUserItem.AnilistUserItem(_id: int,  
    service: otaku_info.enums.ListService,  
    extra_ids: Dict[otaku_info.enums.ListService,  
    str], media_type: otaku_info.enums.MediaType,  
    media_subtype: otaku_info.enums.MediaSubType,  
    english_title: Optional[str],  
    romaji_title: str,  
    cover_url: str, chapters: Optional[int],  
    volumes: Optional[int],  
    episodes: Optional[int],  
    releasing_state: otaku_info.enums.ReleasingState,  
    relations: Dict[Tuple[otaku_info.enums.MediaType,  
    otaku_info.enums.MediaRelationType],  
    score: Optional[int],  
    progress: Optional[int],  
    volume_progress: Optional[int],  
    consuming_state: otaku_info.enums.ConsumingState,  
    list_name: str)
```

Bases: *otaku\_info.external.entities.AnilistItem.AnilistItem*

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

```
__init__ (_id: int, service: otaku_info.enums.ListService, extra_ids:
    Dict[otaku_info.enums.ListService, str], media_type: otaku_info.enums.MediaType,
    media_subtype: otaku_info.enums.MediaSubType, english_title: Optional[str], ro-
    maji_title: str, cover_url: str, chapters: Optional[int], volumes: Optional[int],
    episodes: Optional[int], releasing_state: otaku_info.enums.ReleasingState, relations:
    Dict[Tuple[otaku_info.enums.MediaType, int], otaku_info.enums.MediaRelationType],
    score: Optional[int], progress: Optional[int], volume_progress: Optional[int], consum-
    ing_state: otaku_info.enums.ConsumingState, list_name: str)
```

Initializes the AnilistItem object :param \_id: The anilist ID :param service: Anilist :param extra\_ids: The myanimelist 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 Japanese 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 volumes: The total amount of known manga/ln volumes :param episodes: The total amount of known anime episodes :param releasing\_state: The current releasing state of the series :param relations: Related media items identified by IDs :param score: The user's score for the series :param progress: The user's progress for the series :param volume\_progress: The user's volume progress :param consuming\_state: The user's consumption state for the series :param list\_name: Which of the user's lists this entry belongs to

```
classmethod from_query (media_type: otaku_info.enums.MediaType, data: Dict[str, Any]) →
    otaku_info.external.entities.AnilistUserItem.AnilistUserItem
```

Generates an AnilistUserItem from a dictionary generated by an API query :param media\_type: The media type of the item :param data: The data to use :return: The generated AnilistItem

## otaku\_info.external.entities.AnimeListItem module

```
class otaku_info.external.entities.AnimeListItem.AnimeListItem (_id: int, service:
    otaku_info.enums.ListService,
    extra_ids:
    Dict[otaku_info.enums.ListService,
    str], media_type:
    otaku_info.enums.MediaType,
    media_subtype:
    otaku_info.enums.MediaSubType,
    english_title:
    Optional[str],
    romaji_title:
    str, cover_url:
    str, chapters:
    Optional[int],
    volumes: Op-
    tional[int],
    episodes: Op-
    tional[int],
    releasing_state:
    otaku_info.enums.ReleasingState,
    relations:
    Dict[Tuple[otaku_info.enums.MediaType,
    int],
    otaku_info.enums.MediaRelationType])
```

Bases: object

Class that models a general anime list item

```
__init__ (_id: int, service: otaku_info.enums.ListService, extra_ids: Dict[otaku_info.enums.ListService, str], media_type: otaku_info.enums.MediaType, media_subtype: otaku_info.enums.MediaSubType, english_title: Optional[str], romaji_title: str, cover_url: str, chapters: Optional[int], volumes: Optional[int], episodes: Optional[int], releasing_state: otaku_info.enums.ReleasingState, relations: Dict[Tuple[otaku_info.enums.MediaType, int], otaku_info.enums.MediaRelationType])
```

Initializes the AnimeListItem object :param \_id: The anime list ID :param service: The anime list service :param extra\_ids: Any extra IDs for other services :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 Japanese 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 volumes: The total amount of known manga/volumes :param episodes: The total amount of known anime episodes :param releasing\_state: The current releasing state of the series :param relations: Related media items identified by IDs

```
classmethod from_query (media_type: otaku_info.enums.MediaType, data: Dict[str, Any]) → otaku_info.external.entities.AnimeListItem.AnimeListItem
```

Generates an AnimeListItem from a dictionary generated by an API query :param media\_type: The media type of the item :param data: The data to use :return: The generated AnimeListItem

**property latest\_release**

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

## otaku\_info.external.entities.MangadexItem module

```
class otaku_info.external.entities.MangadexItem.MangadexItem (mangadex_id: int, external_ids: Dict[otaku_info.enums.ListService, str], title: str, cover_url: str, total_chapters: Optional[int], latest_chapter: Optional[int], releasing_state: otaku_info.enums.ReleasingState)
```

Bases: object

Class that models a general anilist list item Represents the information fetched using anilist's API

```
__init__ (mangadex_id: int, external_ids: Dict[otaku_info.enums.ListService, str], title: str, cover_url: str, total_chapters: Optional[int], latest_chapter: Optional[int], releasing_state: otaku_info.enums.ReleasingState)
```

Initializes the MangadexItem object :param mangadex\_id: The mangadex ID of the series :param external\_ids: IDs for other services :param title: The title of the series :param cover\_url: URL to a cover image for this series :param total\_chapters: The total amount of chapters :param latest\_chapter: The latest chapter :param releasing\_state: The releasing state

```
classmethod from_json (mangadex_id: int, data: Dict[str, Any]) → otaku_info.external.entities.MangadexItem.MangadexItem
```

Generates a MangadexItem from a dictionary generated by an API query :param mangadex\_id: The mangadex ID :param data: The data to use :return: The generated MangadexItem

```
static parse_chapter (chapter_string: str) → Optional[int]
```

Converts a mangadex chapter string into an integer :param chapter\_string: The chapter string to convert

:return: The converted integer

**static resolve\_releasing\_state** (*state\_code: int*) → *otaku\_info.enums.ReleasingState*

Translates mangadex status codes to releasing states :param state\_code: The status code to translate :return: The releasing state

## otaku\_info.external.entities.MyanimelistItem module

```
class otaku_info.external.entities.MyanimelistItem.MyanimelistItem(_id: int,
    service: otaku_info.enums.ListService,
    extra_ids: Dict[otaku_info.enums.ListService, str],
    media_type: otaku_info.enums.MediaType,
    media_subtype: otaku_info.enums.MediaSubType,
    english_title: Optional[str],
    romaji_title: str,
    cover_url: str,
    chapters: Optional[int],
    volumes: Optional[int],
    episodes: Optional[int],
    releasing_state: otaku_info.enums.ReleasingState,
    relations: Dict[Tuple[otaku_info.enums.MediaType, otaku_info.enums.MediaRelationType]
```

Bases: *otaku\_info.external.entities.AnimeListItem.AnimeListItem*

Class that models a general myanimelist list item Represents the information fetched using myanimelist's jikan API

**classmethod from\_query** (*media\_type: otaku\_info.enums.MediaType*, *data: Dict[str, Any]*) → *otaku\_info.external.entities.MyanimelistItem.MyanimelistItem*

Generates an MyanimelistItem from a dictionary generated by an API query :param media\_type: The media type of the item :param data: The data to use :return: The generated MyanimelistItem

**static resolve\_media\_subtype** (*subtype\_string: str*) → *otaku\_info.enums.MediaSubType*

Resolves myanimelist media subtype :param subtype\_string: The string to resolve :return: The resolved

MediaSubType

**static resolve\_relation\_type** (*relation\_string: str*) → *otaku\_info.enums.MediaRelationType*

Resolves myanimelist relation types :param relation\_string: The string to resolve :return: The resolved MediaRelationType

**static resolve\_releasing\_state** (*releasing\_string: str*) → *otaku\_info.enums.ReleasingState*

Resolves myanimelist releasing state :param releasing\_string: The string to resolve :return: The resolved ReleasingState

## otaku\_info.external.entities.RedditLnRelease module

```
class otaku_info.external.entities.RedditLnRelease.RedditLnRelease (series_name:  
str, year:  
int, re-  
lease_date_string:  
str, vol-  
ume: str,  
publisher:  
Op-  
tional[str],  
pur-  
chase_link:  
Op-  
tional[str],  
info_link:  
Op-  
tional[str],  
digital:  
bool,  
physical:  
bool)
```

Bases: object

Object that acts as a wrapper around a light novel release on reddit.com

```
__init__ (series_name: str, year: int, release_date_string: str, volume: str, publisher: Optional[str],  
purchase_link: Optional[str], info_link: Optional[str], digital: bool, physical: bool)
```

Initializes the object :param series\_name: The name of the series :param year: The year of release :param release\_date\_string: The month and day of release :param volume: The volume number :param publisher: The publisher of the release :param purchase\_link: A purchase link for the release :param info\_link: Link to information for the release :param digital: Whether or not the release is digital :param physical: Whether or not the release is physical

**property anilist\_id**

**Returns** The anilist ID, if available

```
classmethod from_parts (year: int, parts: List[bs4.element.Tag]) →  
otaku_info.external.entities.RedditLnRelease.RedditLnRelease
```

Generates a reddit LN release from BeautifulSoup td tags :param year: The year of the release :param parts: The td tags :return: The reddit ln release

**property myanimelist\_id**

**Returns** The myanimelist ID, if available

**property release\_date**

**Returns** The release date as a datetime object

**property** `release_date_string`

**Returns** The release date as a string (ISO 8601)

## Module contents

### Submodules

#### otaku\_info.external.anilist module

`otaku_info.external.anilist.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.external.anilist.load_anilist` (*username*: *str*, *media\_type*: *otaku\_info.enums.MediaType*) → List[*otaku\_info.external.entities.AnilistUserItem.AnilistUserItem*]

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

`otaku_info.external.anilist.load_anilist_info` (*service\_id*: *int*, *media\_type*: *otaku\_info.enums.MediaType*, *service*: *otaku\_info.enums.ListService* = *<ListService.ANILIST: 'anilist'>*) → Optional[*otaku\_info.external.entities.AnilistItem.AnilistItem*]

Loads information for a single anilist media item :param *service\_id*: The anilist or myanimelist media ID :param *media\_type*: The media type :param *service*: The service the ID belongs to

(either anilist or myanimelist)

**Returns** The fetched AnilistItem

#### otaku\_info.external.mangadex module

`otaku_info.external.mangadex.fetch_mangadex_item` (*mangadex\_id*: *int*) → Optional[*otaku\_info.external.entities.MangadexItem.MangadexItem*]

Fetches information for a mangadex

#### otaku\_info.external.myanimelist module

`otaku_info.external.myanimelist.load_myanimelist_item` (*myanimelist\_id*: *int*, *media\_type*: *otaku\_info.enums.MediaType*) → Optional[*otaku\_info.external.entities.MyanimelistItem.MyanimelistItem*]

Loads myanimelist data using the jikan API :param *myanimelist\_id*: The myanimelist ID :param *media\_type*: The media type :return: The myanimelist item

## otaku\_info.external.reddit module

`otaku_info.external.reddit.load_ln_releases` (*year*: *Optional[int]* = *None*) → *List[otaku\_info.external.entities.RedditLnRelease.RedditLnRelease]*  
Loads the light novel releases

`otaku_info.external.reddit.load_tables` (*year*: *int*) → *List[bs4.BeautifulSoup]*  
Loads the tables containing the release data

## Module contents

### otaku\_info.routes package

#### Subpackages

#### otaku\_info.routes.api package

#### Submodules

#### otaku\_info.routes.api.media\_api module

`otaku_info.routes.api.media_api.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

#### Submodules

#### otaku\_info.routes.external\_service module

`otaku_info.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.routes.ln module

`otaku_info.routes.ln.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





**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.test.db.TestMangaChapterGuess module

**class** `otaku_info.test.db.TestMangaChapterGuess.TestMangaChapterGuess` (*methodName='runTest'*)  
Bases: `otaku_info.test.TestFramework._TestFramework`

Class that tests the MangaChapterGuess database model

**static generate\_sample\_guess()** → `Tuple[otaku_info.db.MediaItem.MediaItem, otaku_info.db.MediaId.MediaId, otaku_info.db.MangaChapterGuess.MangaChapterGuess]`  
Generates a sample chapter guess :return: The media item, media id and chapter guess

**test\_equality()**  
Tests checking equality for model objects :return: None

**test\_guessing\_latest\_chapter()**  
Tests guessing the latest chapter :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\_uniqueness()**  
Tests if the uniqueness of the model is handled properly :return: None

### otaku\_info.test.db.TestMediaId module

**class** `otaku_info.test.db.TestMediaId.TestMediaId` (*methodName='runTest'*)  
Bases: `otaku_info.test.TestFramework._TestFramework`

Class that tests the MediaId database model

**static generate\_sample\_media\_id()** → `Tuple[otaku_info.db.MediaItem.MediaItem, otaku_info.db.MediaId.MediaId]`  
Generates a media id :return: The media item and media id

**test\_equality()**  
Tests checking equality for model objects :return: None

**test\_generating\_service\_url()**  
Tests generating the generating of service URLs :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_uniqueness ()
    Tests if the uniqueness of the model is handled properly :return: None

```

### otaku\_info.test.db.TestMediaItem module

```

class otaku_info.test.db.TestMediaItem.TestMediaItem (methodName='runTest')
    Bases: otaku_info.test.TestFramework._TestFramework

    Class that tests the MediaItem database model

    static generate_sample_media_item () → otaku_info.db.MediaItem.MediaItem
        Generates a sample media item :return: The media item

    test_equality ()
        Tests checking equality for model objects :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_title ()
        Tests the title attribute of the media item :return: None

    test_uniqueness ()
        Tests that a mediaitem can only exist once :return: None

```

### otaku\_info.test.db.TestMediaList module

```

class otaku_info.test.db.TestMediaList.TestMediaList (methodName='runTest')
    Bases: otaku_info.test.TestFramework._TestFramework

    Class that tests the MediaList database model

    generate_sample_media_list () → Tuple[otaku_info.db.MediaList.MediaList, puffot-
        ter.flask.db.User.User]
        Generates a sample media item :return: The media list and the associated user

    test_equality ()
        Tests checking equality for model objects :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\_uniqueness()**  
Tests if the uniqueness of the model is handled properly :return: None

### otaku\_info.test.db.TestMediaListItem module

**class** `otaku_info.test.db.TestMediaListItem`.**TestMediaListItem** (*methodName='runTest'*)  
Bases: `otaku_info.test.TestFramework._TestFramework`

Class that tests the MediaListItem database model

**generate\_sample\_media\_list\_item()** → Tuple[`otaku_info.db.MediaListItem.MediaListItem`,  
`otaku_info.db.MediaList.MediaList`,  
`otaku_info.db.MediaUserState.MediaUserState`,  
`puffotter.flask.db.User.User`,  
`otaku_info.db.MediaItem.MediaItem`,  
`otaku_info.db.MediaId.MediaId`]

Generates a media list item :return: The media list item, media list, media user state, user,  
media item and media id

**test\_equality()**  
Tests checking equality for model objects :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\_uniqueness()**  
Tests if the uniqueness of the model is handled properly :return: None

**otaku\_info.test.db.TestMediaRelation module****otaku\_info.test.db.TestMediaUserState module**

**class** `otaku_info.test.db.TestMediaUserState.TestMediaUserState` (*methodName='runTest'*)

Bases: `otaku_info.test.TestFramework._TestFramework`

Class that tests the MediaUserState database model

**generate\_sample\_media\_user\_state** () → Tuple[`otaku_info.db.MediaUserState.MediaUserState`,  
`puffotter.flask.db.User.User`,  
`otaku_info.db.MediaItem.MediaItem`,  
`otaku_info.db.MediaId.MediaId`]

Generates a media user state :return: The media user state, user, media item and media id

**test\_equality** ()

Tests checking equality for model objects :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\_uniqueness** ()

Tests if the uniqueness of the model is handled properly :return: None

**otaku\_info.test.db.TestServiceUsername module**

**class** `otaku_info.test.db.TestServiceUsername.TestServiceUsername` (*methodName='runTest'*)

Bases: `otaku_info.test.TestFramework._TestFramework`

Class that tests the ServiceUsername database model

**generate\_sample\_service\_username** () → Tuple[`otaku_info.db.ServiceUsername.ServiceUsername`,  
`puffotter.flask.db.User.User`]

Generates a sample service username :return: The service username and the user

**test\_equality** ()

Tests checking equality for model objects :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\_uniqueness** ()

Tests if the uniqueness of the model is handled properly :return: None

### otaku\_info.test.db.TestUser module

```
class otaku_info.test.db.TestUser.TestUser (methodName='runTest')
    Bases: otaku_info.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.test.misc package

#### Submodules

### otaku\_info.test.misc.TestApiCalls module

```
class otaku_info.test.misc.TestApiCalls.TestConfig (methodName='runTest')
    Bases: otaku_info.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.test.misc.TestConfig module

**class** otaku\_info.test.misc.TestConfig.**TestConfig** (*methodName='runTest'*)

Bases: otaku\_info.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.test.misc.TestErrorHandling module

**class** otaku\_info.test.misc.TestErrorHandling.**TestErrorHandling** (*methodName='runTest'*)

Bases: otaku\_info.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.test.misc.TestMappings module

**class** otaku\_info.test.misc.TestMappings.**TestMappings** (*methodName='runTest'*)

Bases: unittest.case.TestCase

Class that tests enum mappings

**test\_completeness** ()

Tests that mappings include all possible enum types :return: None

### otaku\_info.test.misc.TestServer module

**class** otaku\_info.test.misc.TestServer.**TestServer** (*methodName='runTest'*)

Bases: otaku\_info.test.TestFramework.\_TestFramework

Class that tests starting the server

**test\_starting\_server** ()

Tests starting the server :return: None

## Module contents

otaku\_info.test.routes package

## Subpackages

otaku\_info.test.routes.api package

## Submodules

otaku\_info.test.routes.api.TestApiKeyRoute module

```
class otaku_info.test.routes.api.TestApiKeyRoute.TestApiKeyRoute (methodName='runTest')
    Bases: otaku_info.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.test.routes.TestForgotRoute module

```
class otaku_info.test.routes.TestForgotRoute.TestForgotRoute (methodName='runTest')
    Bases: otaku_info.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.test.routes.TestLoginRoute module

**class** otaku\_info.test.routes.TestLoginRoute.**TestLoginRoute** (*methodName='runTest'*)

Bases: otaku\_info.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.test.routes.TestProfileRoute module

**class** otaku\_info.test.routes.TestProfileRoute.**TestProfileRoute** (*methodName='runTest'*)

Bases: otaku\_info.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.test.routes.TestRegisterRoute module

**class** otaku\_info.test.routes.TestRegisterRoute.**TestRegisterRoute** (*methodName='runTest'*)

Bases: otaku\_info.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.test.routes.TestStaticRoutes module

**class** `otaku_info.test.routes.TestStaticRoutes.TestStaticRoutes` (*methodName='runTest'*)  
Bases: `otaku_info.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

#### otaku\_info.test.utils package

### Module contents

#### Submodules

#### otaku\_info.test.TestFramework module

### Module contents

#### otaku\_info.utils package

### Subpackages

#### otaku\_info.utils.db package

### Submodules

#### otaku\_info.utils.db.DbCache module

**class** `otaku_info.utils.db.DbCache.DbCache`  
Bases: `object`

Class that helps identifying existing items by caching all database entries

**static cleanup()**  
Removes any cached items for the current thread :return: None

**static get\_existing\_item** (*item:* `otaku_info.db.ModelMixin.ModelMixin`) → `Optional[otaku_info.db.ModelMixin.ModelMixin]`  
Retrieves an existing item from the cache if it exists :param item: The item for which to retrieve the existing item :return: The existing item or None

**static load\_existing\_items** (*cls: Type[otaku\_info.db.ModelMixin.ModelMixin], reload: bool = False*) → Dict[Tuple, *otaku\_info.db.ModelMixin.ModelMixin*]

Retrieves all existing items for a database class mapped to their identifier tuples. Automatically separates the results by thread. :param cls: The database class :param reload: Whether to force a reload of database data :return: The existing entries mapped to identifier tuples

**static update\_item** (*item: otaku\_info.db.ModelMixin.ModelMixin*)

Updates a cached item :param item: The new item :return: None

## otaku\_info.utils.db.convert module

otaku\_info.utils.db.convert.**anilist\_user\_item\_to\_media\_list** (*anilist\_user\_item: otaku\_info.external.entities.AnilistUserItem.AnilistUserItem, user: puffot-ter.flask.db.User.User*) → *otaku\_info.db.MediaList.MediaList*

Generates a media list from an anilist user item :param anilist\_user\_item: The anilist user item :param user: The corresponding user :return: The media list object

otaku\_info.utils.db.convert.**anilist\_user\_item\_to\_media\_user\_state** (*anilist\_user\_item: otaku\_info.external.entities.AnilistUserItem.AnilistUserItem, media\_id: otaku\_info.db.MediaId.MediaId, user: puffot-ter.flask.db.User.User*) → *otaku\_info.db.MediaUserState.MediaUserState*

Generates a media user state based on an anilist user item :param anilist\_user\_item: The anilist user item :param media\_id: The corresponding media id :param user: The corresponding user :return: The media user state

otaku\_info.utils.db.convert.**anime\_list\_item\_to\_media\_id** (*anime\_list\_item: otaku\_info.external.entities.AnimeListItem.AnimeListItem, media\_item: otaku\_info.db.MediaItem.MediaItem, list\_service: Optional[otaku\_info.enums.ListService] = None*) → *otaku\_info.db.MediaId.MediaId*

Converts an anime list item into a media id :param anime\_list\_item: The anime list item :param media\_item: The corresponding media item :param list\_service: Uses an extra ID with the specified list service :return: The generated media id

otaku\_info.utils.db.convert.**anime\_list\_item\_to\_media\_item** (*anime\_list\_item: otaku\_info.external.entities.AnimeListItem.AnimeListItem*) → *otaku\_info.db.MediaItem.MediaItem*

Converts an anime list item to a media item object :param anime\_list\_item: The anime list item :return: The media item

otaku\_info.utils.db.convert.**ln\_release\_from\_reddit\_item** (*ln\_release: otaku\_info.external.entities.RedditLnRelease.RedditLnRelease, media\_item: Optional[otaku\_info.db.MediaItem.MediaItem]*) → *otaku\_info.db.LnRelease.LnRelease*

Creates an LnRelease object based on a RedditLnRelease object :param ln\_release: The RedditLnRelease object  
:param media\_item: An associated MediaItem :return: The LnRelease object

```
otaku_info.utils.db.convert.mangadex_item_to_media_item(mangadex_item:
                                                         otaku_info.external.entities.MangadexItem.MangadexItem
                                                         →
                                                         otaku_info.db.MediaItem.MediaItem)
```

Converts a mangadex item to a media item object :param mangadex\_item: The mangadex item :return: The media item

### otaku\_info.utils.db.load module

```
otaku_info.utils.db.load.load_service_ids(limit_media_type_to: Optional[otaku_info.enums.MediaType] = None)
                                         → Dict[otaku_info.enums.ListService, Dict[str, otaku_info.db.MediaId.MediaId]]
```

Loads Media IDs from the database and maps them to their service id :param limit\_media\_type\_to: If set, limits the media type :return: The mapped ids

### otaku\_info.utils.db.updater module

```
otaku_info.utils.db.updater.update_or_insert_item(to_add: otaku_info.db.ModelMixin.ModelMixin,
                                                    commit_updates: bool = False,
                                                    reload_cache: bool = False) → otaku_info.db.ModelMixin.ModelMixin
```

Updates or creates an item that can be identified based on a unique tuple. Ensures that no duplicates are inserted across threads :param to\_add: The item to add :param commit\_updates: Whether or not to commit updates :param reload\_cache: Reloads the cached data if set to True :return: None

## Module contents

### Submodules

#### otaku\_info.utils.dates module

```
otaku_info.utils.dates.map_month_name_to_month_number(month_name: str) → Optional[int]
```

Maps month names to month numbers :param month\_name: The name of the month :return: The month number

```
otaku_info.utils.dates.map_month_number_to_month_name(month_number: int) → Optional[str]
```

Maps month numbers to month names :param month\_number: The month number :return: The name of the month

## Module contents

### otaku\_info.wrappers package

#### Submodules

#### otaku\_info.wrappers.UpdateWrapper module

**class** `otaku_info.wrappers.UpdateWrapper.UpdateWrapper` (*media\_user\_state:* `otaku_info.db.MediaUserState.MediaUserState`)

Bases: `object`

Class that encapsulates important data to display for manga updates

**\_\_init\_\_** (*media\_user\_state:* `otaku_info.db.MediaUserState.MediaUserState`)

Initializes the MangaUpdate object :param `media_user_state`: The media user state contains everything we need. For performance reasons, it's assumed that all relations have been loaded already.

**calculate\_latest** () → `int`

**Returns** The latest release number

**calculate\_progress** () → `int`

**Returns** The user's current progress

**classmethod** **from\_db** (*user:* `puffotter.flask.db.User.User`, *list\_name:* `str`, *service:* `otaku_info.enums.ListService`, *media\_type:* `otaku_info.enums.MediaType`, *media\_subtype:* `Optional[otaku_info.enums.MediaSubType]`, *minimum\_diff:* `int`, *include\_complete:* `bool`)

Generates UpdateWrapper objects based on a couple of parameters and the current database contents :param `user`: The user for whom to load the updates :param `list_name`: The list name for which to load the updates :param `service`: The service for which to load the updates :param `media_type`: The media type for which to load the updates :param `media_subtype`: If specified, limits the results to a specific

`media subtype` (example: Light novels)

#### Parameters

- **minimum\_diff** – Specifies a minimum diff value
- **include\_complete** – Specifies whether completed items should be included

**Returns** A list of UpdateWrapper objects

**classmethod** **from\_media\_lists** (*media\_lists:* `List[otaku_info.db.MediaList.MediaList]`, *media\_subtype:* `Optional[otaku_info.enums.MediaSubType]`, *minimum\_diff:* `int`, *include\_complete:* `bool`) → `List[otaku_info.wrappers.UpdateWrapper.UpdateWrapper]`

Generates UpdateWrapper objects based on media lists To avoid huge performance problems, the relations should be fully loaded beforehand, preferably in a single query. Results can be filtered using the additional parameters :param `media_lists`: The media lists containing the items to wrap :param `media_subtype`: If specified, limits the results to a specific

`media subtype` (example: Light novels)

#### Parameters

- **minimum\_diff** – Specifies a minimum diff value

- `include_complete` – Specifies whether completed items should be included

**Returns** A list of UpdateWrapper objects

## Module contents

### 1.1.2 Submodules

#### 1.1.3 `otaku_info.Config` module

**class** `otaku_info.Config.Config`

Bases: `puffotter.flask.Config.Config`

Configuration for the flask application

**TELEGRAM\_API\_KEY:** `str = None`

API Key for the telegram bot used for notification messages

**TELEGRAM\_BOT\_CONNECTION:** `TelegramBotConnection = None`

Single Telegram bot connection used for all telegram communications

#### 1.1.4 `otaku_info.enums` module

**class** `otaku_info.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.enums.ListService`

Bases: `enum.Enum`

Class that defines available list services

**ANILIST** = `'anilist'`

**ANIMEPLANET** = `'animeplanet'`

**KITSU** = `'kitsu'`

**MANGADEX** = `'mangadex'`

**MANGAUPDATES** = `'mangaupdates'`

**MYANIMELIST** = `'myanimelist'`

**class** `otaku_info.enums.MediaRelationType`

Bases: `enum.Enum`

Class that models a media relation type

**ADAPTATION** = `'adaptation'`

```
ALTERNATIVE = 'alternative'  
CHARACTER = 'character'  
COMPILATION = 'compilation'  
CONTAINS = 'contains'  
OTHER = 'other'  
PARENT = 'parent'  
PREQUEL = 'prequel'  
SEQUEL = 'sequel'  
SIDE_STORY = 'side_story'  
SOURCE = 'source'  
SPIN_OFF = 'spin_off'  
SUMMARY = 'summary'
```

```
class otaku_info.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.enums.MediaType
```

```
    Bases: enum.Enum
```

```
    Class that models a media type for media items
```

```
ANIME = 'anime'  
MANGA = 'manga'
```

```
class otaku_info.enums.NotificationType
```

```
    Bases: enum.Enum
```

```
    Class that defines the possible notification types
```

```
NEW_ANIME_EPISODES = 'new_anime_episodes'  
NEW_MANGA_CHAPTERS = 'new_manga_chapters'
```

```
class otaku_info.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'
```

### 1.1.5 otaku\_info.main module

```
otaku_info.main.main()
    Starts the flask application :return: None
```

### 1.1.6 otaku\_info.mappings module

```
otaku_info.mappings.list_service_id_types: Dict[otaku_info.enums.ListService, Type] = {<L
    Which type a list service ID should have

otaku_info.mappings.list_service_priorities: List[otaku_info.enums.ListService] = [<ListS
    Specifies the order in which list service data is prioritized

otaku_info.mappings.list_service_url_formats: Dict[otaku_info.enums.ListService, str] = {
    Schemas for URLs for external services

otaku_info.mappings.mangadex_external_id_names: Dict[otaku_info.enums.ListService, str] =
    The names used by mangadex to identify IDs from other services
```

### 1.1.7 otaku\_info.template\_extras module

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

### 1.1.8 Module contents

```
otaku_info.root_path: str = '/usr/local/lib/python3.6/dist-packages/otaku_info-0.3.2-py3.
    The root path of the application

otaku_info.sentry_dsn = 'https://f899b0c46d324f37b83527a3994afd8d@sentry.namibsun.net/18'
    The sentry DSN used for error logging
```



## INDICES AND TABLES

- genindex
- modindex
- search



## PYTHON MODULE INDEX

### O

otaku\_info, 36

otaku\_info.background, 4

otaku\_info.background.anilist, 3

otaku\_info.background.ln\_releases, 3

otaku\_info.background.manga\_chapter\_guesses, 4

otaku\_info.background.mangadex, 4

otaku\_info.background.notifications, 4

otaku\_info.Config, 34

otaku\_info.db, 12

otaku\_info.db.LnRelease, 4

otaku\_info.db.MangaChapterGuess, 5

otaku\_info.db.MediaId, 6

otaku\_info.db.MediaItem, 7

otaku\_info.db.MediaList, 8

otaku\_info.db.MediaListItem, 8

otaku\_info.db.MediaNotification, 9

otaku\_info.db.MediaUserState, 10

otaku\_info.db.ModelMixin, 11

otaku\_info.db.NotificationSetting, 11

otaku\_info.db.ServiceUsername, 12

otaku\_info.enums, 34

otaku\_info.external, 20

otaku\_info.external.anilist, 19

otaku\_info.external.entities, 19

otaku\_info.external.entities.AnilistItem, 13

otaku\_info.external.entities.AnilistUserItem, 14

otaku\_info.external.entities.AnimeListItem, 15

otaku\_info.external.entities.MangadexItem, 16

otaku\_info.external.entities.MyanimelistItem, 17

otaku\_info.external.entities.RedditLnRelease, 18

otaku\_info.external.mangadex, 19

otaku\_info.external.myanimelist, 19

otaku\_info.external.reddit, 20

otaku\_info.main, 36

otaku\_info.mappings, 36

otaku\_info.routes, 21

otaku\_info.routes.api, 20

otaku\_info.routes.api.media\_api, 20

otaku\_info.routes.external\_service, 20

otaku\_info.routes.ln, 20

otaku\_info.routes.media, 21

otaku\_info.routes.notifications, 21

otaku\_info.routes.updates, 21

otaku\_info.template\_extras, 36

otaku\_info.test, 30

otaku\_info.test.db, 26

otaku\_info.test.db.TestApiKey, 21

otaku\_info.test.db.TestMangaChapterGuess, 22

otaku\_info.test.db.TestMediaId, 22

otaku\_info.test.db.TestMediaItem, 23

otaku\_info.test.db.TestMediaList, 23

otaku\_info.test.db.TestMediaListItem, 24

otaku\_info.test.db.TestMediaRelation, 25

otaku\_info.test.db.TestMediaUserState, 25

otaku\_info.test.db.TestServiceUsername, 25

otaku\_info.test.db.TestUser, 26

otaku\_info.test.misc, 28

otaku\_info.test.misc.TestApiCalls, 26

otaku\_info.test.misc.TestConfig, 27

otaku\_info.test.misc.TestErrorHandling, 27

otaku\_info.test.misc.TestMappings, 27

otaku\_info.test.misc.TestServer, 27

otaku\_info.test.routes, 30

otaku\_info.test.routes.api, 28

otaku\_info.test.routes.api.TestApiKeyRoute, 28

otaku\_info.test.routes.TestForgotRoute, 28

otaku\_info.test.routes.TestLoginRoute, 29

otaku\_info.test.routes.TestProfileRoute,  
    29  
otaku\_info.test.routes.TestRegisterRoute,  
    29  
otaku\_info.test.routes.TestStaticRoutes,  
    30  
otaku\_info.test.TestFramework, 30  
otaku\_info.test.utils, 30  
otaku\_info.utils, 33  
otaku\_info.utils.dates, 32  
otaku\_info.utils.db, 32  
otaku\_info.utils.db.convert, 31  
otaku\_info.utils.db.DbCache, 30  
otaku\_info.utils.db.load, 32  
otaku\_info.utils.db.updater, 32  
otaku\_info.wrappers, 34  
otaku\_info.wrappers.UpdateWrapper, 33

## Symbols

- `__init__()` (*otaku\_info.db.LnRelease.LnRelease* method), 4
  - `__init__()` (*otaku\_info.db.MangaChapterGuess.MangaChapterGuess* method), 5
  - `__init__()` (*otaku\_info.db.MediaId.MediaId* method), 6
  - `__init__()` (*otaku\_info.db.MediaItem.MediaItem* method), 7
  - `__init__()` (*otaku\_info.db.MediaList.MediaList* method), 8
  - `__init__()` (*otaku\_info.db.MediaListItem.MediaListItem* method), 8
  - `__init__()` (*otaku\_info.db.MediaNotification.MediaNotification* method), 9
  - `__init__()` (*otaku\_info.db.MediaUserState.MediaUserState* method), 10
  - `__init__()` (*otaku\_info.db.NotificationSetting.NotificationSetting* method), 11
  - `__init__()` (*otaku\_info.db.ServiceUsername.ServiceUsername* method), 12
  - `__init__()` (*otaku\_info.external.entities.AnilistUserItem.AnilistUserItem* method), 15
  - `__init__()` (*otaku\_info.external.entities.AnimeListItem.AnimeListItem* method), 16
  - `__init__()` (*otaku\_info.external.entities.MangadexItem.MangadexItem* method), 16
  - `__init__()` (*otaku\_info.external.entities.RedditLnRelease.RedditLnRelease* method), 18
  - `__init__()` (*otaku\_info.wrappers.UpdateWrapper.UpdateWrapper* method), 33
- A**
- ADAPTATION (*otaku\_info.enums.MediaRelationType* attribute), 34
  - ALTERNATIVE (*otaku\_info.enums.MediaRelationType* attribute), 34
  - ANILIST (*otaku\_info.enums.ListService* attribute), 34
  - `anilist_id()` (*otaku\_info.external.entities.RedditLnRelease.RedditLnRelease* property), 18
  - `anilist_user_item_to_media_list()` (in module *otaku\_info.utils.db.convert*), 31
  - `anilist_user_item_to_media_user_state()` (in module *otaku\_info.utils.db.convert*), 31
  - AnilistItem (class in *otaku\_info.external.entities.AnilistItem*), 13
  - AnilistUserItem (class in *otaku\_info.external.entities.AnilistUserItem*), 14
  - ANIME (*otaku\_info.enums.MediaType* attribute), 35
  - `anime_list_item_to_media_id()` (in module *otaku\_info.utils.db.convert*), 31
  - `anime_list_item_to_media_item()` (in module *otaku\_info.utils.db.convert*), 31
  - AnimeListItem (class in *otaku\_info.external.entities.AnimeListItem*), 15
  - ANIMEPLANET (*otaku\_info.enums.ListService* attribute), 34
- B**
- background\_tasks (in module *otaku\_info.background*), 4
  - blueprint\_generators (in module *otaku\_info.background*), 4
  - otaku (*otaku\_info.routes*), 21
- C**
- `calculate_latest()` (*otaku\_info.wrappers.UpdateWrapper.UpdateWrapper* method), 33
  - `calculate_progress()` (*otaku\_info.wrappers.UpdateWrapper.UpdateWrapper* method), 33
  - CANCELLED (*otaku\_info.enums.ReleasingState* attribute), 36
  - chapter\_guess (*otaku\_info.db.MediaId.MediaId* attribute), 6
  - CHARACTER (*otaku\_info.enums.MediaRelationType* attribute), 35
  - `cleanup()` (*otaku\_info.utils.db.DbCache.DbCache* static method), 30
  - COMPLETED (*otaku\_info.enums.MediaRelationType* attribute), 35
  - COMPLETED (*otaku\_info.enums.ConsumingState* attribute), 34

Config (class in *otaku\_info.Config*), 34  
 consuming\_state (*otaku\_info.db.MediaUserState.MediaUserState* class method), 15  
     attribute), 10  
 ConsumingState (class in *otaku\_info.enums*), 34  
 CONTAINS (*otaku\_info.enums.MediaRelationType* attribute), 35  
 cover\_url (*otaku\_info.db.MediaItem.MediaItem* attribute), 7  
 create\_media\_item\_from\_reddit\_ln\_release() (in module *otaku\_info.background.ln\_releases*), 3  
 CURRENT (*otaku\_info.enums.ConsumingState* attribute), 34

## D

DbCache (class in *otaku\_info.utils.db.DbCache*), 30  
 define\_blueprint() (in module *otaku\_info.routes.api.media\_api*), 20  
 define\_blueprint() (in module *otaku\_info.routes.external\_service*), 20  
 define\_blueprint() (in module *otaku\_info.routes.ln*), 20  
 define\_blueprint() (in module *otaku\_info.routes.media*), 21  
 define\_blueprint() (in module *otaku\_info.routes.notifications*), 21  
 define\_blueprint() (in module *otaku\_info.routes.updates*), 21  
 digital (*otaku\_info.db.LnRelease.LnRelease* attribute), 4  
 DROPPED (*otaku\_info.enums.ConsumingState* attribute), 34

## E

english\_title (*otaku\_info.db.MediaItem.MediaItem* attribute), 7

## F

fetch\_mangadex\_item() (in module *otaku\_info.external.mangadex*), 19  
 FINISHED (*otaku\_info.enums.ReleasingState* attribute), 36  
 from\_db() (*otaku\_info.wrappers.UpdateWrapper.UpdateWrapper* class method), 33  
 from\_json() (*otaku\_info.external.entities.MangadexItem.MangadexItem* class method), 16  
 from\_media\_lists() (*otaku\_info.wrappers.UpdateWrapper.UpdateWrapper* class method), 33  
 from\_parts() (*otaku\_info.external.entities.RedditLnRelease.RedditLnRelease* class method), 18  
 from\_query() (*otaku\_info.external.entities.AnilistItem.AnilistItem* class method), 13  
 from\_query() (*otaku\_info.external.entities.AnilistUserItem.AnilistUserItem* class method), 15  
 from\_query() (*otaku\_info.external.entities.AnimeListItem.AnimeListItem* class method), 16  
 from\_query() (*otaku\_info.external.entities.MyanimelistItem.MyanimelistItem* class method), 17

## G

generate\_sample\_guess() (*otaku\_info.test.db.TestMangaChapterGuess.TestMangaChapterGuess* static method), 22  
 generate\_sample\_media\_id() (*otaku\_info.test.db.TestMediaId.TestMediaId* static method), 22  
 generate\_sample\_media\_item() (*otaku\_info.test.db.TestMediaItem.TestMediaItem* static method), 23  
 generate\_sample\_media\_list() (*otaku\_info.test.db.TestMediaList.TestMediaList* method), 23  
 generate\_sample\_media\_list\_item() (*otaku\_info.test.db.TestMediaListItem.TestMediaListItem* method), 24  
 generate\_sample\_media\_user\_state() (*otaku\_info.test.db.TestMediaUserState.TestMediaUserState* method), 25  
 generate\_sample\_service\_username() (*otaku\_info.test.db.TestServiceUsername.TestServiceUsername* method), 25  
 get\_existing\_item() (*otaku\_info.utils.db.DbCache.DbCache* static method), 30  
 get\_ids() (*otaku\_info.db.LnRelease.LnRelease* method), 4  
 guess (*otaku\_info.db.MangaChapterGuess.MangaChapterGuess* attribute), 5  
 guess\_latest\_manga\_chapter() (in module *otaku\_info.external.anilist*), 19

## H

handle\_notification() (in module *otaku\_info.background.notifications*), 4  
 id (*otaku\_info.db.LnRelease.LnRelease* attribute), 4  
 id (*otaku\_info.db.MangaChapterGuess.MangaChapterGuess* attribute), 5  
 id (*otaku\_info.db.MediaId.MediaId* attribute), 6  
 id (*otaku\_info.db.MediaItem.MediaItem* attribute), 7  
 id (*otaku\_info.db.MediaList.MediaList* attribute), 8  
 id (*otaku\_info.db.MediaListItem.MediaListItem* attribute), 8  
 id (*otaku\_info.db.MediaNotification.MediaNotification* attribute), 9

id (*otaku\_info.db.MediaUserState.MediaUserState* attribute), 10  
 id (*otaku\_info.db.NotificationSetting.NotificationSetting* attribute), 11  
 id (*otaku\_info.db.ServiceUsername.ServiceUsername* attribute), 12  
 identifier\_tuple() (*otaku\_info.db.LnRelease.LnRelease* property), 5  
 identifier\_tuple() (*otaku\_info.db.MangaChapterGuess.MangaChapterGuess* property), 5  
 identifier\_tuple() (*otaku\_info.db.MediaId.MediaId* property), 6  
 identifier\_tuple() (*otaku\_info.db.MediaItem.MediaItem* property), 7  
 identifier\_tuple() (*otaku\_info.db.MediaList.MediaList* property), 8  
 identifier\_tuple() (*otaku\_info.db.MediaListItem.MediaListItem* property), 9  
 identifier\_tuple() (*otaku\_info.db.MediaNotification.MediaNotification* property), 9  
 identifier\_tuple() (*otaku\_info.db.MediaUserState.MediaUserState* property), 10  
 identifier\_tuple() (*otaku\_info.db.ModelMixin.ModelMixin* property), 11  
 identifier\_tuple() (*otaku\_info.db.NotificationSetting.NotificationSetting* property), 11  
 identifier\_tuple() (*otaku\_info.db.ServiceUsername.ServiceUsername* property), 12

**K**

KITSU (*otaku\_info.enums.ListService* attribute), 34

**L**

last\_update (*otaku\_info.db.MangaChapterGuess.MangaChapterGuess* attribute), 5  
 last\_update (*otaku\_info.db.MediaNotification.MediaNotification* attribute), 9  
 latest\_release (*otaku\_info.db.MediaItem.MediaItem* attribute), 7  
 latest\_release() (*otaku\_info.external.entities.AnimeListItem.AnimeListItem* property), 16  
 latest\_volume\_release (*otaku\_info.db.MediaItem.MediaItem* attribute), 7  
 list\_service\_id\_types (*otaku\_info.mappings*), 36  
 list\_service\_priorities (*otaku\_info.mappings*), 36  
 list\_service\_url\_formats (*otaku\_info.mappings*), 36  
 ListService (class in *otaku\_info.enums*), 34  
 ln\_release\_from\_reddit\_item() (*otaku\_info.utils.db.convert*), 31  
 ln\_releases (*otaku\_info.db.MediaItem.MediaItem* attribute), 7  
 LnRelease (class in *otaku\_info.db.LnRelease*), 4  
 load\_anilist() (*otaku\_info.external.anilist*), 19  
 load\_anilist\_info() (*otaku\_info.external.anilist*), 19  
 load\_existing\_items() (*otaku\_info.utils.db.DbCache.DbCache* static method), 31  
 load\_ln\_releases() (*otaku\_info.external.reddit*), 20  
 load\_myanimelist\_item() (*otaku\_info.external.myanimelist*), 19  
 load\_service\_ids() (*otaku\_info.utils.db.load*), 32  
 load\_tables() (*otaku\_info.external.reddit*), 20

**M**

main() (*otaku\_info.main*), 36  
 MANGA (*otaku\_info.enums.MediaSubType* attribute), 35  
 MANGA (*otaku\_info.enums.MediaType* attribute), 35  
 MangaChapterGuess (class in *otaku\_info.db.MangaChapterGuess*), 5  
 MANGADDEX (*otaku\_info.enums.ListService* attribute), 34  
 mangadex\_external\_id\_names (*otaku\_info.mappings*), 36  
 mangadex\_item\_to\_media\_item() (*otaku\_info.utils.db.convert*), 32  
 MangadexItem (class in *otaku\_info.external.entities.MangadexItem*), 16  
 MANGAUPDATES (*otaku\_info.enums.ListService* attribute), 34  
 map\_month\_name\_to\_month\_number() (*otaku\_info.utils.dates*), 32  
 map\_month\_number\_to\_month\_name() (*otaku\_info.utils.dates*), 32  
 media\_id (*otaku\_info.db.MangaChapterGuess.MangaChapterGuess* attribute), 6  
 media\_id (*otaku\_info.db.MediaUserState.MediaUserState* attribute), 10

media_id_id (otaku_info.db.MangaChapterGuess.MangaChapterGuess attribute), 6	MangaChapterGuess (class in otaku_info.db.MangaChapterGuess), 6	MediaChapterGuess (class in otaku_info.db.MediaChapterGuess), 8
media_id_id (otaku_info.db.MediaUserState.MediaUserState attribute), 10	MediaUserState (class in otaku_info.db.MediaUserState), 10	MediaNotification (class in otaku_info.db.MediaNotification), 9
media_id_mapping (otaku_info.db.MediaItem.MediaItem property), 7	MediaRelationType (class in otaku_info.enums), 34	MediaSubType (class in otaku_info.enums), 35
media_ids (otaku_info.db.MediaItem.MediaItem attribute), 7	MediaUserState (class in otaku_info.db.MediaUserState), 10	MediaType (class in otaku_info.enums), 35
media_item (otaku_info.db.LnRelease.LnRelease attribute), 5	minimum_score (otaku_info.db.NotificationSetting.NotificationSetting attribute), 11	
media_item (otaku_info.db.MediaId.MediaId attribute), 6	ModelMixin (class in otaku_info.db.ModelMixin), 11	
media_item_id (otaku_info.db.LnRelease.LnRelease attribute), 5	models (in module otaku_info.db), 12	
media_item_id (otaku_info.db.MediaId.MediaId attribute), 6	module	
media_list (otaku_info.db.MediaListItem.MediaListItem attribute), 9	otaku_info, 36	
media_list_id (otaku_info.db.MediaListItem.MediaListItem attribute), 9	otaku_info.background, 4	
media_list_items (otaku_info.db.MediaList.MediaList attribute), 8	otaku_info.background.anilist, 3	
media_list_items (otaku_info.db.MediaUserState.MediaUserState attribute), 10	otaku_info.background.ln_releases, 3	
media_notification (otaku_info.db.MediaUserState.MediaUserState attribute), 10	otaku_info.background.manga_chapter_guesses, 4	
media_subtype (otaku_info.db.MediaId.MediaId attribute), 6	otaku_info.background.mangadex, 4	
media_subtype (otaku_info.db.MediaItem.MediaItem attribute), 7	otaku_info.background.notifications, 4	
media_type (otaku_info.db.MediaId.MediaId attribute), 6	otaku_info.Config, 34	
media_type (otaku_info.db.MediaItem.MediaItem attribute), 7	otaku_info.db, 12	
media_type (otaku_info.db.MediaList.MediaList attribute), 8	otaku_info.db.LnRelease, 4	
media_user_state (otaku_info.db.MediaListItem.MediaListItem attribute), 9	otaku_info.db.MangaChapterGuess, 5	
media_user_state (otaku_info.db.MediaNotification.MediaNotification attribute), 9	otaku_info.db.MediaId, 6	
media_user_state_id (otaku_info.db.MediaListItem.MediaListItem attribute), 9	otaku_info.db.MediaItem, 7	
media_user_state_id (otaku_info.db.MediaNotification.MediaNotification attribute), 9	otaku_info.db.MediaList, 8	
media_user_states (otaku_info.db.MediaId.MediaId attribute), 6	otaku_info.db.MediaListItem, 8	
MediaId (class in otaku_info.db.MediaId), 6	otaku_info.db.MediaNotification, 9	
MediaItem (class in otaku_info.db.MediaItem), 7	otaku_info.db.MediaUserState, 10	
MediaList (class in otaku_info.db.MediaList), 8	otaku_info.db.ModelMixin, 11	
	otaku_info.db.NotificationSetting, 11	
	otaku_info.db.ServiceUsername, 12	
	otaku_info.enums, 34	
	otaku_info.external, 20	
	otaku_info.external.anilist, 19	
	otaku_info.external.entities, 19	
	otaku_info.external.entities.AnilistItem, 13	
	otaku_info.external.entities.AnilistUserItem, 14	
	otaku_info.external.entities.AnimeListItem, 15	
	otaku_info.external.entities.MangadexItem, 16	
	otaku_info.external.entities.MyanimelistItem, 17	
	otaku_info.external.entities.RedditLnRelease, 18	
	otaku_info.external.mangadex, 19	



- otaku\_info.external.myanimelist, 19
  - otaku\_info.external.reddit, 20
  - otaku\_info.main, 36
  - otaku\_info.mappings, 36
  - otaku\_info.routes, 21
  - otaku\_info.routes.api, 20
  - otaku\_info.routes.api.media\_api, 20
  - otaku\_info.routes.external\_service, 20
  - otaku\_info.routes.ln, 20
  - otaku\_info.routes.media, 21
  - otaku\_info.routes.notifications, 21
  - otaku\_info.routes.updates, 21
  - otaku\_info.template\_extras, 36
  - otaku\_info.test, 30
  - otaku\_info.test.db, 26
  - otaku\_info.test.db.TestApiKey, 21
  - otaku\_info.test.db.TestMangaChapterGuess, 22
  - otaku\_info.test.db.TestMediaId, 22
  - otaku\_info.test.db.TestMediaItem, 23
  - otaku\_info.test.db.TestMediaList, 23
  - otaku\_info.test.db.TestMediaListItem, 24
  - otaku\_info.test.db.TestMediaRelation, 25
  - otaku\_info.test.db.TestMediaUserState, 25
  - otaku\_info.test.db.TestServiceUsername, 25
  - otaku\_info.test.db.TestUser, 26
  - otaku\_info.test.misc, 28
  - otaku\_info.test.misc.TestApiCalls, 26
  - otaku\_info.test.misc.TestConfig, 27
  - otaku\_info.test.misc.TestErrorHandling, 27
  - otaku\_info.test.misc.TestMappings, 27
  - otaku\_info.test.misc.TestServer, 27
  - otaku\_info.test.routes, 30
  - otaku\_info.test.routes.api, 28
  - otaku\_info.test.routes.api.TestApiKeyRoute, 28
  - otaku\_info.test.routes.TestForgotRoute, 28
  - otaku\_info.test.routes.TestLoginRoute, 29
  - otaku\_info.test.routes.TestProfileRoute, 29
  - otaku\_info.test.routes.TestRegisterRoute, 29
  - otaku\_info.test.routes.TestStaticRoutes, 30
  - otaku\_info.test.TestFramework, 30
  - otaku\_info.test.utils, 30
  - otaku\_info.utils, 33
  - otaku\_info.utils.dates, 32
  - otaku\_info.utils.db, 32
  - otaku\_info.utils.db.convert, 31
  - otaku\_info.utils.db.DbCache, 30
  - otaku\_info.utils.db.load, 32
  - otaku\_info.utils.db.updater, 32
  - otaku\_info.wrappers, 34
  - otaku\_info.wrappers.UpdateWrapper, 33
  - MOVIE (*otaku\_info.enums.MediaSubType attribute*), 35
  - MUSIC (*otaku\_info.enums.MediaSubType attribute*), 35
  - MYANIMELIST (*otaku\_info.enums.ListService attribute*), 34
  - myanimelist\_id() (*otaku\_info.external.entities.AnilistItem.AnilistItem property*), 13
  - myanimelist\_id() (*otaku\_info.external.entities.RedditLnRelease.RedditLnRelease property*), 18
  - MyanimelistItem (class in *otaku\_info.external.entities.MyanimelistItem*), 17
- ## N
- name (*otaku\_info.db.MediaList.MediaList attribute*), 8
  - NEW\_ANIME\_EPISODES (*otaku\_info.enums.NotificationType attribute*), 35
  - NEW\_MANGA\_CHAPTERS (*otaku\_info.enums.NotificationType attribute*), 35
  - NOT\_YET\_RELEASED (*otaku\_info.enums.ReleasingState attribute*), 36
  - notification\_type (*otaku\_info.db.NotificationSetting.NotificationSetting attribute*), 11
  - NotificationSetting (class in *otaku\_info.db.NotificationSetting*), 11
  - NotificationType (class in *otaku\_info.enums*), 35
  - NOVEL (*otaku\_info.enums.MediaSubType attribute*), 35
- ## O
- ONA (*otaku\_info.enums.MediaSubType attribute*), 35
  - ONE\_SHOT (*otaku\_info.enums.MediaSubType attribute*), 35
  - otaku\_info module, 36
  - otaku\_info.background module, 4
  - otaku\_info.background.anilist module, 3
  - otaku\_info.background.ln\_releases module, 3

otaku\_info.background.manga\_chapter\_guess module, 4  
otaku\_info.background.mangadex module, 4  
otaku\_info.background.notifications module, 4  
otaku\_info.Config module, 34  
otaku\_info.db module, 12  
otaku\_info.db.LnRelease module, 4  
otaku\_info.db.MangaChapterGuess module, 5  
otaku\_info.db.MediaId module, 6  
otaku\_info.db.MediaItem module, 7  
otaku\_info.db.MediaList module, 8  
otaku\_info.db.MediaListItem module, 8  
otaku\_info.db.MediaNotification module, 9  
otaku\_info.db.MediaUserState module, 10  
otaku\_info.db.ModelMixin module, 11  
otaku\_info.db.NotificationSetting module, 11  
otaku\_info.db.ServiceUsername module, 12  
otaku\_info.enums module, 34  
otaku\_info.external module, 20  
otaku\_info.external.anilist module, 19  
otaku\_info.external.entities module, 19  
otaku\_info.external.entities.AnilistItem module, 13  
otaku\_info.external.entities.AnilistUser module, 14  
otaku\_info.external.entities.AnimeListItem module, 15  
otaku\_info.external.entities.MangadexItem module, 16  
otaku\_info.external.entities.Myanimelist module, 17  
otaku\_info.external.entities.RedditLnRelease module, 18  
otaku\_info.external.mangadex module, 19  
otaku\_info.external.myanimelist module, 19  
otaku\_info.external.reddit module, 20  
otaku\_info.main module, 36  
otaku\_info.mappings module, 36  
otaku\_info.routes module, 21  
otaku\_info.routes.api module, 20  
otaku\_info.routes.api.media\_api module, 20  
otaku\_info.routes.external\_service module, 20  
otaku\_info.routes.ln module, 20  
otaku\_info.routes.media module, 21  
otaku\_info.routes.notifications module, 21  
otaku\_info.routes.updates module, 21  
otaku\_info.template\_extras module, 36  
otaku\_info.test module, 30  
otaku\_info.test.db module, 26  
otaku\_info.test.db.TestApiKey module, 21  
otaku\_info.test.db.TestMangaChapterGuess module, 22  
otaku\_info.test.db.TestMediaId module, 22  
otaku\_info.test.db.TestMediaItem module, 23  
otaku\_info.test.db.TestMediaList module, 23  
otaku\_info.test.db.TestMediaListItem module, 24  
otaku\_info.test.db.TestMediaRelation module, 25  
otaku\_info.test.db.TestMediaUserState module, 25  
otaku\_info.test.db.TestServiceUsername module, 25  
otaku\_info.test.db.TestUser module, 26  
otaku\_info.test.misc module, 28  
otaku\_info.test.misc.TestApiCalls module, 26

otaku\_info.test.misc.TestConfig  
     module, 27  
 otaku\_info.test.misc.TestErrorHandling  
     module, 27  
 otaku\_info.test.misc.TestMappings  
     module, 27  
 otaku\_info.test.misc.TestServer  
     module, 27  
 otaku\_info.test.routes  
     module, 30  
 otaku\_info.test.routes.api  
     module, 28  
 otaku\_info.test.routes.api.TestApiKeyRoute  
     module, 28  
 otaku\_info.test.routes.TestForgotRoute  
     module, 28  
 otaku\_info.test.routes.TestLoginRoute  
     module, 29  
 otaku\_info.test.routes.TestProfileRoute  
     module, 29  
 otaku\_info.test.routes.TestRegisterRoute  
     module, 29  
 otaku\_info.test.routes.TestStaticRoutes  
     module, 30  
 otaku\_info.test.TestFramework  
     module, 30  
 otaku\_info.test.utils  
     module, 30  
 otaku\_info.utils  
     module, 33  
 otaku\_info.utils.dates  
     module, 32  
 otaku\_info.utils.db  
     module, 32  
 otaku\_info.utils.db.convert  
     module, 31  
 otaku\_info.utils.db.DbCache  
     module, 30  
 otaku\_info.utils.db.load  
     module, 32  
 otaku\_info.utils.db.updater  
     module, 32  
 otaku\_info.wrappers  
     module, 34  
 otaku\_info.wrappers.UpdateWrapper  
     module, 33  
 OTHER (*otaku\_info.enums.MediaRelationType* attribute),  
     35  
 OVA (*otaku\_info.enums.MediaSubType* attribute), 35  
 own\_url () (*otaku\_info.db.MediaItem.MediaItem* prop-  
     erty), 7  
**P**  
 PARENT (*otaku\_info.enums.MediaRelationType* at-  
     tribute), 35  
 parse\_chapter () (*otaku\_info.external.entities.MangadexItem.MangadexItem*  
     static method), 16  
 PAUSED (*otaku\_info.enums.ConsumingState* attribute),  
     34  
 physical (*otaku\_info.db.LnRelease.LnRelease* at-  
     tribute), 5  
 PLANNING (*otaku\_info.enums.ConsumingState* at-  
     tribute), 34  
 PREQUEL (*otaku\_info.enums.MediaRelationType* at-  
     tribute), 35  
 profile\_extras () (in *otaku\_info.template\_extras* module), 36  
 progress (*otaku\_info.db.MediaUserState.MediaUserState*  
     attribute), 10  
 publisher (*otaku\_info.db.LnRelease.LnRelease*  
     attribute), 5  
 purchase\_link (*otaku\_info.db.LnRelease.LnRelease*  
     attribute), 5  
**R**  
 RedditLnRelease (class in *otaku\_info.external.entities.RedditLnRelease*),  
     18  
 release\_date () (*otaku\_info.db.LnRelease.LnRelease*  
     property), 5  
 release\_date () (*otaku\_info.external.entities.RedditLnRelease.RedditLnRelease*  
     property), 18  
 release\_date\_string  
     (*otaku\_info.db.LnRelease.LnRelease* attribute),  
     5  
 release\_date\_string ()  
     (*otaku\_info.external.entities.RedditLnRelease.RedditLnRelease*  
     property), 19  
 RELEASING (*otaku\_info.enums.ReleasingState* at-  
     tribute), 36  
 releasing\_state (*otaku\_info.db.MediaItem.MediaItem*  
     attribute), 7  
 ReleasingState (class in *otaku\_info.enums*), 35  
 REPEATING (*otaku\_info.enums.ConsumingState* at-  
     tribute), 34  
 resolve\_media\_subtype ()  
     (*otaku\_info.external.entities.MyanimelistItem.MyanimelistItem*  
     static method), 17  
 resolve\_relation\_type ()  
     (*otaku\_info.external.entities.MyanimelistItem.MyanimelistItem*  
     static method), 18  
 resolve\_releasing\_state ()  
     (*otaku\_info.external.entities.MangadexItem.MangadexItem*  
     static method), 17  
 resolve\_releasing\_state ()  
     (*otaku\_info.external.entities.MyanimelistItem.MyanimelistItem*  
     static method), 18

romaji\_title (*otaku\_info.db.MediaItem.MediaItem attribute*), 7  
 root\_path (in module *otaku\_info*), 36

## S

score (*otaku\_info.db.MediaUserState.MediaUserState attribute*), 10  
 send\_new\_update\_notifications() (in module *otaku\_info.background.notifications*), 4  
 sentry\_dsn (in module *otaku\_info*), 36  
 SEQUEL (*otaku\_info.enums.MediaRelationType attribute*), 35  
 series\_name (*otaku\_info.db.LnRelease.LnRelease attribute*), 5  
 service (*otaku\_info.db.MediaId.MediaId attribute*), 6  
 service (*otaku\_info.db.MediaList.MediaList attribute*), 8  
 service (*otaku\_info.db.ServiceUsername.ServiceUsername attribute*), 12  
 service\_icon() (*otaku\_info.db.MediaId.MediaId property*), 6  
 service\_id (*otaku\_info.db.MediaId.MediaId attribute*), 6  
 service\_url() (*otaku\_info.db.MediaId.MediaId property*), 6  
 ServiceUsername (class in *otaku\_info.db.ServiceUsername*), 12  
 SIDE\_STORY (*otaku\_info.enums.MediaRelationType attribute*), 35  
 SOURCE (*otaku\_info.enums.MediaRelationType attribute*), 35  
 SPECIAL (*otaku\_info.enums.MediaSubType attribute*), 35  
 SPIN\_OFF (*otaku\_info.enums.MediaRelationType attribute*), 35  
 SUMMARY (*otaku\_info.enums.MediaRelationType attribute*), 35

## T

TELEGRAM\_API\_KEY (*otaku\_info.Config.Config attribute*), 34  
 TELEGRAM\_BOT\_CONNECTION (*otaku\_info.Config.Config attribute*), 34  
 test\_404() (*otaku\_info.test.misc.TestErrorHandling.TestErrorHandling method*), 27  
 test\_changing\_password() (*otaku\_info.test.routes.TestProfileRoute.TestProfileRoute method*), 29  
 test\_completeness() (*otaku\_info.test.misc.TestMappings.TestMappings method*), 27  
 test\_confirming() (*otaku\_info.test.routes.TestRegisterRoute.TestRegisterRoute method*), 29

test\_db\_config() (*otaku\_info.test.misc.TestConfig.TestConfig method*), 27  
 test\_equality() (*otaku\_info.test.db.TestApiKey.TestApiKey method*), 21  
 test\_equality() (*otaku\_info.test.db.TestMangaChapterGuess.TestMangaChapterGuess method*), 22  
 test\_equality() (*otaku\_info.test.db.TestMediaId.TestMediaId method*), 22  
 test\_equality() (*otaku\_info.test.db.TestMediaItem.TestMediaItem method*), 23  
 test\_equality() (*otaku\_info.test.db.TestMediaList.TestMediaList method*), 23  
 test\_equality() (*otaku\_info.test.db.TestMediaListItem.TestMediaListItem method*), 24  
 test\_equality() (*otaku\_info.test.db.TestMediaUserState.TestMediaUserState method*), 25  
 test\_equality() (*otaku\_info.test.db.TestServiceUsername.TestServiceUsername method*), 25  
 test\_equality() (*otaku\_info.test.db.TestUser.TestUser method*), 26  
 test\_exception() (*otaku\_info.test.misc.TestErrorHandling.TestErrorHandling method*), 27  
 test\_expiration() (*otaku\_info.test.db.TestApiKey.TestApiKey method*), 21  
 test\_expired\_api\_key() (*otaku\_info.test.misc.TestApiCalls.TestConfig method*), 26  
 test\_flask\_properties() (*otaku\_info.test.db.TestUser.TestUser method*), 26  
 test\_generating\_service\_url() (*otaku\_info.test.db.TestMediaId.TestMediaId method*), 22  
 test\_get\_about() (*otaku\_info.test.routes.TestStaticRoutes.TestStaticRoutes method*), 30  
 test\_get\_index() (*otaku\_info.test.routes.TestStaticRoutes.TestStaticRoutes method*), 30  
 test\_get\_privacy() (*otaku\_info.test.routes.TestStaticRoutes.TestStaticRoutes method*), 30  
 test\_guessing\_latest\_chapter() (*otaku\_info.test.db.TestMangaChapterGuess.TestMangaChapterGuess method*), 22  
 test\_hashing() (*otaku\_info.test.db.TestApiKey.TestApiKey method*), 21  
 test\_hashing() (*otaku\_info.test.db.TestMangaChapterGuess.TestMangaChapterGuess method*), 22  
 test\_hashing() (*otaku\_info.test.db.TestMediaId.TestMediaId method*), 22  
 test\_hashing() (*otaku\_info.test.db.TestMediaItem.TestMediaItem method*), 23  
 test\_hashing() (*otaku\_info.test.db.TestMediaList.TestMediaList method*), 23

test\_hashing() (otaku\_info.test.db.TestMediaListItem.TestMediaListItem method), 24  
 test\_hashing() (otaku\_info.test.db.TestMediaUserState.TestMediaUserState method), 25  
 test\_hashing() (otaku\_info.test.db.TestServiceUsername.TestServiceUsername method), 25  
 test\_hashing() (otaku\_info.test.db.TestUser.TestUser method), 26  
 test\_invalid\_confirm() (otaku\_info.test.routes.TestRegisterRoute.TestRegisterRoute method), 29  
 test\_invalid\_login\_attempts() (otaku\_info.test.routes.TestLoginRoute.TestLoginRoute method), 29  
 test\_invalid\_recaptcha() (otaku\_info.test.routes.TestForgotRoute.TestForgotRoute method), 28  
 test\_invalid\_recaptcha() (otaku\_info.test.routes.TestRegisterRoute.TestRegisterRoute method), 29  
 test\_invalid\_registrations() (otaku\_info.test.routes.TestRegisterRoute.TestRegisterRoute method), 29  
 test\_json\_representation() (otaku\_info.test.db.TestApiKey.TestApiKey method), 21  
 test\_json\_representation() (otaku\_info.test.db.TestMangaChapterGuess.TestMangaChapterGuess method), 22  
 test\_json\_representation() (otaku\_info.test.db.TestMediaId.TestMediaId method), 23  
 test\_json\_representation() (otaku\_info.test.db.TestMediaItem.TestMediaItem method), 23  
 test\_json\_representation() (otaku\_info.test.db.TestMediaListItem.TestMediaListItem method), 24  
 test\_json\_representation() (otaku\_info.test.db.TestMediaUserState.TestMediaUserState method), 25  
 test\_json\_representation() (otaku\_info.test.db.TestServiceUsername.TestServiceUsername method), 25  
 test\_json\_representation() (otaku\_info.test.db.TestUser.TestUser method), 26  
 test\_json\_representation() (otaku\_info.test.db.TestMediaList.TestMediaList method), 24  
 test\_json\_representation() (otaku\_info.test.db.TestMediaListItem.TestMediaListItem method), 24  
 test\_json\_representation() (otaku\_info.test.db.TestMediaUserState.TestMediaUserState method), 28  
 test\_json\_representation() (otaku\_info.test.db.TestServiceUsername.TestServiceUsername method), 25  
 test\_json\_representation() (otaku\_info.test.db.TestUser.TestUser method), 26  
 test\_logging\_in\_and\_out() (otaku\_info.test.routes.TestLoginRoute.TestLoginRoute method), 29  
 test\_non\_base64\_header() (otaku\_info.test.db.TestMangaChapterGuess.TestMangaChapterGuess method), 22



<i>method</i> ), 22	<i>test_unsuccessfully_resetting_password()</i> ( <i>otaku_info.test.routes.TestForgotRoute.TestForgotRoute</i> <i>method</i> ), 28
<i>test_string_representation()</i> ( <i>otaku_info.test.db.TestMediaId.TestMediaId</i> <i>method</i> ), 23	<i>test_unsuccessfully_revoking_api_key()</i> ( <i>otaku_info.test.routes.api.TestApiKeyRoute.TestApiKeyRoute</i> <i>method</i> ), 28
<i>test_string_representation()</i> ( <i>otaku_info.test.db.TestMediaItem.TestMediaItem</i> <i>method</i> ), 23	<i>test_user_delete()</i> ( <i>otaku_info.test.routes.TestProfileRoute.TestProfileRoute</i> <i>method</i> ), 29
<i>test_string_representation()</i> ( <i>otaku_info.test.db.TestMediaList.TestMediaList</i> <i>method</i> ), 24	<i>test_using_non_json_data()</i> ( <i>otaku_info.test.misc.TestApiCalls.TestConfig</i> <i>method</i> ), 26
<i>test_string_representation()</i> ( <i>otaku_info.test.db.TestMediaListItem.TestMediaListItem</i> <i>method</i> ), 24	<i>test_verifying_key()</i> ( <i>otaku_info.test.db.TestApiKey.TestApiKey</i> <i>method</i> ), 22
<i>test_string_representation()</i> ( <i>otaku_info.test.db.TestMediaUserState.TestMediaUserState</i> <i>method</i> ), 25	<i>test_verifying_password()</i> ( <i>otaku_info.test.db.TestUser.TestUser</i> <i>method</i> ),
<i>test_string_representation()</i> ( <i>otaku_info.test.db.TestServiceUsername.TestServiceUsername</i> <i>method</i> ), 25	<i>test_version()</i> ( <i>otaku_info.test.misc.TestConfig.TestConfig</i> <i>method</i> ), 27
<i>test_string_representation()</i> ( <i>otaku_info.test.db.TestUser.TestUser</i> <i>method</i> ),	<i>TestApiKey</i> ( <i>class in otaku_info.test.db.TestApiKey</i> ),
26	21
<i>test_title()</i> ( <i>otaku_info.test.db.TestMediaItem.TestMediaItem</i> <i>method</i> ), 23	<i>TestApiKeyRoute</i> ( <i>class in</i> <i>otaku_info.test.routes.api.TestApiKeyRoute</i> ),
<i>test_unauthorized_call()</i> ( <i>otaku_info.test.misc.TestApiCalls.TestConfig</i> <i>method</i> ), 26	28
<i>test_uniqueness()</i> ( <i>otaku_info.test.db.TestMangaChapterGuess.TestMangaChapterGuess</i> <i>method</i> ), 22	<i>TestConfig</i> ( <i>class in</i> <i>otaku_info.test.misc.TestApiCalls</i> ), 26
<i>test_uniqueness()</i> ( <i>otaku_info.test.db.TestMediaId.TestMediaId</i> <i>method</i> ), 23	<i>TestConfig</i> ( <i>class in otaku_info.test.misc.TestConfig</i> ),
<i>test_uniqueness()</i> ( <i>otaku_info.test.db.TestMediaItem.TestMediaItem</i> <i>method</i> ), 23	<i>TestErrorHandling</i> ( <i>class in</i> <i>otaku_info.test.misc.TestErrorHandling</i> ),
<i>test_uniqueness()</i> ( <i>otaku_info.test.db.TestMediaList.TestMediaList</i> <i>method</i> ), 24	27
<i>test_uniqueness()</i> ( <i>otaku_info.test.db.TestMediaListItem.TestMediaListItem</i> <i>method</i> ), 24	<i>TestForgotRoute</i> ( <i>class in</i> <i>otaku_info.test.routes.TestForgotRoute</i> ), 28
<i>test_uniqueness()</i> ( <i>otaku_info.test.db.TestMediaUserState.TestMediaUserState</i> <i>method</i> ), 25	<i>TestLoginRoute</i> ( <i>class in</i> <i>otaku_info.test.routes.TestLoginRoute</i> ), 29
<i>test_uniqueness()</i> ( <i>otaku_info.test.db.TestServiceUsername.TestServiceUsername</i> <i>method</i> ), 25	<i>TestMangaChapterGuess</i> ( <i>class in</i> <i>otaku_info.test.db.TestMangaChapterGuess</i> ),
<i>test_unsuccessful_password_change()</i> ( <i>otaku_info.test.routes.TestProfileRoute.TestProfileRoute</i> <i>method</i> ), 29	22
<i>test_unsuccessful_user_delete()</i> ( <i>otaku_info.test.routes.TestProfileRoute.TestProfileRoute</i> <i>method</i> ), 29	<i>TestMappings</i> ( <i>class in</i> <i>otaku_info.test.misc.TestMappings</i> ), 27
	<i>TestMediaId</i> ( <i>class in</i> <i>otaku_info.test.db.TestMediaId</i> ), 22
	<i>TestMediaItem</i> ( <i>class in</i> <i>otaku_info.test.db.TestMediaItem</i> ), 23
	<i>TestMediaList</i> ( <i>class in</i> <i>otaku_info.test.db.TestMediaList</i> ), 23
	<i>TestMediaListItem</i> ( <i>class in</i> <i>otaku_info.test.db.TestMediaListItem</i> ), 24
	<i>TestMediaUserState</i> ( <i>class in</i> <i>otaku_info.test.db.TestMediaUserState</i> ), 25
	<i>TestProfileRoute</i> ( <i>class in</i> <i>otaku_info.test.routes.TestProfileRoute</i> ), 29
	<i>TestRegisterRoute</i> ( <i>class in</i>

*otaku\_info.test.routes.TestRegisterRoute*),  
 29  
 TestServer (class in *otaku\_info.test.misc.TestServer*),  
 27  
 TestServiceUsername (class in *otaku\_info.test.db.TestServiceUsername*),  
 25  
 TestStaticRoutes (class in *otaku\_info.test.routes.TestStaticRoutes*), 30  
 TestUser (class in *otaku\_info.test.db.TestUser*), 26  
 title() (*otaku\_info.db.MediaItem.MediaItem* prop-  
 erty), 7  
 TV (*otaku\_info.enums.MediaSubType* attribute), 35  
 TV\_SHORT (*otaku\_info.enums.MediaSubType* attribute),  
 35

## U

UNKNOWN (*otaku\_info.enums.MediaSubType* attribute),  
 35  
 UNKNOWN (*otaku\_info.enums.ReleasingState* attribute),  
 36  
 update() (*otaku\_info.db.LnRelease.LnRelease*  
 method), 5  
 update() (*otaku\_info.db.MangaChapterGuess.MangaChapterGuess*  
 method), 6  
 update() (*otaku\_info.db.MediaId.MediaId* method), 7  
 update() (*otaku\_info.db.MediaItem.MediaItem*  
 method), 8  
 update() (*otaku\_info.db.MediaList.MediaList*  
 method), 8  
 update() (*otaku\_info.db.MediaListItem.MediaListItem*  
 method), 9  
 update() (*otaku\_info.db.MediaNotification.MediaNotification*  
 method), 9  
 update() (*otaku\_info.db.MediaUserState.MediaUserState*  
 method), 10  
 update() (*otaku\_info.db.ModelMixin.ModelMixin*  
 method), 11  
 update() (*otaku\_info.db.NotificationSetting.NotificationSetting*  
 method), 11  
 update() (*otaku\_info.db.ServiceUsername.ServiceUsername*  
 method), 12  
 update\_anilist\_data() (in module  
*otaku\_info.background.anilist*), 3  
 update\_guess() (*otaku\_info.db.MangaChapterGuess.MangaChapterGuess*  
 method), 6  
 update\_item() (*otaku\_info.utils.db.DbCache.DbCache*  
 static method), 31  
 update\_ln\_releases() (in module  
*otaku\_info.background.ln\_releases*), 3  
 update\_manga\_chapter\_guesses() (in module  
*otaku\_info.background.manga\_chapter\_guesses*),  
 4  
 update\_mangadex\_data() (in module  
*otaku\_info.background.mangadex*), 4  
 update\_or\_insert\_item() (in module  
*otaku\_info.utils.db.updater*), 32  
 UpdateWrapper (class in  
*otaku\_info.wrappers.UpdateWrapper*), 33  
 user (*otaku\_info.db.MediaList.MediaList* attribute), 8  
 user (*otaku\_info.db.MediaUserState.MediaUserState*  
 attribute), 10  
 user (*otaku\_info.db.NotificationSetting.NotificationSetting*  
 attribute), 11  
 user (*otaku\_info.db.ServiceUsername.ServiceUsername*  
 attribute), 12  
 user\_id (*otaku\_info.db.MediaList.MediaList* at-  
 tribute), 8  
 user\_id (*otaku\_info.db.MediaUserState.MediaUserState*  
 attribute), 10  
 user\_id (*otaku\_info.db.NotificationSetting.NotificationSetting*  
 attribute), 11  
 user\_id (*otaku\_info.db.ServiceUsername.ServiceUsername*  
 attribute), 12  
 username (*otaku\_info.db.ServiceUsername.ServiceUsername*  
 attribute), 12  
 value (*otaku\_info.db.NotificationSetting.NotificationSetting*  
 attribute), 11  
 volume (*otaku\_info.db.LnRelease.LnRelease* attribute),  
 5  
 volume\_number() (*otaku\_info.db.LnRelease.LnRelease*  
 property), 5  
 volume\_progress (*otaku\_info.db.MediaUserState.MediaUserState*  
 attribute), 10