Themerr-plex

Dec 10, 2023

About

CHAPTER 1

Overview

LizardByte has the full documentation hosted on Read the Docs.

1.1 About

Themerr-plex is a metadata agent plug-in for Plex Media Player. The plug-in adds theme music to your movies.

This plugin contributes to the following metadata agents.

- Plex Movie - *tv.plex.agents.movie*
- Plex Movie (Legacy) - *com.plexapp.agents.imdb*
- The Movie Database - *com.plexapp.agents.themoviedb*
- RetroArcher - *dev.lizardbyte.retroarcher-plex*

1.2 Integrations

1.3 Downloads

CHAPTER 2

Installation

The recommended method for running Themerr-plex is to use the *bundle* in the latest release.

2.1 Bundle

The bundle is cross platform, meaning Linux, macOS, and Windows are supported.

1. Download the `themerr-plex.bundle.zip` from the latest release
2. Extract the contents to your Plex Media Server Plugins directory.

Tip: See How do I find the Plug-Ins folder for information specific to your Plex server install.

2.2 Docker

Docker images are available on Dockerhub and ghcr.io.

See *Docker* for additional information.

2.3 Source

Caution: Installing from source is not recommended most users.

1. Follow the steps in *Build*.
2. Move the compiled `themerr-plex.bundle` to your Plex Media Server Plugins directory.

CHAPTER 3

Docker

3.1 lizardbyte/themerr-plex

This is a docker-mod for plex which adds Themerr-plex to plex as a plugin, to be downloaded/updated during container start.

This image extends the plex image, and is not intended to be created as a separate container.

3.2 Installation

In plex docker arguments, set an environment variable `DOCKER_MODS=lizardbyte/themerr-plex:latest` or `DOCKER_MODS=ghcr.io/lizardbyte/themerr-plex:latest`

If adding multiple mods, enter them in an array separated by `|`, such as `DOCKER_MODS=lizardbyte/themerr-plex:latest|linuxserver/mods:other-plex-mod`

3.3 Supported Architectures

Specifying `lizardbyte/themerr-plex:latest` or `ghcr.io/lizardbyte/themerr-plex:latest` should retrieve the correct image for your architecture.

The architectures supported by this image are:

Architecture	Available
x86-64	
arm64	
armhf	

CHAPTER 4

Usage

Minimal setup is required to use Themerr-plex. In addition to the installation, a couple of settings must be configured.

1. Navigate to the *Plugins* menu within the Plex server settings.
2. Select the gear cog when hovering over the Themerr-plex plugin tile.
3. Set the values of the preferences and save.

   Warning: Plex stores configuration values in the log. If you upload your logs for support, it would be wise to review the data in the log file.

4. For legacy agents and plugins, enable *Themerr-plex* in your agent settings. This is not necessary for the new Plex Movie agent.
5. Refresh Metadata

Note: If a movie's metadata was refreshed and no theme song was added, it is most likely that the movie is not in the database. Please see *contributing/database* for information on how to contribute.

Attention: It may take several minutes after completing a metadata refresh for a theme song to be available.

4.1 Web UI

A web interface is provided by the plugin. Currently the web ui only provides a couple of end points.

4.1.1 / (root)

This endpoint will display a report showing the theme song status for each item in a library supported by Themerr-plex. A supported library is any that has the default agent as one supported by Themerr-plex.

The report provides an easy means to contribute to ThemerrDB by providing *Add/Edit* buttons for items that can be added to ThemerrDB.

4.1.2 /status

An endpoint that provides a JSON response. If a valid response is returned, Themerr-plex is running.

Example Response

```
{
  "message":"Ok",
  "result":"success"
}
```

4.2 Preferences

4.2.1 Plex Movie agent support

Description When enabled, Themerr-plex will add themes to movies using the Plex Movie agent. This is the new agent that is not using the Plex plugin framework, so Themerr-plex cannot contribute to this agent with standard techniques. Instead Themerr-plex will start a websocket server and listen for events from the Plex server. Whenever a movie is added or has it's metadata refreshed, Themerr-plex will attempt to add a theme song to the movie (if the theme song is available in ThemerrDB).

Default `True`

4.2.2 Prefer MP4A AAC Codec

Description Some Plex clients, such as AppleTV, do not support the Opus audio codec for theme songs. This setting will force Themerr to select the MP4A AAC codec over the Opus codec when both are available. If the MP4A AAC codec is not available, the Opus codec will be used and the theme song will not be playable on clients that do not support the Opus codec.

Default `True`

4.2.3 Remove unused theme songs

Description When Themerr-plex uploads a theme song to the Plex server, it will remove any existing theme songs for the same item. With this setting enabled, Themerr-plex can free up space in Plex's metadata directory. This will only remove items that were uploaded by Themerr-plex or via the hidden Plex rest API method, it will not affect local media assets.

Default `True`

4.2.4 Remove unused art

Description When Themerr-plex uploads art to the Plex server, it will remove any existing art for the same item. With this setting enabled, Themerr-plex can free up space in Plex's metadata directory. This will only remove items that are user uploaded, it will not affect items added by metadata agents or local media assets.

Default `False`

4.2.5 Remove unused posters

Description When Themerr-plex uploads posters to the Plex server, it will remove any existing posters for the same item. With this setting enabled, Themerr-plex can free up space in Plex's metadata directory. This will only remove items that are user uploaded, it will not affect items added by metadata agents or local media assets.

Default `False`

4.2.6 Automatically update items

Description When enabled, Themerr-plex will periodically check for changes in ThemerrDB and apply the changes to the items in your Plex Media Server automatically.

Default `True`

4.2.7 Update movie themes during automatic update

Description When enabled, Themerr-plex will update movie themes during automatic updates.

Default `True`

4.2.8 Update collection themes during automatic update

Description When enabled, Themerr-plex will update collection themes during automatic updates.

Default `True`

4.2.9 Update collection metadata for Plex Movie agent

Description When enabled, Themerr-plex will update collection metadata for the Plex Movie agent during automatic updates. Requires `Update collection themes during automatic update` to be enabled.

Default `False`

4.2.10 Update collection metadata for legacy agents

Description When enabled, Themerr-plex will update collection metadata for legacy agents during automatic updates. Themerr-plex must also be enabled in the agent settings. Requires `Update collection themes during automatic update` to be enabled.

Default `True`

4.2.11 Interval for automatic update task

Description The interval (in minutes) to run the automatic update task.

Default `60`

Minimum `15`

4.2.12 Interval for database cache update task

Description The interval (in minutes) to run the database cache update task. This data is used to display the Web UI dashboard.

Default `60`

Minimum `15`

4.2.13 PlexAPI Timeout

Description The timeout (in seconds) when uploading media to the Plex server.

Default `180`

Minimum `1`

4.2.14 Max Retries

Description The number of times to retry uploading theme audio to the Plex server. The time between retries will increase exponentially. The time between is calculated as `2 ^ retry_number`. For example, the first retry will occur after 2 seconds, the second retry will occur after 4 seconds, and the third retry will occur after 8 seconds.

Default `6`

Minimum `0`

4.2.15 Multiprocessing Threads

Description The number of simultaneous themes to upload for libraries using the Plex Movie agent. Does not apply to legacy agents or plugin agents.

Default `3`

Minimum `1`

4.2.16 YouTube Cookies

Description The cookies to use for the requests to YouTube. Should be in Chromium JSON export format. Example exporter.

Default None

4.2.17 Web UI Locale

Description The localization value to use for translations.

Default `en`

4.2.18 Web UI Host Address

Description The host address to bind the Web UI to.

> **Attention:** Changing this value requires a Plex Media Server restart.

Default `0.0.0.0`

4.2.19 Web UI Port

Description The port to bind the Web UI to.

> **Attention:** Changing this value requires a Plex Media Server restart.

Default `9494`

4.2.20 Log all web server messages

Description If set to `True`, all web server messages will be logged. This will include logging requests and status codes when requesting any resource. It is recommended to keep this disabled unless debugging.

> **Attention:** Changing this value requires a Plex Media Server restart.

Default `False`

4.2.21 Migrate from < v0.3.0

Description Prior to v0.3.0, Themerr-plex uploaded themes were locked and there was no way to determine if a theme was supplied by Themerr-plex. Therefore, if you used Themerr-plex prior to v0.3.0, you will need to enable this setting to automatically unlock all existing themes (for agents that Themerr-plex supports). Once the migration has completed, the unlock function will never run again.

If you see many of the `Unknown provider` status in the web UI, it is a good indication that you need to enable this option, unless you have many themes provided by other tools.

Default `False`

CHAPTER 5

Troubleshooting

5.1 Rate Limiting / Videos Not Downloading

By default, YouTube-DL will perform queries to YouTube anonymously. As a result, YouTube may rate limit the requests, or sometimes simply block the content (e.g. for age-restricted content, but not only).

Adding your YouTube credentials (e-mail and password) in Themerr's preference may fix the problem. Hoewever, YouTube also sometimes changes the way its login page works, preventing YouTube-DL from using those credentials.

A workaround is to login in a web browser, and then export your YouTube cookies with a tool such as Get cookies.txt locally. Note that Themerr currently only supports Chromium's JSON export format. In the exporter you use, if prompted, you need to use the "JSON" or "Chrome" format.

You can then paste that value in the "YouTube Cookies" field in the plugin preferences page. On the next media update or scheduled run, the cookies will be used and hopefully videos will start downloading again.

5.2 Plugin Logs

See Plugin Log Files for the plugin log directory.

Plex uses rolling logs. There will be six log files available. The newest log file will be named `dev.lizardbyte.themerr-plex.log`. There will be additional log files with the same name, appended with a *1-5*.

It is best to replicate the issue you are experiencing, then review the latest log file. The information in the log file may seem cryptic. If so it would be best to reach out for support.

Attention: Before uploading logs, it would be wise to review the data in the log file. Plex does not filter the masked settings (e.g. credentials) out of the log file.

5.3 Plex Media Server Logs

If you have a more severe problem, you may need to troubleshoot an issue beyond the plugin itself. See Plex Media Server Logs for more information.

CHAPTER 6

Changelog

CHAPTER 7

Contributing

Read our contribution guide in our organization level docs.

CHAPTER 8

Database

The database of themes is held in our ThemerrDB repository. To contribute to the database, follow the documentation there.

CHAPTER 9

Build

Compiling Themerr-plex is fairly simple; however it is recommended to use Python 2.7 since the Plex framework is using Python 2.7.

9.1 Clone

Ensure git is installed and run the following:

```
git clone https://github.com/lizardbyte/themerr-plex.git themerr-plex.bundle
cd ./themerr-plex.bundle
```

9.2 Setup venv

It is recommended to setup and activate a venv.

9.3 Install Requirements

Install Requirements

```
python -m pip install --upgrade --target=./Contents/Libraries/Shared -r␣
↪requirements.txt --no-warn-script-location
```

Development Requirements

```
python -m pip install -r requirements-dev.txt
```

9.4 Build Plist

```
python ./scripts/build_plist.py
```

9.5 npm dependencies

Install nodejs and npm. Downloads available here.

Install npm dependencies.

```
npm install
```

Move modules directory.

Linux/macOS

```
mv ./node_modules ./Contents/Resources/web
```

Windows

```
move .\node_modules .\Contents\Resources\web
```

9.6 Remote Build

It may be beneficial to build remotely in some cases. This will enable easier building on different operating systems.

1. Fork the project
2. Activate workflows
3. Trigger the *CI* workflow manually
4. Download the artifacts from the workflow run summary

CHAPTER 10

Testing

10.1 Flake8

Themerr-plex uses Flake8 for enforcing consistent code styling. Flake8 is included in the `requirements-dev.txt`.

The config file for flake8 is `.flake8`. This is already included in the root of the repo and should not be modified.

Test with Flake8

```
python -m flake8
```

10.2 Sphinx

Themerr-plex uses Sphinx for documentation building. Sphinx is included in the `requirements-dev.txt`.

Themerr-plex follows numpydoc styling and formatting in docstrings. This will be tested when building the docs. *numpydoc* is included in the `requirements-dev.txt`.

The config file for Sphinx is `docs/source/conf.py`. This is already included in the root of the repo and should not be modified.

Test with Sphinx

```
cd docs
make html
```

Alternatively

```
cd docs
sphinx-build -b html source build
```

Lint with rstcheck

```
rstcheck -r .
```

10.3 pytest

Themerr-plex uses pytest for unit testing. pytest is included in the `requirements-dev.txt`.

No config is required for pytest.

Attention: A locally installed Plex server is required to run some of the tests. The server must be running locally so that the plugin logs can be parsed for exceptions. It is not recommended to run the tests against a production server.

A script is provided that allows you to prepare the Plex server for testing. Use the help argument to see the options.

Bootstrap the Plex server for testing .. code-block:: bash

python scripts/plex-bootstraptest.py –help

Test with pytest

```
python -m pytest
```

Tip: Due to the complexity of setting up the environment for testing, it is recommended to run the tests in GitHub Actions. This will ensure that the tests are run in a clean environment and will not be affected by any local changes.

CHAPTER 11

__init__

Code.**Start**()

Start the plug-in.

This function is called when the plug-in first starts. It can be used to perform extra initialisation tasks such as configuring the environment and setting default attributes. See the archived Plex documentation Predefined functions for more information.

Preferences are validated, then additional threads are started for the web server, queue, plex listener, and scheduled tasks.

Examples

```
>>> Start()
...
```

class Code.**Themerr**

Bases: plexhints.agent_kit.Movies

Class representing the Themerr-plex Movie Agent.

This class defines the metadata agent. See the archived Plex documentation Defining an agent class for more information.

References

name [str] A string defining the name of the agent for display in the GUI.

languages [list] A list of strings defining the languages supported by the agent. These values should be taken from the constants defined in the Locale API.

primary_provider [bool] A boolean value defining whether the agent is a primary metadata provider or not. Primary providers can be selected as the main source of metadata for a particular media type. If an agent is secondary (primary_provider is set to False) it will only be able to contribute to data provided by another primary agent.

fallback_agent [Optional[str]] A string containing the identifier of another agent to use as a fallback. If none of the matches returned by an agent are a close enough match to the given set of hints, this fallback agent will be called to attempt to find a better match.

accepts_from [Optional[list]] A list of strings containing the identifiers of agents that can contribute secondary data to primary data provided by this agent.

contributes_to [Optional[list]] A list of strings containing the identifiers of primary agents that the agent can contribute secondary data to.

Examples

```
>>> Themerr()
...
```

Methods

search:	Search for an item.
update:	Add or update metadata for an item.

static search(*results*, *media*, *lang*, *manual*)
Search for an item.

When the media server needs an agent to perform a search, it calls the agent's `search` method. See the archived Plex documentation Searching for results to provide matches for media for more information.

Parameters

results [SearchResult] An empty container that the developer should populate with potential matches.

media [Media.Movie] An object containing hints to be used when performing the search.

lang [str] A string identifying the user's currently selected language. This will be one of the constants added to the agent's `languages` attribute.

manual [bool] A boolean value identifying whether the search was issued automatically during scanning, or manually by the user (in order to fix an incorrect match).

Returns

Optional[SearchResult] The search result object, if the search was successful.

Examples

```
>>> Themerr().search(results=..., media=..., lang='en', manual=True)
...
```

static update(*metadata*, *media*, *lang*, *force*)
Update metadata for an item.

Once an item has been successfully matched, it is added to the update queue. As the framework processes queued items, it calls the `update` method of the relevant agents. See the archived Plex documentation Adding metadata to media for more information.

Parameters

metadata [object] A pre-initialized metadata object if this is the first time the item is being updated, or the existing metadata object if the item is being refreshed.

media [object] An object containing information about the media hierarchy in the database.

lang [str] A string identifying which language should be used for the metadata. This will be one of the constants defined in the agent's `languages` attribute.

force [bool] A boolean value identifying whether the user forced a full refresh of the metadata. If this argument is `True`, all metadata should be refreshed, regardless of whether it has been populated previously.

Examples

```
>>> Themerr().update(metadata=..., media=..., lang='en', force=True)
...
```

`Code.`**`ValidatePrefs`**`()`

Validate plug-in preferences.

This function is called when the user modifies their preferences. The developer can check the newly provided values to ensure they are correct (e.g. attempting a login to validate a username and password), and optionally return a `MessageContainer` to display any error information to the user. See the archived Plex documentation Predefined functions for more information.

Returns

MessageContainer Success or Error message dependeing on results of validation.

Examples

```
>>> ValidatePrefs()
...
```

`Code.`**`copy_prefs`**`()`

Copy the current preferences to the last preferences.

This function is used to copy the current preferences to the last preferences. This is useful to determine if the preferences have changed.

Examples

```
>>> copy_prefs()
```

CHAPTER 12

general_helper

Code.general_helper.**get_media_upload_path**(*item*, *media_type*)

Get the path to the theme upload directory.

Get the hashed path of the theme upload directory for the item specified by the `item`.

Parameters

item [any] The item to get the theme upload path for.

media_type [str] The media type to get the theme upload path for. Must be one of 'art', 'posters', or 'themes'.

Returns

str The path to the theme upload directory.

Raises

ValueError If the `media_type` is not one of 'art', 'posters', or 'themes'.

Examples

```
>>> get_media_upload_path(item=..., media_type='art')
"...bundle/Uploads/art..."
>>> get_media_upload_path(item=..., media_type='posters')
"...bundle/Uploads/posters..."
>>> get_media_upload_path(item=..., media_type='themes')
"...bundle/Uploads/themes..."
```

Code.general_helper.**get_themerr_json_data**(*item*)

Get the Themerr data for the specified item.

Themerr data is stored as a JSON file in the Themerr data directory, and is used to ensure that we don't unnecessarily re-upload media to the Plex server.

Parameters

item [any] The item to get the Themerr data for.

Returns

dict The Themerr data for the specified item, or empty dict if no Themerr data exists.

Code.general_helper.**get_themerr_json_path**(*item*)

Get the path to the Themerr data file.

Get the path to the Themerr data file for the item specified by the `item`.

Parameters

item [any] The item to get the Themerr data file path for.

Returns

str The path to the Themerr data file.

Examples

```
>>> get_themerr_json_path(item=...)
'.../Plex Media Server/Plug-in Support/Data/dev.lizardbyte.themerr-plex/DataItems/
↪...'
```

Code.general_helper.**get_themerr_settings_hash**()

Get a hash of the current Themerr settings.

Returns

str Hash of the current Themerr settings.

Examples

```
>>> get_themerr_settings_hash()
'...'
```

Code.general_helper.**remove_uploaded_media**(*item*, *media_type*)

Remove themes for the specified item.

Deletes the themes upload directory for the item specified by the `item`.

Parameters

item [any] The item to remove the themes from.

media_type [str] The media type to remove the themes from. Must be one of 'art', 'posters', or 'themes'.

Returns

bool True if the themes were removed successfully, False otherwise.

Examples

```
>>> remove_uploaded_media(item=..., media_type='themes')
...
```

Code.general_helper.**remove_uploaded_media_error_handler**(*func*, *path*, *exc_info*)
Error handler for removing themes.

Handles errors that occur when removing themes using `shutil`.

Parameters

func [any] The function that caused the error.

path [str] The path that caused the error.

exc_info [any] The exception information.

Code.general_helper.**update_themerr_data_file**(*item*, *new_themerr_data*)
Update the Themerr data file for the specified item.

This updates the themerr data file after uploading media to the Plex server.

Parameters

item [any] The item to update the Themerr data file for.

new_themerr_data [dict] The Themerr data to update the Themerr data file with.

CHAPTER 13

lizardbyte_db_helper

Code.lizardbyte_db_helper.**get_igdb_id_from_collection**(*search_query*, *collection_type=None*)

Search for a collection by name.

Match a collection by name against the LizardByte db (clone of IGDB), to get the collection ID.

Parameters

search_query [str] Collection name to search for.

collection_type [Optional[str]] Collection type to search for. Valid values are 'game_collections' and 'game_franchises'. If not provided, will first search for 'game_collections', then 'game_franchises', returning the first match.

Returns

Optional[Tuple[int, str]] Tuple of `id` and `collection_type` if found, otherwise None.

Examples

```
>>> get_igdb_id_from_collection(search_query='James Bond', collection_type='game_
↪collections')
326
>>> get_igdb_id_from_collection(search_query='James Bond', collection_type='game_
↪franchises')
37
```

CHAPTER 14

migration_helper

class Code.migration_helper.**MigrationHelper**

Helper class to perform migrations.

Attributes

migration_status_file [str] The path to the migration status file.

migration_status_file_lock [Lock] The lock for the migration status file.

Methods

_validate_migration_key(key, raise_exception=False)	Validate the given migration key.
get_migration_status(key)	Get the migration status for the given key.
set_migration_status(key)	Update the migration status file.
perform_migration(key)	Perform the migration for the given key, if it has not already been performed.
migrate_locked_themes()	Unlock all locked themes.

LOCKED_THEMES = 'locked_themes'

_validate_migration_key(*self*, *key*, *raise_exception=False*)

Validate the given migration key.

Ensure the given key has a corresponding class attribute and function.

Parameters

key [str] The key to validate.

raise_exception [bool] Whether to raise an exception if the key is invalid.

Returns

bool Whether the key is valid.

Raises

AttributeError If the key is invalid and raise_exception is True.

get_migration_status(*self*, *key*)
Get the migration status for the given key.

Parameters

key [str] The key to get the migration status for.

Returns

Optional[bool] The migration status for the given key, or None if the key is not found.

Examples

```
>>> MigrationHelper().get_migration_status(key=self.LOCKED_THEMES)
True
```

static migrate_locked_themes()
Unlock all locked themes.

Prior to v0.3.0, themes uploaded by Themerr-plex were locked which leads to an issue in v0.3.0 and newer, since Themerr-plex will not update locked themes. Additionally, there was no way to know if a theme was added by Themerr-plex or not until v0.3.0, so this migration will unlock all themes.

perform_migration(*self*, *key*)
Perform the migration for the given key, if it has not already been performed.

Parameters

key [str] The key to perform the migration for.

Examples

```
>>> MigrationHelper().perform_migration(key=MigrationHelper.LOCKED_THEMES)
```

set_migration_status(*self*, *key*)
Update the migration status file.

Parameters

key [str] The key to update in the migration status file.

Examples

```
>>> MigrationHelper().set_migration_status(key=self.LOCKED_THEMES)
```

CHAPTER 15

plex_api_helper

Code.plex_api_helper.**add_media**(*item*, *media_type*, *media_url_id*, *media_file=None*, *media_url=None*)

Apply media to the specified item.

Adds theme song to the item specified by the `rating_key`. If the same theme song is already present, it will be skipped.

Parameters

item [PlexPartialObject] The Plex item to add the theme to.

media_type [str] The type of media to add. Must be one of 'art', 'posters', or 'themes'.

media_url_id [str] The url or id of the media.

media_file [Optional[str]] Full path to media file.

media_url [Optional[str]] URL of media.

Returns

bool True if the media was added successfully or already present, False otherwise.

Examples

```
>>> add_media(item=..., media_type='themes', media_url_id=..., media_url=...)
>>> add_media(item=..., media_type='themes', media_url_url=..., media_file=...)
```

Code.plex_api_helper.**change_lock_status**(*item*, *field*, *lock=False*)

Change the lock status of the specified field.

Parameters

item [PlexPartialObject] The Plex item to unlock the field for.

field [str] The field to unlock.

lock [bool] True to lock the field, False to unlock the field.

Returns

bool True if the lock status matches the requested lock status, False otherwise.

Examples

```
>>> change_lock_status(item=..., field='theme', lock=False)
```

Code.plex_api_helper.**get_database_info**(*item*)

Get the database info for the specified item.

Get the `database_type`, `database`, `agent`, `database_id` which can be used to locate the theme song in ThemerrDB.

Parameters

item [PlexPartialObject] The Plex item to get the database info for.

Returns

Tuple[Optional[str], Optional[str], Optional[str], Optional[str]] The `database_type`, `database`, `agent`, `database_id`.

Examples

```
>>> get_database_info(item=...)
```

Code.plex_api_helper.**get_plex_item**(*rating_key*)

Get any item from the Plex Server.

This function is used to get an item from the Plex Server. It can then be used to get the metadata for the item.

Parameters

rating_key [int] The `rating_key` of the item to get.

Returns

PlexPartialObject The Plex item from the Plex Server.

Examples

```
>>> get_plex_item(rating_key=1)
...
```

Code.plex_api_helper.**plex_listener**()

Listen for events from Plex server.

Send events to `plex_listener_handler` and errors to `Log.Error`.

Examples

```
>>> plex_listener()
...
```

Code.plex_api_helper.**plex_listener_handler**(*data*)
Process events from `plex_listener()`.

Check if we need to add an item to the queue. This is used to automatically add themes to items from the new Plex Movie agent, since metadata agents cannot extend it.

Parameters

data [dict] Data received from the Plex server.

Examples

```
>>> plex_listener_handler(data={'type': 'timeline'})
...
```

Code.plex_api_helper.**process_queue**()
Add items to the queue.

This is an endless loop to add items to the queue.

Examples

```
>>> process_queue()
...
```

Code.plex_api_helper.**scheduled_update**()
Update all items in the Plex Server.

This is used to update all items in the Plex Server. It is called from a scheduled task.

See also:

scheduled_tasks.setup_scheduling The method where the scheduled task is configurerd.

scheduled_tasks.schedule_loop The method that runs the pending scheduled tasks.

Examples

```
>>> scheduled_update()
```

Code.plex_api_helper.**setup_plexapi**()
Create the Plex server object.

It is required to use PlexAPI in order to add theme music to movies, as the built-in methods for metadata agents do not work for movies. This method creates the server object.

Returns

PlexServer The PlexServer object.

Examples

```
>>> setup_plexapi()
...
```

Code.plex_api_helper.**start_queue_threads**()
Start queue threads.

Start the queue threads based on the number of threads set in the preferences.

Examples

```
>>> start_queue_threads()
...
```

Code.plex_api_helper.**update_plex_item**(*rating_key*)
Automated update of Plex item using only the rating key.

Given the rating key, this function will automatically handle collecting the required information to update the theme song, and any other metadata.

Parameters

rating_key [int] The rating key of the item to be updated.

Returns

bool True if the item was updated successfully, False otherwise.

Examples

```
>>> update_plex_item(rating_key=12345)
```

Code.plex_api_helper.**upload_media**(*item*, *method*, *filepath=None*, *url=None*)
Upload media to the specified item.

Uploads art, poster, or theme to the item specified by the `item`.

Parameters

item [PlexPartialObject] The Plex item to upload the theme to.

method [Callable] The method to use to upload the theme.

filepath [Optional[str]] The path to the theme song.

url [Optional[str]] The url to the theme song.

Returns

bool True if the theme was uploaded successfully, False otherwise.

Examples

```
>>> upload_media(item=..., method=item.uploadArt, url=...)
>>> upload_media(item=..., method=item.uploadPoster, url=...)
>>> upload_media(item=..., method=item.uploadTheme, url=...)
...
```

CHAPTER 16

scheduled_tasks

Code.scheduled_tasks.**run_threaded**(*target*, *daemon=None*, *args=()*, ***kwargs*)

Run a function in a thread.

Allows to run a function in a thread, which is useful for long-running tasks, and it allows the main thread to continue.

Parameters

target [Callable] The function to run in a thread.

daemon [Optional[bool]] Whether the thread should be a daemon thread.

args [Iterable] The positional arguments to pass to the function.

kwargs [Mapping[str, Any]] The keyword arguments to pass to the function.

Returns

threading.Thread The thread that the function is running in.

Examples

```
>>> run_threaded(target=Log.Info, daemon=True, args=['Hello, world!'])
"Hello, world!"
```

Code.scheduled_tasks.**schedule_loop**()

Start the schedule loop.

Before the schedule loop is started, all jobs are run once.

Examples

```
>>> schedule_loop()
...
```

Code.scheduled_tasks.**setup_scheduling**()

Sets up the scheduled tasks.

The Tasks setup depend on the preferences set by the user.

See also:

plex_api_helper.scheduled_update Scheduled function to update the themes.

Examples

```
>>> setup_scheduling()
...
```

CHAPTER 17

tmdb_helper

Code.tmdb_helper.**get_tmdb_id_from_collection**(*search_query*)

Search for a collection by name.

Use the builtin Plex tmdb api service to search for a tmdb collection by name.

Parameters

search_query [str] Name of collection to search for.

Returns

Optional[int] Return collection ID if found, otherwise None.

Examples

```
>>> get_tmdb_id_from_collection(search_query='James Bond Collection')
645
>>> get_tmdb_id_from_collection(search_query='James Bond')
645
```

Code.tmdb_helper.**get_tmdb_id_from_imdb_id**(*imdb_id*)

Convert IMDB ID to TMDB ID.

Use the builtin Plex tmdb api service to search for a movie by IMDB ID.

Parameters

imdb_id [str] IMDB ID to convert.

Returns

Optional[int] Return TMDB ID if found, otherwise None.

Examples

```
>>> get_tmdb_id_from_imdb_id(imdb_id='tt1254207')
10378
```

CHAPTER 18

webapp

Code.webapp.**cache_data**()
Cache data for use in the Web UI dashboard.

Because there are many http requests that must be made to gather the data for the dashboard, it can be time-consuming to populate; therefore, this is performed within this caching function, which runs on a schedule. This function will create a json file that can be loaded by other functions.

Code.webapp.**get_locale**()
Get the locale from the config.

Get the locale specified in the config. This does not need to be called as it is done so automatically by *babel*.

Returns

str The locale.

Examples

```
>>> get_locale()
en
```

Code.webapp.**home**()
Serve the webapp home page.

This page serves the Themerr completion report for supported Plex libraries.

Returns

render_template The rendered page.

Notes

The following routes trigger this function.

- /

- */home*

Examples

```
>>> home()
```

Code.webapp.**image**(*img*)

Get image from static/images directory.

Returns

flask.send_from_directory The image.

Notes

The following routes trigger this function.

- */favicon.ico*

Examples

```
>>> image('favicon.ico')
```

Code.webapp.**start_server**()

Start the flask server.

The flask server is started in a separate thread to allow the plugin to continue running.

Returns

bool True if the server is running, otherwise False.

See also:

Core.Start Function that starts the plugin.

stop_server Function that stops the webapp.

Examples

```
>>> start_server()
```

Code.webapp.**status**()

Check the status of Themerr-plex.

This can be used to test if the plugin is still running. It could be used as part of a healthcheck for Docker, and may have many other uses in the future.

Returns

dict A dictionary of the status.

Examples

```
>>> status()
```

Code.webapp.**stop_server**()
Stop the web server.

This method currently does nothing.

Returns

bool True if the server was shutdown, otherwise False.

See also:

start_server Function that starts the webapp.

Examples

```
>>> stop_server()
```

Code.webapp.**translations**()
Serve the translations.

Returns

Response The translations.

Examples

```
>>> translations()
```

CHAPTER 19

youtube_dl_helper

Code.youtube_dl_helper.**nsbool**(*value*)

Format a boolean value for a Netscape cookie jar file.

Parameters

value [bool] The boolean value to format.

Returns

str 'TRUE' or 'FALSE'.

Code.youtube_dl_helper.**process_youtube**(*url*)

Process URL using *youtube_dl*

Parameters

url [str] The URL of the YouTube video.

Returns

Optional[str] The URL of the audio object.

Examples

```
>>> process_youtube(url='https://www.youtube.com/watch?v=dQw4w9WgXcQ')
...
```

Python Module Index

C

Index

Symbols

A

C

G

H

I

L

M

N

P

R

S

T

U

V