Themerr-kodi

ReenigneArcher

May 29, 2024

ABOUT

CHAPTER

ONE

OVERVIEW

LizardByte has the full documentation hosted on Read the Docs.

1.1 About

Themerr-kodi is an add-on for Kodi. The add-on plays theme music while browsing movies and tv shows in your library.

1.2 Integrations

1.3 Downloads

CHAPTER

TWO

INSTALLATION

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

2.1 Zip

The zip is cross platform, meaning all Kodi clients are supported.

1. Download the `service.themerr.zip` from the latest release
2. Move the `service.themerr.zip` to a location your Kodi client can access.
3. Follow the steps in how to install from ZIP file.

2.2 Source

Caution: Installing from source is not recommended most users.

1. Follow the steps in *Build*.
2. Move the compiled `service.themerr.zip` to a location your Kodi client can access.
3. Follow the steps in how to install from ZIP file.

CHAPTER

THREE

USAGE

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

3.1 Preferences

3.1.1 Dev mode

Description
When enabled, Themerr-kodi will use Kodi's notification system to output log messages.

Default
`False`

3.1.2 Theme timeout

Description
The amount of time, in seconds, that Themerr-kodi will wait before playing or stopping a theme.

Default
`3`

CHAPTER

FOUR

TROUBLESHOOTING

4.1 Plugin Fails to Install

Try clearing the contents of the following locations, or restart Kodi:

- .kodi/addons/temp
- .kodi/temp/temp
- .kodi/temp/archive_cache

See common errors for more information.

4.2 Logging

Per Kodi Add-on guidelines, the add-on will only log when the user enables debug logging.

Log messages from the add-on will be prefixed with `Themerr:`.

CHAPTER

FIVE

CHANGELOG

CHAPTER

SIX

CONTRIBUTING

Read our contribution guide in our organization level docs.

CHAPTER

SEVEN

DATABASE

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

CHAPTER

EIGHT

BUILD

Follow the steps below to build the add-on.

8.1 Clone

Ensure git is installed and run the following:

```
git clone --recurse-submodules https://github.com/lizardbyte/themerr-kodi.git
cd ./themerr-kodi
```

8.2 Setup venv

It is recommended to setup and activate a venv.

8.3 Install Requirements

Install Requirements (Optional)

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

Development Requirements (Required)

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

8.4 Build Add-on

```
python -m scripts.build
```

8.5 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

NINE

TESTING

9.1 Flake8

Themerr-kodi 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
```

9.2 Sphinx

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

Themerr-kodi 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 .
```

9.3 pytest

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

No config is required for pytest.

Test with pytest

```
python -m pytest -rxXs --tb=native --verbose --cov=src tests
```

CHAPTER

TEN

ADDITIONAL INFORMATION

10.1 References

10.1.1 Kodi Built-in modules

- Kodistubs
- Built in modules

10.1.2 Kodi References

- Add-on development
- Add-on rules
- JSON-RPC API
- Third party python modules

10.1.3 Similar Add-ons

- service.tvtunes

10.2 Notes

10.2.1 Kodistubs

Kodistubs is a project that provides stubs for the Kodi built-in modules. It makes it very easy to develop Kodi add-ons in an IDE like PyCharm. This is included in the `requirements-dev.txt`.

10.2.2 Python Dependencies

Python dependencies can be added in three different ways.

1. Kodi add-on modules
2. PyPI modules
3. Submodules

The preferred method is to use Kodi add-on modules. Using this method allows the dependency to be included without including extra bloat.

1. Add the dependency to the `addon.yaml` file in the `addon['requires']['import']` section.

If the dependency is not available as a Kodi add-on module, the next preferred method is to use PyPI modules. Using this method allows the dependency to be installed from PyPI when the add-on is built.

1. Add the dependency to the `requirements.txt` file, and hard pin the version. e.g. `my_requirement==1.2.3`

If the dependency is not available as a Kodi add-on module or a PyPI module, the last resort is to use submodules.

1. Add the dependency as a submodule in the `third-party` directory.

   ```
   git submodule add <git_url>
   ```

2. Checkout a stable version of the dependency.

   ```
   git checkout <branch, commit, or tag>
   ```

3. Add the branch, that dependabot should track, to the `.gitmodules` file.

   ```
   [submodule "third-party/<submodule_name>"]
       path = third-party/<submodule_name>
       url = <git_url>
       branch = <branch>
   ```

10.2.3 IDE Configuration

To allow your IDE to find dependencies which are provided by Kodi, you may be able to add the `third-party/repo-scripts/script.module.<module_name>/lib` directory to your IDE's sources list. In PyCharm, you can right click the `lib` directory and select `Mark Directory as -> Sources Root`. In VSCode, you can add the following to your `.vscode/settings.json` file:

```
{
    "python.analysis.extraPaths": [
        "./third-party/repo-scripts/script.module.<module_name>/lib"
    ]
}
```

CHAPTER

ELEVEN

SOURCE CODE

Our source code is documented using the numpydoc standard.

11.1 Source

11.1.1 src.service

Main entry point for the Themerr service.

src.service.**main**()

Main entry point for the Themerr service.

Creates a Themerr instance and starts it.

Examples

```
>>> main()
```

11.1.2 src.themerr.gui

class src.themerr.gui.**Window**(*player_instance=None*)

Bases: object

A class to represent the Kodi window.

This class watches for changes to the selected item in the Kodi window and starts/stops the theme accordingly.

Parameters

player_instance
[Optional[player.Player]] A player instance to use for testing purposes.

Examples

```
>>> window = Window()
>>> window.window_watcher()
...
>>> window = Window(player_instance=player.Player())
>>> window.window_watcher()
```

Attributes

log
[logger.Logger] The logger object.

monitor
[monitor.ThemerrMonitor] The monitor object.

player
[player.Player] The player object.

item_selected_for
[int] The number of seconds the current item has been selected for.

playing_item_not_selected_for
[int] The number of seconds the playing item has not been selected for.

current_selected_item_id
[Optional[int]] The current selected item ID.

last_selected_item_id
[Optional[int]] The last selected item ID.

uuid_mapping
[dict] A mapping of uuids to YouTube URLs. The UUID will be the database type and the database ID, separated by an underscore. e.g. *tmdb_1* This is used to cache the YouTube URLs for faster lookups.

Methods

window_watcher()	The main method that watches for changes to the Kodi window.
pre_checks()	Perform pre-checks before starting/stopping the theme.
process_kodi_id(kodi_id: str)	Process the Kodi ID and return a YouTube URL.
process_movie(kodi_id: int)	Process the Kodi ID and return a dictionary of IDs.
find_youtube_url(kodi_id: str, db_type: str)	Find the YouTube URL from the IDs.
any_true(check: Optional[bool] = None, checks: Optional[Union[List[bool], Set[bool]]] = ())	Determine if the check is True or if any of the checks are True.
is_home()	Determine if the Kodi window is the home screen.
is_movies()	Determine if the Kodi window is a movies screen.
is_movie_set()	Determine if the Kodi window is a movie set screen.
is_tv_shows()	Determine if the Kodi window is a TV shows screen.
is_seasons()	Determine if the Kodi window is a seasons screen.
is_episodes()	Determine if the Kodi window is an episodes screen.

`static any_true`(*check: bool | None = None*, *checks: List[bool] | Set[bool] | None = ()*)

Determine if the check is True or if any of the checks are True.

This method can be used to determine if at least one condition is True out of a list of multiple conditions.

Parameters

check
[Optional[bool]] The check to perform.

checks
[Optional[List[bool]]] The checks to perform.

Returns

bool
True if any of the checks are True, otherwise False.

Examples

```
>>> Window().any_true(checks=[True, False, False])
True
>>> Window().any_true(checks=[False, False, False])
False
>>> Window().any_true(check=True)
True
>>> Window().any_true(check=False)
False
```

find_youtube_url(*kodi_id: str*, *db_type: str*) → str | None

Find YouTube URL from the Dictionary of IDs.

Given a dictionary of IDs, this method will query the Themerr DB to find the YouTube URL.

Parameters

kodi_id
[str] The Kodi ID to process.

db_type
[str] The database type.

Returns

Optional[str]
A YouTube URL if found, otherwise None.

Examples

```
>>> window = Window()
>>> window.find_youtube_url(kodi_id='tmdb_1', db_type='movies')
```

is_episodes() → bool

Check if the Kodi window is an episodes screen.

This method uses `xbmc.getCondVisibility()` and `xbmc.getInfoLabel()` to determine if the Kodi window is an episodes screen.

Returns

bool
True if the Kodi window is an episodes screen, otherwise False.

Examples

```
>>> Window().is_episodes()
```

is_home() → bool

Check if the Kodi window is the home screen.

This method uses `xbmc.getCondVisibility()` to determine if the Kodi window is the home screen.

Returns

bool
True if the Kodi window is the home screen, otherwise False.

Examples

```
>>> Window().is_home()
```

is_movie_set() → bool

Check if the Kodi window is a movie set screen.

This method uses `xbmc.getCondVisibility()` and `xbmc.getInfoLabel()` to determine if the Kodi window is a movie set screen.

Returns

bool
True if the Kodi window is a movie set screen, otherwise False.

Examples

```
>>> Window().is_movie_set()
```

is_movies() → bool

Check if the Kodi window is a movies screen.

This method uses `xbmc.getCondVisibility()` and `xbmc.getInfoLabel()` to determine if the Kodi window is a movies screen.

Returns

bool
True if the Kodi window is a movies screen, otherwise False.

Examples

```
>>> Window().is_movies()
```

is_seasons() → bool

Check if the Kodi window is a seasons screen.

This method uses `xbmc.getCondVisibility()` and `xbmc.getInfoLabel()` to determine if the Kodi window is a seasons screen.

Returns

bool
True if the Kodi window is a seasons screen, otherwise False.

Examples

```
>>> Window().is_seasons()
```

is_tv_shows() → bool

Check if the Kodi window is a TV shows screen.

This method uses `xbmc.getCondVisibility()` and `xbmc.getInfoLabel()` to determine if the Kodi window is a TV shows screen.

Returns

bool
True if the Kodi window is a TV shows screen, otherwise False.

Examples

```
>>> Window().is_tv_shows()
```

pre_checks() → bool

Perform pre-checks before starting/stopping the theme.

A series of checks are performed to determine if the theme should be played.

Returns

bool
True if the theme should be played, otherwise False.

Examples

```
>>> window = Window()
>>> window.pre_checks()
True
```

process_kodi_id(*kodi_id: str*) → str | None

Generate YouTube URL from a given Kodi ID.

This method takes a Kodi ID and returns a YouTube URL.

Parameters

kodi_id
[str] The Kodi ID to process.

Returns

Optional[str]
A YouTube URL if found, otherwise None.

Examples

```
>>> window = Window()
>>> window.process_kodi_id(kodi_id='tmdb_1')
```

window_watcher()

Watch the Kodi window for changes.

This method is the main method that watches for changes to the Kodi window.

Examples

```
>>> window = Window()
>>> window.window_watcher()
```

11.1.3 src.themerr.locale

class src.themerr.locale.**Locale**

Bases: object

Locale class.

This class is used to handle the localization of strings. Currently, it used only to extract strings from the `settings.xml` file.

Examples

```
>>> Locale.settings()
{30001: '...', ...}
```

Methods

settings()	Get the strings used in the `settings.xml` file.

static addon() → dict

Get the strings used in the addon.xml file.

This method uses the `pgettext` function to extract the strings needed for the addon.xml file. Using `pgettext` allows us to add the msgctxt to the po file, which is needed for Kodi to find the correct translation.

Returns

dict
Dictionary of strings used in the addon.xml file.

Examples

```
>>> Locale.addon()
{"addon.extension.description": '...', ...}
```

static settings() → dict

Get the strings used in the `settings.xml` file.

This method uses the `pgettext` function to extract the strings needed for the `settings.xml` file. Using `pgettext` allows us to add the msgctxt to the po file, which is needed for Kodi to find the correct translation.

Returns

dict
Dictionary of strings used in the `settings.xml` file.

Examples

```
>>> Locale.settings()
{30001: '...', ...}
```

11.1.4 src.themerr.logger

class src.themerr.logger.**Logger**

Bases: object

Themerr's logger class.

Creates a new logger to log to the Kodi log.

Examples

```
>>> logger = Logger()
```

Attributes

notifier
[Notifier] The notifier to use to display notifications to the user.

icons
[dict] A dictionary mapping log levels to notification icons.

level_mapper
[dict] A dictionary mapping log levels to strings.

Methods

log(msg: str, level: int = xbmc.LOGDEBUG)	Log a message to the Kodi log.
debug(msg: str)	Log a debug message to the Kodi log.
info(msg: str)	Log an info message to the Kodi log.
warning(msg: str)	Log a warning message to the Kodi log.
error(msg: str)	Log an error message to the Kodi log.
fatal(msg: str)	Log a fatal message to the Kodi log.

debug(*msg: str*)

Log a debug message to the Kodi log.

Passes the message to the log method with the debug log level.

Parameters

msg
[str] The message to log.

Examples

```
>>> logger = Logger()
>>> logger.debug("This is a debug message")
```

error(*msg: str*)

Log an error message to the Kodi log.

Passes the message to the log method with the error log level.

Parameters

msg
[str] The message to log.

Examples

```
>>> logger = Logger()
>>> logger.error("This is an error message")
```

fatal(*msg: str*)

Log a fatal message to the Kodi log.

Passes the message to the log method with the fatal log level.

Parameters

msg
[str] The message to log.

Examples

```
>>> logger = Logger()
>>> logger.fatal("This is a fatal message")
```

info(*msg: str*)

Log an info message to the Kodi log.

Passes the message to the log method with the info log level.

Parameters

msg
[str] The message to log.

Examples

```
>>> logger = Logger()
>>> logger.info("This is an info message")
```

log(*msg: str*, *level: int = 0*)

Log a message to the Kodi log.

This method will log a debug message to the Kodi log. The level parameter will be included in the log message. Additionally, a notification will be displayed to the user if the addon is in development mode.

Parameters

msg
[str] The message to log.

level
[int] The log level to log the message at.

Examples

```
>>> logger = Logger()
>>> logger.log("This is a debug message", xbmc.LOGDEBUG)
```

warning(*msg: str*)

Log a warning message to the Kodi log.

Passes the message to the log method with the warning log level.

Parameters

msg
[str] The message to log.

Examples

```
>>> logger = Logger()
>>> logger.warning("This is a warning message")
```

11.1.5 src.themerr.monitor

class src.themerr.monitor.**ThemerrMonitor**

Bases: Monitor

Kodi's monitor class.

Creates a new monitor to notify addon about changes.

Examples

```
>>> monitor = ThemerrMonitor()
```

Attributes

log
[logging.Logger] The logger of the ThemerrMonitor class.

Methods

abortRequested() -> bool	Check if Kodi is requesting an abort.
onSettingsChanged()	Check if Kodi settings have been modified.

abortRequested() → bool

Check if Kodi is requesting an abort.

Re-definition of the abortRequested method from xbmc.Monitor.

Returns

bool
True if Kodi is requesting an abort, False otherwise.

Examples

```
>>> monitor = ThemerrMonitor()
>>> monitor.abortRequested()
False
```

onSettingsChanged()

Check if Kodi settings have been modified.

This method is automatically called when Kodi settings have been modified.

Examples

```
>>> monitor = ThemerrMonitor()
>>> monitor.onSettingsChanged()
```

11.1.6 src.themerr.notifier

class src.themerr.notifier.**Notifier**(*heading: str | None = 'Themerr'*, *icon: str | None = 'info'*, *time: int | None = 5000*, *sound: bool | None = True*)

Bases: object

A class to show notification dialogs.

A wrapper class for the `xbmcgui.Dialog.notification` method.

Parameters

heading
[Optional[str]] The heading of the notification dialog.

icon
[Optional[str]] The icon of the notification dialog.

time
[Optional[int]] The time to show the notification dialog.

sound
[Optional[bool]] Whether to play a sound when showing the notification dialog.

Examples

```
>>> notifier = Notifier()
```

Attributes

dialog
[xbmcgui.Dialog] The notification dialog.

heading
[Optional[str]] The heading of the notification dialog.

icon
[Optional[str]] The icon of the notification dialog.

time
[Optional[int]] The time to show the notification dialog.

sound
[Optional[bool]] Whether to play a sound when showing the notification dialog.

Methods

notify( )	message: str, heading: Optional[str] = None, icon: Optional[str] = None, time: Optional[int] = None, sound: Optional[bool] = None, Show a notification dialog.

notify(*message: str*, *heading: str | None = None*, *icon: str | None = None*, *time: int | None = None*, *sound: bool | None = None*)

Show a notification dialog.

Use the `xbmcgui.Dialog.notification` method to show a notification dialog.

Parameters

message
[str] The message of the notification dialog.

heading
[Optional[str]] The heading of the notification dialog.

icon
[Optional[str]] The icon of the notification dialog.

time
[Optional[int]] The time to show the notification dialog.

sound
[Optional[bool]] Whether to play a sound when showing the notification dialog.

Examples

```
>>> notifier = Notifier()
>>> notifier.notify("Hello World!")
```

11.1.7 src.themerr.player

class src.themerr.player.**Player**

Bases: Player

Kodi's player class.

Creates a new player to control playback.

Examples

```
>>> player = Player()
```

Attributes

log
[logging.Logger] The logger for this class.

theme_is_playing
[bool] True if a theme is currently playing, False otherwise.

theme_is_playing_for
[int] The number of seconds the theme has been playing for.

theme_playing_kodi_id
[Optional[str]] The Kodi ID of the theme currently playing.

theme_playing_url
[Optional[str]] The URL of the theme currently playing.

Methods

ytdl_extract_url(url: str) -> Optional[str]	Extract the audio URL from a YouTube URL.
play_url(url: str, kodi_id: str, windowed: bool = False)	Play a YouTube URL.
stop()	Stop playback.
reset()	Reset the player.

play_url(*url: str*, *kodi_id: str*, *windowed: bool = False*)

Play a YouTube URL.

Given a user facing YouTube URL, extract the audio URL and play it.

Parameters

url
[str] The url to play.

kodi_id
[str] The Kodi ID of the item.

windowed
[bool] True to play in a window, False otherwise.

Examples

```
>>> player = Player()
>>> player.play_url(url="https://www.youtube.com/watch?v=dQw4w9WgXcQ", kodi_id=
↪'tmdb_1')
```

reset()

Reset the player.

Reset class variables to their default values.

Examples

```
>>> player = Player()
>>> player.reset()
```

stop()

Stop playback.

This function will stop playback and reset the player.

Examples

```
>>> player = Player()
>>> player.stop()
```

11.1.8 src.themerr.plugin

class src.themerr.plugin.**Themerr**

Bases: object

The Themerr class is the main class for the Themerr addon.

This class is responsible for starting and terminating the addon.

Examples

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

Attributes

log
[logger.Logger] The logger instance for the Themerr addon.

monitor
[monitor.ThemerrMonitor] The monitor instance for the Themerr addon.

settings
[settings.Settings] The settings instance for the Themerr addon.

gui
[gui.Window] The gui instance for the Themerr addon.

add_on
[xbmcaddon.Addon] The xbmcaddon.Addon instance for the Themerr addon.

cwd
[str] The current working directory for the Themerr addon.

lib_dir
[str] The lib directory for the Themerr addon.

threads
[list] A list of threads for the Themerr addon.

Methods

start()	Start the Themerr addon.
terminate()	Terminate the Themerr addon.

`start()`

Start the Themerr addon.

The window watcher thread is started, then the addon waits for kodi to stop the addon.

Examples

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

`terminate()`

Terminate the Themerr addon.

The monitor is deleted, then all threads are joined.

Examples

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

11.1.9 src.themerr.settings

`class src.themerr.settings.Settings`

Bases: `object`

Settings class to access addon settings.

This class is used to access addon settings.

Examples

```
>>> addon_settings = Settings()
```

Attributes

addon
[xbmcaddon.Addon] addon instance

Methods

dev_mode()	Get the dev mode setting.
theme_timeout()	Get the theme timeout setting.

dev_mode() → bool

Get the dev mode setting.

Get the dev mode setting from the addon settings.

Returns

bool
The dev mode setting.

Examples

```
>>> addon_settings = Settings()
>>> addon_settings.dev_mode()
False
```

theme_timeout() → int

Get the theme timeout setting.

Get the theme timeout setting from the addon settings.

Returns

int
The theme timeout setting.

Examples

```
>>> addon_settings = Settings()
>>> addon_settings.theme_timeout()
3
```

11.1.10 src.themerr.youtube

src.themerr.youtube.**process_youtube**(*url: str*) → str | None

Get URL using *youtube_dl*.

The function will try to get a playable URL from the YouTube video.

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

S

INDEX

T

W