
Themerr-jellyfin

ReenigneArcher

Apr 19, 2024

ABOUT

1	Overview	1
1.1	About	1
1.2	Integrations	1
1.3	Downloads	1
2	Installation	3
2.1	Repository	3
2.2	Portable	3
2.3	Docker	3
2.4	Source	4
3	Docker	5
3.1	lizardbyte/themerr-jellyfin	5
4	Usage	7
4.1	Enable Themes	7
4.2	Directory Structure	7
4.3	Task Activation	7
4.4	Theme Updates	8
5	Troubleshooting	9
5.1	Jellyfin Server Logs	9
6	Changelog	11
7	Database	13
8	Build	15
8.1	Clone	15
8.2	Setup Python venv	15
8.3	Install Requirements	15
8.4	Compile	15
8.5	Remote Build	16
9	Localization	17
9.1	CrowdIn	17
9.2	Extraction	17
10	Testing	19
10.1	SonarAnalyzer.CSharp	19
10.2	StyleCop.Analyzers	19

10.3	Sphinx	19
10.4	Unit Testing	20
11	Source Code	21
11.1	Source	21

OVERVIEW

LizardByte has the full documentation hosted on [Read the Docs](#).

1.1 About

Themerr-jellyfin is a plugin for Jellyfin that adds theme songs to movies using ThemerrDB.

1.2 Integrations

1.3 Downloads

INSTALLATION

The recommended method for running Themerr-jellyfin is to add the *repository* to Jellyfin.

Tip: See [Jellyfin Plugins](#) for more information about installing plugins.

2.1 Repository

1. In Jellyfin, go to <http://localhost:8096/web/index.html#!/repositories.html>.
2. Add the repository

```
https://app.lizardbyte.dev/jellyfin-plugin-repo/manifest.json
```

3. Go to the Catalog and search for *Themerr*.
4. Select and install the plugin.
5. Restart Jellyfin

2.2 Portable

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

1. Download the `themerr-jellyfin.zip` from the [latest release](#)
2. Extract the contents to your Jellyfin plugins directory.
3. Restart Jellyfin

2.3 Docker

Docker images are available on [Dockerhub](#) and [ghcr.io](#).

See [Docker](#) for additional information.

2.4 Source

Caution: Installing from source is not recommended most users.

1. Follow the steps in *Build*.
2. Extract the generated zip archive to your Jellyfin plugins directory.
3. Restart Jellyfin

3.1 lizardbyte/themerr-jellyfin

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

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

3.1.1 Installation

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

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

3.1.2 Supported Architectures

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

The architectures supported by this image are:

Architecture	Available
x86-64	
arm64	

Minimal setup is required to use Themerr-jellyfin. In addition to the installation, a few settings must be configured.

4.1 Enable Themes

1. Navigate to your user settings page.
2. Select *Display* from the user section.
3. Within the *Library* section, ensure *Theme songs* is enabled.

4.2 Directory Structure

Attention: Jellyfin requires your media to be stored in separate subdirectories, with each movie/show in its own folder. See [Movies](#) or [TV Shows](#) for more information.

4.3 Task Activation

4.3.1 Scheduled

Themerr will run automatically on a schedule. You can configure the schedule in the [configuration page](#).

4.3.2 Manual

To initialize a download task manually, follow these steps:

1. Navigate to [configuration page](#).
2. Select *Update Theme Songs*.

Or alternatively:

1. Navigate to <http://localhost:8096/web/index.html#!/scheduledtasks.html>.
2. Select *Update Theme Songs* under the *Themerr* section.

4.4 Theme Updates

Themerr will only add or update a theme song if the following conditions are met.

- A user supplied `theme.mp3` is not present.
- The theme in ThemerrDB is different from the previously added theme by Themerr.

TROUBLESHOOTING

5.1 Jellyfin Server Logs

Unfortunately, Jellyfin does not separate the logs by plugin. You will need to review the server logs.

The log location varies, depending on your environment. See [Jellyfin Server Logs](#) for more information.

Attention: Before uploading logs, it would be wise to review the data in the log file. Themerr does not have control of the information that is logged by the Jellyfin server.

CHANGELOG

DATABASE

The database of themes is held in our [ThemerrDB](#) repository. To contribute to the database, follow the documentation there.

Compiling Themerr-jellyfin requires the following:

- `git`
- `.net6.0` SDK
- `python 3.x`

8.1 Clone

Ensure `git` is installed and run the following:

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

8.2 Setup Python venv

It is recommended to setup and activate a `venv`.

8.3 Install Requirements

Install Requirements

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

8.4 Compile

```
mkdir -p ./build
python -m jprm plugin build --output ./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

LOCALIZATION

Themerr-jellyfin and related LizardByte projects are being localized into various languages. The default language is *en* (English).

9.1 CrowdIn

The translations occur on [CrowdIn](#). Anyone is free to contribute to localization there.

Translations Basics

- The brand names *LizardByte* and *Themerr* should never be translated.
- Other brand names should never be translated. Examples:
 - Jellyfin

CrowdIn Integration

How does it work?

When a change is made to the source locale file, those strings get pushed to CrowdIn automatically.

When translations are updated on CrowdIn, a push gets made to the *l10n_master* branch and a PR is made. Once PR is merged, all updated translations are part of the project and will be included in the next release.

9.2 Extraction

Themerr-jellyfin uses a custom translation implementation for localizing the html config page. The implementation uses a JSON key-value pair to map the strings to their respective translations.

The following is a simple example of how to use it.

- Add the string to *Locale/en.json*, in English.

```
{  
  "hello": "Hello!"  
}
```

Note: The json keys should be sorted alphabetically. You can use [jsonabc](#) to sort the keys.

- Use the string in the config page.

```
<p data-localize="hello">Hello!</p>
```

Note:

- The *data-localize* attribute should be the same as the key in the JSON file.
- The *innerText* of the element should be the default English string, incase the translations cannot be properly loaded.
- The *data-localize* attribute can be added to any element that supports *innerText*.
- Once the page is loaded, the *innerText* will be replaced with their respective translations.
- If the translation is not found, there will be a fallback to the default English string.

-
- **Use the string in javascript.**

```
const hello = translate("hello");
```

TESTING

10.1 SonarAnalyzer.CSharp

Themerr-jellyfin uses [SonarAnalyzers.CSharp](#) to spot Bugs, Vulnerabilities, and Code Smells in the project. This is run automatically as part of the build process.

The config file for SonarAnalyzers.CSharp is `.editorconfig`.

10.2 StyleCop.Analyzers

Themerr-jellyfin uses [StyleCop.Analyzers](#) to enforce consistent code styling. This is run automatically as part of the build process.

The config file for StyleCop.Analyzers is `.editorconfig`.

10.3 Sphinx

Themerr-jellyfin uses [Sphinx](#) for documentation building. Sphinx, along with other required python dependencies are included in the `./docs/requirements.txt` file. Python is required to build sphinx docs. Installation and setup of python will not be covered here.

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 .
```

10.4 Unit Testing

Themerr-jellyfin uses `xUnit` for unit testing.

Test with xUnit

```
dotnet test /p:CollectCoverage=true /p:CoverletOutputFormat=opencover --logger  
↪ "console;verbosity=detailed"
```

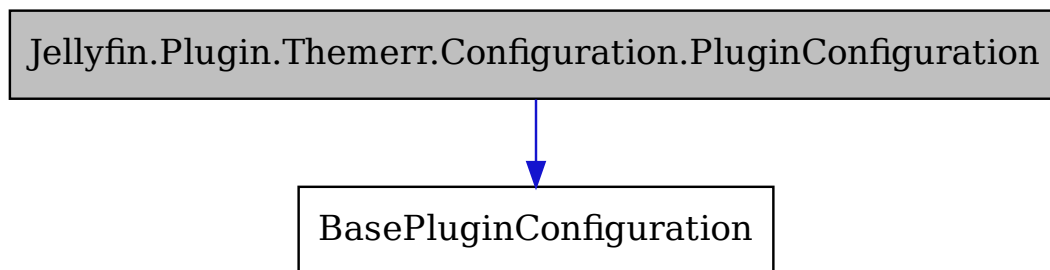

SOURCE CODE

Our source code is documented using the [standard documentation guidelines](#).

11.1 Source

```
class Jellyfin.Plugin.Themerr.Configuration.PluginConfiguration : BasePluginConfiguration
```

Inheritance diagram for Jellyfin::Plugin::Themerr::Configuration::PluginConfiguration:



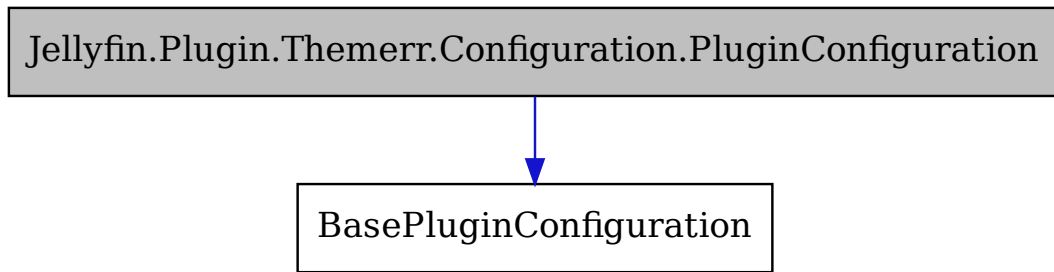
Collaboration diagram for Jellyfin::Plugin::Themerr::Configuration::PluginConfiguration:

Initializes a new instance of the *PluginConfiguration* class.

Public Functions

PluginConfiguration ()

Initializes a new instance of the *PluginConfiguration* class.



Properties

`int? UpdateInterval { get; set; }`

Gets or sets the time between scheduled updates, in minutes.

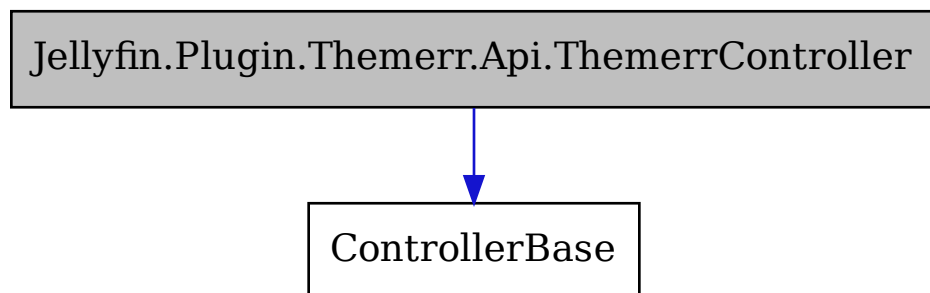
Minimum value of 15.

Private Members

`int _updateInterval`

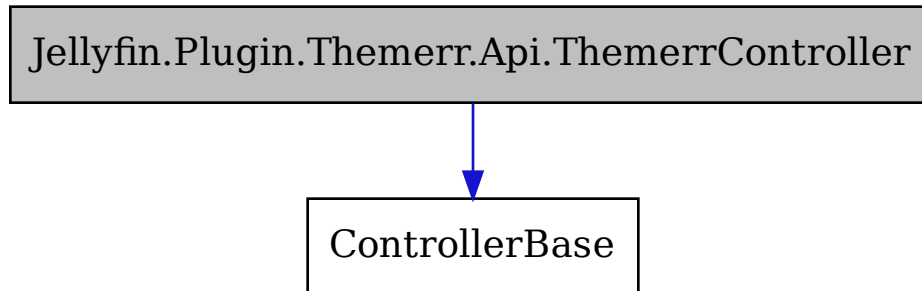
class `Jellyfin.Plugin.Themerr.Api.ThemerrController` : `ControllerBase`

Inheritance diagram for `Jellyfin::Plugin::Themerr::Api::ThemerrController`:



Collaboration diagram for `Jellyfin::Plugin::Themerr::Api::ThemerrController`:

The *Themerr* api controller.



Public Functions

List<string> GetCultureResource (string *culture*)

Get the resources of the given culture.

Param culture

The culture to get the resource for.

Return

A list of file names.

ActionResult GetProgress ()

Get the data required to populate the progress dashboard.

Loop over all *Jellyfin* libraries and supported items, creating a json object with the following structure: { "items": [BaseItems], "media_count": BaseItems.Count, "media_percent_complete": ThemedItems.Count / BaseItems.Count * 100, }

Return

JSON object containing progress data.

ActionResult GetTranslations ()

Get the localization strings from Locale/{selected_locale}.json.

Return

JSON object containing localization strings.

ThemerrController (ILibraryManager *libraryManager*, ILogger<ThemerrController> *logger*,
IServerConfigurationManager *configurationManager*, ILoggerFactory *loggerFactory*)

Initializes a new instance of the *ThemerrController* class.

Param libraryManager

The library manager.

Param logger

The logger.

Param configurationManager

The configuration manager.

Param loggerFactory

The logger factory.

async Task TriggerUpdateRequest ()

Trigger an update from the configuration html page.

A response code of 204 indicates that the download has started successfully.

Return

A NoContentResult indicating success.

Private Members

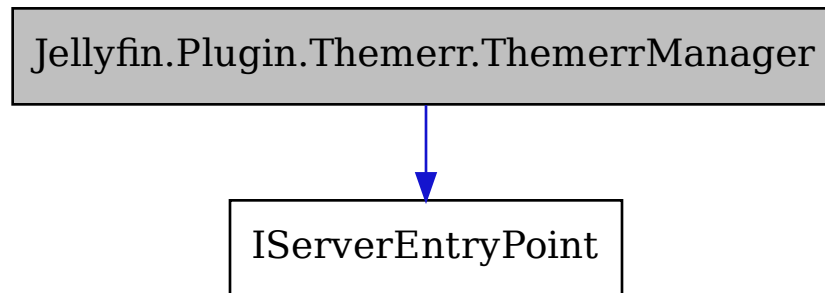
readonly IServerConfigurationManager _configurationManager

readonly ILogger<ThemerrController> _logger

readonly ThemerrManager _themerrManager

class Jellyfin.Plugin.Themerr.ThemerrManager : IServerEntryPoint

Inheritance diagram for Jellyfin::Plugin::Themerr::ThemerrManager:



Collaboration diagram for Jellyfin::Plugin::Themerr::ThemerrManager:

The main entry point for the plugin.

Public Functions

bool ContinueDownload (string *themePath*, string *themerrDataPath*)

Check if the theme song should be downloaded.

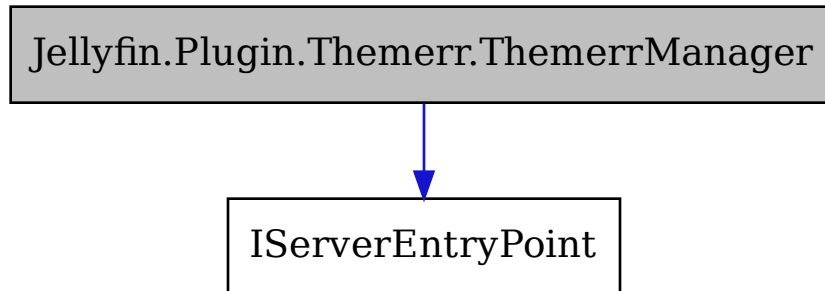
Various checks are performed to determine if the theme song should be downloaded.

Param themePath

The path to the theme song.

Param themerrDataPath

The path to the themerr data file.

**Return**

True to continue with downloaded, false otherwise.

string **CreateThemerrDbLink** (string *tmdbId*, string *dbType*)

Create a link to the themerr database.

Param tmdbId

The tmdb id.

Param dbType

The database type.

Return

The themerr database link.

void **Dispose** ()

Cleanup.

string **GetExistingThemerrDataValue** (string *key*, string *themerrDataPath*)

Get a value from the themerr data file if it exists.

Param key

The key to search for.

Param themerrDataPath

The path to the themerr data file.

Return

The value of the key if it exists, null otherwise.

string **GetIssueUrl** ([BaseItem](#) *item*)

Get ThemerrDB issue url.

This url can be used to easily add/edit theme songs in ThemerrDB.

Param item

The *Jellyfin* media object.

Return

The ThemerrDB issue url.

string **GetMd5Hash** (string *filePath*)

Get the MD5 hash of a file.

Param filePath

The file path.

Return

The MD5 hash of the file.

string **GetThemePath** (BaseItem *item*)

Get the path to the theme song.

Param item

The *Jellyfin* media object.

Return

The path to the theme song.

string **GetThemeProvider** (BaseItem *item*)

Get the theme provider.

Param item

The *Jellyfin* media object.

Return

The theme provider.

string **GetThemerrDataPath** (BaseItem *item*)

Get the path to the themerr data file.

Param item

The *Jellyfin* media object.

Return

The path to the themerr data file.

string **GetTmdbId** (BaseItem *item*)

Get TMDB id from an item.

Param item

The *Jellyfin* media object.

Return

TMDB id.

IEnumerable<BaseItem> **GetTmdbItemsFromLibrary** ()

Get all supported items from the library that have a tmdb id.

Return

List of BaseItem objects.

string **GetYoutubeThemeUrl** (string *themerrDbUrl*, BaseItem *item*)

Get the YouTube theme url from the themerr database.

Param themerrDbUrl

The themerr database url.

Param item

The *Jellyfin* media object.

Return

The YouTube theme url.

void **ProcessItemTheme** (*BaseItem item*)

Download the theme song for a media item if it doesn't already exist.

Param item

The *Jellyfin* media object.

Task **RunAsync** ()

Run the task, asynchronously.

Return

A Task representing the asynchronous operation.

bool **SaveMp3** (string *destination*, string *videoUrl*)

Save a mp3 file from a youtube video url.

Param destination

The destination path.

Param videoUrl

The YouTube video url.

Return

True if the file was saved successfully, false otherwise.

bool **SaveThemerrData** (string *themePath*, string *themerrDataPath*, string *youtubeThemeUrl*)

Save the themerr data file.

Param themePath

The path to the theme song.

Param themerrDataPath

The path to the themerr data file.

Param youtubeThemeUrl

The YouTube theme url.

Return

True if the file was saved successfully, false otherwise.

ThemerrManager (*ILibraryManager libraryManager*, *ILogger<ThemerrManager> logger*)

Initializes a new instance of the *ThemerrManager* class.

Param libraryManager

The library manager.

Param logger

The logger.

Task **UpdateAll** ()

Enumerate through all supported items in the library and downloads their theme songs as required.

Return

A Task representing the asynchronous operation.

bool **WaitForFile** (string *filePath*, int *timeout*)

Wait for file to exist on disk and is not locked by another process.

Param filePath

The file path to check.

Param timeout

The maximum amount of time (in milliseconds) to wait.

Return

True if the file exists and is not locked, false otherwise.

Private Functions

void **OnTimerElapsed** ()

Called when the plugin is loaded.

Private Members

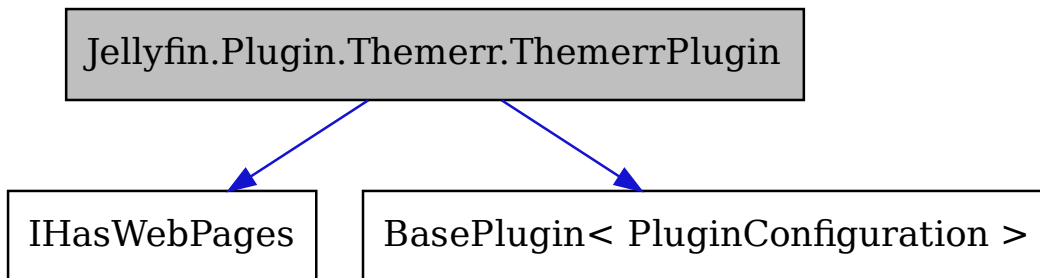
readonly *ILibraryManager* **_libraryManager**

readonly *ILogger<ThemerrManager>* **_logger**

readonly *Timer* **_timer**

class **Jellyfin.Plugin.Themerr.ThemerrPlugin** : *BasePlugin<PluginConfiguration>*, *IHasWebPages*

Inheritance diagram for Jellyfin::Plugin::Themerr::ThemerrPlugin:



Collaboration diagram for Jellyfin::Plugin::Themerr::ThemerrPlugin:

The *Themerr* plugin class.

Public Functions

IEnumerable<PluginPageInfo> **GetPages** ()

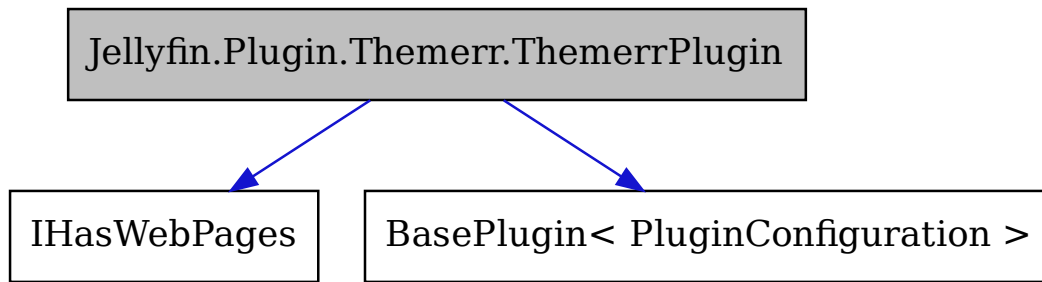
Get the plugin's html config page.

Return

Instance of the *PluginPageInfo* configuration page.

ThemerrPlugin (*IApplicationPaths* *applicationPaths*, *IXmlSerializer* *xmlSerializer*)

Initializes a new instance of the *ThemerrPlugin* class.

**Param applicationPaths**

Instance of the IApplicationPaths interface.

Param xmlSerializer

Instance of the IXmlSerializer interface.

Properties

override string **Description** { **get**; **set**; }

Gets the description of the plugin.

override Guid **Id** { **get**; **set**; }

Gets the plugin instance id.

ThemerrPlugin **Instance** { **get**; **set**; }

Gets the plugin instance.

override string **Name** { **get**; **set**; }

Gets the name of the plugin.

Private Members

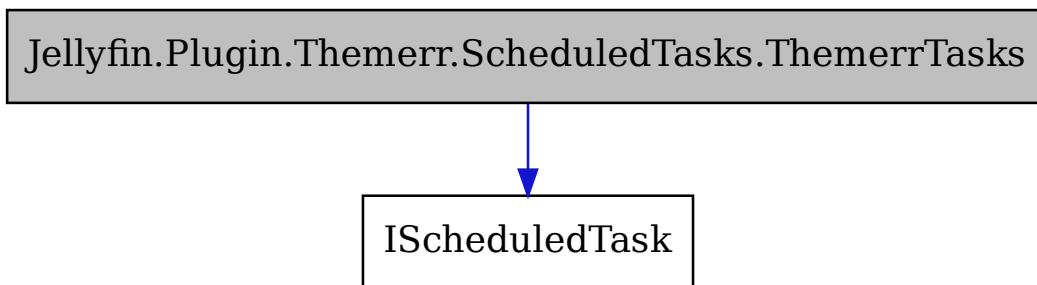
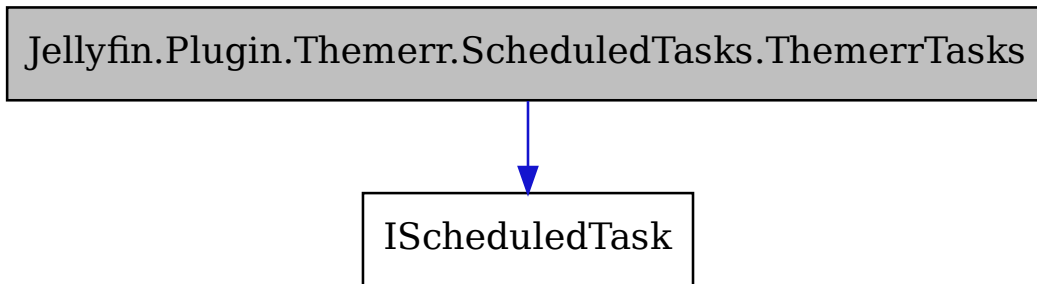
readonly Guid **_id** = new Guid("84b59a39-bde4-42f4-adbd-c39882cbb772")

class Jellyfin.Plugin.Themerr.ScheduledTasks.ThemerrTasks : IScheduledTask

Inheritance diagram for Jellyfin::Plugin::Themerr::ScheduledTasks::ThemerrTasks:

Collaboration diagram for Jellyfin::Plugin::Themerr::ScheduledTasks::ThemerrTasks:

The *Themerr* scheduled task.



Public Functions

async Task ExecuteAsync (IProgress<double> *progress*, CancellationToken *cancellationToken*)

Execute the task, asynchronously.

Param progress

The progress reporter.

Param cancellationToken

The cancellation token.

Return

A Task representing the asynchronous operation.

IEnumerable<TaskTriggerInfo> GetDefaultTriggers ()

Gets the default triggers.

Return

A list of TaskTriggerInfo.

ThemerrTasks (ILibraryManager *libraryManager*, ILogger<ThemerrTasks> *logger*,
ILoggerFactory *loggerFactory*)

Initializes a new instance of the *ThemerrTasks* class.

Param libraryManager

The library manager.

Param logger

The logger.

Param loggerFactory

The logger factory.

Properties

string **Category** { **get**; **set**; }

Gets the category of the task.

string **Description** { **get**; **set**; }

Gets the description of the task.

string **Key** { **get**; **set**; }

Gets the key of the task.

string **Name** { **get**; **set**; }

Gets the name of the task.

Private Members

readonly ILogger<ThemerrTasks> _logger

readonly ThemerrManager _themerrManager

namespace Jellyfin

namespace Jellyfin.Data.Enums

namespace Jellyfin.Plugin

namespace Jellyfin.Plugin.Themerr

namespace Jellyfin.Plugin.Themerr.Api

namespace Jellyfin.Plugin.Themerr.Configuration

namespace Jellyfin.Plugin.Themerr.ScheduledTasks

namespace MediaBrowser.Common.Configuration

namespace MediaBrowser.Common.Plugins

namespace MediaBrowser.Controller.Configuration

namespace MediaBrowser.Controller.Entities

namespace MediaBrowser.Controller.Entities.Movies

namespace MediaBrowser.Controller.Entities.TV

namespace MediaBrowser.Controller.Library

namespace MediaBrowser.Controller.Plugins

namespace MediaBrowser.Model.Entities

namespace MediaBrowser.Model.Plugins

namespace MediaBrowser.Model.Serialization

`namespace MediaBrowser.Model.Tasks`

`namespace Microsoft.AspNetCore.Authorization`

`namespace Microsoft.AspNetCore.Http`

`namespace Microsoft.AspNetCore.Mvc`

`namespace Microsoft.Extensions.Logging`

`namespace Newtonsoft.Json`

`namespace System`

`namespace System.Collections`

`namespace System.Collections.Generic`

`namespace System.IO`

`namespace System.Linq`

`namespace System.Net.Http`

`namespace System.Net.Mime`

`namespace System.Reflection`

`namespace System.Threading`

`namespace System.Threading.Tasks`

`namespace YoutubeExplode`

`namespace YoutubeExplode.Videos.Streams`

file `ThemerrController.cs`

file `PluginConfiguration.cs`

file `ThemerrTasks.cs`

file **ThemerrManager.cs**

file **ThemerrPlugin.cs**

dir **Jellyfin.Plugin.Themerr/Api**

dir **Jellyfin.Plugin.Themerr/Configuration**

dir **Jellyfin.Plugin.Themerr**

dir **Jellyfin.Plugin.Themerr/ScheduledTasks**