PlexyGlass

May 29, 2024

About

CHAPTER 1

Overview

LizardByte has the full documentation hosted on Read the Docs.

1.1 About

PlexyGlass is a Services plug-in for Plex Media Server. The plug-in currently provides a YouTube URL Service. Additional services may be added in the future.

1.2 Integrations

1.3 Downloads

CHAPTER 2

Installation

The recommended method for running PlexyGlass 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 `plexyglass.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 `PlexyGlass.bundle` to your Plex Media Server Plugins directory.

CHAPTER 3

Docker

3.1 lizardbyte/plexyglass

This is a docker-mod for plex which adds PlexyGlass 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/plexyglass:latest` or `DOCKER_MODS=ghcr.io/lizardbyte/plexyglass:latest`

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

3.3 Supported Architectures

Specifying `lizardbyte/plexyglass:latest` or `ghcr.io/lizardbyte/plexyglass: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

4.1 End Users

Minimal setup is required to use PlexyGlass. In addition to the installation, a couple of settings may be configured.

1. Navigate to the *Plugins* menu within the Plex server settings.
2. Select the gear cog when hovering over the PlexyGlass 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.

4.2 Developers

This section is intended for developers utilizing the plugin to support URL services or the like.

It is very easy to use the URL service in your metadata agent. Below is an example.

```
video_title='Rick Astley - Never Gonna Give You Up (Official Music Video)'
video_url='https://www.youtube.com/watch?v=dQw4w9WgXcQ'

metadata.extras.add(OtherObject(title=video_title, url=video_url))
```

You can pass in many other parameters if you'd like, but they are all optional except the url. Below is a bare minimal example.

```
video_url='https://www.youtube.com/watch?v=dQw4w9WgXcQ'

metadata.extras.add(OtherObject(url=video_url))
```

Tip: For help with metadata agent or general plug-in development, check out our plexhints python library.

CHAPTER 5

Changelog

CHAPTER 6

Contributing

Read our contribution guide in our organization level docs.

CHAPTER 7

Build

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

7.1 Clone

Ensure git is installed and run the following:

```
git clone --recurse-submodules https://github.com/lizardbyte/plexyglass.git␣
↪plexyglass.bundle
cd ./plexyglass.bundle
```

7.2 Setup venv

It is recommended to setup and activate a venv.

7.3 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
```

7.4 Build Plist

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

7.5 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 8

Testing

8.1 Flake8

PlexyGlass uses Flake8 for enforcing consistent code styling. Flake 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
```

8.2 Sphinx

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

PlexyGlass 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
```

8.3 pytest

PlexyGlass 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
```

CHAPTER 9

__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.

Examples

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

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()
...
```

CHAPTER 10

YouTube

YouTube.**MediaObjectsForURL**(*url*)

Build the Plex media objects for a given URL.

Parameters

url [str] The url to build media objects for.

Returns

Optional[list] A list of Plex media objects.

Examples

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

YouTube.**MetadataObjectForURL**(*url*)

Get YouTube metadata for a given URL.

Parameters

url [str] The url to get metadata for.

Returns

Optional[VideoClipObject] The Plex video clip object.

Examples

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

YouTube.**NormalizeURL**(*url*)

Get the video webpage url from *youtube-dl*.

Parameters

url [str] A string representation of url as provided by the Plex plugin.

Returns

Optional[str] The video webpage url. If no video webpage is found then `None` is returned.

Examples

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

YouTube.**extract_youtube_data**(*url*)

Extract YouTube data from a given URL.

Parameters

url [str] The video to extract data from.

Returns

Optional[dict] A dictionary containing the video's data.

Examples

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

Python Module Index

c

y

Index