plexhints

May 29, 2024

About

CHAPTER 1

Overview

LizardByte has the full documentation hosted on Read the Docs.

1.1 About

Plexhints is a set of tools to aid in the development of plugins for Plex Media Server. It is not a framework, but rather a set of tools that can be used to make your life easier.

1.2 Features

- Python library providing type hints to aid in the development of Plex plugins. Get rid of all those IDE warnings and errors!
- A GitHub Action that will install and bootstrap a Plex Media Server in a CI environment. The action can install plugins and setup dummy libraries. Additionally the Plex token is provided as an output. This is useful for testing your plugin or other Plex project in a CI environment.

1.3 Integrations

CHAPTER 2

Installation

2.1 Python Package

This library is available on PyPI as `plexhints`. It can be installed in several ways.

PyPI

```
python -m pip install plexhints
```

git

```
python -m pip install git+https://github.com/lizardbyte/plexhints.git@dist
↪#egg=plexhints
```

github archive

```
python -m pip install https://github.com/lizardbyte/plexhints/archive/dist.zip
↪#egg=plexhints
```

CHAPTER 3

Usage

Plexhints can be used by just importing the `plexhints` module and running a couple of functions. After doing so you can use plexhints in your IDE and run most of your code outside of Plex. This is useful for debugging and testing.

3.1 Main Entry Point

Place this at the top of your `Contents/Code/__init__.py` file. It is important to only import these when running outside of Plex.

```
# plex debugging
try:
    import plexhints  # noqa: F401
except ImportError:
    pass
else:  # the code is running outside of Plex
    from plexhints import plexhints_setup, update_sys_path
    plexhints_setup()  # reads the plugin plist file and determine if plexhints␣
↪should use elevated policy or not
    update_sys_path()  # when running outside plex, append the path
```

3.2 Submodules

In files other than the main `__init__.py` file, you can simply import the `plexhints` module and use it as shown.

```
# plex debugging
try:
    import plexhints  # noqa: F401
except ImportError:
    pass
else:  # the code is running outside of Plex
    from plexhints.log_kit import Log
```

3.3 Available Imports

```
from plexhints.agent_kit import Agent, Media  # agent kit
from plexhints.core_kit import Core  # core kit
from plexhints.decorator_kit import handler, indirect, route  # decorator kit
from plexhints.exception_kit import Ex  # exception kit
from plexhints.locale_kit import Locale  # locale kit
from plexhints.log_kit import Log  # log kit
from plexhints.model_kit import Movie, VideoClip, VideoClipObject  # model kit
from plexhints.network_kit import HTTP  # network kit
from plexhints.object_kit import Callback, IndirectResponse, MediaObject,␣
↪MessageContainer, MetadataItem, \
    MetadataSearchResult, PartObject, SearchResult  # object kit
from plexhints.parse_kit import HTML, JSON, Plist, RSS, XML, YAML  # parse kit
from plexhints.prefs_kit import Prefs  # prefs kit
from plexhints.proxy_kit import Proxy  # proxy kit
from plexhints.resource_kit import Resource  # resource kit
from plexhints.shortcut_kit import L, E, D, R, S  # shortcut kit
from plexhints.util_kit import String, Util  # util kit

from plexhints.constant_kit import CACHE_1MINUTE, CACHE_1HOUR, CACHE_1DAY, CACHE_
↪1WEEK, CACHE_1MONTH  # constant kit
from plexhints.constant_kit import ClientPlatforms, Protocols, OldProtocols,␣
↪ServerPlatforms, ViewTypes, \
    SummaryTextTypes, AudioCodecs, VideoCodecs, Containers, ContainerContents, \
    StreamTypes  # constant kit, more commonly used in URL services

# extra objects
from plexhints.extras_kit import BehindTheScenesObject, \
    ConcertVideoObject, \
    DeletedSceneObject, \
    FeaturetteObject, \
    InterviewObject, \
    LiveMusicVideoObject, \
    LyricMusicVideoObject, \
    MusicVideoObject, \
    OtherObject, \
    SceneOrSampleObject, \
    ShortObject, \
    TrailerObject
```

CHAPTER 4

GitHub Action

A GitHub Action is provided to automate installation and preparation of a Plex Media Server in a CI/CD pipeline.

The action does the following:

1. Installs the latest Plex Media Server for the target platform.
2. Installs any specified plugins to the Plex Media Server.
3. Collects the Plex Media Server authentication token.
4. Provides useful output variables for use in subsequent steps.

4.1 Bootstrap Plex server

4.1.1 Inputs

Name	Description	Default	Required
accept_eula	Accept Plex's EULA.	false	false
additional_requests	Space separated list of additional requests to send to the server. The type of request should be at the beginning of the endpoint, followed by a |. If no | is found the default request type of *PUT* will be used. The requests are sent before the library sections are created. You can use this to enable third party metadata agents, as an example. e.g. *put|/system/agents/com.plexapp.agents.imdb/config/1?order=com.plexapp.agents.imdb%2C<my_movie_agent>*	""	false
bootstrap_timeout	Timeout for each step of bootstrap, in seconds.	540	false
docker_tag	Docker image to install. Only used when use_docker is true.	latest	false
expose_plex_data	When using docker, expose the Plex Media Server application data files to the remainder of your workflow at ${{ github.workspace }}/plex.	false	false
language	Language to set inside Plex.	en-US UTF-8	false
plugin_bundles	Space separated list of plugin bundles to install. Provide the relative or absolute path to the bundle.	""	false
timezone	Timezone to set inside Plex.	UTC	false
use_docker	Use Docker to run Plex Media Server. This is only supported on Linux.	false	false
without_movies	Do not create a Movies library (new agent).	false	false
without_movies_imdb	Do not create a Movies library (IMDB agent).	false	false
without_movies_tmdb	Do not create a Movies library (TMDB agent).	false	false
without_music	Do not create a Music library.	false	false
without_photos	Do not create a Photos library.	false	false
without_shows	Do not create a TV Shows library.	false	false

4.1.2 Outputs

Name	Description
PLEXTOKEN	The Plex Media Server authentication token.
PLEX_APP_DATA_PATH	The path to the Plex Media Server application data.
PLEX_PLUGIN_LOG_PATH	The path to the Plex Media Server plugin logs.
PLEX_PLUGIN_PATH	The path to the Plex Media Server plugins.
PLEX_SERVER_BASEURL	The base URL of the Plex Media Server.

4.2 Examples

4.2.1 Basic usage

```
- name: Bootstrap Plex server
  id: bootstrap
  uses: LizardByte/plexhints@latest
```

4.2.2 Install plugins

```
- name: Bootstrap Plex server
  id: bootstrap
  uses: LizardByte/plexhints@latest
  with:
    plugin_bundles_to_install: >-
      MyAwesomePlexPlugin.bundle
      AnotherAwesomePlexPlugin.bundle
```

4.2.3 Disable libraries

```
- name: Bootstrap Plex server
  id: bootstrap
  uses: LizardByte/plexhints@latest
  with:
    without_movies: true
    without_movies_imdb: true
    without_movies_tmdb: true
    without_shows: true
    without_music: true
    without_photos: true
```

4.2.4 Use Docker (Linux only)

```
- name: Bootstrap Plex server
  id: bootstrap
  uses: LizardByte/plexhints@latest
  with:
    use_docker: true
```

4.2.5 Get Outputs

```
- name: Another Step
  env:
    PLEXAPI_AUTH_SERVER_BASEURL: ${{ steps.bootstrap.outputs.PLEX_SERVER_BASEURL }}
    PLEXAPI_AUTH_SERVER_TOKEN: ${{ steps.bootstrap.outputs.PLEXTOKEN }}
    PLEXTOKEN: ${{ steps.bootstrap.outputs.PLEXTOKEN }}
    PLEX_APP_DATA_PATH: ${{ steps.bootstrap.outputs.PLEX_APP_DATA_PATH }}
    PLEX_PLUGIN_LOG_PATH: ${{ steps.bootstrap.outputs.PLEX_PLUGIN_LOG_PATH }}
    PLEX_PLUGIN_PATH: ${{ steps.bootstrap.outputs.PLEX_PLUGIN_PATH }}
```

4.2.6 Complete Example

For a complete example, see our CI.yml.

CHAPTER 5

Changelog

CHAPTER 6

Plex Plugin-in Framework documentation

6.1 References

6.1.1 Plex Plugin-in Framework

A mirror of Plex's Plugin-in Framework is included in our GitHub repository as a submodule, which can be found here or upstream.

6.1.2 Developer Documentation

An archive of the original developer documentation can be found here.

Todo: Add original documentation here.

CHAPTER 7

Contributing

Read our contribution guide in our organization level docs.

CHAPTER 8

Build Plugin

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

8.1 Clone

Ensure git is installed and run the following:

```
git clone https://github.com/lizardbyte/plexhints.git plexhints.bundle
cd ./plexhints.bundle
```

8.2 Setup venv

It is recommended to setup and activate a venv.

8.3 Install Requirements

Install Plexhints

```
python -m pip install -e .
```

Development Requirements

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

8.4 Build Plist

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

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 9

Testing

9.1 Flake8

Plexhints 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

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

Plexhints 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

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

No config file is required for pytest, but pytest relies on a rather specific environment. Plex Media Server must be installed as well as the following environment variables being set.

- `PLEX_PLUGIN_LOG_PATH` - See Plex Plugin Logs
- `PLEXAPI_AUTH_SERVER_BASEURL` - Normally `http://127.0.0.1:32400`

Additionally, the `plexhints.bundle` must be installed in the Plex Media Server plugins directory.

Linux

```
mkdir -p ./plexhints.bundle/Contents
cp -r ./Contents/. ./plexhints.bundle/Contents
```

macOS

```
mkdir -p ./plexhints.bundle/Contents
cp -r ./Contents/. ./plexhints.bundle/Contents
```

Windows

```
mkdir "plexhints.bundle"
xcopy /E /I "Contents" "plexhints.bundle\Contents"
```

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

```
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 10

__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.

Returns

True Always returns `True`.

Examples

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

Python Module Index

C

Index

C

S