pytag Documentation

Release 0.1.5

José Luis Lafuente

December 10, 2013

Contents

pytag is a Python 3 library to handle audio metadata.

Contents:

CHAPTER 1

Introduction

1.1 Requisites

pytag requires Python >= 3.3 and filemagic.

1.2 Installation

```
pip install pytag
```

1.3 Basic usage

```
>>> from pytag import Audio
>>> audio = Audio('/path/to/file.ogg')
>>> audio.get_tags()
PytagDict({'album': 'a cool album'})

>>> audio.write_tags({'album': 'a new name'})
>>> audio.get_tags()
PytagDict({'album': 'a new name'})
```

Note: The returned object (`PytagDict`) extends the Python `dict`, and can be used as a `dict`

For more information about PytagDict see: `pytag.structures.PytagDict`

Is also possible access to the tags/comments as an attribute of the Audio object:

```
>>> from pytag import AudioReader
>>> audio = Audio('/path/to/file.ogg')
>>> audio.album
'a cool album'
```

Take a look to the common interface to see all the valid tags/comments values *Common interface*

CHAPTER 2

Usage

2.1 Common interface

pytag defines a common list of tags (or comments) for all the supported audio formats. This tags are defined at `pytag.constants.FIELD_NAMES` and they are:

- title
- artist
- album
- date
- tracknumber
- organization
- genre
- performer

Using the common interface, doesn't matter if we use want to read from a mp3 or from a ogg vorbis file. If an audio file contains other tags, they are ignored.

There also one extra field in an Audio object, *mimetype*, which contains the file mimetype

Reading metadata from multiple audio files:

```
from pytag import AudioReader, FormatNotSupportedError

files = ['audio.ogg', 'audio.mp3', 'image.png']

try:
    for file in files:
        audio = AudioReader(file)
        print (audio.get_tags())
except FormatNotSupportedError:
    print('Process other file...')
```

Note: If `audio.ogg` has a tag called `band`, this is ignored. If you want all the tags, use the Ogg vorbis interface. See: *Vorbis comments*

Writing metadata to an audio file:

```
from pytag import Audio

audio = Audio('music.ogg')
audio.write_tags({'album': 'cool', 'year': '2000'})
```

Note: Here only tag `album` is saved, `year` is ignored.

`pytag.Audio` extends `pytag.AudioReader`, with `pytag.Audio` is also possible read the tags. Class `pytag.AudioReader` is provided just to avoid write some metadata by mistake.

2.2 Vorbis comments

Using Vorbis comments is possible to save any metadata.

Writing and reading random tags:

```
>>> from pytag.format import OggVorbis
>>> vorbis = OggVorbis('music.ogg')
>>> vorbis.write_tags({'foo': 'bar'})
>>> vorbis.get_tags()
{'foo': 'bar'}
```

Note: Like `pytag.Audio` has a `pytag.AudioReader` only for reading, `pytag.formats.OggVorbis` also has a `pytag.formats.OggVorbisReader` which only is allow to read the comments.

2.3 Mp3 tags

Mp3 files uses ID3 to save the metadata. This format defines a list of codes for the valid tags. See Wikipedia ID3v2 Frames List

As this list is huge and many times confusing, I recommend use only the common interface to read/write Mp3 tags.

CHAPTER 3

Supported formats

3.1 Ogg based

	Extension	Read	Write
Ogg Vorbis	.ogg	Yes	Yes

3.2 Mp3

	Read	Write
ID3v1.1	Yes	No
ID3v2.2	Yes	No
ID3v2.3	Yes	No
ID3v2.4	Yes	Yes

Note: Anyway, is possible to save the metadata in any mp3 file, the tags are just replaced with the last ID3 version (2.4 currently)

CHAPTER 4

Library reference

4.1 pytag

class pytag.**AudioReader**(*path*)
High level interface for pytag. Creates a new object if the audio format is supported, or returns a pytag.FormatNotSupportedError if not.

class pytag.**Audio**(*path*)
Extends pytag.AudioReader and adds a write_tags method.

class pytag.**FormatNotSupportedError**

4.2 Codecs

class pytag.codecs.**VorbisComment**
Base class to read/write vorbis comments as defined at: http://www.xiph.org/vorbis/doc/Vorbis_I_spec.html

process_comments(*self*, *packet*)
Reads the comments.

Parameters **packet** (pytag.containers.PacketReader) – Object to read from, has a read method.

Returns A dict-like object with all the comments.

Return type pytag.structures.CaseInsensitiveDict

class pytag.codecs.**Vorbis**
Bases: pytag.codecs.VorbisComment

4.3 Constants

pytag.constants.**FIELD_NAMES** = **('title', 'artist', 'album', 'date', 'tracknumber', 'organization', 'genre', 'performer')**
Default comments/tags accepted by pytag

4.4 Containers

class pytag.containers.**OggPage**(*fileobj*)

as_bytes(*self*, *update_crc=False*)
Get the complete page as bytes.

Paramer update_crc Flag to know if recalculate the CRC.

Returns Ogg page

Return type array.array of bytes, typecode = 'B'

get_packet_info(*self*)
Gets the size of the next packet (or partial packet) in the current page. This size can be smaller than the full packet because a packet can we splited in severa pages. Also check it the packet finish in the current page. If page has no more bytes to read, this function iterates to the next page. The idea is use this function as callback in PacketReader, this way, when a packet is readed, the reader doesn't need to worry about how the packet is saved inside an ogg stream.

Returns Next packet size and if the packet finish in this page.

Return type collections.namedtuple of type PacketInfo

get_packet_reader(*self*)
Get a packet reader for the current page.

Returns A packet reader over the same stream used by this OggPage

Return type PacketReader

is_last_page(*self*)
Check if the page is the last in a logical bitstream. :returns: True if is the last one, False if not. :rtype: boolean

next_page(*self*)
Iterates to next page.

rest_of_pages(*self*)
Iterator over still not readed pages.

class pytag.containers.**OggReader**(*path*)

comments_page_position(*self*)
Returns the page number where the comments start.

process_comments(*self*, *packet*)
Returns the comments.

class pytag.containers.**Ogg**(*path*)

packets_after_comments(*self*)
Returns the number of packets in the same page after the comments packet

write_tags(*self*, *comments*)
Write the tags to a new file, if no path is provided, the original file is overwrited

class pytag.containers.**PacketReader**(*fileobj*, *get_packet_info_callback*)
My class doc.

read(*self*, *_n=-1*)
Read up to n bytes from the current packet in the stream and return them. If n is unspecified or -1, read and return all the bytes until the packet end.

4.5 Formats

class pytag.formats.**OggVorbisReader**(*path*)
Bases: pytag.codecs.Vorbis, pytag.containers.OggReader

class pytag.formats.**OggVorbis**(*path*)

4.6 Structures

4.6.1 CaseInsensitiveDict

class pytag.structures.**CaseInsensitiveDict**(*data=None*, ***kwargs*)
A case-insensitive dict-like object.

Implements all methods and operations of collections.abc.MutableMapping as well as dict's dict.copy().

All keys are expected to be strings. The structure always set the key to lower case.

```
cid = CaseInsensitiveDict()
cid['Key'] = 'value'
cid['KEY'] == 'value'   # True
list(cid) == ['key']    # True
```

If the constructor, update, or equality comparison operations are given keys that have equal str.lower(), the behavior is undefined.

4.6.2 PytagDict

class pytag.structures.**PytagDict**(*data=None*, ***kwargs*)
A case-insensitive dict-like object where only the values defines in pytag.constants.FIELD_NAMES constant are allowed as keys. If a key is not valid, is ignored without any warning.

CHAPTER 5

Indices and tables

- *genindex*
- *modindex*
- *search*