EbookLib Documentation

Release 0.16

Aleksandar Erkalovic

May 11, 2018

Contents

EbookLib is a Python library for managing EPUB2/EPUB3 and Kindle files. It's capable of reading and writing EPUB files programmatically (Kindle support is under development).

The API is designed to be as simple as possible, while at the same time making complex things possible too. It has support for covers, table of contents, spine, guide, metadata and more. EbookLib works with Python 2.7 and Python 3.3.

Homepage: https://github.com/aerkalov/ebooklib/

CHAPTER 1

ebooklib Package

1.1 ebooklib Package

1.2 epub Module

class ebooklib.epub.**EpubBook**

Bases: object

add_author(*author*, *file_as=None*, *role=None*, *uid='creator'*)

Add author for this document

add_item(*item*)

Add additional item to the book. If not defined, media type and chapter id will be defined for the item.

Args

- item: Item instance

add_metadata(*namespace*, *name*, *value*, *others=None*)

Add metadata

add_prefix(*name*, *uri*)

Appends custom prefix to be added to the content.opf document

```
>>> epub_book.add_prefix('bkterms', 'http://booktype.org/')
```

Args

- name: namespave name
- uri: URI for the namespace

get_item_with_href(*href*)

Returns item for defined HREF.

```
>>> book.get_item_with_href('EPUB/document.xhtml')
```

Args

- href: HREF for the item we are searching for

Returns Returns item object. Returns None if nothing was found.

get_item_with_id(*uid*)
Returns item for defined UID.

```
>>> book.get_item_with_id('image_001')
```

Args

- uid: UID for the item

Returns Returns item object. Returns None if nothing was found.

get_items()
Returns all items attached to this book.

Returns Returns all items as tuple.

get_items_of_media_type(*media_type*)
Returns all items of specified media type.

Args

- media_type: Media type for items we are searching for

Returns Returns found items as tuple.

get_items_of_type(*item_type*)
Returns all items of specified type.

```
>>> book.get_items_of_type(epub.ITEM_IMAGE)
```

Args

- item_type: Type for items we are searching for

Returns Returns found items as tuple.

get_metadata(*namespace*, *name*)
Retrieve metadata

get_template(*name*)
Returns value for the template.

Args

- name: template name

Returns Value of the template.

reset()
Initialises all needed variables to default values

set_cover(*file_name*, *content*, *create_page=True*)
Set cover and create cover document if needed.

Args

- file_name: file name of the cover page
- content: Content for the cover image
- create_page: Should cover page be defined. Defined as bool value (optional). Default value is True.

set_direction(*direction*)

Args

- direction: Options are "ltr", "rtl" and "default"

set_identifier(*uid*)

Sets unique id for this epub

Args

- uid: Value of unique identifier for this book

set_language(*lang*)

Set language for this epub. You can set multiple languages. Specific items in the book can have different language settings.

Args

- lang: Language code

set_template(*name*, *value*)

Defines templates which are used to generate certain types of pages. When defining new value for the template we have to use content of type 'str' (Python 2) or 'bytes' (Python 3).

At the moment we use these templates:

- ncx
- nav
- chapter
- cover

Args

- name: Name for the template
- value: Content for the template

set_title(*title*)

Set title. You can set multiple titles.

Args

- title: Title value

set_unique_metadata(*namespace*, *name*, *value*, *others=None*)

Add metadata if metadata with this identifier does not already exist, otherwise update existing metadata.

class ebooklib.epub.**EpubCover**(*uid='cover-img'*, *file_name=''*)

Bases: *ebooklib.epub.EpubItem*

Represents Cover image in the EPUB file.

get_type()

Guess type according to the file extension. Might not be the best way how to do it, but it works for now.

Items can be of type:

- ITEM_UNKNOWN = 0
- ITEM_IMAGE = 1
- ITEM_STYLE = 2
- ITEM_SCRIPT = 3
- ITEM_NAVIGATION = 4
- ITEM_VECTOR = 5
- ITEM_FONT = 6
- ITEM_VIDEO = 7
- ITEM_AUDIO = 8
- ITEM_DOCUMENT = 9
- ITEM_COVER = 10

We map type according to the extensions which are defined in ebooklib.EXTENSIONS.

Returns Returns type of the item as number.

class ebooklib.epub.**EpubCoverHtml**(*uid='cover'*, *file_name='cover.xhtml'*, *image_name=''*, *title='Cover'*)

Bases: *ebooklib.epub.EpubHtml*

Represents Cover page in the EPUB file.

get_content()

Returns content for cover page as HTML string. Content will be of type 'str' (Python 2) or 'bytes' (Python 3).

Returns Returns content of this document.

is_chapter()

Returns if this document is chapter or not.

Returns Returns book value.

exception ebooklib.epub.**EpubException**(*code*, *msg*)

Bases: exceptions.Exception

class ebooklib.epub.**EpubHtml**(*uid=None*, *file_name=''*, *media_type=''*, *content=None*, *title=''*, *lang=None*, *direction=None*)

Bases: *ebooklib.epub.EpubItem*

Represents HTML document in the EPUB file.

add_item(*item*)

Add other item to this document. It will create additional links according to the item type.

Args

- item: item we want to add defined as instance of EpubItem

add_link(***kwgs*)

Add additional link to the document. Links will be embeded only inside of this document.

```
>>> add_link(href='styles.css', rel='stylesheet', type='text/css')
```

get_body_content()
Returns content of BODY element for this HTML document. Content will be of type 'str' (Python 2) or 'bytes' (Python 3).

Returns Returns content of this document.

get_content(*default=None*)
Returns content for this document as HTML string. Content will be of type 'str' (Python 2) or 'bytes' (Python 3).

Args

- default: Default value for the content if it is not defined.

Returns Returns content of this document.

get_language()
Get language code for this book item. Language of the book item can be different from the language settings defined globaly for book.

Returns As string returns language code.

get_links()
Returns list of additional links defined for this document.

Returns As tuple return list of links.

get_links_of_type(*link_type*)
Returns list of additional links of specific type.

Returns As tuple returns list of links.

get_type()
Always returns ebooklib.ITEM_DOCUMENT as type of this document.

Returns Always returns ebooklib.ITEM_DOCUMENT

is_chapter()
Returns if this document is chapter or not.

Returns Returns book value.

set_language(*lang*)
Sets language for this book item. By default it will use language of the book but it can be overwritten with this call.

class ebooklib.epub.**EpubImage**
Bases: *ebooklib.epub.EpubItem*

Represents Image in the EPUB file.

get_type()
Guess type according to the file extension. Might not be the best way how to do it, but it works for now.

Items can be of type:

- ITEM_UNKNOWN = 0
- ITEM_IMAGE = 1
- ITEM_STYLE = 2
- ITEM_SCRIPT = 3
- ITEM_NAVIGATION = 4
- ITEM_VECTOR = 5

- ITEM_FONT = 6
- ITEM_VIDEO = 7
- ITEM_AUDIO = 8
- ITEM_DOCUMENT = 9
- ITEM_COVER = 10

We map type according to the extensions which are defined in ebooklib.EXTENSIONS.

Returns Returns type of the item as number.

class ebooklib.epub.**EpubItem**(*uid=None*, *file_name=''*, *media_type=''*, *content=''*, *manifest=True*)

Bases: object

Base class for the items in a book.

get_content(*default=''*)

Returns content of the item. Content should be of type 'str' (Python 2) or 'bytes' (Python 3)

Args

- default: Default value for the content if it is not already defined.

Returns Returns content of the item.

get_id()

Returns unique identifier for this item.

Returns Returns uid number as string.

get_name()

Returns name for this item. By default it is always file name but it does not have to be.

Returns Returns file name for this item.

get_type()

Guess type according to the file extension. Might not be the best way how to do it, but it works for now.

Items can be of type:

- ITEM_UNKNOWN = 0
- ITEM_IMAGE = 1
- ITEM_STYLE = 2
- ITEM_SCRIPT = 3
- ITEM_NAVIGATION = 4
- ITEM_VECTOR = 5
- ITEM_FONT = 6
- ITEM_VIDEO = 7
- ITEM_AUDIO = 8
- ITEM_DOCUMENT = 9
- ITEM_COVER = 10

We map type according to the extensions which are defined in ebooklib.EXTENSIONS.

Returns Returns type of the item as number.

set_content(*content*)
Sets content value for this item.

Args

- content: Content value

class ebooklib.epub.**EpubNav**(*uid='nav'*, *file_name='nav.xhtml'*, *media_type='application/xhtml+xml'*)
Bases: *ebooklib.epub.EpubHtml*

Represents Navigation Document in the EPUB file.

is_chapter()
Returns if this document is chapter or not.

Returns Returns book value.

class ebooklib.epub.**EpubNcx**(*uid='ncx'*, *file_name='toc.ncx'*)
Bases: *ebooklib.epub.EpubItem*

Represents Navigation Control File (NCX) in the EPUB.

class ebooklib.epub.**EpubReader**(*epub_file_name*, *options=None*)
Bases: object

DEFAULT_OPTIONS = {}

load()

process()

read_file(*name*)

class ebooklib.epub.**EpubWriter**(*name*, *book*, *options=None*)
Bases: object

DEFAULT_OPTIONS = {'epub2_guide': True, 'epub3_landmark': True, 'landmark_title': '

process()

write()

class ebooklib.epub.**Link**(*href*, *title*, *uid=None*)
Bases: object

class ebooklib.epub.**Section**(*title*, *href=''*)
Bases: object

ebooklib.epub.**read_epub**(*name*, *options=None*)
Creates new instance of EpubBook with the content defined in the input file.

```
>>> book = ebooklib.read_epub('book.epub')
```

Args

- name: full path to the input file
- options: extra options as dictionary (optional)

Returns Instance of EpubBook.

ebooklib.epub.**write_epub**(*name*, *book*, *options=None*)
Creates epub file with the content defined in EpubBook.

```
>>> ebooklib.write_epub('book.epub', book)
```

Args

- name: file name for the output file
- book: instance of EpubBook
- options: extra opions as dictionary (optional)

1.3 `utils` Module

ebooklib.utils.**debug**(*obj*)

ebooklib.utils.**guess_type**(*extenstion*)

ebooklib.utils.**parse_html_string**(*s*)

ebooklib.utils.**parse_string**(*s*)

1.4 Subpackages

1.4.1 plugins Package

1.4.1.1 `base` Module

class ebooklib.plugins.base.**BasePlugin**

Bases: object

after_read(*book*)
Processing after save

after_write(*book*)
Processing after save

before_read(*book*)
Processing before save

before_write(*book*)
Processing before save

html_after_read(*book*, *chapter*)
Processing HTML before read.

html_before_write(*book*, *chapter*)
Processing HTML before save.

item_after_read(*book*, *item*)
Process general item after read.

item_before_write(*book*, *item*)
Process general item before write.

1.4.1.2 booktype Module

class ebooklib.plugins.booktype.**BooktypeFootnotes**(*booktype_book*)
Bases: *ebooklib.plugins.base.BasePlugin*

NAME = 'Booktype Footnotes'

html_before_write(*book*, *chapter*)
Processing HTML before save.

class ebooklib.plugins.booktype.**BooktypeLinks**(*booktype_book*)
Bases: *ebooklib.plugins.base.BasePlugin*

NAME = 'Booktype Links'

html_before_write(*book*, *chapter*)
Processing HTML before save.

1.4.1.3 sourcecode Module

class ebooklib.plugins.sourcecode.**SourceHighlighter**
Bases: *ebooklib.plugins.base.BasePlugin*

html_before_write(*book*, *chapter*)
Processing HTML before save.

1.4.1.4 standard Module

class ebooklib.plugins.standard.**SyntaxPlugin**
Bases: *ebooklib.plugins.base.BasePlugin*

NAME = 'Check HTML syntax'

html_before_write(*book*, *chapter*)
Processing HTML before save.

ebooklib.plugins.standard.**leave_only**(*item*, *tag_list*)

1.4.1.5 tidyhtml Module

class ebooklib.plugins.tidyhtml.**TidyPlugin**(*extra={}*)
Bases: *ebooklib.plugins.base.BasePlugin*

NAME = 'Tidy HTML'

OPTIONS = {'char-encoding': 'utf8', 'tidy-mark': 'no'}

html_after_read(*book*, *chapter*)
Processing HTML before read.

html_before_write(*book*, *chapter*)
Processing HTML before save.

ebooklib.plugins.tidyhtml.**tidy_cleanup**(*content*, ***extra*)

CHAPTER 2

Usage

2.1 Reading

```
from ebooklib import epub
import ebooklib

book = epub.read_epub('test.epub')

for image in book.get_items_of_type(ebooklib.ITEM_IMAGE):
    print image
```

2.2 Writing

```
from ebooklib import epub

book = epub.EpubBook()

# set metadata
book.set_identifier('id123456')
book.set_title('Sample book')
book.set_language('en')

book.add_author('Author Authorowski')
book.add_author('Danko Bananko', file_as='Gospodin Danko Bananko', role='ill', uid=
↪'coauthor')

# create chapter
c1 = epub.EpubHtml(title='Intro', file_name='chap_01.xhtml', lang='hr')
c1.content=u'<h1>Intro heading</h1><p>Žaba je skočila u baru.</p>'

# add chapter
```

(continues on next page)

(continued from previous page)

```
book.add_item(c1)

# define Table Of Contents
book.toc = (epub.Link('chap_01.xhtml', 'Introduction', 'intro'),
             (epub.Section('Simple book'),
             (c1, ))
            )

# add default NCX and Nav file
book.add_item(epub.EpubNcx())
book.add_item(epub.EpubNav())

# define CSS style
style = 'BODY {color: white;}'
nav_css = epub.EpubItem(uid="style_nav", file_name="style/nav.css", media_type="text/
↪css", content=style)

# add CSS file
book.add_item(nav_css)

# basic spine
book.spine = ['nav', c1]

# write to the file
epub.write_epub('test.epub', book, {})
```

Further examples are available in https://github.com/aerkalov/ebooklib/tree/master/samples

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

e

Index

I

L

N

O

P

R

S

T

W