AtomGen Documentation

Release 0.1

Erasmose

Sep 27, 2017

Contents

AtomGen converts a list (or iterable) of dictionaries into Apple Newsstand Atom Feed 1.2

CHAPTER 1

Installation

Install from PyPi:

```
pip install atomgen
```

CHAPTER 2

Atomgen 0.1.9

class atomgen.**AtomGen**(***kwargs*)

Setups the Apple NewsStand Atom feed object (Newsstand Atomfeed v1.2 Specifications). Note that this is different than normal Atom Feed: Specification

Atomgen is tested and used in production on Python 2.7 Atomgen is compatible with Python 3.3 but has not been used in production.

The following parameters are ONLY used if you want to use other names for your dictionary elements than the default ones. Don't touch these parameters to keep the default settings.

Each dictionary item contains the elements of an individual newsstand issue. And it should include the following items:

Parameters **id** : string, optional

By defining this, you can rename id to any name you want to use in your original dictionary. So in your original dictionary instead of "id":123, you can have "whatever_i_want":123 The id element is used to identify the individual newsstand issue. This is an internal identifier that will not be displayed to customers. A new issue will be created if the ID is not found on an existing issue.

updated : datetime, optional

This has to be a datetime object By defining this, you can rename updated to any name you want to use in your original dictionary The updated date should be set to the date the metadata for this issue was most recently updated. Existing issues will be processed only if the updated date is later than the last modified date of the issue.

published : datetime, optional

This has to be a datetime object By defining this, you can rename published to any name you want to use in your original dictionary published: The published date is the earliest date your issue will appear on the App Store. The App Store will display the issue that has the closest published date prior to the current date, provided the issue has not ended. If multiple issues have the same published date, the most recently updated issue will be displayed. The published date should be the same as the end_date of the previous issue. This ensures that there will be no gaps where no issue is available.

end_date : datetime, optional

This has to be a datetime object By defining this, you can rename end_date to any name you want to use in your original dictionary The end_date is optional in your dictionary and the atom feed. If provided, this is the latest date your issue will appear on the App Store. The end_date must be after the published date.

summary : string, optional

By defining this, you can rename summary to any name you want to use in your original dictionary The summary element should contain a description of the issue. This information will be displayed on the App Store when this issue is current. The summary must be between10 and 2000 bytes.

icon : string, optional

By defining this, you can rename icon to any name you want to use in your original dictionary The cover art images should be *.png file and have an aspect ratio between 1:2 and 2:1. This icon must be at least 1024px on the long side.

Returns **AtomGen Object that is ready to parse a list of dictionaries** :

Methods

run (*infeed*, *update_time=datetime.datetime(2017, 9, 27, 22, 30, 17, 839658)*, *validate_image=False*)
Creates the Atom feed from the list (or iterable) of dictionaries

Parameters **infeed** : list or iterable

List of dictionaries. Each dictionary is a Newsstand entry.

update_time : datetime object, optional

This is by default set to the current UTC time. But you can set it manually too.

validate_image : Boolean, optional

This is False by default. It will check the image existance and validates that it is a PNG file and checks for its proper aspect ration based on Apple Newsstand Atom feed specifications. You NEED to make sure you have Python Imaging Library (PIL) installed if you want to use this feature.

Examples

Simple

```
>>> from atomgen import AtomGen
>>> a=[{'id':'1','updated':datetime.datetime(2013, 12, 10, 1, 9, 53,␣
↪977342),
... 'published':datetime.datetime(2013, 12, 10, 1, 10, 53, 977342),
... 'summary':"This is the summary 1",'icon':"http://ccc.com/img.png"},
... {'id':2,'updated':datetime.datetime(2013, 12, 9, 1, 9, 53, 977342),
... 'published':datetime.datetime(2013, 12, 10, 1, 7, 53, 977342),
... 'summary':"This is the summary 2",'icon':"http://ccc2.com/img2.png"}]
>>> my_atom = AtomGen()
>>> print (my_atom.run(a, update_time=datetime.datetime(2013, 12, 10, 1,␣
↪9, 53, 977342)) )
<?xml version='1.0' encoding='UTF-8'?>
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:news="http://itunes.apple.
↪com/2011/Newsstand"><updated>2013-12-10T01:09:53Z</updated><entry><id>1
↪</id><updated>2013-12-10T01:09:53Z</updated><published>2013-12-
↪10T01:10:53Z</published><summary>This is the summary 1</summary>
↪<news:cover_art_icons><news:cover_art_icon size="SOURCE" src="http://
↪ccc.com/img.png" /></news:cover_art_icons></entry><entry><id>2</id>
↪<updated>2013-12-09T01:09:53Z</updated><published>2013-12-10T01:07:53Z</
↪published><summary>This is the summary 2</summary><news:cover_art_icons>
```

Renaming element names in the original dictionary. you need to let AtomGen know the correspondence to the defaul

```
>>> b=[{'my_id':'1','when_updated':datetime.datetime(2013, 12, 10, 1, 9,␣
↪53, 977342),
... 'when_published':datetime.datetime(2013, 12, 10, 1, 10, 53, 977342),
... 'the_summary':"This is the summary 1",'myicon':"http://ccc.com/img.png
↪"},
... {'my_id':2,'when_updated':datetime.datetime(2013, 12, 9, 1, 9, 53,␣
↪977342),
... 'when_published':datetime.datetime(2013, 12, 10, 1, 7, 53, 977342),
... 'the_summary':"This is the summary 2",'myicon':"http://ccc2.com/img2.
↪png"}]
>>> my_atom2 = AtomGen(id="my_id",published="when_published",updated=
↪"when_updated",
... summary="the_summary",icon="myicon")
>>> print (my_atom2.run(b, update_time=datetime.datetime(2013, 12, 10, 1,␣
↪9, 53, 977342)) )
<?xml version='1.0' encoding='UTF-8'?>
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:news="http://itunes.apple.
↪com/2011/Newsstand"><updated>2013-12-10T01:09:53Z</updated><entry><id>1
↪</id><updated>2013-12-10T01:09:53Z</updated><published>2013-12-
↪10T01:10:53Z</published><summary>This is the summary 1</summary>
↪<news:cover_art_icons><news:cover_art_icon size="SOURCE" src="http://
↪ccc.com/img.png" /></news:cover_art_icons></entry><entry><id>2</id>
↪<updated>2013-12-09T01:09:53Z</updated><published>2013-12-10T01:07:53Z</
↪published><summary>This is the summary 2</summary><news:cover_art_icons>
↪<news:cover_art_icon size="SOURCE" src="http://ccc2.com/img2.png" /></
↪news:cover_art_icons></entry></feed>
```

Using a dicionary of dictionaries (like a json structure) for input

```
>>> c={1:{'updated':datetime.datetime(2013, 12, 10, 1, 9, 53, 977342),
... 'published':datetime.datetime(2013, 12, 10, 1, 10, 53, 977342),
... 'summary':"This is the summary 1",'icon':"http://ccc.com/img.png"},
... 2:{'updated':datetime.datetime(2013, 12, 9, 1, 9, 53, 977342),
... 'published':datetime.datetime(2013, 12, 10, 1, 7, 53, 977342),
... 'summary':"This is the summary 2",'icon':"http://ccc2.com/img2.png"},}
>>> print (my_atom.run(c, update_time=datetime.datetime(2013, 12, 10, 1,␣
↪9, 53, 977342)) )
<?xml version='1.0' encoding='UTF-8'?>
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:news="http://itunes.apple.
↪com/2011/Newsstand"><updated>2013-12-10T01:09:53Z</updated><entry><id>1
↪</id><updated>2013-12-10T01:09:53Z</updated><published>2013-12-
↪10T01:10:53Z</published><summary>This is the summary 1</summary>
↪<news:cover_art_icons><news:cover_art_icon size="SOURCE" src="http://
↪ccc.com/img.png" /></news:cover_art_icons></entry><entry><id>2</id>
↪<updated>2013-12-09T01:09:53Z</updated><published>2013-12-10T01:07:53Z</
↪published><summary>This is the summary 2</summary><news:cover_art_icons>
↪<news:cover_art_icon size="SOURCE" src="http://ccc2.com/img2.png" /></
↪news:cover_art_icons></entry></feed>
```

Using a dicionary of dictionaries for input with custom names

```
>>> d={'1':{'when_updated':datetime.datetime(2013, 12, 10, 1, 9, 53,␣
↪977342),
... 'when_published':datetime.datetime(2013, 12, 10, 1, 10, 53, 977342),
```

```
... 'the_summary':"This is the summary 1",'myicon':"http://ccc.com/img.png
↪"},
... 2:{'when_updated':datetime.datetime(2013, 12, 9, 1, 9, 53, 977342),
... 'when_published':datetime.datetime(2013, 12, 10, 1, 7, 53, 977342),
... 'the_summary':"This is the summary 2",'myicon':"http://ccc2.com/img2.
↪png"},}
>>> my_atom2 = AtomGen(id="my_id",published="when_published",updated=
↪"when_updated",
... summary="the_summary",icon="myicon")
>>> print (my_atom2.run(d, update_time=datetime.datetime(2013, 12, 10, 1,␣
↪9, 53, 977342)) )
<?xml version='1.0' encoding='UTF-8'?>
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:news="http://itunes.apple.
↪com/2011/Newsstand"><updated>2013-12-10T01:09:53Z</updated><entry><id>1
↪</id><updated>2013-12-10T01:09:53Z</updated><published>2013-12-
↪10T01:10:53Z</published><summary>This is the summary 1</summary>
↪<news:cover_art_icons><news:cover_art_icon size="SOURCE" src="http://
↪ccc.com/img.png" /></news:cover_art_icons></entry><entry><id>2</id>
↪<updated>2013-12-09T01:09:53Z</updated><published>2013-12-10T01:07:53Z</
↪published><summary>This is the summary 2</summary><news:cover_art_icons>
↪<news:cover_art_icon size="SOURCE" src="http://ccc2.com/img2.png" /></
↪news:cover_art_icons></entry></feed>
```

As you can see it generates exactly the same Atom feed in the end. But it gives you the flexibility of modifying your own dictionary keys with the names you like.

Automatic Validation of images of the feed

It validates the remote images for their existance, their type to be PNG and aspect ratio to be .5 to 2 as defined in Apple Newsstand Atomfeed specifications. You need to turn validate_image=True

Validating images

```
>>> d={'1':{'when_updated':datetime.datetime(2013, 12, 10, 1, 9, 53,␣
↪977342),
... 'when_published':datetime.datetime(2013, 12, 10, 1, 10, 53, 977342),
... 'the_summary':"This is the summary 1",'myicon':"http://cdn.tennis.com/
↪uploads/magazine/test_material/img_1024_600.png"},
... 2:{'when_updated':datetime.datetime(2013, 12, 9, 1, 9, 53, 977342),
... 'when_published':datetime.datetime(2013, 12, 10, 1, 7, 53, 977342),
... 'the_summary':"This is the summary 2",'myicon':"http://cdn.tennis.com/
↪uploads/magazine/test_material/img_1024_600.png"},}
>>> my_atom2 = AtomGen(id="my_id",published="when_published",updated=
↪"when_updated",
... summary="the_summary",icon="myicon")
>>> print (my_atom2.run(d, update_time=datetime.datetime(2013, 12, 10, 1,␣
↪9, 53, 977342), validate_image=True) )
http://cdn.tennis.com/uploads/magazine/test_material/img_1024_600.png␣
↪validated
<?xml version='1.0' encoding='UTF-8'?>
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:news="http://itunes.apple.
↪com/2011/Newsstand"><updated>2013-12-10T01:09:53Z</updated><entry><id>1
↪</id><updated>2013-12-10T01:09:53Z</updated><published>2013-12-
↪10T01:10:53Z</published><summary>This is the summary 1</summary>
↪<news:cover_art_icons><news:cover_art_icon size="SOURCE" src="http://
↪cdn.tennis.com/uploads/magazine/test_material/img_1024_600.png" /></
↪news:cover_art_icons></entry><entry><id>2</id><updated>2013-12-
↪09T01:09:53Z</updated><published>2013-12-10T01:07:53Z</published>
↪<summary>This is the summary 2</summary><news:cover_art_icons>
↪<news:cover_art_icon size="SOURCE" src="http://cdn.tennis.com/uploads/
↪magazine/test_material/img_1024_600.png" /></news:cover_art_icons></
↪entry></feed>
```

CHAPTER 3

Manual image validation

Note that for automatic image validation, turn validate_image=True in atomgen.AtomGen.run

validate_img.**check_img**(*img*)
checks a local image for Apple Newsstand feed specifications v1.2

validate_img.**validate_img_on_web**(*img_url*)
checks a remote image for existance and Apple Newsstand feed specifications v1.2 example:

```
>>> from atomgen.validate_img import validate_img_on_web
>>> validate_img_on_web("http://cdn.tennis.com/uploads/magazine/test_material/img_
↪1200_500.png")
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

CHAPTER 5

Author

Erasmose Github Linkedin

Python Module Index

a

v

Index