Fresh Tomatoes ReadMe

Udacity Project Notes

To see a working example of the Fresh Tomatoes project, simply run *movieentry.py* directly. It will generate example content and launch the Fresh Tomatoes page in your browser. The original *fresh_tomatoes.py* file has been modified, so make sure you are using the version provided.

It makes use of some extra libraries (see below), but they are not required. To get the full effect, I recommend you run it with the *requests* library installed (http://docs.python-requests.org/en/latest/). It's used to fill in missing movie information by sending an HTTP request to the Online Movie Database API (OMDB). If you run *usage_example.py* you will see a full spread displaying complete and missing information without needing to use the *requests* library.

Finally, please note that this project is written for Python 3.

The MovieEntry Class

Descripton

Each instance of the MovieEntry class holds information about one movie in a format that "fresh_tomatoes.py" can properly render into HTML. The minimum information that must be provided is the movie's title and YouTube trailer URL. Note that if you do not enable automatic information retrieval, you must also manually provide a poster image URL.

Requirements

The *MovieEntry* Class has no required libraries outside of the standard **Python 3** libraries. However, some optional functionality requires additional modules.

Bleach: Bleach is used to escape all of the movie information before it is inserted into the Fresh Tomatoes HTML file. If it is not present, the raw values will simply be trusted and used as-is. That's acceptable for this demonstration, but would not be wise development practice.

Requests: The *MovieEntry* Class is capable of retrieving information from the Online Movie Database API (http://www.omdbapi.com). These requests are formatted, sent, and interpretted using the requests module. If it is not availabl, all movie information must be manually provided.

Automatic Information Retrieval

As stated above, the *MovieEntry* Class can automatically retrieve details for the film using only its title. Information is provided by the Online Movie Database (OMDB) API (http://www.omdbapi.com). However, the title provided must exactly match the title of the movie in OMDB. Be careful about "The" and "2" versus "II" and similar title elements.

When using automatic information retrieval, you must still provide the movie's title and YouTube trailer URL. If you manually provide any content, what you provide will take precedent over any information retrieved from OMDB.

To use automatic information retrieval, leave the constructor parameter *get_missing_info* set to its default value of True or invoke the *MovieEntry*.get_missing_info() method at any time.

MovieEntry Class Properties

Class Property	Constructor Parameter and Default Value	Description	Required?	Automatic Retrieval?
MovieEntry.title	title	String formatted title of the movie up to 50 characters.	Yes	No

MovieEntry.trailer_youtube_url	trailer_url	URL to the YouTube entry for the movie's trailer. It must be in one of two formats: https://www.youtube.com/watch?v=4pSe4OGAuI4 or https://youtu.be/4pSe4OGAuI4.	Yes	No
MovieEntry.poster_image_url	poster_url=*None*	URL to an image to use as the poster art. It must be a properly formatted URL beginning with 'http' or 'https' and ending in a valid image extension (.jpg, .gif, .jpeg, .png, .tif).	If automatic retrieval is used, no. Otherwise, yes.	Yes
MovieEntry.plot	plot=*None*	A string formatted description of the movie up to 300 characters.	No	Yes
MovieEntry.year	year=*None*	Integer value of the year the film was released. It may be between 1900 and two years from today for unreleased films.	No	Yes
MovieEntry.actors	actors=*None*	A coma-separated list of the names of the actors in the film. Any number of actors can be supplied, but each name must be under 50 characters.	No	Yes
MovieEntry.directors	directors=*None*	A coma-separated list of the names of the film's director(s). Any number of directors can be provided, but each name must be under 50 characters.	No	Yes

MovieEntry Class Methods

For brevity, some methods intended for internal use have been omitted from this table. Property wrappers and setters exist for each property. Setters call a method of the form *validate_property(self, content)*. Each of these validation functions checks the value of the input and returns a safe representation of the input if one can be found or *None* if one cannot be determined. The other class methods are listed below:

Class Method	**Description**	**Parameters**	**Returns**
MovieEntry.clean_output(self, content)	If the bleach module has been successfully imported, the string *content* will be HTML escaped for safe display. Otherwise, the current value of *content* will be trusted. This method is mainly used by the property getters for each property.	*content='example string'*	An HTML-escaped representation of *content*, if able.
MovieEntry.get_missing_info(self)	If the requests module has been successfully imported, this method will request information on the move from OMDB and fill in any missing information with the result. This is	None.	Nothing.

	generally invoked automatically unless the constructor parameter *get_missing_info* was set to *False*. However, it may be called at other times manually. User-input takes precedence.		
MoveEntry.html(self)	Uses the movie information provided to return a properly formatted representation of that that movie for inclusion in the Fresh Tomatoes webpage. Any non-required, missing information will be ommitted.	None.	HTML representation of the movie information.

Usage Examples

Import the MovieEntry Class and Fresh Tomatoes.

```
from movieentry import MovieEntry
import fresh_tomatoes
```

Create a movie entry with the minimum amount of information required, which is the movie's title and URL of its trailer on YouTube. Fill in the rest automatically.

```
movie_one = MovieEntry(title='Safety Not Guaranteed',
                       trailer_url='https://youtu.be/73jSnAs7mq8')
```

Create a movie entry by manually using only the minimum amount of data needed to display properly, without attempting to fill in the rest.

```
movie_two = MovieEntry(title='History of Future Folk',
                       trailer_url='https://youtu.be/fZrDALCsKwI',
                       poster_url='http://ia.media-
imdb.com/images/M/MV5BNzA3MDI3MzAxMl5BMl5BanBnXkFtZTcwNDY2Mjc0OQ@@._V1_SX300.jpg',
                       get_missing_info=False)
```

Create a movie entry by manually setting all movie properties when the class is constructed.

```
movie_three = MovieEntry(title='Knights of Badassdom',
                         trailer_url='https://youtu.be/tOsGwkUR7Lk',
                         poster_url='http://ia.media-
imdb.com/images/M/MV5BMjMyNzAyNzUzMF5BMl5BanBnXkFtZTgwOTQ0NDMyMTE@._V1_SX300.jpg',
                         plot='''
                            Live-action role players conjure up a demon from
                            Hell by mistake and they must deal
                            with the consequences.
                          ''',
                        year=2013,
                        actors='D.R. Anderson, W. Earl Brown, Michael Carpenter, Kevin Connell',
                        directors='Joe Lynch',
                        get_missing_info=False)
```

Create a movie with the minimum required information, then fill in the rest later manually.

```
movie_four = MovieEntry(title='Serenity',
                        trailer_url='https://youtu.be/JY3u7bB7dZk',
                        get_missing_info=False)
```

```
movie_four.poster_image_url = 'http://ia.media-
imdb.com/images/M/MV5BMTI0NTY1MzY4NV5BMl5BanBnXkFtZTcwNTczODAzMQ@@._V1_SX300.jpg'
movie_four.plot = 'The crew of the ship Serenity tries to evade an assassin sent to recapture one of
their number who is telepathic.'
movie_four.actors = 'Nathan Fillion, Gina Torres, Alan Tudyk, Morena Baccarin'
movie_four.year = 2005
```

Oops, we don't know the director for Serenity. Retrieve it from OMDB.

```
movie_four.get_missing_info()
```

MovieEntry objects must be pass to the Fresh Tomatoes script as a list:

```
my_movie_list = [movie_one, movie_two, movie_three, movie_four]
```

Display the page.

```
fresh_tomatoes.open_movies_page(movies=my_movie_list)
```

All of this code available in *usage_example.py*.