TUBTUB TEAM

PYTUBE 분석

분배

- 박승권: api.py
- 서은정: model.py
- 유동현: util.py
- 박선배: jsinterp.py

TUBTUB TEAM

MODEL.PY -은정

MODELS.PY - IMPORT

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from __future__ import unicode_literals

import os

from time import clock

try:
    from urllib2 import urlopen
except ImportError:
    from urllib.request import urlopen
```

- 파이썬 2의 경우 기본 인코딩이 ASCII, 파이썬 3의 경우 UTF-8 /버전 별로 인코딩이 다르기 때문에 문제가 생길 수 있다
- 코딩을 명시해주거나 from __future__ import print_function 모듈을 사용하면 UTF-8로 인코딩 된다
- import os : 디렉토리 접근 시 이용
- import clock : 시간 기록을 위해 사용
- 파이썬2일 경우 : from urllib2 import urlopen
- 파이썬 3일 경우 : from urllib.request import urlopen
- 원하는 URL로부터 응답을 받을 수 있다

MODELS.PY - VIDEO CLASS __INIT__

```
class Video(object):

    # 내가 가져오고자 하는 비디오를 설정(초기화)한다
    def __init__(self, url, filename, extension, resolution=None, video_codec=None,
                 profile=None, video_bitrate=None, audio_codec=None, audio_bitrate=None):
        """
        :param str url:
            영상의 URL 주소 (e.g.: https://youtube.com/watch?v=...)
        :param str filename:
            저장할 파일명(확장자 빼고)
        :param str extention:
            저장할 확장자명 (e.g.: mp4, flv, webm)
        :param str resolution:
            (옵션) 기본 해상도 (e.g.: 720p, 1080p).
        :param str video_codec:
            (옵션) 영상 코덱
        :param str profile:
            (옵션) 임의의 품질 프로필?? -> 뭐하는 녀석??
        :param str video_bitrate:
            (옵션) 비디오 전송 비트 속도
        :param str audio_codec:
            (옵션) 오디오 코덱
        :param str audio_bitrate:
            (옵션) 오디오 전송 비트 속도
        """

        # 입력받은 인자들을 변수에 저장, 인자 값이 없는 옵션들은 None이 된다
        self.url = url
        self.filename = filename
        self.extension = extension
        # 3개는 필수
        self.resolution = resolution
        self.video_codec = video_codec
        self.profile = profile
        self.video_bitrate = video_bitrate
        self.audio_codec = audio_codec
        self.audio_bitrate = audio_bitrate
```

MODELS.PY - DOWNLOAD

```
# 설정한 비디오를 다운로드 한다
def download(self, path, chunk_size=8 * 1024, on_progress=None, on_finish=None, force_overwrite=False):
    """
    :param str path:
        비디오 저장 위치
    :param int chunk_size:
        한 번에 읽어오는 데이터 정보량 = 기본적으로 8바이트
    :param func on_progress:
        (옵션) 비디오 정보를 한 번 받아올 때마다 실행되는 함수 / 매개변수는 지금까지 받은 데이터양, 파일 사이즈, 시작 시간
    :param func on_finish:
        (옵션) 다운로드 완료 후 실행되는 함수 / 파일 위치를 매개변수로 넘겨줌
    :param bool force_overwrite:
        (옵션) 동일한 파일이 있다면 덮어쓰기를 할지 말지 결정함 ture or false
    """
```

- my_video = Video(“https://www.youtube.com/watch?v=Nck6BZga7TQ”, ‘test’, ‘mp4’)
- my_video.download(‘/User///.////me/Desktop’)

MODELS.PY - DOWNLOAD

```
#입력 받은 파일경로에서 .이나 ..을 최대한 삭제, 슬래쉬 모양이나 방향을 규칙적으로 보이게끔 바꿔줌
path = os.path.normpath(path)

#해당 경로가 존재하면 True 없으면 False
if os.path.isdir(path):

    # '파일네임.확장자명'이라는 문자열이 만들어짐 -> 'test_video.mp4'
    filename = "{0}.{1}".format(self.filename, self.extension)
    # 경로에 파일이름을 합침
    path = os.path.join(path, filename)


# 재지정된 경로가 파일이고(이미 같은 이름의 파일이 있다는 것) 덮어쓰기 허용하지 않는 경우
# 강제로 예외 일으키기
if os.path.isfile(path) and not force_overwrite:
    raise OSError("Conflicting filename:'{0}'".format(self.filename))
```

- my_video.download(‘/User///.////me/Desktop’)
- path = ‘/User/me/Desktop’
- filename = ‘test.mp4’

MODELS.PY - DOWNLOAD

```python
#해당 영상의 URL로 리퀘스트를 날리고 http 응답을 받아옴
response = urlopen(self.url)

# 응답의 header 부분을 딕셔너리로 리턴
meta_data = dict(response.info().items())

#딕셔너리에서 "Content-Length"나 "content-length"값을 가져와 정수로 변환해 저장
# 근데 헤더에 저 키가 없던데???? -> 오류 나면서 함수 종료
file_size = int(meta_data.get("Content-Length") or meta_data.get("content-length"))

# 다운로드 받은 총 데이터 크기를 저장하기 위해 0으로 초기화
self._bytes_received = 0
```

```
meta_data = {dict} {'Content-Type': 'text/html; charset=utf-8', 'Vary': 'Accept-Encoding', 'Date': 'Sun, 17 Jan 2016 14:23:22 GMT', 'P3P': 'CP="This is not a P3P p...
  __len__ = {int} 15
  'Accept-Ranges' (4380263600) = {str} 'none'
  'Alt-Svc' (4380231416) = {str} 'quic=":443"; ma=604800; v="30,29,28,27,26,25"'
  'Alternate-Protocol' (4380243048) = {str} '443:quic,p=1'
  'Cache-Control' (4380262832) = {str} 'no-cache'
  'Connection' (4380191664) = {str} 'close'
  'Content-Type' (4380263024) = {str} 'text/html; charset=utf-8'
  'Date' (4380231192) = {str} 'Sun, 17 Jan 2016 14:23:22 GMT'
  'Expires' (4380231304) = {str} 'Tue, 27 Apr 1971 19:44:06 EST'
  'P3P' (4380231360) = {str} 'CP="This is not a P3P policy! See http://support.google.com/accounts/answer/151657?hl=ko for more info."'
  'Server' (4380231136) = {str} 'gwiseguy/2.0'
  'Set-Cookie' (4380263344) = {str} 'VISITOR_INFO1_LIVE=yyWPo_JUDF0; path=/; domain=.youtube.com; expires=Sat, 17-Sep-2016 02:16:22 GMT; httponly'
  'Vary' (4380231528) = {str} 'Accept-Encoding'
  'X-Content-Type-Options' (4380244128) = {str} 'nosniff'
  'X-Frame-Options' (4380263216) = {str} 'SAMEORIGIN'
  'X-XSS-Protection' (4380242976) = {str} '1; mode=block; report=https://www.google.com/appserve/security-bugs/log/youtube'
```

MODELS.PY - DOWNLOAD

```
# 시작 시간 기록
start = clock()

#지정해준 경로에 데이터 입력을 위해 파일 생성, 끝나면 알아서 파일 닫힘
try:
    with open(path, 'wb') as dst_file:
        while True:
            # 지정해준 크기만큼 읽고, 읽어온 내용을 반환해 buffer에 저장
            self._buffer = response.read(chunk_size)

            # 읽어 온 데이터가 없을 때
            if not self._buffer:

                # on_finish 함수가 있다면 실행
                if on_finish:
                    on_finish(path)
                break

            # 총 받은 크기 += 읽어 온 데이터의 크기
            self._bytes_received += len(self._buffer)

            # 파일에 쓰기
            dst_file.write(self._buffer)

            # 다운로드 중 실행할 함수가 있다면 실행
            if on_progress:
                on_progress(self._bytes_received, file_size, start)

#파일을 쓰는 도중 control+c 나 delete 키가 눌리면 예외 발생
except KeyboardInterrupt:

    # 경로에 있는 파일 지움
    os.remove(path)

    # 예외 발생, 콘솔창에 저 문구가 뜸
    raise KeyboardInterrupt(
        "Interrupt signal given. Deleting incomplete video.")
```

MODELS.PY -__REPR__

```
def __repr__(self):
    # <Vidoe: 코덱(.mp4) - 720p - ? >
    return "<Video: {0} (.{1}) - {2} - {3}>".format(
        self.video_codec, self.extension, self.resolution, self.profile)
```

- 내가 가져오고자 하는 영상의 속성들을 문자열로 출력해 보여 준다.

MODELS.PY -__LT__

```
# 비디오 객체들을 비교할 때 사용한다. 비디오를 정렬(sorting)할 때 유용함.
def __lt__(self, other):
    """
    :param other:
        비교할 다른 비디오 객체
    """
    # other가 비디오 객체인지 확인
    if isinstance(other, Video):
        v1 = "{0} {1}".format(self.extension, self.resolution)
        v2 = "{0} {1}".format(other.extension, other.resolution)
        return (v1 > v2) - (v1 < v2) < 0
```

- 문자열을 비교한 후, other 비디오가 self 비디오 보다 크다면 True, 반대면 False를 반환
- 문자열 비교는 알파벳 소문자 > 알파벳 대문자 > 숫자 순이다. (유니코드)

MODELS.PY 정리

- models.py의 Video클래스는 download 메소드를 가지고 있지만 현재 오류가 발생한다.
- download 메소드는 사용되지 않고 api.py에 있는 YouTube 클래스의 get_video_data 메소드로 영상 데이터를 받는 것 같다.
- Video 클래스는 YouTube 클래스 _add_video 메소드에서 쓰이고 있다. (딱히 큰 쓰임이 보이진 않음)

EXCEPTIONS.PY

```python
class MultipleObjectsReturned(Exception):
    """
    조건(확장자 해상도 프로필)에 해당하는 비디오 객체가 여러 개일 때 예외를 발생시킨다
    api.py
    """
    pass


class ExtractorError(Exception):
    """
    js 문자열 파싱이 제대로 되지 않았을 때
    jsinterp.py
    """


class PytubeError(Exception):
    """
    url을 통해 응답을 받지 못한 경우
    html에서 찾고자 하는 부분이 없는 경우
    api.py
    """
    pass
```

EXCEPTIONS.PY

```python
class CipherError(Exception):
    """
    동영상 서명 암호화에 실패했을 때
    api.py
    """
    pass


class DoesNotExist(Exception):
    """
    조건(확장자 해상도 프로필)에 해당하는 비디오 객체가 하나도 없을 때
    api.py
    """
    pass


class AgeRestricted(Exception):
    """
    연령 제한이 걸린 비디오를 다운로드 하려할 때 발생
    api.py
    """
    pass
```