eGenix.com

Python Meeting Düsseldorf 17.01.2018

YouTube API

mit Python ansprechen

Marc-André Lemburg :: eGenix.com GmbH

Ziel: Automatisierung von YouTube Admin Aufgaben

- Konkret:
  Im EuroPython Video Channel mußte bei 160+ Videos ein Copyright Eintrag hinzugefügt werden
- Geht im YouTube Channel Manager nur per Hand
- Lösung: YouTube API per Python ansprechen

YouTube API mit Python

- Viele Tools und Module für YT Downloads
- Keine brauchbaren für Admin Aufgaben
- Ältere YT API Python Wrapper:
  funktionieren meist nicht mehr

- Einzig gangbarer Weg:
  Google YouTube API direkt ansprechen

Google YouTube API

- REST API ähnlich...
- OAuth2 Authenfizierung
  - Tokens mit Zeitbeschränkung (1 Stunde)
  - Eher für Webanwendungen gedacht, als für Kommandozeilentools
  - Authentifizierung benötigt einen Browser
  - Einrichtung: https://console.developers.google.com/start/api?id=youtube
- Python Modul:
  direkter Wrapper um das REST API

Google YouTube API: Dokumentation

- Google Client Lib:
  - https://developers.google.com/api-client-library/python/start/get_started
- YouTube API:
  - https://developers.google.com/api-client-library/python/apis/youtube/v3
- YouTube API Reference:
  - https://developers.google.com/youtube/v3/docs/
- Sieht alles nett aus, hat aber viele Lücken und gibt keinen guten Überblick

Google YouTube API: Python Support

- Python Modul:
  google-api-python-client
- OAuth Support:
  google-auth-oauthlib
- Credential Management:
  "roll you own"

Sprint Repo & Script

- Auf Github als Repo unter malemburg:
  - https://github.com/malemburg/youtube-manager-cli/
- Script youtube_api.py:
  - https://github.com/malemburg/youtube-manager-cli/blob/master/youtube_api.py

YouTube Service Object

```python
# Google APIs
from googleapiclient import discovery
from google_auth_oauthlib import flow

def youtube_service():

    """ Create an authenticated YouTube service object

        The API will automatically ask for authentication in case the
        existing credential and OAuth2 files do not allow access.

    """
    # Initiate the installed flow for OAuth2
    oauth_flow = flow.InstalledAppFlow.from_client_secrets_file(
        CLIENT_SECRETS_FILE,
        SCOPES)

    # Try to read credentials from file, otherwise ask user for
    # to grant access
    credentials = read_credentials()
    if credentials is not None:
        credentials = refresh_token(credentials)
    if (credentials is None or
        not credentials.valid or
        credentials.expired):
        # Ask the user to authenticate using a browser
        credentials = oauth_flow.run_console()
        write_credentials(credentials)
        print ('Token will expire at %s' % credentials.expiry)

    # Build service object
    return discovery.build(API_SERVICE_NAME,
                           API_VERSION,
                           credentials=credentials)
```

OAuth Token mit Zeitlimit

```
# Google APIs
from googleapiclient import discovery
from google_auth_oauthlib import flow

def youtube_service():

    """ Create an authenticated YouTube service object

        The API will automatically ask for authentication in case the
        existing credential and OAuth2 files do not allow access.

    """
    # Initiate the installed flow for OAuth2
    oauth_flow = flow.InstalledAppFlow.from_client_secrets_file(
        CLIENT_SECRETS_FILE,
        SCOPES)

    # Try to read credentials from file, otherwise ask user for
    # to grant access
    credentials = read_credentials()
    if credentials is not None:
        credentials = refresh_token(credentials)
    if (credentials is None or
        not credentials.valid or
        credentials.expired):
        # Ask the user to authenticate using a browser
        credentials = oauth_flow.run_console()
        write_credentials(credentials)
        print ('Token will expire at %s' % credentials.expiry)

    # Build service object
    return discovery.build(API_SERVICE_NAME,
                           API_VERSION,
                           credentials=credentials)
```

Token erneuern, falls möglich

Service Objekt: liefert Python Dictionaries und Listen

```
def channels_list_by_username(service, **kwargs):

    results = service.channels().list(
         part='snippet,contentDetails,statistics',
         forUsername='GoogleDevelopers',
    ).execute()
    #print ('Results: %r' % results)
    print('This channel\'s ID is %s. Its title is %s, and it has %s views.' %
        (results['items'][0]['id'],
         results['items'][0]['snippet']['title'],
         results['items'][0]['statistics']['viewCount']))
```

Google API Zugriff: Besonderheiten

```
def channels_list_by_username(service, **kwargs):

    results = service.channels().list(
         part='snippet,contentDetails,statistics',
         forUsername='GoogleDevelopers',
    ).execute()
    #print ('Results: %r' % results)
    print('This channel\'s ID is %s. Its title is %s, and it has %s views.' %
        (results['items'][0]['id'],
         results['items'][0]['snippet']['title'],
         results['items'][0]['statistics']['viewCount']))
```

Container-Zugriff über "Aufrufe"

Erst .execute() liefert die Daten

Weg zum Ziel: Update der Video Details

- Service Objekt erzeugen
- Channel ID finden
- Playlist ID finden
- Playlist Items abfragen (Videos)
- Video IDs finden
- Video Details laden, ändern, zurückschreiben

PS: Weg erst nach viel Trial & Error gefunden

Problem: Finde die Channel ID

```python
# Fuzzy matching
from fuzzywuzzy import fuzz

def find_channel_id(service, channel_name, min_ratio=75):

    results = service.search().list(
        q=channel_name,
        part='snippet',
        type='channel',
        maxResults=5).execute()
    #pprint.pprint(results)
    items = results['items']
    if not items:
        raise KeyError('No channel found')
    first_item = items[0]
    details = first_item['snippet']
    channel_title = details['channelTitle']
    ratio = fuzz.partial_ratio(channel_title, channel_name)
    if ratio < min_ratio:
        raise KeyError('Channel found, but does not match (ratio=%s): %r' %
                       (ratio, channel_title))
    return details['channelId']
```

Problem: Finde die Channel ID

```
# Fuzzy matching
from fuzzywuzzy import fuzz

def find_channel_id(service, channel_name, min_r

    results = service.search().list(
        q=channel_name,
        part='snippet',
        type='channel',
        maxResults=5).execute()
    #pprint.pprint(results)
    items = results['items']
    if not items:
        raise KeyError('No channel found')
    first_item = items[0]
    details = first_item['snippet']
    channel_title = details['channelTitle']
    ratio = fuzz.partial_ratio(channel_title, channel_name)
    if ratio < min_ratio:
        raise KeyError('Channel found, but does not match (ratio=%s): %r' %
                       (ratio, channel_title))
    return details['channelId']
```

Objektzugriff am besten über .search()

Fuzzy Matching zur Verifizierung

Problem: Result Paging

```
YT_SEARCH_MAX_RESULTS_LIMIT = 50

def get_channel_video_ids(service, channel_id, query=None, max_results=100):

    # Fetch results in pages
    all_items = []
    next_page = None
    while len(all_items) < max_results:
        get_results = min(max_results - len(all_items),
                          YT_SEARCH_MAX_RESULTS_LIMIT)
        results = service.search().list(
            q=query,
            channelId=channel_id,
            part='id',
            type='video',
            maxResults=get_results,
            pageToken=next_page).execute()
        #pprint.pprint(results)
        all_items.extend(results['items'])
        next_page = results.get('nextPageToken')
        if next_page is None:
            break

    # Extract IDs
    return [entry['id']['videoId'] for entry in all_items]
```

Problem: Result Paging

```
YT_SEARCH_MAX_RESULTS_LIMIT = 50

def get_channel_video_ids(service, channel_id, query=None, max_results=100):

    # Fetch results in pages
    all_items = []
    next_page = None
    while len(all_items) < max_results:
        get_results = min(max_results - len(all_items),
                          YT_SEARCH_MAX_RESULTS_LIMIT)
        results = service.search().list(
            q=query,
            channelId=channel_id,
            part='id',
            type='video',
            maxResults=get_results,
            pageToken=next_page).execute()
        #pprint.pprint(results)
        all_items.extend(results['items'])
        next_page = results.get('nextPageToken')
        if next_page is None:
            break

    # Extract IDs
    return [entry['id']['videoId'] for entry in all_items]
```

Max. 50 Objekte erlaubt

Problem: Video Details laden

```
DEFAULT_VIDEO_PARTS = 'snippet,statistics,status,contentDetails'

def get_video_details(service, video_id):

    results = service.videos().list(
        id=video_id,
        part=DEFAULT_VIDEO_PARTS,
        ).execute()
    #pprint.pprint(results)
    items = results['items']
    if not items:
        raise KeyError('Video with ID %r not found' % video_id)
    return items[0]

def update_video_details(service, video_details):

    assert video_details['id']
    # Note: YT complains when passing in a body structure with parts
    # which are not listed in part.
    results = service.videos().update(
        part=DEFAULT_VIDEO_PARTS,
        body=video_details,
        ).execute()
    #pprint.pprint(results)
    return results
```

Problem: Video Details laden

```
DEFAULT_VIDEO_PARTS = 'snippet,statistics,status,contentDetails'

def get_video_details(service, video_id):

    results = service.videos().list(
        id=video_id,
        part=DEFAULT_VIDEO_PARTS,
        ).execute()
    #pprint.pprint(results)
    items = results['items']
    if not items:
        raise KeyError('Video with ID %r not found' % video_id)
    return items[0]

def update_video_details(service, video_details):

    assert video_details['id']
    # Note: YT complains when passing in a body structure with parts
    # which are not listed in part.
    results = service.videos().update(
        part=DEFAULT_VIDEO_PARTS,
        body=video_details,
        ).execute()
    #pprint.pprint(results)
    return results
```

Zugriff auf Videos nur über videos()

"part" definiert welche Daten man haben möchte

Lösung

```
import youtube_api

### Tasks

def update_europython_video_description(service, video_details):

    # Don't add the footer again
    if 'CC BY-NC-SA' in video_details['snippet']['description']:
        return False

    video_details['snippet']['description'] += """

License: This video is licensed under the CC BY-NC-SA 3.0 license: https://creativeco
Please see our speaker release agreement for details: https://ep2017.europython.eu/en
"""
    return True

def fix_europython_2017_videos(service):

    # EuroPython 2017 playlist
    playlist_id = 'PL8uoeex94UhG9QAoRICebFpeKK2MOHerh'
    items = youtube_api.get_playlist_items(service, playlist_id,
                                           max_results=1000)
    for item in items:
        video_id = item.contentDetails.videoId
        video_details = youtube_api.get_video_details(service, video_id)
        print ('Working on video %r' % video_details['snippet']['title'])
        changed = update_europython_video_description(service, video_details)
        if changed:
            youtube_api.update_video_details(service, video_details)
            print ('    ... fixed')

###

if __name__ == '__main__':
    service = youtube_api.youtube_service()
    fix_europython_2017_videos(service)
```

Lösung

```
import youtube_api

### Tasks

def update_europython_video_description(service, video_details):

    # Don't add the footer again
    if 'CC BY-NC-SA' in video_details['snippet']['description']:
        return False

    video_details['snippet']['description'] += """

License: This video is licensed under the CC BY-NC-SA 3.0 licens
Please see our speaker release agreement for details: https://ep
"""
    return True

def fix_europython_2017_videos(service):

    # EuroPython 2017 playlist
    playlist_id = 'PL8uoeex94UhG9QAoRICebFpeKK2MOHerh'
    items = youtube_api.get_playlist_items(service, playlist_id,
                                           max_results=1000)
    for item in items:
        video_id = item.contentDetails.videoId
        video_details = youtube_api.get_video_details(service, video_id)
        print ('Working on video %r' % video_details['snippet']['title'])
        changed = update_europython_video_description(service, video_details)
        if changed:
            youtube_api.update_video_details(service, video_details)
            print ('    ... fixed')

###

if __name__ == '__main__':
    service = youtube_api.youtube_service()
    fix_europython_2017_videos(service)
```

Shortcut :-)

Weitere Besonderheiten

- Jeder YT API Aufruf erzeugt "Kosten" in Form von Punkten
- Der API Zugang hat eine Begrenzung in Form von Rate Limits (Anzahl Aufrufe pro Minute/Stunde)
- Limits werden scheinbar nicht streng geprüft
- Tokens können scheinbar auch nach Ablauf erneuert werden

Danke für die Aufmerksamkeit

Beautiful is better than ugly.

Contact

eGenix.com Software, Skills and Services GmbH
Marc-André Lemburg
Pastor-Löh-Str. 48
D-40764 Langenfeld
Germany

eMail: mal@egenix.com
Phone: +49 211 9304112
Fax: +49 211 3005250
Web: http://www.egenix.com/