

Technical Note:

**Working with the Youtube Data API**

Project conducted by:

San Diego State University

5500 Campanile Drive

San Diego, CA 92182-4493

Report prepared by:

Qingyun Zhang

Rick8600@gmail.com

Department of Geography, San Diego State University

11/2/2015

***Abstract***

This report introduces the YouTube Data API, which allows developer to connect to the YouTube database for retrieval of users and videos’ information. This report covers details about how to use the API, describing the API method and the example of retrieving data. There are many methods can be used for searching videos, however, this report will only focus on how to search and obtain information about videos using spatial parameters. The data retrieved from YouTube with the API that is used for visualizing and evaluating the accessibility of videos.

This report covers the following items:

# How we start and register to use the API

# How the parameters work

# How the API retrieve the data and give response

***Working with the YouTube Data API***

Introduction

YouTube is a video website that provides a personalized video clips or collections for its users. It is owned by Google Inc. The YouTube API is the official application programming interface (API) developed and provided by Google. The YouTube API allows developer to access the YouTube database and retrieve information about users or videos. There are six major actions can be performed with the API:

In this report, we will focus on the first function Search for videos in an Area, and demonstrate step by step to obtain the information about videos. To demonstrate the use of YouTube API, we will use ***Python***as the programming language. This report uses the YouTube Data API (v3) Code Samples (<https://developers.google.com/youtube/v3/code_samples>.

) to access the YouTube Data API. Documentation of this can be found at

<https://github.com/youtube/api-samples/blob/master/python/geolocation_search.py>

Here is the A web app that uses the YouTube Data API v3's search functionality to find videos tagged with geo-coordinates.

<https://github.com/youtube/geo-search-tool>.

View the live demo at <http://youtube.github.io/geo-search-tool/search.html>

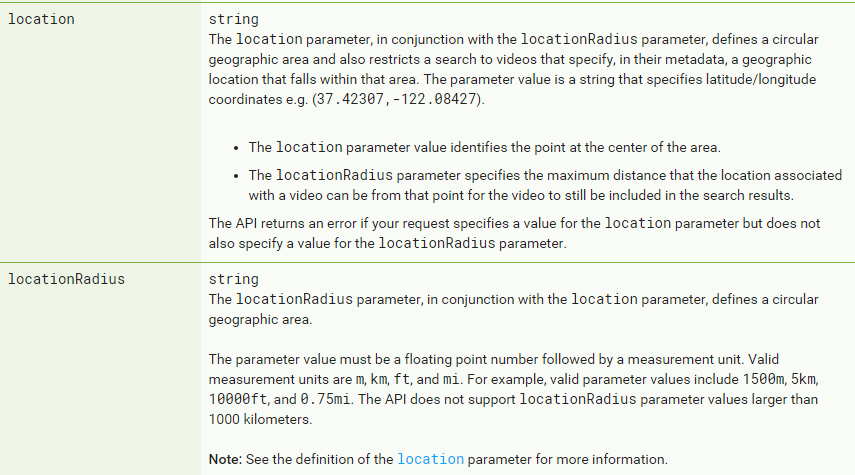
1. You need to register a Google Account to access the Google Developers Console, request an API key, and register your application.
2. Create a project in the Google Developers Console and obtain authorization credentials so your application can submit API requests.
3. After creating your project, make sure the YouTube Data API is one of the services that your application is registered to use:
4. Go to the Developers Console and select the project that you just registered.
5. Open the API Library in the Google Developers Console. If prompted, select a project or create a new one. In the list of APIs, make sure the status is ON for the YouTube Data API v3.
6. If your application will use any API methods that require user authorization, read the authentication guide to learn how to implement OAuth 2.0 authorization.
7. Select a client library to simplify your API implementation.
8. Familiarize yourself with the core concepts of the JSON (JavaScript Object Notation) data format. JSON is a common, language-independent data format that provides a simple text representation of arbitrary data structures. For more information, see json.org.

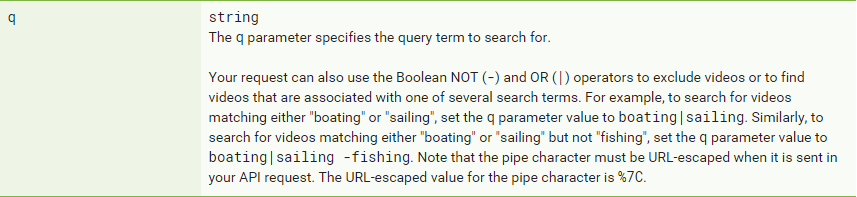
(From Google Developers)

Search for videos in an Area: using the search.list Method

The code sample below calls the API's search.list method with q, location and locationRadius parameters to retrieve search results matching the provided keyword within the radius centered at a particular location. Using the video ids from the search result, the sample calls the API's videos.list method to retrieve location details of each video.

There are many parameters can be used during the search query. Figure 2 lists out several important spatial parameters for the video search method.





**Figure 2**, spatial parameters for the *search.list* method

To demonstrate the use of this method, we perform a geolocational *video search* for video within 5,000 meters radius from the center of Storm Hall at San Diego State University. There are 25 results will be shown. The following Python codes will connect to the Youtube API and execute the *video search*.

#!/usr/bin/python  
  
from apiclient.discovery import build  
from apiclient.errors import HttpError  
from oauth2client.tools import argparser  
  
  
# Set DEVELOPER\_KEY to the API key value from the APIs & auth > Registered apps  
# tab of  
#   https://cloud.google.com/console  
# Please ensure that you have enabled the YouTube Data API for your project.  
DEVELOPER\_KEY = "REPLACE\_ME"  
YOUTUBE\_API\_SERVICE\_NAME = "youtube"  
YOUTUBE\_API\_VERSION = "v3"  
  
def youtube\_search(options):  
  youtube = build(YOUTUBE\_API\_SERVICE\_NAME, YOUTUBE\_API\_VERSION,  
    developerKey=DEVELOPER\_KEY)  
  
  # Call the search.list method to retrieve results matching the specified  
  # query term.  
  search\_response = youtube.search().list(  
    q=options.q,  
    type="video",  
    location=options.location,  
    locationRadius=options.location\_radius,  
    part="id,snippet",  
    maxResults=options.max\_results  
  ).execute()  
  
  search\_videos = []  
  
  # Merge video ids  
  for search\_result in search\_response.get("items", []):  
    search\_videos.append(search\_result["id"]["videoId"])  
  video\_ids = ",".join(search\_videos)  
  
  # Call the videos.list method to retrieve location details for each video.  
  video\_response = youtube.videos().list(  
    id=video\_ids,  
    part='snippet, recordingDetails'  
  ).execute()  
  
  videos = []  
  
  # Add each result to the list, and then display the list of matching videos.  
  for video\_result in video\_response.get("items", []):  
    videos.append("%s, (%s,%s)" % (video\_result["snippet"]["title"],  
                              video\_result["recordingDetails"]["location"]["latitude"],  
                              video\_result["recordingDetails"]["location"]["longitude"]))  
  
  print "Videos:\n", "\n".join(videos), "\n"  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
  argparser.add\_argument("--q", help="Search term", default="Storm Hall SDSU")  
  argparser.add\_argument("--location", help="Location", default="32.776750, -117.074019")  
  argparser.add\_argument("--location-radius", help="Location radius", default="5km")  
  argparser.add\_argument("--max-results", help="Max results", default=25)  
  args = argparser.parse\_args()  
  
  try:  
    youtube\_search(args)  
  except HttpError, e:  
    print "An HTTP error %d occurred:\n%s" % (e.resp.status, e.content)

**Figure 3.** The python code of Geolocation search.

If you have you endpoint setup correctly. You should be able to retrieve a response body like following structure.

{  
  "kind": "youtube#searchListResponse",  
  "etag": ***etag***,  
  "nextPageToken": ***string***,  
  "prevPageToken": ***string***,  
  "pageInfo": {  
    "totalResults": ***integer***,  
    "resultsPerPage": ***integer***  
  },  
  "items": [  
   {  
  "[**kind**](https://developers.google.com/youtube/v3/docs/search#kind)": "youtube#searchResult",  
  "**[etag](https://developers.google.com/youtube/v3/docs/search" \l "etag)**": ***etag***,  
  "[**id**](https://developers.google.com/youtube/v3/docs/search#id)": {  
    "[**kind**](https://developers.google.com/youtube/v3/docs/search#id.kind)": ***string***,  
    "**[videoId](https://developers.google.com/youtube/v3/docs/search" \l "id.videoId)**": ***string***,  
    "**[channelId](https://developers.google.com/youtube/v3/docs/search" \l "id.channelId)**": ***string***,  
    "**[playlistId](https://developers.google.com/youtube/v3/docs/search" \l "id.playlistId)**": ***string***  
  },  
  "[**snippet**](https://developers.google.com/youtube/v3/docs/search#snippet)": {  
    "**[publishedAt](https://developers.google.com/youtube/v3/docs/search" \l "snippet.publishedAt)**": ***datetime***,  
    "**[channelId](https://developers.google.com/youtube/v3/docs/search" \l "snippet.channelId)**": ***string***,  
    "[**title**](https://developers.google.com/youtube/v3/docs/search#snippet.title)": ***string***,  
    "[**description**](https://developers.google.com/youtube/v3/docs/search#snippet.description)": ***string***,  
    "[**thumbnails**](https://developers.google.com/youtube/v3/docs/search#snippet.thumbnails)": {  
      ***(key)***: {  
        "**[url](https://developers.google.com/youtube/v3/docs/search" \l "snippet.thumbnails.(key).url)**": ***string***,  
        "[**width**](https://developers.google.com/youtube/v3/docs/search#snippet.thumbnails.(key).width)": ***unsigned integer***,  
        "[**height**](https://developers.google.com/youtube/v3/docs/search#snippet.thumbnails.(key).height)": ***unsigned integer***  
      }  
    },  
    "**[channelTitle](https://developers.google.com/youtube/v3/docs/search" \l "snippet.channelTitle)**": ***string***,  
    "**[liveBroadcastContent](https://developers.google.com/youtube/v3/docs/search" \l "snippet.liveBroadcastContent)**": ***string***  
  }

  ]  
}

Figure 4. JSON structure of representation.

#kind Identifies the API resource's type.

#etag The Etag of this resource.

#id The id object contains information that can be used to uniquely identify the resource that matches the search request.

#id.kind The type of the API resource.

#id.videoId If the id.type property's value is youtube#video, then this property will be present and its value will contain the ID that YouTube uses to uniquely identify a video that matches the search query.

#id.channelId If the id.type property's value is youtube#channel, then this property will be present and its value will contain the ID that YouTube uses to uniquely identify a channel that matches the search query.

#id.playlistId If the id.type property's value is youtube#playlist, then this property will be present and its value will contain the ID that YouTube uses to uniquely identify a playlist that matches the search query.

#snippet The snippet object contains basic details about a search result, such as its title or description. For example, if the search result is a video, then the title will be the video's title and the description will be the video's description.

#snippet.publishedAt The creation date and time of the resource that the search result identifies. The value is specified in ISO 8601 (YYYY-MM-DDThh:mm:ss.sZ) format.

#snippet.channelId The value that YouTube uses to uniquely identify the channel that published the resource that the search result identifies.

#snippet.title The title of the search result.

#snippet.description A description of the search result.

#snippet.thumbnails A map of thumbnail images associated with the search result. For each object in the map, the key is the name of the thumbnail image, and the value is an object that contains other information about the thumbnail.

#snippet.thumbnails. (key)

Valid key values are:

default – The default thumbnail image for this resource. The default thumbnail for a video – or a resource that refers to a video, such as a playlist item or search result – is 120px wide and 90px tall. The default thumbnail for a channel is 88px wide and 88px tall.

medium – A higher resolution version of the thumbnail image. For a video (or a resource that refers to a video), this image is 320px wide and 180px tall. For a channel, this image is 240px wide and 240px tall.

high – A high resolution version of the thumbnail image. For a video (or a resource that refers to a video), this image is 480px wide and 360px tall. For a channel, this image is 800px wide and 800px tall.

#snippet.thumbnails.(key).url

The image's URL.

#snippet.thumbnails.(key).width

The image's width.

#snippet.thumbnails.(key).height

The image's height.

#snippet.channelTitle

The title of the channel that published the resource that the search result identifies.

#snippet.liveBroadcastContent

An indication of whether a video or channel resource has live broadcast content.

Valid property values are upcoming, live, and none.

For a video resource, a value of upcoming indicates that the video is a live broadcast that has not yet started, while a value of live indicates that the video is an active live broadcast. For a channel resource, a value of upcoming indicates that the channel has a scheduled broadcast that has not yet started, while a value of live indicates that the channel has an active live broadcast.