Android Final Project - API Docs

Mohawk Course Browser

URI : https://csunix.mohawkcollege.ca/~geczy/mohawkprograms.php

Parameters: None

Returns:

_id	type:	number	Unique ID for each record
program	type:	number	ie "559" for Computer Systems Softwr
semesterNum	type:	number	ie "5"
courseCode	type:	string	ie "COMP-10073"
courseTitle	type:	string	ie "Android Application Development"
courseDescription	type:	string	
courseOwner	type:	string	ie "George Geczy"
optional	type:	number	1=optional/elective
hours	type:	number	ie "4" for 4 hours per week

Notes:

Returns full dataset (all records). No Parameters.

Note that course descriptions are still being updated, may be the same as courseTitle for now.

Your app should support program numbers 455, 555, 558, 559 (Mohawk's CSAIT Programs)

Your app should support semesters 1 to 6.

BASEF Project Browser

URI: http://www.basef.ca/api/basefws.php

Parameters: sort=[{"column":"*colname*","direction":"*dir*"}]
filter=[{"type":"*type*","column":"*colname*","value":*value*}]

Example:

http://www.basef.ca/api/basefws.php?

sort=[{"column":"_id","direction":"DESC"}]&filter=[{"type":"number","column":"Year","value":2017}]

Returns:

_id	type: number		Unique ID for each record
Year	type: number		ie "2017"
ProjectNum	type: string		ie “B10”
NameFull	type: string		ie “John Smith”
NameFull2	type: string		ie “Bob Jones”
NameBoth	type: string		ie “John Smith, Bob Jones”
Level	type: string		ie “Junior 7/8”
SchoolName	type: string		ie “Ancaster Meadows”
SchoolBoard	type: string		ie “HWDSB”
SchoolCity	type: string		ie “Ancaster”
Title	type: string		
Description	type: string		
picURI	type: string		URL to jpg or empty string

Notes:

picURI may return an empty string (""), so be sure to check for that. Otherwise it should return a URL for a .jpg picture of the project.

NameFull2 may also be an empty string if the project was a single (not a pair) project. You can search using “NameBoth” to find a match to either of two student names.

All strings are searched as a case-insensitive partial match (ie “sm” matches “Smith” and “Jaismir”)

Multiple filters are supported (JSON encoding), such as:

filter=[{"type":"number","column":"Year","value":2017},
{"type":"string","column":"NameFull","value":"mogus"}]

Open Movie Database Browser

Information: http://www.omdbapi.com/

You will need to register to get a free API Access key under the “API Key” tab.

Examples:

Query title:

http://www.omdbapi.com/?apikey=xxxxx&s=music

Find by ID:

http://www.omdbapi.com/?apikey=xxxxx&i=tt0059742

Google Books Browser

Information: https://developers.google.com/books/docs/overview

https://developers.google.com/books/docs/v1/using#WorkingVolumes

Examples:

https://www.googleapis.com/books/v1/volumes?q=intitle:android

https://www.googleapis.com/books/v1/volumes?q=intitle:android+inauthor:Conder

(other supported queries: inpublisher:, subject:, isbn:)

You can add “&projection=lite” to the url to return a bit less data

In this case of the Google Books API and the Movie DB API, you will get a return of a single master object that contains one or more arrays of data. You will need to create nested classes to retrieve the data.

For example, for the Google Books API return, the JSON looks as follows:

```
{
 "kind": "books#volumes",
 "totalItems": 2319,
 "items": [
  {
   "kind": "books#volume",
   "id": "2BMDuAEACAAJ",
   "etag": "7Q7Hz1v4sGk",
   "volumeInfo": {
    "title": "Harry Potter and the Cursed Child",
    "subtitle": "Parts One and Two Playscript",
    "authors": [
     "J. K. Rowling",
     "Jack Thorne",
     "John Tiffany"
    ],
    "publisher": "Arthur A. Levine Books",
    "description": "The Eighth Story. Nineteen Years Later.”
   }
  },
  { ... another item here ... } ]
}
```

Notice that the primary return has three items – “kind”, “totalItems”, and an array of “items”. The “items” object includes a number of objects, including a “volumeInfo” object with a number of contained fields. Some of those fields (such as “authors”) are also variable sized arrays as well.

The classes to decode this JSON would like the following (only a few fields are included here):

```
static public class VolumeInfo {
    public String   title;
    public ArrayList<String> authors;
    public String   description;
}

static public class Items {
    public String   kind;
    public String   id;
    public String   etag;
    public VolumeInfo   volumeInfo;

    @Override
    public String toString() {
        return kind + " " + id + "\n" + volumeInfo.title + "\n";
    }
}

static public class Books {
    public String   kind;
    public int      totalItems;
    public ArrayList<Items> items;
}
```

You could retrieve the JSON data using the GSON Library as follows, and add it to an ArrayAdapter:

```
Gson gson = new Gson();
Books books = gson.fromJson(s, Books.class);
ArrayAdapter<Items> adapter =
        new ArrayAdapter<Items>(MainActivity.this,
                android.R.layout.simple_list_item_1, books.items);
```