Branch: master ▾ Find file Copy path

buckets / **bucket_collection.py**

trodicaro Finish test of dup keys to be empty

aadfbd5 on Mar 9, 2018

1 contributor

Raw Blame History

80 lines (63 sloc) 3.14 KB

```
import csv
import json
import inspect
from random import randint

class Bucket:
    # https://pythonconquerstheuniverse.wordpress.com/2012/02/15/mutable-default-arguments/
    def __init__(self, case_preserving_key = ""):
        self.case_preserving_key = case_preserving_key
        self.purchases = []

class BucketCollection:
    def __init__(self, buckets_file_name, purchases_file_name):
        self.buckets = {}

        self.buckets['*,*,*'] = Bucket('*,*,*')

        with open(buckets_file_name) as buckets_file:
            readCSV = csv.reader(buckets_file)

            for row in readCSV:
                current_key = ",".join([row[0],row[1],row[2]])
                if current_key.upper() in self.buckets:
                    current_key += '-dup' + randint(1, 9999).__str__()
                bucket = Bucket(current_key)
                # losing the original key here
                self.buckets[current_key.upper()] = bucket

        self.populate_buckets(purchases_file_name)

    def to_json(self):
        results = []

        for key, bucket in self.buckets.items():
              current_group = {}
              json_key = bucket.case_preserving_key
              if "-dup" in json_key:
                  json_key = json_key.split("-dup")[0]
              current_group["bucket"] =  json_key
              current_group["purchases"] =  bucket.purchases
              results.append(current_group)

        return results

    def to_file(self, result_file_name):
      results_file = open(result_file_name, 'w')
      results_file.write(json.dumps(self.to_json(), indent = 4, sort_keys = True))
      results_file.close()

    # make private
    def populate_buckets(self, purchases_file_name):
        with open(purchases_file_name) as purchases_file:
            readCSV = csv.reader(purchases_file)

            for row in readCSV:
                order_id = row[0]
                publisher = row[2]
                price = row[4]
                duration = row[5]
                key = ",".join([publisher, price, duration])

                complete_key = ",".join([publisher, price, duration]).upper()
                publisher_duration_key = ",".join([publisher, "*", duration]).upper()
                publisher_price_key = ",".join([publisher, price, "*"]).upper()
                price_duration_key = ",".join(["*", price, duration]).upper()
                publisher_only = ",".join([publisher, "*", "*"]).upper()
                duration_only_key = ",".join(["*", "*", duration]).upper()
                price_only_key = ",".join(["*", price, "*"]).upper()
                catch_all_key = ",".join(["*","*","*"]).upper()

                possible_keys = [complete_key, publisher_duration_key, publisher_price_key, price_duration_key, publisher_

                for possible_key in possible_keys:
                    #I don't think following line is efficient; alternatives?
                    if possible_key in self.buckets.keys():
                        stringified_record = ",".join(map(str, row))
                        self.buckets[possible_key].purchases.append(stringified_record)
                        break

```