CS-340-R1891 Project One

By Ryan LeChien

Table of Contents

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

Python-MongoDB CRUD README

About the Project/Project Title

This project provides CRUD functionality via a Python API for a MongoDB database.

Motivation

This project exists to provide a programmatic Python API for CRUD functionality.

Getting Started

To get a local copy up and running, download the *animal_shelter.py* script and place it in the directory of your project. Within your project, you can import it via the statement:

```
from animal_shelter import *
```

Installation

This project was built in Python version 3.10.7. The *animal_shelter.py* script must be placed in the same directory of whichever project calls for this library.

Usage

Available Methods

- *__init__(self)*: The initializer method
- *create(self, data)*: The *data* parameter is a JSON object (parsed as a Python tuple). This method creates one.
- *read(self, data)*: The *data* parameter is a JSON object (parsed as a Python tuple). This method finds one.
- *update(self, filt, data)*: The *data* parameter is a JSON object (parsed as a Python tuple); the *filt* parameter specifies the matching filter. This method updates one.
- *delete(self, data)*: The *data* parameter is a JSON object (parsed as a Python tuple). This method deletes one, specifically the first one found.

Code Example

```
# The C in CRUD.
# creates ONE
def create(self, data):
    if data is not None:
        inserted = self.database.animals.insert_one(data)  # data should be dictionary

        return inserted != 0

    else:
        raise Exception("Nothing to create, because data parameter is empty")
```

The implementation of the method *create()*

```
    # The U in CRUD.
    # updates ONE
    def update(self, filt, data):
        if data is not None:
            self.database.animals.update_one(filt, data)  # data should be dictionary

            return 1

        else:
            raise Exception("Nothing to update, because data parameter is empty")
```

The implementation of the method *update()*

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

Screenshots of Testing with Jupyter

```
In [1]: from animal_shelter import *

In [2]: crudder = AnimalShelter() # username and password hard-coded in class

In [3]: print(crudder.create({"animal_id" : "1", "breed" : "clone"})) #make a test animal
        True

In [4]: print(crudder.create({"animal_id" : "2", "breed" : "clone"})) #make a second test animal
        True

In [5]: crudder.read({"breed" : "clone"}) #find the test animals
        {'animal_id': '1', 'breed': 'clone'}
        {'animal_id': '2', 'breed': 'clone'}

In [6]: crudder.update({"animal_id" : "2"}, {"$set" : {"breed" : "mutant"}}) #update one (the second) test animal's breed
Out[6]: 1
```

The Python CRUD module was tested in the Jupyter notebook per the following steps: import *animal_shelter* module; instantiate the AnimalShelter class; **create** two animal entries, with different *animal_id*s but of the same breed (“clone“); **read** all the animals of type ”clone”; **update** the *breed* of the animal with *animal_id* 2 to ”mutant”; read the animals of type “clone” and “mutant,” verifying the update; **delete** the animal with *animal_id* 2; read the animals of type “clone” and “mutant” again, verifying the deletion of “mutant” animal from the collection. These steps are illustrated in screenshots above and below.

```
In [7]: crudder.read({"breed" : "clone"}) #find the test animal
        {'animal_id': '1', 'breed': 'clone'}

In [8]: crudder.read({"breed" : "mutant"}) #find the mutant animal
        {'animal_id': '2', 'breed': 'mutant'}

In [9]: crudder.delete({"animal_id" : "2"}) #delete the mutant animal
Out[9]: 1

In [10]: crudder.read({"breed" : "clone"}) #find the original test animal in the collection
         {'animal_id': '1', 'breed': 'clone'}

In [11]: crudder.read({"breed" : "mutant"}) #look for the deleted mutant in the collection (no results because it's deleted)

In [ ]:
```

Roadmap/Features

- Full Python CRUD API for a MongoDB database
- Ability to pass credential arguments to the constructor to allow connection to different databases

Contact

Ryan LeChien
ryan.lechien@snhu.edu

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

Module Three Milestone

Part I: Importing and Indexing a Data Set

```
(base) ryanlechien_snhu@nv-snhu7-102:~$ script moduleThree.txt
Script started, output log file is 'moduleThree.txt'.
(base) ryanlechien_snhu@nv-snhu7-102:~$ cd /usr/local/datasets/
(base) ryanlechien_snhu@nv-snhu7-102:/usr/local/datasets$ mongoimport --username="${MONGO_USER}" --passwor
d="${MONGO_PASS}" --port=${MONGO_PORT} --host=${MONGO_HOST} --db AAC --collection animals --authentication
Database admin --type=csv --file=aac_shelter_outcomes.csv --headerline
2023-09-17T04:46:45.135+0000    connected to: mongodb://nv-desktop-services.apporto.com:31314/
2023-09-17T04:46:46.224+0000    10000 document(s) imported successfully. 0 document(s) failed to import.
(base) ryanlechien_snhu@nv-snhu7-102:/usr/local/datasets$
```

Start a script output file, change the directory, and import the CSV file

```
AAC> db.animals.createIndex({"breed" : 1})
breed_1
AAC> db.animals.find().explain()
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'AAC.animals',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: '17830885',
    planCacheKey: '17830885',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: { stage: 'COLLSCAN', direction: 'forward' },
    rejectedPlans: []
  },
  command: { find: 'animals', filter: {}, '$db': 'AAC' },
  serverInfo: {
    host: 'csdev-mongodb-75c4b75ffb-dsfpg',
    port: 27017,
    version: '6.0.4',
    gitVersion: '44ff59461c1353638a71e710f385a566bcd2f547'
  },
  serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
    internalQueryProhibitBlockingMergeOnMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600
  },
```

Create a simple index on the key *breed*, then explain the finding from the collection *animals*

```
AAC> db.animals.dropIndex({"breed" : 1})
{ nIndexesWas: 2, ok: 1 }
```

Drop the simple index

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

```
AAC> db.animals.createIndex({"breed" : 1, "outcome_type" : 1})
breed_1_outcome_type_1
AAC> db.animals.find().explain()
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'AAC.animals',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: '17830885',
    planCacheKey: '17830885',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: { stage: 'COLLSCAN', direction: 'forward' },
    rejectedPlans: []
  },
  command: { find: 'animals', filter: {}, '$db': 'AAC' },
  serverInfo: {
    host: 'csdev-mongodb-75c4b75ffb-dsfpg',
    port: 27017,
    version: '6.0.4',
    gitVersion: '44ff59461c1353638a71e710f385a566bcd2f547'
  },
  serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
    internalQueryProhibitBlockingMergeOnMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600
  },
  ok: 1
```

Create a compound index and explain a finding

```
AAC> db.animals.dropIndex({"breed" : 1, "outcome_type" : 1})
{ nIndexesWas: 2, ok: 1 }
```

Drop the compound index

Part II: User Authentication

```
AAC> use admin
switched to db admin
admin>
```

Switch to *admin* database

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

```
admin> db.createUser({user : "aacuser", pwd : "letmein", roles : [{role : "readWrite", db : "aac"}]})
{ ok: 1 }
admin>
```

Create user *aacuser* with read-write privileges

```
(base) ryanlechien_snhu@nv-snhu7-102:~$ MONGO_USER=aacuser
(base) ryanlechien_snhu@nv-snhu7-102:~$ MONGO_PASS=letmein
(base) ryanlechien_snhu@nv-snhu7-102:~$ mongosh
Current Mongosh Log ID: 65069805964f6763043338ce
Connecting to:          mongodb://<credentials>@nv-desktop-services
:1314/?directConnection=true&appName=mongosh+1.8.0
Using MongoDB:          6.0.4
Using Mongosh:          1.8.0
```

Login using new account credentials

```
admin> db.getUsers()
{
  users: [
    {
      _id: 'admin.aacuser',
      userId: new UUID("badc35f4-e54f-4ea5-9566-2d6e5aeb319e"),
      user: 'aacuser',
      db: 'admin',
      roles: [ { role: 'readWrite', db: 'aac' } ],
      mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
    },
    {
      _id: 'admin.root',
      userId: new UUID("c3366109-3a48-4d28-84d4-66e762fad86b"),
      user: 'root',
      db: 'admin',
      roles: [ { role: 'root', db: 'admin' } ],
      mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
    }
  ],
  ok: 1
}
admin>
```

From the *admin* terminal, verify both users are using *admin*

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.

```
admin> db.runCommand({connectionStatus : 1})
{
  authInfo: {
    authenticatedUsers: [ { user: 'root', db: 'admin' } ],
    authenticatedUserRoles: [ { role: 'root', db: 'admin' } ]
  },
  ok: 1
}
admin>
```

Connection status from the *admin* terminal

```
admin> db.runCommand({connectionStatus : 1})
{
  authInfo: {
    authenticatedUsers: [ { user: 'aacuser', db: 'admin' } ],
    authenticatedUserRoles: [ { role: 'readWrite', db: 'aac' } ]
  },
  ok: 1
}
admin>
```

Connection status from the *aacuser* terminal

Note: This template has been adapted from the following sample templates: Make a README, Best README Template, and A Beginners Guide to Writing a Kickass README.