Jul 15, 2024
Full Foundations
Average: 100.00%

0x02. AirBnB clone - MySQL

Group project | Python | OOP | Back-end | SQL | MySQL | ORM | SQLAlchemy

Background Context

Environment variables will be your best friend for this project

- `export environment=development` It can be "dev" or "test" for the moment ("production" soon)
- `export mysql_user` the username of your MySQL
- `export mysql_pwd` the password of your MySQL
- `export mysql_host` the hostname of your MySQL
- `export mysql_db` the database name of your MySQL
- `export mysql_type_storage` the type of storage used. It can be "file" (using `FileStorage`) or `db` (using `DBStorage`)

Resources

Read or watch:

- Joins tutorial
- Python MySQL connector page
- pep8 style guide
- mysql-hexagon
- SQL Alchemy tutorial
- How To Create a New User and Grant Permissions in MySQL
- Object-relational mapping in enterprise applications
- What ORM means
- MySQL 8.0 SQL Statement Syntax

Learning Objectives

At the end of this project, you are expected to be able to **explain to anyone, without the help of Google:**

General

- What is unit testing and how to implement it in a large project
- What is `pep8`, and how to use it
- What is `pep8` and how to use it
- How to create a MySQL database from a function
- How to create a MySQL database
- How to create a MySQL user and grant it privileges
- What ORM means
- How to map a class to a MySQL table
- How to handle 2 different storage engine with the same database
- How to use environment variables

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else's work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

Python Scripts

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be interpreted/compiled on Ubuntu 20.04 LTS using python3 (version 3.8.5)
- All your files should end with a new line
- The first line of all your files should be exactly `#!/usr/bin/python3`
- A `README.md` file, at the root of the folder of the project, is mandatory
- Your code should use the `pep8` style (version 1.7.x)
- All your files must be executable
- The length of your files will be tested using `wc`
- All your modules should have documentation (`python3 -c 'print(import module.__doc__)'`)
- All your classes should have documentation (`python3 -c 'print(module.ClassName.__doc__)'`)
- All your functions (inside and outside a class) should have documentation (`python3 -c 'print(module.ClassName.fun.__doc__)'` and `python3 -c 'print(module.fun.__doc__)'`)
- A documentation is not a simple word, it's a real sentence explaining what's the purpose of the module, class or method (the length of it will be verified)

Python Unit Tests

- Allowed editors: `vi`, `vim`, `emacs`
- All your files should end with a new line
- All your test files should be inside a folder `tests`
- You have to use the `unittest` module
- All your test files should be executable (files extension `.py`)
- All your test files should have documentation (`python3 -c 'print(module.__doc__)'`)
- Your file organization in the tests folder should be the same as your project: ex: `models/base_model.py` unit tests must be in `tests/test_models/test_base_model.py`
- All your tests should be executed by using this command: `python3 -m unittest discover tests`
- or `python3 -m unittest tests/test_models/test_base_model.py` to run only one test file: `python3 -m unittest tests/test_models/test_base_model.py`
- All your modules should have documentation (`python3 -c 'print(module.__doc__)'`)
- All your classes should have documentation (`python3 -c 'print(module.ClassName.__doc__)'`)
- All your functions (inside and outside a class) should have documentation (`python3 -c 'print(module.ClassName.fun.__doc__)'` and `python3 -c 'print(module.fun.__doc__)'`)
- We strongly encourage you to work together on test cases, so find you don't miss any edge case

SQL Scripts

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be executed on Ubuntu 20.04 LTS using `MySQL 8.0`
- Your files will be executed with `cat filename | mysql -uroot -p pwd_root`
- All your files should end with a comment: `-- my first comment`
- You have to use the `pep8` style (version 1.1.x)
- The length of your files will be tested using `wc`
- All your files should start by a comment that explains what the purpose of the file is
- All your queries should have comment
- Your file organization in the tests folder should be the same as your project: ex: `models/base_model.py` unit tests must be in `tests/test_models/test_base_model.py`
- We strongly encourage you to work together on test cases, so find you don't miss any edge case

More Info

Comments for your SQL file:

```
# cat my_script.sql
-- Show all databases in the back DB
-- Session Batch 1 is the best
SELECT id, name FROM states; WHERE batch_id = 1 ORDER BY created_at DESC LIMIT 5;
```

Video library (1 video)

Search by title

- HBNB - Storage abstraction — 40:00
- Airbnb Console — 40:00

HBNB - storage abstraction | **Airbnb console**

Tasks

0. Fork me if you can!

mandatory

Score: 100.0% (Checks completed: 100.0%)

In the industry, you will work on an existing codebase 90% of the time. Your first thoughts upon looking at it might include:

- "Who did this code?"
- "How it works?"
- "Where are unittests?"
- "Where is this?"
- "Why is it done like this?"
- "I don't understand why"
- "... I will refactor everything..."

But the worst thing you could possibly do is to **redo everything**. Please don't do that! **Note: the existing codebase might be perfect, or it might have errors. Don't always start from the existing codebase!**

For this project you will fork this codebase:

- update the repository name to `AirBnB_clone`
- update the `README.md` with your information **but don't delete the initial authors**

If you are the owner of this repository, please create a new repository named `AirBnB_clone_v3` with the same content of `AirBnB_clone`

Repo:

- GitHub repository: `AirBnB_clone_v3`

Done! | Check your code | Ask for a new correction | Get a sandbox | QA Review

1. Bug free!

mandatory

Score: 38.17% (Checks completed: 81.7%)

Do you remember the `pycodestyle` module?

This codebase contains many test cases. Some are missing, but the ones included cover the basic functionality of the program.

```
pycodestyle*.py tests/*/*.py python3 -m unittest discover tests | tail -n1 | wc -l
0
pycodestyle*.py tests/*/*.py python3 -m unittest discover tests | tail -n1
OK
pycodestyle*.py tests/*/*.py
```

All your unittests **must** pass without any errors at anytime in this project, **with each storage engine**. Same for PEP8

Some tests won't be relevant for some type of storage, please ignore them by using the `skipUnless` feature of the **Unittest module - 26.3.5. Disparate test organization**. Of course, the number of tests must be higher than the current number of tests, so if you decide to skip a test, you should write a new test!

How to test with MySQL?

First, you need to create a MySQL database for testing. After, you have to remember what the purpose of an unittest:

"Assert a current state (objects/tables/database), do an action, and validate this action changed (or not) the state of your objects/tables/ database"

For example, you want to validate that the `BaseModel.save()` `save()` method `pycodestyle*.py` command in the console will add a new record in your table `models` in your database. Four steps for your unittest:

- get the number of current records in the table `models`: this using a `SELECT`, for example - but not SQL/Alchemy (remember, you want to test if it works, so it's better to code by hand)
- do an action: `save()` method
- get the number of current records in the table `models`: same method, with `SELECT`
- if the difference is `+1`, it's test passed!

Repo:

- GitHub repository: `AirBnB_clone_v3`
- File: `tests/test_console.txt`

Done! | Check your code | Ask for a new correction | Get a sandbox | QA Review

2. Console improvements

mandatory

Score: 77.27% (Checks completed: 77.27%)

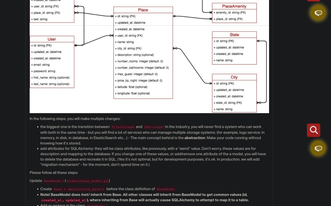
Update the `do_create(self, arg)` function of your command interpreter (`console.py`) to allow for object creation with given parameters:

- Command syntax: `Create <class name> name=value1 number=value2 boolean=value3`
- Name rules:
  - Key name must be valid
  - String: `"<key>"` => starts with a double quote
    - any double quote inside the value must be escaped with a backslash
    - all underscores `_` must be replace by spaces. Example: You want to set the string `my little book` to the attribute `name` => your command line must be: `name="my_little_book"`
  - Float: `<key>=<unsigned number>.<unsigned number>` => dot/period `.` is required
  - Integer: `<key>=<unsigned number>` => default case
- If any parameter doesn't fit with these requirements or can't be recognized correctly by your program, it must be skipped

Don't forget to add tests for this new feature!

Also, this feature will be tested here only with `BaseModel` engine.

```
pycodestyle*.py tests/*/*.py vi tests/test_console.txt
pycodestyle*.py tests/*/*.py python3 -m unittest discover tests
create Base name="my_little_book" number_pages=65536 numbers_authors=3 has_paper=False price_by_chapter=50 rating=9.75 created=1412147
all Base
pycodestyle*.py tests/*/*.py vi tests/test_console.txt console.py
echo "create Base name=hbtn0123456789 id=983768540987654321987654321 created=1412147 updated=1412147 extra_float=3.14159265358979 extra_int=9 extra_str=\"I'm_a_string\" | python3 -u console.py | cat -e
echo "create Base name=hbtn0123456789 id=983768540987654321987654321 created=1412147 updated=1412147 extra_float=3.14159265358979 extra_int=9 extra_str=\"I'm_a_string\" | python3 -u console.py | cat -e
echo "create Base name=\"hbtn0123456789\"
```



In the following slides, you will make multiple changes:

- Site Upgrade: `upgrade_site_to_v2.php`
- Create `User\Table\UserTable` and `User\Entity\User` to the already you will write that in the system which can serve `Place` and `PlaceAnarchy` in the same time - but you will find a lot of services who can manage multiple storage systems (for example, logs service: in memory, in disk, in database, in Elasticsearch, in Redis, in Amazon Cloudwatch, in Prometheus). **Be pragmatic**: Make your code running without knowing how it is stored.
- Set attributes for SQL Alchemy: they will be stored in the table `properties`, with a "weird" value. Don't worry these values are for `Place` and `PlaceAnarchy`, but it will be a good example of how to deal with a set/get/check/unset attributes in an attribute of the a model, you will have to delete the database and recreate it in SQL. Yes, it's not optimal, but for development purposes, it's ok. In production, we will add "migration functions" for the attributes. (don't spend time on it)

Please follow all these steps:

Update `upgrade_site_to_v2.php` and `setup_v2.sql`:

- Create `User\Table\UserTable` and `User\Entity\User` before the class definition of `PlaceTable`.
- **Make! Sure!!!

Many-to-Many

To make this link work, we will create a third table called `place_amenity` that will create these links.

And you are good, you have a new engine!

```
pulisamk@ubuntu:~/hbnb_v23 cat main_place_amenities.py
#!/usr/bin/python3
""" Test link Many-To-Many Place <> Amenity
"""
from models import *

# creation of a place
place = Place(name="Paris Plaza")
place.save()

# creation of a city
city = City(state_id=state.id, name="San Francisco")
city.save()

# creation of a user
user = User(email="jodhnson.com", password="jodpwd")
user.save()

# creation of 2 places
place_1 = Place(name_place=lis, city_id=city.id, name="House 1")
place_1.save()
place_2 = Place(name_place=lis, city_id=city.id, name="House 2")
place_2.save()

# creation of 3 various Amenity
amenity_1 = Amenity(name="Wifi")
amenity_1.save()
amenity_2 = Amenity(name="Cable")
amenity_2.save()
amenity_3 = Amenity(name="Cool")
amenity_3.save()

# link place_1 with 2 amenities
place_1.amenities.append(amenity_1)
place_1.amenities.append(amenity_2)

# link place_2 with 3 amenities
place_2.amenities.append(amenity_1)
place_2.amenities.append(amenity_2)
place_2.amenities.append(amenity_3)

storage.save()

print("OK")

pulisamk@ubuntu:~/hbnb_v23
pulisamk@ubuntu:~/hbnb_v23 HBNB_MYSQL_USER=hbnb_dev HBNB_MYSQL_PWD=hbnb_dev_pwd HBNB_MYSQL_HOST=localhost
HBNB_MYSQL_DB=hbnb_dev_db HBNB_TYPE_STORAGE=db ./main_place_amenities.py
OK
pulisamk@ubuntu:~/hbnb_v23
pulisamk@ubuntu:~/hbnb_v23 echo "SELECT * FROM amenities;" | mysql -uhbnb_dev -p hbnb_dev_db
Enter password:
*************************** 1. row ***************************
        id: 4710ca40-12be-404f-954a-445ae47a4d39
created_at: 2017-11-10 06:22:33
updated_at: 2017-11-10 06:22:33
      name: Cable
*************************** 2. row ***************************
        id: 4a4c7e7c-a65-4ce8-81c9-832f89e8ac4
created_at: 2017-11-10 06:22:33
updated_at: 2017-11-10 06:22:33
      name: Wifi
*************************** 3. row ***************************
        id: b93ee02-d6c8-43ba-847c-325d7215d8e4
created_at: 2017-11-10 06:22:33
updated_at: 2017-11-10 06:22:33
      name: Wifi
pulisamk@ubuntu:~/hbnb_v23
pulisamk@ubuntu:~/hbnb_v23 echo "SELECT * FROM places;" | mysql -uhbnb_dev -p hbnb_dev_db
Enter password:
*************************** 1. row ***************************
        id: 8d7b7c7-3b69-5385-8ccb-a12f8b4ada
   created_at: 2017-11-10 06:22:33
   updated_at: 2017-11-10 06:22:33
      city_id: 9a8b4dac-3137-4bbc-8142-04e0f8a17ee
      user_id: 4a4c7e7c-a65-4ce8-a01f-47a4f5ba1e9
         name: House 1
  description: NULL
    number_rooms: 0
number_bathrooms: 0
     max_guest: 0
price_by_night: 0
     latitude: NULL
    longitude: NULL
*************************** 2. row ***************************
        id: 8d7b7c7-3b69-5385-8ccb-a12f8b4ada
   created_at: 2017-11-10 06:22:33
   updated_at: 2017-11-10 06:22:33
      city_id: 9a8b4dac-3137-4bbc-8142-04e0f8a17ee
      user_id: 4a4c7e7c-a65-4ce8-a01f-47a4f5ba1e9
         name: House 2
  description: NULL
    number_rooms: 0
number_bathrooms: 0
     max_guest: 0
price_by_night: 0
     latitude: NULL
    longitude: NULL
*************************** 3. row ***************************
        id: 8d7b7c7-3b69-5385-8ccb-a12f8b4ada
   created_at: 2017-11-10 06:22:33
   updated_at: 2017-11-10 06:22:33
      city_id: 9a8b4dac-3137-4bbc-8142-04e0f8a17ee
      user_id: 4a4c7e7c-a65-4ce8-a01f-47a4f5ba1e9
         name: House 3
  description: NULL
    number_rooms: 0
number_bathrooms: 0
     max_guest: 0
price_by_night: 0
     latitude: NULL
    longitude: NULL
pulisamk@ubuntu:~/hbnb_v23
pulisamk@ubuntu:~/hbnb_v23 echo "SELECT * FROM place_amenities;" | mysql -uhbnb_dev -p hbnb_dev_db
Enter password:
*************************** 1. row ***************************
place_id: 8d7b7c7-3b69-5385-8ccb-a12f8b4ada
amenity_id: 4710ca40-12be-404f-954a-445ae47a4d39
*************************** 2. row ***************************
place_id: 8d7b7c7-3b69-5385-8ccb-a12f8b4ada
amenity_id: 4a4c7e7c-a65-4ce8-81c9-832f89e8ac4
*************************** 3. row ***************************
place_id: 8d7b7c7-3b69-5385-8ccb-a12f8b4ada
amenity_id: b93ee02-d6c8-43ba-847c-325d7215d8e4
*************************** 4. row ***************************
place_id: 8d7b7c7-3b69-5385-8ccb-a12f8b4ada
amenity_id: b93ee02-d6c8-43ba-847c-325d7215d8e4
*************************** 5. row ***************************
place_id: 8d7b7c7-3b69-5385-8ccb-a12f8b4ada
amenity_id: b93ee02-d6c8-43ba-847c-325d7215d8e4
pulisamk@ubuntu:~/hbnb_v23
```

Repo:

- GitHub repository: `hbnb_clone_v2`
- File: `main_place_amenities.py`

Done! | Check your code | Ask Review

```
*************************** 1. row ***************************
        id: b93ee02-d6c8-43ba-847c-325d7215d8e4
created_at: 2017-11-10 06:22:33
updated_at: 2017-11-10 06:22:33
      name: Wifi
pulisamk@ubuntu:~/hbnb_v23
```

Repo:

- GitHub repository: `hbnb_clone_v2`
- File: `main_place_amenities.py`

Done! | Check your code | Ask Review

Copyright © 2018 ALX, All rights reserved