tp_2_configurations

In this TP we'll use MongoDB with the python driver PyMongo.

Overview

We are going to have series of README instructions to be able to setup our MFLIX application successfully. With MFLIX application, you will learn to:

- create and share a database connection,
- perform the basic Create, Read, Update, and Delete operations through the driver,
- utilize the MongoDB best practices and more.

Mflix is composed of two main components:

- *Frontend*: All the UI functionality is already implemented for you, which includes the built-in React application that you do not need to worry about.
- *Backend*: The project that provides the necessary service to the application. The code flow is already implemented except some functions.

You'll only be implementing the functions which directly call to MongoDB.

Summary

- README.md file contains detailed setup instructions
- API layer is implemented by `movies.py` and `user.py` in the **mflix/api** folder
  - Do not modify either of these files
- `db.py` file contains all methods that interact with the database
  - Modify this file to implement required functionality
- **tests** directory contains all unit tests
  - Run these tests as you go
  - We recommend you focus on making tests pass one by one rather than trying to make all tests in the suite pass at once.

Database Layer

We will be using *MongoDB Atlas*, MongoDB's official Database as a Service (DBaaS), so you will not need to manage the database component yourself.

In order to run properly, the MFlix software project has some installation requirements and environmental dependencies.

Download the mflix-python.zip file

You can download the **mflix-python.zip** file by clicking the link in the "Handouts" section of this page. Downloading this handout may take a few minutes. When the download is complete, unzip the file and cd into the project's root directory, **mflix-python**.

```
cd ~/Downloads
unzip mflix-python.zip
cd mflix-python
```

Project Structure

Everything you will implement is located in the `mflix/db.py` file, which contains all database interfacing methods. The API will make calls to db.py to interact with MongoDB.

The unit tests in tests will test these database access methods directly, without going through the API. The UI will run these methods in integration tests, and therefore requires the full application to be running.

The API layer is fully implemented, as is the UI. If you need to run on a port other than 5000, you can edit the **index.html** file in the build directory to modify the value of **window.host**.

Please do not modify the API layer in any way, movies.py and user.py under the **mflix/api** directory. Doing so will most likely result in the frontend application failing to validate some of the labs.

Local Development Environment Configuration

IDE

Choose an IDE, for exercices part. I would recommend **PyCharm Community Edition**.

Anaconda

We're going to use Anaconda to install Python 3 and to manage our Python 3 environment.

Once Anaconda is installed, you will have to create and enable a conda environment.

```
# enter mflix-python folder
cd mflix-python

# create a new environment for MFlix
conda create --name mflix

# activate the environment
activate mflix
```

You can deactivate the environment with the following command:

```
deactivate
```

Python Library Dependencies

Once the Python 3 environment is activated, we need to install our python dependencies. These dependencies are defined in the requirements.txt file, and can be installed with the following command:

```
pip install -r requirements.txt
```

Running the Application

In the mflix-python directory you can find a file called dotini.

Open this file and enter your Atlas SRV connection string as directed in the comment. This is the information the driver will use to connect. Make sure **not** to wrap your Atlas SRV connection between quotes:

```
MFLIX_DB_URI = mongodb+srv://...
```

Rename this file to .ini with the following command:

```
mv dotini_unix .ini  # on Unix
ren dotini_win .ini # on Windows
```

Note: Once you rename this file to .ini, it will no longer be visible in Finder or File Explorer. However, it will be visible from Command Prompt or Terminal, so if you need to edit it again, you can open it from there:

```
vi .ini       # on Unix
notepad .ini  # on Windows
```

To start MFlix, run the following command:

```
python run.py
```

This will start the application. You can then access the MFlix application at http://localhost:5000/.

Running the Unit Tests

To run the unit tests for this course, you will use pytest and needs to be run from mflix-python directory. Each course lab contains a module of unit tests that you can call individually with a command like the following:

```
pytest -m LAB_UNIT_TEST_NAME
```

Each ticket will contain the command to run that ticket's specific unit tests. For example to run the Connection Ticket test your shell command will be:

```
pytest -m connection
```