Flask Templates and Data with Bootstrap

Overview

For this assignment, your goal is setup a Bootstrap HTML template using Flask Templates and use it to display data retrieved from your database.

The completed code for this project can be found here:
https://github.com/kaw393939/PythonDockerFlaskPycharm

Readings and Videos

1. Lecture Video https://youtu.be/tylzleJDlkc
2. Getting started and Reference for Bootstrap https://getbootstrap.com/docs/4.4/getting-started/introduction/
3. Tutorials this is based on: https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-ii-templates
4. Documentation for Flask Templates https://flask.palletsprojects.com/en/1.1.x/tutorial/templates/

Optional Content – From other courses that is relevant for review or more information

1. IS117 – HTML / Bootstrap - This is not going to work with BrowserSync and how I have it setup here for you. This is setup for using WebStorm and geared for front end development. However, the last 3 videos about Bootstrap will give more information that you can use to make a site and understand HTML/Bootstrap better.
    a. https://github.com/kaw393939/bootstrap-4-boilerplate
2. IS219 – JavaScript – This course is a continuation of the above and adds JavaScript.
    a. https://github.com/kaw393939/is219Project2

Pre-Requisites

1. *Windows 10 Pro, Windows 10 Education Edition, MacOS*
2. *PyCharm Installed*
3. *Docker Desktop Installed*

Project Steps

Step 1 – Create a new project with PyCharm

Step 2 – Create the following files and folders within the root directory of your project:

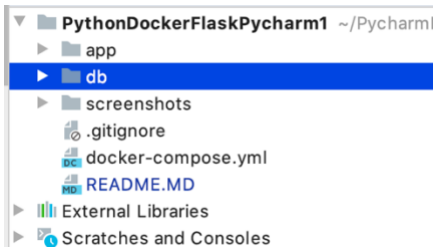


Figure 1 – Project Directory Structure and Files

- Create a ".gitignore" file
- Create a folder called "app"
- Create a folder called "db"
- Create a folder called "screenshots"
- Docker Compose File "docker-compose.yml"
- Create a "README.MD" file

Step 3 – Add the following text to your .gitignore

```
# Project exclude paths
/venv/
.idea
```

Figure 2 - .gitignore contents

This will tell GIT not to track your ".idea" folder and your "/venv/" folder.

Step 4 – Add the following code to your "docker-compose.yml" file

```yaml
version: "2"
services:
  app:
    build: ./app
    container_name: "Flask_App"
    links:
      - db
    ports:
      - "5000:5000"

  db:
    image: mysql:5.7
    container_name: "MySQL_Database"
    ports:
      - "32000:3306"
    environment:
      MYSQL_ROOT_PASSWORD: root
    volumes:
      - ./db:/docker-entrypoint-initdb.d/:ro
```

Figure 3 - docker-compose.yml contents

Step 5 – Add the following Text to your README.MD file

```
#Project Description
This project is a homework assignment to teach how to get Pycharm setup with Docker, Flask, MySQL
#Postman Screenshot
![postman request output](screenshots/postman.png)
```

Figure 4 - README.MD Contents

Step 7 – Create the following files within the “app” directory of your project:

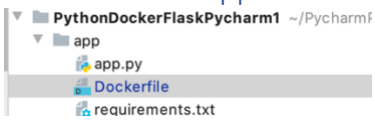


Figure 5 - app folder contents

- Create a “app.py” file
- Create a “Dockerfile” file
- Create a “requirements.txt” file

Step 8 - Add the following Text to your “app.py” file

```python
from typing import List, Dict
import mysql.connector
import simplejson as json
from flask import Flask, Response

app = Flask(__name__)


def cities_import() -> List[Dict]:
    config = {
        'user': 'root',
        'password': 'root',
        'host': 'db',
        'port': '3306',
        'database': 'citiesData'
    }
    connection = mysql.connector.connect(**config)
    cursor = connection.cursor(dictionary=True)

    cursor.execute('SELECT * FROM tblCitiesImport')
    result = cursor.fetchall()

    cursor.close()
    connection.close()

    return result


@app.route('/')
def index() -> str:
    js = json.dumps(cities_import())
    resp = Response(js, status=200, mimetype='application/json')
    return resp


if __name__ == '__main__':
    app.run(host='0.0.0.0')
```

Figure 6 - app.py contents

Step 9 - Add the following Text to your “Dockerfile” file

```
FROM python:3.8

EXPOSE 5000

WORKDIR /app

COPY requirements.txt /app
RUN pip install -r requirements.txt

COPY app.py /app
CMD python app.py
```

Figure 7 - Dockerfile Code

Step 10 - Add the following Text to your “requirements.txt file

```
Flask
mysql-connector
simplejson
```

Figure 8 - requirements.txt code

Step 11 – Create an “intit.sql” file within the “db” directory of your project:

Figure 9 - db folder contents

- Create a “init.sql” file

Step 12 – Copy the contents from my “init.sql” file and paste it in yours

Copy and paste the content from this link into your init.sql:

https://raw.githubusercontent.com/kaw393939/PythonDockerFlaskPycharm/master/db/init.sql

Step 13 – Run the project and view the results in Postman

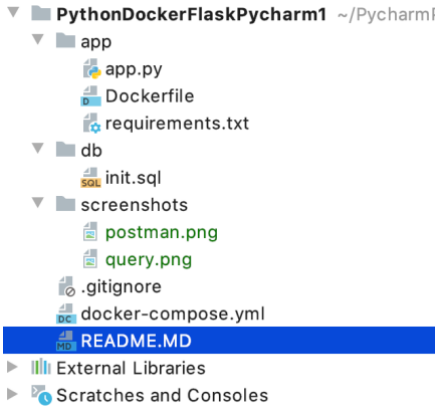


Figure 10 - Completed Project Structure

1. Download and Install PostMan: https://www.postman.com
2. Add a run configuration to use your Dockerfile
3. Run the project

4. Connect the Database Manager in PyCharm to connect to the project's MySQL Database

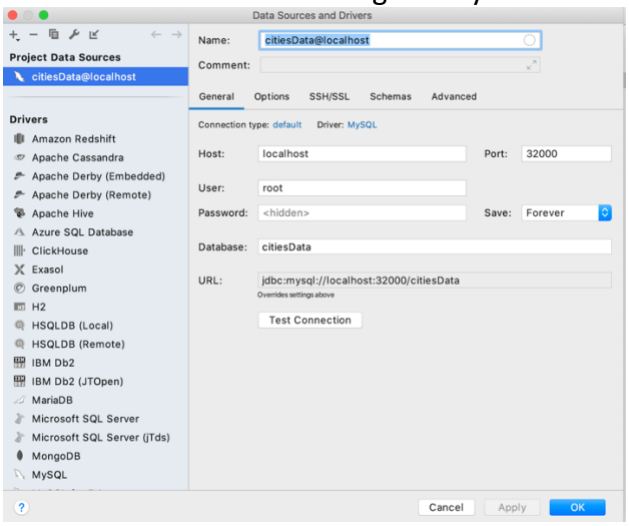


Figure 11 - Database Connection Settings

5. Connect the project interpreter by adding a new interpreter that uses Docker Compose and selecting that interpreter

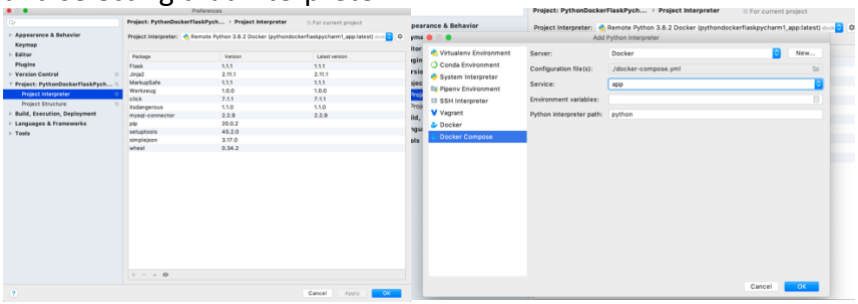
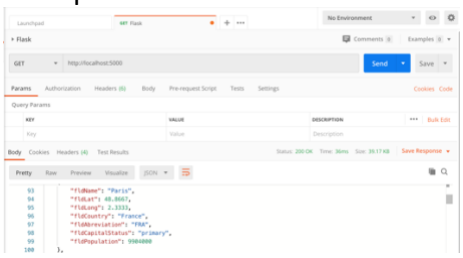


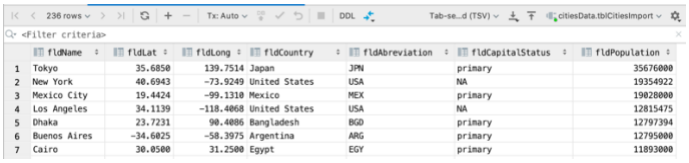
Figure 12 - Adding Remote Docker Compose Interpeter

6. Start postman and create a "GET" request for http://localhost:5000 and view the results



Submit your Project

1. Create a project on GitHub and a New Pycharm Project
2. Create commits for each completed step
3. Take a screenshot of the data in your MySQL database using PyCharm Database Manager and put it in your screen shots folder. Add the image to your README.MD file, so it appears on your project page when you submit to Github.



	fldName	fldLat	fldLong	fldCountry	fldAbreviation	fldCapitalStatus	fldPopulation
1	Tokyo	35.6850	139.7514	Japan	JPN	primary	35676000
2	New York	40.6943	-73.9249	United States	USA	NA	19354922
3	Mexico City	19.4424	-99.1310	Mexico	MEX	primary	19028000
4	Los Angeles	34.1139	-118.4068	United States	USA	NA	12815475
5	Dhaka	23.7231	90.4086	Bangladesh	BGD	primary	12797394
6	Buenos Aires	-34.6025	-58.3975	Argentina	ARG	primary	12795000
7	Cairo	30.0500	31.2500	Egypt	EGY	primary	11893000

Figure 13 - Query Data from PyCharm

4. Take a screenshot of your successful request in Postman and put it in your project's screenshot's folder. Add the image to your README.MD file, so it appears on your project page when you submit to GitHub.

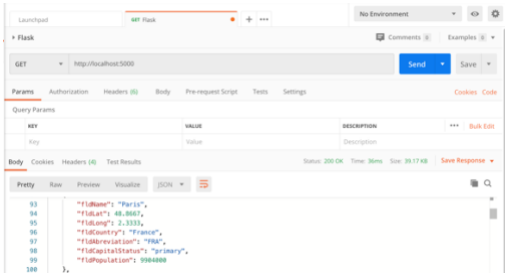


Figure 14 - Postman Request

5. Submit the project by sending a link to the repository in the assignment for your course.

Additional Reading

1. [What is the difference between a Docker file and docker-compose?]