Python & Flask Web-Application Example

Table of Contents

Python & Flask Installation Recommendations

If you're not familiar with Python web-development and want to try it out, we recommend:

1. Download & Install Oracle Virtual Box – software for installing virtual (guest) operating systems and running them from existing (host) operating system - https://www.virtualbox.org/wiki/Downloads

- User Manual - https://www.virtualbox.org/manual/UserManual.html

2. Download & Install Ubuntu 20.04 LTS as your guest operating system
   - https://ubuntu.com/download/desktop
3. Install Flask and PyODBC from your Ubuntu terminal (Python should be already installed)

   3.1. Install Python package installer – PIP
   - `> sudo apt install python3-pip`

   3.2. Install Flask
   - `> pip3 install --user flask`

   3.3. Install Python ODBC driver for Azure SQL
   - `> pip3 install pyodbc`

Useful Links

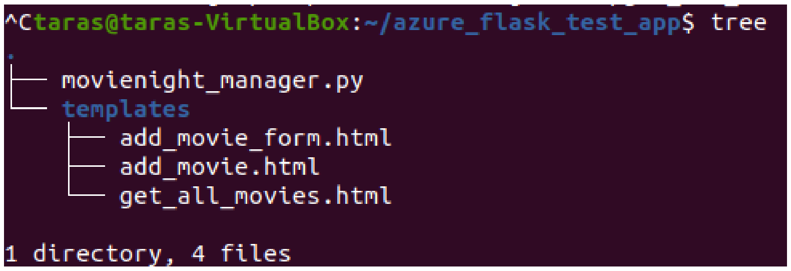
- Python Language Reference - https://docs.python.org/3/reference/index.html
- Flask Microframework Documentation - https://flask.palletsprojects.com/en/2.0.x/#
- Python & Azure SQL - https://docs.microsoft.com/en-us/python/api/overview/azure/sql?view=azure-python
  - *Just don't create SQL servers and databases from you python applications (and any other applications for that matter). They create databases configured by default which are very expensive if you lose track of them.*
- Some Web-application tutorials we found to be useful
  - *https://blog.ajbothe.com/a-guide-on-flask-and-azure-sql*

- https://www.codementor.io/@garethdwyer/building-a-crud-application-with-flask-and-sqlalchemy-dm3wv7yu2
- https://github.com/mkleehammer/pyodbc/wiki/Getting-started

Source Files (also available on Canvas)

Below source files compose a simple Python & Flask web-application which uses Azure SQL database to store and update the data about upcoming movie nights at someone's house.

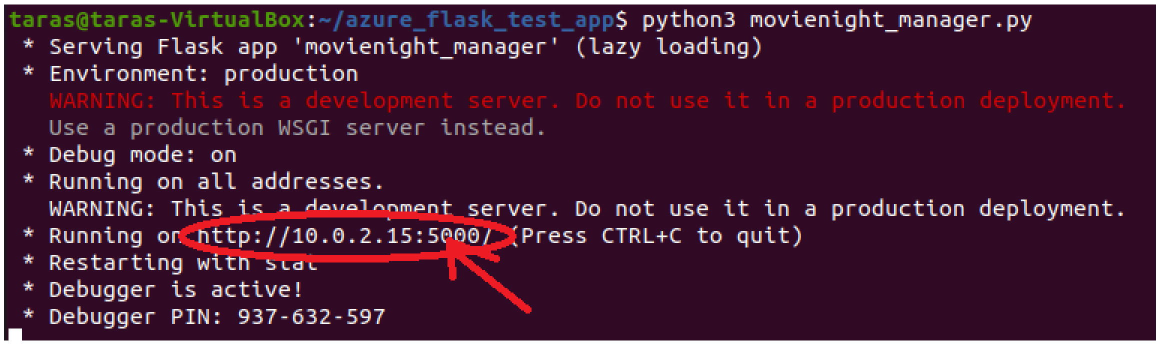
- SQL file is expected to be executed once in your SQL IDE of choice (Azure Data Studio, for example).
- The files are expected to be placed at some directory of your choosing (we created "azure_flask_test_app" directory) following the below directory structure



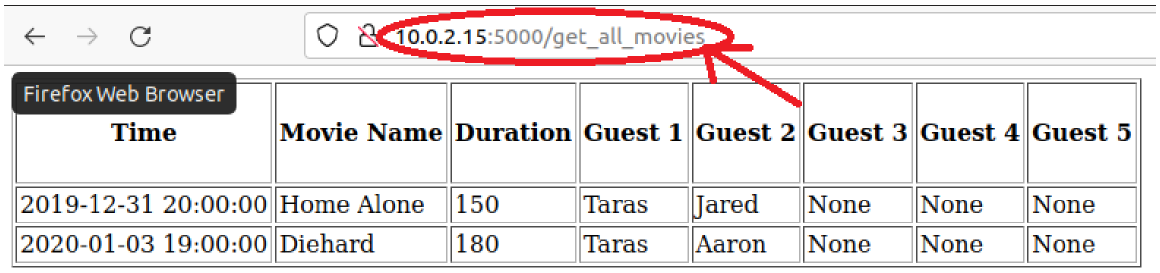
```
^Ctaras@taras-VirtualBox:~/azure_flask_test_app$ tree
.
├── movienight_manager.py
└── templates
    ├── add_movie_form.html
    ├── add_movie.html
    └── get_all_movies.html

1 directory, 4 files
```

- Once the files are copied over to the intended destination, start the web app by executing "python3 movienight_manager.py" command, see which endpoint your web-server is running on and use it for the testing.



```
taras@taras-VirtualBox:~/azure_flask_test_app$ python3 movienight_manager.py
 * Serving Flask app 'movienight_manager' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://10.0.2.15:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 937-632-597
```



Time	Movie Name	Duration	Guest 1	Guest 2	Guest 3	Guest 4	Guest 5
2019-12-31 20:00:00	Home Alone	150	Taras	Jared	None	None	None
2020-01-03 19:00:00	Diehard	180	Taras	Aaron	None	None	None

create_table.sql

Executing the queries in the below .sql file (with Azure Data Studio, for example) creates a very simple database for storing information about upcoming movie nights at someone's house.

```
DROP TABLE IF EXISTS movie_night; --Drop the table if it was previously created

--Create the new table for movie_nights schedule


CREATE TABLE movie_night (

    start_time DATETIME PRIMARY KEY,

    movie_name VARCHAR(64),

    duration_min INT,

    guest_1 VARCHAR(64),

    guest_2 VARCHAR(64),

    guest_3 VARCHAR(64),
    guest_4 VARCHAR(64),

    guest_5 VARCHAR(64),

);


--Insert two records to begin with

INSERT INTO movie_night

 (start_time, movie_name, duration_min, guest_1, guest_2)

VALUES

 ('2019-12-31 20:00:00', 'Home Alone', 150, 'Taras', 'Jared'),

 ('2020-01-03 19:00:00', 'Diehard', 180, 'Taras', 'Aaron');
```

movienight_manager.py

Below Python file contains code used to connect to the Azure SQL Database and configure the Flask web-application to respond to a number of URL requests.

Make sure to substitute your own values into "server", "database", "username" and "password".

```
from flask import Flask, render_template, request


# Import the ODBC driver used to connect to the database

import pyodbc


"""

    This method returns a connection cursor to the Azure SQL database.

"""

def get_db_cursor():
    # Connection credentials
    server = '<Replace Me!>-sql-server.database.windows.net'
    database = 'cs-dsa-4513-sql-db'
    username = '<Replace Me!>'
    password = '<Replace Me!>'
    driver= '{ODBC Driver 17 for SQL Server}'

    # Connect to the database using the above credentials and return the cursor
    cnxn =
pyodbc.connect('DRIVER='+driver+';PORT=1433;SERVER='+server+';PORT=1443;DATABASE='+database+';UID='+use
rname+';PWD='+ password)
    return cnxn.cursor()

# Instantiate the Flas
app = Flask(__name__)

"""
    Creates a response when the for the /get_all_movies URL
"""
@app.route("/get_all_movies", methods=["GET"])
def get_all_movies():
    # Get a database cursor and submit a query to fetch all movie nights
    cursor = get_db_cursor()
    cursor.execute("SELECT * FROM movie_night")
    rows = cursor.fetchall()

    # Repackage the returned rows into a python list of associative arrays to be used for the template rendering.
    movies = []
    for row in rows:
        movies.append({"start_time": str(row[0]), "movie_name": row[1], "duration_min": row[2], "guest_1": row[3],
"guest_2": row[4], "guest_3": row[5], "guest_4": row[6], "guest_5": row[7]})

    # Return the template rendered with the list of fetched movie nights
    return render_template("get_all_movies.html", movies=movies)

"""
    When the /add_movie_form URL is requested just render the form template as it takes no database data or user
input.
"""
@app.route("/add_movie_form")
def add_movie_form():
    return render_template("add_movie_form.html")

"""
    Requesting the /add_movie URL is only expected by submitting the form from /add_movie_form page
    We will collect the user data from the request object, submit an INSERT query using it,
    and render the response page confirming the success.
"""
@app.route("/add_movie", methods=["POST"])
def add_movie():
    # If the request was submitted by the form
    if request.form:
        # Extract the user data from the request object
        startTime = request.form["start_time"];
        movieName = request.form["movie_name"];
        durationString = request.form["duration_min"];
        g1 = request.form["guest_1"];

        g2 = request.form["guest_2"];

        g3 = request.form["guest_3"];

        g4 = request.form["guest_4"];

        g5 = request.form["guest_5"];


        # If no start time or movie name was given by the user, redirect back to the form page

        if not startTime or not movieName:

            return redirect("/add_movie_form")


        # Get the database cursor, populate the INSERT query with the user data and submit it

        cursor = get_db_cursor()

        cursor.execute("INSERT INTO movie_night (start_time, movie_name, duration_min, guest_1, guest_2, guest_3,
guest_4, guest_5) VALUES (?, ?, ?, ?, ?, ?, ?, ?)",

            startTime, movieName, durationString, g1, g2, g3, g4, g5)

        cursor.commit()


        # Render the success confirmation page

        return render_template("add_movie.html", movie = {

            "startTime" : startTime, "movieName" : movieName, "durationString" : durationString, "g1" : g1, "g2" : g2, "g3"
: g3, "g4" : g4, "g5" : g5})

    else:

        # Getting here by not submitting the form is unexpected, so just redirect to the form page

        return redirect("/add_movie_form")


# When this file is executed (and not imported by another file) launch the Web-App

if __name__ == "__main__":

    app.run(host='0.0.0.0', debug=True)
```

get_all_movies.html

Rendering below HTML file generates a table containing all the records from the movie_night SQL database table.

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Query Result</title>
  </head>
  <body>
    <!-- The table for displaying all the movie records -->
    <table cellspacing="2" cellpadding="2" border="1">
      <tr> <!-- The table headers row -->
        <td align="center">
          <h4>Time</h4>
        </td>
        <td align="center">
          <h4>Movie Name</h4>
        </td>
        <td align="center">
          <h4>Duration</h4>
        </td>
        <td align="center">
          <h4>Guest 1</h4>
        </td>
        <td align="center">
          <h4>Guest 2</h4>
        </td>
        <td align="center">
          <h4>Guest 3</h4>
        </td>
        <td align="center">
          <h4>Guest 4</h4>
        </td>
        <td align="center">
          <h4>Guest 5</h4>
        </td>
      </tr>


      <!--
          For each movie given, this will render a table row with every movie attribute in a separate cell.
      -->
      {% for m in movies %}
          <tr>
              <td>{{m.start_time}}</td>
              <td>{{m.movie_name}}</td>
              <td>{{m.duration_min}}</td>
              <td>{{m.guest_1}}</td>
              <td>{{m.guest_2}}</td>
              <td>{{m.guest_3}}</td>
              <td>{{m.guest_4}}</td>
              <td>{{m.guest_5}}</td>
          </tr>
      {% else %}
          <tr><td colspan="8">No movie nights scheduled yet :(</td></tr>
                  {% endfor %}


              </table>

          </body>

      </html>
```

add_movie_form.html

Below HTML file generates a form for collection of user input to insert a new record into a movie_night table. Upon form submission, user input will be processed, the query submitted by movienight_manager.py file, and the add_movie.html (see below) file will be rendered to confirm the user input.

```
<!DOCTYPE html>

<html>

    <head>

        <meta charset="UTF-8">

        <title>Add Movie Night</title>

    </head>

    <body>

        <h2>Add Movie Night</h2>

        <!--

            Form for collecting user input for the new movie_night record.

            Upon form submission, /add_movie URL will be invoked.

        -->

        <form action="/add_movie" method="POST">

            <!-- The form organized in an HTML table for better clarity. -->

            <table border=1>

                <tr>
    <th colspan="2">Enter the Movie Night Data:</th>

  </tr>

  <tr>

    <td>Movie night time:</td>

    <td><div style="text-align: center;">

    <input type=text name=start_time>

    </div></td>

  </tr>

  <tr>

    <td>Movie Name:</td>

    <td><div style="text-align: center;">

    <input type=text name=movie_name>

    </div></td>

  </tr>

  <tr>

    <td>Duration:</td>

    <td><div style="text-align: center;">

    <input type=text name=duration_min>

    </div></td>

  </tr>

  <tr>

    <td>Guest 1 Name:</td>

    <td><div style="text-align: center;">

    <input type=text name=guest_1>

    </div></td>

  </tr>

  <tr>

    <td>Guest 2 Name</td>
    <td><div style="text-align: center;">
    <input type=text name=guest_2>
    </div></td>
  </tr>
  <tr>
    <td>Guest 3 Name</td>
    <td><div style="text-align: center;">
    <input type=text name=guest_3>
    </div></td>
  </tr>
  <tr>
    <td>Guest 4 Name</td>
    <td><div style="text-align: center;">
    <input type=text name=guest_4>
    </div></td>
  </tr>
  <tr>
    <td>Guest 5 Name</td>
    <td><div style="text-align: center;">
    <input type=text name=guest_5>
    </div></td>
  </tr>
  <tr>
    <td><div style="text-align: center;">
    <input type=reset value=Clear>
    </div></td>
    <td><div style="text-align: center;">
    <input type=submit value=Insert>
```

```
                </div></td>
            </tr>
        </table>
    </form>
  </body>
</html>
```

add_movie.html

Below HTML when rendered confirms the insertion of the new record in the database.

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Query Result</title>
    </head>
    <body>
        <!--
            This simply displays the user input back to the user as a list.
        -->
        <h2>The Movie Night:</h2>
        <ul>
            <li>Start Time: {{movie.startTime}}</li>
            <li>Movie Name: {{movie.movieName}}</li>
            <li>Duration: {{movie.durationString}}</li>
            <li>Guest 1: {{movie.g1}}</li>
            <li>Guest 2: {{movie.g2}}</li>
            <li>Guest 3: {{movie.g3}}</li>
            <li>Guest 4: {{movie.g4}}</li>

            <li>Guest 5: {{movie.g5}}</li>

        </ul>

        <h2>Was successfully inserted.</h2>

        <a href="/get_all_movies">See all movie nights.</a>

    </body>

</html>
```