UWI

INFO2602 – Lab 7
SQLAlchemy, Flask and Templates

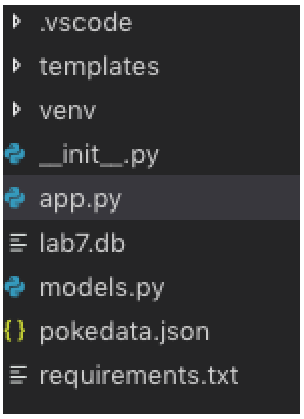
Section 1 – SQLAlchemy

Introduction

This lab is a continuation of developing backend REST-based APIs using Flask. We will also explore create some UI using Templates.

Setup

1. Download the lab7 starter zip file from the course website.
2. Extract the file into your development folder. Ensure that you have all the required files from the zip archive.



```
.vscode
templates
venv
__init__.py
app.py
lab7.db
models.py
pokedata.json
requirements.txt
```

3. Create the virtual environment

python3 -m venv venv
source venv/bin/activate

4. Install the dependencies using the supplied requirements.txt file.

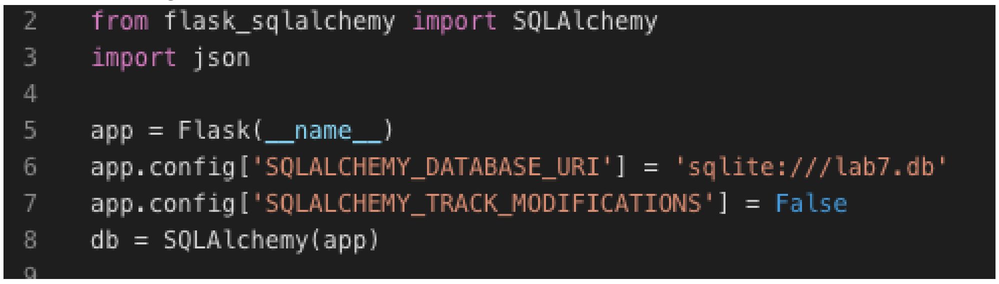
pip install -r requirements.txt

5. This allows us to install a consistent version of the software packages between different environments. This installs all the packages in one command. Inspect the requirements.txt file to see the packages installed.
6. Perform a check to ensure that all packages installed correctly.

pip freeze

Review

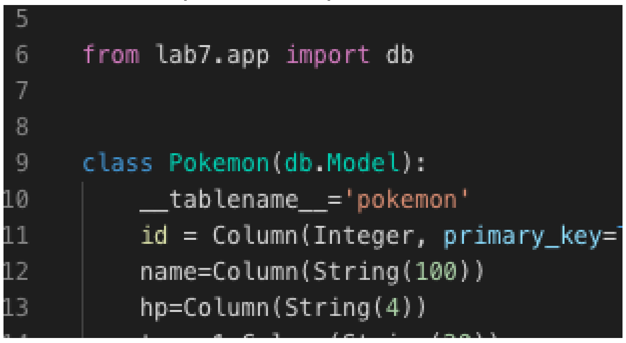
1. In this lab, we are using the flask_sqlalchemy package which provides a better way of managing the connection between our application and the database as the number of sessions (individual requests) increases.
2. This requires a slight modification to our previous example. Most notably:
    a. The SQLAlchemy class from the flask_sqlalchemy package is used to manage the connection to the database



```python
from flask_sqlalchemy import SQLAlchemy
import json

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///lab7.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)
```

    b. The models inherit from the Model class in the SQLAlchemy (db.Model) rather than the Base (from Lab 6)

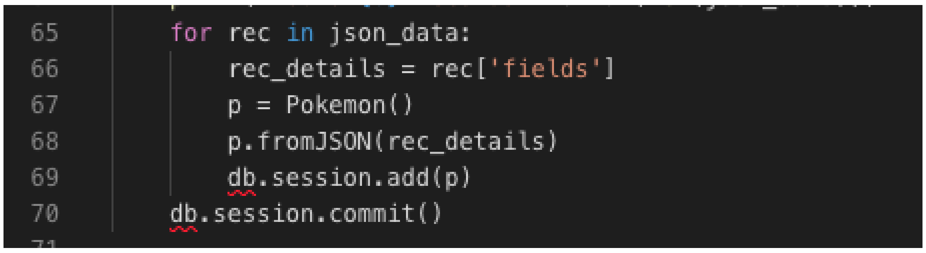


```python

from lab7.app import db


class Pokemon(db.Model):
    __tablename__='pokemon'
    id = Column(Integer, primary_key=
    name=Column(String(100))
    hp=Column(String(4))
```

    c. The db connection from SQLAlchemy is also used to save models to the database



```python
    for rec in json_data:
        rec_details = rec['fields']
        p = Pokemon()
        p.fromJSON(rec_details)
        db.session.add(p)
    db.session.commit()
```

3. Recall that we define the model which is also used to generate the table in the SQL database. We have considered Column types so far (Integers, Strings, Datetime)
4. We create the toDict method to make it easier to convert the model to JSON for transmission (serialisation)
5. We create a method 'fromJSON' to move from the text-based JSON representation to populate the fields of the class. (deserialisation)

Section 2 - Flask

Introduction

This lab aims to provide students with a guided experience of using existing patterns and the software to create new functional requirements. The functions 'get_all_pokemon_type' and 'show_all_pokemon' are implemented to as a guide for the following questions

Questions

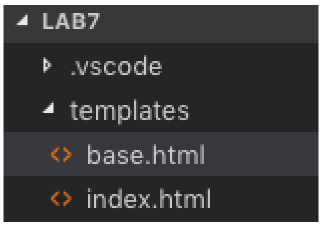
1. Complete the functionality for the ‘get_pokemon_by_type’ function. The function will use the type specified in the URL to display only records that fit the criteria given. (hint you are required to filter the data by querying information on the column using the SQLAlchemy Query API http://flask-sqlalchemy.pocoo.org/2.3/queries/) [Note that the current implementation is case sensitive.]
   a. Test your API with the following queries:
      i. http://localhost:5000/api/pokemon/type1/Ground
      ii. http://localhost:5000/api/pokemon/type1/Poison
      iii. http://localhost:5000/api/pokemon/type1/Fighting
2. Define the route to retrieve pokemon records by their primary key “/api/pokemon/<pk_id>”, associate the route with the “get_pokemon_by_id” function and complete the functionality to retrieve a record via its primary key. (getting record by id http://flask-sqlalchemy.pocoo.org/2.3/queries/)

Section 3 - Template

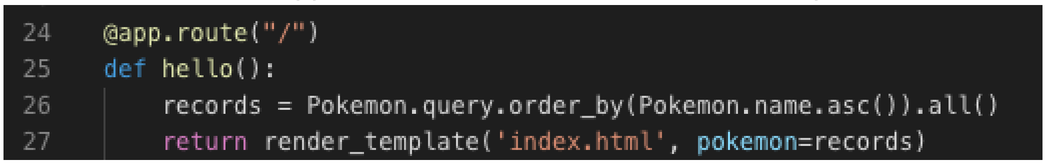
Introduction

Templates allow us to create presentation elements (HTML) for our applications. It emphasizes the philosophy of DRY to make development of dynamic front-end interfaces easier. It also provides additional functionality like the ability to have conditional information displayed based on information passed (for example, display information only if user is logged in)

We have pre-built out some components of a template for you to extend. The template files are found in the ‘templates’ folder.



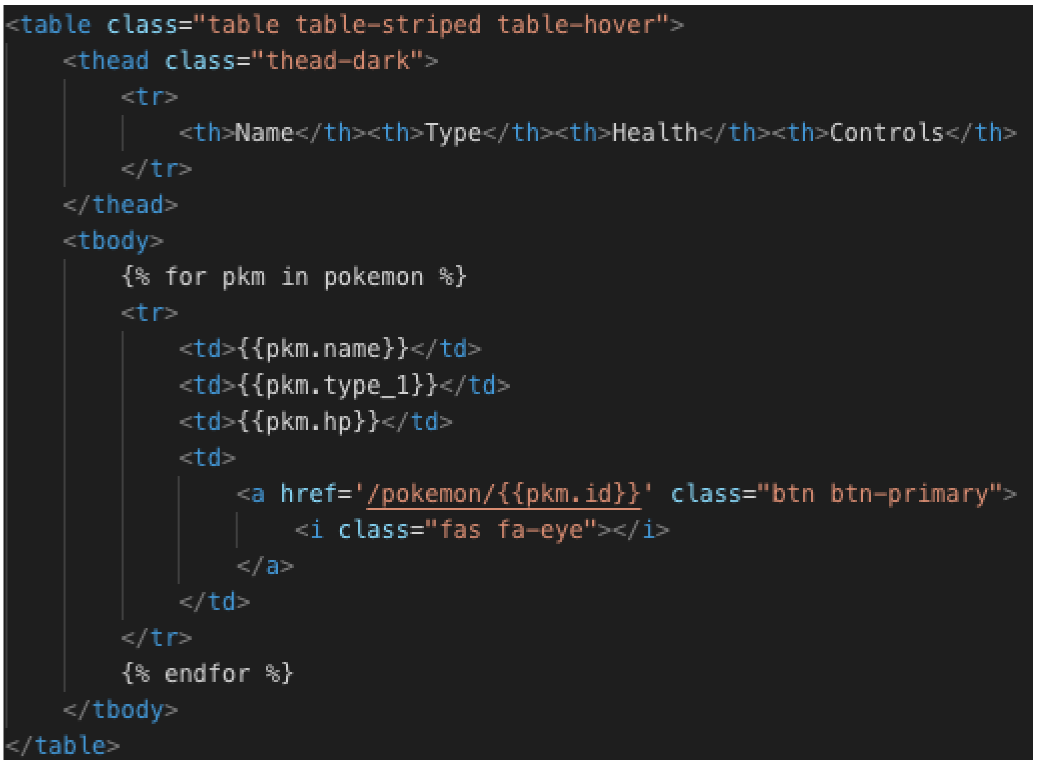
The templates are rendered when a particular route is executed. In this case when the client navigates to the root of our application, we render the index.html template file.



```
@app.route("/")
def hello():
    records = Pokemon.query.order_by(Pokemon.name.asc()).all()
    return render_template('index.html', pokemon=records)
```

Note that we are passing the records from our query into the template using the render_template function.

The records are then displayed in the template using a for-loop like structure



```html
<table class="table table-striped table-hover">
    <thead class="thead-dark">
        <tr>
            <th>Name</th><th>Type</th><th>Health</th><th>Controls</th>
        </tr>
    </thead>
    <tbody>
        {% for pkm in pokemon %}
        <tr>
            <td>{{pkm.name}}</td>
            <td>{{pkm.type_1}}</td>
            <td>{{pkm.hp}}</td>
            <td>
                <a href='/pokemon/{{pkm.id}}' class="btn btn-primary">
                    <i class="fas fa-eye"></i>
                </a>
            </td>
        </tr>
        {% endfor %}
    </tbody>
</table>
```

Questions:

1. Create a new route 'pokemon/<id>' that will display the details of the pokemon selected by the id specified. The system should load a details.html file that displays information in table-based format. You should think through what is required and also how to handle invalid values.

Section 4 - Heroku (H.W)

1. Sign up for a free heroku account: https://signup.heroku.com
2. Install Heroku for Ubuntu using snap: https://devcenter.heroku.com/articles/heroku-cli
   sudo snap install --classic heroku
3. Login to Heroku from command line: https://devcenter.heroku.com/articles/heroku-cli#getting-started
   heroku login
4. Ensure that you are in the lab7 folder, and create a new heroku project
   heroku create
5. Add the procfile for python projects: https://devcenter.heroku.com/articles/getting-started-with-python#deploy-the-app
6. Git add and commit all files
7. Push to heroku
8. Test if publically available from URL provided.