Solution: Retrieving Pets from the Home and Details Pages

In this lesson, we will take a look at the solution to the challenge presented in the previous lesson.

WE'LL COVER THE FOLLOWING

Solution

```
"""Flask Application for Paws Rescue Center."""
from flask import Flask, render_template, abort
from forms import SignUpForm, LoginForm
from flask import session, redirect, url_for
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SECRET_KEY'] = 'dfewfew123213rwdsgert34tgfd1234trgf'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///paws.db'
db = SQLAlchemy(app)

"""Model for Pets."""
class Pet(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String, unique=True)
    age = db.Column(db.String)
    bio = db.Column(db.String)
    posted_by =  db.Column(db.String, db.ForeignKey('user.id'))


"""Model for Users."""
class User(db.Model):
```

```
    id = db.Column(db.Integer, primary_key=True)
    full_name = db.Column(db.String)
    email = db.Column(db.String, unique=True)
    password = db.Column(db.String)
    pets = db.relationship('Pet', backref = 'user')

db.create_all()

# Create "team" user and add it to session
team = User(full_name = "Pet Rescue Team", email = "team@petrescue.co", password = "adminpass
db.session.add(team)

# Create all pets
nelly = Pet(name = "Nelly", age = "5 weeks", bio = "I am a tiny kitten rescued by the good pe
yuki = Pet(name = "Yuki", age = "8 months", bio = "I am a handsome gentle-cat. I like to dres
basker = Pet(name = "Basker", age = "1 year", bio = "I love barking. But, I love my friends m
mrfurrkins = Pet(name = "Mr. Furrkins", age = "5 years", bio = "Probably napping.")

# Add all pets to the session
db.session.add(nelly)
db.session.add(yuki)
db.session.add(basker)
db.session.add(mrfurrkins)

# Commit changes in the session
try:
    db.session.commit()
except Exception as e:
    db.session.rollback()
finally:
    db.session.close()

@app.route("/")
def homepage():
    """View function for Home Page."""
    pets = Pet.query.all()
    return render_template("home.html", pets = pets)


@app.route("/about")
def about():
    """View function for About Page."""
    return render_template("about.html")


@app.route("/details/<int:pet_id>")
def pet_details(pet_id):
    """View function for Showing Details of Each Pet."""
    # pet = next((pet for pet in pets if pet["id"] == pet_id), None)
    pet = Pet.query.get(pet_id)
    if pet is None:
        abort(404, description="No Pet was Found with the given ID")
    return render_template("details.html", pet = pet)


@app.route("/signup", methods=["POST", "GET"])
def signup():
    """View function for Showing Details of Each Pet."""
    form = SignUpForm()
    if form.validate_on_submit():
        new_user = User(full_name = form.full_name.data, email = form.email.data, password =
        db.session.add(new_user)
```

```
        try:
            db.session.commit()
        except Exception as e:
            print(e)
            db.session.rollback()
            return render_template("signup.html", form = form, message = "This Email already
        finally:
            db.session.close()
        return render_template("signup.html", message = "Successfully signed up")
    return render_template("signup.html", form = form)


@app.route("/login", methods=["POST", "GET"])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email = form.email.data, password = form.password.data).f
        if user is None:
            return render_template("login.html", form = form, message = "Wrong Credentials. P
        else:
            session['user'] = user.id
            return render_template("login.html", message = "Successfully Logged In!")
    return render_template("login.html", form = form)

@app.route("/logout")
def logout():
    if 'user' in session:
        session.pop('user')
    return redirect(url_for('homepage', _scheme='https', _external=True))

if __name__ == "__main__":
    app.run(debug=True, host="0.0.0.0", port=3000)
```

Explanation

Let's break down the solution of this challenge to figure out how we solved it.

Home Page

Modifications in the `homepage` view

In the `homepage`, we were previously sending the whole `pets` list as a variable to the template. So, instead of the list, we will query the `Pet` model and retrieve all objects from the database. In **line 58**, you can see that we have used the `Pet.query.all()` method to retrieve a list of `Pet` objects. Then, this list is passed to the `home.html` template.

Modifications in the `home.html` template

As the `pets` variable that we passed is a list of `Pet` objects and not a list of `Dictionary` objects, we will have to change the syntax to access it. In **lines 20 and 21**, you can see that we replaced `pet['id']` with `pet.id`. Similarly, in **lines 24 to 26**, we replaced `pet['name']`, `pet['age']` and `pet['bio']` with

`pet.name`, `pet.age` and `pet.bio` respectively.

Pet Details page

Modifications in the `pet_details` view

In the `pet_details`, we receive a `pet_id` variable from the dynamic `URL`. Previously, we were searching for the **pet** associated with the `pet_id` in the `pets` list and sending it as a variable to the template.

However, we will now query for the corresponding object from the `Pet` model and retrieve it. In **line 71**, you can see that we have used the `Pet.query.get()` method to retrieve a single instance of the `Pet` class. You might recall that the `query.get()` function takes the value of the `primary_key` column as a parameter. Therefore, we passed it the `pet_id` variable. We had already handled the case for `None` type objects so no other changes were needed in this function.

Modifications in the `details.html` template

Similar to the changes in `home.html`, we just modified the syntax of accessing values from the `pet` variable.

> 📢 **We have finally gotten rid of the `pets` list that we were using prior. Hurrah!**

In the next challenge, we will deal with the update and deletion operations of the `Pet` model.