

Project Challenge: Inserting Pets

In this challenge, we will insert some pets in the system by default and retrieve them to show on the home page.

WE'LL COVER THE FOLLOWING ^

- Problem statement
- Your implementation

Problem statement

In the previous challenges, we got rid of the `users` list. Now, we also want to replace the `pets` list. Therefore, in this challenge, you will perform the following tasks:

1. Create objects of the `Pet` model **corresponding** to the dictionaries that are present in the `pets` list.
2. Add these objects to the `db.session` and **commit** them to the database.
3. Handle any **exceptions** while committing.
4. You can implement this below the insertion of the `team` object of the `User` model.

🔗 **Note:** you do not have to add the value of the `posted_by` column of the `pets`.

Your implementation

Implement the features described above in the application provided below.

⚠ **Disclaimer:** please do not remove the `pets` list from the application

yet, as it may break the existing features of the application.

```
"""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)

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

"""Information regarding the Pets in the System."""
pets = [
    {"id": 1, "name": "Nelly", "age": "5 weeks", "bio": "I am a tiny kitten rescued by a pet rescue team."},
    {"id": 2, "name": "Yuki", "age": "8 months", "bio": "I am a handsome gentle-cat."},
    {"id": 3, "name": "Basker", "age": "1 year", "bio": "I love barking. But, I love napping."},
    {"id": 4, "name": "Mr. Furrkins", "age": "5 years", "bio": "Probably napping."},
]

@app.route("/")
def homepage():
    """View function for Home Page."""
    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)
    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)
        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)

```

In the next lesson, we'll look at the solution to this challenge.

