# This file defines command line commands for manage.py

#

# Copyright 2014 SolidBuilds.com. All rights reserved

#

# Authors: Ling Thio ling.thio@gmail.com

import datetime

from flask import current_app
from flask_script import Command

from app import db
from app.models.employee_models import Employee, Role
from app.models.menuItem_models import Menu, MenuItem, MenuItems
from app.models.order_models import Order, OrderItems, OrderAssignments
from app.models.category_models import Category,MenuCategories,ItemCategory
from app.models.restaurant_models import
Restaurant,RestaurantMenus,RestaurantEmployees

from sqlalchemy import and_, or_

class InitDbCommand(Command):
""" Initialize the database."""

```
def run(self):
    init_db()
    print('Database has been initialized.')
```

def init_db():
""" Initialize the database."""
db.drop_all()
db.create_all()
create_employees()

def create_employees():
""" Create users """

```
# Create all tables
db.create_all()

# Adding roles
# Roles include
# Owner
# Director
# Waiters
owner_role      = find_or_create_role('owner', u'Owner')
director_role   = find_or_create_role('director', u'Director')
waiter_role     = find_or_create_role('waiter', u'Waiter')

# Add users
owner = find_or_create_user(u'Owner', u'Example', u'owner@example.com', 'pass', owne
director = find_or_create_user(u'Director', u'Example', u'director@example.com', 'pa
waiter = find_or_create_user(u'Waiter', u'Example', u'waiter@example.com', 'pass',wa
waiter1 = find_or_create_user(u'Waiter1', u'Example', u'waiter1@example.com', 'pass

r1 = find_or_create_restaurants("Flavortownne","picture/is/here.png","Welcome to fla
r2 = find_or_create_restaurants("Flavortownne II","picture/is/hereII.png","Welcome t

m = find_or_create_menu("General")
m1 = find_or_create_menu("Gen1")
db.session.commit()

# Note: Assuming the db.create_all() command has already been called.
find_or_create_menu_item("Risoto De Milano","27",active=True,category="Dinner",ingre
find_or_create_menu_item("Shrimp Skampi","22",active=True,category="Dinner",ingredie
find_or_create_menu_item("Spaghetti","22",active=True,category="Lunch",ingredients="
find_or_create_menu_item("Breadsticks","22",active=True,category="Appetizer",ingredi
find_or_create_menu_item("Eggplant Parmesean","25",active=False,category="Dinner",in
find_or_create_menu_item("Sushi","25",active=False,category="Appetizer",allergy_info

db.session.commit()
find_or_create_restaurant_employees(r1.id,owner.id)
find_or_create_restaurant_employees(r2.id,owner.id)
find_or_create_restaurant_employees(r1.id,waiter.id)
find_or_create_restaurant_employees(r2.id,waiter.id)
find_or_create_restaurant_employees(r1.id,director.id)
find_or_create_restaurant_employees(r2.id,director.id)
find_or_create_restaurant_employees(r1.id,waiter1.id)

find_or_create_restaurant_menu(restaurant_id=r1.id,menu_id=m.id)
```

```
find_or_create_restaurant_menu(restaurant_id=r2.id,menu_id=m1.id)

find_or_create_categories(categories=["Breakfast","Lunch","Dinner","Dessert"])

# c=Category.query.filter(Category.name=="Breakfast").first()
#
# c=Category.query.filter(Category.name=="Lunch").first()
#
# c=Category.query.filter(Category.name=="Dinner").first()
#
# c=Category.query.filter(Category.name=="Dessert").first()


# Save to DB
db.session.commit()
```

def find_or_create_menu(name):
menu = Menu.query.filter(Menu.name==name).first()
if not menu:
menu = Menu(name=name)
db.session.add(menu)
return menu

def find_or_create_restaurant_menu(restaurant_id,menu_id):
rm = RestaurantMenus.query.filter(RestaurantMenus.menu_id==menu_id).first()
if not rm:
rm = RestaurantMenus(restaurant_id=restaurant_id,menu_id=menu_id)
db.session.add(rm)
db.session.commit()
return rm

def
find_or_create_menu_item(name,price,menu_id=1,active=None,category=None,information=None,ingredients=None,allergy_information=None):
menu = Menu.query.get(menu_id)
menuItem = MenuItem.query.filter(MenuItem.name==name).first()
if not menuItem:
menuItem =
MenuItem(name=name,price=price,active=active,category=category,information=information,ingredients=ingredients,allergy_information=allergy_information)
db.session.add(menuItem)
db.session.commit()

```
    menuItems = MenuItems(menu_id=menu.id,item_id=menuItem.id)
```

```
      db.session.add(menuItems)
      db.session.commit()
  return menuItem
```

def find_or_create_role(name, label):
""" Find existing role or create new role """
role = Role.query.filter(Role.name == name).first()
if not role:
role = Role(name=name, label=label)
db.session.add(role)
return role

def find_or_create_user(first_name, last_name, email, password, role=None):
""" Find existing user or create new user """
employee = Employee.query.filter(Employee.email == email).first()
if not employee:
employee = Employee(email=email,
first_name=first_name,
last_name=last_name,
password=current_app.user_manager.password_manager.hash_password(password),
active=True,
email_confirmed_at=datetime.datetime.utcnow())
if role:
employee.roles.append(role)
db.session.add(employee)
return employee

def find_or_create_restaurants(name,picture_url,tagline):
r = Restaurant.query.filter(Restaurant.name==name).first()
if(not r):
r = Restaurant(name=name,picture_url=picture_url,tagline=tagline)
db.session.add(r)
db.session.commit()
return r

def find_or_create_categories(category_name=None,categories=[]):
if(category_name):
c = Category.query.filter(Category.name==category_name).first()
if(not c):
c = Category(name=category_name)
elif(categories):
for c in categories:

c = Category.query.filter(Category.name==category_name).first()
if(not c):
c = Category(name=category_name)

def add_menu_category(menu_name,category_name):
m = Menu.query.filter(Menu.name==menu_name).first()
c = Category.query.filter(Category.name==category_name).first()
if(not (m and c)):
return
mc = MenuCategories.query.filter(and_(*
[MenuCategories.category_id==c.id,MenuCategories.menu_id==m.id])).first()

```
if(not mc):
    mc = MenuCategories(menu_id=m.id,category_id=c.id)
    db.session.add(mc)
    db.session.commit()
```

def add_item_category(item_name,category_name):
m = Menu.query.filter(Menu.name==menu_name).first()
c = Category.query.filter(Category.name==category_name).first()
if(not (m and c)):
return
mc = MenuCategories.query.filter(and_(*
[MenuCategories.category_id==c.id,MenuCategories.menu_id==m.id])).first()

```
if(not mc):
    mc = MenuCategories(menu_id=m.id,category_id=c.id)
    db.session.add(mc)
    db.session.commit()
```

def find_or_create_restaurant_employees(restaurant_id,employee_id):
ri = RestaurantEmployees.query.filter(and_( *
[RestaurantEmployees.employee_id==employee_id,RestaurantEmployees.restaurant_id==rest
aurant_id] )).first()
if(not ri):
ri = RestaurantEmployees(restaurant_id=restaurant_id,employee_id=employee_id)
db.session.add(ri)
db.session.commit()
return ri