Анализ проекта seed_database.py от 2024-12-01 02:46:49.267642 UTC+3

Дата последнего изменения проекта : 2024-12-01 02:46:49.267642 UTC+3
Общее количество ошибок : 1 Ошибки логирования : 0; Ошибки стандарта PEP : 1

Ошибки логирования

print_statements

logging_imports

basic_config_usage

non_lazy_logging

json_logger_usage

global_loggers

scoped_loggers

Ошибки стандарта PEP

./path/to/seed_database.py//seed_database.py: Строка 1: Превышена максимальная длина (>79 символов)

```
| 1 -> b'import datetime\nimport random\nimport sys\n\nimport faker\nfrom sqlalchemy.orm import load_only\nfrom sqlalchemy.sql import ClauseElement\nfrom sqlalchemy.sql.expression import func\n\nfrom addresses.models import Address\n\nfrom categories.models import Category\nfrom comments.models import Comment\nfrom ecommerce_api.factory import db, bcrypt\nfrom file_uploads.models import ProductImage, TagImage, CategoryImage\nfrom orders.models import Order, OrderItem\nfrom products.models import Product\nfrom roles.models import Role, UserRole\nfrom tags.models import Tag\nfrom users.models import User\n\nfake = faker.Faker()\ntags = []\ncategories = []\n\n\ndef get_or_create(session, model, defaults=None, **kwargs):\n instance = session.query(model).filter_by(**kwargs).first()\n if instance:\n return
```

```
instance, False\n else:\n params = dict((k, v) for k, v in kwargs.iteritems() if
not isinstance(v, ClauseElement))\n params.update(defaults or {})\n instance =
model(**params)\n session.add(instance)\n return instance, True\n\n\ndef
encrypt_password(password):\n return
bcrypt.generate_password_hash(password).decode(\'utf-8\')\n\n\ndef
generate_image(model):\n # pattern = "".join([random.choice([\'?\', \'#\']) for i
in range(0, 10)]) + \'.png\'\n filename_pattern =
"".join(fake.random_choices(elements=(\'?\', \'#\'),\n
length=fake.random_int(min=16, max=32))) + \'.png\'\n #
file_name=fake.md5(raw_output=False) + \'.png\'\n return
model(file_name="".join(fake.random_letters(length=16)) + \'.png\',\n
file_path=fake.image_url(width=None, height=None),\n
file_size=fake.random_int(min=1000, max=15000),\n
original_name=fake.bothify(text=filename_pattern))\n\n\ndef seed_admin():\n role,
created = get_or_create(db.session, Role,\n defaults={\'description\': \'for
admin only\'},\n name=\'ROLE_ADMIN\')\n \'\'\'\n # count admin\n admin_count =
User.query.filter(User.roles.any(id=role.id)).count()\n\n # 4 ways of retrieving
the admin users\n admin_users =
User.query.filter(User.users_roles.any(role_id=role.id)).all()\n admin_users =
User.query.filter(User.roles.any(id=role.id)).all()\n admin_users =
User.query.filter(User.roles.any(name=\'ROLE_ADMIN\')).all()\n admin_users =
User.query.join(User.roles).filter_by(name=role.name).all()\n \'\'\'\n\n user,
created = get_or_create(db.session, User, defaults={\'first_name\':
\'adminFN\',\n \'last_name\': \'adminFN\',\n \'email\':
\'admin@flaskblogapi.app\',\n \'password\':
bcrypt.generate_password_hash(\'password\')},\n **{\'username\': \'admin\'})\n\n
db.session.commit()\n\n if len(user.roles) == 0:\n #
user.users_roles.append(UserRole(user_id=user.id, role_id=role.id))\n
user.roles.append(role)\n db.session.commit()\n\n\ndef seed_authors():\n role,
created = get_or_create(db.session, Role, defaults={\'description\': \'for
authors only\'},\n name=\'ROLE_AUTHOR\')\n\n authors_count =
db.session.query(User.id).filter(User.roles.any(id=role.id)).count()\n
authors_count = User.query.filter(User.roles.any(id=role.id)).count()\n
authors_to_seed = 5\n authors_to_seed -= authors_count\n\n for i in range(0,
authors_to_seed):\n profile = fake.profile(fields=\'username,mail,name\')\n
username = profile[\'username\']\n first_name = profile[\'name\'].split()[0]\n
last_name = profile[\'name\'].split()[1]\n email = profile[\'mail\']\n password =
bcrypt.generate_password_hash(\'password\')\n user = User(username=username,
first_name=first_name, last_name=last_name, email=email,\n password=password,
roles=[role])\n db.session.add(user)\n db.session.commit()\n #
db.session.add(UserRole(user_id=user.id, role_id=role.id))\n\n
db.session.commit()\n\n\ndef seed_users():\n role, created =
get_or_create(db.session, Role,\n defaults={\'description\': \'for standard
users\'},\n name=\'ROLE_USER\')\n db.session.commit()\n non_standard_user_ids =
db.session.query(User.id) \\\n .filter(~User.roles.any(id=role.id)).all()\n\n
all_users_count = db.session.query(func.count(User.id)).all()[0][0]\n
all_users_count = db.session.query(User.id).count()\n\n #
User.query.filter(User.roles.any(UserRole.role_id.in_([1,2]))).count()\n
standard_users_count = db.session.query(User).filter(User.roles.any(UserRole.role
_id.in_([role.id]))).count()\n standard_users_count =
db.session.query(User.id).filter(\n User.roles.any(id=role.id)).count()\n\n
users_to_seed = 23\n users_to_seed -= standard_users_count\n
sys.stdout.write(\'[+] Seeding %d users\\n\' % users_to_seed)\n\n for i in
range(0, users_to_seed):\n profile =
fake.profile(fields=\'username,mail,name\')\n username = profile[\'username\']\n
# fake.first_name() fake.first_name_male() fake.first_name_female(), same for
last_name()\n first_name = profile[\'name\'].split()[0]\n last_name =
profile[\'name\'].split()[1]\n email = profile[\'mail\']\n password =
bcrypt.generate_password_hash(\'password\')\n user = User(username=username,
first_name=first_name, last_name=last_name, email=email,\n password=password)\n
user.roles.append(role)\n db.session.add(user)\n db.session.commit()\n\n
db.session.commit()\n\n\ndef seed_tags():\n sys.stdout.write(\'[+] Seeding
tags\\n\')\n pairs = []\n\n tag, created = get_or_create(db.session, Tag,
defaults={\'description\': \'shoes for everyone\'}, name=\'Shoes\')\n
tags.append(tag)\n pairs.append((tag, created))\n\n tag, created =
get_or_create(db.session, Tag, defaults={\'description\': \'jeans for
everyone\'}, name=\'Jeans\')\n tags.append(tag)\n pairs.append((tag,
created))\n\n tag, created = get_or_create(db.session, Tag,
defaults={\'description\': \'jackets for everyone\'}, name=\'Jackets\')\n
tags.append(tag)\n pairs.append((tag, created))\n\n tag, created =
get_or_create(db.session, Tag, defaults={\'description\': \'shorts for
everyone\'}, name=\'Shorts\')\n tags.append(tag)\n pairs.append((tag,
created))\n\n for pair in pairs:\n if pair[1]: # if created\n for i in range(0,
random.randint(1, 2)):\n pi = generate_image(TagImage)\n
pair[0].images.append(pi)\n\n db.session.add_all(tags)\n
db.session.commit()\n\n\ndef seed_categories():\n sys.stdout.write(\'[+] Seeding
categories\\n\')\n pairs = []\n category, created = get_or_create(db.session,
Category,\n defaults={\'description\': \'clothes for men\'},\n name=\'Men\')\n
categories.append(category)\n pairs.append((category, created))\n\n category,
created = get_or_create(db.session, Category,\n defaults={\'description\':
\'clothes for women\'}, name=\'Women\')\n categories.append(category)\n
pairs.append((category, created))\n\n category, created =
get_or_create(db.session, Category,\n defaults={\'description\': \'clothes for
kids\'}, name=\'Kids\')\n categories.append(category)\n pairs.append((category,
created))\n\n category, created = get_or_create(db.session, Category,\n
defaults={\'description\': \'clothes for teenagers\'}, name=\'Teenagers\')\n
categories.append(category)\n pairs.append((category, created))\n\n for pair in
pairs:\n if pair[1]: # if created\n for i in range(0, random.randint(1, 2)):\n
category_image = generate_image(CategoryImage)\n
pair[0].images.append(category_image)\n\n db.session.add_all(categories)\n
db.session.commit()\n\n\ndef seed_products():\n products_count =
db.session.query(func.count(Product.id)).all()[0][0]\n products_to_seed = 23\n
sys.stdout.write(\'[+] Seeding %d products\\n\' % products_to_seed)\n\n # tag_ids
= [tag[0] for tag in db.session.query(Tag.id).all()]\n # category_ids =
[category[0] for category in db.session.query(Category.id).all()]\n\n for i in
range(products_count, products_to_seed):\n name = fake.sentence()\n description =
fake.text()\n\n start_date = datetime.date(year=2017, month=1, day=1)\n
random_date = fake.date_between(start_date=start_date, end_date=\'+4y\')\n
publish_on = random_date\n product = Product(name=name, description=description,
price=fake.random_int(min=50, max=2500),\n stock=fake.random_int(min=5,
max=1000), publish_on=publish_on)\n\n #
product.tags.append(db.session.query(Tag).order_by(func.random()).first())\n\n
tags_for_product = []\n categories_for_product = []\n\n for i in range(0,
random.randint(1, 2)):\n tag_to_add = random.choice(tags)\n if tag_to_add.id not
in tags_for_product:\n product.tags.append(tag_to_add)\n
tags_for_product.append(tag_to_add.id)\n\n for i in range(0, random.randint(1,
2)):\n category_to_add = random.choice(categories)\n if category_to_add.id not in
categories_for_product:\n product.categories.append(category_to_add)\n
categories_for_product.append(category_to_add.id)\n\n for i in range(0,
random.randint(1, 2)):\n product_image = generate_image(ProductImage)\n
product.images.append(product_image)\n\n db.session.add(product)\n
db.session.commit()\n\n\ndef seed_comments():\n comments_count =
db.session.query(func.count(Comment.id)).scalar()\n comments_to_seed = 31\n
comments_to_seed -= comments_count\n sys.stdout.write(\'[+] Seeding %d
comments\\n\' % comments_to_seed)\n comments = []\n\n user_ids = [user[0] for
user in User.query.with_entities(User.id).all()]\n product_ids = [product[0] for
product in Product.query.with_entities(Product.id)]\n # equivalent:\n # user_ids
= [user[0] for user in db.session.query(User.id).all()]\n # product_ids =
[product[0] for product in db.session.query(Product.id).all()]\n\n for i in
range(comments_count, comments_to_seed):\n user_id = random.choice(user_ids)\n
product_id = random.choice(product_ids)\n rating = fake.random_int(min=1, max=5)
if fake.boolean(chance_of_getting_true=50) else None\n
comments.append(Comment(product_id=product_id,\n user_id=user_id,
rating=rating,\n content=fake.paragraph(nb_sentences=4,
variable_nb_sentences=True, ext_word_list=None)))\n\n
db.session.add_all(comments)\n db.session.commit()\n\n\ndef seed_addresses():\n
addresses_count = db.session.query(func.count(Address.id)).scalar()\n
addresses_to_seed = 30\n user_ids = [user[0] for user in
db.session.query(User.id).all()]\n\n for i in range(addresses_count,
addresses_to_seed):\n user_id = random.choice(user_ids) if
fake.boolean(chance_of_getting_true=80) else None\n\n first_name =
fake.first_name()\n last_name = fake.last_name()\n zip_code =
fake.zipcode_plus4() # postcode(), postalcode(), zipcode(), postalcode_plus4\n
street_address = fake.address()\n phone_number = fake.phone_number()\n city =
fake.city()\n country = fake.country()\n db.session.add(Address(user_id=user_id,
first_name=first_name, last_name=last_name, zip_code=zip_code,\n
street_address=street_address, phone_number=phone_number, city=city,
country=country))\n\n db.session.commit()\n\n\ndef seed_orders():\n orders_count
= db.session.query(func.count(Order.id)).scalar()\n orders_to_seed = 31\n
addresses = db.session.query(Address).options(load_only(\'id\',
\'user_id\')).all()\n products =
db.session.query(Product).options(load_only(\'id\', \'name\', \'slug\',
\'price\')).all()\n\n for i in range(orders_count, orders_to_seed):\n address =
random.choice(addresses)\n tracking_number = fake.uuid4()\n order_status =
fake.random_int(min=0, max=2)\n user_id = address.user_id\n order =
Order(tracking_number=tracking_number, order_status=order_status,
address_id=address.id,\n user_id=user_id)\n\n db.session.add(order)\n\n \'\'\'\n
this is to save the order now, so I can have the id to be used in order items\n or
the other way is to comment flush(), order_id=order.id, and session.add(oi).\n
Instead use order.order_items.append(oi); See below. Both ways lead to the same
result\n \'\'\'\n\n db.session.flush()\n\n for i in range(fake.random_int(min=1,
max=6)):\n product = random.choice(products)\n oi = OrderItem(name=product.name,
slug=product.slug, price=product.price,\n order_id=order.id,\n
product_id=product.id, user_id=user_id, quantity=fake.random_int(min=1, max=5))\n
db.session.add(oi)\n\n # order.order_items.append(oi)\n\n
db.session.commit()\n\n\nif __name__ == \'__main__\':\n seed_admin()\n
seed_users()\n seed_tags()\n seed_categories()\n seed_products()\n
seed_comments()\n seed_addresses()\n seed_orders()\n'
```