Использование SQLAlchemy

№ урока: 4 **Курс:** Flask

Средства обучения: Python3 и любимая среда разработки, например, PyCharm.

Обзор, цель и назначение урока

Повторить, что такое ORM и зачем она нужна, познакомиться с паттернами проектирования, которые используются в SQLAlchemy, научиться делать более сложные запросы с помощью ORM и узнать, как делать агрегационные запросы. Также обсудить подводные камни работы с ORM.

Изучив материал данного занятия, учащийся сможет:

- Использовать более сложные возможности SQLAlchemy.
- Писать продвинутую логику запросов с помощью SQLAlchemy.
- Понять внутреннее устройство SQLAlchemy.
- Избегать и использовать техники решения n+1 problem.

Содержание урока

1. Как устроена SQLAlchemy
2. Архитектурные паттерны, используемые в SQLAlchemy
3. Практика написания запросов с помощью SQLAlchemy
4. Подводные камни SQLAlchemy

Резюме

- **ORM -** инструмент Object Relational Mapper (ORM), который переводит классы Python в таблицы реляционных баз данных и автоматически преобразует вызовы функций в операторы SQL. ORM предоставляет стандартный интерфейс, который позволяет разработчикам создавать независимый от базы данных код для взаимодействия с широким спектром механизмов баз данных.
- **Паттерн проектирования Object pool** - порождающий паттерн (шаблон) проектирования, набор инициализированных и готовых к использованию объектов. Когда системе требуется объект, он не создаётся, а берётся из пула. Когда объект больше не нужен, он не уничтожается, а возвращается в пул.
- **Unit of Work -** с помощью этого шаблона система прозрачно отслеживает изменения объектов и периодически сбрасывает все эти ожидающие изменения в базу данных. Это означает, что все модификации, отслеживаемые сессиями (Units of Works), будут применены к базе данных вместе, или ни одна из них не будет применена. Другими словами, он используется для обеспечения согласованности базы данных.
- Преимущества использования ORM следующие:
    - Принцип DRY: вы пишете свою модель данных только в одном месте, и вам проще обновлять, поддерживать и повторно использовать код.
    - Вы не привязываетесь конкретно к одной базе данных, ORM позволяет гибко менять базу данных, при этом, не меняя запросы, с помощью диалектов.
    - Многое делается автоматически, от обработки базы данных до I18N.
    - Заставляет вас писать код MVC, что в итоге делает ваш код немного чище.
    - Использование подготовленных операторов или транзакций так же просто, как вызов метода.

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Недостатки использования ORM:
    - Главный недостаток в том, что ORM это отдельная библиотека и чаще всего у каждой ORM свой синтаксис, который отличается от синтаксиса других ORM и от SQL. SQLAlchemy старается как можно больше быть похожей на обычный SQL.
    - Нужно знать о подводных камнях, например, n+1 problem.
    - Кроме того, ORM приносят некий overhead на скорость выполнения запроса.

- **Что такое связи между таблицами?** - в реляционной базе данных отношения позволяют предотвратить избыточные данные. Например, при разработке базы данных, которая будет отслеживать информацию о книгах, может быть таблица "Названия", в которой хранится информация о каждой книге, например название книги, дата публикации и издатель. Существует также информация, которую вы можете хранить об издателе, например, номер телефона издателя, адрес и почтовый индекс. Если вы храните всю эту информацию в таблице "Названия", номер телефона издателя будет дублироваться для каждого названия, которое печатает издатель.
  Лучшим решением является хранение информации издателя только один раз, в отдельной таблице, которую мы будем называть "Издатели". Затем вы поместите указатель в таблице "Названия", которая ссылается на запись в таблице "Издатели".
  Чтобы убедиться, что данные остаются синхронизированными, можно обеспечить целостность данных между таблицами. Отношения целостности данных помогают убедиться, что информация в одной таблице соответствует информации в другой. Например, каждое название в таблице "Названия" должно быть связано с конкретным издателем в таблице "Издатели". Название не может быть добавлено в базу данных для издателя, которого не существует в базе данных.

- Связи "**один ко многим**".
  Связь "один ко многим" является наиболее распространенным типом связи. В такого рода связях строка в таблице A может иметь много строк в таблице B. Но строка в таблице B может иметь только одну строку в таблице A. Например, таблицы "Издатели" и "Названия" имеют связь "один ко многим". То есть, каждый издатель выпускает много названий. Но каждое название принадлежит только одному издателю.
- Связи "**многие ко многим**".
  В связи "многие ко многим" строка в таблице A может иметь много совпадающих строк в таблице B, и наоборот. Вы создаете такую связь, определяя третью таблицу, которая называется промежуточной таблицей. Первичный ключ промежуточной таблицы состоит из внешних ключей как таблицы A, так и таблицы B. Например, таблица "Авторы" и таблица "Названия" имеют связь "многие ко многим", которая определяется связью "один ко многим" из каждой из этих таблиц к таблице "TitleAuthors".
- Связи "**один к одному**".
  В связи "один к одному" строка в таблице A может иметь не более одной совпадающей строки в таблице B, и наоборот. Связь "один к одному" создается, если оба связанных столбца являются первичными ключами или имеют уникальные ограничения.

- **N plus one problem** - проблема N+1 является общим побочным эффектом шаблона отложенной загрузки (lazy loading), при работе с сущностями, которые содержат коллекции других сущностей. Первый запрос выберет только корневые сущности, а каждая связанная коллекция будет загружена отдельным запросом. Таким образом ORM выполняет N+1 SQL запросов, где N — количество корневых сущностей в результирующей выборке запроса. То есть, эта проблема появляется только при отношениях Many To Many или One to Many.
- N плюс одна проблема решается благодаря активной или нетерпеливой загрузке (eager loading). Идея такая: разработчик должен заранее сообщить ORM, что ему потребуются дополнительные данные. Затем ORM может забрать эти данные заранее («eager

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

loading»). При активной загрузке связанные данные выбираются вместе с родительским объектом.

Закрепление материала

- Что такое ORM?
- Назовите преимущества и недостатки ORM.
- Что такое паттерн проектирования Object pool?
- Что такое паттерн Unit of Work?
- Какие бывают связи между таблицами?
- Что такое N+1 problem?
- Когда возникает N+1 problem?
- Как избежать N+1 problem?

Дополнительное задание

Задание

Создайте CRUD-запросы (create, read, update, delete) для модели Actor и добавьте их во View Actors, по примеру с Films. Также, создайте таблицу каскадеров, которые будут иметь связь один-к-одному с Актёрами. В этом классе определите следующие поля: id – Primary key, name - String, active – Boolean, actor_id – ссылка на актера Foreign Key, actor – связь с таблицей актеров. Для указания связи один-к-одному используется аргумент uselist=False.

Самостоятельная деятельность учащегося

Задание 1
Выучите основные понятия, рассмотренные на уроке.

Задание 2
Изучите материалы из рекомендуемых ресурсов.

Задание 3
Напишите агрегационные запросы с помощью SQLAlchemy: найдите минимальную и максимальную продолжительность фильма, найдите количество фильмов, которые были сняты Disney и найдите максимальное количество актеров, которые снимались в одних и тех же фильмах.

Рекомендуемые ресурсы

SQLAlchemy docs:
https://docs.sqlalchemy.org/en/13/

Паттерны проектирования, используемые в SQLAlchemy:
https://techspot.zzzeek.org/2012/02/07/patterns-implemented-by-sqlalchemy/

N+1 problem:
https://docs.sqlalchemy.org/en/13/glossary.html#term-N-plus-one-problem

Техники загрузки сущностей, которые внутри объектов в SQLAlchemy:
https://docs.sqlalchemy.org/en/13/orm/loading_relationships.html

SQLAlchemy: Python vs Raw SQL:

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

https://www.youtube.com/watch?v=jUGK-CtM-Mk&t=1689s

Основы SQL:
https://www.youtube.com/watch?v=WpojDnclWOw

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com