Automatic REST API For SQLAlchemy

Duy Nguyen, Kaikang Zhu

Abstract

REST API has become a standard of any modern web application. However, creating such API still involve a lot of template code which kept developer from being productive with their project. We observed that most web application are divided into a front-end component and a back-end component. The back-end can be made of Python, typically composed of a database layer which some are using ORM like sqlalchemy and an API server(Flask for example). The front-end tends to be made in Javascript front-end framework or library that served as an API client. We can optimize this pipeline by encapsulating the database, the API server and the API client in one framework. If we have access to the sqlalchemy's database model, we can generate an API interface for said model. This would allow us to combine both the database ORM and the API together. Further, we can also extend ORM functionality to the client.

Goal

Client side: Have a library like SQLAlchemy but a lot smaller. Client side can send REST API automatically to the server by calling method in this library directly.

Server side: Automatically read REST API calls and automatically query the data and send it back to client side.

Data Model

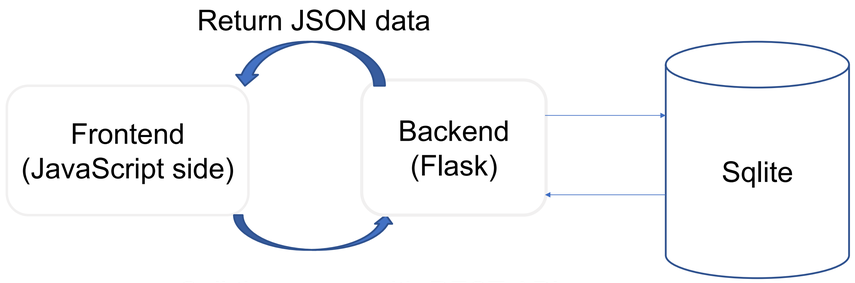
- User class: id, name, fullname, nickname

Info: id is the primary key.

- Address class: id, email_address, user_id

Info: id is the primary key, user_id is foreign key referred to User.id.

Overview



JavaScript Phase

1. Initialize a query object and pass in URL of server.
2. Call needed functions.
3. Pass in required parameters.
4. Generate API call.

Flask Phase

1. Read API call.
2. Make SQLAlchemy query.
3. Dump result into JSON format.

Sample procedure

If client side wants to find all the user name equals to Jones, it can call filter() method from the library.

Query_object.filter(name="Jones")

If client side wants to get all records matching the criteria, it calls all() from the library.

Query_object.filter(name="Jones").all()

Instead of using template to call REST API, such as,

var xhr = new XMLHttpRequest()
xhr.setRequestHeader(…)
xhr.send()…

The javascript code interact with REST API without directly making HTTP request.

Result

Constructs we support

- Filter, including equality, <, >, <=, >= and or and and.
- OrderBy, including ascending and descending.
- Join, including implicit join and explicit join.

Future Improvement

- If we have enough time, we would try to support a bigger set of constructs than listed above.
- Try to support more complex queries in different situations and more tables that just two tables.
- Optimize for developers experience.

Reference

- http://zike.io/posts/Full-Stack-Beginner-s-Guide-Flask-React-SQLAlchemy/
- https://stackoverflow.com/questions/5022066/how-to-serialize-sqlalchemy-result-to-json
- https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world