API Wrapper

ORM vs sqlalchemy

ORM – Object relational mapping

* Uses the active record implementation
* Each row in database is directly related to an object in code vice versa
* Doesn’t’ require pre-defining the schema to use properties in code.

Sqlalcehmy

* Uses data mapper implementation
* Separations between database structure and object structure

Spotify Api Project

The api wrapper here is basically an class that can be used to add functions that the api might not have, like for example

Wrappers are made to help out those of us who might not understand how to use the API directly, but I’ve found that they are mostly used to add features

Authentication and authorization ;

* **Authentication** is the verification of the credentials of the connection attempt. This process consists of sending the credentials from the remote access client to the remote access server in an either plaintext or encrypted form by using an authentication protocol.
* **Authorization** is the verification that the connection attempt is allowed. Authorization occurs after successful authentication.

**In other words:** Authentication is stating that you are who are you are and Authorization is asking if you have access to a certain resource.

1. The client has registered with the authorization server.
2. The client has received its client ID and client secret provided by the authorization server.

Client Credentials Flow:

1. Sending the credentials to authorization server
2. Validating the credentials and return an access token
3. Using the access token to access resources

An OAuth 2.0 flow works as follows:

1. A client application makes a request for the user to authorize access to their data.
2. If the user grants access, the application then requests an access token from the service provider, passing the access grant from the user and authentication details to identify the client.
3. The service provider validates these details and returns an access token.
4. The client uses the access token to request the user data via the service provider.

Access tokens in this sample only contains information regarding its issuer and expiration date.

Django orm vs sqlalchemy

* Django orm uses the active record design pattern where each table in database is wrapped
* Inside the model and each objecy of that model will be ltightly attache to a database record
* No need to define model properties
* active Record ORMs usually have built-in CRUD (Create — Read — Update — Delete) methods like save(), create(), … supporting object manipulation.

Sql alchemy uses the data wrapper

Data Mapper ORMs, instead of directly connecting each object to the corresponding database record, the ORM will serve as a layer in charge of separating and transferring data bidirectionally between database and application code. This means that application objects have no knowledge or information about database schema. Created objects will have no ideas about the existence of the database and vice versa.

The Person objects returned by this query will be deferred model instances(see [defer()](https://www.bookstack.cn/read/django-v3.0/66e6abec22a16e77.md#django.db.models.query.QuerySet.defer)). This means that thefields that are omitted from the query will be loaded on demand.

One of the most powerful parts of Django is the automatic admin interface. It reads metadata from your models to provide a quick, model-centric interface where trusted users can manage content on your site. The admin’s recommended use is limited to an organization’s internal management tool. It’s not intended for building your entire front end around.