Previous Next

Creating the Models

Estimated Time: 2-3 hours

In this lesson you will be starting to create the **models** which represent the different elements of TBay - users, items and bids. SQLAlchemy models are used to create tables in the database. They describe the structure and layout of the data which will be held in the tables. Instances of the model then represent rows of data in the database.

Engines, Bases and Sessions

To get started, create a new project in a directory called *tbay*, and `cd` into the directory. Then initialize and activate a virtual environment:

```
sudo apt-get update
sudo apt-get install python3.4-venv
python3 -m venv env
source env/bin/activate
```

Next, install SQLAlchemy and psycopg2 using `pip install sqlalchemy psycopg2` . Then create a new Postgres database for the project by running: `createdb tbay` .

Now you can get started on the Python code. Create a new file called *tbay.py* and add the following code:

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from sqlalchemy.ext.declarative import declarative_base


engine = create_engine('postgresql://ubuntu:thinkful@localhost:5432/tb
Session = sessionmaker(bind=engine)
session = Session()
Base = declarative_base()
```

Here three key components of SQLAlchemy are introduced. First the **engine** is created. This will talk directly to your database using the raw SQL commands.

Take a look at the long connection string passed to `create_engine`. Let's break this down:

- `postgresql://` - Connect to a Postgres database
- `ubuntu:thinkful` - Connect as the user `ubuntu`, with the password `thinkful`. `ubuntu` is the name of the user we set up in Cloud9.
- `@localhost` - Connect to a database running on the same machine as the code
- `:5432` - Connect to a database listening on port 5432 (the default Postgres port)
- `/tbay` - Connect to the database called `tbay`

Next the **session** is created. This is the equivalent to a psycopg2 cursor - it allows you to queue up and execute database transactions. Multiple sessions can take place on a single database simultaneously.

Finally a **declarative base** is created. This acts like a repository for the models, and will issue the `create table` statements to build up the database's table structure.

Your first model

Now that you have everything set up to talk to a database it's time to create a model. The first model you will create is the Item model. Add the following code to *tbay.py*:

```
from datetime import datetime

from sqlalchemy import Column, Integer, String, DateTime

class Item(Base):
    __tablename__ = "items"

    id = Column(Integer, primary_key=True)
    name = Column(String, nullable=False)
    description = Column(String)
    start_time = Column(DateTime, default=datetime.utcnow)
```

The Item model is represented by a class called `Item`. Subclassing `Base` registers the Item model with the declarative base, so when you later ask for the tables to be created, the Item table will be included. Inside the class there is a string `__tablename__`, which will be used to name the items table in the database.

Then there are objects which represent the four columns that the table should contain. Firstly there is the `id` column. This is an integer primary key, which will be used to uniquely identify each item. Next there is the name of the item, a string which has the `not null` constraint applied to it. Then there is the description of the item, also a string. Finally there is the auction start time, which contains a `DateTime` object. This has a value which defaults to the current UTC time.

To actually create the table there has to be a `create table` command issued. This will be issued by the declarative base. Add the following line to the bottom of *tbay.py*:

```
Base.metadata.create_all(engine)
```

This creates a new table for each subclass of Base, ignoring any tables which already exist in the database.

Run your script using `python tbay.py`. Then make sure that the items table was created successfully by running `psql -d tbay`, and entering `\d+ items`. You should see the four columns that your table contains printed out. Then exit the Postgres interpreter by entering `\q`.

Challenge: Create the user and bid models

Try to create the two further models which you will use for the auction site.

The user model should contain three columns:

- An integer id, which is the primary key
- A username string, which cannot be null
- A password string, which cannot be null

The bid model should contain two columns:

- An integer id, which is the primary key
- A floating-point price, which cannot be null

Previous

Next