Previous Next

Creating and Querying Data

Estimated Time: 2-3 hours

In the last assignment you set up the models (and the equivalent tables) to allow you to store users, items and bids for your auction site. In this assignment you'll use these models to create, retrieve, update and delete rows from your database.

Creating rows

To create a row you need to create an instance of the model. You can set the values of the columns by setting properties of the model, in the same way as you would with any other class. For example, to create a user you could use the following code:

```
beyonce = User()
beyonce.username = "bknowles"
beyonce.password = "uhohuhohuhohohnana"
session.add(beyonce)
session.commit()
```

First an instance of the User class is created, and the username and password are set. Note that SQLAlchemy creates a default `__init__` method for models, so you could also say: `beyonce = User(username="bknowles", password="uhohuhohuhohohnana")`. Then the `beyonce` object is added to the session. This queues up the instruction to create the row in the database,

ready to be executed. Finally you commit the session, which will actually run the SQL command and create the row.

Try it!

Open up a Python interpreter in the same directory as your *tbay.py* script, and import the session and your models using: `from tbay import User, Item, Bid, session` . Create two users, and two items, and commit them to the database.

Querying for rows

To query for rows you can use the `session.query` method. This provides a flexible way to perform the queries which you have written in SQL up to this point. The following example covers the most common usages of the query interface:

```
# Returns a list of all of the user objects
# Note that user objects won't display very prettily by default -
# you'll see their type (User) and their internal identifiers.
session.query(User).all() # Returns a list of all of the user objects

# Returns the first user
session.query(User).first()

# Finds the user with the primary key equal to 1
session.query(User).get(1)

# Returns a list of all of the usernames in ascending order
session.query(User.username).order_by(User.username).all()

# Returns the description of all of the basesballs
session.query(Item.description).filter(Item.name == "baseball").all()

# Return the item id and description for all baseballs which were crea
session.query(Item.id, Item.description).filter(Item.name == "baseball
```

Try it!

Try to make up four different queries, which will find each of your two users and two items. Try using a different filtering technique for each query. Experiment with returning different parts of the model in your queries.

Updating rows

Updating rows using SQLAlchemy is fairly straightforward. You simply need to query for a row, and then change an attribute of the returned object. For example:

```
user = session.query(User).first()
user.username = "solange"
session.commit()
```

First you query for the first user, change their username, and commit the changes to store the new username.

Try It!

Change the names and descriptions of your two items.

Deleting rows

To delete a row you need to use the `session.delete` function. For example:

```
user = session.query(User).first()
session.delete(user)
session.commit()
```

Again, you query for the first user, then call `session.delete` to queue up the delete command. This will execute when `session.commit` is called.

Try It!

Delete all of the the items and users from your database. In the next assignment you'll rebuild the database from scratch, and add in relationships between the different parts of the system.

☆ ☆ ☆ ☆ ☆ · Report a typo or other issue

Mark as completed

Previous Next