# Recitation 8

- Flask Models

## Creating a SQLite Database -Installation

- Download the appropriate files from https://www.sqlite.org/download.html
- For Windows, follow this
  - https://www.youtube.com/watch?v=wXEZZ2JT3-k
- For Ubuntu, follow this
  - https://www.youtube.com/watch?v=C16QgidWZsU
- For Mac, follow this
  - https://tableplus.io/blog/2018/08/download-install-sqlite-for-mac-osx-in-5-minutes.html

## Creating the SQLite Database

- After the download and installation of sqlite3, we need to create a database
- Example: Creating a database named *simpledb.db* in *C:/database/*

```
C:\database>sqlite3 simpledb.db
SQLite version 3.25.2 2018-09-25 19:08:10
Enter ".help" for usage hints.
sqlite> .tables
sqlite> .exit

C:\database>
```

No table yet in the database simpledb.db

## Creating a Table in simpledb.db using Flask-SQLAlchemy

yourapp.py

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] =
'sqlite:///C:/database/simpledb.db'

db = SQLAlchemy(app)

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
```

Giving the URI of the Database

Setting the database columns

## Creating a Table in simpledb.db using Flask-SQLAlchemy

- Assuming that you saved yourapp.py in C:/flask_proj,

```
C:\flask_proj>python
>>>from yourapp import db
>>>db.create_all()
```

- Now if you check the database

```
C:\database>sqlite3 simpledb.db
SQLite version 3.25.2 2018-09-25 19:08:10
Enter ".help" for usage hints.
sqlite> .tables
user
```

Table name *user* is created in the database simpledb.db

# Inserting to the Database

- Inserting data in a database is a 3-step process
  - Create the Python object
  - Add it to the session
  - Commit the session

```
>>> from yourapp import User
>>> me = User('admin', 'admin@example.com')
>>> db.session.add(me)
>>> db.session.commit()
>>>me.id
1
```

**add()** issues an INSERT statement for the database

## Deleting from the Database

```
>>> from yourapp import User
>>> me = User('admin', 'admin@example.com')
>>> db.session.delete(me)
>>> db.session.commit()
```

## Querying From the Database

user

| id | username | email |
|---|---|---|
| 1 | admin | admin@example.com |
| 2 | peter | peter@example.org |
| 3 | guest | guest@example.com |

Retrieving a user by username

```
>>> peter = User.query.filter_by(username='peter').first()
>>> peter.id
2
>>> peter.email
u'peter@example.org'
```

# Querying From the Database

Ordering Users by something

```
>>> User.query.order_by(User.username).all()
[<User u'admin'>, <User u'guest'>, <User u'peter'>]
```

Limiting users

```
>>> User.query.limit(1).all()
[<User u'admin'>]
```

Getting users by primary key

```
>>> User.query.get(1)
<User u'admin'>
```

# Updating A Database

```
>>> update_this = User.query.filter_by(id = 3).first()
>>>update_this.username = ‘henry’
>>> db.session.commit()
```

| id | username | email |
|---|---|---|
| 1 | admin | admin@example.com |
| 2 | peter | peter@example.org |
| 3 | **henry** | guest@example.com |

## Model Relationships - One-to-Many

```
class Person(db.Model):
        id = db.Column(db.Integer, primary_key=True)
        name = db.Column(db.String(20))
        pets = db.relationship('Pet', backref = 'owner')

class Pet(db.model):
        id = db.Column(db.Integer, primary_key=True)
        name = db.Column(db.String(20))
        owner_id = db.column(db.Integer, db.ForeignKey('person.id'))
```

# Example

Person

| id | name |
|---|---|
| 1 | Anthony |
| 2 | Michelle |

Adding pets to the Pet table

```
>>>anthony = Person.query.filter_by(name = ‘Anthony’).first()
>>>michelle = Person.query.filter_by(name = ‘Michelle’).first()
>>> spot = Pet(name = ‘Spot’, owner = anthony)
>>>brian = Pet(name = “Brian’, owner = michelle)
>>>clifford = Pet(name = “Clifford’, owner = anthony)
```

## Example

Pet

| id | name | owner_id |
|---|---|---|
| 1 | Spot | 1 |
| 2 | Brian | 2 |
| 3 | Clifford | 1 |

Accessing Pets from an owner enrtry in Person table

```
>>>anthony.pets
[<Pet1>, <Pet3>]
>>> anthony.pets[0].name
‘Spot’
```

# Many-to-Many Relationship

```
subs = db.table('subs',
        db.Column('user_id', db.Integer, db.ForeignKey('user.user_id')),
        db.Column('channel_id', db.Integer, db.ForeignKey('channel.channel_id'))
 )
class User(db.Model):
            user_id = db.Column(db.Integer, primary_key=True)
            name = db.Column(db.String(20))
            subscriptions = db.relationship('Channel', secondary = subs, backref = db.backref('subscribers', lazy =
'dynamic'))

class Channel(db.model):
            channel_id = db.Column(db.Integer, primary_key=True)
            channel_name = db.Column(db.String(20))
```

## Example

User

| user_id | name |
|---|---|
| 1 | Anthony |
| 2 | Stacy |
| 3 | George |

Channel

| channel_id | channel_name |
|---|---|
| 1 | tedx |
| 2 | Ted_ed |

## Example

- Subscribing a particular user to a channel

Adding to the association table

```
>>>channel1 = Channel.query.filter_by(id = 1).first()
>>>user2 = User. query.filter_by(id = 2).first()
>>>channel1.subscribers.append(user2)
>>>db.session.commit()
```

## Example - Association Table

subs

| user_id | channel_id |
|---|---|
| 2 | 1 |

Similarly, we can fill this up to create multiple many-to-many channel-subscriber relations by using the append() function multiple times

subs

| user_id | channel_id |
|---|---|
| 1 | 2 |
| 2 | 1 |
| 3 | 2 |
| 2 | 2 |
| 3 | 1 |

## Lazy Parameter

- Consider the example from slide 11, but with the lazy parameter set

lazy.py

```
class Person(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(20))
    pets = db.relationship('Pet', backref='owner', lazy='dynamic')


class Pet(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(20))
    owner_id = db.Column(db.Integer, db.ForeignKey('person.id'))
```

## Setting lazy = ‘dynamic’

```
C:\flask_proj>python
>>>from lazy import *
>>>person1 = Person.query.filter_by(id = 1)
>>>person1.pets
>>><sqlalchemy.orm.dynamic.AppenderBaseQuery object at 0x7f2345c7c8d0>
>>>person.pets.all()
>>>[<Pet1>, <Pet3>]
>>>person.pets.filter_by(id = 1).all()
[<Pet1>]
```

Gives us the opportunity to query the results from the database rather than returning all the results at once

## Setting lazy = ‘select’/ lazy = ‘true’

```
C:\flask_proj>python
>>>from lazy import *
>>>person1 = Person.query.filter_by(id = 1)
>>>person1.pets
>>>[<Pet1>, <Pet3>]
```

Returns all the Pets at once and does not give us the opportunity to query results

- For more details, see this https://youtu.be/g1oFlq7D_nQ

## References

- https://www.youtube.com/playlist?list=PLXmMXHVSvS-BlLA5beNJojJLlpE0PJgCW
- http://flask-sqlalchemy.pocoo.org/2.3/