CS1520 Recitation: Flask 4: Datamodel

...

Jeongmin Lee

NOTICE: No recitation next week!

- Due to my dental surgery. (Also no regular office hour on Friday)
- Instead, I will have additional office hours to answer questions if you have any.
  - 14th, Monday (4:00-5:00pm) RM 5505 SENSQ
  - 15th, Tuesday (4:00-5:00pm) RM 5317 SENSQ
  - 16th, Wednesday (3:00-4:00pm) RM 5505 SENSQ

- One week after, I will start to cover topics of Ajax and so on.

Plans

- Data Insertion
- Data Selection
- Data Delete
- Query with Filters

Inserting Data

- Inserting data into the database is a three step process:

  1. Create the Python object
  2. Add it to the session
  3. Commit the session

- Here’s an example case of User object

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test.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)
    major = db.Column(db.String(80), unique=False, nullable=True)
    email = db.Column(db.String(120), unique=True, nullable=False)
    # NOTE: without constructor, the implicit constructor
    #       from db.Model allows us to create new elements
    def __repr__(self):
        return '<User %r>' % self.username
```

- Create DB fist.

```
db.create_all()
```

- Here’s an example case of User object

```
me = User(username='admin0', email='admin0@example.com')
db.session.add(me)
guest = User(username='guest0', email='guest1@example.com')
db.session.add(guest)
db.session.commit()
```

- NOTE: user id is assigned after the commit!

```
me.id
```

Deleting Data

- Deleting records by “delete()”

- Here’s an example case of User object

```
db.session.delete(me)
db.session.commit()
```

Querying Data

- You can retrieve data by filter() method :
- adding .first() gets you the first record of retrieved ones
- adding .all() gets you all records (returns a list)

Querying Data

- You can retrieve data by **filter()** method or filter_by() method:
  - `User.query.filter_by(major='CS').all()`

    as an argument
  - `User.query.filter(User.major=='CS').all()`

    as a boolean statement : more versatile!

- Let’s add some more records

```
user1 = User(username='peter', major='CS', email='peter@pitt.edu')
user2 = User(username='jane', major='CS', email='jane@pitt.edu')
user3 = User(username='tim', major='CS', email='tim@pitt.edu')
db.session.add(user1)
db.session.add(user2)
db.session.add(user3)
db.session.commit()
```

```
cs_students = User.query.filter_by(major='CS').all()
cs_student = User.query.filter_by(major='CS').all()[0]
cs_student.username
cs_student.email
cs_student.major


cs_student = User.query.filter_by(major='CS').first()

cs_student = User.query.filter(User.major=='CS').first()
```

Querying With Filters

- equals:
  User.query.filter(User.username == 'peter')

- not equals:
  User.query.filter(User.username != 'peter')

- like:
  User.query.filter(User.username.like('%et%'))

Querying With Filters

- starts with:
  User.query.filter(User.username.startswith('pet'))

- ends with:
  User.query.filter(User.username.endswith('er'))

Querying With Filters

- IN:
  User.query.filter(User.username.in_(['ed', 'jane', 'tim']))

- Not IN:
  User.query.filter(~User.username.in_(['ed', 'jane', 'tim']))

- IS Null:
  User.query.filter(User.username is None)

Querying With Filters

- Is Not Null:
  User.query.filter(User.username is not None)