CS1520 Recitation: Flask 5: Examples in Model Relation

•••

Jeongmin Lee

Plans

- Basics of Model
- Relation Example 1
- Relation Example 2

Note: this slide is largely based on materials by Eueung Mulyana and other:
http://eueung.github.io/python/flask-sqlalchemy/#1
https://techarena51.com/blog/many-to-many-relationships-with-flask-sqlalchemy/

Basic Model

- Let’s have a model on User with these attributes:
    - id
    - username
    - email

Basic User Model

```python
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

#------------------------------------------
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = True
db = SQLAlchemy(app)
#------------------------------------------
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True)
    email = db.Column(db.String(120), unique=True)

    def __init__(self, username, email):
        self.username = username
        self.email = email

    def __repr__(self):
        return '<User %r>' % self.username
```

Basic User Model

```
db.create_all()

admin = User('admin', 'admin@example.com')

guest = User('guest', 'guest@example.com')

db.session.add(admin)

db.session.add(guest)

db.session.commit()

users = User.query.all()

print(users)

admin = User.query.filter_by(username='admin').first()

print(admin)
```

Relation Example One

- Let's have a model on Person and Address
  - Each person can have multiple addresses (email)
- This is One-to-Many model

```python
from flask import Flask
from flask.ext.sqlalchemy import SQLAlchemy
#------------------------------------------
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = True
db = SQLAlchemy(app)
#------------------------------------------
class Person(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(50))
    addresses = db.relationship('Address', backref='person', lazy='dynamic')
    def __init__(self, name):
        self.name = name
    def __repr__(self):
        return '<Person %r>' % self.name

class Address(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(120), unique=True)
    person_id = db.Column(db.Integer, db.ForeignKey('person.id'))
    #person = db.relationship('Person', backref=db.backref('addresses', lazy='dynamic'))
    def __init__(self, email, pers):
        self.email = email
        self.person_id = pers.id
    def __repr__(self):
        return '<Address %r>' % self.email
```

```
db.create_all()

otong = Person('otong')

ujang = Person('ujang')

db.session.add(otong)

db.session.add(ujang)

db.session.commit()

otongemail1 = Address('otong@example.com',otong)

otongemail2 = Address('otong@nasa.com',otong)

ujangemail = Address('ujang@example.com',ujang)

db.session.add(otongemail1)

db.session.add(otongemail2)

db.session.add(ujangemail)

db.session.commit()

print(otong.addresses.all())

print(otong.addresses.first())

print(ujang.addresses.all())

print(otongemail1.person)
```

Relation Example Two

- Let’s have a model on Page and Tag
  - Each page can have multiple tags
  - Each tags can also have multiple pages
- This is Many-to-Many model

```python
from flask import Flask
from flask.ext.sqlalchemy import SQLAlchemy
#-----------------------------------------
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test2.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = True
db = SQLAlchemy(app)
#-----------------------------------------
tags = db.Table('tags',
    db.Column('tag_id', db.Integer, db.ForeignKey('tag.id')),
    db.Column('page_id', db.Integer, db.ForeignKey('page.id'))
)
class Page(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(80))
    body = db.Column(db.Text)
    tags = db.relationship('Tag', secondary=tags, backref=db.backref('pages', lazy='dynamic'))
    def __init__(self, title):
        self.title = title
    def __repr__(self):
        return '<Page %r>' % self.title
class Tag(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    label = db.Column(db.String(50))
    def __init__(self, label):
        self.label = label
    def __repr__(self):
        return '<Tag %r>' % self.label
```

1) Two Models for Page and Tag

```
from flask import Flask
from flask.ext.sqlalchemy import SQLAlchemy
#-------------------------------------------
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = True
db = SQLAlchemy(app)
#-------------------------------------------
tags = db.Table('tags',
    db.Column('tag_id', db.Integer, db.ForeignKey('tag.id')),
    db.Column('page_id', db.Integer, db.ForeignKey('page.id'))
)
class Page(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(80))
    body = db.Column(db.Text)
    tags = db.relationship('Tag', secondary=tags, backref=db.backref('pages', lazy='dynamic'))
    def __init__(self, title):
        self.title = title
    def __repr__(self):
        return '<Page %r>' % self.title
class Tag(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    label = db.Column(db.String(50))
    def __init__(self, label):
        self.label = label
    def __repr__(self):
        return '<Tag %r>' % self.label
```

```python
from flask import Flask
from flask.ext.sqlalchemy import SQLAlchemy
#------------------------------------------
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = True
db = SQLAlchemy(app)
#------------------------------------------
tags = db.Table('tags',
    db.Column('tag_id', db.Integer, db.ForeignKey('tag.id')),
    db.Column('page_id', db.Integer, db.ForeignKey('page.id'))
)
class Page(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(80))
    body = db.Column(db.Text)
    tags = db.relationship('Tag', secondary=tags, backref=db.backref('pages', lazy='dynamic'))
    def __init__(self, title):
        self.title = title
    def __repr__(self):
        return '<Page %r>' % self.title
class Tag(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    label = db.Column(db.String(50))
    def __init__(self, label):
        self.label = label
    def __repr__(self):
        return '<Tag %r>' % self.label
```

2) Helper table which consists of two Foreign Keys

- One foreign key to the Post table
- Another foreign key to the Tags table

**It is not a db.model but a db.Table

```python
from flask import Flask
from flask.ext.sqlalchemy import SQLAlchemy
#-------------------------------------------
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = True
db = SQLAlchemy(app)
#-------------------------------------------
tags = db.Table('tags',
    db.Column('tag_id', db.Integer, db.ForeignKey('tag.id'))
    db.Column('page_id', db.Integer, db.ForeignKey('page.id')
)
class Page(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(80))
    body = db.Column(db.Text)
    tags = db.relationship('Tag', secondary=tags, backref=db.backref('pages', lazy='dynamic'))
    def __init__(self, title):
        self.title = title
    def __repr__(self):
        return '<Page %r>' % self.title
class Tag(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    label = db.Column(db.String(50))
    def __init__(self, label):
        self.label = label
    def __repr__(self):
        return '<Tag %r>' % self.label
```

- **Tags**: Is the child table which I want to create the relationship with.
- **secondary=relationship_table**: specify the associative table (relationship_table).
- **backref='posts'**: the *backref* argument will create a page attribute on the child table.

we will assume tags will be under page (implication:) pages made first, then tags are added to them.

```
db.create_all()

tagpython = Tag('python')

tagtuts = Tag('tutorial')

tagjava = Tag('java')

db.session.add(tagpython)

db.session.add(tagjava)

db.session.add(tagtuts)

#db.session.commit()


pagepython1 = Page('pagepython 1')

pagepython2 = Page('pagepython 2')

pagejava = Page('pagejava')

db.session.add(pagepython1)

db.session.add(pagepython2)

db.session.add(pagejava)

#db.session.commit()
```

```
pagepython1.tags.append(tagpython)

pagepython1.tags.append(tagtuts)

pagepython2.tags.append(tagpython)

pagejava.tags.append(tagjava)

db.session.commit()

print(tagpython.pages.all())

print(pagepython1.tags)
```