Flask-SQLAlchemy

Eueung Mulyana

http://eueung.github.io/python/flask-sqlalchemy

Python CodeLabs |

- Basics #1, #2
- Relationships #3, #4, #5
- Declarative (ORM) #6
- Manual OR Mapping #7
- SQL Abstraction Layer #8

Basics

Example #1

```
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 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
```

```
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
```

```
[<User u'admin'>, <User u'guest'>]
<User u'admin'>
```

- `unicode('abcdef')` -> `u'abcdef'`
- `u'hello world'.encode('utf-8')` or `str(u'hello world')`

Example #2

```
3
1
admin@example.com
True
[<User u'admin'>, <User u'guest'>]
[<User u'admin'>, <User u'guest'>]
[<User u'admin'>, <User u'guest'>]
[<User u'admin'>]

<User u'admin'>
```

- This will raise 404 errors instead of returning None: `get_or_404()` or `first_or_404()` (for view functions)

```
db.create_all()

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

db.session.add(admin)
db.session.add(guest)
db.session.add(me)
db.session.commit()

print me.id #after commit

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

admin = User.query.filter_by(username='admin').first()
print admin.id
print admin.email

missing = User.query.filter_by(username='missing').first()
print missing is None

print User.query.all()
print User.query.filter(User.email.endswith('@example.com'
print User.query.order_by(User.username).all()
print User.query.limit(1).all() # 1 user

print User.query.get(1) # by primarykey
```

Relationships

```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
    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('perso
    #person = db.relationship('Person', backref=db.backref
    def __init__(self, email,pers):
        self.email = email
        self.person_id = pers.id

    def __repr__(self):
        return '<Address %r>' % self.email
```

Example #3

- How does it know that this will return more than one address? Because SQLAlchemy **guesses** a useful default from your declaration.
- If you would want to have a one-to-one relationship you can pass `uselist=False` to `relationship()`.
- Two possibilities (TBT)

See: Declaring Models

Example #3

```
[<Address u'otong@example.com'>, <Address u'otong@nasa.com
<Address u'otong@example.com'>
[<Address u'ujang@example.com'>]
<Person u'otong'>
```

```
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
```

Example #4

- **Page.tags** : a list of tags once loaded
- **Tag.pages** : a dynamic backref
- Declaring Models — Flask-SQLAlchemy Documentation
- Many to Many Relationships with Flask-SQLALchemy
- Basic Relationship Patterns — SQLAlchemy

```
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.i
)

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=
    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
```

```
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
```

Example #4

```
[<Page u'pagepython 1'>, <Page u'pagepython 2'>]
[<Tag u'tutorial'>, <Tag u'python'>]
```

Example #5

```
from datetime import datetime
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 Post(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(80))
    body = db.Column(db.Text)
    pub_date = db.Column(db.DateTime)

    category_id = db.Column(db.Integer, db.ForeignKey('cat
    category = db.relationship('Category', backref=db.back

    def __init__(self, title, body, category, pub_date=Non
        self.title = title
        self.body = body
        if pub_date is None:
            pub_date = datetime.utcnow()
        self.pub_date = pub_date
        self.category = category

    def __repr__(self):
        return '<Post %r>' % self.title
```

```
class Category(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(50))

    def __init__(self, name):
        self.name = name

    def __repr__(self):
        return '<Category %r>' % self.name
#---------------------------------------
db.create_all()
py = Category('Python')
p = Post('Hello Python!', 'Python is pretty cool', py)
db.session.add(py)
db.session.add(p)
db.session.commit() #journal?

print py.posts
print py.posts.all()
```

```
SELECT post.id AS post_id, post.title AS post_title, post.

[<Post u'Hello Python!'>]
```

Declarative

Example #6

database.py

```
from sqlalchemy import create_engine
from sqlalchemy.orm import scoped_session, sessionmaker
from sqlalchemy.ext.declarative import declarative_base
#----------------------------------------
engine = create_engine('sqlite:///test.db', convert_unicod
db_session = scoped_session(sessionmaker(autocommit=False,
                                         autoflush=False,
                                         bind=engine))
Base = declarative_base()
Base.query = db_session.query_property()
#----------------------------------------
def init_db():
    import models
    Base.metadata.create_all(bind=engine)
```

models.py

```
from sqlalchemy import Column, Integer, String
from database import Base
#----------------------------------------
class User(Base):
    __tablename__ = 'users'
    id = Column(Integer, primary_key=True)
    name = Column(String(50), unique=True)
    email = Column(String(120), unique=True)

    def __init__(self, name=None, email=None):
        self.name = name
        self.email = email

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

Example #6

```
[<User u'admin'>]
<User u'admin'>
```

app.py

```
from database import db_session, init_db
from models import User
from flask import Flask
#---------------------------------------
app = Flask(__name__)
#---------------------------------------
@app.teardown_appcontext
def shutdown_session(exception=None):
    db_session.remove()
#---------------------------------------

init_db()

u = User('admin', 'admin@localhost')
db_session.add(u)
db_session.commit()

print User.query.all()
print User.query.filter(User.name == 'admin').first()
```

Manual Object Relational Mapping

Example #7

database.py

```
from sqlalchemy import create_engine
from sqlalchemy.orm import scoped_session, sessionmaker
#from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy import MetaData
#----------------------------------------
engine = create_engine('sqlite:///test.db', convert_unicod
db_session = scoped_session(sessionmaker(autocommit=False,
                                         autoflush=False,
                                         bind=engine))
metadata = MetaData()
#Base = declarative_base()
#Base.query = db_session.query_property()
#----------------------------------------
def init_db():
    #import models
    #Base.metadata.create_all(bind=engine)
    metadata.create_all(bind=engine)
```

models.py

```
from sqlalchemy import Column, Integer, String
#from database import Base
from sqlalchemy import Table
from sqlalchemy.orm import mapper
from database import metadata, db_session
#----------------------------------------
#class User(Base):
class User(object):
    #__tablename__ = 'users'
    #id = Column(Integer, primary_key=True)
    #name = Column(String(50), unique=True)
    #email = Column(String(120), unique=True)
    query = db_session.query_property()

    def __init__(self, name=None, email=None):
        self.name = name
        self.email = email

    def __repr__(self):
        return '<User %r>' % (self.name)
#----------------------------------------
users = Table('users', metadata,
    Column('id', Integer, primary_key=True),
    Column('name', String(50), unique=True),
    Column('email', String(120), unique=True)
)
mapper(User, users)
```

SQL Abstraction Layer

```
from sqlalchemy import create_engine, MetaData
from sqlalchemy import Table, Column, Integer, String

engine = create_engine('sqlite:///test.db', convert_unicod
metadata = MetaData(bind=engine)

users = Table('users', metadata,
    Column('id', Integer, primary_key=True),
    Column('name', String(50), unique=True),
    Column('email', String(120), unique=True)
)
metadata.create_all(bind=engine)
#users = Table('users', metadata, autoload=True)
#if previously exists

con = engine.connect()
con.execute(users.insert(), name='admin', email='admin@loc
# SQLAlchemy will automatically commit for us.

print users.select(users.c.id == 1).execute().first()
r = users.select(users.c.id == 1).execute().first()
print  r['name']
print engine.execute('select * from users where id = :1',
```

Example #8

```
(1, u'admin', u'admin@localhost')
admin
(1, u'admin', u'admin@localhost')
```

References

- Flask-SQLAlchemy Documentation
- SQLAlchemy Documentation
- Patterns for Flask - Flask Documentation
- Patterns - SQLAlchemy in Flask

END

Eueung Mulyana

http://eueung.github.io/python/flask-sqlalchemy

Python CodeLabs | Attribution-ShareAlike CC BY-SA