

Python Object-Oriented Program with Libraries

Unit 6: Flask Web App Development

CHAPTER 4: LINKED WITH DATABASE

DR. ERIC CHOU

IEEE SENIOR MEMBER



Objectives

- Connect your web-page with the data base on server.

Overview

LECTURE 1



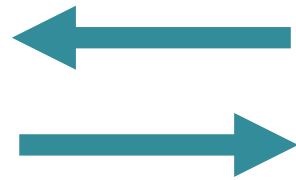
Overview of SQLAlchemy

- **SQLAlchemy** is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL.
- **SQLAlchemy** provides a full suite of well known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.

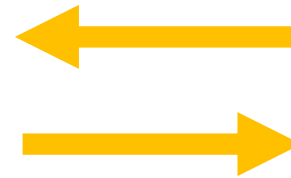


Database

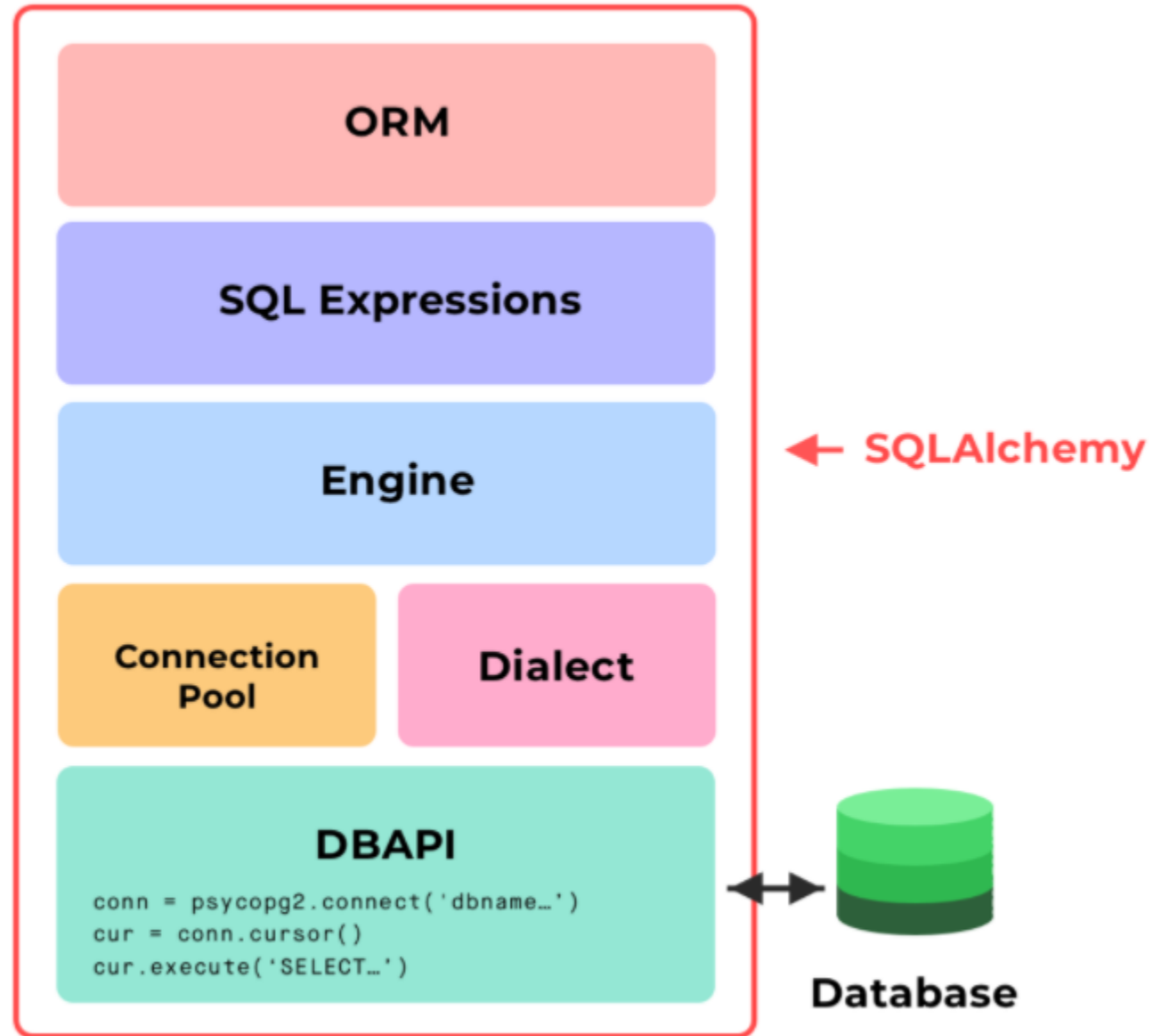
SQL
Statements



Python
Objects



Python Code





Features

- An industrial strength ORM, built from the core on the **identity map**, **unit of work**, and data **mapper patterns**. These patterns allow transparent persistence of objects using a declarative configuration system. Domain models can be constructed and manipulated naturally, and changes are synchronized with the current transaction automatically.



Features

- A relationally-oriented query system, exposing the full range of SQL's capabilities explicitly, including **joins**, **subqueries**, **correlation**, and most everything else, in terms of the object model. Writing queries with the ORM uses the same techniques of relational composition you use when writing **SQL**. While you can drop into literal SQL at any time, it's virtually never needed.



Features

- A comprehensive and flexible system of eager loading for related collections and objects. Collections are cached within a session, and can be loaded on individual access, all at once using joins, or by query per collection across the full result set.







Features

- A Core **SQL** construction system and **DBAPI** interaction layer. The **SQLAlchemy Core** is separate from the **ORM** and is a full database abstraction layer in its own right, and includes an extensible **Python-based SQL** expression language, schema metadata, connection pooling, type coercion, and custom types.



Features

- All primary and foreign key constraints are assumed to be composite and natural. Surrogate integer primary keys are of course still the norm, but **SQLAlchemy** never assumes or hardcodes to this model.
- Database introspection and generation. Database schemas can be “reflected” in one step into Python structures representing database metadata; those same structures can then generate `CREATE` statements right back out - all within the **Core**, independent of the **ORM**.

web framework	None	Flask	Flask	Bottle
Database interaction	SQLAlchemy Core	SQLAlchemy Core	SQLAlchemy Core + ORM	SQLAlchemy Core + ORM
database connector	(built into Python stdlib)	psycopg	psycopg	psycopg
relational database	 SQLite	 PostgreSQL	 PostgreSQL	 PostgreSQL

Creation of a Table in the Database

LECTURE 2



Install SQLAlchemy

- `pip install SQLAlchemy`
- `pip install flask_sqlalchemy`



Connect SQLite Database to Our Python Program

- Setup up the rules for Database and connect to database to the program

```
app.config['SQLALCHEMY_DATABASE_URI']='sqlite:///task.db'
```

- Define the record entry
- Create the table
- Commit the change

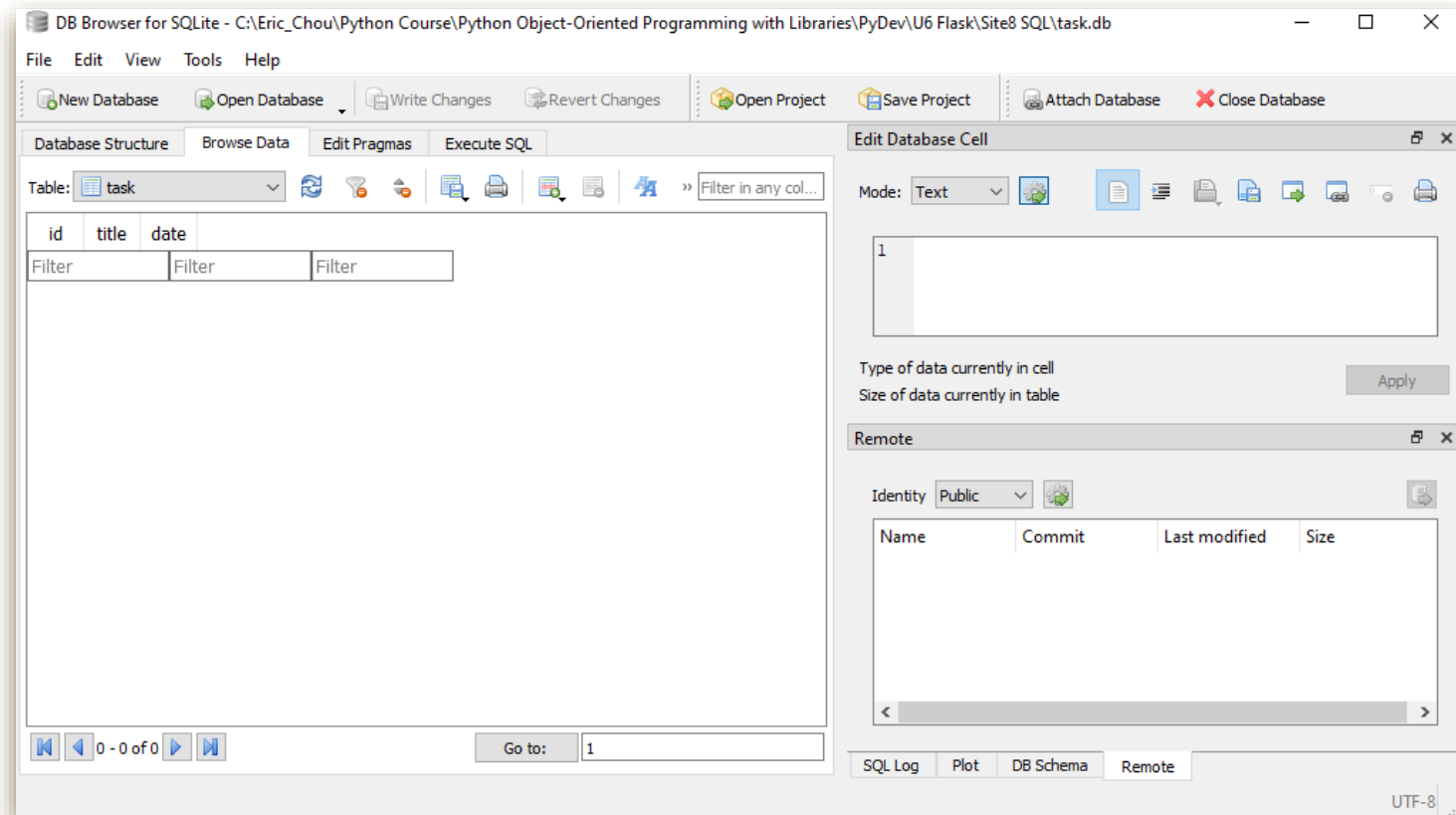
```
from flask import Flask, render_template
from flask_sqlalchemy import SQLAlchemy
app = Flask(__name__)    # create Flask as app
app.config['SECRET_KEY'] = 'fdahdfahlfa'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///task.db'

# Create Database
db = SQLAlchemy(app)
class Task(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(100), nullable=False)
    date = db.Column(db.Date, nullable=False)

    def __repr__(self):
        return f'{self.title} created on {self.date}'

db.create_all()
db.session.commit()

from routes import *    # same effects as the @app.route in app7.py
if __name__ == "__main__":
    app.run(debug=True)
```

Use DB Browser to check your database

Using and Updating the Database

LECTURE 3



DB Operations

```
C:> python
>>> from app11 import db, Task
>>> from datetime import datetime
>>> t = Task(title="xyz", date=datetime.utcnow())
>>> db.session.add(t)
>>> db.session.commit()
>>> t = Task(title="abd", date=datetime.utcnow())
>>> db.session.add(t)
>>> db.session.commit()
```

DB Browser for SQLite - C:\Eric_Chou\Python Course\Python Object-Oriented Programming with Libraries\PyDev\U6 Flask\Site8 SQL\task.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

Table: task

	id	title	date
	Filter	Filter	Filter
1	1	xyz	2021-06-26
2	2	abc	2021-06-26

1 - 2 of 2

Go to: 1

Edit Database Cell

Mode: Text

1

Type of data currently in cell: Text / Numeric
1 character(s)

Apply

Remote

Identity Public

Name	Commit	Last modified	Size
------	--------	---------------	------

SQL Log Plot DB Schema Remote

UTF-8

Display Data on the Index Page

LECTURE 4



Add Task when Form Submitted

Perform two tasks:

- Add the title and datetime of an object to the data base data.db
- Show the transaction record on the index page.

```
from flask import render_template, redirect, url_for
from app12 import app, db, Task
from datetime import datetime
import forms

@app.route('/')
@app.route('/index')
def index():
    tasks = Task.query.all()
    return render_template('index.html', tasks=tasks)

@app.route('/about', methods=['GET', 'POST'])
def about():
    form = forms.AddTaskForm()
    if form.validate_on_submit():
        t = Task(title=form.title.data, date=datetime.utcnow())
        db.session.add(t)
        db.session.commit()
        return redirect(url_for('index'))
    return render_template('about.html', form=form)
```

About this web-site

A multiple-page site

- [Home Page](#)
 - [About Page](#)
-

Some Content on the about Page

Title

About this web-site

A multiple-page site

- [Home Page](#)
 - [About Page](#)
-

Tasks:

- Eric Chou created on 2021-06-26
- Tommy Au created on 2021-06-26

Update the href on the index and about pages

LECTURE 5



Update the Base Page with url_for()

Site9/templates/base.html

```
<h1>About this web-site</h1>
<h3>A multiple-page site</h3>
<hr>
<ul>
  <li><a href="{{ url_for('index') }}">Tasks</a></li>
  <li><a href="{{ url_for('about') }}">Add New</a></li>
</ul>
<hr>
{% block main %}
{% endblock %}
```

About this web-site

A multiple-page site

- [Tasks](#)
 - [Add New](#)
-

Tasks:

- Eric Chou created on 2021-06-26
- Tommy Au created on 2021-06-26
- Nancy Brown created on 2021-06-26
- Ted Easy created on 2021-06-26
- Joyce Hong created on 2021-06-26