NOTES

Bachelor of Science (Second Year)
CSS104- Web Application Development Using Flask
Title of the Unit : SQLAlchemy
Module Name : SQL Alchemy a ORM, Installing SQL-Alchemy, Basic Database Operations using SQL Alchemy
Module Number : 22

Flask – SQLAlchemy

Introduction:

- Using raw SQL in Flask to perform operations on database can be tedious.
- **SQLAlchemy**, a Python toolkit is a powerful **OR Mapper** that gives developers the full power and flexibility of SQL.
- Flask-SQLAlchemy is the Flask extension that adds support for SQLAlchemy to your Flask application.

What is ORM (Object Relation Mapping)?

- Most programming language platforms are object oriented.
- Data in RDBMS servers on the other hand is stored as tables(structured form).
- ORM is a technique of mapping object parameters to the underlying RDBMS table structure.
- An ORM API provides methods to perform CRUD operations without having to write raw SQL statements.

Building a small web application using ORM techniques of Flask-SQLAlchemy.

Step 1 − Install Flask-SQLAlchemy extension.

pip install flask-sqlalchemy

Complete code of application **(app.py)**.

```
from flask import Flask, request, flash, url_for, redirect,
render_template
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] =
'sqlite:///students.sqlite3'
app.config['SECRET_KEY'] = "random string"

db = SQLAlchemy(app)

class students(db.Model):
   id = db.Column('student_id', db.Integer, primary_key = True)
   name = db.Column(db.String(100))
   city = db.Column(db.String(50))
   addr = db.Column(db.String(200))
   pin = db.Column(db.String(10))

   def __init__(self, name, city, addr,pin):
       self.name = name
```

```
        self.city = city
        self.addr = addr
        self.pin = pin

@app.route('/')
def show_all():
   return render_template('show_all.html', students =
students.query.all() )

@app.route('/new', methods = ['GET', 'POST'])
def new():
   if request.method == 'POST':
      if not request.form['name'] or not request.form['city'] or
not request.form['addr']:
         flash('Please enter all the fields', 'error')
      else:
         student = students(request.form['name'],
request.form['city'],
            request.form['addr'], request.form['pin'])

         db.session.add(student)
         db.session.commit()
         flash('Record was successfully added')
         return redirect(url_for('show_all'))
   return render_template('new.html')

if __name__ == '__main__':
   db.create_all()
   app.run(debug = True)
```

- You need to import SQLAlchemy class from this module.
  from flask_sqlalchemy import SQLAlchemy
- Now create a Flask application object and set URI for the database to be used.
  app = Flask(__name__)
  app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///students.sqlite3'
- Then create an object of SQLAlchemy class with application object as the parameter.
  db = SQLAlchemy(app)
- This object contains helper functions for ORM operations.
  It also provides a parent Model class using which user defined models are declared.

- To create and use database mentioned in URI, run the create_all() method.
  db.create_all()
- The Session object of SQLAlchemy manages all persistence operations of ORM object.
- The following session methods perform CRUD operations −
- **db.session.add**(model object) − inserts a record into mapped table
- **db.session.delete**(model object) − deletes record from table
- **model.query.all()** − retrieves all records from table (corresponding to SELECT query).
- The entry point of the application is show_all() function bound to ‘/’ URL.

- The Record set of students table is sent as parameter to the HTML template.
- The Server side code in the template renders the records in HTML table form.