TRƯỜNG ĐẠI HỌC QUỐC GIA HÀ NỘI

CÔNG TY RIKKEISOFT

TỰ HỌC LẬP TRÌNH WEBSITE

ĐỀ TÀI

LẬP TRÌNH FASTAPI BẰNG PYTHON (SỬ DỤNG FRAMEWORK FASTAPI)

Tác giả: TẠ QUANG TÙNG
ID code: TTS_5966
Division: D6 - G2

Hà Nội, 2024

Mục lục

1 Bài 1: Cài đặt vào API đơn giản

1.1 Cách cài đặt CLI:

- Tải framework: pip install fastapi
- Cài đặt ASGI server: pip install server

1.2 Tạo API trên trình duyệt:

- Chạy app –> *uvicorn main:app –reload*
- Mở trên trình duyệt –> *http://127.0.0.1:8000/docs*
- Chạy trên postman –> *http://127.0.0.1:8000*

1.3 Câu lệnh cơ bản:

Python Code

```python
from fastapi import FastAPI

app = FastAPI()

@app.get("/path/.../{paramID}")
async def root(param: type):
    return ...
```

- Có thể thay "get()" thành các chức năng khác như: "post() - thêm", "put() - sửa", "delete() - xóa", ...

1.4 Khai báo biến và kiểu dữ liệu:

```
nameVariable: <type>
nameVariable: List[Type]
```

2 Bài 2: Xây dựng đối tượng

2.1 Tạo đối tượng:

Python Code

```
from pydantic import BaseModel
from fastapi import FastAPI

class NameModel(BaseModel):
    id: int
    name1: str
    name2: type
    ...

    def __str__(self) -> str:
        return something

app = FastAPI()

@app.get("/path/...")
async def root(param1: type, param2: NameModel):
    return ...
```

- Trích dẫn thuộc tính –> object.param
- Gán phương thức –> nameFunction = function()

2.2 Chức năng đối tượng trong Postman:

Python Code

```
from fastapi import FastAPI, Form

app = FastAPI()

@app.get("/path/...")
async def root(param: type = Form()):
    return ...
```

- Cho phép nhập dữ liệu vào CSDL như post(), put(), delete(), ...
- Trong Postman:
  - Param: tham số truyền vào trong function(x1, x2, ...)
  - Body (form-data): trường thông tin nếu là Form()

3 Bài 3: Kết nối cơ sở dữ liệu với SQLite

3.1 Import một vài thư viện cần có:

Python Code

```
from fastapi import FastAPI
from sqlalchemy import create_engine, Column
from sqlalchemy.ext.declarative import
   declarative_base
from sqlalchemy.orm import sessionmaker
```

3.2 Tạo file kết nối CSDL:

Python Code

```
DATABASE_URL = "sqlite:///./test.db"
engine = create_engine(DATABASE_URL)
SessionLocal = sessionmaker(autocommit=False,
   autoflush=False, bind=engine)
Base = declarative_base()
```

3.3 Xây dựng Model Object:

Python Code

```
class NameObject(Base):
    __tablename__ = 'nameTable'
    id = Column(Integer, primary_key = True, index
        = True)
    name1 = Column(String, index = True)
    name2 = Column(Type)
    ...

Base.metadata.create_all(bind=engine)
```

3.4 Kết nối tới CSDL:

Python Code

```
def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()
```

3.5 Truy vấn với CSDL:

3.5.1 Chức năng thêm:

Python Code

```
from fastapi import FastAPI, Form, HTTPException,
   Depends
from sqlalchemy.orm import Session
from namefile import NameObject, get_db

app = FastAPI()

# Add function
@app.post('/path1/path2/...')
async def functionAdd(id: str = Form(), name1:
   type = Form(), ..., db: Session = Depends(
   get_db)):
    newElement = Object(...)
    db.add(newElement)
    db.commit()
    db.refresh(newElement)
    return ...
```

3.5.2 Chức năng sắp xếp:

Python Code

```
from fastapi import FastAPI, Form, HTTPException,
   Depends
from sqlalchemy.orm import Session
from namefile import NameObject, get_db

app = FastAPI()

# Sort function
@app.get('/path1/path2/...)
async def functionSort(choice: str, ..., db:
   Session = Depends(get_db)):
    a = db.query(NameObject).order_by(name1).all()
    ...
    return ...
```

3.5.3 Chức năng tìm kiếm:

Python Code

```
from fastapi import FastAPI, Form, HTTPException,
   Depends
from sqlalchemy.orm import Session
from namefile import NameObject, get_db

app = FastAPI()

# Search function
@app.get('/path1/path2/...')
async def functionSearch(searching: str, db:
   Session = Depends(get_db)):
    b = db.query(NameObject).filter(
        (NameObject.name1.contains(searching)) |
        (NameObject.name2.contains(searching)) |
        ...).all()
    return ...
```

3.5.4 Chức năng sửa:

Python Code

```
from fastapi import FastAPI, Form, HTTPException,
   Depends
from sqlalchemy.orm import Session
from namefile import NameObject, get_db

app = FastAPI()

# Edit function
@app.put('/path1/path2/...')
async def functionEdit(edit: str = Form(), db:
   Session = Depends(get_db)):
    b = db.query(NameObject).filter(NameObject.
       name1 = name1).first()
    b.name1 = newName1
    b.name2 = newName2
    ...
    db.commit()
    db.refresh(b)
    return ...
```

3.5.5 Chức năng xóa:

Python Code

```
from fastapi import FastAPI, Form, HTTPException,
   Depends
from sqlalchemy.orm import Session
from namefile import NameObject, get_db

app = FastAPI()

# Edit function
@app.delete('/path1/path2/...')
async def functionEdit(del: str = Form(), db:
   Session = Depends(get_db)):
    b = db.query(NameObject).filter(NameObject.
       name1 = name1).first()
    ...
    db.delete(b)
    db.commit()
    return ...
```

3.6 Xây dựng thêm cột dữ liệu:

Nếu gặp tình huống đã xây dựng xong model nhưng mà muốn bổ sung thêm cột, hay thuộc tính khác thì ta sẽ sử dụng câu lệnh "ADD COLUMN" trong SQL:

```
ALTER TABLE <nameTable> ADD COLUMN <newColumn>
   BOOLEAN DEFAULT FALSE;
```

VD: (thêm cột delete_flag vào bảng users)

```
ALTER TABLE users ADD COLUMN delete_flag BOOLEAN
   DEFAULT FALSE
```

4 Bài 4: Tạo khóa chính, khóa ngoại

4.1 Tạo khóa chính:

Python Code

```
class NameObject1(Base):
    __tablename__ = 'nameTable1'
    linkingValue = Column(Type, primary = key,
       index = True)
    name = Column(Type(size), index = True)
    ...
```

4.2 Tạo khóa ngoại:

Python Code

```
class NameObject2(Base):
    __tablename__ = 'nameTable2'
    linkedValue = Column(Type, ForeignKey('
       nameTable1.linkingValue'), nullable = False
       )
    name = Column(Type(size), index = True)
    ...
```

4.3 Tạo liên kết giữa các bảng:

4.3.1 file/database.py

Python Code

```
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import
   declarative_base
from sqlalchemy.orm import sessionmaker

DATABASE_URL = "sqlite:///./database.db"
engine = create_engine(DATABASE_URL)
SessionLocal = sessionmaker(autocommit = False,
   autoflush = False, bind = engine)
Base = declarative_base()
```

4.3.2 file/models.py

Python Code

```
from sqlalchemy import Column, ForeignKey
from database import Base, engine, SessionLocal
from sqlalchemy.orm import relationship

class NameObject1(Base):
    __tablename__ = 'nameTable1'
    linkingValue = Column(Type, primary = key,
       index = True)
    name = Column(Type(size), index = True)
    ...

    object2 = relationship("NameObject2",
       back_populates = "nameTable1"

class NameObject2(Base):
    __tablename__ = 'nameTable2'
    linkedValue = Column(Type, ForeignKey('
       nameTable1.linkingValue'), nullable = False
       )
    name = Column(Type(size), index = True)
    ...

    object1 = relationship("NameObject1",
       back_populates = "nameTable2"

Base.metadata.create_all(bind = engine)

# Connect to Database
def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()
```

4.3.3 file/main.py

Python Code

```
from fastapi import FastAPI, Depends
from sqlalchemy.orm import Session
import models

app = FastAPI()
...
```

5 Bài 5: Mã hóa với Token

5.1 Mã hóa mật khẩu:

Python Code

```
from passlib.context import CryptContext

pwd_context = CryptContext(schemes=["bcrypt"],
   deprecated="auto")

# Compare password
def verify_password(passwordINP, passwordCSDL):
    return pwd_context.verify(passwordINP,
       passwordCSDL)

# HashCode password
def get_password_hash(password):
    return pwd_context.hash(password)
```

5.2 Sử dụng Token:

Python Code

```
import jwt

# encode
encoded_jwt = jwt.encode({"some": "payload"}, "
   secret", algorithm="HS256")
print(encoded_jwt)
>>> eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJz

# decode
jwt.decode(encoded_jwt, "secret", algorithms=["
   HS256"])
>>> {'some': 'payload'}
```

6 Bài 6: Phân trang và giới hạn

6.1 Câu lệnh:

Python Code

```
@app.get("/link/...")
assync def function(limit: int=Query(5,gt=0),
   offset: int=Query(0, ge=0)):
     val = db.query(models.A).offset(offset).limit(
        limit).all()
     return val
```

6.2 Chú thích:

- limit: int = Query(5, gt = 0)
  => Số lượng tối đa mỗi trang (mặc định là 5)
- offset: int = Query(0, ge = 0)
  => Vị trí bắt đầu của trang (mặc định là 0)

7 Bài 7: Kết nối cơ sở dữ liệu với MySQL

7.1 Tải thư viện:

- Terminal: *pip install sqlalchemy mysql-connector-python*
- Công cụ: MySQL và MySQL Workbench

7.2 Minh họa:

database.py

```
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import
   declarative_base
from sqlalchemy.orm import sessionmaker

# Connect to database by MySQL
DATABASE\_URL = "mysql+pymysql://<username>:<
   password>@localhost/<database>"

engine = create_engine(DATABASE_URL)
SessionLocal = sessionmaker(autocommit=False,
   autoflush=False, bind=engine)
Base = declarative_base()
```

8 Bài 8: Template HTML cho FastAPI

8.1 Thư viện giao diện:

main.py

```
from fastapi.templating import Jinja2Temaplates
from fastapi.responses import HTMLResponse,
   RedirectResponse
from fastapi.staticfiles import StaticFiles
from fastapi import FastAPI, Request

app = FastAPI()
templates = Jinja2Templates(directory="templates")
app.mount("/static", StaticFiles(directory="static
   "), name="static")
```

- StaticFiles dùng để lưu trữ cố định các file css, fonts, images, templates, ...
- Request để thực hiện điều hướng theo yêu cầu: GET, POST (thường HTML chỉ hỗ trợ 2 phương thức này)

8.2 Gọi API vào trong Template HTML:

main.py

```
@app.get("/link", response_class-HTMLResponse)
assync def function(request: Request):
    return templates.TemplateResponse("file.html",
        {"request": request, ...})
```

- Có thể thay đổi API theo các phương thức GET, POST, PUT, DELETE, ...
- Xây dựng HTML tương tự như trong Django, Flask, ...

9 Bài 9: Lưu trữ Token vào bộ nhớ Cookies

9.1 Lưu trữ token:

main.py

```
@app.post("/save_token")
assync def function(request: Request):
    response = RedirectResponse(url="/",
       status_code=303)
    response.set_cookie(key="token", value=
       encoded_jwt)
    return response
```

9.2 Gọi và trả về token:

main.py

```
token = request.cookies.get("token")
```

9.3 Xóa dữ liệu token:

main.py

```
@app.get("/delete_token")
assync def function(request: Request):
    response = RedirectResponse(url="/",
       status_code=303)
    response.delete_cookie(key="token", value=
       encoded_jwt)
    return response
```

9.4 Kiểm tra token:

main.py

```
from fastapi import Cookie

@app.get("/check_token")
assync def function(token: str = Cookie(None)):
    if token:
        return true
```

10 Bài 10: Phân cấp thư mục

10.1 Thư mục nhánh con:

- Tạo thư mục nhánh con có tên "/routers" chứa các file.py các API theo đối tượng và chức năng
- Xuất thư viện –> from fastapi import APIRouter

routers/file.py

```
from fastapi import RouterAPI

router = APIRouter()

@router.get("/check_token")
assync def function(token: str = Cookie(None)):
    if token:
        return true
```

10.2 Thư mục chính:

- File chạy chính là main.py

main.py

```
from fastapi import FastAPI
from routers import filename

app = FastAPI()

app.include_router(filename.router)

@app.get("/check_token")
assync def function(token: str = Cookie(None)):
    if token:
        return true
```