HƯỚNG DẪN DỰ ÁN PYTHON: CRUD VỚI MONGODB TRÊN CONSOLE

Tác giả: Đặng Kim Thi

Mục đích: Hướng dẫn xây dựng một ứng dụng console Python đơn giản để thực hiện các thao tác CRUD (Create, Read, Update, Delete) với cơ sở dữ liệu MongoDB.

MỤC TIÊU BÀI HỌC

1. **Hiểu cơ bản về MongoDB**: Làm quen với cách kết nối và thao tác với cơ sở dữ liệu NoSQL MongoDB trong Python.
2. **Sử dụng thư viện PyMongo**: Học cách sử dụng PyMongo để kết nối và thực hiện các lệnh CRUD.
3. **Xây dựng ứng dụng console**: Tạo một giao diện dòng lệnh đơn giản cho phép người dùng tương tác với cơ sở dữ liệu.
4. **Thực hành lập trình Python**: Áp dụng các khái niệm như lớp (class), vòng lặp, điều kiện và xử lý ngoại lệ trong Python.

YÊU CẦU TRƯỚC KHI BẮT ĐẦU

1. **Cài đặt MongoDB**:
   - Tải và cài đặt MongoDB từ trang chính thức: mongodb.com.
   - Chạy MongoDB server trên localhost (port mặc định 27017) bằng lệnh `mongod` trong terminal.
2. **Cài đặt Python**:
   - Đảm bảo Python đã được cài đặt (khuyến nghị phiên bản 3.7 trở lên).
3. **Cài đặt PyMongo**:
   - Mở terminal và chạy lệnh:

     ```
     pip install pymongo
     ```

   - PyMongo là thư viện Python dùng để kết nối và thao tác với MongoDB.

CODE DỰ ÁN

Dưới đây là code hoàn chỉnh, sau đó tôi sẽ giải thích từng phần chi tiết.

```
from pymongo import MongoClient
from pprint import pprint
```

```
import sys

class MongoCRUD:
    def __init__(self):
        # Kết nối đến MongoDB
        self.client = MongoClient('mongodb://localhost:27017/')
        self.db = self.client['mydatabase']
        self.collection = self.db['users']

    def create(self):
        print("\n=== Thêm người dùng mới ===")
        name = input("Nhập tên: ")
        age = int(input("Nhập tuổi: "))
        email = input("Nhập email: ")

        user = {
            "name": name,
            "age": age,
            "email": email
        }

        result = self.collection.insert_one(user)
        print(f"Đã thêm người dùng với ID: {result.inserted_id}")

    def read(self):
        print("\n=== Danh sách người dùng ===")
        users = self.collection.find()
        for user in users:
            pprint(user)

    def update(self):
        print("\n=== Cập nhật người dùng ===")
        email = input("Nhập email của người dùng cần cập nhật: ")
        print("Nhập thông tin mới (để trống nếu không muốn thay đổi):")
        new_name = input("Tên mới: ")
        new_age = input("Tuổi mới: ")

        update_data = {}
        if new_name:
            update_data["name"] = new_name
        if new_age:
            update_data["age"] = int(new_age)

        if update_data:
            result = self.collection.update_one(
                {"email": email},
                {"$set": update_data}
            )
            if result.modified_count > 0:
                print("Cập nhật thành công!")
            else:
                print("Không tìm thấy người dùng hoặc không có thay đổi.")

    def delete(self):
```

```
        print("\n=== Xóa người dùng ===")
        email = input("Nhập email của người dùng cần xóa: ")
        result = self.collection.delete_one({"email": email})
        if result.deleted_count > 0:
            print("Xóa thành công!")
        else:
            print("Không tìm thấy người dùng với email này.")

    def menu(self):
        while True:
            print("\n=== QUẢN LÝ NGƯỜI DÙNG ===")
            print("1. Thêm người dùng (Create)")
            print("2. Xem danh sách (Read)")
            print("3. Cập nhật người dùng (Update)")
            print("4. Xóa người dùng (Delete)")
            print("5. Thoát")

            choice = input("Chọn chức năng (1-5): ")

            if choice == '1':
                self.create()
            elif choice == '2':
                self.read()
            elif choice == '3':
                self.update()
            elif choice == '4':
                self.delete()
            elif choice == '5':
                print("Tạm biệt!")
                self.client.close()
                sys.exit(0)
            else:
                print("Lựa chọn không hợp lệ!")

def main():
    try:
        crud = MongoCRUD()
        crud.menu()
    except Exception as e:
        print(f"Có lỗi xảy ra: {e}")

if __name__ == "__main__":
    main()
```

GIẢI THÍCH CHI TIẾT TỪNG LỆNH

1. Import thư viện

```
from pymongo import MongoClient
from pprint import pprint
```

```
import sys
```

- `from pymongo import MongoClient`: Nhập lớp `MongoClient` từ thư viện PyMongo để kết nối với MongoDB.
- `from pprint import pprint`: Nhập hàm `pprint` (pretty print) để in dữ liệu đẹp và dễ đọc hơn so với `print` thông thường.
- `import sys`: Nhập thư viện `sys` để sử dụng `sys.exit()` khi thoát chương trình.

---

2. Khai báo lớp `MongoCRUD`

```
class MongoCRUD:
    def __init__(self):
        self.client = MongoClient('mongodb://localhost:27017/')
        self.db = self.client['mydatabase']
        self.collection = self.db['users']
```

- `class MongoCRUD`: Định nghĩa một lớp để chứa tất cả các chức năng CRUD.
- `def __init__(self)`: Hàm khởi tạo của lớp, được gọi khi tạo đối tượng.
- `self.client = MongoClient('mongodb://localhost:27017/')`: Tạo kết nối đến MongoDB server chạy trên `localhost` với port `27017`.
- `self.db = self.client['mydatabase']`: Chọn hoặc tạo database có tên `mydatabase`.
- `self.collection = self.db['users']`: Chọn hoặc tạo collection có tên `users` trong database.

---

3. Hàm `create` - Thêm người dùng mới

```
def create(self):
    print("\n=== Thêm người dùng mới ===")
    name = input("Nhập tên: ")
    age = int(input("Nhập tuổi: "))
    email = input("Nhập email: ")

    user = {
        "name": name,
        "age": age,
        "email": email
    }

    result = self.collection.insert_one(user)
    print(f"Đã thêm người dùng với ID: {result.inserted_id}")
```

- `print("\n=== Thêm người dùng mới ===")`: In tiêu đề để người dùng biết đang thực hiện chức năng gì.
- `name = input("Nhập tên: ")`: Yêu cầu người dùng nhập tên và lưu vào biến `name`.

- `age = int(input("Nhập tuổi: "))`: Yêu cầu nhập tuổi, chuyển thành số nguyên bằng `int()`.
- `email = input("Nhập email: ")`: Yêu cầu nhập email.
- `user = {...}`: Tạo một dictionary chứa thông tin người dùng.
- `result = self.collection.insert_one(user)`: Chèn document `user` vào collection `users`. Hàm `insert_one()` trả về một đối tượng chứa thông tin về kết quả.
- `print(f"Đã thêm người dùng với ID: {result.inserted_id}")`: In ID của document vừa thêm (`inserted_id` là thuộc tính của kết quả).

---

4. Hàm `read` - Xem danh sách người dùng

```
def read(self):
    print("\n=== Danh sách người dùng ===")
    users = self.collection.find()
    for user in users:
        pprint(user)
```

- `print("\n=== Danh sách người dùng ===")`: In tiêu đề.
- `users = self.collection.find()`: Lấy tất cả document từ collection `users`. Hàm `find()` trả về một con trỏ (cursor).
- `for user in users`: Duyệt qua từng document trong con trỏ.
- `pprint(user)`: In mỗi document một cách đẹp mắt.

---

5. Hàm `update` - Cập nhật người dùng

```
def update(self):
    print("\n=== Cập nhật người dùng ===")
    email = input("Nhập email của người dùng cần cập nhật: ")
    print("Nhập thông tin mới (để trống nếu không muốn thay đổi):")
    new_name = input("Tên mới: ")
    new_age = input("Tuổi mới: ")

    update_data = {}
    if new_name:
        update_data["name"] = new_name
    if new_age:
        update_data["age"] = int(new_age)

    if update_data:
        result = self.collection.update_one(
            {"email": email},
            {"$set": update_data}
        )
        if result.modified_count > 0:
            print("Cập nhật thành công!")
        else:
            print("Không tìm thấy người dùng hoặc không có thay đổi.")
```

- `email = input(...)`: Yêu cầu nhập email để tìm người dùng cần cập nhật.
- `new_name = input(...)` và `new_age = input(...)`: Yêu cầu nhập thông tin mới (có thể bỏ trống).
- `update_data = {}`: Tạo dictionary rỗng để chứa các trường cần cập nhật.
- `if new_name:`: Nếu người dùng nhập tên mới, thêm vào `update_data`.
- `if new_age:`: Nếu nhập tuổi mới, thêm vào `update_data` (chuyển thành số nguyên).
- `if update_data:`: Kiểm tra xem có dữ liệu để cập nhật không.
- `result = self.collection.update_one(...)`: Cập nhật một document dựa trên email, sử dụng toán tử `$set` để thay đổi các trường trong `update_data`.
- `result.modified_count`: Số document đã được sửa đổi. Nếu > 0 thì cập nhật thành công.

---

6. Hàm `delete` - Xóa người dùng

```
def delete(self):
    print("\n=== Xóa người dùng ===")
    email = input("Nhập email của người dùng cần xóa: ")
    result = self.collection.delete_one({"email": email})
    if result.deleted_count > 0:
        print("Xóa thành công!")
    else:
        print("Không tìm thấy người dùng với email này.")
```

- `email = input(...)`: Yêu cầu nhập email để tìm người dùng cần xóa.
- `result = self.collection.delete_one(...)`: Xóa một document dựa trên email.
- `result.deleted_count`: Số document đã bị xóa. Nếu > 0 thì xóa thành công.

---

7. Hàm `menu` - Giao diện chính

```
def menu(self):
    while True:
        print("\n=== QUẢN LÝ NGƯỜI DÙNG ===")
        print("1. Thêm người dùng (Create)")
        print("2. Xem danh sách (Read)")
        print("3. Cập nhật người dùng (Update)")
        print("4. Xóa người dùng (Delete)")
        print("5. Thoát")

        choice = input("Chọn chức năng (1-5): ")

        if choice == '1':
            self.create()
        elif choice == '2':
            self.read()
        elif choice == '3':
            self.update()
        elif choice == '4':
```

```
            self.delete()
        elif choice == '5':
            print("Tạm biệt!")
            self.client.close()
            sys.exit(0)
        else:
            print("Lựa chọn không hợp lệ!")
```

- `while True`: Vòng lặp vô hạn để hiển thị menu liên tục.
- `print(...)`: Hiển thị các lựa chọn.
- `choice = input(...)`: Lấy lựa chọn của người dùng.
- `if choice == '1': ...`: Gọi hàm tương ứng với lựa chọn (1-5).
- `self.client.close()`: Đóng kết nối MongoDB trước khi thoát.
- `sys.exit(0)`: Thoát chương trình.

---

8. Hàm `main` - Điểm bắt đầu chương trình

```
def main():
    try:
        crud = MongoCRUD()
        crud.menu()
    except Exception as e:
        print(f"Có lỗi xảy ra: {e}")

if __name__ == "__main__":
    main()
```

- `try ... except`: Bọc code chính trong khối xử lý lỗi để bắt ngoại lệ.
- `crud = MongoCRUD()`: Tạo đối tượng của lớp `MongoCRUD`.
- `crud.menu()`: Gọi hàm menu để bắt đầu chương trình.
- `print(f"Có lỗi xảy ra: {e}")`: In thông báo nếu có lỗi.
- `if __name__ == "__main__":`: Đảm bảo `main()` chỉ chạy khi file được thực thi trực tiếp.

CÁCH CHẠY CHƯƠNG TRÌNH

1. Lưu code vào file `mongo_crud.py`.
2. Mở terminal, đảm bảo MongoDB đang chạy (`mongod`).
3. Chạy lệnh:

```
python mongo_crud.py
```

4. Chọn các chức năng từ 1-5 để thao tác.

KẾT LUẬN

Dự án này giúp bạn hiểu cách tích hợp Python với MongoDB để thực hiện các thao tác CRUD cơ bản. Bạn có thể mở rộng bằng cách thêm các trường dữ liệu khác, cải thiện giao diện, hoặc thêm xử lý lỗi chi tiết hơn.

Tác giả: Đặng Kim Thi