

FastAPI 回傳資料格式



我在寫fastapi，寫好的api要給react call，把值存到 mariadb，為什麼我看mariadb存的值是{message: "hello"}，在swagger的/doc看到的reponse是 `"/"message/":"/"hello/"`



你遇到的問題可能是 **FastAPI 回傳的資料格式**或是資料在送進 **MariaDB** 時沒有正確解析造成的。以下是針對你描述的情況做的分析與解法：

問題描述拆解：

1. **FastAPI** 的 **Swagger** `/docs` 顯示回傳是：

```
"/"message/":"/"hello/"
```

這看起來像是字串被多次序列化（例如 JSON -> string -> 再變成字串形式的 JSON）。

2. **MariaDB** 裡存的是：

```
{message: "hello"}
```

如果它是字串，代表資料被當作一整塊 JSON 字串塞進資料庫了，而不是個別欄位。

常見原因：

✅ 原因一：你把整個 JSON 當作一個字串存在資料庫

範例程式碼：

```
@app.post("/save")
async def save_data(data: dict):
    cursor.execute("INSERT INTO my_table (payload) VALUES (%s)",
        (json.dumps(data),))
```

這會把整個 `{"message": "hello"}` 存成字串放到一個欄位裡（例如 `payload`），導致資料庫看到的就是整串 JSON。

✅ 原因二：你在回傳 response 時，把 JSON 當作字串回傳

範例錯誤寫法：

```
return JSONResponse(content=json.dumps({"message": "hello"}))
```

這樣會把 JSON 物件先變成字串 `{"message": "hello"}`，然後又被 FastAPI 包一次，就變成：

```
"\"{\\\\"message\\\\"": \\\\"hello\\\\"}"\""
```

在 Swagger `/docs` 看起來就像你說的那樣亂碼。

✅ 正確寫法示範

✅ FastAPI 正確的 Response 回傳：

```

from fastapi import FastAPI
from pydantic import BaseModel
import mariadb

app = FastAPI()

class Message(BaseModel):
    message: str

# 假設你有一個資料表 `messages` 欄位為 `message` (VARCHAR)

@app.post("/save")
async def save_message(msg: Message):
    # 儲存到資料庫
    conn = mariadb.connect(user="root", password="pass", database="test")
    cursor = conn.cursor()
    cursor.execute("INSERT INTO messages (message) VALUES (?)", (msg.message,))
    conn.commit()
    conn.close()

    # 正確回傳 JSON
    return {"status": "success", "message": msg.message}

```

🔍 `/docs` 頁面看到的會是：

```

{
  "status": "success",
  "message": "hello"
}

```

✅ **React** 呼叫這個 **API** 的方式建議：

```

fetch("http://localhost:8000/save", {
  method: "POST",
  headers: {

```

```
"Content-Type": "application/json"
},
body: JSON.stringify({ message: "hello" })
})
```

✅ 資料庫內容會是：

id	message
1	hello

💡 小結論

問題原因	解法
回傳內容被重複 JSON 序列化	不要用 <code>json.dumps()</code> 再傳給 <code>JSONResponse</code>
MariaDB 儲存 JSON 字串而非欄位值	將字典中的值取出來存欄位

如果你能貼出你目前的 FastAPI 處理與寫入 MariaDB 的程式碼，我可以幫你更具體對症下藥。需要我幫你檢查嗎？



我想要理解json資料存進mariadb的用法(資料格式 LONGTEXT)，為了做之後跟react串接的實驗，我對fastapi的

寫法與mariadb的用法不熟，你能教我嗎，或是你能給更簡單的範例：



已更新儲存的記憶

當然可以！以下是一個**簡單又完整的範例**，教你如何使用 FastAPI 接收 JSON 並存進 MariaDB 的 `LONGTEXT` 欄位中，適合做與 React 串接的實驗：

資料庫準備 (MariaDB)

假設你要儲存的資料像這樣：

```
{
  "username": "佳恩",
  "age": 28,
  "hobbies": ["coding", "gaming"]
}
```

那你可以在 MariaDB 建一張表：

```
CREATE TABLE json_data (
  id INT AUTO_INCREMENT PRIMARY KEY,
  content LONGTEXT
);
```

FastAPI 範例

✅ 安裝必要套件

```
pip install fastapi uvicorn mariadb pydantic
```

✅ `main.py` 完整程式

```
from fastapi import FastAPI, Request
import mariadb
import json

app = FastAPI()

# 資料庫連線設定
def get_connection():
    return mariadb.connect(
        user="your_user",
        password="your_password",
        host="127.0.0.1",
        port=3306,
        database="your_database"
    )

# 接收 JSON 並存進資料庫
@app.post("/store")
async def store_json(request: Request):
    data = await request.json() # 取得 JSON 資料

    conn = get_connection()
    cursor = conn.cursor()

    # 存成 LONGTEXT, 使用 json.dumps() 轉成字串
    cursor.execute("INSERT INTO json_data (content) VALUES (?)",
        (json.dumps(data),))

    conn.commit()
    cursor.close()
    conn.close()
```

```
return {"status": "ok", "stored_data": data}
```

用 Swagger `/docs` 測試

啟動服務：

```
uvicorn main:app --reload
```

開啟瀏覽器到 <http://localhost:8000/docs>，你會看到 `/store` API。

你可以直接貼上 JSON：

```
{  
  "username": "佳恩",  
  "age": 28,  
  "hobbies": ["coding", "gaming"]  
}
```

送出後，這份資料會被轉為字串存進資料庫 `content` 欄位中。

資料庫裡看到的會是：

id	content
1	{"username": "佳恩", "age": 28, "hobbies": ["coding", "gaming"]}

❑ React 未來串接方式

未來你可以用 React 傳送資料這樣：

```
fetch("http://localhost:8000/store", {
  method: "POST",
  headers: {
    "Content-Type": "application/json"
  },
  body: JSON.stringify({
    username: "佳恩",
    age: 28,
    hobbies: ["coding", "gaming"]
  })
})
```

如果你想要我幫你改成特定的欄位 schema（例如不是存整包 JSON，而是 username、age 分開存），我也可以調整給你！

需要我幫你做「讀取資料」的 API 嗎？還是你現在想先把資料成功塞進 MariaDB 就好？



我要存的格式是

```
{
  "username": "佳恩",
  " age ": 28,
  " hobbies ": ["coding", "gaming"],
  " content":
  {
    message: "hello",
    student: "allen"
  }
}, username、username、hobbies、content 都是mariadb
schema的欄位
```




了解！你想要的資料結構如下：

```
{
  "username": "佳恩",
  "age": 28,
  "hobbies": ["coding", "gaming"],
  "content": {
    "message": "hello",
    "student": "allen"
  }
}
```

而你希望 MariaDB 中的欄位如下：

欄位名	資料型別	說明
`username`	`VARCHAR`	使用者名稱
`age`	`INT`	年齡
`hobbies`	`LONGTEXT`	陣列轉成 JSON 存文字
`content`	`LONGTEXT`	巢狀 JSON 物件，轉文字存入

✅ 第一步：MariaDB 建表語法

```
CREATE TABLE user_data (
  id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(100),
```

```
    age INT,  
    hobbies LONGTEXT,  
    content LONGTEXT  
);
```

✅ 第二步：FastAPI 實作

🔧 安裝套件（如果還沒裝）

```
pip install fastapi uvicorn mariadb pydantic
```

🐍 `main.py` 完整範例

```
from fastapi import FastAPI  
from pydantic import BaseModel  
from typing import List, Dict, Any  
import mariadb  
import json  
  
app = FastAPI()  
  
# 定義資料格式  
class Content(BaseModel):  
    message: str  
    student: str  
  
class UserData(BaseModel):  
    username: str  
    age: int  
    hobbies: List[str]  
    content: Content
```

```

# 連線設定
def get_connection():
    return mariadb.connect(
        user="your_user",
        password="your_password",
        host="127.0.0.1",
        port=3306,
        database="your_database"
    )

# API 接收 JSON 並存入資料庫
@app.post("/store")
async def store_user(data: UserData):
    conn = get_connection()
    cursor = conn.cursor()

    # hobbies 和 content 是複雜資料，轉成 JSON 字串儲存
    cursor.execute(
        "INSERT INTO user_data (username, age, hobbies, content) VALUES (?, ?, ?, ?)",
        (
            data.username,
            data.age,
            json.dumps(data.hobbies),
            json.dumps(data.content.dict()) # content 是 Pydantic Model, 要先轉
成 dict
        )
    )

    conn.commit()
    cursor.close()
    conn.close()

    return {"status": "ok", "stored": data}

```

✅ 第三步：用 `./docs` 測試

啟動 FastAPI：

```
uvicorn main:app --reload
```

然後開瀏覽器打開 <http://localhost:8000/docs>，測試 `/store` API。

貼入這段資料測試：

```
{
  "username": "佳恩",
  "age": 28,
  "hobbies": ["coding", "gaming"],
  "content": {
    "message": "hello",
    "student": "allen"
  }
}
```

✅ 結果：MariaDB 會儲存成這樣

id	username	age	hobbies	content
1	佳恩	28	<code>["coding", "gaming"]</code>	<code>{"message": "hello", "student": "allen"}</code>

是否需要我幫你做一個 `GET /user/{id}` 的 API，把資料從 MariaDB 讀出、還原成原本的 JSON 結構？還是你目前想先專注在寫入就好？



我要用sessionmaker不要async



了解！你想使用 **同步版（非 async）FastAPI** 且用 ``sessionmaker`` + **SQLAlchemy ORM** 存資料進 MariaDB，欄位分別為 ``username``、``age``、``hobbies``、``content``，其中 ``hobbies`` 和 ``content`` 是要用 ``LONGTEXT`` 存 JSON。

我這裡提供一個**簡單完整範例**，不使用 ``async``，使用同步的 ``FastAPI`` + ``SQLAlchemy``（``sessionmaker``），幫你打好實驗的基礎。

✓ 1. 安裝必要套件

```
pip install fastapi uvicorn sqlalchemy mariadb pydantic
```

✓ 2. 建立資料庫表（MariaDB）

```
CREATE TABLE user_data (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    username VARCHAR(100),  
    age INT,  
    hobbies LONGTEXT,  
    content LONGTEXT  
);
```

✓ 3. `main.py` 同步 + sessionmaker 實作

```
from fastapi import FastAPI, Depends
from pydantic import BaseModel
from typing import List
import json

from sqlalchemy import create_engine, Column, Integer, String, Text
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker, Session

# MariaDB 連線設定 (請改成你自己的帳密/資料庫)
DB_URL =
"mariadb+mariadbconnector://user:password@localhost:3306/your_database"

engine = create_engine(DB_URL, echo=True)
SessionLocal = sessionmaker(bind=engine)

Base = declarative_base()

app = FastAPI()

# 資料表對應的 ORM Model
class UserData(Base):
    __tablename__ = "user_data"

    id = Column(Integer, primary_key=True, index=True)
    username = Column(String(100))
    age = Column(Integer)
    hobbies = Column(Text) # LONGTEXT
    content = Column(Text) # LONGTEXT

# 如果資料表還沒建立, 這行會建立
Base.metadata.create_all(bind=engine)

# Pydantic Schema
class Content(BaseModel):
    message: str
    student: str

class UserInput(BaseModel):
```

```

username: str
age: int
hobbies: List[str]
content: Content

# Dependency: 取得 DB Session
def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()

# 寫入資料的 API
@app.post("/store")
def store_user(user: UserInput, db: Session = Depends(get_db)):
    db_user = UserData(
        username=user.username,
        age=user.age,
        hobbies=json.dumps(user.hobbies),
        content=json.dumps(user.content.dict())
    )
    db.add(db_user)
    db.commit()
    db.refresh(db_user)

    return {"status": "ok", "id": db_user.id}

```

4. 測試用 JSON (Swagger `/docs`)

打開 <http://localhost:8000/docs>, 傳入：

```

{
  "username": "佳恩",
  "age": 28,
  "hobbies": ["coding", "gaming"],

```

```
"content": {  
  "message": "hello",  
  "student": "allen"  
}  
}
```

✅ 資料庫內容長這樣：

id	username	age	hobbies	content
1	佳恩	28	`["coding", "gaming"]`	`{"message": "hello", "student": "allen"}`

需要我再補充：

- 查詢資料 (`GET /user/{id}`) ?
- 修改資料？
- 或搭配 React 使用方式？

都可以幫你擴充！