Flask Introduction

Frank Yang

Outline

- Model-View-Controller (MVC)
- Flask
- Flask-SQLAlchemy
- Flask-Migrate

為什麼需要框架？

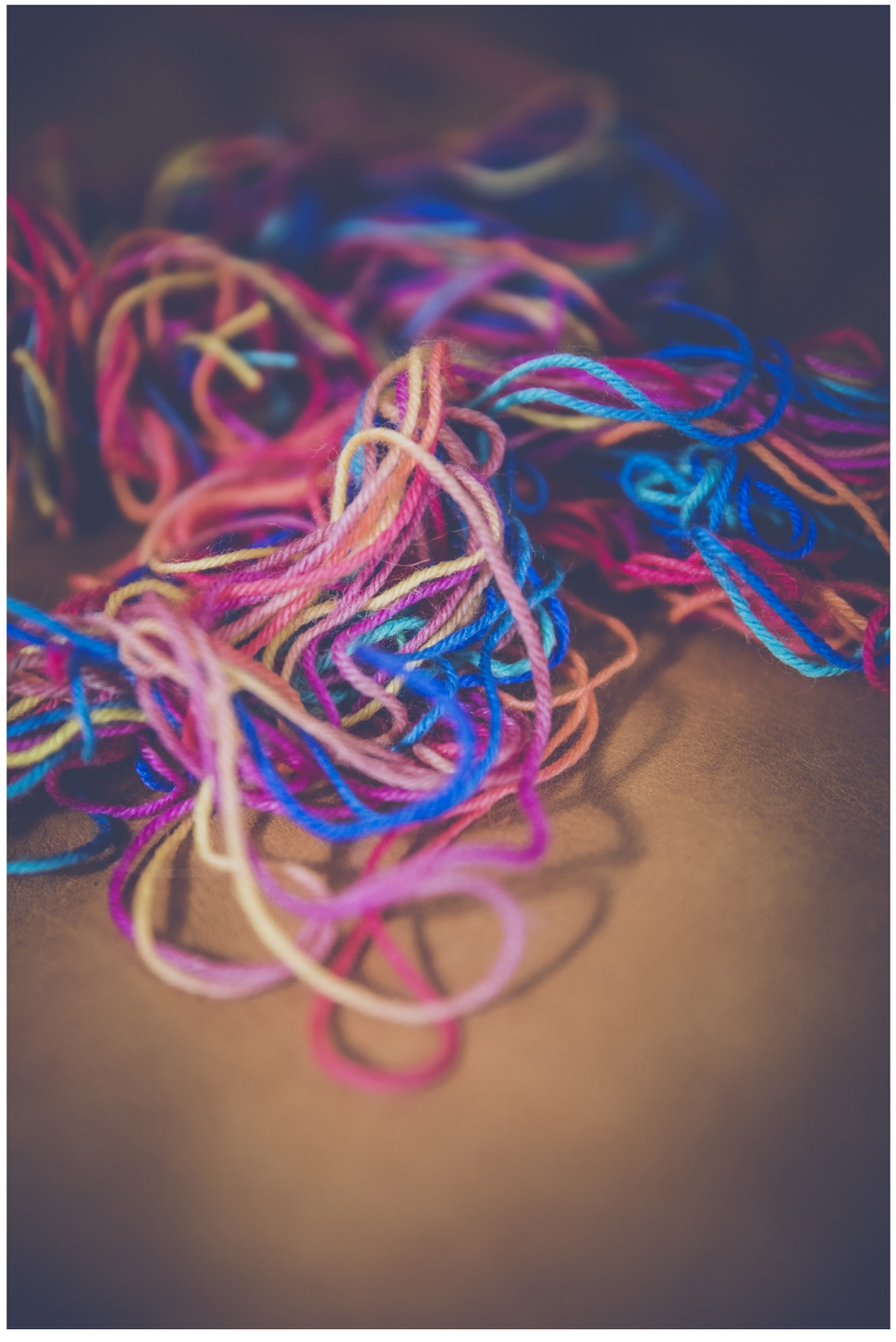
```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>X-Village</title>
    <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
</head>
<body>
    <div id="div1">
        ...
    </div>
    <script>
        $('#div1')...
    </script>
</body>
</html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>X-Village</title>
    <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
    ...
</head>
<body>
    <div id="div1">
        ...
        <div id="div2">
            ...
        </div>
    </div>
    <script>
        $('#div1')...
    </script>
    <script>
        $('#div2')...
    </script>
</body>
</html>
```

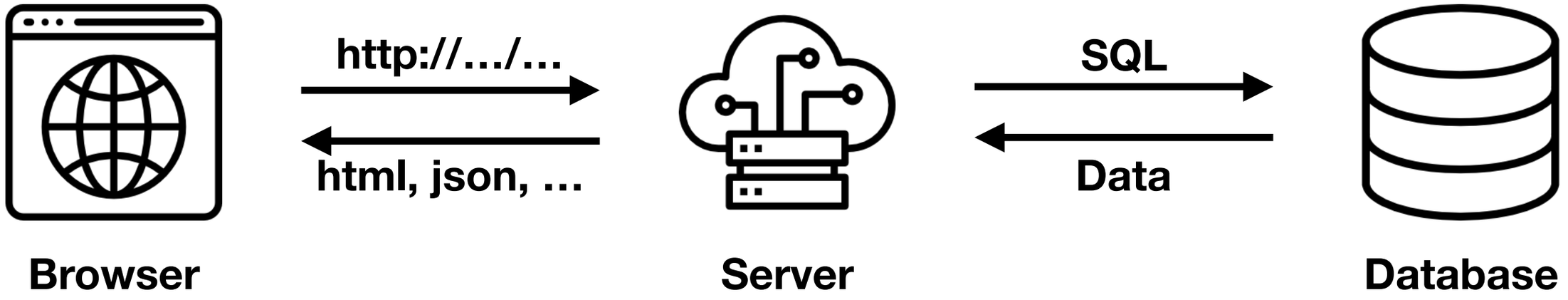
```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>X-Village</title>
    <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
</head>
<body>
    <div id="div1">
        ...
        <div id="div2">
            ...
            <div id="div3">...</div>
        </div>
    </div>
    <script>
        $('#div1')...
    </script>
    <script>
        $('#div2')...
    </script>
    <script>
        $('#div3')...
    </script>
</body>
</html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>X-Village</title>
    <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
</head>
<body>
    <div id="div1" style="color:red;">
        ...
        <div id="div2" style="color:blue;">
            ...
            <div id="div3" style="color:green;">...</div>
        </div>
    </div>
    <script>
        $('#div1')...
    </script>
    <script>
        $('#div2')...
    </script>
    <script>
        $('#div3')...
    </script>
</body>
</html>
```

沒有框架的專案像一團毛球



後端包含了哪些部分？



- 保持喚醒
- 接收 HTTP Request，判斷 URL
- 邏輯操作
- 操作 Database
- 回傳 HTML, JSON, ...

```
def server():
    while True: # 保持唤醒
```

```
def server():
    while True: # 保持喚醒
        # 接收 HTTP Request
        # ...
```

```
def server():
    while True: # 保持喚醒
        # 接收 HTTP Request
        # ...

        # 判斷 URL
        # if ...
```

```
def server():
    while True: # 保持喚醒
        # 接收 HTTP Request
        # ...

        # 判斷 URL
        # if ...

        # 操作 Database
        # SELECT ...
```

```
def server():
    while True: # 保持喚醒
        # 接收 HTTP Request
        # ...

        # 判斷 URL
        # if ...

        # 操作 Database
        # SELECT ...

        # 回傳 HTML, JSON, ...
        # return '<html>..</html>'
```

```
def server():
    while True: # 保持喚醒
        # 接收 HTTP Request
        # ...

        # 判斷 URL
        # if ...

        # 操作 Database
        # SELECT ...

        # 回傳 HTML, JSON, ...
        # return '<html>..</html>'


if __name__ == '__main__':
    server()
```

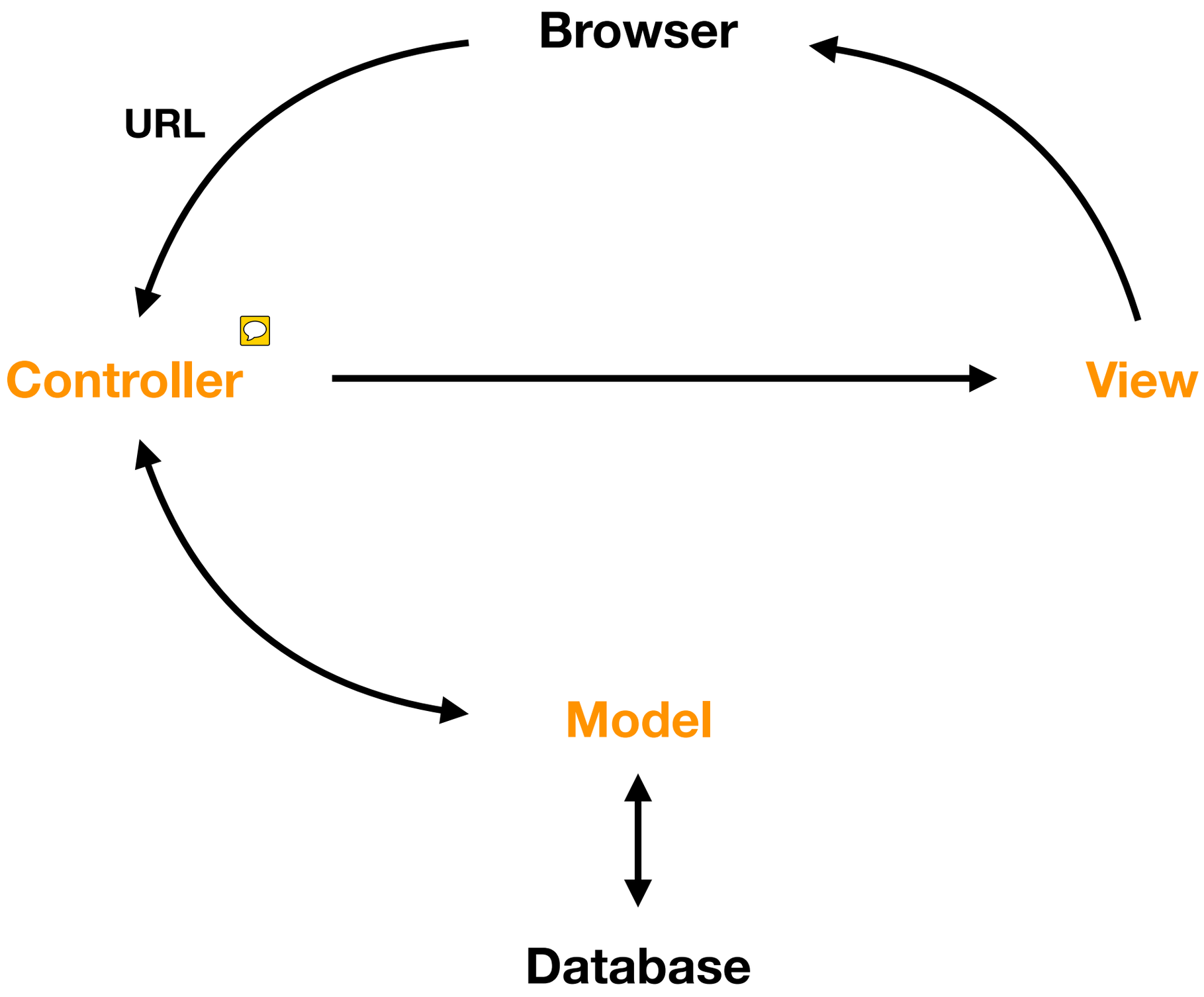
另一團毛球...

沒有架構好的程式碼

會隨著需求變化，越來越難管理

所以我們需要一個好的架構

大多數後端框架都依循 MVC


Browser
URL
Controller
View
Model
Database

MVC

- Controller
  - 接收 HTTP Request，判斷 URL
  - 邏輯操作
- Model
  - 操作 Database
- View
  - 回傳 HTML

什麼是 Flask？

Flask 是一個以 Python 為基礎的後端框架

Install Flask

```
pip install Flask
```

Flask Hello World

```python
# app.py
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"
```

```
$: flask run
* Running on http://127.0.0.1:5000/ (Press CTRL+C to
quit)
```

Exercise 1

讓網站在連結 http://127.0.0.1:5000/name

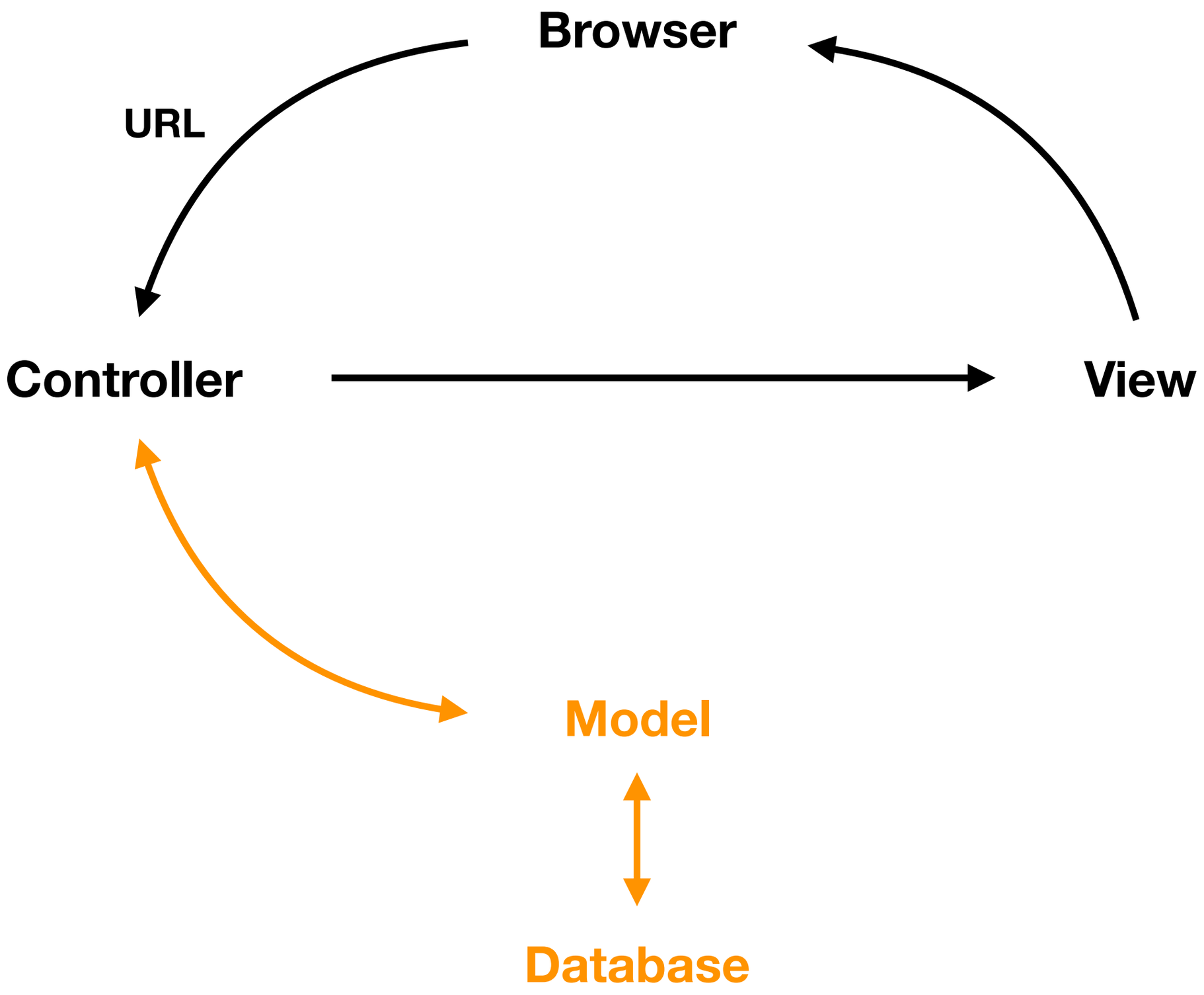
顯示 Hello <Your name>

Exercise 1

```python
# app.py
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

@app.route("/name")
def name():
    return "Hello Frank"
```


Browser
URL
Controller
View
Model
Database

Flask 如何操作 Database？

Install Flask-SQLAlchemy

```
pip install Flask-SQLAlchemy
```

Connect Database

```
# app.py
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test.db'
db = SQLAlchemy(app)
```

如何透過 Python 物件操作 Database？

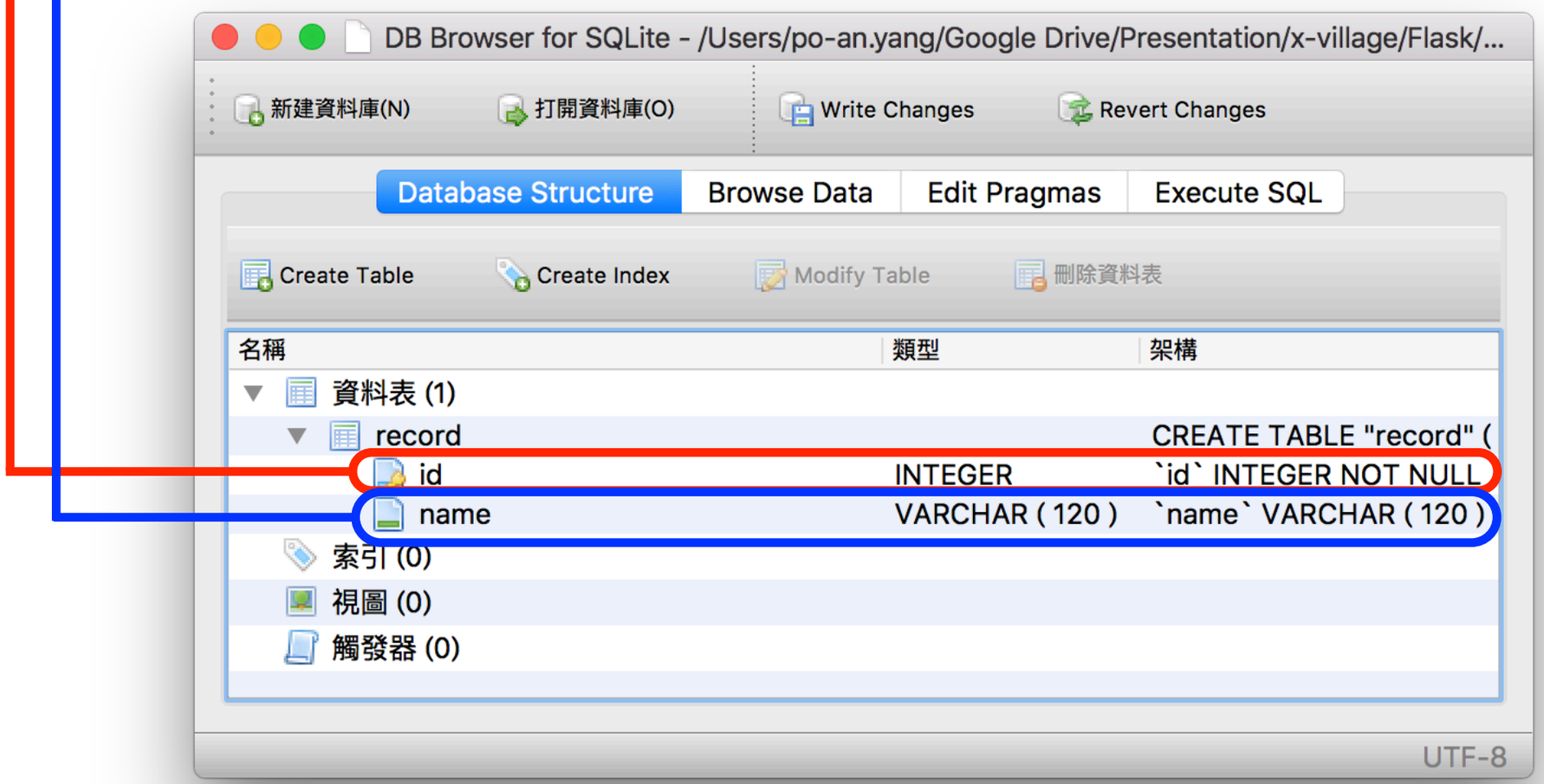
Object-Relational Mapping (ORM)

Object-Relational Mapping (ORM)

將 Database 欄位映射為 Python 物件

Object-Relational Mapping

```python
class Record(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(120), nullable=True)
```



Object-Relational Mapping

```python
class Record(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(120), nullable=True)
```

```python
record = Record(id=1, name='test')
record.id # 1
record.name # 'test'
```

透過物件操作 Database

Flask-SQLAlchemy

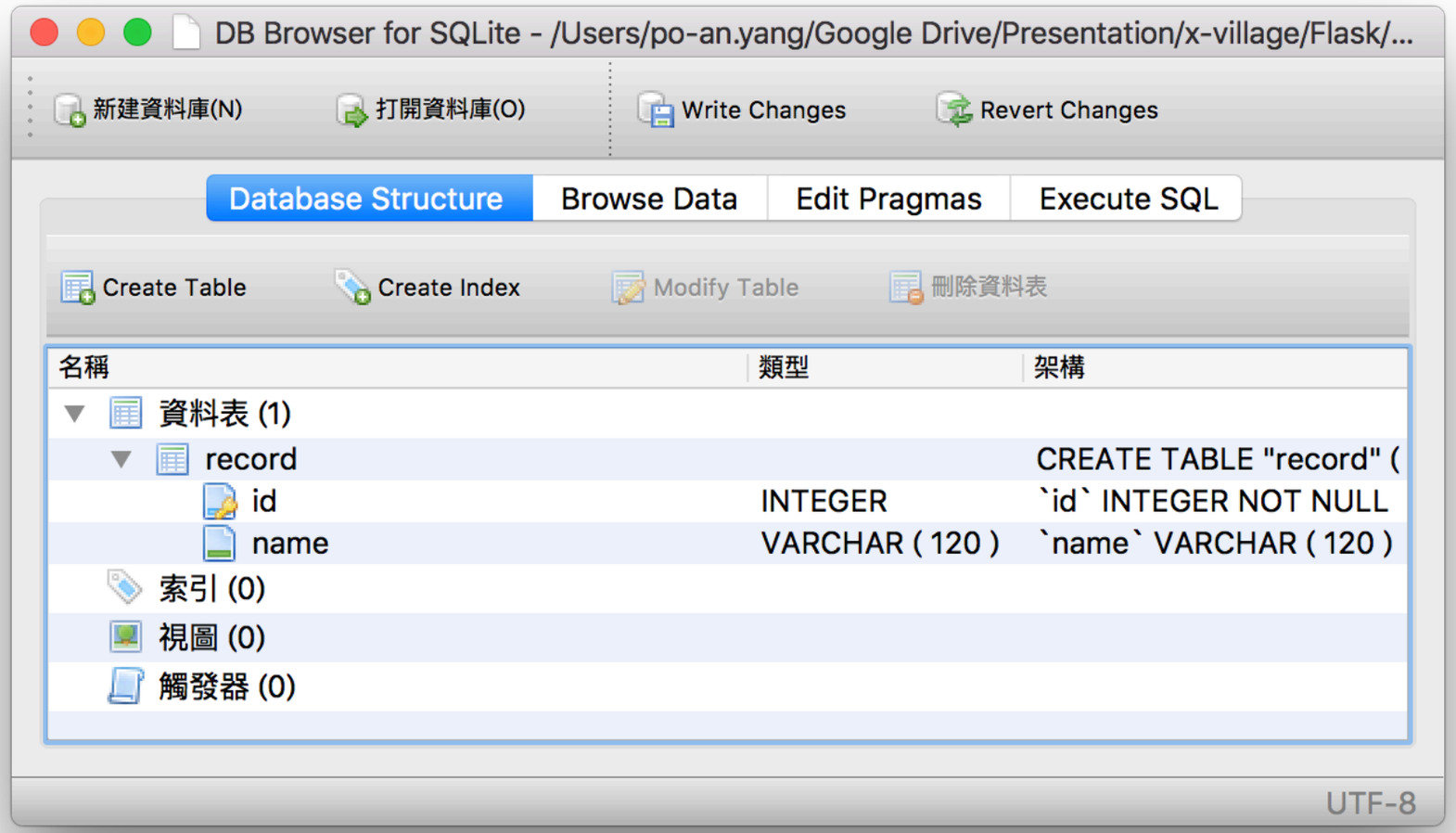
```
# app.py
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test.db'
db = SQLAlchemy(app)

class Record(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(120), nullable=True)
```

Create Database

```
$ python
>>> from app import db
>>> db.create_all()
```



如果想再新增欄位，可以再用一次 *`db.create_all()`* 嗎？

不行！因為 Database 不會做任何更動

請使用 Flask-Migration

Install Flask-Migration

```
pip install Flask-Migration
```

Flask-Migration

```
# app.py
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from flask_migrate import Migrate

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test.db'
db = SQLAlchemy(app)
migrate = Migrate(app, db)

class Record(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(120), nullable=True)
```

Flask-Migration: init

```
$: flask db init
```

```
.
├── app.py
└── migrations
    ├── README
    ├── alembic.ini
    ├── env.py
    ├── script.py.mako
    └── versions
```

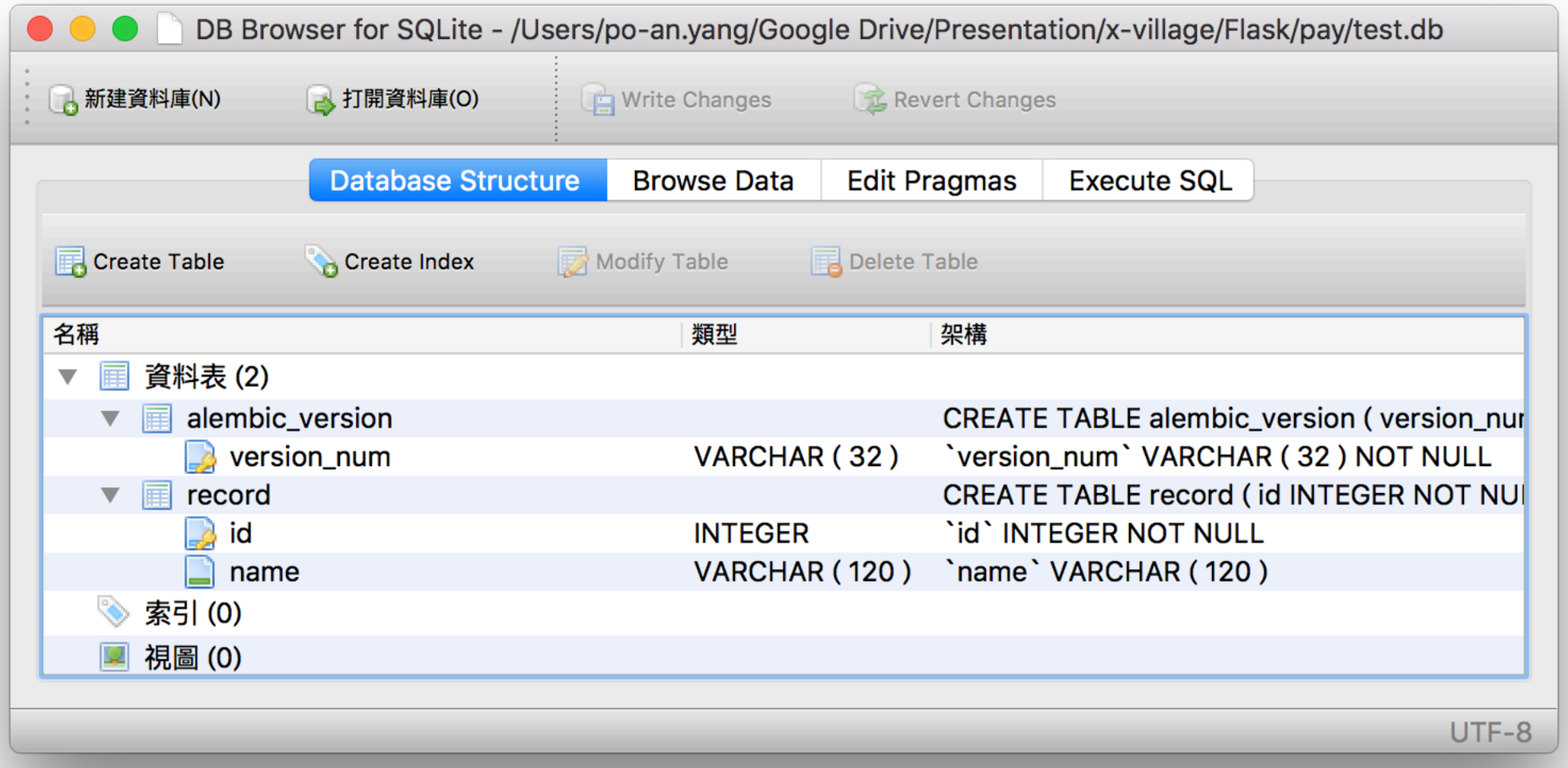
Flask-Migration: migrate

```
$: flask db migrate
```

```
.
├── app.py
├── migrations
│   ├── README
│   ├── alembic.ini
│   ├── env.py
│   ├── script.py.mako
│   └── versions
│       └── 130e52081025_.py
└── test.db
```

Flask-Migration: upgrade

```
$: flask db upgrade
```



可以開始新增欄位了！

Flask-Migration: 新增欄位

```
# app.py
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from flask_migrate import Migrate

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test.db'
db = SQLAlchemy(app)
migrate = Migrate(app, db)

class Record(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(120), nullable=True)
    cost = db.Column(db.Integer, nullable=True)
```

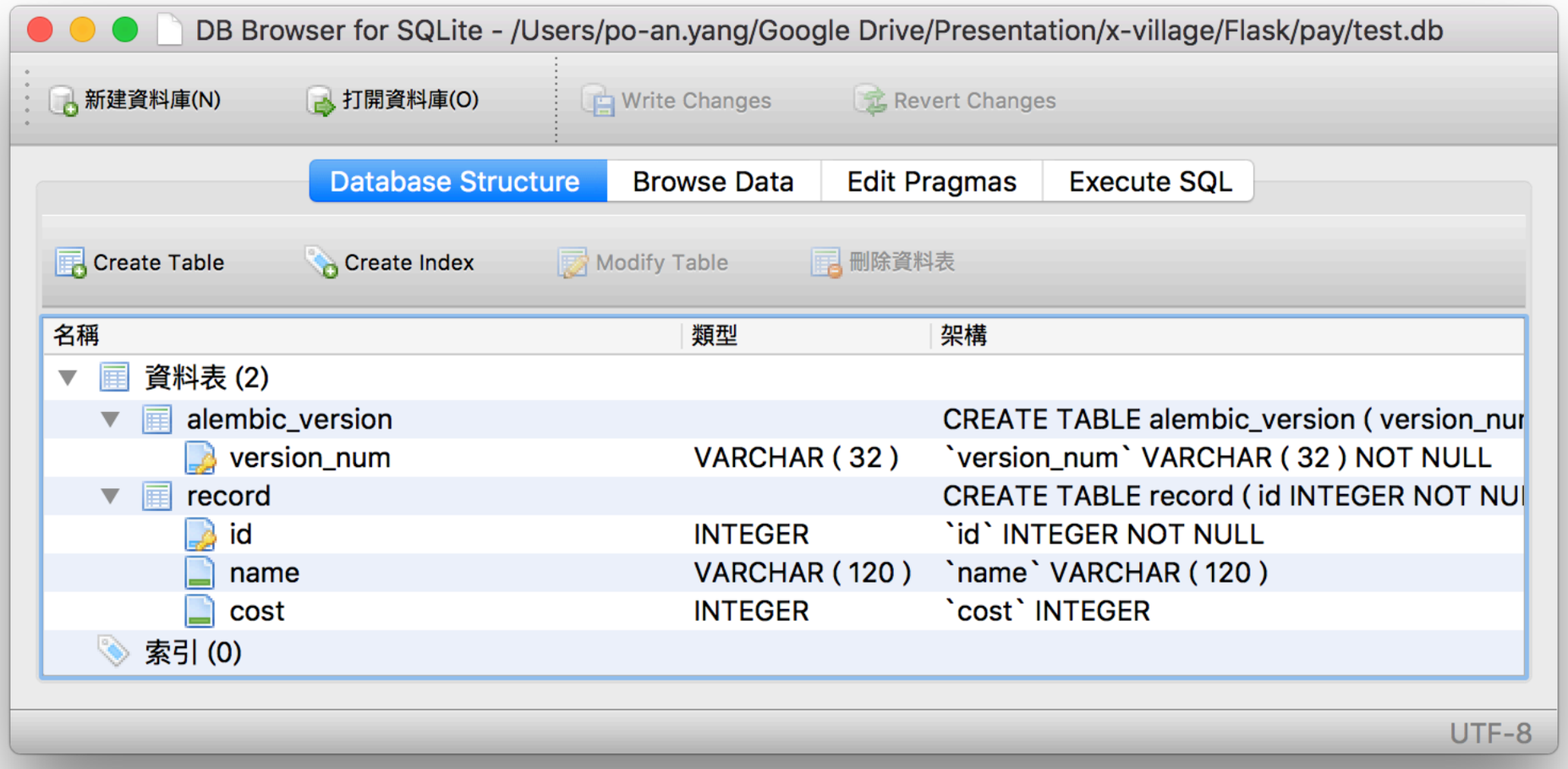
Flask-Migration: 新增欄位

```
$: flask db migrate
```

```
.
├── app.py
├── migrations
│   ├── README
│   ├── alembic.ini
│   ├── env.py
│   ├── script.py.mako
│   └── versions
│       └── 130e52081025_.py
│       └── afb3b6c950b7_.py
└── test.db
```

Flask-Migration: 新增欄位

```
$: flask db upgrade
```



Flask-Migration: 新增欄位

一旦 `flask db init` 初始化設定好，之後要更新 Database 只需要：

```
$: flask db migrate
$: flask db upgrade
```