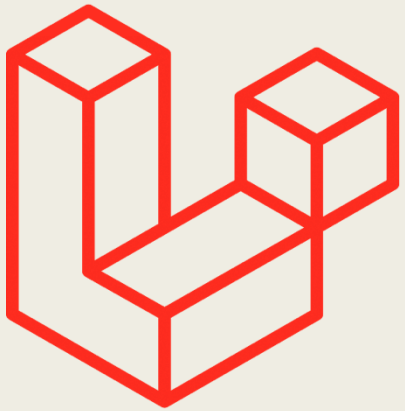
資料庫

朱克剛



資料庫設定

- Laravel 內建資料庫元件，與資料庫連接與執行 SQL 指令非常方便
- 設定在專案根目錄的 .env 檔

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=AddressBook
DB_USERNAME=root
DB_PASSWORD=
```

匯入元件與執行 SQL

- 可以放在路由、Controller 或是 Model 中
- 若要符合 MVC 架構，應放在 Model 中

```
use Illuminate\Support\Facades\DB;

$users = DB::select("select * from UserInfo");
foreach($users as $user) {
   echo $user->cname . "<br>";
}
```

綁定參數

- 語法一

```
$users = DB::select("select * from UserInfo where uid = ?", ['A01']);
```

- 語法二

```
$users = DB::select(
   "select * from UserInfo where uid = :uid",
   ['uid' => 'A01']
);
```

DB 中的重要方法

- DB::select()
- DB::insert()
- DB::update()
- DB::delete()

這三個通用，傳回影響的資料筆數

- DB::scalar() // 傳回單一值的時候使用（僅一筆資料一個欄位）
  - `$address = DB::scalar("select address from House where hid = 1");`
- DB::statement() // 沒有傳回值的 SQL command
  - `DB::statement("drop table ...");`

TRANSACTION

自動

- 當有任何 exception 出現，會自動 rollback，例如主索引重複

```
DB::transaction(function() {
    DB::delete("delete from Live");
    DB::insert("insert into UserInfo values ('A01', '吳小美')");
});
```

錯誤處理

- 透過 try catch 攔截錯誤
- report 會將錯誤訊息寫入 log，最後丟出 503 錯誤
- 需要 use Throwable;

```
try {
    DB::transaction(function() {
        DB::delete("delete from Live");
        DB::insert("insert into UserInfo values ('A01', '吳小美')");
    });
} catch (Throwable $e) {
    report($e);
    abort(503);
}
```

手動 commit 與 rollback

```
try {
    DB::beginTransaction();
    DB::delete("delete from Live");
    DB::insert("insert into UserInfo values ('A01', '吳小美')");
    DB::commmit();
} catch (Throwable $e) {
    DB::rollBack();
    report($e);
    abort(503);
}
```

QUERY BUILDER

目的

- 將查詢的 SQL 指令包裝成 PHP 函數，因此可使用相同函數操作不同的資料庫
- 可有效防範 SQL Injection 攻擊
- 讓又臭又長的 SQL 指令變的優雅
- 目前只能用於資料查詢，不支援資料異動指令

基本查詢

傳回結果為陣列

- 不帶條件：查詢所有使用者資料

```
$users = DB::table('UserInfo')->get();
```

- 條件：查詢王大明資料

```
$users = DB::table('UserInfo')
        ->where('cname', '王大明')
        ->get();
```

查詢特定欄位

- 使用 select() 選取要的欄位，例如只要 UserInfo 中的 cname 與 uid 這兩個欄位

```
$users = DB::table('UserInfo')
            ->select('cname', 'uid')
            ->get();
```

取得一筆資料中的特定欄位值

- 查詢帳號為 A01 的姓名

```
$cname = DB::table('UserInfo')
            ->where('uid', 'A01')
            ->value('cname');
echo $cname;
```

JOIN

- 查詢 A01 的姓名、住址、電話

```
$users = DB::table('UserInfo')
            ->join('Live', 'UserInfo.uid', '=', 'Live.uid')
            ->join('House', 'Live.hid', '=', 'House.hid')
            ->where('UserInfo.uid', 'A01')
            ->get();
```

- 外部連結
  - *將 join() 換成 leftJoin() 或 rightJoin()*
- 交叉連結
  - *將 join() 換成 crossJoin()，並且不設定關連參數*

排序

- 按照帳單金額順向排序

```
$bill = DB::table('Bill')
        ->orderBy('fee')
        ->get();
```

- 反向排序

```
orderBy('fee', 'desc')
```

除錯

- 顯示 SQL 指令與停止執行後續程式碼

```
DB::table('UserInfo')->dd();
```

- 顯示 SQL 指令與繼續執行後續程式碼

```
DB::table('UserInfo')->dump();
```

- 加上 get() 可以用除錯方式顯示查詢結果（dd() 與 dump() 都可以使用）

```
DB::table('UserInfo')->get()->dd();
```

其他指令

- 其他對映於 SQL 指令，例如 group by、distinct、union all...等，請自行連至 Laravel 官網查看
  - *https://laravel.com/docs/10.x/queries*

ELOQUENT

目的

- 將資料庫中的資料表與 Model 直接對映，之後透過物件導向語法就可以操作資料庫
- 優點：純粹 PHP 程式碼，完全不用接觸 SQL 指令

建立對映 Model

- 若資料庫中有一資料表，名稱為 UserInfo，建立與之對映的 Model
  - *php artisan make:model UserInfo*
- 將此 Model 與 UserInfo 資料表對映、設定主索引以及當資料新增與修改時，Eloquent 預設會在每筆資料的 `updated_at` 與 `created_at` 欄位填入異動時間，如資料表沒有這兩個欄位，可把此功能取消

```
class UserInfo extends Model
{
    protected $table = 'UserInfo';
    protected $primaryKey = 'uid'; // 不支援複合欄位
    protected $keyType = 'string'; // 配合 PK 型態，預設 int
    public $timestamps = false;
}
```

查詢資料

- 使用時要 use UserInfo

```
use App\Models\UserInfo;
```

- 查詢所有資料

```
foreach (UserInfo::all() as $user) {
    echo $user->cname . "<br>";
}
```

- 查詢部分資料

```
foreach (UserInfo::where('uid', 'A01')->get() as $user) {
    echo $user->cname . "<br>";
}
```

新增資料

- 在 UserInfo 中新增一筆資料

```
$user = new UserInfo;
$user->uid = 'Z01';
$user->cname = '吳小美';

$user->save();
```

修改資料

- 將 Z01 的名字改為「吳美美」
- 方法一：使用 find()

```
$user = UserInfo::find('Z01');
$user->cname = '吳美美';
$user->save();
```

find() 函數只找主索引欄位

- 方法二

```
UserInfo::where('uid', 'Z01')
        ->update(['cname' => '吳美美']);
```

刪除資料

delete 後不用下 save()，如果反悔了，可以下 save() 把資料重新寫回資料庫

- 將 Z01 的資料刪除
- 方法一：使用 find()

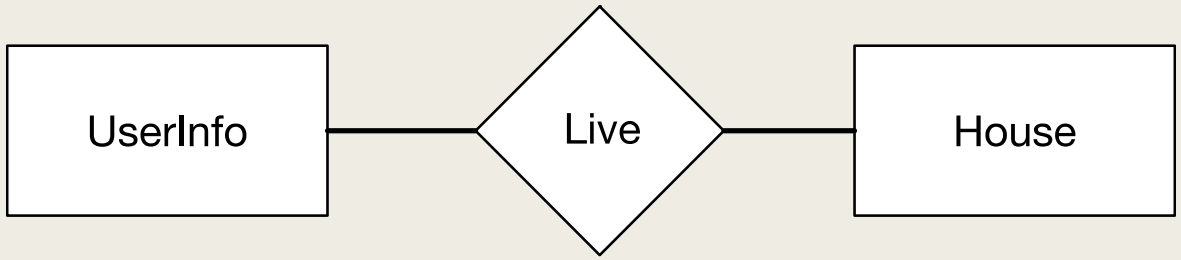
```
$user = UserInfo::find('Z01');
$user->delete();
```

- 方法二

```
UserInfo::where('uid', 'Z01')->delete();
```

關連

多對多關連



- 設定 UserInfo 與 House 間的多對多關連。開啟 UserInfo.php

```php
use Illuminate\Database\Eloquent\Relations\BelongsToMany;

class UserInfo extends Model
{
    ...
    public function lives(): BelongsToMany {
        return $this->belongsToMany(
            House::class,
            'Live', // 多對多關連所衍生出來的資料表名稱
            'uid',  // Live.uid
            'hid'   // Live.hid
        );
    }
}
```

多對多關連查詢

- 根據 UserInfo 的 uid 查詢姓名與住址

```php
Route::get('/query/{uid}', function($uid) {
    $user = UserInfo::find($uid);
    echo $user->cname . '<br>';

    foreach($user->lives as $house) {
        echo $house->address . '<br>';
    }
});
```

lives 不要加 ()

多對多關連新增資料

- 設定 A04 住在 4 號屋子裡

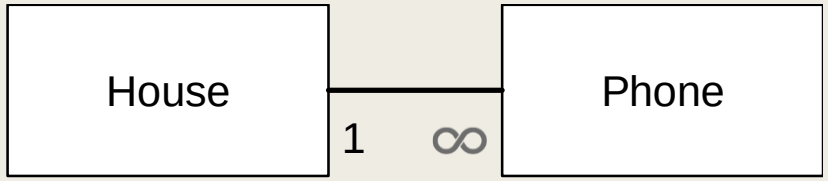
```php
$user = UserInfo::find('A04');
$house = House::find(4);

$user->lives()->save($house);
```

lives 要加 ()

另一個範例

```php
$user = new UserInfo();
$user->uid = $uid;
$user->cname = $cname;

$house = new House();
$house->address = $address;

$user->lives()->save($house);
$user->save();
```

一對多關連



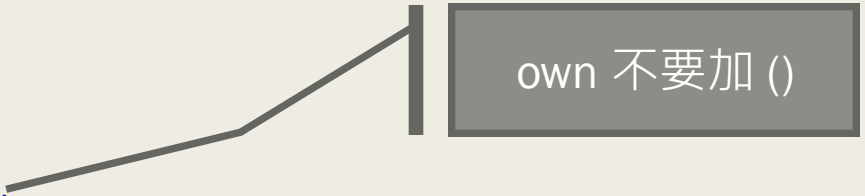
- 設定 House 中與 Phone 間的關係

```php
use Illuminate\Database\Eloquent\Relations\HasMany;

class House extends Model
{
    ...
    public function own(): HasMany {
        return $this->hasMany(
            Phone::class,
            'hid', // Phone.hid
            'hid'  // House.hid
        );
    }
}
```

一對多關連查詢

- 列出 1 號屋子中有多少電話

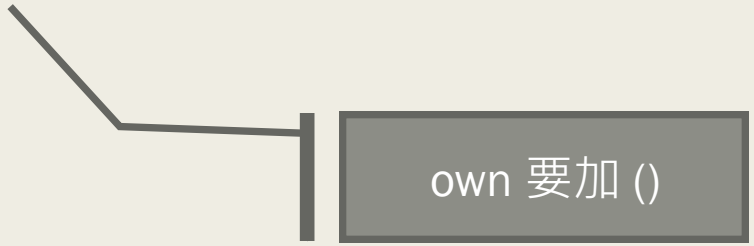
```php
$house = House::find(1);

foreach($house->own as $phone) {
    echo $phone->tel . "<br>";
}
```



一對多關連新增資料

- 在 4 號屋子中安裝電話

```php
$house = House::find(4);
$phone = new Phone();
$phone->tel = '1414';

$house->own()->save($phone);
```



一對多關連但經過中間資料表

- 例如 House 與 Bill 的關係是一對多，但是經過了 Phone

```php
use Illuminate\Database\Eloquent\Relations\HasManyThrough;

class House extends Model
{
    ...
    public function bills(): HasManyThrough {
        return $this->hasManyThrough(
            Bill::class, // 目標
            Phone::class, // 經過
            'hid', // Phone.hid
            'tel', // Bill.tel
            'hid', // House.hid
            'tel' // Phone.tel
        );
    }
}
```

此種沒有 save()，不可用他來儲存資料到 Phone

一對多關連但經過中間資料表查詢

- 列出 1 號屋子的所有帳單資料

```
$house = House::find(1);
foreach($house->bills as $bill) {
    echo "{$bill->tel} {$bill->dd}: {$bill->fee}<br>";
}
```

- 輸出結果

```
1111 2019-01-01 00:00:00: 300
1111 2019-02-01 00:00:00: 700
1112 2019-01-01 00:00:00: 700
1112 2019-02-01 00:00:00: 450
1112 2019-03-01 00:00:00: 200
```

轉成 JSON 字串

將查詢結果以 JSON 格式輸出

- 可以將此路由放到 api.php 中，輸入以下網址試試
  - *http://hostname/laravel/public/**api**/userinfo*

有中文時一定要加

```
Route::get('/userinfo', function() {
    $users = UserInfo::all();
    $json = $users->toJson(JSON_UNESCAPED_UNICODE);

    return response($json)
            ->header('content-Type', 'application/json')
            ->header('charset', 'utf-8');
});
```