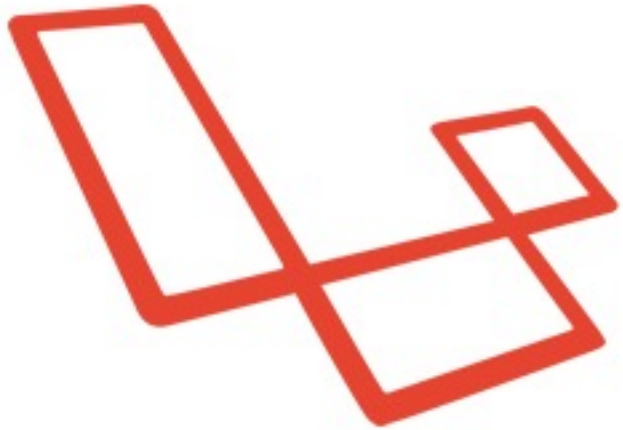
Laravel Eloquent ORM

Giảng viên: Bùi Quang Đăng

Contents

Laravel Framework



Laravel

LÀM VIỆC VỚI MYSQL TRONG LARAVEL

Làm việc với MySQL trong Laravel

- Laravel hỗ trợ thực hiện làm việc với cơ sở dữ liệu MySQL bằng các lệnh SQL truyền thống (raw SQL query)
- Có thể thực hiện và làm việc với nhiều database qua hàm connection và tên kết nối trong config/database.php như sau:

```
$users = DB::connection('foo')->select(...);
```

Làm việc với MySQL trong Laravel

- Khai báo lớp sử dụng:

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

- Thực hiện các câu lệnh với SQL trong Laravel:

```
DB::select("Select * from Users");
```

Làm việc với MySQL trong Laravel

- Hàm select thực hiện để lấy thông tin
  - Cú pháp:

```
DB::select("Câu lệnh SQL cần lấy thông tin");
```

```php
public function danhSach()
{
    $chuDe = DB::select('select * from ChuDe');

    return view('chude.danhsach', ['chude'=>$chuDe]);
}
```

Làm việc với MySQL trong Laravel

- Hàm select để lấy thông tin chi tiết:
  - :ma là tên tham số và gán giá trị như câu lệnh dưới

```
if(isset($id))
{
    $chuDe =  DB::select('select * from ChuDe where MaChuDe =:ma',
    ['ma'=>$id]);
    $objChuDe;
    foreach($chuDe as $cd)
    {
        $objChuDe = $cd;
    }
}
return view('chude.chudeadd',['ma'=>$id, 'chuDe'=>$objChuDe]);
```

Làm việc với MySQL trong Laravel

- Hàm insert, update để thực hiện thêm mới hoặc cập nhật lại thông tin:

```
$ma = $request->input("txtMaChuDe");
$ten = $request->input("txtTenChuDe");
$moTa = $request->input("txtMoTa");
if(isset($id)){
    DB::update('update ChuDe set TenChuDe=?, MoTa=? where
    MaChuDe=?', [$ten, $moTa, $ma]);
    $thongBao = "Cập nhật chủ đề thành công";
}
else {
    DB::insert('insert into ChuDe values(?, ?, ?)', [$ma, $ten,
    $moTa]);
    $thongBao = "Thêm mới chủ đề thành công";
}
```

Làm việc với MySQL trong Laravel

- Hàm delete để xoá thông tin:

```
public function xoaChuDe($id)
{
    if(isset($id))
    {
        DB::delete('Delete from ChuDe where MaChuDe =:ma',['ma'=>$id]);
        return redirect('chude-list');
    }

    return view('chude.danhsach');
}
```

Làm việc với MySQL trong Laravel

- Thực hiện một câu lệnh SQL bất kỳ trong Laravel với cú pháp như sau:

```
DB::statement('drop table users');
```

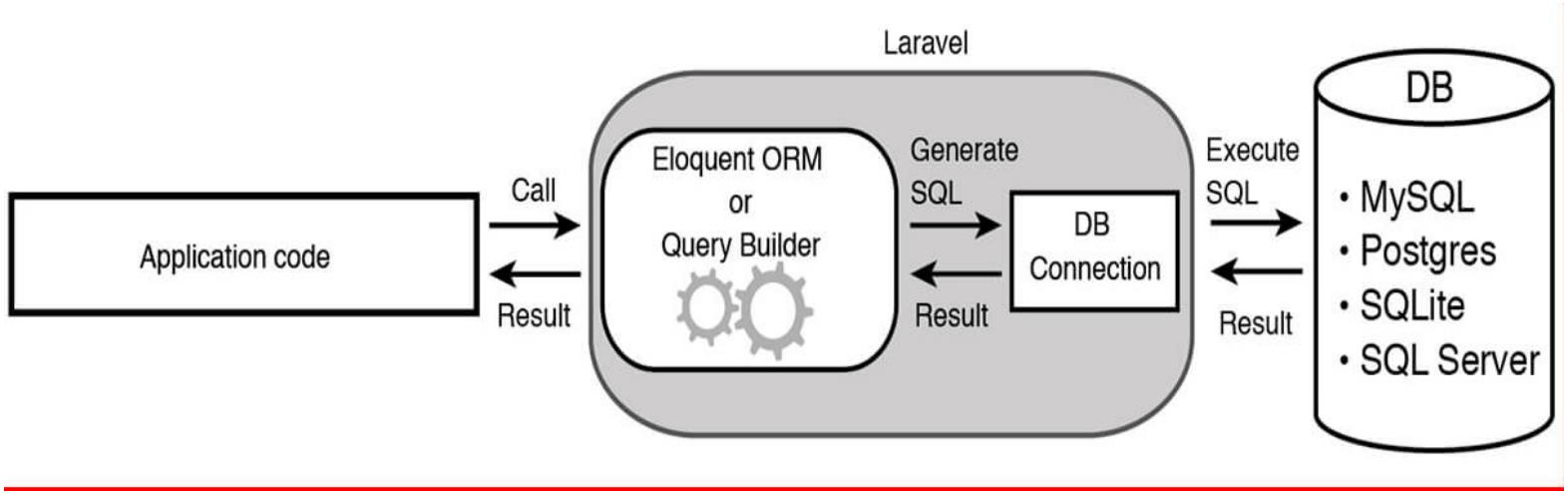
  - Câu lệnh trên thực hiện xoá bảng users.

LÀM VIỆC VỚI QUERY BUILDER

Làm việc với Query Builder trong Laravel

- Khi làm việc với câu lệnh SQL thuần người lập trình sẽ thấy đơn giản nhưng sẽ có một số vấn đề hạn chế khi câu lệnh phức tạp, lỗi bảo mật sql injection,…
- Làm việc với cơ sở dữ liệu bằng Query Builder sử dụng PDO để xử lý, khắc phục được các hạn chế khi dùng với SQL thuần.

❖ Làm việc với Query Builder trong Laravel



Làm việc với Query Builder trong Laravel

- Hàm lấy danh sách trong một bảng:
  - Cú pháp:

```
DB::table("tên bang")->get();
```

  - Ví dụ: Lấy danh sách từ bảng ChuDe

```
$chuDe = DB::table('ChuDe')->get();

return view('chude.danhsach', ['chude'=>$chuDe]);
```

Làm việc với Query Builder trong Laravel

- Hàm lấy chi tiết trong một bảng:
  - Cú pháp:

```
DB::table("tên bang")->where([Điều kiện])->first();
```

  - Ví dụ: Lấy thông tin chi tiết bảng ChuDe theo mã

```
$chuDe = DB::table('ChuDe')->where('MaChuDe',$id)->first();

return view('chude.chudeadd',['ma'=>$id, 'chuDe'=>$chuDe]);
```

Làm việc với Query Builder trong Laravel

- Laravel hỗ trợ các phương thức tổng hợp:
  - min: giá trị nhỏ nhất
  - max: giá trị lớn nhất
  - avg: giá trị trung bình
  - count: tổng số bản ghi
  - sum: lấy tổng một trường nào đó của bảng.

```
$users = DB::table('users')->count();
$price = DB::table('orders')->max('price');
```

Làm việc với Query Builder trong Laravel

- Liệt kê các trường cần lấy, không lấy trùng lặp:

```
$users = DB::table('users')->select(username', 'email as user_email')->get();


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

Làm việc với Query Builder trong Laravel

- Sử dụng Raw expression:

```
$users = DB::table('users') ->select(DB::raw('count(*)
as user_count, status'))
->where('status', '<>', 1)
->groupBy('status')
->get();
```

Làm việc với Query Builder trong Laravel

- Sử dụng Where để lọc thông tin:

```
$users = DB::table('users')->where('votes', '=', 100)-
>get();
$users = DB::table('users') ->where('name', 'like',
'%Nam%') ->get();

//Sử dụng orWhere
$users = DB::table('users') ->where('vote', '>', 100) -
>orWhere('comment', '>', 100) ->get();
//Sử dụng whereBetween
$users = DB::table('users') ->whereBetween('age', [18, 35])-
>get();
//Sử dụng về whereIn
$users = DB::table('users') ->whereIn('id', [1, 2, 3]) ->get();
```

Làm việc với Query Builder trong Laravel

- Sử dụng Where để lọc thông tin:

```
//whereNull
$users = DB::table('users') ->whereNull('vote') ->get();
//whereDate
$users = DB::table('users') ->whereDate('join_date', `2020-03-
10') ->get();
//whereDay
$users = DB::table('users') ->whereDay('join_date', `10') ->get();
//whereMonth
$users = DB::table('users') ->whereMonth('join_date', '03') -
>get();
```

Làm việc với Query Builder trong Laravel

- Sử dụng Where để lọc thông tin:

```
//whereYear
$users = DB::table('users') ->whereYear('join_date', '2017') -
>get();
//whereColumn
$users = DB::table('account') ->whereColumn([
['working_balance', '=', 'last_balance'], ['updated_at', '>',
'created_at'] ])
->get();
//whereExist DB::table('users')
->whereExists(function ($query) {
$query->select(DB::raw(1)) ->from('orders')
->whereRaw('orders.user_id = users.id');
}) ->get();
```

Làm việc với Query Builder trong Laravel

- Sử dụng Where để lọc thông tin:

```
//Sử dụng nhóm các điều kiện trong mệnh đề điều kiện
DB::table('users')
->where('name', '=', 'FirebirD')
->orWhere(function ($query) {
$query->where('vote', '>', 50)
->where('role', '<>', 'superadmin');
})
->get();
```

Làm việc với Query Builder trong Laravel

- Sắp xếp thông tin:

```
$users = DB::table('users')
->orderBy(join_date', 'desc')
->get();
```

- Lấy ngẫu nhiên 1 bản ghi:

```
$randomUser = DB::table('users')
->inRandomOrder()
->first();
```

Làm việc với Query Builder trong Laravel

- Sử dụng Group và Having:

```
users = DB::table('users’)
->groupBy('account_id’)
->having('account_id', '>’, 50)
->get();
```

Làm việc với Query Builder trong Laravel

- Thêm mới một hoặc nhiều bản ghi:

```
//Thêm một bản ghi
DB::table('users')->insert( ['name' => 'Nguyễn Văn A', 'email'
=> 'anv@gmail.com', 'votes' => 0] );
//Thêm nhiều bản ghi
DB::table('users')->insert([
['name' => 'Nguyễn Văn A', 'email' => 'anv@gmail.com', 'votes'
=> 0],
['name' => 'Trần Thị B', 'email' => 'btt@gmail.com', 'votes' =>
0],
['name' => 'Vũ Văn C', 'email' => 'cvv@gmail.com', 'votes' => 0]
]);
```

Làm việc với Query Builder trong Laravel

- Thêm mới một hoặc nhiều bản ghi:
  - Sau khi thực hiện sẽ trả về id của thông tin vừa them:

```
$user_id = DB::table('users')->insertGetId(
['name' => 'Nguyễn Văn A',
'email' => 'anv@gmail.com',
'votes' => 0] );
echo 'User vừa thêm mới có id là' . $user_id;
```

Làm việc với Query Builder trong Laravel

- Cập nhật lại thông tin:

```
DB::table('users')
->whereYear('join_date', '2020')
->orWhere('vote', '>', 50)
->update(['generate_promote_code' => 1,
'fullname'=>'John Hat']);
```

- Xoá dữ liệu:

```
DB::table('users')
->where('active', '=', 0)
->whereYear('join_date', '2018')
->delete();
```

Làm việc với Query Builder trong Laravel

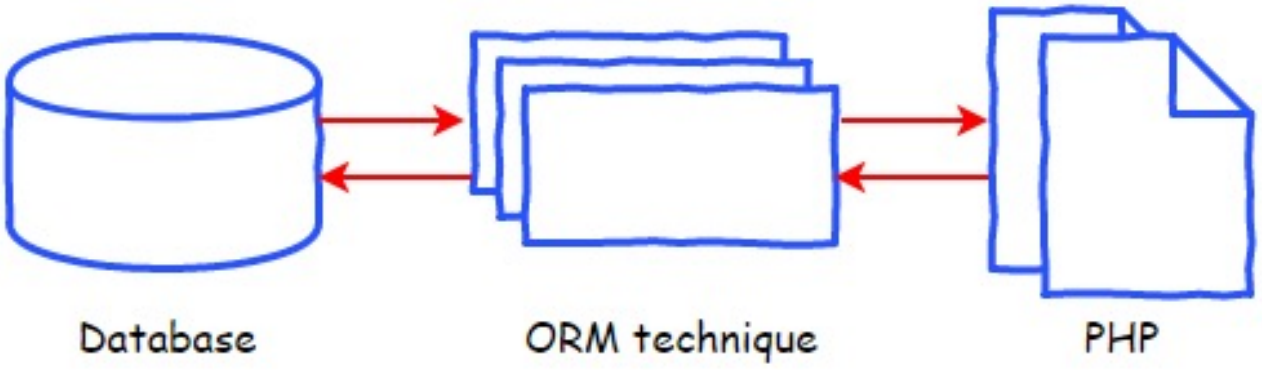
- Xoá thông tin và reset lại id về 0 sử dụng:

```
DB::table('users')->truncate();
```

LÀM VIỆC VỚI
Laravel Eloquent ORM

Làm việc với Laravel Eloquent ORM

- **ORM** (**O**bject **R**elational **M**apping) là một kỹ thuật lập trình dùng để chuyển đổi các bảng trong cơ sở dữ liệu sang các đối tượng như lập trình hướng đối tượng trong PHP.



Làm việc với Laravel Eloquent ORM

- Trong phần này chúng ta sẽ nói nhiều đến Laravel Eloquent Model (gọi tắt là Model) là một phần trong mô hình MVC ở trên, các Model này sẽ thao tác trực tiếp với cơ sở dữ liệu, xử lý các logic nghiệp vụ và trả dữ liệu về cho các Controller.

❖ Làm việc với Laravel Eloquent ORM

- **Tạo Model để sử dụng:**

```
php artisan make:model [Tên model]
```

- Ví dụ: Tạo model chứa thông tin TacGia

```
php artisan make:model TacGia
```

❖ Làm việc với Laravel Eloquent ORM

- **Tạo Model để sử dụng:**
  - Sử dụng Laravel Migration, Laravel cũng hỗ trợ chúng ta một cách tự động luôn là tạo Model kèm với file migrate luôn thông qua việc thêm tham số --migrate hoặc -m trong câu lệnh tạo Model:

```
php artisan make:model TacGia --migration
php artisan make:model TacGia -m
```

❖ Làm việc với Laravel Eloquent ORM

- **Tạo Model để sử dụng:**
  - Khai báo các cột trong bảng nếu chưa có

```
Schema::create('TacGia', function (Blueprint $table) {
    $table->id('MaTG');
    $table->string('HoTen')->nullable();
    $table->integer('GioiTinh')->nullable();
    $table->date('NgaySinh')->nullable();
    $table->string('DienThoai')->nullable();
    $table->string('Email')->nullable();
    $table->string('DiaChi')->nullable();
    //$table->timestamps();
});
```

❖ Làm việc với Laravel Eloquent ORM

- **Tạo Model để sử dụng:**
  - Khai báo thông tin trong class tạo ra nếu cần

```
// Thay đổi các thiết lập ngầm định của Eloquent Model
protected $table = 'TacGia';
public $primaryKey = 'MaTG';
public $incrementing = false;

//Không sử dụng thông tin log của model khi thay đổi
public $timestamps = false;
```

❖ Làm việc với Laravel Eloquent ORM

- **Tạo Model để sử dụng:**
  - **Mass Assignment** là tính năng cho phép lập trình một cách tự động gán các tham số của một HTTP request vào các biến hoặc đối tượng trong lập trình.
    - Ví dụ: Chúng ta có một form đăng ký người dùng như sau, các tên trường nhập liệu trùng với tên cột trong bảng users trong CSDL.

❖ Làm việc với Laravel Eloquent ORM

- Tạo Model để sử dụng:
  - **Mass Assignment**: Khai báo các trường cho phép sử dụng trong lớp tương ứng

```php
/**
 * Thiết lập các cột tự động gán được dữ liệu từ form
 *
 * @var array
 */
protected $fillable = [
    'MaTG',
    'HoTen',
    'GioiTinh',
    'NgaySinh',
    'DienThoai',
    'Email',
    'DiaChi',
];
```

Làm việc với Laravel Eloquent ORM

- Hàm lấy danh sách

```
//Hàm lấy danh sách tác giả
public function layDanhSach()
{
    $tacGia = TacGia::all();

    return view('tacgia.danhsach', ['tacgia'=>$tacGia]);
}
```

Làm việc với Laravel Eloquent ORM

- Hàm lấy thông tin chi tiết theo mã

```php
//Lấy chi tiết
$objTacGia = TacGia::find(1);

//Hoặc
$objTacGia = TacGia::where('MaTG', 1)->first();
```

Làm việc với Laravel Eloquent ORM

- Hàm lấy thông tin chi tiết theo mã

```php
//Lấy chi tiết
$objTacGia = TacGia::find(1);

//Hoặc
$objTacGia = TacGia::where('MaTG', 1)->first();
```

Làm việc với Laravel Eloquent ORM

- Hàm thêm mới thông tin

```
//Lấy thông tin từ giao diện
$hoTen = $request->input('HoTen');
$dienThoai = $request->input('DienThoai');
$email = $request->input('Email');
$diaChi = $request->input('DiaChi');
//Thêm mới
TacGia::create(['HoTen'=>$hoTen, 'DienThoai'=>$dienThoai,
'Email'=>$email,'DiaChi'=>$diaChi]);
```

Làm việc với Laravel Eloquent ORM

- Hàm cập nhật sửa đổi thông tin

```php
//Lấy thông tin từ giao diện
$hoTen = $request->input('HoTen');
$dienThoai = $request->input('DienThoai');
$email = $request->input('Email');
$diaChi = $request->input('DiaChi');
//Lấy thông tin cần sửa
$tacGia = TacGia::find($id);
$tacGia->HoTen = $hoTen;
$tacGia->DienThoai = $dienThoai;
$tacGia->Email = $email;
$tacGia->DiaChi = $diaChi;
//Lưu lại
$tacGia->save();
```

Làm việc với Laravel Eloquent ORM

- Hàm xoá mềm thông tin

```php
public function xoaMemTacGia($id)
{
    $tacGia = TacGia::find($id);
    //Xoá mềm bằng việc chuyển trạng thái
    $tacGia->delete();

    if ($tacGia->trashed()) {
        echo "Xoá tạm sản phẩm thành công";
    }

    return redirect('tacgia');
}
```

Làm việc với Laravel Eloquent ORM

- Hàm lấy danh sách thông tin đã xoá mềm

```
public function layDanhSachXoa()
{
    $tacGia = TacGia::onlyTrashed()->get();

    return view('tacgia.danhsachxoa', ['tacgia'=>$tacGia]);
}
```

Làm việc với Laravel Eloquent ORM

- Xoá thông tin khỏi bảng của cơ sở dữ liệu

```
public function xoaTacGia($id)
{
    $tacGia = TacGia::onlyTrashed()
            ->where('MaTG', $id)->first();
    //Xoá khỏi thông tin trong bảng
    $tacGia->forceDelete();

    //Di chuyển đến trang xoá
    return redirect('tacgiaxoa');
}
```

❖ Làm việc với Laravel Eloquent ORM

- Biểu diễn quan hệ 1-M trong cơ sở dữ liệu
  - Trong lớp ChuDe

```php
//Một chủ đề có nhiều sách
public function Sachs()
{
    return $this->hasMany('App\Models\Sach', 'MaChuDe', 'MaChuDe');
}
```

  - Trong lớp Sach

```php
//Một sách chỉ thuộc 1 chủ đề
public function ChuDe()
{
    return $this->belongsTo('App\Models\ChuDe','MaChuDe','MaChuDe');
}
```

❖ Làm việc với Laravel Eloquent ORM

- Biểu diễn quan hệ 1-M trong cơ sở dữ liệu
  - Xử lý lấy các sách trong chủ đề

```
$sach = ChuDe::find($id);

foreach($sach->Sachs as $s)
{
    echo $s->TenSach . "<br>";
    echo $s->ChuDe->TenChuDe . "<br>";
}
```

Exercises


Stanford
nâng tầm tri thức
Thank You !
www.stanford.com.vn