
LAB 8 MÔN LẬP TRÌNH PHP3: RESTFUL API WITH LARAVEL**MỤC TIÊU:**

Kết thúc bài thực hành này bạn có khả năng

- ✓ Hiểu về Restful API của ứng dụng web
- ✓ Triển khai Restful API trong Laravel

NỘI DUNG**Chuẩn bị project:**

1. Cài Laravel vào folder **la8**
2. Tạo database có tên **la8** và file .env cấu hình kết nối đến database vừa tạo.
3. Cài đặt postman (nếu chưa cài)

Bài 1: Triển khai Resful

1. Tạo table và dữ liệu
 - a. Sinh viên tạo table products với thông tin yêu cầu như sau

```
public function up() {  
    Schema::create('products', function (Blueprint $table) {  
        $table->id();  
        $table->string('tenSP');  
        $table->integer('gia');  
        $table->integer('anHien')->default(1);  
        $table->timestamps();  
    });  
}
```

- b. Nhập vài record dữ liệu vào table (dùng seeder)
 - c. Tạo model **Product** rồi mở file model khai báo các field :

```
class Product extends Model{  
    use HasFactory;  
    protected $fillable = ['tenSP', 'gia', 'anHien'];  
}
```

- d. Tạo eloquent resource
 - Chạy lệnh sau để tạo eloquent resource có tên Product

```
php artisan make:resource Product
```

- Mở file resouse mới tạo (trong app/http/Resources) , khai báo tên thân thiện và định dạng cho các field

```
public function toArray($request) {
    return [
        'id' => $this->id,
        'tenSP' => $this->tenSP,
        'giaSP' => $this->gia,
        'status' => $this->anHien,
        'created_at' => $this->created_at->format('d/m/Y'),
        'updated_at' => $this->updated_at->format('d/m/Y'),
    ];
}
```

e. Tạo controller resource

- Chạy lệnh:

```
php artisan make:controller ProductController --resource
```

- Thêm code ở đầu controller: (khai báo dùng Validator, model Product và ProductResource)

```
use Validator;
use App\Models\Product;
use App\Http\Resources\Product as ProductResource;
```

f. Tạo Route API: routes/api.php

```
use App\Http\Controllers\ProductController;
Route::resource('products', ProductController::class);
```

g. Xem kết quả các route được hỗ trợ: **php artisan route:list**

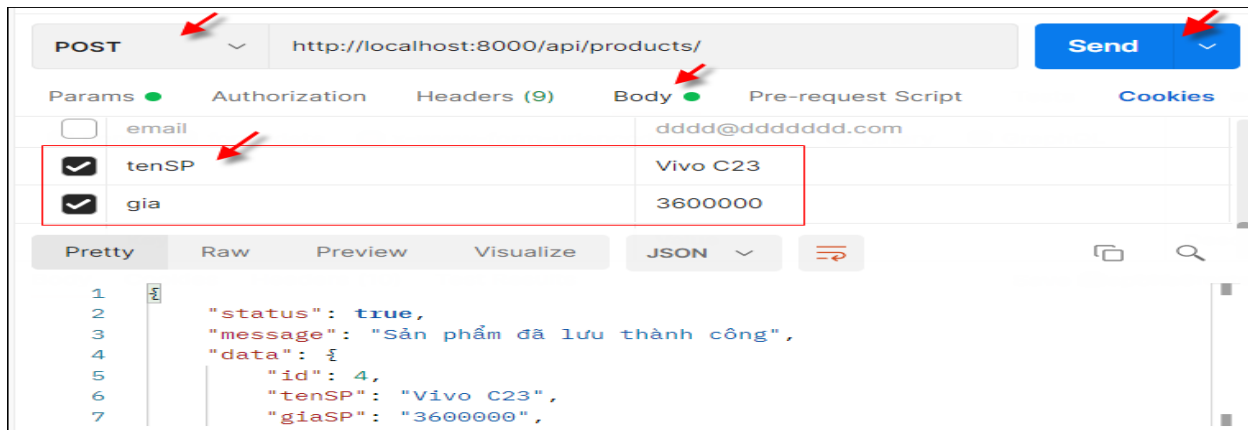
Bài 2: Tạo các chức năng quản lý resource

1. Tạo resource mới

- Code trong hàm **store** của ProductController:

```
public function store(Request $request) {
    $input = $request->all();
    $validator = Validator::make($input, ['tenSP'=>'required', 'gia'=>'required']);
    if($validator->fails()){
        $arr = ['success' => false,
            'message' => 'Lỗi kiểm tra dữ liệu',
            'data' => $validator->errors()
        ];
        return response()->json($arr, 200);
    }
    $product = Product::create($input);
    $arr = ['status' => true,
        'message'=>"Sản phẩm đã lưu thành công",
        'data'=> new ProductResource($product)
    ];
    return response()->json($arr, 201);
}
```

- Test với Postman, xong xem trong db phải có dữ liệu mới

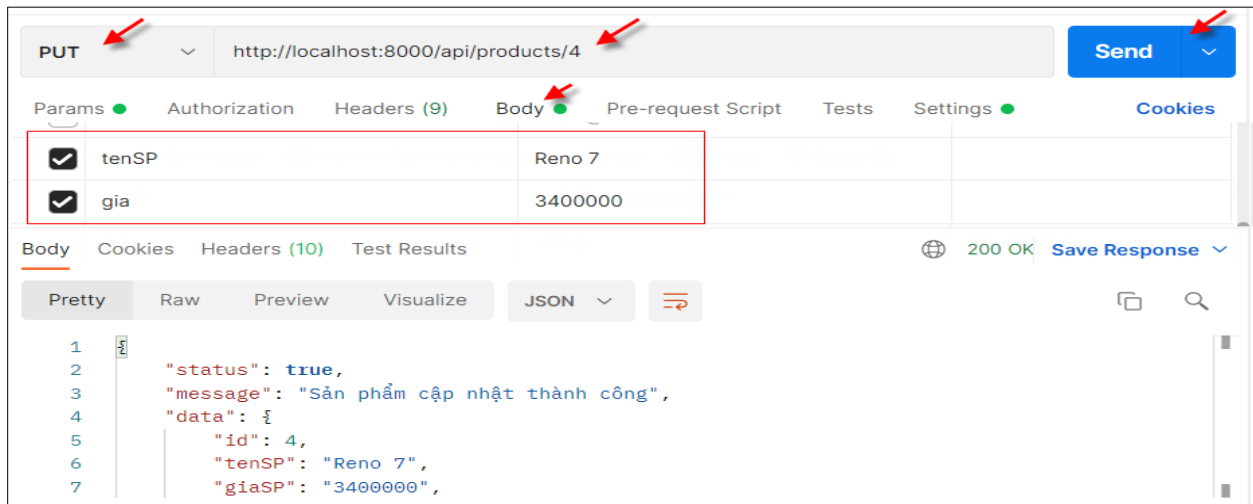


2. Cập nhật 1 resource

- Code trong hàm **update** của ProductController:

```
public function update(Request $request, Product $product){
    $input = $request->all();
    $validator = Validator::make($input, ['tenSP'=>'required', 'gia'=>'required']);
    if($validator->fails()){
        $arr = ['success' => false,
            'message' => 'Lỗi kiểm tra dữ liệu',
            'data' => $validator->errors()
        ];
        return response()->json($arr, 200);
    }
    $product->tenSP = $input['tenSP'];
    $product->gia = $input['gia'];
    $product->save();
    $arr = ['status' => true,
        'message' => 'Sản phẩm cập nhật thành công',
        'data' => new ProductResource($product)
    ];
    return response()->json($arr, 200);
}
```

- Test với Postman, xong xem trong db phải thấy dữ liệu mới được cập nhật



3. Trả về danh sách resource

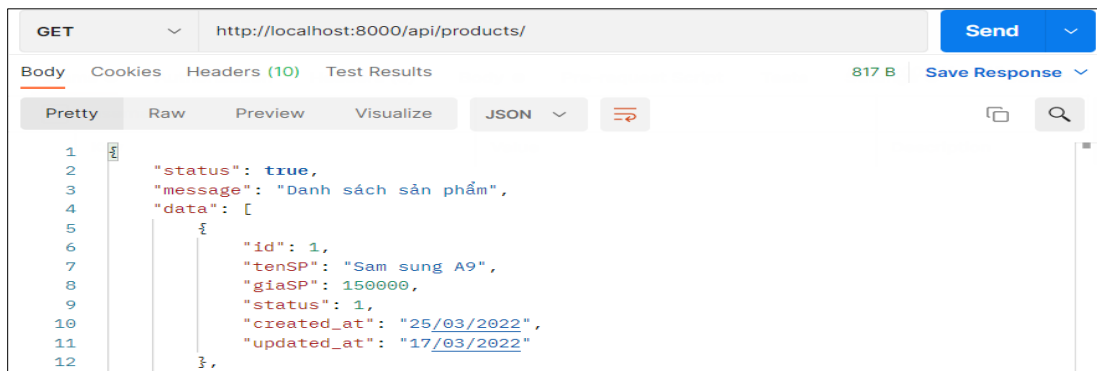
- Code trong hàm **index** của ProductController:

```

public function index() {
    $products = Product::all();
    $arr = [
        'status' => true,
        'message' => "Danh sách sản phẩm",
        'data'=>ProductResource::collection($products)
    ];
    return response()->json($arr, 200);
}

```

- Test với Postman, sẽ thấy dữ liệu. *Chú ý : các cột created_at và updated_at không được có giá trị null, nếu không , hàm format trong Eloquent Resource sẽ báo lỗi*

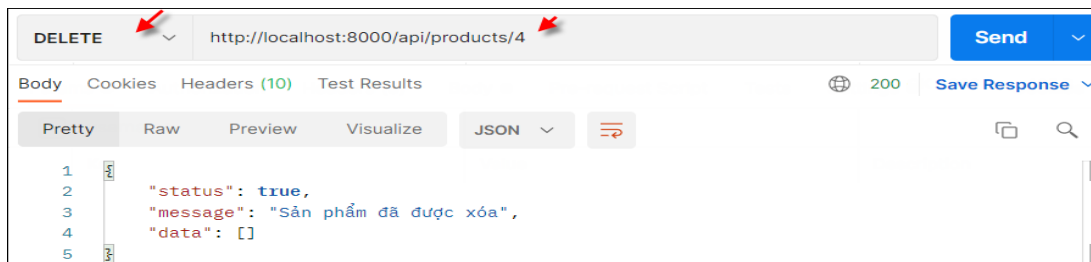


4. Xóa 1 resource

- Code trong hàm **destroy()** của ProductController:

```
public function destroy(Product $product){
    $product->delete();
    $arr = [
        'status' => true,
        'message' => 'Sản phẩm đã được xóa',
        'data' => [],
    ];
    return response()->json($arr, 200);
}
```

- Test với Postman, xong xem trong db phải thấy dữ liệu bị xóa

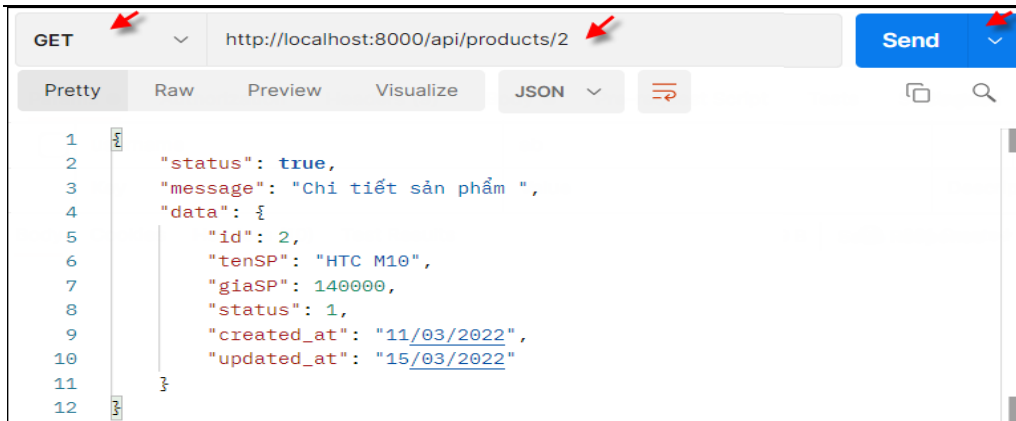


5. Trả về chi tiết 1 resource

- Code trong hàm **show** của ProductController:

```
public function show($id) {
    $product = Product::find($id);
    if (is_null($product)) {
        $arr = [ 'success' => false,
                'message' => 'Không có sản phẩm này',
                'data' => []
            ];
        return response()->json($arr, 200);
    }
    $arr = ['status' => true,
           'message' => "Chi tiết sản phẩm ",
           'data' => new ProductResource($product)
    ];
    return response()->json($arr, 201);
}
```

- Test với Postman, sẽ phải thấy dữ liệu đổ về



```

1  {
2    "status": true,
3    "message": "Chi tiết sản phẩm ",
4    "data": {
5      "id": 2,
6      "tenSP": "HTC M10",
7      "giaSP": 140000,
8      "status": 1,
9      "created_at": "11/03/2022",
10     "updated_at": "15/03/2022"
11   }
12 }

```

Bài 3:

Sinh viên làm tương tự với table loaisanpham (tên field tùy ý)

*** Yêu cầu nộp bài:

SV nén file (hoặc share thư mục google drive) bao gồm các yêu cầu đã thực hiện trên, nộp LMS đúng thời gian quy định của giảng viên. KHÔNG NỘP BÀI COI NHƯ KHÔNG CÓ ĐIỂM.

--- Hết ---