# Code Conventions

## Biến

* Quy tắc chung của tên biến là ở dạng **camelCase**
* Tên phải có nghĩa, không được đặt tên kiểu viết tắt. VD: uName, pName, idl, a, a1, doFA.
* Khi khai báo biến, một dòng chỉ chứa một biến.

**Good**

$articlesWithAuthor

**Bad**

$articles\_with\_author

* Tên của collection nên theo quy tắc Mô tả collection + danh từ sô nhiều

**Good**

$activeUsers = User::active()->get()

**Bad**

$users = User::active()->get()

$user = User::active()->get()

$User = User::active()->get()

* Tên 1 object nên theo quy tắc Mô tả collection + danh từ sô ít

**Good**

$activeUser = User::active()->first()

**Bad**

$users = User::active()->first()

## Hàm

* Quy tắc chung của tên hàm là ở dạng **camelCase**
* Tên hàm thường bắt đầu bằng động từ. VD: getUserName, setUserName, increaseDownloadCounter
* Tránh đặt những tên quá chung chung, tối nghĩa. VD: top, doIncrease, getAll.
* Hàm không nên quá 30 dòng.
* Một hàm không được vượt quá 5 tham số. (nên giữ <=3).
* Một hàm chỉ làm duy nhất 1 việc, trong trường hợp cần thiết, có thể cho phép làm 2 việc, tuy nhiên tên hàm phải nói rõ điều này. VD: increaseDownloadCounterAndSaveToDatabase.
* Các câu lệnh lồng nhau tối đa 4 cấp.

## Hằng số

* Đặt theo UPPER\_CASE. VD: CLICK\_COUNTER
* Khai báo ở đầu file, class

## Controller

* Tên controller phải bắt đầu bằng một danh từ
* Danh từ đó ở dạng số ít
* Theo sau đó là hậu tố "Controller"
* Tên Lớp trùng với tên file
* Nên dùng Resource Controllers

**Good**

**class** **ArticleController** **extends** **Controller**

//

{

**Bad**

**class** **ArticlesController** **extends** **Controller**

{

**class** **wp\_articlesController** **extends** **Controller**

{

**class** **Article** **extends** **Controller**

{

**Good**

**class** **DomainController** **extends** **Controller**

{

**public** **function** **index**(){} // list domains

**public** **function** **create**(){} // show create form

**public** **function** **store**(Request $request){ } // handle the form POST

**public** **function** **show**($id){} // show a single domain

**public** **function** **edit**($id){} // show edit page

**public** **function** **update**(Request $request, $id){} // handle show edit page POST

**public** **function** **destroy**($id){} // delete a domain

}

**Bad**

**class** **DomainController** **extends** **Controller**

{

**public** **function** **list**(){} // list domains

**public** **function** **create\_or\_save**(){} // show create form then handle save

**public** **function** **show\_edit**($id){} // show a single domain then show edit page

**public** **function** **delete**($id){} // delete a domain

}

## Models

* Dạng CamelCase
* Nên trùng tên table nếu đại diện cho table, chỉ thay snake\_case thành CamelCase

**Good**

**class** **ProductCategory** **extends** **Model**

{

//

}

**Bad**

**class** **Product\_Category** **extends** **Model**

{

**class** **product\_category** **extends** **Model**

{

* Phương thức định nghĩa mối quan hệ (relationship methods) hasOne hoặc belongsTo phải là danh từ số ít
* Các phương thức quan hệ khác (relationship methods) phải là dạng danh từ số nhiều
* **Nếu thuộc tính của model là field của table thì phải ở dạng snake\_case và giống với field của table cho dễ nhận biết và dễ tìm. Nếu không phải thì ở dạng camelCase**

**Good**

$user->created\_at

**Bad**

$user->createdAt

* Phương thức của model phải ở dạng camelCase
* Ưu tiên dùng Query Builder hơn là dùng raw query vì sau test, nâng cấp version dễ hơn
* Nếu dùng raw query thì phải log query

## Routers

* Router phải ở dạng số nhiều và các chữ cái đều là chữ thường
* Tên của router phải ở dạng snake\_case

**Good**

Route::get('/users', '[[email protected]](https://vncoder.vn/cdn-cgi/l/email-protection)');

Route::resource('photos', 'PhotoController');

**Bad**

Route::get('/user', '[[email protected]](https://vncoder.vn/cdn-cgi/l/email-protection)');

Route::get('/UsersList', '[[email protected]](https://vncoder.vn/cdn-cgi/l/email-protection)');

Route::resource('PHOTO', 'PhotoController');

## Helper

* Với các function tự viết (ví dụ isNull, firstNull….) mà không biết đặt vào đâu thì tạo 1 file helper và đặt vào đó
* Sử dụng autoloading của Composer để load

**Good**

project\_folder/app/helper.php

project\_folder/app/Http/helper.php

**Bad**

project\_folder/functions.php

**Good**

// file composer.json

...

"autoload": {

"files": [

"app/helpers.php"

],

...

**Bad**

// file app/Http/Controllers/HomeController.php

**class** **HomeController**.**php**

{

**function** **index**(){

**require\_once**(app\_path("helpers.php"));

}

}

## 1.8 View

* View đặt tên dạng snake\_case. Ví dụ

show\_filtered.blade.php

* Với các view phụ thuộc nhau thì view con lấy tiền tố là view cha

show\_filtered\_keyword.blade.php

* Không xử lý logic code trong file view mà viết phải trong controller ví dụ:

**Good**

// $api\_results is passed by controller

<ul>

@**foreach**($api\_results **as** $result)

<li>{{ $result->name }}</li>

@**endforeach**

</ul>

**Bad**

@php

$api\_results = json\_decode(file\_get\_contents("https://api.example.com"));

@endphp

<ul>

@**foreach**($api\_results **as** $result)

<li>{{ $result->name }}</li>

@**endforeach**

</ul>

# DB Conventions

## 2.1. Một số quy tắc chung về đặt tên

|  |  |  |  |
| --- | --- | --- | --- |
| Nên | Ví dụ | Không nên | Giải thích |
| Tên là danh từ tiếng Anh |  |  |  |
| Chỉ dùng danh từ số ít | inventory shelf octopus | inventories  octopodes  shelves octopuses octopi | - Comment là từ chỉ chung cho tất cả các comment chứ không phải comments.  - Đỡ phải nghĩ và tìm bảng dễ hơn |
| Chỉ dùng lower\_case | customer | Customer | - Viết cho dễ dọc  - Rất nhiều db không phân biệt hoa thường  - Đỡ thắc mắc khi nào thì có chữ hoa, khi nào thì có gạch dưới  - Tránh nhầm psychotherapist (nhà trị liệu tâm thần học) với PsychoTheRapist (kẻ hấp dâm tâm thần) |
| Chỉ dùng dấu gạch dưới \_ để nối các từ | first\_name | FirstName  firstName  "First Name" |
| Tên có tính tự giải thích. – Không viết tắt – Tránh dùng kiểu dữ liệu thay cho tên | middle\_name  blog.content amt | mid\_nm  blog .text amount | Dễ hiểu |
| Tránh dùng từ khóa của SQL | display\_order  updated\_at user\_name | order date  name | Có thể bị báo lỗi syntax nếu không enquote |
| Tên ngắn gọn, không nên dài quá 64 ký tự |  |  |  |

## 2.2. Table

|  |  |  |  |
| --- | --- | --- | --- |
| Nên | Ví dụ | Không nên | Giải thích |
| Đặt prefix cho các table liên quan đến nhau | catalog\_category catalog\_product |  | Tìm kiếm table dễ hơn |
| Thêm suffix \_tmp cho các table dùng tạm trong tính toán nhưng không xóa | catalog\_product\_price\_tmp |  |  |
| Thêm prefix tmp\_ cho các table dùng tạm, có thể xóa | tmp\_im\_calculating |  |  |

## 2.3. Column

### 2.3.1. Column

|  |  |  |  |
| --- | --- | --- | --- |
| Nên | Ví dụ | Không nên | Giải thích |
| Tránh thêm tiền tố không cần thiết | product.name | product.product\_name | – Tên table đã nêu rõ context |
| Thêm prefix is\_ cho các field dạng YES/NO | is\_active  is\_delivered is\_free\_shipping | active delivered  free\_shipping | Nhìn field là biết chỉ có 2 giá trị |
| Nên lưu các thời điểm thay đổi dữ liệu với từng record | created\_at updated\_at  deleted\_at |  | Theo dõi tính toàn vẹn dữ liệu |
| Với trường chứa dữ nhiều dữ liệu, dùng tên số nhiều và dữ liệu phân cách bằng dấu “,” | User\_id\_allows, dữ liệu dạng 1,2,3,4 |  |  |
| Không đặt tên chứa kiểu dữ liệu | return\_code | int\_return\_code | Kiểu dữ liệu có thể thay đổi: – date => timestamp – int => bigint |

### 2.3.2. Primary Key

|  |  |  |  |
| --- | --- | --- | --- |
| Nên | Ví dụ | Không nên | Giải thích |
| Chỉ nên dùng PK là id | table .id | table .table\_id | – Dễ nhớ – Giảm effort khi đổi tên table sau này |
| Mỗi table nên có 1 PK, bên cạnh các UNIQUE KEY khác | PRIMARY KEY (id), UNIQUE KEY idx\_unique (key1,key2) |  | Làm việc với các record nhanh hơn |
| PK mặc định nên dùng kiểu Interger, Auto-increment |  |  |  |

### 2.3.3. Foreign Keys

|  |  |  |
| --- | --- | --- |
| Nên | Không nên | Giải thích |
| Tên FK được kết hợp từ tên field và tên table mà nó tham chiếu tới | person\_id là FK của table person và field id | Dễ hiểu |
| Tùy chọn Cascading Update có thể dùng, nhưng Cascading Delete thì nên tránh |  | Giảm rủi ro khi lỡ xóa record ở main table khiến cho toàn bộ dữ liệu liên quan biến mất |

### 2.3.4. Indexes

|  |  |  |
| --- | --- | --- |
| Nên | Ví dụ | Giải thích |
| Thêm prefix idx\_ ở đầu | idx\_created\_at | Dễ nhớ, nhất là khi ALTER TABLE mà không dùng GUI Tool |