|  |  |  |
| --- | --- | --- |
| **HTTP VERB** | **PATH** | **Controller Action/method** |
| **GET** | **/Users** | **Index** |
| **GET** | **/Users/create** | **Create** |
| **POST** | **/Users** | **Store** |
| **GET** | **/Users/{id}** | **Show (individual record)** |
| **GET** | **/Users/{id}/edit** | **Edit** |
| **PUT** | **/Users/{id}** | **Update** |
| **DELETE** | **/Users/{id}** | **Destroy** |

**AJAX Technologies**

Như mô tả trước đó, ajax không phải là một công nghệ mà nhóm các công nghệ liên quan. Công nghệ AJAX bao gồm:

* HTML / XHTML và CSS
* DOM
* XML hoặc JSON
* XMLHttpRequest
* JavaScript

**HTML/XHTML and CSS**

Những công nghệ này được sử dụng để hiển thị nội dung và phong cách. Nó chủ yếu được sử dụng để trình bày.

**DOM**

Nó được sử dụng để hiển thị năng động và tương tác với dữ liệu.

**XML or JSON**

Để mang dữ liệu đến và đi từ máy chủ. JSON (JavaScript Object Notation) là như XML nhưng ngắn hơn và nhanh hơn so với XML.

**XMLHttpRequest**

Đối với giao tiếp không đồng bộ giữa máy khách và máy chủ. Đối với lần nữa trang tiếp theo.

**JavaScript**

Nó được sử dụng để mang lại công nghệ trên với nhau.

Độc lập, nó được sử dụng chủ yếu để xác nhận phía máy khách.

Chân thành cảm ơn các bạn đã quan tâm theo dõi bài viết.

## 1. Tạo Model trong laravel

Bài viết được đăng tại freetuts.net - thehalfheart@gmail.com

Tất cả Model của laravel được đặt trong thư mục app/models và khi tạo mới một model bạn cần tuân theo những quy tắc sau:

* Tên file phải đặt trùng với tên class
* Trong laravel có 2 tùy chọn Eloquent ORM và Query Builder. Nếu sử dụng **Eloquent**thì bắt buộc class phải extends \Illuminate\Database\Eloquent\Model hoặc Eloquent, còn sử dụng **Query Builder** thì không cần extends gì.

**Ví dụ: tạo User model với query builder:**

**Bước 1**: Tạo file User.php trong thư mục app/models.

**Bước 2**: Viết code trong file User.php như sau:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <?php    class User{      public static function all(){          return DB::table('users')->get();      }  } |

Vậy là ta đã tạo ra model User và phương thức all để lấy ra tất cả các user.

### Tạo model trong thư mục con (sub folder)

Laravel hỗ trợ rất tốt việc tạo các class trong sub folder, và cũng tương tự như việc tạo [controller](http://freetuts.net/su-dung-controller-trong-laravel-300.html) trong sub folder, sau khi tạo xong bắt buộc chạy lệnh sau (run command): php artisan dump-autoload.

Ví dụ tạo model Product.php trong thư mục demosubfolder, thì **bắt buộc** ngay sau khi tạo bạn phải chạy lệnh php artisan dump-autoload để laravel hiểu được model Product này.

## 2. Tương tác với cơ sở dữ liệu bằng Query Builder

Để tương tác với cơ sở dữ liệu bằng **query builder** thì laravel cung cấp cho ta class DB để làm việc đó.

**Cú pháp sử dụng**:

|  |  |
| --- | --- |
| 1 | DB::table($ten\_bang)->bieuthuc->laydulieu(); |

**Trong đó**:

* $ten\_bang là đặt bảng mà trong truy vấn sẽ truy vấn tới.
* bieuthuc là các biểu thức mà trong câu truy vấn sẽ thực hiện như: where, orWhere, orderBy, groupBy, ...
* laydulieu() là phương thức để thực thi câu truy vấn bao gồm các biểu thức trước đó, cơ bản nhất là get() ngoài ra còn có, first(), pluck(), lists(), ...

**Ví dụ**: tạo phương thức tìm ra user đầu tiên theo username => đầu vào là $username:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <?php    class User{        public static function getByUserName($username)      {          return DB::table('users')->where('username','=',$username)->first();      }  } |

Sử dụng: $user=User::getByUserName('freetuts'); thì ta sẽ nhận được $user tương ứng với username là **freetuts**(nếu có trong database).

Để biết các biểu thức trong query builder hỗ trợ, bạn chạy đoạn code sau và bạn sẽ thấy là có rất nhiều query được hỗ trợ:

|  |  |
| --- | --- |
| 1 | var\_dump(get\_class\_methods('\Illuminate\Database\Query\Builder')); |

Bây giờ ta sẽ tìm hiểu thêm một số biểu thức thường dùng trong Query Builder.

### Biểu thức Where

Biểu thức Where sẽ có các tham số truyền vào và tùy vào số lượng tham số đó mà Laravel sẽ có những cách xử lý khác nhau.

**where($column\_value)**: trường hợp này $column\_value sẽ là một mảng theo dạng $field => $value.

|  |  |
| --- | --- |
| 1  2  3  4 | $column\_value=[      'username'=>'freetuts',      'email'=>'thinhbuzz@freetuts.net'  ]; |

**where($column,$value)** <=> where column=value

|  |  |
| --- | --- |
| 1 | where('username','freetuts'); |

**where($column,$operator,$value)** <=> where column operator value

|  |  |
| --- | --- |
| 1  2 | where('id', '>', 13);  //sql: where id > 13 |

**->where()->orWhere()**

|  |  |
| --- | --- |
| 1  2 | where('username','freetuts')->orWhere('email','admin@freetuts.net');  //sql: where username='freetuts' or email='admin@freetuts.net' |

Vẫn còn các biểu thức tương tự như:

* whereIn
* whereNotIn($column,$array\_value)
* whereNull
* whereNotNull
* orWhereNull
* orWhereNotNull($column), ....

**Offset & Limit**:

|  |  |
| --- | --- |
| 1  2  3  4 | $users = DB::table('users')->skip(10)->take(5)->get();  //sql: select \* from `users` limit 5 offset 10  $users = DB::table('users')->offset(10)->take(5)->get();  //sql: select \* from `users` limit 5 offset 10 |

**Order By, Group By, và Having**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | $users = DB::table('users')                      ->orderBy('name', 'desc')                      ->groupBy('count')                      ->having('count', '>', 100)                      ->get(); |