HTTP REQUESTS

Request Trong Laravel là gì?

Request trong Laravel là một object chứa các thông tin liên quan đến HTTP request hiện tại. Dựa vào object này chúng ta có thể lấy được các thông tin như input, cookie, file,...

Class `Request` này nằm trong `Illuminate\Http\Request` core của Laravel. Object này được base trên `http-foundation` package của Symfony Laravel chỉ custom lại một chút và thêm một số phương thức.

Tương tác với Request

Mặc định `Request` object đã được binding vào service container của Laravel rồi, nên khi dùng nó thì có thể binding trực tiếp vào trong argument.

Inject trong Route

```
use Illuminate\Http\Request;

Route::get('/', function (Request $request) {
    //
});
```

Inject trong Controller

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class UserController extends Controller
{
    public function update(Request $request, $id) {
        //
    }
}
```

Lúc này ở route sẽ không cần phải khai báo param `$request` nữa.

```
use App\Http\Controllers\UserController;
```

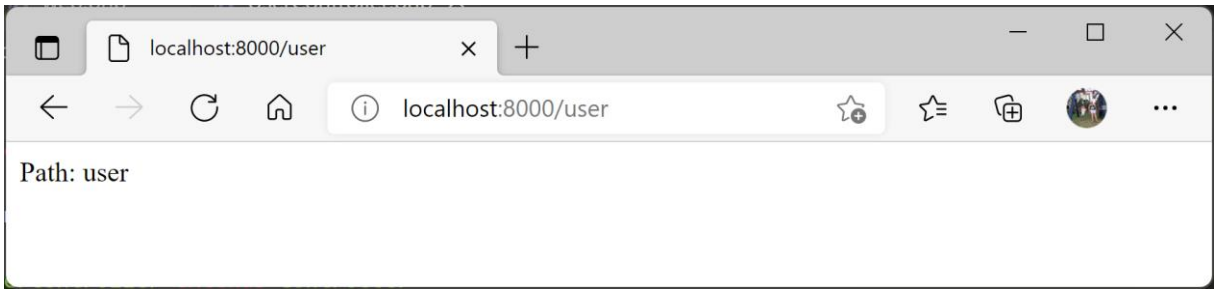
```
Route::put('/user/{id}', [UserController::class, 'update']);
```

Truy xuất thông tin PATH/URL

Để lấy path của request sử dụng method `path`.

```
$uri = $request->path();
```

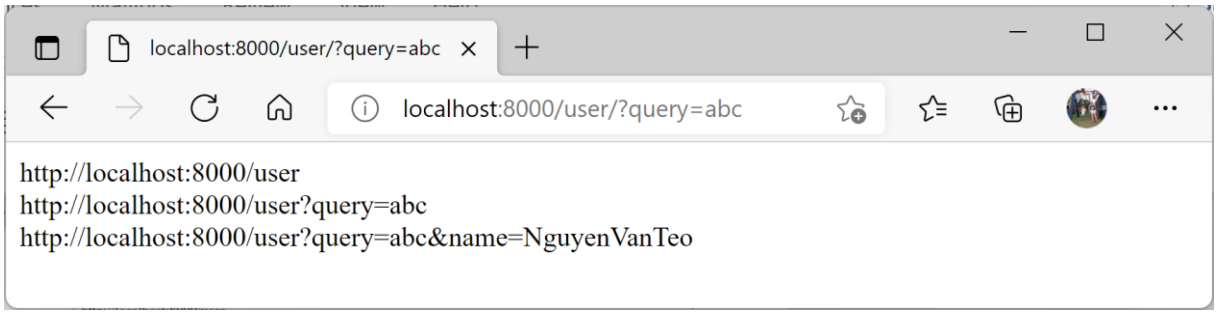
Kết quả:



Lấy URL của request sử dụng phương thức `url`, lấy full URL của request sử dụng phương thức `fullUrl`, muốn add thêm query vào trong path sử dụng phương thức `fullUrlWithQuery`.

```
$url = $request->url();
$urlWithQueryString = $request->fullUrl();
$request->fullUrlWithQuery(['name' => 'NguyenVanTeo']);
```

Kết quả:



Kiểm tra Request Path / Route

Để kiểm tra path hiện tại có match với một path rule nào đó hay không sử dụng phương thức `is`.

Kiểm tra xem path hiện tại có phải bắt đầu bằng `admin` hay không?

```
if ($request->is('admin/*')) {
    //
```

```
}
```

Sử dụng phương thức `routeIs` để kiểm tra tương tự như phương thức `is`, nhưng là check qua `route name`.

Kiểm tra xem path hiện tại có phải nằm trong route name bắt đầu bằng `admin` không?

```
if ($request->routeIs('admin.*')) {
    //
}
```

Lấy phương thức (Method) của Request

Để lấy ra phương thức (method) của request sử dụng phương thức `method`, có thể sử dụng phương thức `isMethod` để kiểm tra phương thức của request.

```
$method = $request->method();
if ($request->isMethod('post')) {
    //
}
```

Truy xuất thông tin Request Headers

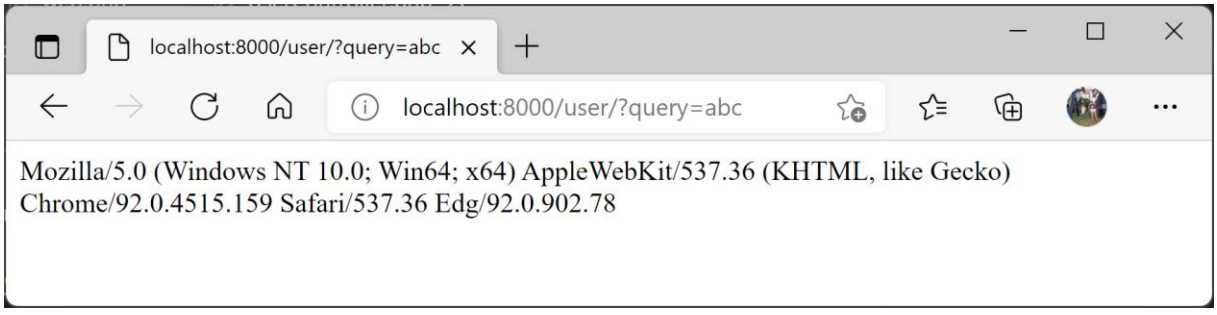
Nếu muốn truy vấn thông tin liên quan đến header của request sử dụng phương thức `header`.

```
$value = $request->header('X-Header-Name')
```

Lấy ra `user-agent` của request.

```
$request->header('user-agent');
```

Kết quả:



Trường hợp request header cần lấykhông tồn tại thì phương thức header sẽ trả về null. Muốn thay đổi giá trị default này có thể truyền thêm tham số thứ 2 vào hàm header.

```
$value = $request->header('X-Header-Name', 'default');
```

Cũng có thể lấy ra bearer token của request bằng cách sử dụng phương thức `bearerToken`.

```
$token = $request->bearerToken();
```

Cách viết trên tương tự với:

```
$token = $request->header('Authorization', '');
```

Request IP Address

Để lấy ra địa chỉ ip address của người dùng sử dụng phương thức `ip`.

```
$ipAddress = $request->ip();
```

Input

Truy xuất tất cả Input Data

Để nhận tất cả dữ liệu input gửi lên request sử dụng phương thức `all`.

```
$input = $request->all();
```

Truy xuất Input Value

Trong trường hợp muốn lấy ra dữ liệu của một input cụ thể (bao gồm payload và query string) sử dụng phương thức `input`.

```
$name = $request->input('name');
```

Mặc định, Laravel sẽ trả về `null` nếu input cần lấy không tồn tại. Có thể xác định giá trị mặc định cho input bằng cách truyền thêm tham số thứ 2 vào phương thức `input`.

```
$name = $request->input('name', 'Nguyen Van Teo');
```

Trong trường hợp dữ liệu cần truy xuất là một mảng thì có thể sử dụng "`.`" thay cho cách truy cập mảng.

```
$name = $request->input('products.0.name');
// tương đương product[0]['name']
```

```
$names = $request->input('products.*.name');
// tương đương với lấy hết ra name của products
```

Nếu không truyền tham số nào vào phương thức `input` thì Laravel sẽ trả về tất cả dữ liệu gửi lên request (tương tự như phương thức `all`).

```
$input = $request->input();
// Tương đương với
$input = $request->all();
```

Truy xuất Boolean Input Values

Trong một số trường hợp có sử dụng các thẻ HTML như checkbok, radio,... Và chỉ cần kiểm tra giá trị của nó là `true` hay `false` thôi, thì có thể sử dụng phương thức `boolean`. Phương thức này sẽ nhận các giá trị `1`, "`1`", `true`, "`true`", "`on`", và "`yes`" là `true`, còn ngược lại sẽ là `false`.

```
$archived = $request->boolean('archived');
```

Nhận dữ liệu từ Query String

Trong một số trường hợp chỉ muốn lấy ra query string của dữ liệu gửi lên request. Có thể sử dụng phương thức `query`. Hàm này sẽ trả về một mảng chứa các query string gửi lên request.

```
$query = $request->query();
```

Nếu cần lấy giá trị của một query string cụ thể, có thể truyền query string name vào phương thức `query`.

```
$name = $request->query('name');
```

Mặc định, Laravel sẽ trả về `null` nếu query string cần lấy không tồn tại. Có thể xác định giá trị mặc định cho input bằng cách truyền thêm tham số thứ 2 vào phương thức `query`.

```
$name = $request->query('name', 'Tran Van Ty');
```

Truy xuất dữ liệu thông qua Dynamic Properties

Laravel cũng có apply magic method vào trong Request, nên hoàn toàn có thể truy cập đến một input dưới dạng property trong object. Đối với cách này thì Laravel sẽ ưu tiên lấy ra giá trị

trong payload trước.

Lấy ra dữ liệu của input name.

```
$name = $request->name;
```

Lấy một phần dữ liệu đầu vào

Trường hợp cần lấy ra một số input nhất định nào đó nhưng không phải tất cả thì sử dụng phương thức `only`.

Chỉ lấy ra giá trị input username, password:

```
$input = $request->only(['username', 'password']);

$input = $request->only('username', 'password');
```

Hoặc ngược lại muốn bỏ qua input nào đó còn lại sẽ lấy hết thì sử dụng phương thức `except`.

Bỏ qua input credit_card còn lại lấy hết:

```
$input = $request->except(['credit_card']);

$input = $request->except('credit_card');
```

Xác định dữ liệu đầu vào

Có thể sử dụng phương thức `has` để xác định xem một giá trị có xuất hiện trong `request` hay không. Phương thức `has` trả về `true` nếu giá trị có trong `request` và ngược lại sẽ là `false`.

```
if ($request->has('name')) {
    //
}
```

Cũng có thể kiểm tra nhiều input với phương thức `has` bằng cách truyền vào một mảng các input. Lúc này phương thức sẽ trả về `true` nếu tất cả các input cần kiểm tra đều xuất hiện trong `request`, ngược lại nó sẽ trả về `false`.

```
if ($request->has(['name', 'email'])) {
    //
}
```

Cách viết trên tương tự với:

```
if ($request->has('name') && $request->has('email')) {
     //
}
```

Khi cần xử lý một số hành động, logic khi input xuất hiện trong `request` sử dụng hàm `whenHas`. Lúc này nếu trong `request` có input name thì callback function sẽ được thực thi.

```
$request->whenHas('name', function ($input) {
    //
});
```

Nếu cần kiểm tra một mảng các input và sẽ trả về `true` nếu có một hoặc nhiều input xuất hiện trong danh sách đó sử dụng phương thức `hasAny`.

```
if ($request->hasAny(['name', 'email'])) {
    //
}
```

Cách viết trên tương tự với:

```
if ($request->has('name') || $request->has('email')) {
    //
}
```

Khi cần kiểm tra thêm input nào đó có xuất hiện trong input và phải có giá trị hay không? Có thể sử dụng phương thức `filled`. Phương thức này sẽ trả về `true` nếu input có xuất hiện trong input và có giá trị kèm theo.

```
if ($request->filled('name')) {
    //
}
```

Tương tự, có thể dùng hàm `whenFilled` để thực thi logic khi input nào đó tồn tại và có giá trị.

```
$request->whenFilled('name', function ($input) {
    //
});
```

Khi cần check xem một input nào đó không xuất hiện trong request sử dụng phương thức `missing`. Phương thức này sẽ trả về `true` nếu input không xuất hiện trong request.

```
if ($request->missing('name')) {
    //
}
```

Old input

Mặc định, Laravel sẽ lưu trữ lại giá trị các input của request để phục vụ request tiếp theo như recover lại dữ liệu khi validate sai,... và quá trình này đã được làm tự động. Nhưng cũng có thể sử dụng một số phương thức trong trường hợp cần thiết.

Flashing Input To The Session

Phương thức `flash` trong class `Illuminate\Http\Request` sẽ lưu hết dữ liệu input vào session để nó có sẵn trong lần yêu cầu tiếp theo của người dùng đối với ứng dụng. Và sẽ được xóa đi nếu request lại lần tiếp theo.

```
$request->flash();
```

Có thể sử dụng các phương thức `flashOnly` để lưu một tập hợp con dữ liệu input được xác định vào session, còn lại sẽ bỏ qua, và `flashExcept` bỏ qua các input được xác định vào session, còn lại sẽ lưu trữ.Các phương pháp này hữu ích để giữ thông tin nhạy cảm như passwords ra khỏi phiên:

```
request->flashOnly(['username', 'email']);

$request->flashExcept('password');
```

Flashing Input Then Redirecting

Since you often will want to flash input to the session and then redirect to the previous page, you may easily chain input flashing onto a redirect using the `withInput` method:

```
return redirect('form')->withInput();

return redirect()->route('user.create')->withInput();

return redirect('form')->withInput(
```

```
    $request->except('password')
);
```

Truy xuất Old Input

Để lấy ra giá trị của input trước đó đã được store sử dụng phương thức `old`. Nếu dữ liệu không có giá trị phương thức sẽ trả về `null`.

```
$username = $request->old('username');
```

Hoặc có thể sử dụng hàm `old` trong helper.

```
<input type="text" name="username" value="{{ old('username') }}">
```

Retrieving Cookies From Requests

All cookies created by the Laravel framework are encrypted and signed with an authentication code, meaning they will be considered invalid if they have been changed by the client. To retrieve a cookie value from the request, use the `cookie` method on an `Illuminate\Http\Request` instance:

```
$value = $request->cookie('name');
```

Input Trimming & Normalization

Mặc định, Laravel sử dụng 2 middleware `App\Http\Middleware\TrimStrings` và `App\Http\Middleware\ConvertEmptyStringsToNull` để làm nhiệm vụ xử lí, chuẩn hóa dữ liệu trước khi đưa vào Request object.

- Middleware `TrimStrings` có tác dụng loại bỏ đi các khoảng trống (white space) ở các input gửi lên request.
- Middleware `ConvertEmptyStringsToNull` có tác dụng chuyển đổi string rỗng thành null.

Nếu muốn vô hiệu hóa nó, có thể remove 2 middleware ttong class `App\Http\Kernel`.

File

Truy xuất Files Uploaded

Có thể lấy ra được file upload lên trên request qua phương thức file. Phương thức này sẽ trả về một Object `UploadedFile` (`Illuminate\Http\UploadedFile`) nếu file đó tồn tại và null nếu file đó không tồn tại. Object `UploadedFile` này được kế thừa từ `SplFileInfo` mặc định của PHP, nên có thể sử dụng được các phương thức trong `SplFileInfo`.

Lấy ra thông tin file photo upload lên request.

```
$file = $request->file('photo');
```

Cũng có thể trỏ trực tiếp đến file name qua property.

```
$file = $request->photo;
```

Trong trường hợp cần kiểm tra xem một file nào đó có xuất hiện trong request hay không thì có thể sử dụng phương thức `hasFile`.

Kiểm tra xem request gửi lên có file photo không?

```
f ($request->hasFile('photo')) {
    //
}
```

Validating Successful Uploads

Có thể kiểm tra xem file được upload lên có thành công hay không. Bằng cách sử dụng phương thức `isValid`.

```
if ($request->file('photo')->isValid()) {
    //
}
```

File Paths & Extensions

The `UploadedFile` class also contains methods for accessing the file's fully-qualified path and its extension. The `extension` method will attempt to guess the file's extension based on its contents. This extension may be different from the extension that was supplied by the client:

```
$path = $request->photo->path();
```

```
$extension = $request->photo->extension();
```

Storing Uploaded Files

To store an uploaded file, you will typically use one of your configured filesystems. The `UploadedFile` class has a `store` method that will move an uploaded file to one of your disks, which may be a location on your local filesystem or a cloud storage location like Amazon S3.

The `store` method accepts the path where the file should be stored relative to the filesystem's configured root directory. This path should not contain a filename, since a unique ID will automatically be generated to serve as the filename.

The `store` method also accepts an optional second argument for the name of the disk that should be used to store the file. The method will return the path of the file relative to the disk's root:

```
$path = $request->photo->store('images');

$path = $request->photo->store('images', 's3');
```

If you do not want a filename to be automatically generated, you may use the `storeAs` method, which accepts the path, filename, and disk name as its arguments:

```
$path = $request->photo->storeAs('images', 'filename.jpg');

$path = $request->photo->storeAs('images', 'filename.jpg', 's3');
```