



---

# **Bài 8**

# **Session và Cookie**

Module: BOOTCAMP WEB-BACKEND DEVELOPMENT WITH  
LARAVEL 2.1

# Mục tiêu

---



- Trình bày được các cơ chế quản lý session trong laravel
- Thực hiện được các thao tác cơ bản với cookie
- Lưu trữ và truy xuất được dữ liệu từ session
- Quản lý được vòng đời của session
- Thay đổi được các cấu hình của session

---

# HTTP Session

Giới thiệu

Sử dụng Session

Thêm tùy biến Session Drivers

# Giới thiệu

---



- Cấu hình
- Driver Prerequisites

- Khi hệ thống HTTP không có chỗ lưu trữ, sessions cung cấp một cách để lưu thông tin từ các yêu cầu của người dùng.
- Laravel cung cấp đầy đủ session backends thông qua API để hỗ trợ việc này. Hỗ trợ các backend như Memcached, Redis, và cơ sở dữ liệu đã có sẵn.

- File cấu hình session lưu ở config/session.php .
- Hãy chắc rằng bạn nắm rõ tất cả các thông tin cấu hình của session trước khi tùy chỉnh lại tệp tin này.
- Mặc định, Laravel sẽ cấu hình sử dụng file session driver, nó sẽ hoạt động tốt cho nhiều ứng dụng.
- Đối với production, bạn có thể cân nhắc sử dụng memcached hoặc redis drivers để cho hiệu năng của session tốt hơn.

- Các session driver được định nghĩa là nơi lưu trữ dữ liệu session qua các request. Laravel đã tích hợp sẵn một số session driver sau:
  - `file` - sessions sẽ lưu tại `storage/framework/sessions`.
  - `cookie` - sessions sẽ lưu có bảo mật, mã hóa cookies.
  - `database` - sessions sẽ lưu trong cơ sở dữ liệu được dùng trong ứng dụng của bạn.
  - `memcached` / `redis` - sessions sẽ lưu và truy suất nhanh hơn, dựa trên cache.
  - `array` - sessions sẽ được lưu trong mảng PHP và sẽ tồn tại lâu.



Với array driver chỉ nên sử dụng khi chạy testing để có các dữ liệu tồn tại trong thời gian dài.

# Driver Prerequisites

---



- Cơ sở dữ liệu
- Redis



- Khi sử dụng database session driver, bạn cần phải tạo bảng chứa dữ liệu session trong cơ sở dữ liệu.
- Bên dưới là một ví dụ Schema dùng tạo bảng

```
Schema::create('sessions', function ($table) {  
    $table->string('id')->unique();  
    $table->integer('user_id')->nullable();  
    $table->string('ip_address', 45)->nullable();  
    $table->text('user_agent')->nullable();  
    $table->text('payload');  
    $table->integer('last_activity');  
});
```

# Cơ sở dữ liệu

---



- Bạn có thể sử dụng session:table trong Artisan command để tự động tạo migration:

```
php artisan session:table
```

```
php artisan migrate
```

# Redis

---



- Trước khi sử dụng Redis sessions với Laravel, bạn cần cài đặt gói predis/predis package (~1.0) qua Composer.
- Bạn cấu hình Redis của bạn kết nối trong file cấu hình database .
- Trong file cấu hình session , thuộc tính connection có thể được sử dụng để xác định kết nối với Redis là sử dụng session.



# Sử dụng Session

---

- Nhận dữ liệu
- Lưu dữ liệu
- Flash dữ liệu
- Xóa dữ liệu
- Regenerating The Session ID

# Nhận dữ liệu

---

- Có hai cách chính để làm việc với dữ liệu session data trong Laravel: phương thức global session và qua thể hiện Request .
- Đầu tiên, Chúng ta nhìn cách truy cập session qua thể hiện Request , có thể được type-hinted trong phương thức controller.
- Nhớ rằng, phương thức controller dependencies tự động injected qua Laravel service container:

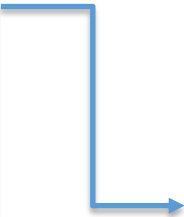
# Nhận dữ liệu



```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
```



```
class UserController extends Controller
{
    /**
     * Show the profile for the given user.
     *
     * @param Request $request
     * @param int $id
     * @return Response
     */
    public function show(Request $request, $id)
    {
        $value = $request->session()->get('key');

        //
    }
}
```

# Nhận dữ liệu

---

- Khi bạn nhận giá trị từ session, bạn cũng có thể truyền giá trị mặc định qua tham số thứ hai của phương thức get .
- Giá trị mặc định sẽ được trả về nếu key không tồn tại trong session.
- Nếu bạn truyền vào một Closure là giá trị mặc định của phương thức get và requested key không tồn tại, thì Closure sẽ được thực thi và trả về giá trị return:

```
$value = $request->session()->get('key', 'default');  
  
$value = $request->session()->get('key', function() {  
    return 'default';  
});
```

# Phương thức Global Session

---

- Bạn cũng có thể sử dụng hàm global session của PHP và lưu dữ liệu trong session.
- Khi hàm session được gọi, chuỗi tham số, nó sẽ trả về giá trị của key session.
- Khi hàm được gọi với một cặp giá trị key / value, giá trị sẽ lưu trong session



# Phương thức Global Session

```
Route::get('home', function () {  
    // Retrieve a piece of data from the session...  
    $value = session('key');  
  
    // Specifying a default value...  
    $value = session('key', 'default');  
  
    // Store a piece of data in the session...  
    session(['key' => 'value']);  
});
```



Có rất ít sự khác biệt giữa sử dụng session qua HTTP request và sử dụng hàm global `session`. Cả hai phương thức là testable qua phương thức `assertSessionHas` nó tồn tại trong tất cả các test cases của bạn.

# Nhận tất cả dữ liệu Session

---

- Nếu bạn muốn nhận tất cả dữ liệu của session, bạn có thể sử dụng phương thức all :

```
$data = $request->session()->all();
```

# Kiểm tra sự tồn tại của Session

---



- Để xác sự tồn tại session, bạn có thể sử dụng phương thức has.
- Phương thức sẽ trả về has true , nếu giá trị của session không bằng null :

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

# Kiểm tra sự tồn tại của Session



- Để xác sự tồn tại session, ngay cả giá trị của nó bằng null , bạn có thể sử dụng phương thức exists .
- Phương thức exists trả về true nếu giá trị tồn tại:

```
if ($request->session()->exists('users')) {  
    //  
}
```

- Để lưu dữ liệu trong session, bạn sẽ thường sử dụng phương thức put hoặc hàm session :

```
// Via a request instance...  
$request->session()->put('key', 'value');  
  
// Via the global helper...  
session(['key' => 'value']);
```

# Đẩy giá trị vào mảng Session

---



- Phương thức push có thể sử dụng để đẩy một giá trị vào một biến mảng session.
- Ví dụ, nếu trong user.teams là một mảng chứa tên nhóm, bạn có thể đẩy tên nhóm mới vào mảng theo cách sau:

```
$request->session()->push('user.teams', 'developers');
```

# Nhận & xóa một item

---

- Phương thức pull sẽ nhận và xóa một item từ session trong một lệnh

```
$value = $request->session()->pull('key', 'default');
```

- Phương thức forget sẽ xóa từng phần dữ liệu từ session. Nếu bạn muốn xóa tất cả dữ liệu session, bạn có thể sử dụng phương thức flush

```
$request->session()->forget('key');
```

```
$request->session()->flush();
```



# Regenerating The Session ID

---



- Regenerating the session ID thường gặp khi ngăn một mã độc từ người dùng khai thác một session fixation tấn công ứng dụng của bạn.
- Laravel tự động regenerates the session ID khi xác thực nếu bạn sử dụng built-in LoginController ; tuy nhiên, nếu bạn cần tự tay regenerate the session ID, bạn có thể sử dụng phương thức regenerate .

```
$request->session()->regenerate();
```

# Adding Custom Session Drivers

---



- Implementing The Driver
- Đăng ký The Driver

# Implementing The Driver

- Tùy biến session driver của bạn nên thực hiện trong SessionHandlerInterface . Nó chứa một vài phương thức chúng ta cần để thực thi. Một stubbed MongoDB thực hiện giống như bên dưới:

```
<?php

namespace App\Extensions;

class MongoHandler implements SessionHandlerInterface
{
    public function open($savePath, $sessionName) {}
    public function close() {}
    public function read($sessionId) {}
    public function write($sessionId, $data) {}
    public function destroy($sessionId) {}
    public function gc($lifetime) {}
}
```

# Implementing The Driver



Laravel không cung cấp đường dẫn chứa extensions của bạn. bạn có thể thoải mái đặt ở đâu bạn thích. Trong ví dụ trên, chúng ta sẽ tạo một đường dẫn `Extensions` chứa

`MongoHandler`.

- Mục đích của những phương thức này không thật sự khó hiểu, chúng ta sẽ tìm hiểu sơ về những phương thức đó:
  - Phương thức open thường được sử dụng trong các fiel. Từ khi Laravel cung cấp một file session driver, Bạn sẽ hầu như không cần dùng phương thức này.
  - Bạn có thể để nó như 1 empty stub. Nó chỉ đơn giản là thực tế của giao diện kém (chúng ta sẽ tìm hiểu nó sau) mà PHP yêu cầu thực hiện phương thức này.
  - Phương thức close , giống như phương thức open , cũng có thể thường bỏ qua. hầu hết các drivers, nó không cần thiết.

# Implementing The Driver

---



- Phương thức close , giống như phương thức open , cũng có thể thường bỏ qua. hầu hết các drivers, nó không cần thiết.
- Phương thức read sẽ trả về chuỗi các dữ liệu session liên quan đến kiểm soát của \$sessionId . Ở đây không phải làm bất kỳ serialization hoặc encoding khi nhận hoặc lưu dữ liệu session trong driver, Laravel sẽ làm serialization cho bạn.
- Phương thức write sẽ viết các giá trị chuỗi \$data liên quan tới \$sessionId đến một vài hệ thống persistent storage như MongoDB, Dynamo, etc. Một lần nữa, bạn không cần phải serialization - Laravel đã xử lý việc đó cho bạn.
- Phương thức destroy sẽ xóa dữ liệu session liên quan tới \$sessionId từ persistent storage.
- Phương thức gc sẽ xóa tất cả dữ liệu session cũ hơn giá trị mới \$lifetime , bằng đoạn UNIX timestamp. Đối với hệ thống tự hết hạn như Memcached và Redis, phương thức này có thể được bỏ trống.

# Đăng ký The Driver

---

- Khi driver của bạn được thực hiện, bạn đã sẵn sàng đăng ký nó với framework.
- Bổ sung thêm drivers vào session backend của Laravel, bạn có thể sử dụng phương thức extend trong Session facade.
- Bạn nên gọi phương thức extend từ phương thức boot của service provider.
- Bạn có thể làm điều này từ AppServiceProvider hoặc tạo mới một provider:

# Đăng ký The Driver



```
<?php
```

```
namespace App\Providers;
```

```
use App\Extensions\MongoSessionStore;
```

```
use Illuminate\Support\Facades\Session;
```

```
use Illuminate\Support\ServiceProvider;
```

```
class SessionServiceProvider extends ServiceProvider
{
    /**
     * Perform post-registration booting of services.
     *
     * @return void
     */
    public function boot()
    {
        Session::extend('mongo', function($app) {
            // Return implementation of SessionHandlerInterface...
            return new MongoSessionStore;
        });
    }

    /**
     * Register bindings in the container.
     *
     * @return void
     */
    public function register()
    {
        //
    }
}
```



# Đăng ký The Driver

---

- Khi session driver đã được đăng ký, bạn có thể sử dụng mongo driver trong file cấu hình config/session.php



---

# Hướng dẫn

- Hướng dẫn làm bài thực hành và bài tập
- Chuẩn bị bài tiếp: ***Web Service and RESTful***