**Day 3**

**Migration, Seeder, Cookie, Seasion, Helpers,**

**File Storage, Upload file**

1. **Migration**

[Tìm Hiểu Về Migration trong Laravel - Viblo](https://viblo.asia/p/tim-hieu-ve-migration-trong-laravel-bWrZn1MpKxw)

[Migration trong trong Laravel (viblo.asia)](https://viblo.asia/p/migration-trong-trong-laravel-5OXLAX5wJGr)

[Tìm hiểu về Migration trong Laravel. (viblo.asia)](https://viblo.asia/p/tim-hieu-ve-migration-trong-laravel-LzD5dgWEljY)

[Migration | Laravel | Hướng dẫn học | Học web chuẩn (hocwebchuan.com)](https://hocwebchuan.com/tutorial/laravel/laravel_migrate.php)

[Database: Migrations - Laravel 11.x - The PHP Framework For Web Artisans](https://laravel.com/docs/11.x/migrations#main-content)

[Cascading on update (and on delete) in migration - DEV Community](https://dev.to/rafaelfranca/cascading-on-update-and-on-delete-in-migration-2jad)

* Migration là gì?
* Migration được hiểu là một hệ thống quản lí phiên bản (giống như git) của cơ sở dữ liệu, giúp quản lí schema database. Ví dụ, khi đang làm việc nhóm, thành viên khác đã thêm cột vào schema database cục bộ của họ. Nếu mình muốn sự thay đổi đó cập nhập trong schema database của mình, ta sử dụng migration để sinh ra một schema database mới như đã định nghĩa trước đó.
* Migration cho phép định nghĩa các table trong CSDL, định nghĩa nội dung table, cập nhập thay đổi các table, các cột đã tồn tại bằng ngôn ngữ PHP. Có thể sử dụng trên nhiều loại CSDL khác nhau (SQL Server, mySQL, Postgres,...) mà không cần phải sửa lại code.
* Giúp share schema database khi làm việc nhóm. Chỉ cần chạy migration, có thể sinh ra một schema giống như đã định nghĩa.
* Điều kiện để chạy được migration thành công:
* Phải có kết nối với database
* Phải lưu trữ migrations muốn sử dụng trong thư mục App/database/migrations.
* Tạo Migration: chạy lệnh dưới, mỗi file migration được được tên gồm timestamp + tên migration để xác định thứ tự.

**Php artisan make:migration <tên\_migration> --create=<tên\_table> :tạo migration và nó có tạo table mới**

**Php artisan make:migration <tên\_migration> --table=<tên\_table> : taok migration và nó sẽ không tạo table mới, dùng trong việc thay đổi cột.**

* Nếu khi chạy lệnh dưới bị lỗi, trước khi chạy hãy chạy trước câu lệnh: composer dump-autoload
* Cấu trúc file Migration: được lưu trữ trong thư mục App/database/migrations

****

* Trong Migration trên hình, sẽ có 2 function up() và down()
* up(): khi ta thực thi lệnh php artisan migrate, thì up() sẽ chạy, lúc đó 1 table flights sẽ được sinh ra
* down(): khi ta thực thi lệnh php artisan migrate:rollback, thì down() sẽ chạy, lúc đó table flights sẽ bị xóa.
* Nếu khi ta migration với 1 loại CSDL khác với loại cơ sở dữ liệu mà mình đang sử dụng, ta nên đặc thuộc tính cho migration. Vd dưới nghĩa là ta đang sử dụng CSDL PostgreSQL

A blue background with white text

Description automatically generated

* Thực thi Migration

|  |  |
| --- | --- |
| **php artisan migrate** | Thực thi file migration |
| **php artisan migrate:status** | Xem quá trình thay đổi migrate từ trước đến nay |
| **php artisan migrate --pretend** | Xem các câu lệnh SQL sẽ được thực thi như thế nào bởi migration mà không chạy migration |
| **php artisan migrate --isolated** | Khi đang deploy trên nhiều server, ta không muốn các server đó cùng migrate database cùng một lúc |
| **php artisan migrate --forced** | Một số migration có tính chất phá hoại, có thể khiến mất dữ liệu. Để khỏi chạy các lệnh này trên production database. |
| **php artisan migrate:rollback** | Quay lai trạng thái trước đó |
| **php artisan migrate:rollback –step=<số bước>** | Quay lại với số bước nhất định |
| **php artisan migrate:rollback –batch=<số bước>** |  |
| **php artisan migrate:reset** | Để hoàn tác các migration đã làm rollback trước đó |
| **php artisan migrate:refresh** | Hoàn tác, tạo lại database và chạy migration |
| **php artisan migrate:refresh --seed** |  |
| **php artisan migrate:refresh –step=<số bước>** |  |
| **php artisan migrate:fresh** | Xóa tất cả các bảng trong database và migration lại. |
| **php artisan migrate:fresh --seed** |  |
| **php artisan migrate:fresh –database=admin** |  |

* Tao bảng

**A close-up of a computer code

Description automatically generated**

* Kiểm tra xem table hay cột có tồn tại hay không

**A screenshot of a computer code

Description automatically generated**

**A computer screen shot of a message

Description automatically generated**

* Đổi tên hoặc xóa table

**A white background with black text

Description automatically generated**

* Tạo cột

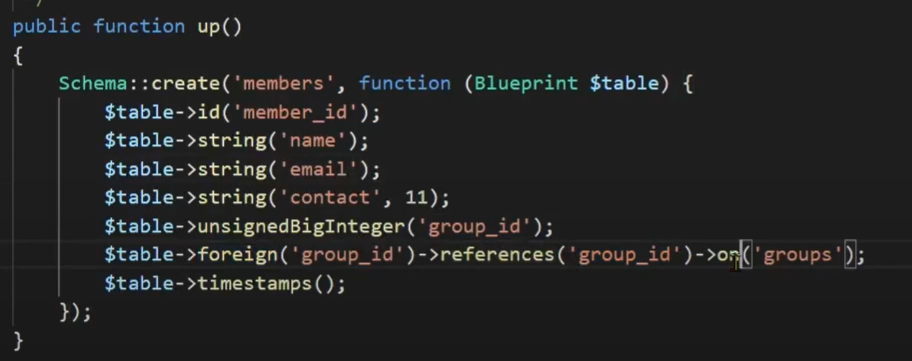
**A computer screen shot of a function

Description automatically generated**

* Tạo type cho cột: [Tìm hiểu về Migration trong Laravel. (viblo.asia)](https://viblo.asia/p/tim-hieu-ve-migration-trong-laravel-LzD5dgWEljY)

| **Cú pháp** | **Mô tả** |
| --- | --- |
| **$table->id()** | Tăng ID(primary key) sử dụng như unsigned integer |
| **$table->bigIncrements('id');** | Tăng ID (primary key) sư dụng như "UNSIGNED BIG INTEGER". |
| **$table->bigInteger('votes');** | Tương đương với BIGINT. |
| **$table->binary('data');** | Tương đương với BLOB. |
| **$table->boolean('confirmed');** | Tương đương với BOOLEAN. |
| **$table->char('name', 4);** | Tương đương với CHAR với độ dài cho trước. |
| **$table->date('created\_at');** | Tương đương với DATE. |
| **$table->dateTime('created\_at', 0);** | Cột tương đương với dữ liệu chính xác (tổng chữ số). |
| **$table->dateTimeTz('created\_at', 0);** | Cột tương đương DATETIME (với múi giờ) với độ chính xác (tổng chữ số). |
| **$table->decimal('amount', 8, 2);** | Cột tương đương DECIMAL với độ chính xác (tổng chữ số) và tỷ lệ (chữ số thập phân). |
| **$table->double('amount', 8, 2);** | Cột tương đương NHÂN ĐÔI với độ chính xác (tổng chữ số) và tỷ lệ (chữ số thập phân). |
| **$table->enum('level', ['easy', 'hard']);** | Cột tương đương ENUM. |
| **$table->float('amount', 8, 2);** | Cột tương đương FLOAT với độ chính xác (tổng chữ số) và tỷ lệ (chữ số thập phân). |
| **$table->geometry('positions');** | Cột tương đương GEOMETRY. |
| **$table->geometryCollection('positions');** | Cột tương đương GEOMETRYCOLLMENT. |
| **$table->increments('id');** | Cột tự động tăng tương đương cột không tương đương (khóa chính). |
| **$table->integer('votes');** | Cột tương đương INTEGER. |
| **$table->ipAddress('visitor');** | Địa chỉ IP cột tương đương. |
| **$table->json('options');** | Cột tương đương JSON. |
| **$table->jsonb('options');** | Cột tương đương JSONB. |
| **$table->lineString('positions');** | LINE LINEING cột tương đương. |
| **$table->longText('description');** | Cột tương đương LONGTEXT. |
| **$table->macAddress('device');** | Cột địa chỉ MAC tương đương. |
| **$table->mediumIncrements('id');** | Cột tự động tăng tương đương cột MEDIUMINT (khóa chính) không tương tự. |
| **$table->mediumInteger('votes');** | Cột tương đương MEDIUMINT. |
| **$table->mediumText('description');** | Cột tương đương MEDIUMTEXT. |
| **$table->morphs('taggable');** | Thêm các cột tương đương taggable\_idBIGINT và taggable\_typeVARCHAR. |
| **$table->uuidMorphs('taggable');** | Thêm các cột tương đương taggable\_idCHAR (36) và taggable\_typeVARCHAR (255) UUID. |
| **$table->multiLineString('positions');** | MULTILINESTRING cột tương đương. |
| **$table->multiPoint('positions');** | Cột tương đương MULTIPOINT. |
| **$table->multiPolygon('positions');** | Cột tương đương MULTIPOLYGON. |
| **$table->nullableMorphs('taggable');** | Thêm các phiên bản nullable của cột. |
| **$table->nullableUuidMorphs('taggable');** | Thêm các phiên bản nullable của cột. |
| **$table->nullableTimestamps(0);** | Bí danh của phương pháp. |
| **$table->point('position');** | ĐIỂM cột tương đương. |
| **$table->polygon('positions');** | Cột tương đương POLYGON. |
| **$table->rememberToken();** | Thêm một remember\_tokencột tương đương VARCHAR (100) không thể rỗng . |
| **$table->set('flavors', ['strawberry', 'vanilla']);** | SET cột tương đương. |
| **$table->smallIncrements('id');** | Cột tương đương tự động tăng không giới hạn SMALLINT (khóa chính). |
| **$table->smallInteger('votes');** | Cột tương đương SMALLINT. |
| **$table->softDeletes('deleted\_at', 0);** | Thêm một deleted\_atcột tương đương TIMESTAMP không thể xóa để xóa mềm với độ chính xác (tổng số chữ số). |
| **$table->softDeletesTz('deleted\_at', 0);** | Thêm một deleted\_atcột tương đương TIMESTAMP (có múi giờ) không thể xóa để xóa mềm với độ chính xác (tổng chữ số). |
| **$table->string('name', 100);** | Cột tương đương VARCHAR với chiều dài. |
| **$table->text('description');** | Cột tương đương. |
| **$table->time('sunrise', 0);** | Cột tương đương TIME với độ chính xác (tổng chữ số) |
| **$table->timeTz('sunrise', 0);** | Cột tương đương TIME (với múi giờ) với độ chính xác (tổng chữ số). |
| **$table->timestamp('added\_on', 0);** | Cột tương đương TIMESTAMP với độ chính xác (tổng chữ số). |
| **$table->timestampTz('added\_on', 0);** | Cột tương đương TIMESTAMP (với múi giờ) với độ chính xác (tổng chữ số). |
| **$table->timestamps(0);** | Thêm các cột tương đương nullable created\_atvà updated\_atTIMESTAMP với độ chính xác (tổng chữ số). |
| **$table->timestampsTz(0);** | Thêm các cột tương đương nullable created\_atvà updated\_atTIMESTAMP (với múi giờ) với độ chính xác (tổng chữ số). |
| **$table->tinyIncrements('id');** | Tự động tăng cột tương đương TINYINT (khóa chính) không được ký. |
| **$table->tinyInteger('votes');** | Cột tương đương TINYINT. |
| **$table->unsignedBigInteger('votes');** | Cột tương đương BIGINT chưa ký. |
| **$table->unsignedDecimal('amount', 8, 2);** | Cột tương đương KHAI THÁC KHÔNG GIỚI HẠN với độ chính xác (tổng chữ số) và tỷ lệ (chữ số thập phân). |
| **$table->unsignedInteger('votes');** | Cột tương đương INTEGER INTEGER. |
| **$table->unsignedMediumInteger('votes');** | Cột tương đương MEDIUMINT. |
| **$table->unsignedSmallInteger('votes');** | Cột tương đương KHÔNG NHỎ. |
| **$table->unsignedTinyInteger('votes');** | Cột tương đương TINYINT chưa ký. |
| **$table->uuid('id');** | Cột tương đương UUID. |
| **$table->year('birth\_year');** | Cột tương đương NĂM. |

* Modify Column
* Modify Table
* Tạo rang buộc khóa ngoại cho table:
* Viết trực tiếp trong schema::create của table cần liên kết khóa ngoại.



Trên vd trên, lệnh dùng để liên kết khóa ngoại như sau:

**$table ->Foreign(‘<*tên column cần liên kết*>) ->reference(‘<*tên column của table nguồn*)->on(‘<*tên table nguồn*>)**

* Tạo 1 file migration mới, viết liên kết khóa ngoại trong Schema:table

**A screen shot of a computer code

Description automatically generated**

**A computer screen shot of text

Description automatically generated**

1. **Seeder**

[Tìm hiểu về Seeder trong Laravel (viblo.asia)](https://viblo.asia/p/tim-hieu-ve-seeder-trong-laravel-bWrZn1MmKxw)

[Database: Seeding - Laravel 11.x - The PHP Framework For Web Artisans](https://laravel.com/docs/11.x/seeding#running-seeders)

[Seeder trong Laravel 8 (toidicode.com)](https://toidicode.com/seeder-trong-laravel-8-467.html)

[Tạo dữ liệu bằng cách sử dụng Seeder trong Laravel (igotocode.com)](https://igotocode.com/vi/creating-data-using-seeder-in-laravel)

[Bài 6: Tạo dữ liệu mẫu với Seeding - Học lập trình Laravel (vncoder.vn)](https://vncoder.vn/bai-hoc/tao-du-lieu-mau-voi-seeding-6)

* Seeder là gì
* Seeder là một class cho phép tạo (là dữ liệu mẫu dummy data hoặc dữ liệu giả) và chèn dữ liệu vào cơ sở dữ liệu
* Thay vì nhập dữ liệu thủ công, ta có thể sử dụng seeder để nhanh chóng tạo dữ liệu và phát triển tính năng của ứng dụng
* Ta viết code Query Builder hoặc dùng Eloquent Model trong Seeder
* Cấu trúc thư mục của Seeder
* Tất cả seed class được chứa ở thư mục database/seeds
* Mặc định của Laravel đã cung sẵn 1 file DatabaseSeeder.php

A screen shot of a computer

Description automatically generated

* Function run(): Một Class Seed mặc định chỉ chứa duy nhất 1 phương thức run().
* Trong run(), ta sử dụng Query Buider để làm việc với database

VD:

A computer code with text

Description automatically generated with medium confidence

* Run(): cũng cho phép call tới seed class khác, cho phép control được thứ tự thêm dữ liệu. Có thể call 1 lần nhiêu seed class khác.

VD

A screen shot of a computer code

Description automatically generated

* Để tạo 1 file Seed: sử dụng lệnh dưới trong terminal

**Php artisan make:seeder <tên Seeder>**

* Insert
* Thực thi Seeder:
* Để thực thi các seeder, sử dụng câu lệnh sau

**Php artisan db:seed --class=<tên Seeder> : thực thi 1 seeder cụ thể,**

**Để thực thi 2 hoặc nhiều seeder, thì trong function run() của DatabaseSeeder, ta call các seeder kia rồi thực thi seeder mặc định DatabaseSeeder. Chú ý, các seeder sẽ thực thi theo thứ tự.**

**Hoặc**

**Php artisan db:seed : thực thi seeder mặc định DatabaseSeeder**

* Để xóa tất cả dữ liệu trước và tạo mới lại dữ liệu trong database

**Php artisan migrate:fresh --seed : thực thi 1 seeder cụ thể**

**Hoặc**

**Php artisan migrate:fresh --seed --seeder=<tên seeder> : thực thi seeder mặc định DatabaseSeeder**

* Sử dụng tham số --force ở cuối lệnh thực thi seeder để bỏ qua thông báo xác nhận (bypass prompt) khi chạy seeder trên môi trường production
* Roolback (quay lại): để hoàn tác batch Seeder cuối cùng, ta sử dụng tham số --reverse ở cuối lệnh thực thi seeder
* Model Factories (chưa xong)
* Thay vì insert thủ công (như hình dưới) là phải thêm từng record cho một bảng bằng tay. Khi ta muốn insert một lượng dữ liệu lớn, ta không thể dùng thủ công như thế, bây giờ ta sẽ sử dụng Model Facories.

A computer code with text

Description automatically generated

* Tron
* Muting model event: tắt kích hoạt của các sự kiện trên model trong quá trình seeding dữ liệu vào CSDL.
* Khi sử dụng use WithoutModelEvents , các sự kiện như created, updated, deleted,... sẽ không được kích hoạt trong quá trình seeding trong phạm vi seeder đó. Giúp tăng tốc độ quá trình seeding, tránh gửi các thông báo hoặc thực thi các hành động không cần thiết trong quá trình seeding dữ liệu

A screen shot of a computer program

Description automatically generated

1. **Cookie, Session**

* **Cookie**
* Cookie là gì?
* Là một phương thức dùng để lưu trữ thông tin tạm thời trên trình duyệt của người dung
* Cookie đươc tạo bởi server và được gửi đến trình duyệt, sau đó, trình duyệt sẽ lưu trữ cookie trên máy tính của client. Khi client truy cập lại trang web, trình duyệt sẽ gửi cookie đến server để server có thể đọc thông tin lưu trữ trong cookie đó.
* Laravel cùng cấp sẵn các phương thứ sẵn để quản lí cookie, dùng để tạo, truy cập và xóa cookie. Cookie được lưu trữ trong đối tượng **Illuminate\Http\Response** hoặc **Illuminate\Http\RedirectResponse**
* Cookie dùng để lưu trữ thông tin đăng nhập, cài đặt người dùng, các giá trị phiên, lịch sử truy cập trang web,…
* Cookie có thể được sử dụng để theo dõi người dùng, thu thập thông tin riêng tư, vì thế, khi sử dụng cookie nên cẩn thận và chỉ nên lưu trữ những thông tin cần thiết.
* Trong Laravel, cookie được mã hóa để đảm bảo tính bảo mật, và chỉ được gửi qua HTTPS.
* Tạo Cookie, sử dụng **Illuminate\Http\Response.**
* Cookie(): Truyền vào 2 tham số: tên cookie và giá trị cookie

A close up of text

Description automatically generated

* Dùng tham số thứ 3 truyền vào phương thức cookie để thiết lập thời gian sống của cookie. Được tính bằng giây.

A close-up of a computer screen

Description automatically generated

* Dùng mảng thứ 4 để thiết lập các thuộc tính khác của cookie, ví dụ như chỉ được truyền qua HTTPS

A computer screen shot of a computer code

Description automatically generated with medium confidence

* Để xóa cookie trong Laravel, có 2 cách
* Thiết lập thời gian sống hết hạn: cần phải trả về một đối tượng response từ ứng dụng của mình. Dùng phương thức withCookie() để thiết lập cookie với thời gian sống đã hết hạn

A close-up of a computer code

Description automatically generated

* Truyền vào cookie một thời gian đã hết hạn, và thiết lập nó như một cookie mới thay thế cookie cũ. Trong ví dụ dưới, cookie tên name được thiết lập với giá trị trống và thời gian sống đã hết hạn, do đó cookie này sẽ bị xóa khỏi trình duyệt web người dùng.

A close-up of a computer screen

Description automatically generated

* **Session**

[HTTP Session - Laravel 11.x - The PHP Framework For Web Artisans](https://laravel.com/docs/11.x/session#main-content)

[Tìm hiểu về Session trong laravel (viblo.asia)](https://viblo.asia/p/tim-hieu-ve-session-trong-laravel-yMnKMgaAl7P)

[Session trong Laravel - Học Laravel (hoclaravel.vn)](https://hoclaravel.vn/chi-tiet/session-trong-laravel)

[Cách tạo Session và cookie trong Laravel dễ dàng như ăn cháo (freetuts.net)](https://freetuts.net/session-va-cookie-trong-laravel-5650.html)

[Local Storage, Session Storage và Cookies | Ông Dev | Techlog (youtube.com)](https://www.youtube.com/watch?v=9dloQfxucAc)

* Session là gì??
* Session là một tính năng dùng để lưu trữ các dữ liệu tạm thời của ứng dụng web trên phiên làm việc. Session sẽ bị xóa khi server bị reset hoặc khi các tệp session được xóa.
* Khi người dùng tương tác với ứng dụng web, các dữ liệu được lưu trữ trong session sẽ được lưu giữ tạm thời và có thể try cập và sử dụng trên các trang hoặc ứng dụng khác.
* Các biến của session được lưu trữ trong 1 phiên làm việc, được xác định bằng 1 cookie duy nhất trên trình duyệt của người dùng. Các giá trị session có thể được truy cập, sửa đổi, và xóa trên nhiều trang và nhiều request HTTP khác nhau.
* Session rất hữu ích trong việc lưu trữ các các thông tin như thông tin đăng nhật, thông tin giỏ hàng, cài đặt ngôn ngữ, tùy chọn người dùng. Ngoài ra còn giúp ứng dụng web bảo mật hơn vì các dữ liệu đưuọc lưu trữ trên server, đảm bảo rằng các thông tin riêng tư của người dùng không bị thay đổi hoặc xóa bởi người dùng.
* Laravel sử dụng API để quản lí session, trong session, mỗi drive được sử dụng để định nghĩa nơi dữ liệu session được lưu trữ.
* Cấu hình
* Thông tin cấu hình của session được chứa tại cofig/session.php.
* Laravel có các drive session sau
* File: Dữ liệu session được lưu trữ trong file mã hóa đặt tại storage/framework/session. Thông thường drive file sẽ được sử dụng vì nó nhe.
* Cookie: Dữ liệu session được lưu trữ trong cookie mã hóa được an toàn và bảo mật
* Database: dữ liệu session được lưu trữ trong database của ứng dụng. Session sẽ được lưu trữ vĩnh viễn. Để sử dụng được drive database ta phải cần tạo thêm table session. Có 2 cách tạo table session:
* Sử dụng make:session-table trong Artisan command để tự tạo migration

A blue background with white text

Description automatically generated

* Cách 2: đầu tiên tạo table trong thư mục database/migrations bằng lệnh **php artisan make:migration creat\_sessions\_table.**

sau đó sửa lại function up() thành như sau.

A screen shot of a computer code

Description automatically generated

Cuối cùng chạy lệnh **php artisan migrate**

* Apc: dữ liệu session được lưu trữ trong APC
* Memcached/Redis: dữ liệu session được lưu trữ và truy suất nhanh hơn, dựa trên cache. Session sẽ được lưu trữ vĩnh viễn. Trước khi dùng Redis drive cần phải cài gói predis/predis package (~1.0) thông qua composer
* Dynamodb: dữ liệu session được lưu trữ lại AWS DynamoDB
* Array: dữ liệu session sẽ được lưu trữ trong mảng của php, và tồn tại rất lâu. Chỉ nên sử dụng array drive khi chạy file tests để có các dữ liệu tồn tại trong thời gian dài.
* Tạo session: khai báo thư viện Session và dùng phương thức put()

A close-up of a computer code

Description automatically generated

* Truy cập session: (để sử dụng các phương thức sau, cần đẩm bảo cookie đươc bật trên trình duyệt người dùng vì laravel sẽ sử dụng cookie để lưu trữ session ID trên trình duyệt của người dùng.
* Get(): lấy giá trị của 1 session. Nếu session không tồn tại, get() sẽ trả về null, để trả về giá trị mặc định nếu session không tồn tại, ta sử dụng tham số thứ 2

A close-up of a computer screen

Description automatically generated

A close-up of a computer screen

Description automatically generated

* Push(): thêm một giá trị vào mảng session.

A close-up of a computer screen

Description automatically generated

* All(): lấy tất cả giá trị của session

A black text with black letters

Description automatically generated

* Has(): kiểm tra session có tồn tại hay không

A close-up of a computer code

Description automatically generated

* Pull(): Kiểm tra và lấy giá trị của một session nếu nó tồn tại

A black text on a white background

Description automatically generated

* Forget(): Xóa session

A close up of a text

Description automatically generated

* Flush(): xóa tất cả các giá trị của session

A close up of a computer screen

Description automatically generated

* Lưu trữ session tạm thời: dùng session flash để lưu trữ. Session flash sẽ chỉ tồn tại trọng 1 request duy nhất sau đó sẽ bị xóa. Thường dùng để lưu trữ thông báo hoặc trạng thái tạm thời để hiển thị cho người dùng. Dùng phương thức flash()

A white background with black and green text

Description automatically generated

* Để truy cập các giá trị session flash, dùng phương thức get() hoặc all()

A close-up of a message

Description automatically generated

* Session flat sẽ chỉ tồn tại trong 1 request duy nhất, nên nếu muốn lưu trưc các giá trị session dài hạn, dùng phương thức put() và push().
* Reflash():Để giữa lại tất cả giá trị flash session cho request tiếp theo

A close up of a white background

Description automatically generated

* Keep(): giữ lại một số giá trị flash session cho request tiếp theo

A close-up of a computer code

Description automatically generated

* Lưu trữ session vĩnh viễn: vì session được lưu trữ dưới dạng file trên server mặc định nên các session sẽ bị xóa khi server bị reset hoặc khi các têp session được xóa. Để lưu trữ session vĩnh viễn, dùng drive database hoặc redis.
* **So sánh cookie và session**

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **cookie** | **session** |
| Mục đích sử dụng | Dùng để lưu trữ dữ liệu dài hạng | Lưu trữ dữ liệu người dùng trong một phiên làm việc |
| Độ bảo mật | Bảo mật kém hơn, vì có thể bị tấn công bởi XSS (cross-site scripting) CSRF (Cross-Site Request Forgery) | Có độ bảo mật cao hơn, vì dữ liệu session được lưu trữ trên máy chủ và chỉ được tru cập thông tin qua 1 session ID được mã hóa. |
| Độ lớn | Có giới hạn dung lượng nhất định, thường là vài KB | Lưu trữ được nhiều dữ liệu hơn, vì ohiene làm việc của người dùng có thể kéo dài trong một khoảng thời gian dài |
| Cách lưu trữ | Lưu trữ trong máy tính của client. | Được lưu trữ treeb server. Do đó có thể được sử dụng để lưu trữ thông tin mật hoặc nhạy cảm hơn. |

1. **Helpers, File Storage**

* **Helpers**

[Helpers - Laravel 11.x - The PHP Framework For Web Artisans](https://laravel.com/docs/11.x/helpers#main-content)

* Helpers là gì?
* Helpers là các hàm toàn cục giúp thực hiện các tác vụ phổ biến như làm việc với mảng, chuỗi, đường dẫn và dữ liệu. Các hàm này đã có sẵn để sử dụng trong Laravel.
* Các hàm hỗ trợ có thể tự tạo hoặc sử dụng sẵn có để tương tác với Controller, View, và các Model trong dự án Laravel.
* Tạo thư mục helpers và helperSerrviceProvider: Để tạo 1 helper, chỉ cần tạo thư mục helpers trong thư mục app. Có thể sử dụng lệnh cmd hoặc tạo thủ công. Sau khi đã tạo helper, ở register ta sẽ chèn đoạn mã dưới để gọi các helper ta đã tạo trước đó trong thư mục helpers.

****

**A close-up of a computer screen

Description automatically generated**

* Đăng kí helper: để sử dụng helper, cần đăng kí helperServiceProvider.php trong cofig/app.php và chèn mã cần thiết để gọi các helper từ thư mục helpers.

**A screenshot of a computer program

Description automatically generated**

* Tạo helper: sau khi đã tạo thư mục Helpers trong app, ta sẽ tạo file helper. Ví dụ ta sẽ tạo 1 file helper khai báo hàm hello() tên CustomHelper.php

**A screen shot of a computer

Description automatically generated**

* Để có thể sử dụng được lệnh helper trên, ta sẽ gọi hàm hello(). Helper sẽ sử dụng được ở bất kì đâu trong project.
* Một số helper phổ biến

|  |  |
| --- | --- |
| **Mảng và đối tượng** | **Arr::add, Arr::collapse, Arr::divide, Arr::dot, Arr::except, Arr::first, Arr::flatten, Arr::get, Arr::has, Arr::set, Arr::sort, data\_get, data\_set.**  **Chuỗi: Str::contains,** |
| **Chuỗi** | **Str::endsWith, Str::startsWith, Str::slug, Str::snake, Str::camel, Str::kebab, Str::limit, Str::lower, Str::upper, Str::title, Str::singular, Str::plural.** |
| **Đường dẫn** | **app\_path, base\_path, config\_path, database\_path, public\_path, resource\_path, storage\_path** |
| **URLs** | **Action, asset, route, url** |
| **Một số hàm khác** | **abort, abort\_if, auth, bcrypt, cache, config, cookie, csrf\_field, dd, encrypt, event, info, logger, old, redirect, response, session, view** |

* **File storage**

1. **Upload File**

A diagram of a computer system

Description automatically generated