Laravel Complete Cheatsheet! Everything You Need To Know

Laravel Cheatsheet

Table of contents for this Cheat sheet:

A list of operators on Eloquent's where() method

When you use Eloquent's `where()` method, by default it will use the `=` operator (i.e. `->where('fieldname','value')` will generate ``... WHERE `fieldname` = 'value'....``

However you are not limited to just `=`. You can do something like this:

```php
$posts = Post::where("id","!=",123)->get()
```

Any of the following may be used as the second parameter (and use the third param for the value)

`=`, `<`, `>`, `<=`, `>=`, `<>`, `!=`, `LIKE`, `NOT LIKE`, `BETWEEN`, `ILIKE`

(ILIKE = case insensitive like, I believe this is just for Postgre)

VIEW MORE DETAILS (AND 12 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How do Invokable Controllers work in Laravel?

If a controller in Laravel has just one action then you simplify the controller by using the `__invoke()` method. Create a controller with a `__invoke()` method, then in your routes files you can just do the following:

```
Route::get('home','HomeController');
```

The controller code for HomeController could look something like this:

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class HomeController extends Controller
{
    public function __invoke(Request $request)
    {
        return "Welcome to our homepage"
    }
}
```

Create the controller using this artisan command:

```
$ php artisan make:controller HomeController --invokable
```

VIEW MORE DETAILS (AND 9 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to access the Laravel's route parameters outside of the controller?

If you want to read the route params (that are normally passed to the controller) from somewhere else, you can use `Route::current()->parameters()`.

For example:

```
$request_params = Route::current()->parameters();
```

VIEW MORE DETAILS (AND 5 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to add a name prefix to all routes in a group?

You can use Route Name Prefixes. If you wanted to have a group of Laravel routes all with names like 'admin.users', 'admin.news', etc then you can use a route name prefix to prepend 'admin.' to the start of all of the names

```
Route::name('admin.')->group(function () {
    Route::get('users', function () {
        })->name('users'); // Route will have the name "admin.users"...

});
```

However I do not like doing this, as it means I can't easily search for a particular route. But it does have the advantage of enforcing all in a group to have the same naming structure.

VIEW MORE DETAILS (AND 5 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to add the CSRF (Cross-site request forgery) token in Laravel?

While there are a few ways to do this, the easiest way for a normal `<form>` is to use (inside the form).

BTW, make sure you have the VerifyCsrfToken middleware enabled to make use of the CSRF protection!

VIEW MORE DETAILS (AND 9 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to change the order that migrations happen?

Rename the migration file, as it is sorted alphabetically

For example if you change `database/migrations/2014_10_12_000000_create_users_table.php` to something like `database/migrations/2018_10_11_000000_create_users_table.php` then it will be the most recent one processed when doing `php artisan migrate`.

VIEW MORE DETAILS (AND 6 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to check if a (blade) view file exists

Checking if a blade view file exists is very easy.

```
    if (\View::exists('some.view')) { ... }
```

Or if in a blade view:

```
@if(\View::exists('some.view'))
 ...
@endif
```

Some related things to be aware of:

`@includeIf` - include a view from within blade, if the view exists

```
@includeIf('view.name', ['some' => 'data'])
```

`@includeFirst` - Include the first view that exists (the final one would be the default shown if no others exist). In this example it would try and see if `custom.admin` existed (/resources/views/custom/admin.blade.php), but if it didn't then it would try `admin`.

```
@includeFirst(['custom.admin', 'admin'], ['some' => 'data'])
```

`@includeWhen` - Used to include a view when a boolean condition is true

VIEW MORE DETAILS (AND 3 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to check if a relation was loaded on an Eloquent model already?

Checking if a relation was loaded (maybe by a `with()` function call when using the Eloquent query builder) is easy.

From inside your model

If you want to do this within your model, you can use `$this->relationLoaded()`:

```
// from inside your model:
if ($this->relationLoaded("comments")) {
        //..
}
```

From outside of your model

```
$user = User::first();
//
if ($user->relationLoaded("comments")) {
  //
}
```

VIEW MORE DETAILS (AND 7 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to check if a user is logged in, inside a blade file?

You can use the `@auth` directive.

```
@auth
            Hello, logged in user!
@endauth
```

Related: `@guest` to check if user is a guest (not logged in)

```
@guest
            Hello guest, do you want to register?
@endguest
```

VIEW MORE DETAILS (AND 5 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to do comments in Blade files?

Comments in Blade are very simple!

You can either do normal PHP comments:

```
<? /* some comment here */
// or single line comments
# or these :)
?>
```

Or blade comments:

```
{{-- Blade comments that wil not appear in the rendered HTML output--}}
```

If you use an IDE that supports blade formatting (such as PHPStorm) then you can get it to output blade comments in the same style as normal comments.

VIEW MORE DETAILS (AND 11 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to eager load multiple levels of relationships in Eloquent?

You can use the 'dot notation' to eager load multiple levels of relationships in Eloquent

```
$posts = Posts::with("comments.author")->get();
```

This will load the comments relationship, and load the authors for each comment. Of course this assumes that you have set up the relationships correctly in your models

VIEW MORE DETAILS (AND 10 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to exclude URLs from the Laravel CSRF protection?

Sometimes you have to disable the CSRF protection. A common reason for this is because you have an incoming API webhook.

You can exclude URLs from the CSRF middleware by editing `app/Http/Middleware/VerifyCsrfToken`.

Edit the `$except` property.

```
protected $except = [
  "incoming/api/*",
];
```

VIEW MORE DETAILS (AND 5 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to find an item by Primary Key in Eloquent, or throw a ModelNotFoundException exception if it isn't found?

Use Eloquent's `findOrFail($id)` method - this will find a model by its primary key or throw an exception.

```
        $user_id=123;
$user = User::findOrFail($user_id);
// if it cannot find a row with $user_id as the primary key, it will throw a \Illuminate\Database\Eloquent\ModelNotFoundException\ModelNotFoundException exception
```

If you don't want to throw an exception if none was found then you should just use `find()` (which will return null if nothing is found.

VIEW MORE DETAILS (AND 6 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to find out what version of Laravel you are using?

1) The artisan command: `php artisan --version`

2) You can also find it in the file vendor/laravel/framework/src/Illuminate/Foundation/Application.php

```
class Application extends Container implements ApplicationContract, HttpKernelInterface
{
    /**
     * The Laravel framework version.
     *
     * @var string
     */
    const VERSION = '5.6.29';
```

VIEW MORE DETAILS (AND 9 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to find the average value of items in a Laravel Collection?

Use the `avg()` method. You can provide a value indicating which key to average (first example) or not (2nd example).

```
$average = collect(
        [
                ['foo' => 10],
                ['foo' => 10],
                ['foo' => 20],
                ['foo' => 40]
        ]
        )->avg('foo');

// 20

$average = collect(
        [1, 1, 2, 4]
        )->avg();

// 2
```

VIEW MORE DETAILS (AND 5 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to fix the MySQL related "Specified key was too long error" error in Laravel

This is a common problem (especially if you use a local development environment such as XAMPP).

The error will look something like this:

Sample output

```
[Illuminate\Database\QueryException] SQLSTATE[42000]: Syntax error or access violation: 1071 Speci
ed key was too long; max key length is 767 bytes

 [PDOException] SQLSTATE[42000]: Syntax error or access violation: 1071 Specified key was too long
max key length is 767 bytes
```

Luckily the fix is simple. Add the following to the boot() method of your app/Providers/AppServiceProvider.php file:

```
Schema::defaultStringLength(191);
```

VIEW MORE DETAILS (AND 3 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to generate an array for a dropdown (with ['key' => 'value']) suitable for a dropdown, from a Laravel collection of Eloquent objects?

The answer used to be `list()`, but now the answer is `pluck()`

```
$posts = Post::all();

    echo Form::select("post_id", $posts->pluck("title","id"));

    // $posts->pluck("title","id") will produce something like this:
    // [
    //   1 => "post title 1",
    //   5 => "Another post",
    // ...
    // ]
```

The first param is the actual value you want to 'pluck', and the second value is how you want it keyed. For most cases the second param will be 'id'.

You can also use `pluck` with just 1 param like this:

```
$collection = collect([
    ['product_id' => 'prod-100', 'name' => 'Desk'],
    ['product_id' => 'prod-200', 'name' => 'Chair'],
]);

$plucked = $collection->pluck('name');

$plucked->all();

// ['Desk', 'Chair']
```

VIEW MORE DETAILS (AND 11 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to get a route to directly show a view?

If you have a simple controller method that is just:

```
public function about_me() {
            return view('static.about_me');
            }
```

Then you can do the following in your routes file:

```
Route::view('/about','static.about_me');
```

This makes it a bit quicker to code. Sometimes it is common to end up with a bunch of functions (in a controller named something like StaticPagesController) that don't do anything apart from return a view. Using the `Route::view` method in your routes file means you can skip making that function.

VIEW MORE DETAILS (AND 8 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to get table column names from a database table in Eloquent?

There are a few methods to get the table column info in Eloquent

```
\Schema::getColumnListing($this->table)
```

```
class User extends Model {
    public function getTableColumns() {
        return $this
            ->getConnection()
            ->getSchemaBuilder()
            ->getColumnListing($this->getTable());
    }
}
```

VIEW MORE DETAILS (AND 11 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to get the first row that matches some where queries, or create it if it doesn't exist (in Laravel's Eloquent)?

You can use the very useful `firstOrCreate()` to try and find a row, and create it if it doesn't exist.

```
// Retrieve flight by name, or create it if it doesn't exist...
$flight = App\Flight::firstOrCreate(['name' => 'Flight 10']);

// Retrieve flight by name, or create it with the name and delayed attributes...
$flight = App\Flight::firstOrCreate(
    ['name' => 'Flight 10'], ['delayed' => 1]
);

// Retrieve by name, or instantiate...
$flight = App\Flight::firstOrNew(['name' => 'Flight 10']);

// Retrieve by name, or instantiate with the name and delayed attributes...
$flight = App\Flight::firstOrNew(
    ['name' => 'Flight 10'], ['delayed' => 1]
);
```

There is also the `firstOrNew` which works in the same way, however it will provide an unsaved object (i.e. you will have the model object returned from the method call, but it won't be saved in the DB). There is also the `updateOrCreate` if you need to update models, or create it if none exist.

VIEW MORE DETAILS (AND 10 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to group a SQL statement in brackets/parenthesis

It is very easy to chain `where()` methods together in Laravel like this:

```
$posts = Posts::where("category",1)->where("published",true)->where("active",1)->get()
```

But this will create a long list of `AND WHERE` statements (You can also use orWhere for OR WHERE).

But what if you need to group your where statements in brackets (parenthesis)?

If you want something like this (with one part of the WHERE grouped by brackets (parenthesis):

```
SELECT *
FROM `orders`
WHERE
    category = 1

    AND (
        featured = 1
        OR
        discounted IS NOT NULL
     )
```

Then you will have to do something called parameter grouping

VIEW MORE DETAILS (AND 3 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to increment a value in the database

You can just use the `::increment` methods to 'plus one' a value. You can also send how many to increment in the 2nd param.

```
\App\User::findOrFail($user_id)->increment("profile_view_count");
// or to increment by 5
\App\User::findOrFail($user_id)->increment("profile_score_count",5);
```

You can also of course use the ::decrement method too.

If you use this as part of the query builder you will face some problems chaining this. It will return an int, not the query builder object. You have to do it like this:

```
$query = \App\User::where("id",1);

$query->decrement("profile_view_count",5);
$query->increment("profile_rating",1);


// this will result in an error:
 \App\User::where("id",1)->decrement("profile_view_count",5)->increment("profile_rating",1);

 //although you could do it with just one:
 $num = \App\User::where("id",1)->decrement("profile_view_count",5);
```

VIEW MORE DETAILS (AND 3 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to make multiple routes share the same URL structure or URL parameters

If you have the following URLs:

```
/control-panel/{team_id}/
/control-panel/{team_id}/schedule
/control-panel/{team_id}/delete
        etc...
```

(They all share the first part of the url - `/control-panel/{team_id}`)

Then you don't have to write out the routes like this:

```
Route::get("control-panel/{team_id}","TeamController@index");
Route::get("control-panel/{team_id}/schedule","TeamController@schedule");
Route::delete("control-panel/{team_id}/delete","TeamController@destroy");
```

Instead you can **group the routes and add a prefix**:

```
Route::group(["prefix"=>"control-panel/{team_id}"], function() {
    Route::get("/","TeamController@index");
    Route::get("schedule","TeamController@schedule");
    Route::delete("delete","TeamController@destroy");
});
```

Putting a group within a group

VIEW MORE DETAILS (AND 9 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to namespace a Laravel route group?

You can add a namespace for all items within a route group. If your controller (or controllers) are in `app/Http/Controllers/AdminPanel` then you could use the following code in your routes file (web.php):

```
Route::group(["namespace"=>"AdminPanel"],function() {
    Route::get("/","AdminController@index"); // actually calls \App\Http\Controllers\AdminPanel\AdminControllers because of the namespace
});
```

You can use relative (relative to the normal controller namespace - \App\Http\Controllers) or absolute (\App\Admin\Controllers\)

VIEW MORE DETAILS (AND 12 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to order a Laravel hasMany relationship?

You can simply add a `->orderBy()` to the hasMany relationship

```
return $this->hasMany(Post::class)->orderBy('updated_at');
```

VIEW MORE DETAILS (AND 5 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to prevent Eloquent from adding created_at or updated_at timestamps?

If you want to disable both `created_at` and `updated_at` then add this to your model:

```
const UPDATED_AT = null;
const CREATED_AT = null;
```

If you want to disable just one of those fields (i.e. turn off created_at but keep updated_at, or keep created_at but disable the updated_at field) then just add the one line as required.

Also remember to remove this from the migrations:

```
$table->timestamps()
```

Note: There is also the `DELETED_AT` const, if you `use SoftDeletes`.

VIEW MORE DETAILS (AND 10 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to provide a default model object for relationships?

If you have a relationship (such as a belongsTo()) relationship, it might not always return a result.

This will then mean that every time you do something like this:

```
$post = Post::first();
$author = $post->author; // what if the relationship in author() returns nothing?
echo $author->name; // $author might have been null, so this will cause an error
```

There are three main options to handling this:

1) wrap everything in `optional()` (which will let you do anything to that variable and it will return null if there was nothing set).

2) Wrap everything in `if ($author) { ... }` to make sure $author isn't a null value

3) or set a default model. If nothing is found for a relationship then it will create an empty object (of the same type as the relationship defines - in the case below it will be \App\User).

```
/**
 * Get the author of the post.
 */
public function user()
{
    return $this->belongsTo('App\User')->withDefault();
}

// you can also define some defaults for the default!
```

VIEW MORE DETAILS (AND 6 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to return an item in Eloquent, or throw a ModelNotFoundException exception if it isn't found?

Use Eloquent's `firstOrFail()`. This is similar to `findOrFail()`, however `findOrFail()` does a 'WHERE id = ?' (assuming id is the primary key). With `firstOrFail()` it will return the first row based on whatever WHERE statements you have.

```
 $user = User::where("email","test@example.com"->firstOrFail()
```

If you don't nt to throw an exception if none was found then you should just use `first()` (which will return null if nothing is found.

VIEW MORE DETAILS (AND 12 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to return JSON data to the user in Laravel?

The correct way to do it in Laravel is returning `response()->json(...)` such as:

```
    return response()->json(['foo'=>'bar']);
```

If you need to return a JSONP response (with a callback function), then you can use `->withCallback($callback_function_name)`, such as:

```
return response()
        ->json(['name' => 'Abigail', 'state' => 'CA'])
        ->withCallback($request->input('callback'));
```

VIEW MORE DETAILS (AND 10 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to run an artisan command from a controller

Apart from within another command, I am not really sure I can think of a good reason to do this. But if you really want to call a Laravel command from a controller (or model, etc.) then you can use `Artisan::call()`

```
Artisan::call('email:send', [
        'user' => 1, '--queue' => 'default'
    ]);
```

One interesting feature that I wasn't aware of until I just Googled this to get the right syntax is `Artisan::queue()`, which will process the command in the background (by your queue workers):

```
Route::get('/foo', function () {
    Artisan::queue('email:send', [
        'user' => 1, '--queue' => 'default'
    ]);

    //
});
```

If you are calling a command from within another command you don't have to use the `Artisan::call` method - you can just do something like this:

```
public function handle()
{
    $this->call('email:send', [
        'user' => 1, '--queue' => 'default'
    ]);
```

VIEW MORE DETAILS (AND 4 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to see all of the options for an artisan command?

Just enter `--help` to the end of a command to see its help.

```
$ php artisan make:controller --help
```

Sample output

```
Description:
  Create a new controller class

Usage:
  make:controller [options] [--] <name>

Arguments:
  name                       The name of the class

Options:
  -m, --model[=MODEL]    Generate a resource controller for the given model.
  -r, --resource         Generate a resource controller class.
  -i, --invokable        Generate a single method, invokable controller class.
  -p, --parent[=PARENT]  Generate a nested resource controller class.
```

VIEW MORE DETAILS (AND 12 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to select all rows for a certain month (or day, year or time), using Eloquent

Eloquent makes many things easy, and working with dates is one of them!

There are bunch of built in functions for building a query that work with dates and times:

- whereDate
- whereMonth
- whereDay
- whereYear
- whereTime

You can use them like this:

You can use the `whereDate()` method to compare a column's value against a date:

```
$users = DB::table('users')
        ->whereDate('created_at', '2019-11-31')
        ->get();
```

The `whereMonth()` method can be used for comparing a column's value against a specific month of a year:

```
$users = DB::table('users')
        ->whereMonth('created_at', '10')
```

VIEW MORE DETAILS (AND 7 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to select what columns to return when calling ::all()

You can provide an array of column names to returns, such as:

```
$posts = Posts::all(["post_name","id"]);
```

VIEW MORE DETAILS (AND 6 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to show (or log) all SQL queries executed by Laravel

There are a few ways to show the SQL queries that are generated and executed by Laravel

`toSql()` - You can replace `->get()` (or `->paginate()`) with `->toSql()` to return a string of the SQL query that was generated

```
$posts = Posts::where("type",$type)->get();
// or as SQL statement
$postsSql =  Posts::where("type",$type)->toSql();
```

But maybe you want to see all of the queries that Laravel executes for each request?

Query Logging - you can enable the query log, then somewhere else loop through each logged query:

```
// put this somewhere...
DB::connection()->enableQueryLog();
// run some queries
Posts::where("title","test")->get();
User::find(1);

// then loop through this:
$queries = DB::getQueryLog();
```

Or add a **event listener for each query** - Put this in your AppServiceProvider (or another file, such as your routes file or even in a controller)

```
Event::listen('illuminate.query', function($query)
```

VIEW MORE DETAILS (AND 10 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to show a specific view file for certain error codes?

You can create files such as `resources/views/errors/$some_error_code.blade.php`

For example if you make `resources/views/errors/402.blade.php`, then whenever a 402 error is to be shown it will load that view and show it to the user.

VIEW MORE DETAILS (AND 6 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to undelete a soft deleted Eloquent row?

Use the `restore()` method.

```
User::withTrashed()->where("id",1)->restore()
```

VIEW MORE DETAILS (AND 8 DISCUSSIONS ABOUT THIS TOPIC) HERE...

How to use subdomains as parameters in your routes (or in your web.php file)

You can use subdomains in your routes. You can either have certain routes for certain subdomains, or pass the subdomain as a parameter in the routing.

```
Route::domain('{account}.example.com')->group(function () {
    Route::get('user/{id}', function ($account, $id) {
        //
    });
});
```

VIEW MORE DETAILS (AND 3 DISCUSSIONS ABOUT THIS TOPIC) HERE...

What are macros in Laravel?

Macros are a quite clever, but simple way to add methods to existing classes. Many of Laravel's built in classes, such as the Request facade, can have user defined macros added. Let me explain by a simple example.

To enable macros, you just have to add `use \Illuminate\Support\Traits\Macroable;` in the class to use the Macroable trait. You can add it to any class. Many of the built in classes that come with Laravel will already use this trait.

A simple example to explain and show how to use macros in Laravel

I think a simple example from scratch should explain things quickly:

```
class SomeClass
{
   use \Illuminate\Support\Traits\Macroable;
   protected $some_var='aaaaa';
}

SomeClass::macro("getSomeVar",function() {
   return $this->some_var;
});

$a = new SomeClass;
dump($a->getSomeVar());
```

Obviously, without the macro, there would be no way from anything outside of the class itself to access the `$some_var` property. But after adding the `getSomeVar` macro, you can access it.

This isn't a particularly useful macro... but it serves the point of showing how macros are set and how to use them

VIEW MORE DETAILS (AND 7 DISCUSSIONS ABOUT THIS TOPIC) HERE...

What are route patterns in Laravel?

In your routes file you can add code like this:

```
   Route::pattern('id', '[0-9]+');
```

Then you can use that pattern later on in your routes file like this:

```
Route::get('user/{id}', function ($id) {
   // Only executed if {id} is numeric...
});
```

I've not seen this used too often, however this is a useful feature and keeps things quite clean.

Some other examples of defining some route patterns:

```
Route::pattern('slug', '[a-z0-9-]+');
Route::pattern('hash', '[a-z0-9]+');

Route::get('forum-thread/{slug}', 'ForumController@viewThread'); // {slug} has to be alpha numeric (lowercase), but can include a dash
Route::get('forum-post/{hash}', 'ForumController@hash'); // {hash} has to be alpha numeric, lowercase
```

VIEW MORE DETAILS (AND 11 DISCUSSIONS ABOUT THIS TOPIC) HERE...

What is the loop variable in a @foreach loop in Blade?

If you have the following code in a Blade file:

```
@foreach(\User::all() as $user)

{{-- ... some code here ... --}}

@endforeach
```

You have access to `$user` (obviously) but also the `$loop` object.

The `$loop` object has the following data:

```
object(stdClass)#290 (8) {
  ["iteration"]=> int(1) // starts at 1 (on first loop)
   ["index"]=> int(0) // starts at 0 (on first loop)
   ["remaining"]=> int(4)
   ["count"]=> int(5)
   ["first"]=> bool(true)
   ["last"]=> bool(false)
   ["depth"]=>int(1)
   ["parent"]=>NULL
}
```

So if you wanted to do something different for the first or last rows then you could easily do this:

```
@foreach(\App\User::all() as $user)
```

VIEW MORE DETAILS (AND 12 DISCUSSIONS ABOUT THIS TOPIC) HERE...

What kind of database field type should you (normally) use for foreign keys?

For most cases you will want to use `unsignedInteger` for foreign keys. The actual answer is that the foreign key should be the same type as the primary key on the main table, which is normally an unsignedInteger.

Even if it isn't an actual foreign key you should be using this for anything that relates to another unsignedInteger. It is marginally quicker than just using integer.

```
$table->unsignedInteger('user_id');
$table->foreign('user_id')->references('id')->on('users');
```

VIEW MORE DETAILS (AND 7 DISCUSSIONS ABOUT THIS TOPIC) HERE...

Why should you update APP_URL from from http://localhost?

Your site will work fine without updating it - until you start using jobs and commands. When viewed in the browser (such as http://example.com, http://localhost:8000) Laravel can work out what your actual URL is. However, when you start sending emails, running commands and more it won't know what is your actual URL and will revert back to whatever is set in APP_URL (`config('app.url')`). So make sure you keep it correct, even if it seems to work ok with localhost.

VIEW MORE DETAILS (AND 6 DISCUSSIONS ABOUT THIS TOPIC) HERE...

Search for something

Search and hit Enter...

WebDevEtc

my web development blog

Welcome to my web dev blog! I'm a 29 year old guy from London, UK. I've loved programming since I was around 9 years old, and have run websites since being an early teenager.

For the last 12 or so years I've been working **full time as a backend web developer (mostly with PHP)**. This is my site where I write up about various web development topics (including Laravel stuff, some JS articles, Vim tips, server stuff, etc.)

I've got a few public Laravel packages here too.

Blog Post Categories

General	5
WebDev Programming	17
PHP	16
PHP Design Patterns	12
MySQL / MariaDB	3
Laravel	3
HTML	2
CSS	2
Vim Tricks and Tips	6
Server Stuff	5
Security and netsec	2
Bash and Linux CLI commands	2
PHPStorm	1

My Recent Blog Posts

Laravel features you may not know about

6 months ago

Suspending in Bash: Quickly switch between running processes in bash

6 months ago

The Repository Pattern in PHP (and Laravel)

8 months ago

Laravel Naming Conventions

8 months ago

Vimattack: How to get someone's database credentials while they are editing config files on a live

server

8 months ago

WebDev Tricks

As well as normal blog posts, I also have a bunch of pages about the following topics

- Apache Tricks
- CentOS and Linux Stuff Tricks
- Composer Tricks
- General Webdev and Programming Stuff Tricks
- JavaScript Tricks
- Laravel Tricks
- PHP Tricks
- WordPress Tricks

WebDevEtc web dev blog

Welcome to my site about all topics related to webdev (website development), but mostly focusing on PHP and Laravel.

I mostly work with PHP/Laravel nowadays - my time is split between working for some UK firms (handling their website stuff and coding their web based projects), and my own personal projects, sites and services.

Read More About Me here.

If you need some web development work done (especially anything with Laravel) please get in touch.

Navigation

- Home
- Sitemap
- Contact me
- About
- Laravel Blog
- Package

Contact me

Please find my contact details on the contact page...

SEND ME A MESSAGE

Follow Me:

Request id: Y7ZNPH