Laravel Cheat Sheet

Autor: Iván E. Tinajero Díaz

COMPOSER

Crear proyecto Laravel

- composer create-project --prefer-dist laravel/laravel blog
- composer create-project --prefer-dist laravel/laravel blog **"5.3.*"**

Actualizar dependencias (composer.json)

- composer update

PHP ARTISAN

Iniciar Web Server Development

- php artisan serve
  http://localhost:8000/

Ver comandos disponibles de artisan

- php artisan list

Ver ayuda de un comando de artisan

- php artisan help **route:list**

ROUTING

Archivo de rutas (Laravel 5.4)

- routes/web.php

Ruta ROOT (default)

```
Route::get('/', function () {
    return view('welcome');
});
```

Ruta básica (función php anónima)

```
Route::get('hello', function () {
    return "Hola Mundo";
});
```

Ruta (respuesta JSON)

```
Route::get('json', function () {
    return [ 'nombre'=>'maria','edad'=>23,'invitado'=>true ];
});
```

Ruta (Controller)

```
Route::get('foo', 'ControllerName@function');
```

ROUTING PARAMETERS (URI)

Ruta dinámica – 1 Parámetro (Obligatorio)

```
Route::get('person/{id}', function ($id) {
    return "id Persona: " . $id;
});
```

Ruta dinamica – 2 Parámetros (Obligatorios)

```
Route::get('person/{id}/estatus/{estatus}', function ($id,$estatus) {
    return "idPersona: " . $id . ", estatus: " .$estatus;
});
```

Ruta dinámica – Parámetros opcionales

```
Route::get('person/{id}/estatus/{estatus?}', function ($id,$estatus="activo") {
    return "idPersona: " . $id . ", estatus: " .$estatus;
});
```

Ruta dinámica – Validación de parámetros

```
Route::get('category/{id}/', function ($id) {
    return "idCategory: " . $id;
})->where('id', '[0-9]+');
```

HTTP VERBS

Tipo HTTP Request

```
Route::get('foo', function(){ });
Route::post('foo', function(){ });
Route::put('foo', function(){ });
Route::delete('foo', function(){ });
Route::any('foo', function(){ });
```

MIGRACIONES

Crear tabla de migraciones

- php artisan migrate:install

Crear migraciones

- php artisan migrate

Eliminar migraciones o tablas de BD

- php artisan migrate:rollback

Actualizar la base de datos con todas las migraciones

- php artisan migrate:refresh

Crear una nueva migracion (CLASE)

- php artisan make:migration create_tableName_table --create=nameTable
  La nueva migracion será creada en el directorio “database/migrations”

```
public function up()
{
    Schema::create('productos', function (Blueprint $table) {
        $table->engine = 'InnoDB';
        $table->increments('id');
        $table->string('descripcion',100);
        $table->enum('estatus', ['Activo', 'Inactivo']);
        $table->float('precio', 8, 2);
        $table->text('comentarios')->nullable();
        //$table->date('created_at');
        //$table->time('sunrise');

        $table->timestamps(); // Adds nullable created_at and updated_at columns.
    });
}
```

Más documentación: https://laravel.com/docs/5.4/migrations#creating-tables

ELOQUENT ORM

Crear un nuevo modelo (CLASE)

- php artisan make:model Producto
  El nuevo modelo será creado en app/

INSERT

```
$producto = new Producto();
$producto->descripcion="Computadora Acer";
$producto->precio="1500";
$producto->save();
```

SELECT (Primary Key)

```
$producto = Producto::find($id);
```

UPDATE

```
$producto = Producto::find($id);
$producto->estatus="Inactivo";
$producto->save();
```

DELETE

```
$producto = Producto::find($id);
$producto->delete();
```

DELETE (Si ya sabemos la Primary Key)

```
Producto::destroy(4);
Producto::destroy([6,7]);
```

DELETE (CONDICIONADO)

```
$deletedRows = Producto::where('estatus', 'inactivo')->delete();
```

SELECT (FILTROS)

```
$productos = Producto::all(); // Todos

foreach ($productos as $prod) {
  echo $prod->precio;
}
```

```
$productos = Producto::find(19); // PrimaryKey
$productos = Producto::find([14, 17, 19]); // Varios PrimaryKey
$productos = Producto::where('estatus', 'activo')->first(); // El primero
$productos = Producto::where('estatus', 'activo')->count(); // Count
$productos = Producto::where('precio','=', 1000)->count();
$productos = Producto::where('precio','>=', 1500)->get();
$productos = Producto::where('precio','>=', 1500)->first();

$productos = Producto::where('precio','>=', 1500)
        ->where('estatus','=', 'inactivo')->get();
```

SQL NATIVO

```
$productos = Producto::whereRaw('precio >= ?',[1500])->get();
$productos = Producto::whereRaw('precio >= ? and estatus=?',[1500,'inactivo'])->get();
```