**1. difference between POST,GET,PUT,DELETE**

**Specifically, we use:**

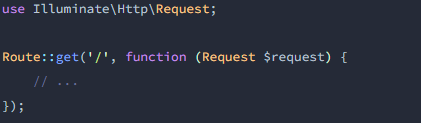
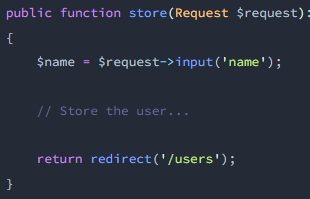
1. POST لإدخال بيانات جديدة الى قاعدة البيانات:
2. GET لإسترجاع بيانات من قاعدة البيانات:
3. PUT تحديث او استبدال بيانات من قاعدة البيانات. الريكويست يحتوي على بيانات جديدة كليا عن الريكورد الموجود في قاعدة البيانات:
4. DELETE لمسح بيانات من قاعدة البيانات:

**2.Request in Laravel**

Laravel's  (Illuminate\Http\Request) class provides an object-oriented way to interact with the current HTTP request being handled by your application as well as retrieve the input, cookies, and files that were submitted with the request.

الريكويست كلاس مقدم من قبل لارفيل للتعامل مع الريكويستات و استخراج البيانات منها

-To obtain an instance of the current HTTP request via dependency injection, you should type-hint the Illuminate\Http\Request class on your route closure or controller method.



**- Retrieving The Request Path :** $uri = $request->path();

**- Inspecting The Request Path / Route :**

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

// ...

}

Using the (routeIs) method, you may determine if the incoming request has matched a named route:

if ($request->routeIs('admin.\*')) {

// ...

}

**-Retrieving The Request Method:**

$method = $request->method();

if ($request->isMethod('post')) {}

**-Retrieving Input**

$input = $request->all(); => لاسترجاع جميع البيانات

$input = $request->input(); => لاسترجاع جميع البيانات

$input = $request->collect(); => نستقبل البيانات في الريكويست و نحولها لكولكشن بحيث ممكن نستفيد من الدوال بتاعت الكولكشن و نعمل عمليات على البيانات

$request->collect('users')->each(function (string $user) {

// ...

});

$name = $request->input('name'); ده في الفورم اللي اليوزر باعتها في انبوت بالاسم

$name = $request->input('products.0.name'); => for arrays

$names = $request->input('products.\*.name');

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

$query = $request->query();

**-Determining If Input Is Present**

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

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

if ($request->hasAny(['name', 'email'])) {} => if one is present

$request->whenHas('name', function (string $input) {}

if ($request->filled('name')) {} => value present and not null string value

**- Retrieving Old Input**

$username = $request->old('username'); => هيجيب اليوزر اللي تم ارساله قبل المرة دي

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

**-** **Retrieving Uploaded Files**

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

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

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

$file = $request->photo;

if ($request->hasFile('photo')) {}

**2. the difference between All() , Get() in elquant**

All() and get () in Eloquent basically has the same functions. All() is a static method. Get() method is on Eloquent\Builder and you can modify the query if you want to. For example User::where(‘name’,’John’)->get().

All() creates new query object then calls get() on it. You cannot modify the query performed. Get() will always return a collection.

كلاهما يقومان بنفس العملية الا و هي استرجاع بيانات من قاعدة البيانات و لكن الفرق ان الاولى ممكن ان نقوم بوضع شروط على الاستعلام و الثانية لا يمكننا ذلك فهي تأتي بكل الموجود في الجدول

**2. the difference between authentication authorization**

authentication is the process of verifying who someone is, whereas authorization is the process of verifying what specific applications, files, and data a user has access to. Authentication is used to verify that users really are who they represent themselves to be. Once this has been confirmed, authorization is then used to grant the user permission to access different levels of information and perform specific functions, depending on the rules established for different types of users.

الاولى تتحقق من المستخدم هل هو موجود و يحق له دخول الموقع ؟

الثانية تتحقق من المستخدم الذي بالفعل يحق له الدخول الى الموقع و لكن تتحقق من صلاحياته للوصول الى الملفات

|  |  |
| --- | --- |
| **Authentication** | **Authorization** |
| Authentication verifies who the user is. | Authorization determines what resources a user can access. |
| Authentication works through [passwords](https://www.sailpoint.com/products/password-management/), one-time pins, biometric information, and other information provided or entered by the user. | Authorization works through settings that are implemented and maintained by the organization. |

**4.Settings in Laravel by two ways:**

First: store settings in DB (dynamic)

1. Settings Model with Migration and Seeder : “php artisan make:model Setting -mcs”

2. Let's add the key and value columns to our migration file. We ensure that the key is always unique and that value can have a null value.

3. In the model, allow key and value attributes to be mass assignable by adding the $fillable property.

4. add some initial entries into the seeder, call SettingSeeder by adding it to DatabaseSeeder :

” public function run(): void

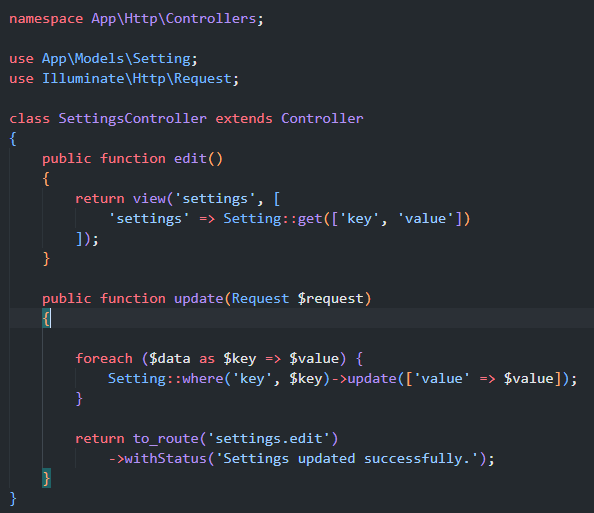
{

$this->call(SettingSeeder::class);

}”

5. run migrations with seeders :” php artisan migrate --seed”.

6. Controller will have edit() and update() methods to show form and update values.



7. set our routes and blades

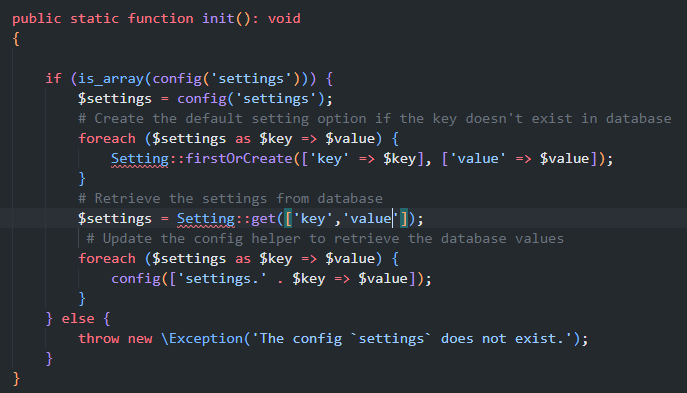
Second: store settings in config directory

1. create settings.php file in config directory

2. return an array of key => value like this:

<?php   
return [  
'title' =>'App Title',  
'description' =>'App Description'  
];

3.store the settings in DB an update config helper function to add settings in SettingController



4. if we need to change the setting we go to settings.php file and edit it then we call init() to update database or make this check happens every time without manually running the init() method, we can include that in the AppServiceProvider using : \App\Http\Controller\SiteSettingsController::init();

5. in blade call {{ config(‘settings.\*’) }} or retrieve the settings from database.

**5.Difference between @yeild,@include,@stack**

@yeild - You can add your content once. يمكن اضافة محتوى مرة واحدة

@stack - Push content into the stack (page) many times.يمكن اضافة محتوى على الذي قبله

@include - with include we blindly imports a full file and insert its content to the location of the @include directive.

عندما نستدعي انكلود فاننا نضع محتوى الصفحة كاملة في المكان المحدد و لا يمكن تغيير المحتوى او استبداله في الصفحة التي تم استدعاؤها

@extends - With yield, yield will only consume the @section that has the name provided in the @yield directive.

عندما نستدعي اكستيند في صفحة فانها تورث المحتوى للصفحة الاصلية و و تستبدل فقط سكشن الييلد الذي في الصفحة الرئيسية

**6.SQL FOREIGN KEY Constraint**

Restrict (on update, on delete) لا يمكن تعديل او حذف الريكورد الاب لان له ابناء في جداول اخرى , يمكن فقط التعديل و الحذف اذا لم يكن له ابناء

Cascade (on update, on delete) اذا تم تعديل على الريكورد الاب سيتم التعديل على الريكورد الابن و اذا تم حذف ريكورد الاب سيتم حذف جميع الريكوردات الابناء

Set null (on update, on delete) عند تعديل او حذف الريكورد الاب يتم حذف القيمة من ريكورد الابن و ليس حذف الريكورد كله

**7.Soft delete in Laravel**

Soft delete hides information from end-user or flags data as deleted while it still remains visible or active in your database.

عندما يحذف المستخدم بيانات فانها تختفي من شاشة المستخدم و لكنها تبقى في قاعدة البيانات و يمكن استرجاعها

To perform soft delete in Laravel, you need to have a deleted\_at column that should be set to default null , add that column in migration file “ $table->softDeletes(); ” as it should be of timestamp data type in the model. “ use SoftDeletes” ; inside the model and import the following class at the top “ use illuminate\Database\Eloquent\SoftDeletes; ”

 To get the deleted data back, you have to add ( ->withTrashed() )to your query.

＄post = Post::where(‘id’,1)->withTrashed()->get() or $post = Post::where(‘id’,1)->withTrashed()->restore();

You can permanently delete a record from the database with the forceDelete method.

“Post::first()->forceDelete(); “