Create a CRUD in Laravel 5

Step 1: create model and migration file

Command: `php artisan make:model Plan -m`

Effect:

1. A model named Plan will be created
2. A migration file named **_create_plans_table will be created

Alternative way:

Command: `php artisan make:model Plan`

Effect: A model named Plan will be created

Command: `php artisan make:migration create_plans_table --create=plans`

Effect: A migration file named **_create_plans_table will be created

Step 2: edit migration file with correct database format.

Code:

```
Schema::create('plans', function (Blueprint $table) {
    $table->increments('id');
    $table->string('title');
    $table->text('body');
    $table->timestamps();
});
```

Effect: A table named plans with columns id, title, body, created_at, updated_at is designed.

File: database>migrations>**_create_plans_table

Step 3: add fillable fields to model

Code: `protected $fillable = ['title', 'body'];`

Effect: with fillable property it is possible to mass assign column title and body

File: app>Plan.php

Step 4: migrate table

Command: `php artisan migrate`

Effect: plans table will be created in database

Step 5: make resource controller

Command: `php artisan make:controller PlanController –resource`

Effect: a resource controller will be created

Step 6: add to route

Code:

```
Route::resource('plans', 'PlanController');
```

Effect: controller methods are now accessible

File: routes>web.php

Step 7: create view files

Directory: resources>views>

Make a folder named plan

Make below files inside the folder

- index.blade.php
- show.blade.php

- edit.blade.php
- create.blade.php

Step 8: update PlanController

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Plan;

class PlanController extends Controller
{

    public function index()
    {

        $plans = Plan::all();

        return view('plan.index', compact('plans'));
    }

    public function create()
    {
        return view('plan.create');
    }

    public function store(Request $request)
    {
        Plan::create(request()->all());

        return redirect()->route('plans.index');
    }

    public function show(Plan $plan)
    {
        return view('plan.show', compact('plan'));
    }

    public function edit(Plan $plan)
    {
        return view('plan.edit', compact('plan'));
    }
    public function update(Request $request, Plan $plan)
    {
        $plan->update($request->all());
        return redirect()->route('plans.index');
    }


    public function destroy(Plan $plan)
    {
        $plan->delete();
        return redirect()->route('plans.index');
    }
}
```

Step 9: update view files

index.blade.php

```
<a href="{{ route('plans.create') }}">Add New</a>

@if(count($plans) > 0)
<table border="1">
    @foreach($plans as $plan)
        <tr>
            <td><a href="{{route('plans.index')}}/{{$plan->id}}"> {{
$plan->title }} </a></td>
            <td><a href="{{route('plans.index')}}/{{$plan-
>id}}/edit">Edit</a></td>
            <form action="{{ route('plans.index') }}/{{$plan->id}}"
method="post">
                {{ method_field('DELETE') }}
                {{ csrf_field() }}
                <td><input type="submit" name="submit"
value="Delete"></td>
            </form>
        </tr>
    @endforeach
</table>
@else
No entry found
@endif
```

show.blade.php

```
<h2>{{ $plan->title }}</h2>
<h3>{{ $plan->body }}</h3>
```

edit.blade.php

```
<form action="{{ route('plans.index') }}/{{$plan->id}}" method="post">
    {{ method_field('PATCH') }}
    Plan title <input type="text" name="title" value="{{ $plan->title
}}"> <br>
    Plan body <input type="text" name="body" value="{{ $plan->body
}}"> <br>
    {{ csrf_field() }}
    <input type="submit" name="submit" value="Submit">
</form>
```

create.blade.php

```
<form action="{{ route('plans.store') }}" method="post">
    Plan title <input type="text" name="title"> <br>
    Plan body <input type="text" name="body"> <br>
    {{ csrf_field() }}
    <input type="submit" name="submit" value="Submit">
</form>
```

CRUD is now fully functional. Now we add custom validations.

Step 10: adding custom validation

Command: php artisan create:request CreatePlanRequest

Effect: create CreatePlanRequest.php file in requests folder

Code:

```php
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class CreatePlanRequest extends FormRequest
{

    public function authorize()
    {
        return true;
    }

    public function rules()
    {
        return [
            'title' => 'required',
        ];
    }
}
```

Command: php artisan create:request UpdatePlanRequest

Effect: create UpdatePlanRequest.php file in requests folder

Code:

```php
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class UpdatePlanRequest extends FormRequest
{

    public function authorize()
    {
        return true;
    }

    public function rules()
    {
        return [
            'title' => 'required',
        ];
    }
}
```

Step 11: update PlanController

```
<?php

namespace App\Http\Controllers;

use App\Http\Requests\CreatePlanRequest;
use App\Http\Requests\UpdatePlanRequest;
use Illuminate\Http\Request;
use App\Plan;

class PlanController extends Controller
{

    public function index()
    {

        $plans = Plan::all();

        return view('plan.index', compact('plans'));
    }

    public function create()
    {
        return view('plan.create');
    }

    public function store(CreatePlanRequest $request)
    {
        Plan::create(request()->all());

        return redirect()->route('plans.index');
    }
    public function show(Plan $plan)
    {
        return view('plan.show', compact('plan'));
    }
    public function edit(Plan $plan)
    {
        return view('plan.edit', compact('plan'));
    }


    public function update(UpdatePlanRequest $request, Plan $plan)
    {
        $plan->update($request->all());
        return redirect()->route('plans.index');
    }


    public function destroy(Plan $plan)
    {
        $plan->delete();
        return redirect()->route('plans.index');
    }
}
```

Step 12: show validation errors

Add below code to these files for showing validation errors

create.blade.php

update.blade.php

```
@if (count($errors) > 0)
    <ul>
        @foreach ($errors->all() as $error)
            <li>{{ $error }}</li>
        @endforeach
    </ul>
@endif
```