Laravel

A. Atlas

Introduction

- Laravel est le framework PHP le plus populaires et les plus utilisés.
- créé par Taylor Otwell en 2011
- permet de simplifier le développement d'applications web tout en gardant le code bien organisé.
- Laravel est basé sur le modèle MVC (Modèle-Vue-Contrôleur)

Introduction

Laravel fournit de nombreux composants prêts à l'emploi, tels que

- l'authentification,
- la validation,
- la gestion des fichiers,
- la mise en cache,
- la gestion des sessions,
- la gestion des tâches planifiées
- …..

Installation

Pour créer un projet laravel, il faut installer ***composer*** puis

$composer create-project laravel/laravel=10 myproject

Structure d’un projet Laravel

app/

bootstrap/

config/

database/

public/

resources/

Les Routes

```php
// fichier routes/web.php
<?php

use Illuminate\Support\Facades\Route;

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

Les Routes

```php
<?php

use Illuminate\Support\Facades\Route;

Route::get('/', function (Request $req) {
    return ["name"=>"Mohammed", "age"=>41];
});
```

Les Routes: querystring

```php
<?php

use Illuminate\Support\Facades\Route;
use Illuminate\Http\Request;

Route::get('/data', function (Request $req) {
    //return $req->all();
    return $req->input("param1","param2");
});
```

Les Routes: params

```php
<?php

use Illuminate\Support\Facades\Route;
use Illuminate\Http\Request;

Route::get('/user/{id}-{name}', function (String $id, String $name)
{
    return ["id"=>$id, "name"=>$name];
});
```

Les Routes: params

```php
<?php
use Illuminate\Support\Facades\Route;
use Illuminate\Http\Request;
Route::get('/user/{id}-{name}', function (String $id, String $name)
{
    return ["id"=>$id, "name"=>$name];
})->name("user");
Route::get('/user', function () {
    return ["link"=>\route("user",["id"=>2,"name"=>"Mohammed"])];
});
```

Les Routes: prefix

```php
<?php
use Illuminate\Support\Facades\Route;
use Illuminate\Http\Request;
Route::get('/user/{id}-{name}', function (String $id, String $name) {
    return ["id"=>$id, "name"=>$name];
})->name("user");
Route::get('/user', function () {
    return ["link"=>\route("user",["id"=>2,"name"=>"Mohammed"])];
});
```

$php artisan route:list

Les Base de données: migrations

- Dans config/database.php sont définis les config des différentes types de base de données
- Dans .env, choisir le type de base de donnée souhaité
  - `DB_CONNECTION=mysql`
- Créer une migration
  - *$ php artisan make-migration CreateUsersTable*

Migrations

```php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
return new class extends Migration{
   public function up(): void{
       Schema::create('users', function (Blueprint $table) {
           $table->id();
            $table->string("name");
            $table->string("email")->unique();
            $table->string("password");
           $table->timestamps();
       });
   }
   public function down(): void{
       Schema::dropIfExists('users');
   }
};
```

$php artisan migrate

Models: insert

$ php artisan make:model Post

```
Route::post('/post', function () {
    $post=new \App\Models\Post();
    $post->title="Introduction à Laravet";
    $post->slug="intro-laravel2";
    $post->content="Un contenu ....";
    $post->save();
    return $post;
});
```

Models: get

```php
Route::get('/post', function () {
    $posts = \App\Models\Post::all();
    //$posts= \App\Models\Post::all(["title","content"]);
    //$posts= \App\Models\Post::find(1);//id=1
    //$posts= \App\Models\Post::findOrFail(1);//id=1
    //$posts= \App\Models\Post::paginate(10);
    //$posts= \App\Models\Post::paginate(10,["id","content"]);
    //$posts = \App\Models\Post::where("id",">",0)->limit(1)->get();
    //dd($posts);
    return $posts;
});
```

Models: update

```
Route::put('/post', function () {

    $post= \App\Models\Post::find(1);
    $post->title="Introduction à Nodejs";
    $post->save();

    return $post;
});
```

Models: delete

```
Route::delete('/post/{id}', function (String $id) {

    $post= \App\Models\Post::find($id);
    $post->delete();

    return $post;
});
```

Le moteur de template Blade

- Blade est le moteur de template utilisé par Laravel.
- Permet d'utiliser du php sur les vues d'une manière particulière.
- Pour créer un fichier qui utilise le moteur de template Blade il faut ajouter l'extension ".blade.php".
- Les fichiers des vues se situent dans le dossier resources/views.

Le moteur de template Blade

- Blade est le moteur de template utilisé par Laravel.
- Permet d'utiliser du php sur les vues d'une manière particulière.
- Pour créer un fichier qui utilise le moteur de template Blade il faut ajouter l'extension ".blade.php".
- Les fichiers des vues se situent dans le dossier resources/views.

Blade

Dans Base.blade.php

…

@yeild(‘content’)

….

Blade

Dans index.blade.php

@extends(‘base’)

@section(‘content’)

…. continue..

@endsection

@section(‘content’,’….contenue…’)

Blade

Pour afficher un contenu, en peut utiliser le php

<?= ‘Un texte important’ ?>

Mais aussi la méthode laravel {{ ‘Un texte important’ }}

Remarque: la méthode 2 échappe automatiquement le HTML

@dump(mavariable)

@if

@if(true)

…

@else

…

@endif

@foreach

@foreach($users as $user)

…

@endforeach

@forelse

@forelse($users as $user)

….

@empty

….

@endforeach

Les controleurs

- Les contrôleurs jouent un rôle central dans la gestion des requêtes HTTP.
- Les contrôleurs sont responsables de:
  - traiter les requêtes entrantes,
  - interagir avec les modèles pour récupérer ou manipuler des données,
  - renvoyer la réponse appropriée au client.

Création d'un contrôleur

- On peut créer un contrôleur à l'aide de la commande artisan de Laravel.
- Par exemple, pour créer un contrôleur nommé ***ArticleController***, on peut exécuter la commande suivante :
    - ***$php artisan make:controller ArticleController***
- Les contrôleurs sont généralement placés dans le répertoire ***app/Http/Controllers***.
- Chaque méthode du contrôleur correspond généralement à une action spécifique que l'application doit effectuer en réponse à une requête.

Validation des données

- La validation des données est une partie cruciale du développement web pour garantir que les données entrantes dans une application sont correctes et sûres.
- Laravel offre un système de validation puissant et flexible qui facilite cette tâche.

Validation au niveau du contrôleur

```php
use Illuminate\Support\Facades\Validator;
use Illuminate\Http\Request;
class UserController extends Controller{
   public function methode(Request $request){
        $validator = Validator::make($request->all(), [
            'nom' => 'required|string|max:255',
            'email' => 'required|email|max:255',
            'passwd' => 'required|string|min:6',
        ]);
      // $validator->errors(), $validator->validated() ou $validator->fails()
        if ($validator->fails()) {
            return redirect('route_de_redirection');
        }// Le reste du code si la validation réussit
   }
```

Validation au niveau du contrôleur

```php
use Illuminate\Support\Facades\Validator;
use Illuminate\Http\Request;
class UserController extends Controller{
   public function methode(Request $request){
        $validator = Validator::make($request->all(), [
            'nom' => ['required', 'string' ,'max:255' ],
            'email' => ['required',email ','max:255'],
            'passwd' => ['required',string','min:6'],
        ]);
      // $validator->errors(), $validator->validated() ou $validator->fails()
        if ($validator->fails()) {
            return redirect('route_de_redirection');
        }// Le reste du code si la validation réussit
   }
```

Validation au niveau d'une requête

Créer un requête personnalisée

$php artisan make:request UserFilterRequest

```php
use Illuminate\Foundation\Http\FormRequest;
class UserFilterRequest extends FormRequest
{
   // Determine if the user is authorized to make this request.
   public function authorize(): bool{
       return false;
   }
   //Get the validation rules that apply to the request.
```

```php
public function rules(): array{
   return [
      'nom' => ['required', 'string' ,'max:255'],
      'email' => ['required','email','max:255'],
      'passwd' => ['required','string','min:6']
   ];
   }
}
```

Validation au niveau du contrôleur

```php
use Illuminate\Support\Facades\Validator;
use Illuminate\Http\Request;
class UserController extends Controller{
   public function methode(UserFilterRequest $request){
     // $request->errors(), $request->validated() ou $request->fails()
       if ($request->fails()) {
           return redirect('route_de_redirection');
       }// Le reste du code si la validation réussit
   }
```

Model Binding

- Le "Model Binding" (liaison de modèle) est une fonctionnalité puissante de Laravel qui simplifie la tâche d'interrogation de la base de données et la récupération de modèles en fonction des paramètres de la requête.
- Cette fonctionnalité est souvent utilisée dans les contrôleurs pour récupérer automatiquement un modèle en fonction de son identifiant.

Model Binding Implicite

```
Route::get('/users/{user}', ['UserController','show']);
```

Laravel va automatiquement extraire l'ID ({user}) de la route, récupérer le modèle correspondant depuis la base de données, et l'injecter dans la méthode du contrôleur.

```
public function show(User $user)
{
// $user est déjà récupéré du modèle en fonction de l'ID fourni dans l'URL
    return view(users.show', $user));
}
```

Model Binding: Personnalisation de la clé

```
Route::get('/users/{user:email}', ['UserController','show']);
```

Laravel va automatiquement extraire l'ID ({user}) de la route, récupérer le modèle correspondant depuis la base de données, et l'injecter dans la méthode du contrôleur.

```
public function show(User $user)
{
// $user est déjà récupéré du modèle en fonction de l'ID fourni dans l'URL
    return view(users.show', $user));
}
```

Les formulaires

Les Relations

Un article appartient à une catégorie

Php artisan make:category -m

BelongsTo

HaseMany

BelongsToMany

Upload des fichier dans Laravel

Pour permettre aux utilisateurs de télécharger des fichiers dans une application Laravel, vous pouvez suivre ces étapes générales :

1. Configurer le formulaire HTML
2. Créer le contrôleur
3. Configurer la route
4. Afficher les messages de succès/erreur

Configurer le formulaire HTML

```
<!-- resources/views/upload.blade.php -->

<form action="{{ route('upload') }}" method="post"
enctype="multipart/form-data">
    @csrf
    <input type="file" name="file">
    <button type="submit">Upload</button>
</form>
```

Créer le contrôleur

$ php artisan make:controller UploadController

app/Http/Controllers/UploadController.php

```php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class UploadController extends Controller{
   public function upload(Request $request){
       $request->validate([
           'file' => 'required|mimes:jpeg,png,jpg,gif,svg,pdf|max:2048',
       ]);
       $file = $request->file('file');
       $fileName = $file->store("blog","public");
       return back()->with('success', 'File uploaded successfully:'.$fileName);
   }
}
```

Configurer la route

```php
// routes/web.php

use App\Http\Controllers\UploadController;

Route::post('/upload', [UploadController::class,
'upload'])->name('upload');
```

Afficher les messages de succès/erreur

```
<!-- resources/views/upload.blade.php -->
@if(session('success'))
<p style="color:green;">{{ session('success') }}</p>
@endif
@if($errors->any())
<ul>
    @foreach($errors->all() as $error)
        <li style="color:red;">{{ $error }}</li>
    @endforeach
</ul>
@endif
```