S2: Structure d’un projet Laravel

Un projet Laravel est organisé de manière logique avec plusieurs répertoires et fichiers importants :

- **app/** : Contient la logique de l'application, notamment les modèles, les contrôleurs et les services.
- **resources/** : Contient les vues (fichiers Blade) et les fichiers de langue.
- **routes/** : Contient les fichiers de définition des routes de l'application (`web.php` et `api.php`).
- **public/** : Dossier public contenant les fichiers accessibles publiquement, comme les images, les fichiers CSS et JavaScript.
- **database/** : Contient les migrations et les modèles Eloquent pour interagir avec la base de données.

Laravel : Une Architecture MVC avec Laravel

Laravel repose sur une architecture MVC.

Modèle (Model)

Les modèles Eloquent de Laravel permettent d’interagir facilement avec la base de données. Un modèle représente une table dans la base de données et contient des méthodes pour récupérer et manipuler les données.

Exemple de modèle `Post` :

```
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Post extends Model
```

```
{
    protected $fillable = ['title', 'content'];
}
```

Vue (View)

Les vues sont des fichiers Blade qui sont utilisés pour afficher l'interface utilisateur. Blade est le moteur de templates de Laravel, qui permet de rendre du HTML avec une syntaxe simple.

Exemple de vue `welcome.blade.php` :

```
<!DOCTYPE html>
<html>
<head>
    <title>Laravel App</title>
</head>
<body>
    <h1>Bienvenue dans Laravel!</h1>
    <p>{{ $message }}</p>
</body>
</html>
```

Contrôleur (Controller)

Les contrôleurs contiennent la logique métier. Ils reçoivent les requêtes HTTP, interagissent avec les modèles et renvoient les vues.

Exemple de contrôleur `PostController` :

```
namespace App\Http\Controllers;

use App\Models\Post;
use Illuminate\Http\Request;

class PostController extends Controller
{
    public function index()
    {
        $posts = Post::all();
```

```
        return view('posts.index', compact('posts'));
    }
}
```

6. Routes dans Laravel

Laravel utilise un système de routage très flexible. Les routes sont définies dans le fichier `routes/web.php` pour les requêtes web classiques et `routes/api.php` pour les API.

Exemple de route basique :

```
use App\Http\Controllers\PostController;

Route::get('/', [PostController::class, 'index']);
```

Cette route mappe la requête HTTP GET à l'URL racine `/` vers la méthode `index` du contrôleur `PostController`.

```
class Post extends Model
{
    // Define which attributes to cast to specific data types
    protected $casts = [
        'is_published' => 'boolean',  // Cast the 'is_published' attribute as boole
an
        'published_at' => 'datetime', // Cast the 'published_at' attribute as a dat
etime
        'metadata' => 'array',        // Cast the 'metadata' attribute as an array
        'price' => 'decimal:2',       // Cast the 'price' attribute to a decimal with 2
decimal places
    ];

    // These are fillable attributes, which means they can be mass-assigned
```

```
    protected $fillable = ['title', 'content', 'is_published', 'published_at', 'meta
data', 'price'];
}
```

7. Migration et base de données

Laravel simplifie la gestion de la base de données avec les **migrations** et l'**ORM Eloquent**.

Créer une migration :

Les migrations permettent de versionner la structure de la base de données. Pour créer une migration, utilisez la commande artisan suivante :

```
php artisan make:migration create_posts_table
```

Exécuter les migrations :

Une fois la migration définie, exécutez-la avec la commande suivante :

```
php artisan migrate
```

Cela appliquera toutes les migrations en attente et mettra à jour la base de données.

---

8. Validation des données avec Laravel

Laravel offre un système de validation très puissant, avec des règles simples à appliquer sur les entrées des utilisateurs.

Exemple de validation dans un contrôleur :

```
$request→validate([
    'title' ⇒ 'required|max:255',
    'content' ⇒ 'required',
```

```
]);
```

9. Authentification et sécurité

Laravel facilite l'implémentation de l'authentification avec des fonctionnalités intégrées comme l’enregistrement des utilisateurs, la connexion, et la gestion des sessions. Il inclut aussi des mécanismes de sécurité pour protéger votre application contre les attaques.

Pour installer l’authentification de base :

```
composer require laravel/ui
php artisan ui bootstrap --auth
npm install && npm run dev
```

Cela générera des vues et des routes pour l’inscription, la connexion et la déconnexion.

10. Bonnes pratiques et conventions de codage avec Laravel

Laravel suit une série de bonnes pratiques qui permettent de garder le code propre et maintenable. Voici quelques points importants :

- **Utilisez des contrôleurs et des services** pour organiser la logique métier.
- **Adoptez les conventions de nommage** de Laravel (par exemple, les modèles au singulier, les noms de tables au pluriel).
- **Utilisez les migrations et les seeders** pour gérer les changements dans la base de données et les données de test.
- **Sécurisez vos applications** avec des protections CSRF, des validations et l’utilisation des middlewares.