TP : Utilisation des routes avec les contrôleurs et les modèles dans Laravel

Objectifs :

- Comprendre les différents types de routes dans Laravel.
- Apprendre à utiliser des contrôleurs pour gérer les requêtes.
- Utiliser un modèle pour interagir avec la base de données.

---

1. Créer des routes simples

Théorie :

Une **route** dans Laravel relie une requête à un traitement spécifique. Les routes peuvent renvoyer des vues, des données JSON, ou rediriger vers d'autres pages.

Question :

Créez une route simple qui retourne du texte dans le fichier `routes/web.php`.

```
Route::get('/hello', function () {
    return 'Hello, world!';
});
```

Réponse :

Cette route est une route GET qui renvoie simplement "Hello, world!" lorsqu'on accède à l'URL `http://localhost:8000/hello`.

Explication :

Dans cet exemple, la méthode **GET** est utilisée pour répondre à une requête en affichant du texte. Cette méthode est couramment utilisée pour afficher des pages ou des informations statiques.

---

2. Route avec paramètres dynamiques

Théorie :

Une route peut accepter des **paramètres dynamiques** qui sont passés à la fonction pour être utilisés dans le traitement de la requête.

Question :

Créez une route qui prend un paramètre dynamique `name` et renvoie "Bonjour, {name}".

```
Route::get('/bonjour/{name}', function ($name) {
```

```
    return "Bonjour, $name!";
});
```

Réponse :

Accéder à `http://localhost:8000/bonjour/Jean` affichera "Bonjour, Jean!".

Explication :

Les accolades `{}` permettent de capturer des variables dynamiques dans l'URL, qui sont ensuite accessibles comme paramètres de la fonction.

---

3. Utiliser un contrôleur pour organiser le code

Théorie :

Les contrôleurs permettent de mieux organiser le code en séparant la logique des routes. Ils agissent comme intermédiaires entre les routes et les modèles.

Question :

Créez un contrôleur appelé `WelcomeController` et déplacez-y la logique de la route `hello`.

1. Créez le contrôleur :

   ```
   php artisan make:controller WelcomeController
   ```

2. Dans le contrôleur `WelcomeController.php`, ajoutez la méthode suivante :

   ```
   public function sayHello()
   {
       return 'Hello, world!';
   }
   ```

3. Modifiez la route pour utiliser ce contrôleur :

   ```
   Route::get('/hello', [App\Http\Controllers\WelcomeController::class,
   'sayHello']);
   ```

Réponse :

La logique de la route a été déplacée dans le contrôleur `WelcomeController`, et la route utilise maintenant ce contrôleur pour retourner "Hello, world!".

Explication :

Les **contrôleurs** permettent de garder le fichier de routes propre en déléguant les traitements à des classes dédiées.

4. Route avec paramètres dynamiques dans un contrôleur

Théorie :

Un contrôleur peut également accepter des paramètres dynamiques passés via une route.

Question :

Modifiez la route `bonjour/{name}` pour utiliser un contrôleur. Créez une méthode dans `WelcomeController` qui accepte un paramètre `name`.

1. Ajoutez cette méthode au `WelcomeController` :

```
public function sayBonjour($name)
{
    return "Bonjour, $name!";
}
```

2. Modifiez la route pour qu'elle utilise le contrôleur :

```
Route::get('/bonjour/{name}',
[App\Http\Controllers\WelcomeController::class, 'sayBonjour']);
```

Réponse :

La route `/bonjour/{name}` est maintenant gérée par le contrôleur, et elle affiche "Bonjour, {name}" pour tout nom passé en paramètre.

5. Utiliser un modèle avec un contrôleur

Théorie :

Les modèles représentent les entités de la base de données et sont utilisés pour interagir avec la base de données via Eloquent ORM.

Question :

Créez un modèle `Post` et une table associée via une migration, puis configurez un contrôleur pour récupérer et afficher des articles depuis la base de données.

1. Créez un modèle avec sa migration :

```
php artisan make:model Post -m
```

2. Dans la migration `create_posts_table.php`, ajoutez les colonnes suivantes :

```
Schema::create('posts', function (Blueprint $table) {
```

```
        $table->id();
        $table->string('title');
        $table->text('content');
        $table->timestamps();
    });
```

3. Exécutez la migration pour créer la table :

```
php artisan migrate
```

4. Ajoutez des données dans la base de données à l'aide de Tinker ou directement via un seeder :

```
php artisan tinker
$post = new App\Models\Post;
$post->title = 'Premier article';
$post->content = 'Voici le contenu du premier article.';
$post->save();
```

5. Créez un contrôleur `PostController` :

```
php artisan make:controller PostController
```

6. Dans le contrôleur, ajoutez une méthode pour récupérer et afficher tous les articles :

```
public function index()
{
    $posts = \App\Models\Post::all();
    return view('posts.index', compact('posts'));
}
```

7. Créez une vue `posts/index.blade.php` :

```
@foreach ($posts as $post)
    <h2>{{ $post->title }}</h2>
    <p>{{ $post->content }}</p>
@endforeach
```

8. Modifiez la route pour qu'elle pointe vers le contrôleur :

```
Route::get('/posts', [App\Http\Controllers\PostController::class,
'index']);
```

Réponse :

La route `/posts` affiche tous les articles présents dans la base de données. La méthode `index` du contrôleur utilise le modèle **Post** pour récupérer les données et les afficher dans une vue.

Explication :

Les **modèles** permettent d'interagir avec la base de données en utilisant des méthodes simples comme `all()` pour récupérer toutes les entrées d'une table.

6. Route POST et formulaire

Théorie :

Les routes **POST** sont utilisées pour envoyer des données au serveur, souvent à partir d'un formulaire.

Question :

Créez un formulaire pour ajouter un nouvel article et gérez l'envoi avec une route POST dans un contrôleur.

1. Dans le contrôleur `PostController`, ajoutez la méthode suivante pour afficher le formulaire :

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

2. Créez une vue `posts/create.blade.php` :

```
<form action="/posts" method="POST">
    @csrf
    <label for="title">Titre :</label>
    <input type="text" name="title" id="title">
    <label for="content">Contenu :</label>
    <textarea name="content" id="content"></textarea>
    <button type="submit">Ajouter l'article</button>
</form>
```

3. Ajoutez une méthode pour gérer la soumission du formulaire :

```
public function store(Request $request)
{
    $post = new \App\Models\Post;
    $post->title = $request->title;
    $post->content = $request->content;
    $post->save();

    return redirect('/posts');
}
```

4. Modifiez les routes pour inclure la route GET et POST :

```
Route::get('/posts/create',
[App\Http\Controllers\PostController::class, 'create']);
Route::post('/posts', [App\Http\Controllers\PostController::class,
'store']);
```

Réponse :

Le formulaire permet d'ajouter un nouvel article dans la base de données. La route POST gère l'envoi des données, et le contrôleur les enregistre.

Conclusion

Ce TP dirigé couvre :

- Les routes simples, avec et sans paramètres.
- L'utilisation des contrôleurs pour organiser la logique.
- L'utilisation d'un modèle pour interagir avec la base de données.
- La gestion des formulaires et des routes POST.

Ce TP peut être étendu avec des validations de formulaires, des mises à jour et des suppressions d'articles pour une couverture plus complète.