

Bloc Laravel · chapitre 5

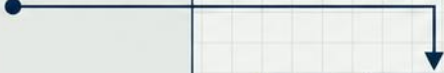
MVC

Un fichier de 26 lignes devient quatre fichiers, que chaque requête traverse dans le même ordre.



Un blog en un seul fichier : posts.php

```
1 <?php
2 $pdo = new PDO('sqlite:' . __DIR__ . '/database.sqlite');
3
4 $order = ($_GET['ordre'] ?? '') === 'ancien' ? 'ASC' : 'DESC';
5
6 $posts = $pdo
7     ->query("SELECT title, body FROM posts ORDER BY created_at $order")
8     ->fetchAll(PDO::FETCH_ASSOC);
9 ?>
```



Un paramètre d'URL ajoute une valeur nommée à une adresse, ici ?ordre=ancien.

Le même fichier fabrique aussi la page

```
...
16 <body>
17     <h1 style="color: navy">Le blog</h1>
18     <p><?= count($posts) ?> articles</p>
19     <ul>
20         <?php foreach ($posts as $post): ?>
21             <li><strong style="color: navy"><?= $post['title'] ?></strong> : <?= $post['body'] ?></li>
22         <?php endforeach; ?>
23     </ul>
24     <a href="?ordre=ancien">Du plus ancien au plus récent</a>
25 </body>
```

Cinq minutes en binôme : quel fichier, quelles lignes ?

...

```
<h1 style="color: navy">Le blog</h1>
```

```
<p>3 articles</p>
```

...

1 couleur → ?

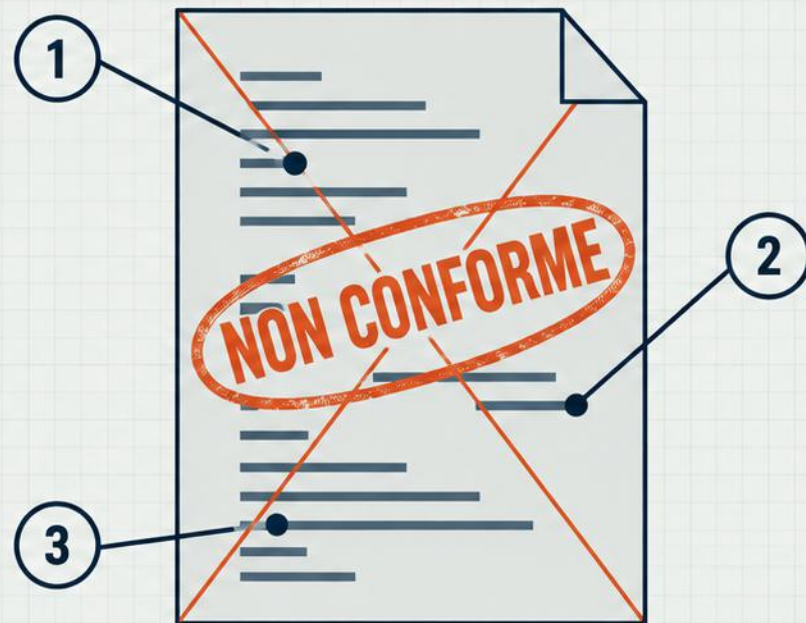
2 tri → ?

3 MySQL → ?

Que faut-il modifier pour changer la couleur du titre ? pour trier autrement ? pour passer de SQLite à MySQL ?

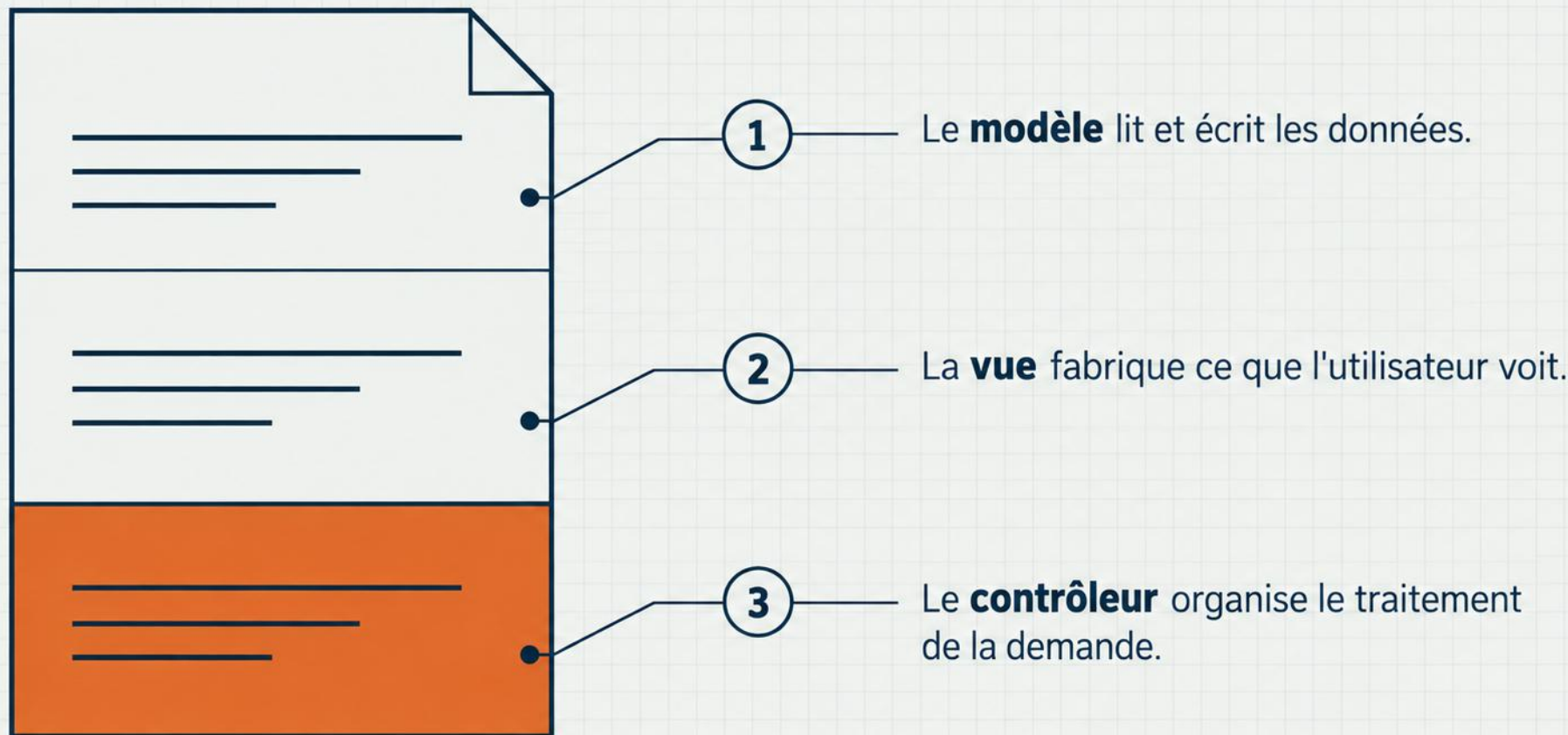
Trois demandes sans rapport, un seul fichier : posts.php

Demande	Lignes de posts.php
La couleur	17 et 21
Le tri	4, 7 et 24
MySQL	2, et chaque autre page



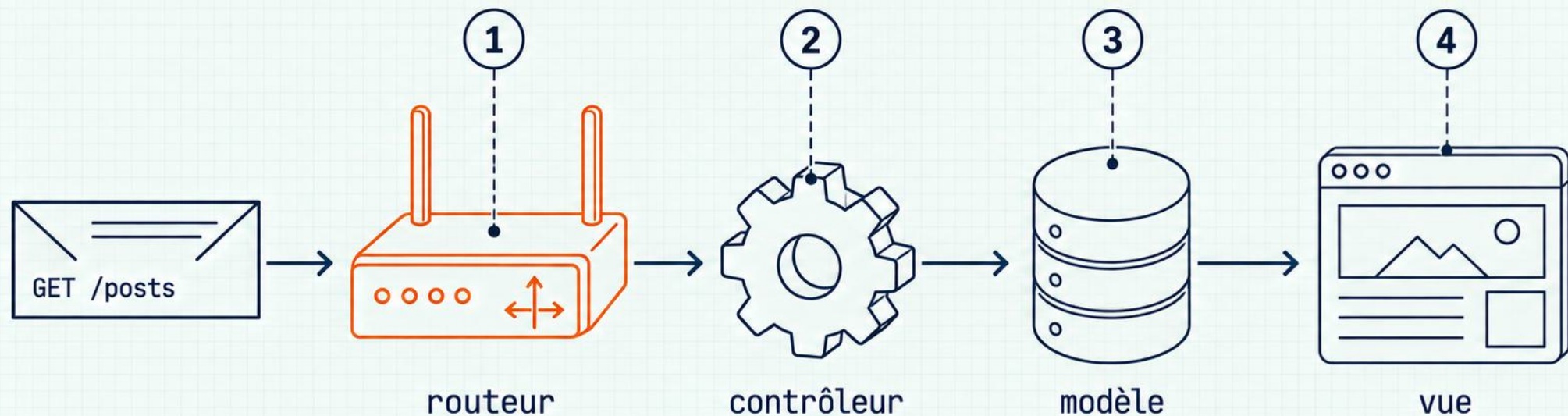
Une personne qui change une couleur peut casser la requête SQL.

Une responsabilité est une tâche précise confiée à une partie du programme



Le modèle s'occupe des données, la vue de l'affichage, le contrôleur coordonne les deux et le routeur choisit le contrôleur

Le routeur relie une adresse à la méthode chargée de la traiter. Sans lui, `/posts.php` lance le fichier `posts.php`.



routes/web.php : l'adresse n'est plus un nom de fichier

```
1 <?php
2
3 use App\Http\Controllers\PostController;
4 use Illuminate\Support\Facades\Route;
5
6 Route::get('/', function () {
7     return view('welcome');
8 });
9
10 Route::get('/posts', [PostController::class, 'index']);
```

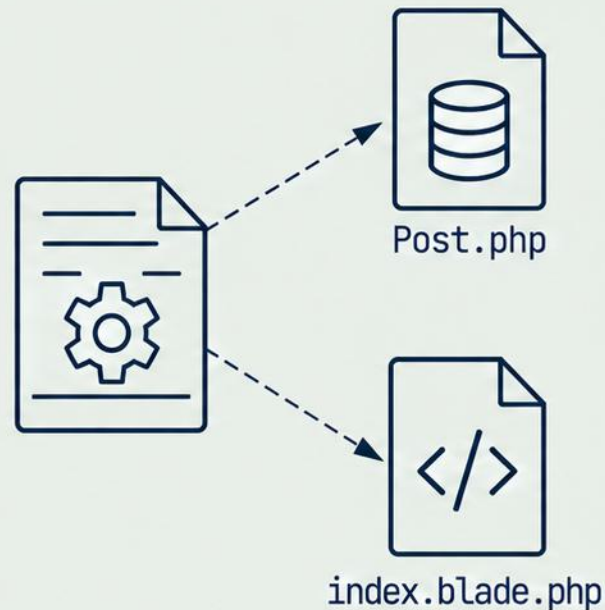


PostController.php

Une route est une association entre une requête et le code qui la traite.

Le contrôleur demande au modèle, puis passe le résultat à la vue

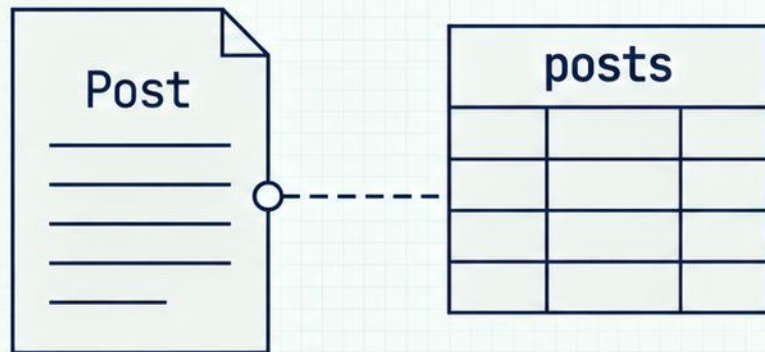
```
1  class PostController extends Controller
2  {
3      public function index(): View
4      {
5          $posts = Post::latest()->get();
6
7          return view('posts.index', ['posts' => $posts]);
8      }
9  }
```



posts.index désigne resources/views/posts/index.blade.php. La clé 'posts' crée la variable \$posts de la vue.

Post est vide, et Eloquent écrit le SQL à sa place

```
1 class Post extends Model
2 {
3     //
4 }
```



Eloquent est la partie de Laravel qui relie les classes PHP aux tables de la base de données. `Post::latest()->toRawSql()` affiche :

```
select * from "posts" order by "created_at" desc
```

La migration décrit la table posts

```
1 public function up(): void
2 {
3     Schema::create('posts', function (Blueprint $table) {
4         $table->id();
5         $table->string('title');
6         $table->text('body');
7         $table->timestamps();
8     });
9 }
```

```
$ php artisan migrate
```

```
INFO Running migrations.
```

```
2026_09_26_093317_create_posts_table ..... 3.11ms DONE
```

id	title	body	created_at	updated_at

Une migration est un fichier qui décrit une modification de la base de données.

{{ }} échappe : l'apostrophe devient '

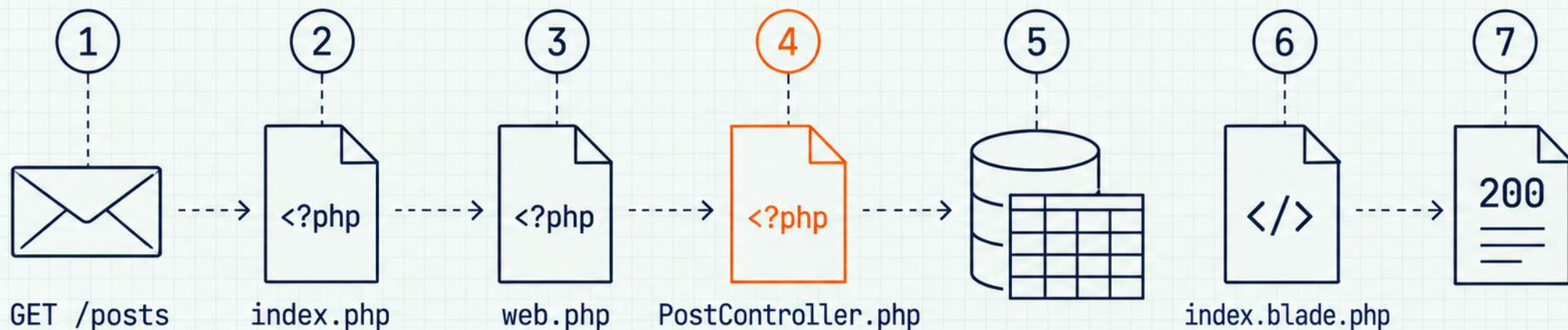
```
1  ...
2  <body>
3      <h1>Le blog</h1>
4      <ul>
5          @foreach ($posts as $post)
6              <li><strong>{{ $post->title }}</strong> : {{ $post->body }}</li>
7          @endforeach
8      </ul>
9  </body>
10 ...
```



```
$ curl -s http://premier-test-laravel.test/posts
...
    <li><strong>Le trajet d'</strong>une requête</strong> : De l'URL à la page HTML.</li>
...
```

Un template Blade est un fichier qui décrit le HTML à produire.

Le contrôleur coordonne, la vue n'interroge jamais la base



GET /posts traverse sept étapes, de public/index.php à la réponse 200.

MVC ne garantit rien : Laravel exécute du SQL dans une vue

```
1 @php
2     $posts = DB::select('SELECT title FROM posts ORDER BY created_at DESC');
3 @endphp
4 @foreach ($posts as $post)
5     <li>{{ $post->title }}</li>
6 @endforeach
```



- Pour changer le tri, il faut modifier le template.
- Chaque autre vue des articles recopie la requête.

Du HTML dans un modèle

```
1 class Post extends Model
2 {
3     public function toHtml(): string
4     {
5         return '<li><strong style="color: navy">'. $this->title . '</strong></li>';
6     }
7 }
```



- Changer une couleur oblige à modifier le modèle.
- Et le titre n'est pas échappé : un `<script>` serait exécuté.

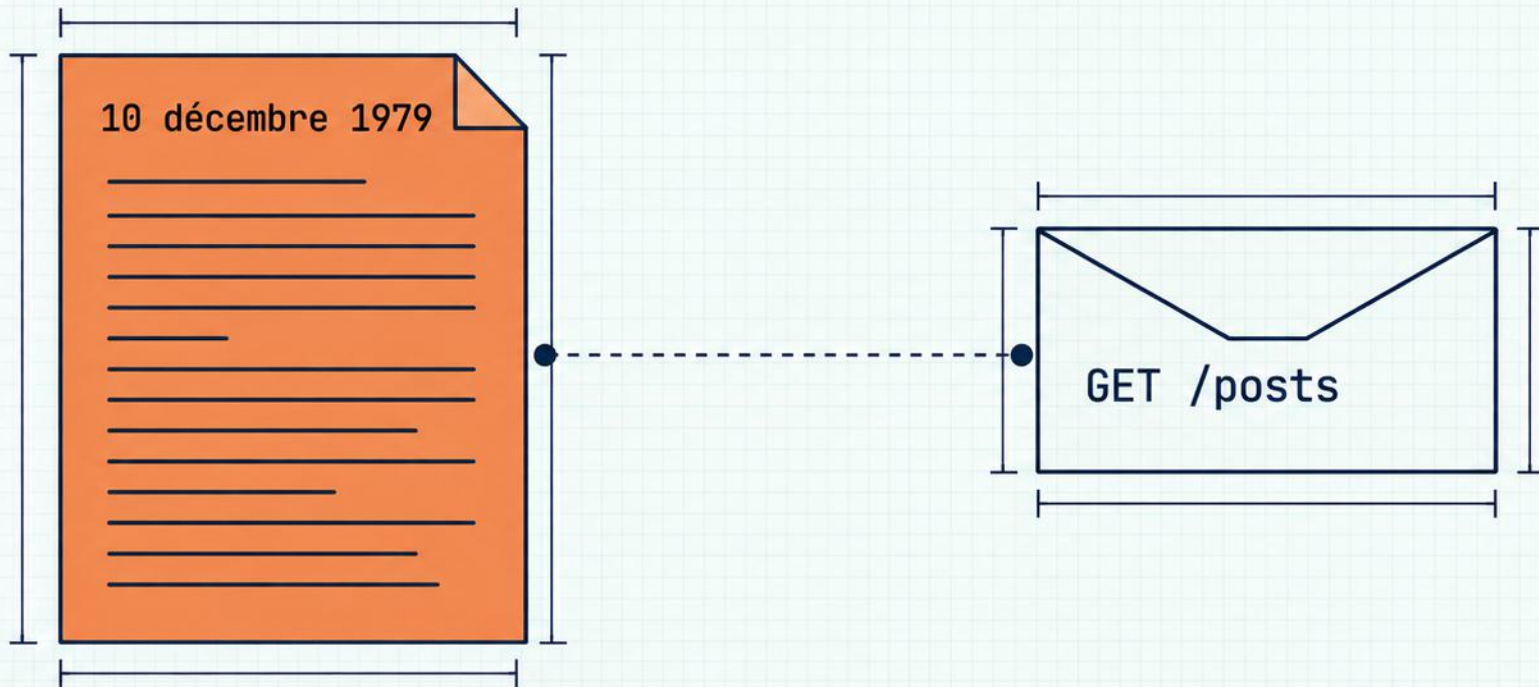
De la logique métier dans un contrôleur de 200 lignes

```
1 $posts = Post::latest()->get();  
2  
3 foreach ($posts as $post) {  
4     $words = str_word_count($post->body);  
5     $post->reading_time = max(1, (int) ceil($words / 200));  
6 }  
7  
8 // ... 190 lignes de plus : filtres, statistiques, envoi d'e-mails
```



- La logique métier regroupe les règles propres à votre application, comme « un article se lit à 200 mots par minute ».

MVC n'est pas une invention de Laravel



Trygve Reenskaug travaille à Xerox PARC sur les interfaces graphiques de Smalltalk.
Sa note du 10 décembre 1979 s'intitule « Models-Views-Controllers ».

À vous : ranger, tracer, construire

40 minutes, en séance.

- Ex. 1, sur papier : ranger huit extraits.
- Ex. 2, sur papier : tracer GET /posts/3.
- Ex. 3 : les quatre fichiers dans premier-test-laravel.

