Système de routing

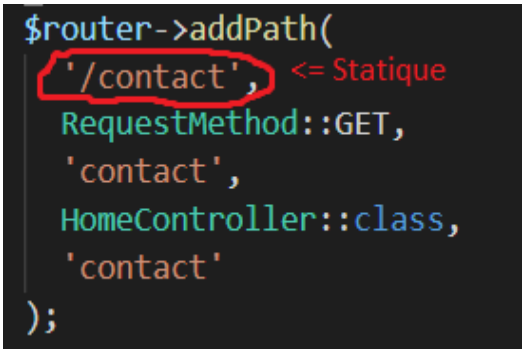
Problématique

Afin d'améliorer notre architecture MVC de notre application, nous souhaiterions mettre en place un système de routing dynamique. Cela permettrait de passer et d'accepter différents paramètres GET dans la route.

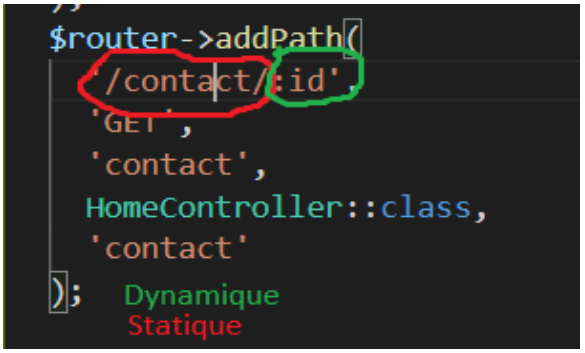
Situation actuelle

A l'heure actuelle, notre système de routing ne permet au paths de fonctionner seulement de façon statique. En effet, le path inséré dans la route est le seul qui permet d'accéder à la vue correspondante. De fait, nous ne pouvons pas insérer des paramètres dynamiques tels que des "id", nous rencontrerons une erreur 404. Nous allons rendre ce système de routing plus dynamique afin de pouvoir passer un ou plusieurs paramètres.

Exemple d'une route actuelle statique

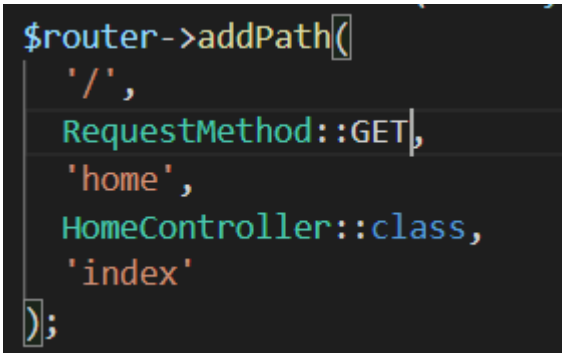


Exemple d'une future route dynamique



Mise en place des énumérations des méthodes

En premier lieu, nous avons mis en place une classe qui répertorie les types de méthodes en constante. Ce qui nous permettra de ne plus indiquer "en dur" le type de méthode. Voici son utilisation :



Mise en place des paramètres dynamique de la route

Nous sommes partie sur la piste des mêmes routeurs que Laravel.
Dans la route, nous spécifions si nous attendons un paramètre dynamique ou non en utilisant la forme "**:param**".
En creusant cette piste, nous avons commencé à couper en plusieurs morceaux le path avec la fonction "**explode()**" comme ci-dessous.
Avec la fonction "**preg_match()**", on cible les différents morceaux du path pour récupérer ceux qui ont la forme d'un paramètre dynamique :param".

En prenant l'exemple d'un **path = /contact/:id/id2**
Nous retrouverons **$parameters = [':id', ':id2']**.

On utilise ensuite la fonction "**str_replace()**" pour enlever le " : " dans le tableau **$parameters**.
Pour finir, on envoie le paramètre dans les paramètres du contrôleur en question.

Voici ce que donne le code :

```php
$url = trim($requestPath, '/'); // enlever les espace en trop
$url = filter_var($url, FILTER_SANITIZE_URL); // Retirer tous les caractères inconnue dans une URL
$url = explode('/', $url); // Couper l'URL à chaque "/"
foreach($url as $parameters) // Pour chaque parametres
{
  if(preg_match('/\:.+\b/', $parameters)) // regarder s'il s'agit d'un parametre de la forme :param
  {
    $parameters = str_replace(':', '', $parameters); // enlever le :
    array_push($params, $parameters); // mettre son nom en parametre
  }
}
```

On se retrouve très vite que ce bout de code ne serait pas bon car il manquera de beaucoup d'éléments. Le fait de ne pas prendre en compte les paramètres complexes tel que **path = contact/edit/person-2** au lieu de **path = contact/edit/2**

Pour justifier ce problème, il faudrait englober les paramètres, c'est à dire de passer de la forme **path = contact/edit/:id** à **path = contact/edit/{id}**

Un autre problème allait survenir, en essayant d'approfondir notre compréhension d'une architecture MVC, nous nous sommes rendu compte que dans l'algorithme ci-dessus, nous ne comparons jamais l'URL du serveur avec les paths. Ce qui ne nous permet pas d'avoir des paramètres dynamiques.

Voici l'algorithme que j'ai mis en place, il est encore en développement mais je vais expliquer comment je suis parvenu à faire cet algorithme.

Premièrement, comme nous pouvons voir la première ligne correspond à une expression régulière, qui veut dire que tout ce qui est à l'intérieur des accolades et qui correspond à tous types de caractères (chiffre et/ou lettre), on le remplace par tous sauf les slash sur la variable $path qui correspond à notre route.

Deuxièmement la variable $urlToMatch, veut dire en expression régulière qu'on veut remplacer toute la chaîne de caractère.

Ensuite, nous allons effectuer un match cela veut dire comparer le path (la route) à l'URL du serveur. Si le match fonctionne nous aurons un tableau appelé $results, qui comprendra en première position l'URL complètes, et les positions suivantes seront les valeurs du ou des paramètres dynamiques .

De plus, j'utilise array_shift pour écraser la premières valeur du tableau qui correspond à l'URL complète, je veux seulement les valeurs des paramètres.

Et pour finir, je vérifie si $results n'est pas vide et qu'il a bien un paramètre au minimum, si oui je stock la valeurs de ou des paramètres dans le tableau global $params, qui correspond au paramètres à injecter.

```php
$url = preg_replace('#({[\w]+})#', '([^/]+)', $path);
$urlToMatch = "#^$url#";

if(preg_match($urlToMatch, $requestPath, $results))
{
  array_shift($results);
  if($results != null)
  {
    array_push($params, $results);
  }
}
```

Après avoir essayé ce bout de code en BRUT en rentrant manuellement mon path et mon URL, je peux voir qu'il fonctionne pratiquement, mais à l'inverse si j'appel mon url normalement j'obtiens une erreur 404. Je pense que cette erreur vient de la partie checkPath. Je m'explique comme on peut le voir sur le photo ci-dessous, nous vérifions si la route actuelle est égale à l'URL du serveur mais en aucun cas elle va prendre en compte ces changements de paramètres dynamiques. Je suis encore à la recherche d'informations pour combler cette erreur et avancer.

```php
public function checkPath(string $requestPath, string $requestMethod)
{
  foreach ($this->paths as $path) {
    if ($path['path'] === $requestPath && $path['http_method'] === $requestMethod) {
      return $path;
    }
  }
  return false;
}
```

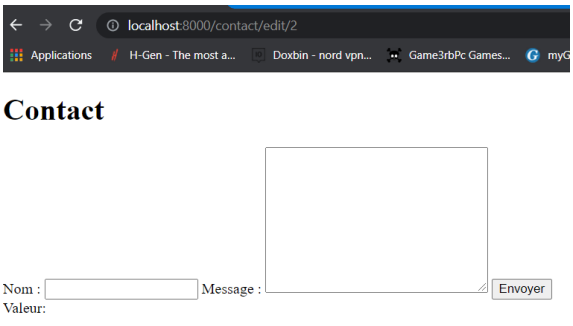
Après quelques jours d'analyse de la structure du MVC, je me suis aperçu que mon code concernant le match entre l'URL et la route n'était pas au bon endroit. Il fallait que je place ce code au sein de la fonction checkPath. Parce que le problème actuel c'est que la fonction checkPath analyse seulement la route brute et pas en mode dynamique.

```php
public function checkPath(string $requestPath, string $requestMethod)
{
  foreach ($this->paths as $path) {
    $Pregpath = preg_replace('#({[\w]+})#', '([^/]+)', $path['path']);
    $PathToMatch = "#^$Pregpath$#";

    if(preg_match($PathToMatch, $requestPath, $matches) && $path['http_method'] == $requestMethod)
    {
      array_shift($matches); // Suppression du premier index qui équivaut à l'URL
      if(isset($matches))
      {
        // Inserer paramètres
        return $path;
      }
    }
  }
}
```

Après avoir fait tous ces changements, le réjouissement est apparu, le fonctionnement de mes routes dynamiques fonctionnait enfin ! (voir photo ci-dessous). Après plusieurs tests, les routes dynamiques fonctionnaient enfin, mais malheureusement il reste un élément essentiel c'est l'injection des paramètres dans la fonction pour pouvoir obtenir la valeur du paramètres (par exemple l'ID).

(exemple d'une route dynamique fonctionnelle)



Injection des paramètres et obtenir la valeur des paramètres

Comme je l'ai expliqué plus tôt, il manquait une partie essentielle à ce projet de route dynamique, c'est l'injection des paramètres et pouvoir obtenir leurs valeurs. Après de nombreuses recherches, et d'analyse du code, j'ai une 2 pistes. La première était de partir sur la fonction InvokeArgs de la méthode ReflexionMethod, ce qui permettait d'invoquer des paramètres. Mais après plusieurs essais, j'ai senti que ce n'était pas la bonne solution. De plus, en me penchant plus sur le code, j'ai aperçu l'utilisation de la variable global $this->params, qui permettait d'injecter des paramètres. En observant cette méthode, il fallait que j'obtienne l'index qui équivaut au nom du paramètre dynamique (ex: id) et avoir la valeur du paramètre. Actuellement j'ai seulement accès à la valeur du paramètre mais pas à son nom. Voici le code complet ci-dessous, que je vais expliquer en général.

```php
public function checkPath(string $requestPath, string $requestMethod)
{
  foreach ($this->paths as $path) {
    $Pregpath = preg_replace('#({[\w]+})#', '([^/]+)', $path['path']);
    preg_match('#\{.+}#', $path['path'], $pathParameters); // obtenir les différent nom des paramètres

    foreach($pathParameters as $parameter)
    {
      $pathParameters = explode('/', $parameter); // Mettre chaque paramètres dans un tableau et enlevant le "/" car sinon nous aurons
      // Enlève les {} pour obtenir les noms des paramètres en clair
      $pathParameters = str_replace('{', '', $pathParameters);
      $pathParameters = str_replace('}', '', $pathParameters);
    }

    $PathToMatch = "#^$Pregpath$#";

    if(preg_match($PathToMatch, $requestPath, $matches) && $path['http_method'] == $requestMethod)
    {
      array_shift($matches); // Suppression du premier index qui équivaut à l'URL
      if(isset($matches))
      {
        // Pour chaque paramètres dynamique du path, on récupère sa valeur et on l'injecte dans la variables des paramètres à injecter
        for($i = 0; $i < count($pathParameters); $i++)
        {
          $this->params[$pathParameters[$i]] = $matches[$i];// ex : 'id' => '4'
        }
        return $path;
      }
    }
  }
}
```

On peut voir quelques lignes ont été rajoutées, je vais premièrement faire un preg_match pour obtenir les différents noms des paramètres. Ensuite, je vais les découper dans un tableau pour seulement obtenir les noms simples sans les accolades etc.. Je passe rapidement sur l'explication parce que j'ai commenté dans l'ensemble le code.
Après avoir obtenu un tableau avec les noms des paramètres dynamiques, il me fallait juste insérer la valeur du paramètre dynamique pour l'index du paramètre en question. On peut voir que je réitère l'opération pour chaque paramètre dynamique, ce qui veut dire que cela laisse la possibilité d'avoir un nombre de paramètres dynamique illimité.

Après avoir testé ce code, je me suis aperçu que j'obtenais à chaque fois une erreur avec le getName des paramètres injectés, il arrivais pas à obtenir le titre. J'ai donc changé la ligne du $typeName pour contre passer cette erreur.

```php
$paramName = $param->getName();
$typeName = $param->getType() ? $param->getType()->getName() : $paramName; // si aucun type défini, considéré comme un atribut de l
```

Résultat obtenues

Après avoir codé tout ce que j'ai expliqué précédemment, il fallait juste que je renseigne le nom du paramètre dans les attributs de la fonction du controller.

Voici les paramètres attendus de la route.

```php
$router->addPath(
    '/contact/edit/{id}/{name}',
    RequestMethod::GET,
    'contact',
    HomeController::class,
    'contact'
);
```

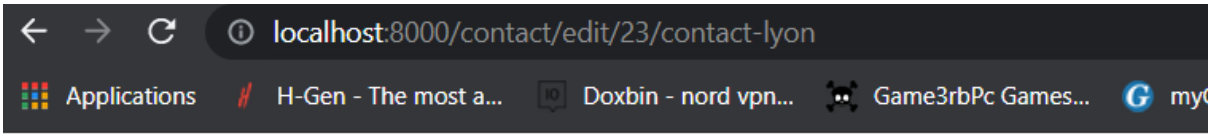
Voici les paramètre rempli dans la fonction du controller

```php
public function contact(EntityManager $em, $id, $name)
{
    echo $this->twig->render('contact.html.twig', ['id' => $id, 'name' => $name]);
}
```

Voici le code HTML pour obtenir la valeur des paramètres envoyés

```
</form>
Valeur ID: {{ id }}
<br>
Valeur name: {{ name }}
{% endblock %}
```

Voici le résultat de la vue en remplissant les paramètre dynamique



Contact

Nom : Message : Envoyer

Valeur ID: 23

Valeur name: contact-lyon

Bonus - Doctrine

Il est impossible de lancer la commande php vendor/bin/doctrine pour pouvoir utiliser les commandes doctrine.

Pour corriger cette erreur, il suffit de vérifier à l'exécution des routes si nous sommes sous le terminal ou pas.

```php
// Sinon impossible de lancer la commande php vendor/bin/doctrine
if(php_sapi_name() != 'cli')
  $router->execute($_SERVER['REQUEST_URI'], $_SERVER['REQUEST_METHOD']);
```