2 minutes

Host et préfixes

L'option `host`

Il est possible de définir une option `host` pour que l'hôte soit pris en compte lorsqu'une requête est examinée par le `Router`.

L'hôte est la partie de l'`URL` contenant le nom de domaine. Par exemple, `google.fr` ou `docs.google.com`.

Un exemple d'utilisation est :

```
<?php

namespace App\Controller;

use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class DefaultController
{
  #[Route('/', name: 'mobile', host: 'm.dyma.fr')]
  public function mobileHomepage()
  {
    // ...
  }

  #[Route('/', name: 'homepage')]
  public function homepage()
  {
    // ...
  }
}
```

Copier

Si vous avez une version mobile en production, vous pouvez modifier les méthodes des contrôleurs qui sont exécutées (les actions), en fonction du nom de l'hôte.

C'est une fonctionnalité assez avancée dont vous n'aurez pas besoin avant une mise en production avec des sous-domaines. L'important est simplement de savoir que cela existe.

Groupes de routes et préfixes

Il est commun dans une application Web d'avoir des groupes de routes correspondant à des ressources.

Par exemple, `/users/quelquechose` pour toutes les routes relatives aux utilisateurs.

`Symfony` permet donc de faciliter la déclaration de routes en ajoutant automatiquement un préfixe à l'ensemble des routes d'un contrôleur.

Pour ce faire, il suffit d'utiliser un attribut au-dessus de la classe du contrôleur.

En reprenant notre exemple avec les routes `blogName` et `blogId`, nous pouvons donc écrire :

```
<?php
```

```
namespace App\Controller;

use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

#[Route(path: '/blog')]
class DefaultController
{
  #[Route(path: '/{id}', name: 'blogId', requirements: ['id' => '\d+'])]
  public function blogById(int $id)
  {
    return new Response('<h1>Blog ID</h1>' . $id);
  }

  #[Route(path: '/{name}', name: 'blogName')]
  public function blogByName(string $name = 'all')
  {
    return new Response('<h1>Blog NAME</h1>' . $name);
  }
}
```

Copier

Notez bien l'attribut `#[Route(path: '/blog')]` au dessus de `class DefaultController` : toutes les routes dans la classes seront automatiquement préfixées par `/blog` !

Cette fonctionnalité est très utile et régulièrement utilisée.

En plus du chemin, vous pouvez préfixer d'autres paramètres de route, par exemple le nom :

```
<?php

namespace App\Controller;

use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

#[Route(path: '/blog', name: 'blog_')]
class DefaultController
{
  #[Route(path: '/{id}', name: 'id', requirements: ['id' => '\d+'])]
  public function blogById(int $id)
  {
    return new Response('<h1>Blog ID</h1>' . $id);
  }

  #[Route(path: '/{name}', name: 'name')]
  public function blogByName(string $name = 'all')
  {
    return new Response('<h1>Blog NAME</h1>' . $name);
  }
}
```

Copier

Toutes les routes auront leur nom automatiquement préfixé par `blog`.

Les noms de nos deux routes seront donc `blog_id` et `blog_name`.

Vous pouvez le vérifier en faisant :

```
symfony console debug:router
```

Copier