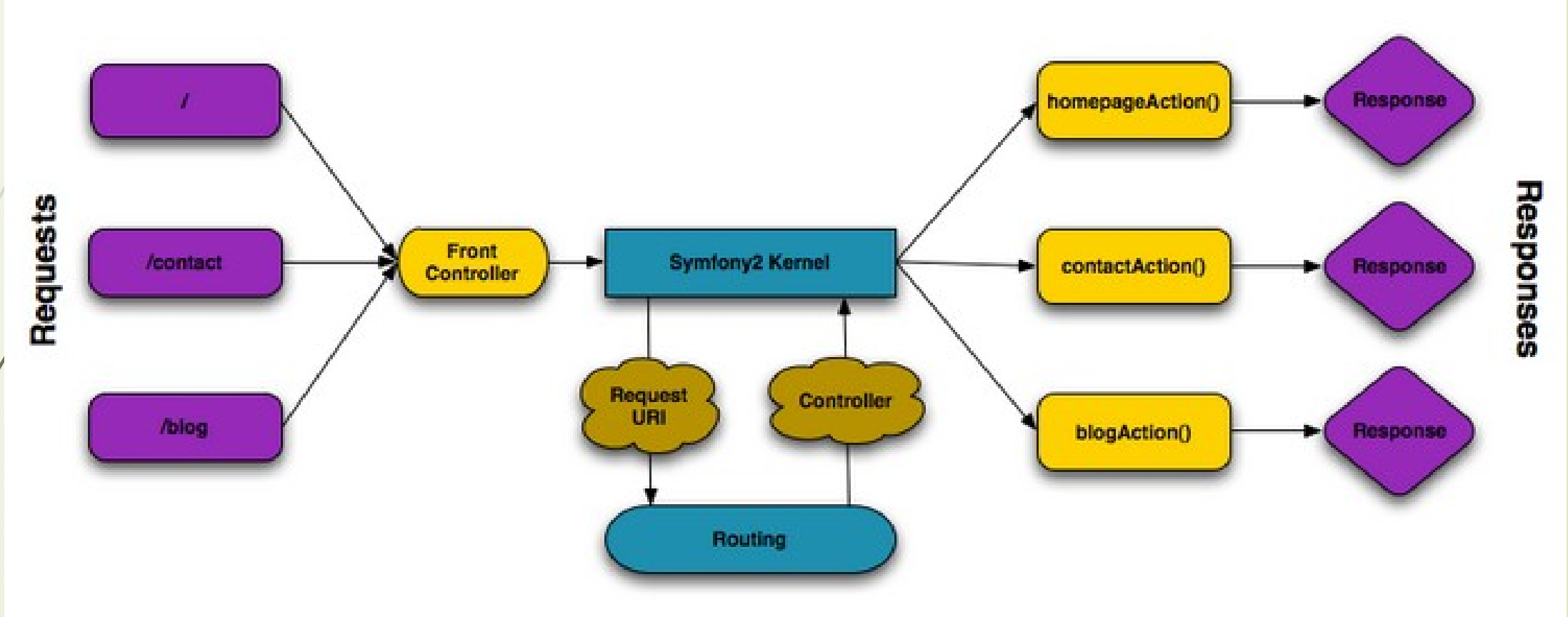
Le Routage

Lotfi KHEDIRI

Introduction

- Lorsque votre application reçoit une demande, elle exécute une *action du contrôleur* pour générer la réponse.
- La configuration de routage définit **l'action** à exécuter pour chaque **URL entrante**.
- Il fournit également d'autres fonctionnalités utiles.
- Comme la génération d'URL optimisées pour le référencement (par exemple /read/intro-to-symfony au lieu de index.php?article_id=57). (les slugs)

Principe du routage



Création de Routes

- Les routes peuvent être configurés en YAML, XML, PHP ou en utilisant des annotations.
- Tous les formats offrent les mêmes fonctionnalités et performances
- Symfony recommande les annotations car il est pratique de placer la route et le contrôleur au même endroit.

Route avec annotation

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\
AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class BlogController extends AbstractController
{
    /**
     * @Route("/blog", name="blog_list")
     */
    public function list()
    {
        // ...
    }
}
```

Route avec YAML

YAML | XML | PHP

```yaml
# config/routes.yaml
blog_list:
    path: /blog
    # the controller value has the format 'controller_class::method_name'
    controller: App\Controller\BlogController::list

    # if the action is implemented as the __invoke() method of the
    # controller class, you can skip the '::method_name' part:
    # controller: App\Controller\BlogController
```

Route avec XML

YAML | XML | PHP

```xml
<!-- config/routes.xml -->
<?xml version="1.0" encoding="UTF-8" ?>
<routes xmlns="http://symfony.com/schema/routing"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://symfony.com/schema/routing
        https://symfony.com/schema/routing/routing-1.0.xsd">

    <!-- the controller value has the format 'controller_class::method_name' -->
    <route id="blog_list" path="/blog"
           controller="App\Controller\BlogController::list"/>

    <!-- if the action is implemented as the __invoke() method of the
         controller class, you can skip the '::method_name' part:
         controller="App\Controller\BlogController"/> -->
</routes>
```

Route avec PHP

YAML XML PHP

```php
// config/routes.php
use App\Controller\BlogController;
use Symfony\Component\Routing\Loader\Configurator\RoutingConfigurator;

return function (RoutingConfigurator $routes) {
    $routes->add('blog_list', '/blog')
        // the controller value has the format [controller_class, method_name]
        ->controller([BlogController::class, 'list'])

        // if the action is implemented as the __invoke() method of the
        // controller class, you can skip the ', method_name]' part:
        // ->controller([BlogController::class])
    ;
};
```

Débogage des routes

- Symfony inclut quelques commandes pour aider à déboguer les problèmes de routage

```
$ php bin/console debug:router

---------------  -------  -------  -----  ----------------------------------------
Name             Method   Scheme   Host   Path
---------------  -------  -------  -----  ----------------------------------------
homepage         ANY      ANY      ANY    /
contact          GET      ANY      ANY    /contact
contact_process  POST     ANY      ANY    /contact
article_show     ANY      ANY      ANY    /articles/{_locale}/{year}/{title}.{_format}
blog             ANY      ANY      ANY    /blog/{page}
blog_show        ANY      ANY      ANY    /blog/{slug}
---------------  -------  -------  -----  ----------------------------------------
```

Débogage des routes

Passez le nom (ou une partie du nom) d'une route à cet argument pour imprimer les détails de la route

```
$ php bin/console debug:router app_lucky_number
```

La commande **router:match** montre quelle route correspondra à l'URL donnée

```
$ php bin/console router:match /lucky/number/8

  [OK] Route "app_lucky_number" matches
```

Paramètres de route

- il est courant de définir des routes où certaines parties sont variables (les **paramètres**)
- Les paramètres sont entourés par **{ ... }**
- Elles doivent avoir un nom **unique**
- Les routes peuvent définir n'importe quel nombre de paramètres

```
/**
 * @Route("/blog/{id}", name="blog_show")
 */
public function show($id)
{
    // ...
    // ...
}
```

Validation des paramètres

- Imaginez qu'une application possède ces deux routes : **/blog/{nom}** et **/blog/{page}**.
- Il n'y a aucun moyen de différencier les deux routes.
- Symfony utilisera la route qui a été définie en premier
- Pour résoudre ce problème -> validation du paramètre {page}

```
/**
 * @Route("/blog/{page}", name="blog_list", requirements={"page"="\d+"} )
 */
public function list(int $page)
{    // ...    }
/**
 * @Route("/blog/{slug}", name="blog_show")
 */
public function show($slug)
{     // ...    }
```

- L'option requirements définit les expressions régulières PHP auxquelles les paramètres de route doivent correspondre pour que la route entière corresponde

Paramètre optionnel

```
class BlogController extends AbstractController
{
    /**
     * @Route("/blog/{page}", name="blog_list",
requirements={"page"="\d+"})
     */
    public function list(int $page = 1)
    {
        // ...
    }
}
```

Préfixes et Groupes de routage

- Si par exemple, toutes les routes liées au blog commencent par /blog Symfony permet le partage d'une partie des routes.

```
/**
 * @Route("/blog",name="blog_")
 */
class BlogController
{
    /**
     * @Route("/", name="index")
     */
    public function index(){...}
    /**
     * @Route("/posts/{id}", name="show")
     */
    public function show(Post $post){...}
}
```

Obtenir le nom et les paramètres 'une route

- L' objet **Request** créé par Symfony stocke toute la configuration des routes (comme le nom et les paramètres) dans les "attributs de requête".
- On peut recupérer ces informations via l'objet **Request**

```
class BlogController extends AbstractController
{
    /**
     * @Route("/blog", name="blog_list")
     */
    public function list(Request $request)
    {
        // ...
        $routeName = $request->attributes->get('_route');
        $routeParameters = $request->attributes->get('_route_params');

        // use this to get all the available attributes (not only routing ones):
        $allAttributes = $request->attributes->all();
    }
}
```

Générer des URL

- Pour générer une URL, spécifier le **nom de la route** (par exemple blog_show) et les **valeurs des paramètres** définis par la route (par exemple id=1)
- Chaque route a un nom qui doit être **unique** dans l'application
- Si le nom de la route n'est pas défini explicitement avec l'option **name** , Symfony génère un nom automatique basé sur le contrôleur et l'action.

Générer des URL dans les contrôleurs

```
 /**
*@Route("/blog", name="blog_list")
*/
public function list()
{
   // generate a URL with no route arguments

   $signUpPage = $this->generateUrl('sign_up');

   // generate a URL with route arguments

   $userProfilePage = $this->generateUrl('user_profile',
                      ['username' =>$user->getUsername(),]);
}
```

Générer des URL dans les Templates

- Utilisez la fonction Twig **path()** pour créer un **lien** vers des pages et passez
  - le nom de la route comme **premier argument** et
  - les paramètres de la route comme **deuxième argument** facultatif

Exemple de route sans paramètres

```
<a href="{{ path('blog_index') }}">Homepage</a>
```

Exemple de route avec paramètre

```
<a href="{{ path('blog_post', {id: post.id}) }}">{{ post.title }}</a>
```