Routing

Le composant **router** de symfony permet de solliciter la méthode d'un contrôleur, en lui passant éventuellement des paramètres, en fonction de l'url demandée.

Les routes peuvent être définies par l'intermédiaire :

- de l'annotation @Route
- en php
- en yaml
- en xml

La méthode la plus souple est l'utilisation d'annotations. Elle nécessite l'installation du composant annotations :

```
composer require annotations
```

Création

Création de routes avec l'annotation @Route

```
namespace App\Controller;

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

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

    /**
     * Matches /blog/*
     *
     * @Route("/blog/{slug}", name="blog_show")
     */
    public function show($slug)
    {
        // $slug will equal the dynamic part of the URL
        // e.g. at /blog/yay-routing, then $slug='yay-routing'

        // ...
    }
```

```
}
```

Dans la console, la commande suivante liste les routes existantes :

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

Paramètres de route et requirements

Les paramètres de routes sont passés à l'aide des accolades :

```
namespace App\Controller;

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

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

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

Dans l'exemple précédent, les 2 routes sont en conflit, puisqu'elles correspondent toutes les 2 à une url du type /blog/*.

Le router choisira toujours dans ce cas la première correspondance trouvée : la route **blog_list**, ce qui est problématique.

Il est possible d'affiner ce routage, en précisant pour la route **blog_list** que le paramètre **page** doit être un entier :

On ajoute dans ce cas un “requirement”, correspondant à une expression régulière :

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
```

```
use Symfony\Component\Routing\Annotation\Route;

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

    // ...
}
```

Valeur par défaut

Il est possible de donner à un paramètre une valeur par défaut :

```
namespace App\Controller;

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

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

Méthode HTTP

L'ajout de l'attribut **methods** à la route permet de spécifier la/les méthode(s) utilisables :

```
namespace App\Controller;

// ...

class BlogApiController extends Controller
{
    /**
     * @Route("/api/posts/{id}", methods={"GET","HEAD"})
     */
    public function show($id)
    {
```

```
        // ... return a JSON response with the post
    }

    /**
     * @Route("/api/posts/{id}", methods="PUT")
     */
    public function edit($id)
    {
        // ... edit a post
    }
}
```

Paramètres de routage spéciaux

4 paramètres spéciaux sont utilisable au niveau des routes :

_controller	Permet de spécifier le contrôleur à exécuter quand la route est sollicitée.
_format	Utilisé pour définir le format de la requête (html, json...) voir format param.
_fragment	Utilisé pour définir le fragment après le # dans le cas d'une url interne à la page
_locale	Utilisé pour définir la “locale” de la requête (fr/en...) voir locale

Exemple :

```
// ...
class ArticleController extends Controller
{
    /**
     * @Route(
     *     "/articles/{_locale}/{year}/{slug}.{_format}",
     *     defaults={"_format": "html"},
     *     requirements={
     *         "_locale": "en|fr",
     *         "_format": "html|rss",
     *         "year": "\d+"
     *     }
     * )
     */
    public function show($_locale, $year, $slug)
    {
    }
}
```

Génération d'URLs

Depuis un contrôleur

```
class MainController extends Controller
```

```
{
    public function show($slug)
    {
        // ...

        // /blog/my-blog-post
        $url = $this->generateUrl(
            'blog_show',
            array('slug' => 'my-blog-post')
        );
    }
}
```

URL avec paramètres

Les variables supplémentaires sont ajoutées en tant que query string :

```
$this->generateUrl('blog_list', array(
    'page' => 2,
    'category' => 'Symfony',
));
// /blog/2?category=Symfony
```

Génération depuis un service

Il est nécessaire d'injecter une instance de **UrlGeneratorInterface**

```
use Symfony\Component\Routing\Generator\UrlGeneratorInterface;

class SomeService
{
    private $router;

    public function __construct(UrlGeneratorInterface $router)
    {
        $this->router = $router;
    }

    public function someMethod()
    {
        $url = $this->router->generate(
            'blog_show',
            array('slug' => 'my-blog-post')
        );
        // ...
    }
}
```