

[illegible]index.php?article_id=57

URL

SEO ☐☐☐

URL

/read/intro-to-symfony



YAML XML PHP

[illegible]

```
attributes [ ] [ ] [ ] route [ ] controller [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
```

Symfony 



PHP []

PHP 8

Symfony Flex

```
# config/routes/attributes.yaml
```

controllers:

resource:

```
path: ../../src/Controller/
```

namespace: App\Controller

type: attribute

kernel:

resource: App\Kernel

type: attribute

Symfony  App\Controller 

PSR-4 [] src/Controller/ []

Symfony

[illegible]/blog URL

controller 

--	--

--	--

111

POS

PU*

ods

}

--	--	--

111

{

```

#[Route(
    '/contact',
    name: 'contact',
    condition: "context.getMethod() in ['GET', 'HEAD'] and request.headers.get('User-Agent') matches
'/firefox/i'",
    // expressions can also include config parameters:
    // condition: "request.headers.get('User-Agent') matches '%app.allowed_browsers%'"
)]
public function contact(): Response
{
    // ...
}

#[Route(
    '/posts/{id}',
    name: 'post_show',
    // expressions can retrieve route parameter values using the "params" variable
    condition: "params['id'] < 1000"
)]
public function showPost(int $id): Response
{
    // ... return a JSON response with the post
}
}

```

condition `context.getMethod() in ['GET', 'HEAD'] and request.headers.get('User-Agent') matches '/firefox/i'`

Symfony

- context RequestContext `request.getContext()`
- request `request` Symfony Request `$request`
- params `request->attributes->all()`

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

- env(string \$name) `$env` Environment Variable Processors `$env`
- service(string \$alias) `$services->get($alias)` `#[AsRoutingConditionService]` `$services->get($alias)`
`routing.condition_service` `$services->get($alias)` `use`
`Symfony\Bundle\FrameworkBundle\Routing\Attribute\AsRoutingConditionService;` `use`
`Symfony\Component\HttpFoundation\Request;`
`#[AsRoutingConditionService(alias: 'route_checker')] class RouteChecker { public function`
`check(Request $request): bool { // ... } }`

```

    service()  conditions  ...
// Controller (using an alias):
#[Route(condition: "service('route_checker').check(request)")]
// Or without alias:
#[Route(condition: "service('App\\Service\\RouteChecker').check(request)")]

```

PHP
condition
PHP

Symfony
debug
router
Symfony

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}
-----

```

php bin/console debug:router app_lucky_number

```

+-----+
| Property | Value |
+-----+
| Route Name | app_lucky_number |
| Path      | /lucky/number/{max} |
| ...      | ... |

```

```
| Options | compiler_class: Symfony\Component\Routing\RouteCompiler |
|         | utf8: true |
+-----+-----+
```

```
##### router match##### URL#### URL
#####
```

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

```
[OK] Route "app_lucky_number" matches
```

```
####
```

```
##### URL ##### /blog
##### URL ##### slug####
/blog/my-first-post | /blog/all-about-symfony|

| Symfony ##### { } ##### /blog/{slug}|
```

```
// src/Controller/BlogController.php
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Attribute\Route;

class BlogController extends AbstractController
{
    // ...

    #[Route('/blog/{slug}', name: 'blog_show')]
    public function show(string $slug): Response
    {
        // $slug will equal the dynamic part of the URL
        // e.g. at /blog/yay-routing, then $slug='yay-routing'

        // ...
    }
}
```

```
}  
}
```

{slug}##### PHP #####
/blog/my-first-post URL[] Symfony [] BlogController [] show[] [] \$slug = 'my-
first-post' [] show[] []

/blog/posts-about-
{category}/page/{pageNumber} []

####

blog_show [] [] URL[] /blog/{slug}[] [] blog_list [] [] URL[]
/blog/{page}#####

/blog/my-first-post##### Symfony
requirements [] {page}

```
// src/Controller/BlogController.php  
namespace App\Controller;  
  
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;  
use Symfony\Component\HttpFoundation\Response;  
use Symfony\Component\Routing\Attribute\Route;  
  
class BlogController extends AbstractController  
{  
    #[Route('/blog/{page}', name: 'blog_list', requirements: ['page' => '\d+'])]  
    public function list(int $page): Response  
    {  
        // ...  
    }  
  
    #[Route('/blog/{slug}', name: 'blog_show')]  
    public function show($slug): Response  
    {  
        // ...  
    }  
}
```

URLRoute Parameters

/blog/2blog_listpage = 2

/blog/my-first-postblog_showslug = my-first-post \$slug

{parameter_name} .

```
// src/Controller/BlogController.php
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Attribute\Route;

class BlogController extends AbstractController
{
    #[Route('/blog/{page<\d+>}', name: 'blog_list')]
    public function list(int $page): Response
    {
        // ...
    }
}
```

blog_list URL /blog/{page} /blog/1

/blog

{page}

/blog blog_list

controller defaults

```
// src/Controller/BlogController.php
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
```



```

use Symfony\Component\Routing\Attribute\Route;

class BlogController extends AbstractController
{
    #[Route('/blog/{page}', name: 'blog_list', requirements: ['page' => '\d+'])]
    public function list(int $page = 1): Response
    {
        // ...
    }
}

```

```

# /blog 1 blog_list 1 $page 1 1

# URL /blog/1 /blog
# /blog/{page}

# requirements {parameter_name?default_value}

```

```

// src/Controller/BlogController.php
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Attribute\Route;

class BlogController extends AbstractController
{
    #[Route('/blog/{page<\d+>?1}', name: 'blog_list')]
    public function list(int $page): Response
    {
        // ...
    }
}

```

Symfony

YAML XML

PHP

[illegible]

```
// src/Controller/BlogController.php
namespace App\Controller;

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

class BlogController extends AbstractController
{
    /**
     * This route has a greedy pattern and is defined first.
     */
    #[Route('/blog/{slug}', name: 'blog_show')]
    public function show(string $slug): Response
    {
        // ...
    }

    /**
     * This route could not be matched without defining a higher priority than 0.
     */
    #[Route('/blog/list', name: 'blog_list', priority: 2)]
    public function list(): Response
    {
        // ...
    }
}
```

0□



ID
"param converter"

controller `BlogPost $post` `$slug`

```
// src/Controller/BlogController.php
namespace App\Controller;
```

```
use App\Entity\BlogPost;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

use Symfony\Component\HttpFoundation\Response;

use Symfony\Component\Routing\Attribute\Route;

class BlogController extends AbstractController
{
    // ...

    #[Route('/blog/{slug}', name: 'blog_show')]
    public function show(BlogPost $post): Response
    {
        // $post is the object whose slug matches the routing parameter

        // ...
    }
}
```

[illegible]

BlogPost[] "param converter"

slug

Symfony ☐ ☐ ☐ ☐ ☐

404

--	--	--

▣▣ Doctrines param conversion ▣▣▣▣

#[MapEntity]

route

--	--	--	--	--	--	--

□□□□ PHP □□□□□□□□□□□□□□

[illegible]

```
// src/Controller/OrderController.php
```

```
namespace App\Controller;
```

```
use App\Enum\OrderStatusEnum;
```

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

```
use Symfony\Component\HttpFoundation\Response;
```

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

```
class OrderController extends AbstractController
```

{

```
#[Route('/orders/list/{status}', name: 'list_orders_by_status')]
public function list(OrderStatusEnum $status = OrderStatusEnum::Paid): Response
{
    // ...
}
```



Symfony

- _controller [] Request
- _format [] Content-Type
[] json [] application/json [] Content-Type
- _fragment [] URL [] #
[]
- _locale []

_fragment Symphony

[illegible]

```
// src/Controller/ArticleController.php

namespace App\Controller;

// ...

class ArticleController extends AbstractController
{
    #[Route(
        path: '/articles/{_locale}/search.{_format}',
        locale: 'en',
        format: 'html',
        requirements: [
            '_locale' => 'en|fr',
            '_format' => 'html|xml',
        ],
    )]
    public function search(): Response
    {
    }
}
```



defaults



```
// src/Controller/BlogController.php
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Attribute\Route;

class BlogController extends AbstractController
{
    #[Route('/blog/{page}', name: 'blog_index', defaults: ['page' => 1, 'title' => 'Hello world!'])]
    public function index(int $page, string $title): Response
    {
        // ...
    }
}
```



/ URL

/share/{token} token /

parameter requirements

```
// src/Controller/DefaultController.php
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Attribute\Route;

class DefaultController extends AbstractController
{
    #[Route('/share/{token}', name: 'share', requirements: ['token' => '.+'])]
    public function share($token): Response
```

```
{  
    // ...  
}  
}
```



Route alias

```
# config/routes.yml  
new_route_name:  
  alias: original_route_name
```

original_route_name new_route_name



```
new_route_name:  
  alias: original_route_name  
  
# this outputs the following generic deprecation message:  
# Since acme/package 1.2: The "new_route_name" route alias is deprecated. You should stop using it, as it  
will be removed in the future.  
deprecated:  
  package: 'acme/package'  
  version: '1.2'  
  
# you can also define a custom deprecation message (%alias_id% placeholder is available)  
deprecated:  
  package: 'acme/package'  
  version: '1.2'  
  message: 'The "%alias_id%" route alias is deprecated. Do not use it anymore.'
```

new_route_name

```
%alias_id% [ ]
```

--	--	--	--	--	--

/blog

Symfony

[illegible]

controller \sqcap $\#[\text{Route}]$

```

index  blog_index  URL  /blog/{_locale}  show
blog_show  URL  /blog/{_locale}/posts/{slug}  _locale
class

```



Symfony  Request  name  parameters  "request
attributes"  Request 

```
// src/Controller/BlogController.php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Attribute\Route;

class BlogController extends AbstractController
{
    #[Route('/blog', name: 'blog_list')]
    public function list(Request $request): Response
    {
        $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();

        // ...
    }
}
```

```

services [ ] RequestStack [ ] Twig [ ] app
[ ] app.current_route [ ] app.current_route_parameters [ ]

```



Symfony



[[[Symfony]]] route []]

[[[[[[[[[[URL []]

[] RedirectController []] URL[]

```
# config/routes.yaml
doc_shortcut:
  path: /doc
  controller: Symfony\Bundle\FrameworkBundle\Controller\RedirectController
  defaults:
    route: 'doc_page'
    # optionally you can define some arguments passed to the route
    page: 'index'
    version: 'current'
    # redirections are temporary by default (code 302) but you can make them permanent (code 301)
    permanent: true
    # add this to keep the original query string parameters when redirecting
    keepQueryParams: true
    # add this to keep the HTTP method when redirecting. The redirect status changes
    # * for temporary redirects, it uses the 307 status code instead of 302
    # * for permanent redirects, it uses the 308 status code instead of 301
    keepRequestMethod: true
    # add this to remove all original route attributes when redirecting
    ignoreAttributes: true
    # or specify which attributes to ignore:
    # ignoreAttributes: ['offset', 'limit']

legacy_doc:
  path: /legacy/doc
  controller: Symfony\Bundle\FrameworkBundle\Controller\RedirectController
  defaults:
    # this value can be an absolute path or an absolute URL
    path: 'https://legacy.example.com/doc'
    permanent: true
```

[[[[[[[[[[URL

[[[[[URL [] UNIX [] https://example.com/foo/
[[[[[[[] https://example.com/foo[]] URL

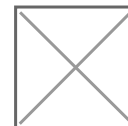
URL

URL

Symfony

URL

GET HEAD



--	--	--	--

host

HTTP

[illegible]

11

```
// src/Controller/MainController.php
```

```
namespace App\Controller;
```

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

```
use Symfony\Component\HttpFoundation\Response;
```

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

```
class MainController extends AbstractController
```

{

```
#[Route('/', name: 'mobile homepage', host: 'm.example.com')]
```

public function mobileHomepage(): Response

{

// ...

}

```
#[Route('/', name: 'homepage')]
```

```
public function homepage(): Response
```

{

// ...

}

}

host

```
// src/Controller/MainController.php
```



```
namespace App\Controller;
```


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


```
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Attribute\Route;
```

```
class MainController extends AbstractController
{
  #[Route(
    '/',
    name: 'mobile_homepage',
    host: '{subdomain}.example.com',
    defaults: ['subdomain' => 'm'],
    requirements: ['subdomain' => 'm|mobile'],
  )]
  public function mobileHomepage(): Response
  {
    // ...
  }

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

subdomain 

subdomain 

URL

☐ i18n ☐

URL

```
// src/Controller/CompanyController.php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Attribute\Route;
```

```
class CompanyController extends AbstractController
{
  #[Route(path: [
    'en' => '/about-us',
    'nl' => '/over-ons'
  ], name: 'about_us')]
  public function about(): Response
  {
    // ...
  }
}
```

Symfony

HTTP

#####

```
// src/Controller/MainController.php
namespace App\Controller;

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

class MainController extends AbstractController
{
    #[Route('/', name: 'homepage', stateless: true)]
    public function homepage(): Response
    {
        // ...
    }
}
```

kernel.debug

- enabled UnexpectedSessionUsageException
- disabled

#####

URL

#####

- URL ;
- URL

URL HTML [#####](“...”>#####)

 URL blog_show slug = my-blog-post

name
Symfony

```
// src/Controller/MainController.php

namespace App\Controller;

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

final class MainController extends AbstractController
{
    #[Route('/', name: 'homepage')]
    public function homepage(): Response
    {
        // ...
    }
}
```

Symfony ☐ App\Controller\MainController::homepage.

□ Controller □□□ URL

AbstractController

--	--	--	--	--	--	--	--

generateUrl□□ □□□□

```
// src/Controller/BlogController.php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Attribute\Route;
use Symfony\Component\Routing\Generator\UrlGeneratorInterface;

class BlogController extends AbstractController
{
    #[Route('/blog', name: 'blog_list')]

    public function list(): Response
    {
        // 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->getUserIdentifier(),
]);

// generated URLs are "absolute paths" by default. Pass a third optional
// argument to generate different URLs (e.g. an "absolute URL")
$signUpPage = $this->generateUrl('sign_up', [], UrlGeneratorInterface::ABSOLUTE_URL);

// when a route is localized, Symfony uses by default the current request locale
// pass a different '_locale' value if you want to set the locale explicitly
$signUpPageInDutch = $this->generateUrl('sign_up', ['_locale' => 'nl']);

// ...
}
}

```

```

class AbstractController
{
}

```

Services and URL

```

class SomeService
{
    public function generate()
    {
    }
}

```

UrlGeneratorInterface

```

// src/Service/SomeService.php
namespace App\Service;

use Symfony\Component\Routing\Generator\UrlGeneratorInterface;

class SomeService
{
    public function __construct(
        private UrlGeneratorInterface $router,
    ) {
    }

    public function someMethod(): void
    {
        // ...
    }
}

```

```

// generate a URL with no route arguments
$signUpPage = $this->router->generate('sign_up');

// generate a URL with route arguments
$userProfilePage = $this->router->generate('user_profile', [
    'username' => $user->getUserIdentifier(),
]);

// generated URLs are "absolute paths" by default. Pass a third optional
// argument to generate different URLs (e.g. an "absolute URL")
$signUpPage = $this->router->generate('sign_up', [], UrlGeneratorInterface::ABSOLUTE_URL);

// when a route is localized, Symfony uses by default the current request locale
// pass a different '_locale' value if you want to set the locale explicitly
$signUpPageInDutch = $this->router->generate('sign_up', ['_locale' => 'nl']);
}
}

```

Twig URL

Twig Symfony Twig

JavaScript Twig URL

Twig JavaScript Twig path Twig url Twig URL
Twig JavaScript escape Twig JavaScript

```

<script>
    const route = "{ { path('blog_show', {slug: 'my-blog-post'})|escape('js') } }";
</script>

```

Twig URL Twig JavaScript Twig
FOSJsRoutingBundle

Twig URL

Twig URL Twig URL HTTP
Twig URL http://localhost/
Twig

Twig default_uri Twig URL “request context”


```
# config/packages/routing.yaml
framework:
  router:
    # ...
    default_uri: 'https://example.org/my/path/'
```

■■■■■■■

URL ■■■■■■■■

```
// src/Command/SomeCommand.php
namespace App\Command;

use Symfony\Component\Console\Command\Command;
use Symfony\Component\Console\Input\InputInterface;
use Symfony\Component\Console\Output\OutputInterface;
use Symfony\Component\Routing\Generator\UrlGeneratorInterface;
// ...

class SomeCommand extends Command
{
    public function __construct(private UrlGeneratorInterface $urlGenerator)
    {
        parent::__construct();
    }

    protected function execute(InputInterface $input, OutputInterface $output): int
    {
        // generate a URL with no route arguments
        $signUpPage = $this->urlGenerator->generate('sign_up');

        // generate a URL with route arguments
        $userProfilePage = $this->urlGenerator->generate('user_profile', [
            'username' => $user->getUserIdentifier(),
        ]);

        // by default, generated URLs are "absolute paths". Pass a third optional
        // argument to generate different URIs (e.g. an "absolute URL")
        $signUpPage = $this->urlGenerator->generate('sign_up', [], UrlGeneratorInterface::ABSOLUTE_URL);

        // when a route is localized, Symfony uses by default the current request locale
```

```
// pass a different '_locale' value if you want to set the locale explicitly
$signUpPageInDutch = $this->urlGenerator->generate('sign_up', ['_locale' => 'nl']);

// ...
}
}
```

```


```

```

URL
getRouteCollection()

```

```

URL RouteNotFoundException

```

```
use Symfony\Component\Routing\Exception\RouteNotFoundException;

// ...

try {
    $url = $this->router->generate($routeName, $routeParameters);
} catch (RouteNotFoundException $e) {
    // the route is not defined...
}
```

```

URL HTTPS

```

```

URL HTTP HTTP URL
http getContext()

```

```
# config/services.yaml
parameters:
    router.request_context.scheme: 'https'
    asset.request_context.secure: true
```

```

schemes








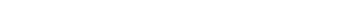


```

```
// src/Controller/SecurityController.php
namespace App\Controller;

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

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

```
class SecurityController extends AbstractController
{
  #[Route('/login', name: 'login', schemes: ['https'])]
  public function login(): Response
  {
    // ...
  }
}
```


 URL 
 HTTPS 
 path 
 Twig 
 URL 

 HTTP 
 URL 
 URL 

```
{# if the current scheme is HTTPS, generates a relative URL: /login #}
{{ path('login') }}

{# if the current scheme is HTTP, generates an absolute URL to change
the scheme: https://example.com/login #}
{{ path('login') }}
```

scheme HTTP /login URL
 URL HTTPS

Diagram illustrating the structure of a data stream:

- Top row: 8 blocks followed by the text "HTTPS".
- Bottom row: 24 blocks followed by the text "HTTPS".

```
# config/routes/attributes.yaml
controllers:
  resource: '../src/Controller/'
  type: attribute
  schemes: [https]
```

URI □□□□

The diagram shows a mapping from multiple URIs to a single URI. The top row contains four boxes, each labeled 'URI' followed by a box of varying size (4, 4, 12, and 4 boxes respectively). The bottom row contains a single large box labeled 'URI' followed by a box of 20 boxes. A vertical arrow points from the first box of the top row to the first box of the bottom row. A horizontal arrow points from the second box of the top row to the second box of the bottom row. A horizontal arrow points from the third box of the top row to the third box of the bottom row. A horizontal arrow points from the fourth box of the top row to the fourth box of the bottom row.

Symfony [] UriSigner [] URI []


```
use Symfony\Component\HttpFoundation\UriSigner;
```

```
class SomeService
```

```
{  
    public function __construct(  
        private UriSigner $uriSigner,  
    ) {  
    }  
  
    public function someMethod(): void  
    {  
        // ...  
  
        // generate a URL yourself or get it somehow...  
        $url = 'https://example.com/foo/bar?sort=desc';  
  
        // sign the URL with an explicit expiration date  
        $signedUrl = $this->uriSigner->sign($url, new \DateTimeImmutable('2050-01-01'));  
        // $signedUrl = 'https://example.com/foo/bar?sort=desc&_expiration=2524608000&_hash=e4a21b9'  
  
        // if you pass a \DateInterval, it will be added from now to get the expiration date  
        $signedUrl = $this->uriSigner->sign($url, new \DateInterval('PT10S')); // valid for 10 seconds from now  
        // $signedUrl = 'https://example.com/foo/bar?sort=desc&_expiration=1712414278&_hash=e4a21b9'  
  
        // you can also use a timestamp in seconds  
        $signedUrl = $this->uriSigner->sign($url, 4070908800); // timestamp for the date 2099-01-01  
        // $signedUrl = 'https://example.com/foo/bar?sort=desc&_expiration=4070908800&_hash=e4a21b9'  
    }  
}
```



Controller "App\\Controller\\BlogController::show()" requires that you provide a value for the "\$slug" argument.



\$slug

```
public function show(string $slug): Response
{
    // ...
}
```

```

[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] {slug} [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] /blog/show[ ] [ ] {slug} [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
/blog/show/{slug} [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] $slug = null[ ] [ ]

```

Some mandatory parameters are missing ("slug") to generate a URL for route "blog_show".

```

XXXXXXXXXXXXXXXXX      blog_show XXXX      URLXXXXXXXXX      slug
XXXXXXXXXXXXXXXXXXXXXXXXX      {slug} XXXXXXXXXXXXXXXXXXXX      slug XXX

```

```
$this->generateUrl('blog_show', ['slug' => 'slug-value']);
```

or, in Twig: `[[[[]]]]` Twig `[[[]]]`

```
{{ path('blog_show', {slug: 'slug-value'}) }}
```

Revision #1
Created 20 November 2024 00:36:55 by David Wang
Updated 20 November 2024 01:23:40 by David Wang

Created 20 November 2024 00:36:55 by David Wang

Created 20 November 2024 00:36:55 by David Wang
Updated 20 November 2024 01:23:40 by David Wang