



## DEFINING ROUTES

The order of the routes is important!

Earlier routes always win

### Where?

All routes are loaded via a single configuration file - usually `app/config/routing.yml`  
You can change this through the `app/config/config.yml` file:

```
framework:
  # ...
  router: { resource: "%kernel.root_dir%/config/routing.yml" }
```

### Defining your routes

YAML

`MyBundle/Resources/config` - you can create your routes here and import the file in the `app/config/routing.yml`:

```
acme_hello:
  resource: "@AcmeHelloBundle/Resources/config/routing.yml"
```

You can also create several route files in `MyBundle/Resources/config/routing` and import them in your `MyBundle/Resources/config/routing.yml`

Annotation

To use route annotation, you have to import routes from a controller class:

```
# app/config/routing.yml
post:
  resource: "@SensioBlogBundle/Controller/BlogController.php"
  type: annotation
```

### Prefixing Imported Routes

You can also choose to provide a "prefix" for the imported routes

YAML

```
# app/config/routing.yml
acme_hello:
  resource: "@AcmeHelloBundle/Resources/config/routing.yml"
  prefix: /admin
```

The string `/admin` will now be prepended to the pattern of each route loaded from the new routing resource.

Annotation

A `@Route` annotation on a controller class defines a prefix for all action routes:

```
/**
 * @Route("/blog")
 */
class PostController extends Controller{
  /**
   * @Route("/{id}")
   */
  public function showAction($id){
  }
}
```

### How?

The route below match: `http://www.myurl.com/blog/en/2`

```
blog:
  path: /blog/{culture}/{page}
  defaults: { _controller: AcmeBlogBundle:Blog:index, page: 1 }
  requirements:
    page: \d+
    culture: en|pt
  methods: [GET]
```

**Placeholders:** Required by default. Set a default value to make it optional  
in this example: culture - required, page - optional

**Name of the route:** `blog:`

**Pattern to match:** `path: /blog/{culture}/{page}`  
older versions: `pattern`

**Regex requirement (only digits) to the page parameter:** `requirements: page: \d+`

**Controller to be executed (format: Bundle:Controller:method):** `defaults: { _controller: AcmeBlogBundle:Blog:index, page: 1 }`

**Requirement to the culture parameter accepts only the en and pt values:** `requirements: culture: en|pt`

**Requirement to the method:** `methods: [GET]`  
older versions: `_method` - This route accepts only the GET method

Each routing parameter or default value is available as an argument in the controller method.

### Special Routing Parameters

<code>_controller</code>	determine which controller is executed when the route is matched
<code>_format</code>	set the request format (html, rss, json, ...)
<code>_locale</code>	set the locale on the request (the value will also be stored on the session)
<code>methods</code>	set the request method (GET, POST, PUT, DELETE, HEAD)

### Adding a Host

```
user_homepage:
  path: /
  host: "{user}.example.com"
  defaults: { _controller: AcmeDemoBundle:User:profile }

main_homepage:
  path: /
  defaults: { _controller: AcmeDemoBundle:Main:homepage }
```

Annotation

```
class BlogController extends Controller{
  /**
   * @Route("/blog/{culture}/{page}", requirements={"page" = "\d+"}, defaults={"page" = 1}, name="blog")
   * @Method({"GET"})
   */
  public function indexAction($page){
  }
}
```

maps a route pattern with a controller

HTTP method. Only considered when the `@Route` annotation is present

or

```
public function indexAction($page = 1){
}
```

Namespaces which the controller should have:

```
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Method;
```



### USING ROUTES

#### Controller

##### Relative URL

```
$url_index = $this->get('router')->generate('blog', array('culture' => 'en', 'page' => 2)); // /blog/en/2
$url_show = $this->get('router')->generate('blog_show', array('slug' => 'my-blog-post')); // /blog/my-blog-post
```

Name of the route

##### Absolute URL

```
$url_index = $this->get('router')->generate('blog', array('culture' => 'en', 'page' => 2), true);
// http://www.myurl.com/blog/en/2
$url_show = $this->get('router')->generate('blog_show', array('slug' => 'my-blog-post'), true);
// http://www.myurl.com/blog/my-blog-post
```

To generate an absolute URL, pass **true** to the third argument

#### Template

##### Relative URL

```
Twig {{ path('blog', {'culture': 'en', 'page': 2}) }}
      {{ path('blog_show', {'slug': 'my-blog-post'}) }}
PHP  echo $view['router']->generate('blog', array('culture' => 'en', 'page' => 2))
      echo $view['router']->generate('blog_show', array('slug' => 'my-blog-post'))
```

##### Absolute URL

```
Twig {{ url('blog', {'culture': 'en', 'page': 2}) }}
      {{ url('blog_show', {'slug': 'my-blog-post'}) }}
PHP  echo $view['router']->generate('blog', array('culture' => 'en', 'page' => 2), true)
      echo $view['router']->generate('blog_show', array('slug' => 'my-blog-post'), true)
```

### EXTRA

#### FOSJsRoutingBundle

<https://github.com/FriendsOfSymfony/FOSJsRoutingBundle>

Expose your Symfony2 routing in your JavaScript code.

In the routes you want to use in your JS, you have to add the expose option:

```
YAML route_to_expose:
      pattern: /blog/{page}
      defaults: { _controller: AcmeBlogBundle:Blog:index }
      options:
        expose: true
Annotation /**
            * @Route("/blog/{page}", name="route_to_expose", options={"expose"=true})
            */
            public function indexAction($page)
```

Using:

```
<script src="{ asset('bundles/fosjsrouting/js/router.js') }"></script>
<script src="{ path('fos_js_routing_js', {"callback": "fos.Router.setData"}) }"></script>
```

##### Relative URL

```
Routing.generate('route_to_expose', { page: 2 });
// /blog/2
```

##### Absolute URL

```
Routing.generate('route_to_expose', { page: 2 }, true);
// http://www.myurl.com/blog/2
```

#### Command Line

```
$ php app/console router:debug           debug and visualize routes
$ php app/console router:debug blog     debug and visualize a single route
$ php app/console router:match /blog/my-latest-post test which route match the given url
```