



Création de la première page

Sources :

http://symfony.com/fr/doc/current/book/http_fundamentals.html

Avertissement



Vous adapterez vos chemins et URL en fonction de la solution choisie

...



Les règles de nommage :

La classe correspondante doit avoir

- Comme préfixe, le nom du contrôleur
- Comme suffixe le mot Controller

De plus, pour permettre l'autochargement des classes, le nom du fichier doit être le même que le nom de la classe.

Le nom de la méthode action est composée du nom de l'action concaténé avec le mot clé Action()

Exemple : `afficheAction()`

Partie 1 : Objectifs

Créer une page qui dit Bonjour

Dans un nouveau projet TutoSymfonyBR – utilisation du bundle AppBundle

url de la page : http://symfony.br/TutoSymfonyBR/web/app_dev.php/mapagebonjour

1. Démarche :

Pour en revenir à notre mission, on va :

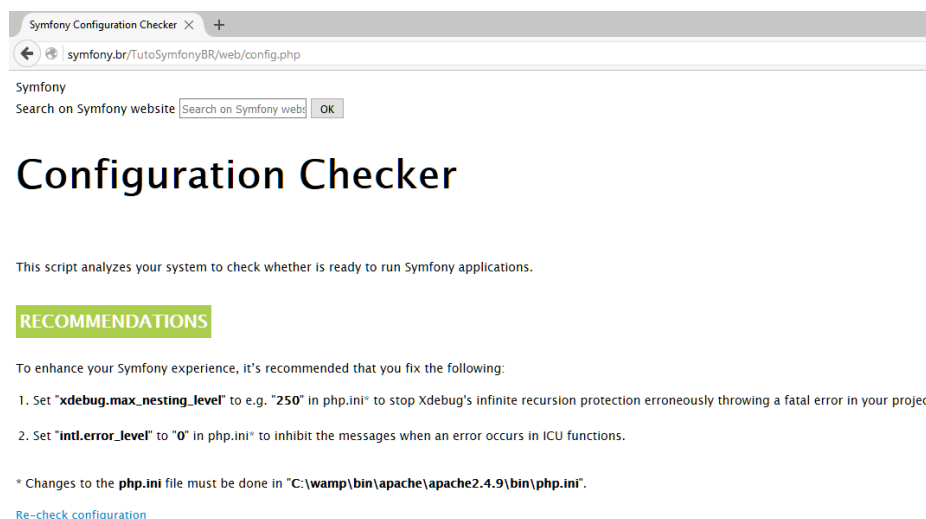
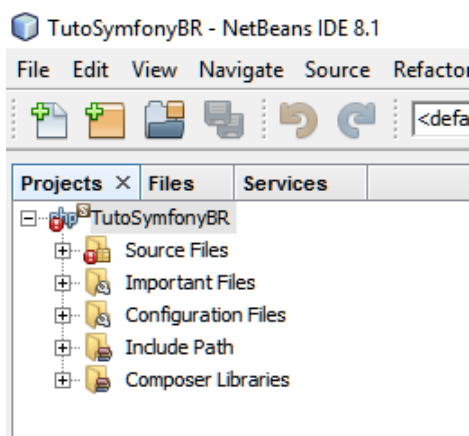
1. Créer le projet TutoSymfonyBR
2. Ouvrir ce projet sous netbeans
3. Vérifier que le routeur de l'application va chercher ses routes grâce aux annotations des méthodes Action contenues dans les contrôleurs du bundle AppBundle (fichier `.../app/config/config.yml`)
4. Créer un contrôleur (la classe PrincipalController) qui contiendra la méthode welcomeAction
5. Modifier le template twig pour l'affichage

Partie 2 : Mise en oeuvre

1. Créer le projet TutoSymfonyBR

Normalement vous savez faire avec la commande `php symfony....`

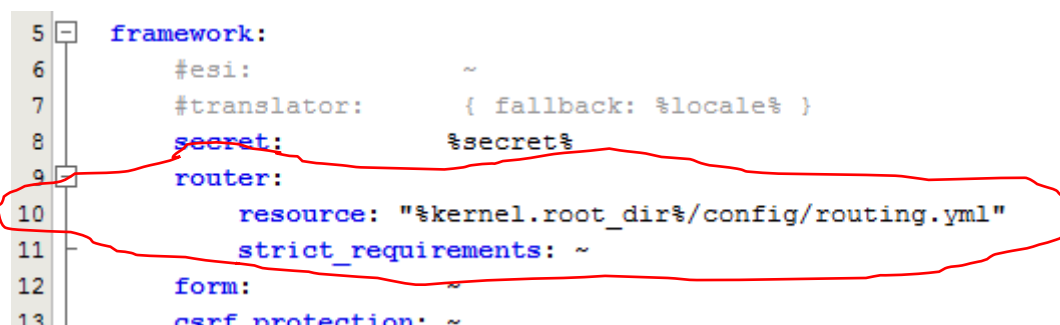
2. Ouvrir ce projet sous netbeans



3. Vérification du routeur de l'application

Dans cette étape, il n'y aura rien à faire.
On va juste voir comment Symfony retrouve ses routes.

On va d'abord aller voir le fichier de configuration de l'application : ...\\app\\config\\config.yml
Format du fichier : yaml



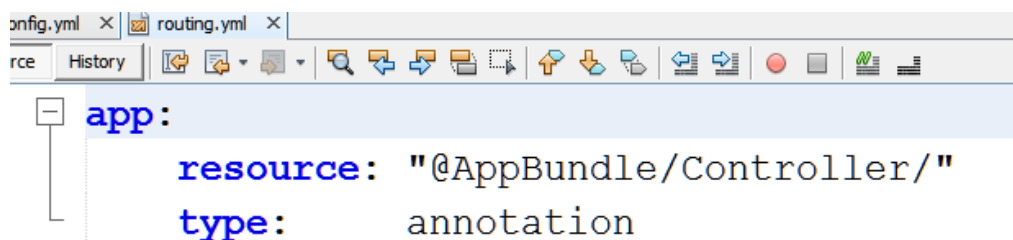
Ce paramètre de configuration indique que le routeur de l'application est le fichier
"%kernel.root_dir%/config/routing.yml"

"%kernel.root_dir%" représente le chemin C:\wamp\www\TutoSymfonyBR\app\

Donc on va ouvrir le fichier

...\TutoSymfonyBR\app\config\routing.yml

Format yaml



```
app:
    resource: '@AppBundle/Controller/'
    type:      annotation
```



L'entrée app rappelle que ce qui suit va concerner le bundle AppBundle. C'est un nom libre.

La suite indique à Symfony qu'il faut interpréter les routes directement dans les annotations des contrôleurs du bundle AppBundle

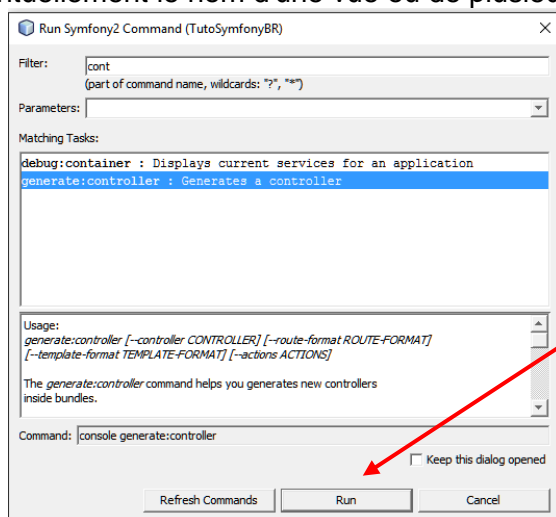
4. Créer un contrôleur (la classe PremierCtlController)

Pour créer notre nouveau contrôleur, nous allons passer ... par une commande symfony2 :

Ce n'est pas la peine de fournir les paramètres, ils nous seront demandés au fur et à mesure de la création. Notamment :

- ✓ Le nom du contrôleur
- ✓ Eventuellement le nom d'une action ou de plusieurs actions

Eventuellement le nom d'une vue ou de plusieurs vues



- Le nom du contrôleur:

Il doit être de la forme NomBundle:nomDucontrôleur (donc AppBundle:Principal)

```
Output - Symfony 3 (MonPremierProjetSymfony) × Notifications Git - config [intranet] - developpeur
"C:\wamp64\bin\php\php7.0.10\php.exe" "T:\Wampsites\CoursSymfony\MonPremierProje

Controller name: 
Welcome to the Symfony controller generator

Every page, and even sections of a page, are rendered by a controller.
This command helps you generate them easily.

First, you need to give the controller name you want to generate.
You must use the shortcut notation like AcmeBlogBundle:Post

AppBundle:Principal|
```

-
- Le format du routing : par défaut il propose annotation, on garde et on valide sans rien saisir

```
AppBundle:Principal
Routing format (php, xml, yml, annotation) [annotation]:
Determine the format to use for the routing.
```

- Le moteur de template voulu : twig par défaut, on valide

```
Template format (twig, php) [twig]:
Determine the format to use for templating.
|
```

- Les actions

on peut lui demander de créer plusieurs action (méthodes de la classe du contrôleur que l'on vient de créer).

Attention, elles sont suffixées par le mot clé *Action*

On va créer l'action welcome (welcomeAction)

```
New action name (press <return> to stop adding actions):
Instead of starting with a blank controller, you can add some actions now. An action
is a PHP function or method that executes, for example, when a given route is matched.
Actions should be suffixed by Action.

welcomeAction|
```

Il nous demande alors la route pour cette action.

Par défaut il propose /welcome .

On va lui mettre /welcome/{nom}

```
welcomeAction
Action route [/welcome]: /welcome/{nom}|
```

Et enfin, il nous propose de créer un squelette de vue, après avoir créé les bons dossiers. On valide !

```
Action route [/welcome]: /welcome/{nom}
Templatenname (optional) [AppBundle:Principal:welcome.html.twig]:
```

A nouveau on peut créer une nouvelle action :

```
New action name (press <return> to stop adding actions): |
```

On validera, on n'en a pas besoin actuellement. Et puis ce sera facile d'en rajouter par la suite à la main dans la classe du contrôleur.

Il ne reste plus qu'à confirmer la création du contrôleur :

```
You are going to generate a "AppBundle:Principal" controller
using the "annotation" format for the routing and the "twig" format
for templating
Do you confirm generation [yes]? |
```

Et voilà c'est fait !!!

```
Do you confirm generation [yes]?
```

```
Controller generation
```

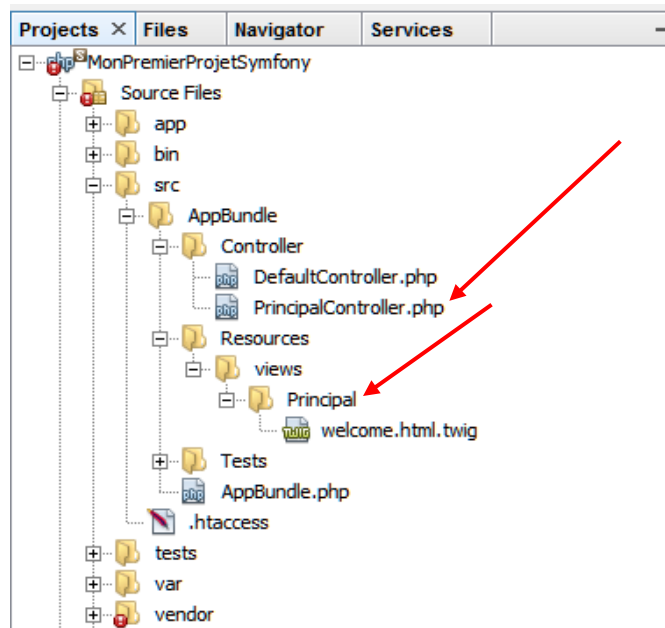
```
created .\src\AppBundle\Resources\views\Principal/
created .\src\AppBundle\Resources\views\Principal\welcome.html.twig
created .\src\AppBundle\Controller\PrincipalController.php
created .\src\AppBundle\Tests\Controller/
created .\src\AppBundle\Tests\Controller\PrincipalControllerTest.php
Generating the bundle code: OK
```

```
Everything is OK! Now get to work :).
```

```
Done.
|
```

➤ Vérification

On va aller voir l'arborescence du projet :



- Le fichier PrincipalController a été créé
- Le dossier views/Principal a été créé
- La vue a bien été créée dans ce dossier !

Allons voir le contenu des fichiers :

Le contrôleur :

```

3  namespace AppBundle\Controller;
4
5  use Symfony\Bundle\FrameworkBundle\Controller\Controller;
6  use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
7
8  class PrincipalController extends Controller
9  {
10
11      /**
12       * @Route("/welcome/{nom}")
13       */
14      public function welcomeAction($nom)
15      {
16          return $this->render('AppBundle:Principal:welcome.html.twig', array(
17              // ...
18          ));
19      }
20  }

```

Red arrows point to the namespace, the class declaration, the route attribute, the action name, and the render method call.

Remarque : en fonction de la configuration, on aurait pu avoir ceci, la vue étant appelée par l'intermédiaire de l'annotation @template

```
<?php

namespace AppBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Template;

class PrincipalController extends Controller {

    /**
     * @Route("/welcome/{nom}")
     * @Template()
     */
    public function welcomeAction($nom) {
        return array(
            // ...
        );
    }
}
```

Le fichier template :

```
{% extends "::base.html.twig" %}

{% block title %}AppBundle:Principal:welcome{% endblock %}

{% block body %}
<h1>Welcome to the Principal:welcome page</h1>
{% endblock %}
```

On remarque que ce template hérite du template principal base.html.twig (dossier app/Resources/views) qui ressemble à ceci :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <title>{% block title %}Welcome!{% endblock %}</title>
        {% block stylesheets %}{% endblock %}
        <link rel="icon" type="image/x-icon" href="{{ asset('favicon.ico') }}" />
    </head>
    <body>
        {% block body %}{% endblock %}
        {% block javascripts %}{% endblock %}
    </body>
</html>
```



On peut tester notre page par l'URL :

http://symfony.br/TutoSymfonyBR/web/app_dev.php/welcome/Benoît

Mais on voit que le paramètre Benoît n'a pas été pris en compte. Il faudra donc modifier la méthode welcomeAction et el formulaire welcome.html.twig

5. Modifier le template twig pour l'affichage

On va modifier la vue pour afficher le nom :

Pour que le nom passé dans l'URL, il faut le préciser dans la vue à l'aide d'une variable twig.

```
{% extends "::base.html.twig" %}

{% block title %}AppBundle:Principal:welcome{% endblock %}

{% block body %}
    <h1>Welcome to {{ nom }}</h1>
{% endblock %}
```

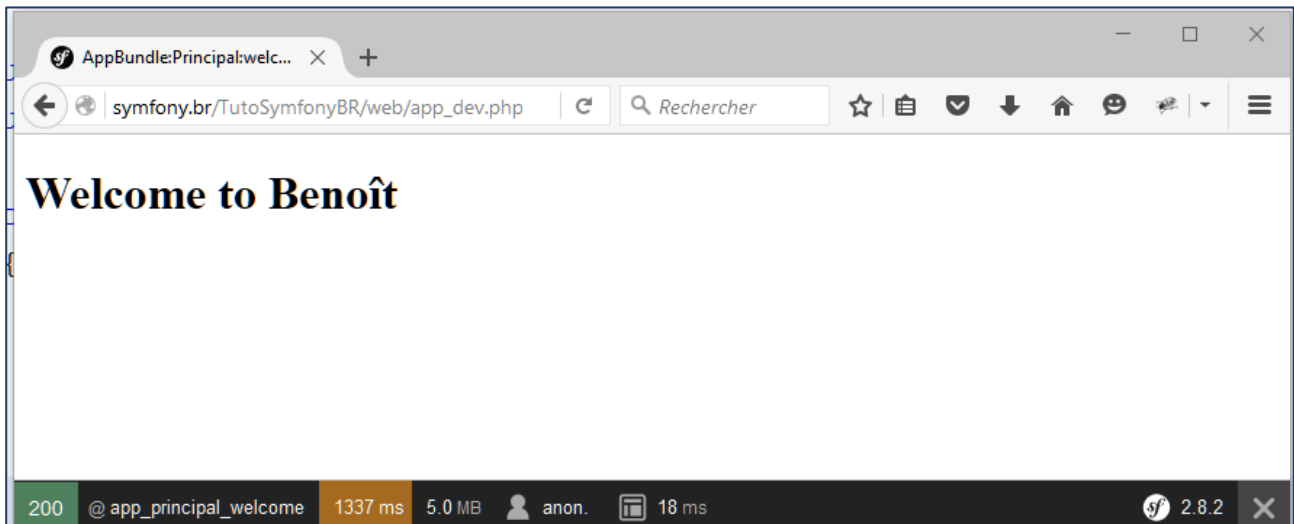
Mais pour que cette variable arrive jusqu'à la vue, il faut la placer dans le tableau passé en paramètre à la méthode render de la méthode welcomeAction.

```
/**
 * @Route("/welcome/{nom}")
 */
public function welcomeAction($nom)
{
    return $this->render('AppBundle:Principal:welcome.html.twig', array(
        "nom"=>$nom
    ));
}
```

On remarque que la variable \$nom passée en paramètre correspond à la partie {nom} de l'URL !!!

http://symfony.br/TutoSymfonyBR/web/app_dev.php/welcome/Benoît

Et voici le résultat.

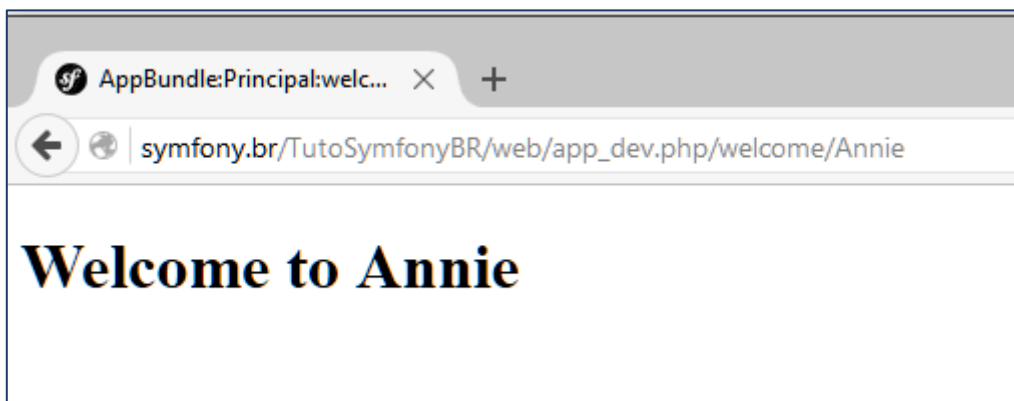


On remarque aussi que l'on est en mode développement (app_dev.php) et donc que la toolbar de débogage symfony est affichée

Essayons avec l'url suivante :

http://symfony.br/TutoSymfonyBR/web/app_dev.php/welcome/Annie

On obtient le résultat suivant :



Partie 3 : Exercices

Exercice 1



Exercice:

Vider les cache prod et dev :

- ✓ En mode console
- ✓ En mode console symfony
- ✓ A la main

Exercice 2



Exercice:

Créer une nouvelle action dans le contrôleur PrincipalController qui réponde à l'URL http://symfony.br/TutoSymfonyBR/web/app_dev.php/message/?? et qui affiche ce que vous voulez ... vous passerez au moins 2 paramètres à la vue !

Exemple :



Vous êtes né dans le département 33

Nous sommes le : Wednesday 30 November 2016 10:25

Le dossier de base de votre application est :

T:\Wampsites\CoursSymfony\MonPremierProjetSymfony

Partie 3 : Nettoyage

Pour chaque bundle créé, on supprime :

- ✓ Le contrôleur Controller/DéfaultController.php
- ✓ Les vues du dossier Resources/views/Default