Programmation avancée & Symfony

BUT Informatique - 3ème année

TP 2

Contrôleur, Route et Template

Premières pages

©Claire BOMBEAUX
Version : **06/09/2025**

Objectifs du TP 2

- **Les bases :**
  - **Créer un premier contrôleur** : Une fonction PHP qui construit la page et retourne un objet Response
  - **Créer une route** : définir l'URL qui pointe vers le contrôleur
  - **Créer un template Twig**

Sommaire

1. Créer votre première page dans Symfony

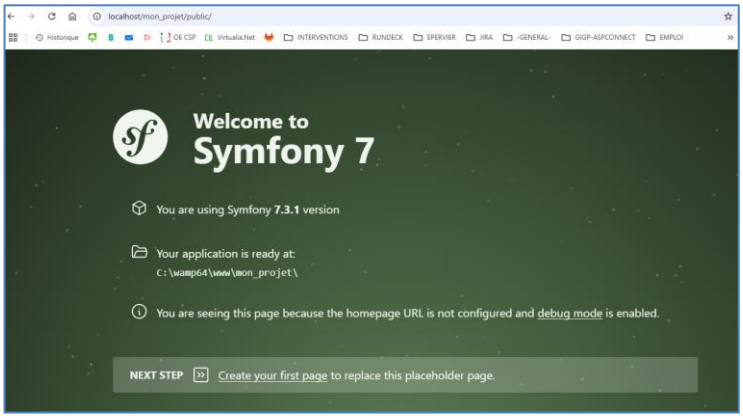
Le but ici est de créer une page `/lucky/number` qui génère un nombre aléatoire.
Cela ne fait pas vraiment parti de notre projet, considèrerez le comme un exercice préparatoire.

Cet exemple est celui proposé par Symfony, il est à maitriser, car il permet de comprendre rapidement les concepts de de contrôleurs, routes et templates, indispensables !

Pour créer une page avec Symfony, il vous faut :

- Une **route** : pour faire le lien entre une URL et une méthode d'un contrôleur
- Un **contrôleur** : qui contient des méthodes, chacune, en générale, associée à une route
- Une **méthode (fonction)** : qui permet l'exécution d'une action précise, généralement en lien avec une route

Prérequis : Fin du TP 1



http://127.0.0.1/mon_projet/~~public~~/

1.1 Le contrôleur

Créez le fichier *LuckyController.php* dans */src/Controller* avec une fonction *number()*

src/Controller/LuckyController.php

```php
<?php
namespace App\Controller;
use Symfony\Component\HttpFoundation\Response;

class LuckyController
{
    public function number(): Response
    {
        $number = random_int(0, 100);

        return new Response(
            '<html><body>Lucky number: '.$number.'</body></html>'
        );
    }
}
```

Ce contrôleur est composé d'une méthode qui calcul un nombre aléatoire et retourne une

réponse. Ce code n'utilise pas directement les vues de Symfony, et ne fonctionne pas (en tout cas il n'est pas possible de l'appeler), car il n'est actuellement pas lié à une route.

La route

Associez un chemin (path) au contrôleur avec l'attribut `#[Route]` :

src/Controller/LuckyController.php

1.2

```php
//route globale au controlleur
#[Route('/lucky')]
class LuckyController
{
    // Route de la fonction
    #[Route('/number', 'lucky_number', methods: ['GET', 'POST'])]

    public function number(): Response
    {
        // Même code que précédemment
    }
```

Et activer (inclure) le support de l'attribut Route, via le use :

src/Controller/LuckyController.php

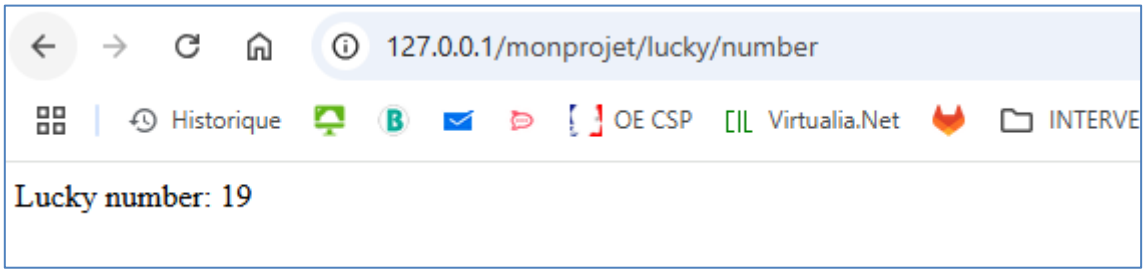
```php
use Symfony\Component\Routing\Attribute\Route;

class LuckyController
{
    // Même code que précédemment
```

Cela est nécessaire, sinon la route ne sera pas reconnue (Error http 404).

(Ancienne méthode : Annotations `use Symfony\Component\Routing\Annotation\Route;`)

La route (-> url) sera donc : [baseUrl]/[routeGlobaleCtl]/[routeFct] :
soit [baseUrl]**/lucky/number** :



Nous avons défini notre route directement dans le contrôleur. ***Quelle est l'autre solution pour définir une route ?***

Rendu de templates avec Twig

Pour créer des pages HTML plus complexes, Symfony utilise Twig.

Le HTML n'est donc plus dans votre contrôleur, il est à déporter dans la vue → MVC !

Pour ce faire, commencer par installer Twig :

1.3

```
composer require twig
```

```
PS C:\wamp64\www\mon_projet> composer require twig
./composer.json has been updated
Running composer update symfony/twig-pack
```

Étendre de **`AbstractController`** dans votre contrôleur (sans oublier l'inclusion de la classe `AbstractController` via le « use ») :

php

```php
<?php
// src/Controller/LuckyController.php

namespace App\Controller;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

class LuckyController extends AbstractController
{
    // même code que précédemment
}
```

Utiliser la méthode **`render()`** pour rendre un template :

php

```php
public function number(): Response
{
    // …
    return $this->render('lucky/number.html.twig', [
        'number' => $number,
    ]);
// …
```

Créer le fichier template correspondant dans le dossier `templates/`

twig

```twig
{# templates/lucky/number.html.twig #}
<h1>Your lucky number is {{ number }}</h1>
```

📖 **Documentation** : https://symfony.com/doc/current/page_creation.html

Toolbar et Profiler

Le Web Profiler capture de nombreuses informations lors de l'exécution d'une requête :

- **Performances générales** - temps d'exécution total, utilisation mémoire, nombre de requêtes base de données
- **Requêtes de base de données** - toutes les requêtes SQL exécutées, leur temps d'exécution, et les paramètres utilisés

1.4

- **Rendu des templates** - templates Twig utilisés, temps de rendu, variables passées
- **Événements et listeners** - événements Symfony déclenchés et leurs listeners
- **Informations sur la requête/réponse** - headers HTTP, paramètres, cookies, session
- **Cache et logs** - opérations de cache, messages de logs par niveau

Utilisez le Profiler pour regarder ce qui se passe lors de l'appel et affichage de la page liée à votre URL [*baseUrl*]**/lucky/number**.

Utilisez notamment l'onglet ***Performance***, qui donne des informations sur toutes les étapes de génération d'une page Symfony.
Comme vous pouvez le voir, l'utilisation du Web Profiler a un impact sur les performances, c'est pour cela qu'il ne doit être utilisé qu'en phase de développement/test.

📖 **Documentation** : Symfony – Profiler : https://symfony.com/doc/current/profiler.html

2. Maker : Création avec la console

Symfony propose le *Maker*, un outil permettant de générer automatiquement du code. Le code produit est générique, basique, et ne correspond pas forcément exactement à aux besoins, mais il permet d'avoir une première structure et base de travail, ce qui peut s'avérer être un gain de temps intéressant.

Installation du Maker (si nécessaire),

Se placer dans le répertoire de votre projet (wamp64\www\mon_projet), puis commande :

```
composer require maker –dev
```

ou

```
symfony composer require maker –dev
```

Utilisation du maker pour générer un Controller :

```
bin/console make:controller NameController
```

Cette commande va générer un controller nommé "`NameController`" (dans `src/controller`) et la vue associée dans le repertoire `templates/NameController`.

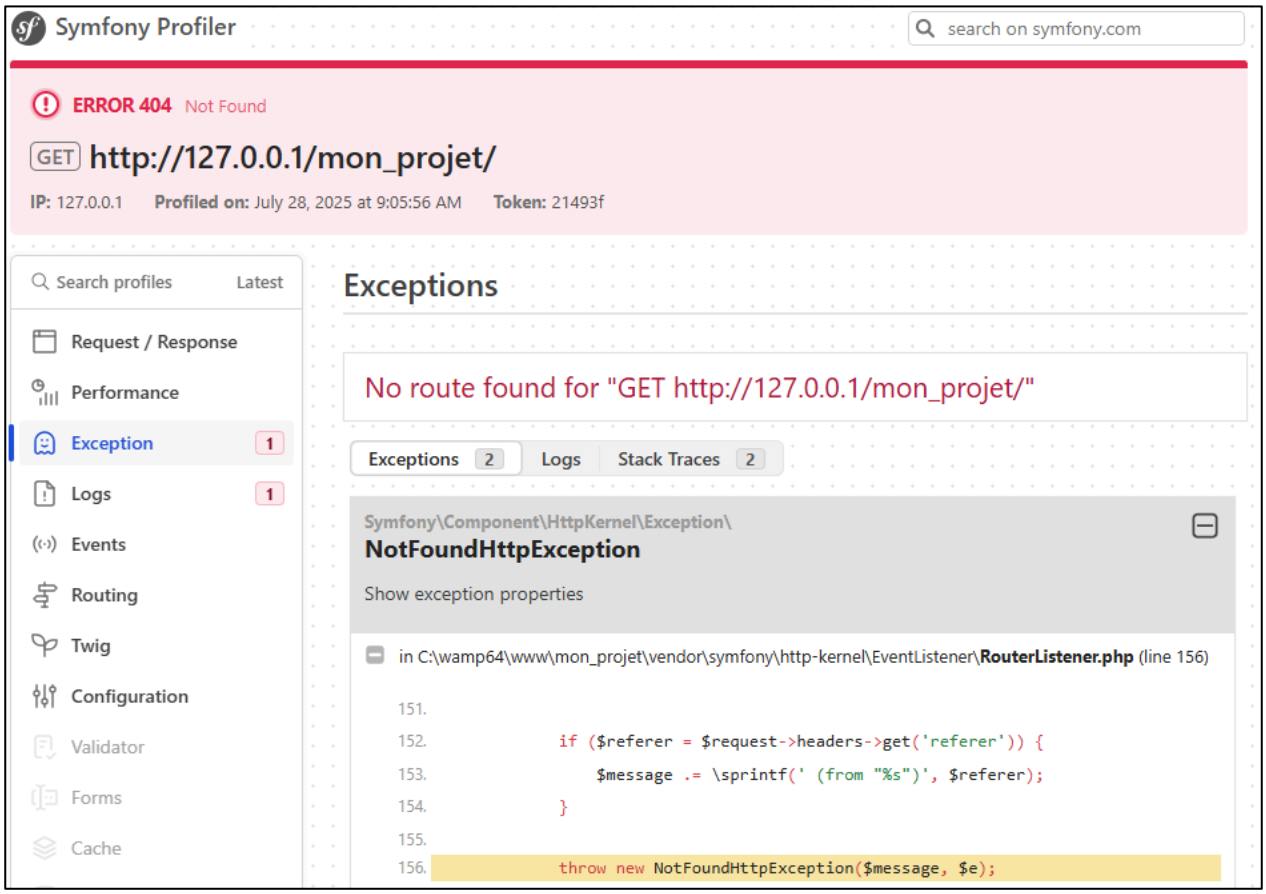
Nous utiliserons Maker dans le prochain TP

(sauf si vous préférez dév. manuellement)

📖 **Documentation** : https://symfony.com/bundles/SymfonyMakerBundle/current/index.html

3. Créer la page d'index

Actuellement votre application n'a pas de route principale, vers la racine, ce qui génère une **erreur HTML 404 Not found** :



http://127.0.0.1/mon_projet/

Créer maintenant votre page principale, c'est-à-dire celle associée à la route **/**.

Le contrôleur sera : **DefaultController.php**

Le template sera : **index.html.twig**
HTML valide W3C

Validator W3C : https://validator.w3.org

→ Pensez dès maintenant que toutes vos pages HTML devront avoir une structure

Cette page doit (pour l'instant) afficher, à minima, des renseignements d'identification : le nom de votre projet, vos noms et prénoms de binôme, votre groupe de TP…

http://127.0.0.1/mon_projet/

Récap.

Structure finale attendue Symfony

```
src/
├── Controller/
│   ├── LuckyController.php
│   └── DefaultController.php
└── templates/
    └── index.html.twig
    └── lucky
        └── number.html.twig
```

MEMO Structure d'un projet Symfony

- `config/` : Configuration (routes, services, packages)
- `src/` : Code PHP
    - `src/Controller` : Controlleurs
    - `src/Entity` : Entities
- `templates/` : Vues = templates Twig
- `bin/` : Fichiers exécutables (comme `bin/console`)
- `var/` : Fichiers générés automatiquement (cache, logs)
- `vendor/` : Symfony et bibliothèques tierces
- `public/` : Fichiers publiquement accessibles (CSS, images …)