ChapIII: Les contrôleurs et les routes

I: Les contrôleurs

1- Defintion

Un contrôleur est une fonction PHP que vous créez et qui lit les informations de l'objet Request et crée et renvoie un objet Response. La réponse peut être une page HTML, JSON, XML, un téléchargement de fichier, une redirection, une erreur 404 ou autre. Le contrôleur exécute toute logique arbitraire dont votre application a besoin pour rendre le contenu d'une page.

2- Création d'un controleur

```
D:\Symfonyprojects\Apptest>php bin/console make:controller

Choose a name for your controller class (e.g. FierceGnomeController):

> OperationsController

created: src/Controller/OperationsController.php
created: templates/operations/index.html.twig

Success
```

Symfony console make:controller

I: Les contrôleurs

1- Structure du classe contrôleur

Un contrôleur est une fonction PHP que vous créez et qui lit les informations de l'objet Request et crée et renvoie un objet Response. La réponse peut être une page HTML, JSON, XML, un téléchargement de fichier, une redirection, une erreur 404 ou autre. Le contrôleur exécute toute logique arbitraire dont votre application a besoin pour rendre le contenu d'une page.

```php
<?php

namespace App\Controller;

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

class OperationsController extends AbstractController
{
    #[Route('/operations', name: 'app_operations')]
    public function Addition(): Response
    {
        // ..................................................
    }
}
```

- namespace: Symfony profite de la fonctionnalité d'espace de noms de PHP pour nommer la classe de contrôleur entière.
- use : importations des packages et des bibliothèques.
- Extend: heritage de la classe système AbstractController
- Addition: une propriété/action dans le contrôleur

I: Les contrôleurs

2- Héritage du contrôleur de base : AbstractController

La classe AbstractController va nous fournir des méthodes auxiliaires, ou des helper methods en anglais, qui vont nous permettre, par exemple, de rendre un template facilement, ou d'effectuer une redirection, ou encore de gérer des erreurs. Nous allons l'implémenter. Pour utiliser cette classe AbstractController, nous devons étendre notre classe à la classe AbstractController. Cela se fait grâce au mot-clé extends AbstractController. Vous notez un apport qui a été fait automatiquement. Donc, grâce à ce mot-clé extends, qu'est-ce qui se passe ? Ma classe va pouvoir accéder à l'ensemble des propriétés et méthodes publiques et protégées de la classe AbstractController. En fait, cela réfère au principe d'héritages en programmation orientée objet, où notre classe fille, va hériter des propriétés et méthodes publiques et…

- Voici quelques propriétés de la classe :
  - ✓ function generateUrl()
  - ✓ function forward()
  - ✓ function redirect()
  - ✓ function redirectToRoute()
  - ✓ function isGranted()
  - ✓ function render()
  - ✓ function getUser()

I: Les contrôleurs

3- Objet : Response

3.1: Retourner une réponse simple

Avec Symfony, un contrôleur est requis pour renvoyer un **objet Response** La réponse peut être une page HTML, JSON, XML, un téléchargement de fichier, une redirection, une erreur 404 ou autre. Voici quelque exemples des réponses.

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

```
class OperationsController extends AbstractController
{
   #[Route('/index', name: 'app_operations')]
   public function index(Request $request): Response
   {
      $response = new Response('<html> <Body><h1>merci</h1></body></html>');

      return $response;
   }
}
```

I: Les contrôleurs

3.2: Retourner un json

Le JavaScript Object Notation (JSON) est un format standard utilisé pour représenter des données structurées de façon semblable aux objets Javascript. Il est habituellement utilisé pour structurer et transmettre des données sur des sites web (par exemple, envoyer des données depuis un serveur vers un client afin de les afficher sur une page web ou vice versa).

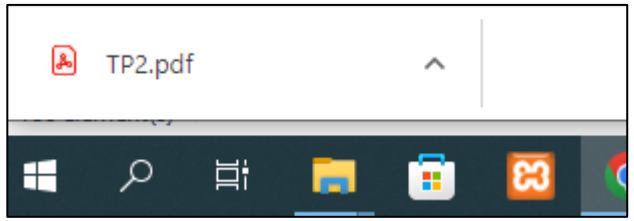
Un objet JSON accepte comme valeur les types: chaînes de caractères, nombres, tableaux, booléens et tout autre objet littéral. Cela vous permet de hiérarchiser vos données

```php
class OperationsController extends AbstractController
{
    #[Route('/index', name: 'app_operations')]
    public function index(Request $request): Response
    {
        $response = new Response(
            '<html> <Body><h1>'.$this->json(['key1' => 'value1','key2' => 'value2']).'</h1></body></html>'
        );
        return $response;
    }
}
```

I: Les contrôleurs

3.3: Retourner un fichier

```php
class OperationsController extends AbstractController
{
    #[Route('/index', name: 'app_operations')]
    public function index(Request $request): Response
    {
        $response = $this->file('files\TP2.pdf');
        return $response;
    }
}
```



Remarque : le chemin absolue de Tp2.pdf est : ***D:\Symfonyprojects\Apptest\public\files\TP2.pdf***
par default symfony se pointe sur le dossier « public »

I: Les contrôleurs

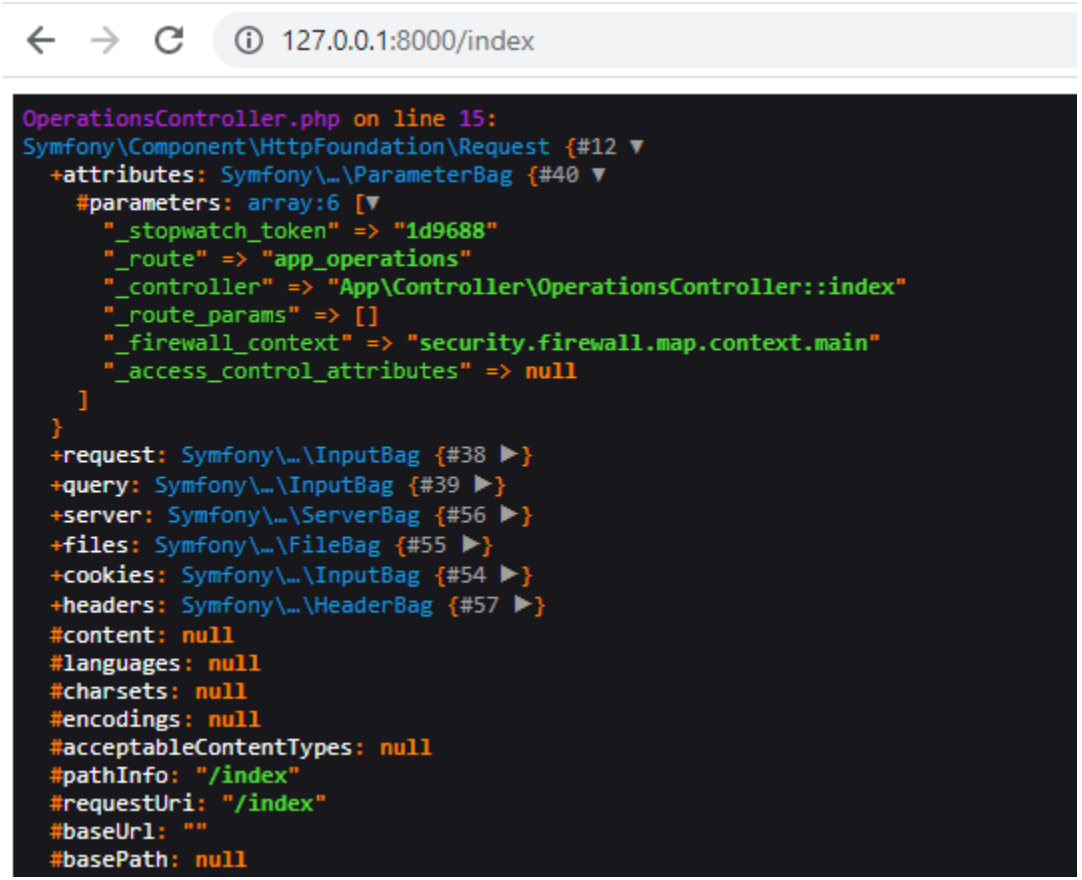
4- Objet : Request

3.1: Retourner une requête http

Symfony fournit une approche à travers deux classes pour interagir avec la requête et la réponse HTTP. La classe Request est une représentation de la requête HTTP, tandis que la classe Response est évidemment une représentation de la réponse HTTP.

```php
use Symfony\Component\HttpFoundation\Request;
```

```php
class OperationsController extends AbstractController
{
    #[Route('/index', name: 'app_operations')]
    public function index(Request $request): Response
    {
        dd($request);
        $response = $this->file('files\TP2.pdf');
        return $response;
    }
}
```



I: Les contrôleurs

4- Redirection

4.1: Rediriger vers une route sans paramètres

```php
#[Route('/index/test', name: 'IndexLink')]
public function index(): response
{
    //MSLink est le nom d'une route
    return $this->redirectToRoute('MSLink');
}
```

4.2: Rediriger vers une route avec paramètres

```php
#[Route('/index/test', name: 'IndexLink')]
public function index(): response
{
    return $this->redirectToRoute('LuckynumberLink', ['max' => 10]);
}
```

4.3: Rediriger vers un controleur

```php
#[Route('/index/test', name: 'IndexLink')]
public function index(): response
{
     return $this->forward('App\Controller\OperationsController::Add', ['a' => 56, 'b' => 6,]);
}
```

4.3: Rediriger vers un lien externe

```php
#[Route('/index/test', name: 'IndexLink')]
public function index(): response
{
    return $this->redirect('http://www.facebook.com');
}
```

II: Les Routes

1- Structure d'une route

```php
<?php

namespace App\Controller;

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

class BlogController extends AbstractController
{
   #[Route('/blog', name: 'BlogLink')]
   public function  download(): Response
   {
      // ……..
   }
}
```

Cette configuration définit une route appelée BlogLink qui correspond lorsque l'utilisateur demande l'URL /blog. Lorsque la correspondance se produit, l'application exécute la méthode download() de la classe BlogController.

II: Les Routes

2- Création de routes dans le fichiers YAML

config/routes.yaml

```yaml
controllers:
   resource: ../src/Controller/
   type: attribute
IndexLink:
      path: /index/test
      controller: App\Controller\TabController::index
```

```php
//#[Route('/index/test', name: 'IndexLink')]
public function index(): response
{

   return $this->forward('App\Controller\OperationsController::Add', ['a' => 56, 'b' => 6,]);

}
```

Au lieu de définir des routes dans les classes de contrôleur, vous pouvez les définir dans un fichier YAML.
Le principal avantage est qu'ils ne nécessitent aucune dépendance supplémentaire. Le principal inconvénient est que vous devez travailler avec plusieurs fichiers lors de la vérification du routage de certaines actions du contrôleur.

II: Les Routes

3- Débogage des routes

```
D:\Symfonyprojects\MyFirstApp>php bin/console debug:router
```

```
 ------------------------ -------- -------- ------ ------
  Name                     Method   Scheme   Host   Path
 ------------------------ -------- -------- ------ ------
  AddLink                  ANY      ANY      ANY    /operations/add/{a}/{b}
  SousLink                 ANY      ANY      ANY    /operations/sous/{a}/{b}
  MultiLink                ANY      ANY      ANY    /operations/multi/{a}/{b}
  MSLink                   ANY      ANY      ANY    /tab/msp
  LuckynumberLink          ANY      ANY      ANY    /myluckynumber/{max}
  IndexLink                ANY      ANY      ANY    /index/test
```

4- Déboguer une route

```
D:\Symfonyprojects\MyFirstApp>php bin/console debug:router LuckynumberLink
```

5- Chercher une route à partir d'une URL

```
D:\Symfonyprojects\MyFirstApp>php bin/console route:match /myluckynumber/10

 [OK] Route "LuckynumberLink" matches
```