



TEK-UP Ecole Supérieure Privée Technologie &
Ingénierie

Développement Web (PHP **Symfony 5**)

Wissem ELJAOUED
wissem.eljaoued@ensi-uma.tn

A.U. 2021-2022

Plan du cours

I. Introduction Symfony

TP1: Création d'un projet

II. Routing (Routeur de Symfony)

TP2: Router + Controller

III. Les contrôleurs

IV. Les Twig

TP3: Twig

V. ORM Doctrine

TP4: Doctrine

TP5: DQL

VI. Les Formulaire

TP6: Forms

Ch. III

Les Contrôleurs

Ch. III: Les Contrôleurs

III.1. Introduction

III.2. Route vers Contrôleur

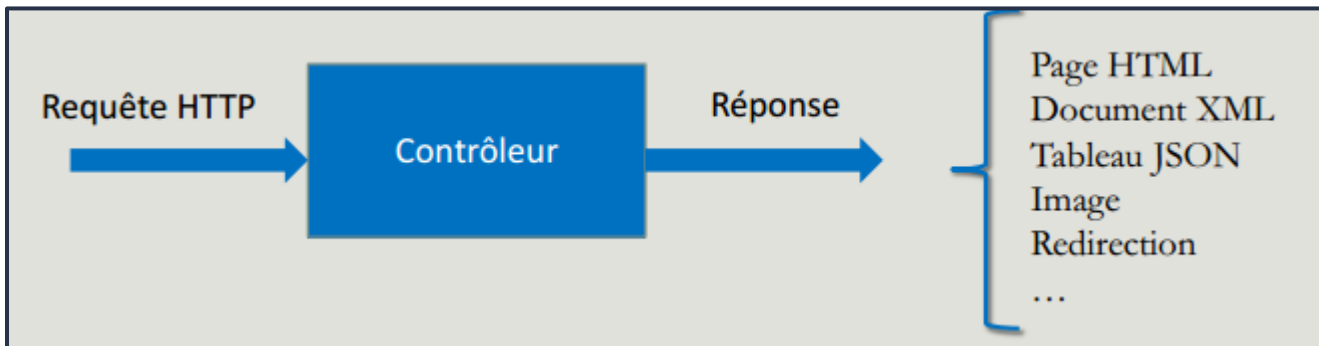
III.3. Manipulation de l'objet Request

III.4. Manipulation de l'objet Response

III.5. Manipulation d'une Session

- Fonction PHP (Action)
- **Rôle** : Répondre aux requêtes des clients
➔ Retourner Une **Réponse**

Il existe dans Symfony les classes **Request** et **Response**. Retourner une réponse signifie donc tout simplement : instancier un objet **Response**, disons `$response`, et faire un `return $response`.



Exemple 1:

Ce contrôleur dispose d'une seule méthode, nommée « index », et retourne une réponse qui ne contient que « Hello World ! »

```
1 <?php
2
3 // src/Controller/AdvertController.php
4
5 namespace App\Controller;
6
7 use Symfony\Component\HttpFoundation\Response;
8
9 class AdvertController
10 {
11     public function indexAction()
12     {
13         return new Response("Hello World !");
14     }
15 }
```

Exemple 2:

```
<?php
namespace Forma\TestBundle\Controller; //espace de nom
use Symfony\Bundle\FrameworkBundle\Controller\Controller; // on importe la classe Controller
use Symfony\Component\HttpFoundation\Response;

class HelloController extends Controller // Si on veut que notre contrôleur hérite de la classe afin de pouvoir utiliser
ces méthodes helper (http://api.symfony.com/2.8/Symfony/Bundle/FrameworkBundle/Controller/Controller.html)
{ // Par convention chaque contrôleur (action) se termine par le mot clé Action
    public function indexAction() // le nom index est l'identifiant de l'action c'est avec ce nom que nous
    // l'appelons dans la root
    { // On crée un objet Response puis on la retourne c'est le rôle du contrôleur
        $resp = new Response('<html><body>Bonjour le monde !</body></html>');
        return $resp;
    }
}
```

❖ Fonctions de base de la classe Controller

Méthode	fonctionnalité	Valeur de retour
generateUrl(string \$route, array() \$parameters)	Génère une URL à partir de la route	String
forward(String Controller, array () \$parameters)	Forward la requête vers un autre contrôleur	Response
Redirect(string \$url int \$statut)	Redirige vers une url	RedirectResponse
Render(string \$view array \$parameters)	Affiche une vue	Response
getRequest()	Raccourci pour récupérer le service request	Request
Get(string \$id)	Retourne un service à travers son id	object
createNotFoundException(String \$messag)	Retourne une NotFoundException	NotFoundException

Ch. III: Les Contrôleurs

III.1. Introduction

III.2. Route vers Contrôleur

III.3. Manipulation de l'objet Request

III.4. Manipulation de l'objet Response

III.5. Manipulation d'une Session

- Une route est composée au minimum de deux éléments:
 - Une entrée (ligne **path**)
 - Une sortie (ligne **controller**)




controller: **App\Controller\NameController::action**

- Afin de récupérer les paramètres de la Route dans le contrôleur nous utilisons les noms des paramètres:

Exemple:

```
/**
 * @Route("/etudiant/{id}", name="affichage_etudiant")
 */
public function affichageEtudiant($id): Response
{
    return new Response( content: "Bonjour l'etudiant numéro ".$id);
}
```



Ch. III: Les Contrôleurs

III.1. Introduction

III.2. Route vers Contrôleur

III.3. Manipulation de l'objet Request

III.4. Manipulation de l'objet Response

III.5. Manipulation d'une Session

❖ Les paramètres contenus dans les routes

```
1 <?php
2 // src/Controller/AdvertController.php
3
4 /**
5  * @Route("/view/{id}", name="oc_advert_view", requirements={"id" = "\d+"})
6  */
7 public function view($id)
8 {
9     return new Response("Affichage de l'annonce d'id : ".$id);
10 }
```

Ici, le paramètre **{id}** de la requête est récupéré par la route, qui le transforme en argument **\$id** pour le contrôleur. La première manière de récupérer des arguments : ceux contenus dans la route.

❖ Les paramètres hors routes

On peut récupérer les autres paramètres de l'URL. Prenons par exemple l'URL `/advert/view/5?tag=developer`, il nous faut bien un moyen pour récupérer ce paramètre `tag` : c'est ici qu'intervient l'objet `Request`.

Pour récupérer la requête depuis un contrôleur, il faut l'injecter comme nous avons injecté des objets. Pour cela, ajoutez un argument à votre méthode avec le *typehint* `Request` comme ceci :

```
1 <?php
2 public function view($id, Request $request)
3 {
4     // On récupère notre paramètre tag
5     $tag = $request->query->get('tag');
6
7     return new Response(
8         "Affichage de l'annonce d'id : ".$id.", avec le tag : ".$tag
9     );
10 }
```

❖ Les paramètres hors routes

Type de paramètres	Méthode Symfony	Méthode traditionnelle	Exemple
Variables d'URL	<code>\$request->query</code>	<code>\$_GET</code>	<code>\$request->query->get('tag')</code>
Variables de formulaire	<code>\$request->request</code>	<code>\$_POST</code>	<code>\$request->request->get('tag')</code>
Variables de cookie	<code>\$request->cookies</code>	<code>\$_COOKIE</code>	<code>\$request->cookies->get('tag')</code>
Variables de serveur	<code>\$request->server</code>	<code>\$_SERVER</code>	<code>\$request->server->get('REQUEST_URI')</code>

❖ Autres méthodes de **Request**

- Récupérer la **méthode** de la requête HTTP:

Pour savoir si la page a été récupérée via GET (clic sur un lien) ou via POST (envoi d'un formulaire), il existe la méthode `$request->isMethod()`

```
1 <?php
2 if ($request->isMethod('POST'))
3 {
4     // Un formulaire a été envoyé, on peut le traiter ici
5 }
```

- **Toutes les autres:**

Consulter l'API de Request sur le site de Symfony

Ch. III: Les Contrôleurs

III.1. Introduction

III.2. Route vers Contrôleur

III.3. Manipulation de l'objet Request

III.4. Manipulation de l'objet Response

III.5. Manipulation d'une Session

❖ Décomposition de la construction d'un objet Response

Comment construire et retourner un objet Response ??

Exemple: cas d'une page d'erreur 404 (page introuvable)

```
1 <?php
2 // src/Controller/AdvertController.php
3
4 public function view($id, Request $request)
5 {
6     // On crée la réponse sans lui donner de contenu pour le moment
7     $response = new Response();
8
9     // On définit le contenu
10    $response->setContent("Ceci est une page d'erreur 404");
11
12    // On définit le code HTTP à « Not Found » (erreur 404)
13    $response->setStatusCode(Response::HTTP_NOT_FOUND);
14
15    // On retourne la réponse
16    return $response;
17 }
```

❖ **Réponses et vues**

- On préfère que notre réponse soit contenue dans une vue tel que le préconise l'architecture MVC !!!
- **La solution:** La méthode **Render()**: Elle prend en paramètres le **nom du template** et ses **variables** (créer la réponse, lui passer le contenu du template, et la retourner)

```
1 <?php
2 // src/Controller/AdvertController.php
3
4 public function view($id, Request $request)
5 {
6     $tag = $request->query->get('tag');
7
8     // On utilise le raccourci : il crée un objet Response
9     // Et lui donne comme contenu le contenu du template
10    return $this->render(
11        'Advert/view.html.twig',
12        ['id' => $id, 'tag' => $tag]
13    );
14 }
```

Réponses et Redirection (1/2)

- Comment gérer une redirection vers une autre page ?

- **La solution:**

1. L'objet **RedirectResponse()**: Cet objet prend en argument de son constructeur l'URL vers laquelle rediriger, URL que vous générez grâce au routeur bien entendu

```
1 <?php
2 // src/Controller/AdvertController.php
3
4 namespace App\Controller;
5
6 use Symfony\Bundle\FrameworkBundle\Controller\Controller;
7 use Symfony\Component\HttpFoundation\RedirectResponse; // Nouveau use
8
9 class AdvertController extends Controller
10 {
11     public function view($id)
12     {
13         $url = $this->generateUrl('oc_advert_index');
14
15         return new RedirectResponse($url);
16     }
17 }
```

Réponses et Redirection (2/2)

2. La méthode **Redirect()** prend en argument URL

```
1 <?php
2
3 public function view($id)
4 {
5     $url = $this->generateUrl('oc_advert_index');
6
7     return $this->redirect($url);
8 }
```

3. La méthode **RedirectToRoute()** prend en argument le nom de la route

```
1 <?php
2
3 public function viewAction($id)
4 {
5     return $this->redirectToRoute('oc_advert_index');
6 }
```

Changer le Content-Type de la réponse

Lorsque vous retournez autre chose que du HTML, il faut que vous changiez l'en-tête Content-Type de la réponse. Ce dernier permet au navigateur qui recevra votre réponse de savoir à quoi s'attendre dans le contenu. Prenons l'exemple suivant : vous recevez une requête AJAX et souhaitez retourner un tableau en JSON :

```
9  class AdvertController extends AbstractController
10 {
11     public function view($id)
12     {
13         // Créons nous-mêmes la réponse en JSON, grâce à la fonction json_encode()
14         $response = new Response(json_encode(['id' => $id]));
15
16         // Ici, nous définissons le Content-Type pour dire au navigateur
17         // que l'on renvoie du JSON et non du HTML
18         $response->headers->set('Content-Type', 'application/json');
19
20         return $response;
21     }
22 }
```

Ch. III: Les Contrôleurs

III.1. Introduction

III.2. Route vers Contrôleur

III.3. Manipulation de l'objet Request

III.4. Manipulation de l'objet Response

III.5. Manipulation d'une Session

III.5. Manipulation d'une Session

- Une des fonctionnalités de base d'un contrôleur est la manipulation des sessions.
- Un objet `session` est fourni avec l'objet Request.
- La méthode `getSession()` permet de récupérer la session.
- L'objet `Session` fournit deux méthodes : `get()` pour récupérer une variable de session et `set()` pour la modifier ou l'ajouter.
- `get` prend en paramètre la variable de session.
- `set` prend en entrée deux paramètres la clef et la valeur.
- Dans la TWIG on récupère les paramètres de la session avec la méthode `app.session.get('nomParamètre')`


```
1 <?php
2 // src/Controller/AdvertController.php
3
4 namespace App\Controller;
5
6 use Symfony\Bundle\FrameworkBundle\Controller\Controller;
7 use Symfony\Component\HttpFoundation\Response;
8 use Symfony\Component\HttpFoundation\Session\SessionInterface; // Nouveau use
9
10 class AdvertController extends AbstractController
11 {
12     // On injecte la session avec SessionInterface
13     public function viewAction($id, SessionInterface $session)
14     {
15         // On récupère le contenu de la variable userId
16         $userId = $session->get('userId');
17
18         // On définit une nouvelle valeur pour cette variable userId
19         $session->set('userId', 91);
20
21         // On n'oublie pas de renvoyer une réponse
22         return new Response("<body>Je suis une page de test, je n'ai rien à dire</body>");
23     }
24 }
```

- **all()**
Retourne tous les attributs de la session dans un tableau de la forme clef valeur
- **has()**
Permet de vérifier si un attribut existe dans la session. Retourne Vrai s'il existe Faux sinon
- **replace()**
Définit plusieurs attributs à la fois: prend un tableau et définit chaque paire clé => valeur
- **remove()**
Efface un attribut d'une clé donnée.
- **clear()**
Efface tous les attributs.

- Les variables de sessions qui ne durent que le temps d'une seule page sont appelées **message Flash**.
- Utilisées généralement pour afficher un message après un traitement particulier (Ajout d'un enregistrement, connexion, ...).
- La méthode **getFlashBag()** permet de récupérer l'objet **FlashBag** (contient les messages flash dans la session) à partir de l'objet session.
- La méthode **add** de cet objet permet d'ajouter une variable à cet objet.
- Pour récupérer le Flash message de la TWIG on utilise **app.session.flashbag.get('nomParamètre')**.