Support de cours Symfony 4

Jérôme AMBROISE

Support de cours Symfony 4 - Partie 2

Sommaire

Sommaire

Sommaire

Sommaire

5.

Le contrôleur

Quel est le rôle du contrôleur ?

Que contient la classe AbstractController ?

Comment accéder aux élément de la requête ?

Comment créer un réponse ?

Comment accéder à la session ?

“

Quel est le rôle d’un contrôleur ?

Le contrôleur

Rôle du contrôleur

Un contrôleur a pour but de transformer une requête en réponse !

- Symfony propose une classe qui permet d'avoir des méthodes génériques liées aux contrôleurs.

Pour pouvoir l'utiliser, il faut étendre cette classe.

Le contrôleur

La classe AbstractController

Namespace complet de la classe mère de contrôleur :

```
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
```

Explorons cette classe pour voir ce qu'elle contient.

Le contrôleur

La classe AbstractController

- **$container** : permet de récupérer le conteneur d'injection de dépendances
- **getParameter()** : permet de récupérer un paramètre (défini par Symfony ou nous-mêmes)

Nous voyons que AbstractController utilise un trait, explorons ce trait.

Symfony\Bundle\FrameworkBundle\Controller\ControllerTrait

Le contrôleur

La classe AbstractController

Certaines méthodes sont liées au conteneur d'injection de dépendances.

- **has()** : vérifie si la clef est définie dans le conteneur
- **get()** : récupère la définition de la clef dans le conteneur

Le contrôleur

La classe AbstractController

D'autres méthodes permettent de rendre des vues :

- **renderView()** : rend le contenu HTML d'une vue
- **render()** : rend un objet de type Response contenant le HTML d'une vue

Le contrôleur

La classe AbstractController

Le contrôleur peut aussi rendre du JSON ou des fichiers :

- **json()** : retourne une réponse JSON
- **file()** : retourne un fichier

Le contrôleur

La classe AbstractController

Le contrôleur peut aussi retourner une redirection.

- **redirect()** : retourne une réponse de redirection vers une URL
- **redirectToRoute()** : retourne une réponse de redirection à partir du nom d'une route

Le contrôleur

La classe AbstractController

Il y a aussi d'autres méthodes utilitaires que nous verrons en temps voulus, qui traitent de :

- La session PHP
- Les utilisateurs, droits d'accès
- Les formulaires
- Doctrine
- …

Le contrôleur

La classe AbstractController

Maintenant que nous savons les possibilités de la classe abstraite de contrôleur de Symfony, intéressons-nous à la requête.

- La Requête est définie dans une classe Request du composant HttpFoundation.

Namespace complet :

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

Le contrôleur
La requête

L’objet provenant de Request contient les variables globales en tant que propriétés.

$_GET	$query
$_POST	$request
$_SERVER	$server
$_FILES	$files
$_COOKIE	$cookies

Le contrôleur
La requête

La classe Request définit aussi des méthodes utilitaires pour la récupération de l'URL.

Intéressons-nous désormais à la création de la réponse.

Le contrôleur
La réponse

La réponse est représentée par une classe Response du composant HttpFoundation.

Namespace complet :

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

Le contrôleur
La réponse

Le réponse peut être instanciée afin d'être retournée.

Son constructeur prend 3 paramètres facultatifs :

- Le contenu de son corps
- Le statut code
- Les headers

Le contrôleur
La réponse

Les propriétés de la réponse peuvent être changés avec des mutateurs :

- setContent()
- setStatusCode()
- Les headers sont généralement changés pour gérer le cache.

Le contrôleur
La réponse

Le header “Content-type” définit le type de données renvoyé par le serveur.

- Par défaut, le “content-type” est du HTML (text/html)

Afin de retourner d’autres types de données, Symfony prévoit des classes enfant de la classe Response.

Le contrôleur
La réponse

Tableau associant le type de données de réponse avec la classe dérivée de Response.

Type de retour	Classe
HTML	Response
JSON	JsonResponse
Fichier	BinaryFileResponse
Redirection	RedirectResponse

Le contrôleur
La réponse

Les classes dérivées de Response peuvent être utilisées de manière transparente grâce à l'utilisation des méthode de la classe AbstractController.

Le contrôleur
La réponse

Tableau associant le type de données de réponse avec les méthodes de AbstractController.

Type de retour	Classe	Méthode
HTML	Response	render()
JSON	JsonResponse	json()
Fichier	BinaryFileResponse	file()
Redirection	RedirectResponse	redirect(), redirectToRoute()

Le contrôleur
La réponse

Nous savons désormais comment retourner une réponse.

Nous allons désormais voir comment récupérer et utiliser la session.

“

Qu’est-ce que la session ? A quoi cela sert-il ?

Le contrôleur
La session

- La session permet de sauvegarder des données liées à chaque utilisateur individuellement
- La session est sauvegardée de page en page durant la session de navigation de l'utilisateur

Le contrôleur
La session

La session ($_SESSION) est représentée par une classe Session définie par le composant HttpFoundation.

Namespace complet :

Symfony\Component\HttpFoundation\Session

Le contrôleur
La session

- La session peut être récupérée dans une méthode de contrôleur grâce à l'autowiring.

On peut typer la Session ou son interface :

- Symfony\Component\HttpFoundation\Session
- Symfony\Component\HttpFoundation\SessionInterface
- La session peut aussi être récupérée grâce à la méthode "getSession()" de la classe Request

Le contrôleur
La session

La commande suivante permet de lister l'ensemble des interfaces que l'on peut demander dans une méthode de contrôleur grâce à l'autowiring :

```
php bin/console debug:autowiring
```

Le contrôleur
La session

La session est démarrée automatiquement.

Plusieurs méthodes permettent de la manipuler :

- **all()** : récupère tous les éléments de la session
- **has()** : vérifie si une valeur est définie pour la clef donnée

Le contrôleur
La session

Nous pouvons ajouter et récupérer des éléments spécifiques de la session grâce à un accesseur et un mutateur :

- **get()** : récupère une valeur pour la clef données
- **set()** : définit la valeur donnée à la clef donnée

Le contrôleur
La session

Voyons désormais un élément particulier de la session : les messages flashs !

“

Qu’est-ce qu’un message flash ?

Le contrôleur
Les messages flash

Les messages "flash" sont intéressants pour le stockage des notifications des utilisateurs.

- Les messages flash sont censés être utilisés exactement une fois: ils disparaissent de la session automatiquement dès qu'ils sont récupérés.

Le contrôleur
Les messages flash

Les messages flash sont composés :

- **D'un type** : libre (success, notice, warning, danger, …)
- **Une valeur** : le message à afficher

Le contrôleur
Les messages flash

Les messages flash sont généralement créés dans le contrôleur.

Le plus simple est d'utiliser la méthode "addFlash()" de l'AbstractController

Cela raccourcit l'appel de :

- $this->getSession()->getFlashBag()->add()

6.

Les vues avec TWIG

Qu’est ce qu’un moteur de templating?

Qu’est-ce que TWIG?

Comment définir des pages d’erreurs personnalisées ?

Les vues avec TWIG
Présentation

Les vues correspondant à la couche présentation du MVC.

Les vues combinent le PHP et l'HTML pour finalement retourner du HTML.

Les moteurs de templates nous permettent :

- D'alléger la syntaxe PHP-HTML
- Mettre en place des layouts
- …

Les vues avec TWIG

Présentation

- Symfony préconise l'utilisation du moteur de templating TWIG
- Symfony prévoit un bundle "TwigBundle" afin d'intégrer TWIG dans Symfony

Documentation officielle de TWIG.

Pour installer TWIG :

composer require twig

Les vues avec TWIG
Présentation

TWIG est un moteur de template PHP.

- **Rapide** : la surcouche est limitée au maximum
- **Sécurisé** : échappement automatique
- **Extensible** : possibilité d'ajouter de nouvelles fonctions, filtres, …

La syntaxe de TWIG ne veut **simple**.

Il nous permettra aussi de mettre en place un système de **layout** facilement.

Les vues avec TWIG
Présentation

Syntaxe de TWIG

Syntaxe	Explication
{{ … }}	Affichage d'une variable ou le résultat d'une expression dans le modèle.
{% … %}	Contrôle la logique du modèle; Il est utilisé pour exécuter des instructions telles que des boucles for par exemple.
{# … #}	Permet d'inclure des commentaires (équivalent PHP : /* … */)

Les vues avec TWIG
Présentation

TWIG traduit les structures de base de PHP :

- if ... elseif ... else ... endif
- for ... in ... endfor

Nous pouvons définir des variables

- set ... = ...

Les vues avec TWIG
Présentation

TWIG nous fournit des filtres utilisables sur des variables grâce à la pipe (|)

- upper
- length
- keys
- date
- round
- sort
- json_encode
- ...

Les vues avec TWIG
Présentation

TWIG nous fournit aussi des fonctions :

- dump()
- include()
- random()
- ...

Les vues avec TWIG
Présentation

Enfin TWIG fournit un système d'héritage de "block".

- On définit des "blocks" dans une vue parent
- On peut hériter d'une vue parent en utilisant "extends"
  - On bénéficie alors des blocks existants
  - On peut surcharger les blocks existants

Les vues avec TWIG
Présentation

Lors de l'intégration de TWIG dans Symfony, de nouvelles fonctions et filtres sont disponibles :

Symfony Twig Extensions

- On y retrouve la fonction "path()" permettant de générer une URL

Les vues avec TWIG
Présentation

Si les extensions ne sont pas suffisantes, il existe des extensions TWIG :

- Twig Extensions

Et enfin, si vous voulez étendre TWIG avec vos propres extensions, cela est possible facilement :

- Définir une extension TWIG

Les vues avec TWIG

Présentation

Une variable globale “app” est aussi disponible, elle permet d’accéder à :

Variable	Description	Type d’objet
app.user	Si retourne l’éventuel utilisateur connecté	UserInterface
app.request	Retourne la requête	Request
app.session	Retourne la session	Session
app.environment	Retourne l’environnement (dev, prod, …)	string
app.debug	Retourne true ou false	boolean
app.flashes	Retourne les messages flash	array

7.

Base de données avec Doctrine

Qu’est-ce qu’un ORM?

Qu’est-ce Doctrine?

Qu’est-ce qu’une entité ?

Qu’est-ce qu’une migration?

Comment créer une base de données avec Doctrine?

“

Qu’est-ce qu’un ORM ?

Qu’est-ce que Doctrine ?

Base de données avec Doctrine
Présentation

L'usage d'une base de données est très fréquent dans un site web.

- Construction de la structure
- Insertion de données
- Lecture de données
- Mise à jour de données
- Suppression de données

L'objectif d'un ORM est de faciliter ses différentes interactions avec la base de données.

Base de données avec Doctrine
Présentation

Un ORM, **O**bject-**R**elational **M**apping, est une couche d'abstraction pour les interactions avec la base de données.

- Symfony ne possède pas d'ORM mais permet l'intégration de Doctrine

Documentation officielle de Doctrine ORM

Base de données avec Doctrine
Installation

Symfony propose un “recipe” pour installer doctrine.

Installation de doctrine :

composer require doctrine

Base de données avec Doctrine
Paramétrage

Le recipe a ajouté une variable dans le fichier “/.env”.

- DATABASE_URL

Nous devons configurer les paramètres par rapport à notre système :

- db_user
- db_password
- db_name

Base de données avec Doctrine
Commande de création de BDD

Comment vérifier que cela fonctionne ?

- Doctrine propose une commande en console afin de créer la base de données pour nous

Commande de création de base de données :

```
php bin/console doctrine:database:create
```

Base de données avec Doctrine
Commande de suppression de BDD

De la même manière, Doctrine peut supprimer la base de données pour nous.

Commande de suppression de base de données :

```
php bin/console doctrine:database:drop
```

“

Qu’est-ce qu’une entité ?

Base de données avec Doctrine
Présentation des entités

Une entité est une classe PHP qui représente une table de la base de données.

- On crée une classe PHP dont les propriétés seront les futures colonnes de la table
- On “mappe” les propriétés grâce à Doctrine
- Doctrine peut ensuite “transformer” l’entité en une table dans la base de données

On ne créé donc pas la table dans PHPMyAdmin, Doctrine va le faire pour nous, nous allons voir comment.

Base de données avec Doctrine
Présentation des entités

La philosophie d'un ORM, et de Doctrine, est de simuler une "base de données objet".

- On ne raisonne plus en "table" mais en "objet"

Base de données avec Doctrine

Création d'une entité avec MakerBundle

La création d'une entité peut être longue : nous allons nous servir d'un bundle pour automatiser la création de l'entité "MakerBundle".

Installation de MakerBundle :

```
composer require maker
```

Base de données avec Doctrine

Création d'une entité avec MakerBundle

MakerBundle fonctionne en console, il propose diverses commandes pour générer du code.

La liste des commandes est accessible avec la commande suivante :

```
php bin/console list make
```

Base de données avec Doctrine

Création d'une entité avec MakerBundle

Création d'une entité avec MakerBunde :

php bin/console make:entity

- Maker va alors nous demander le nom de l'entité que l'on veut créer
- Maker va ensuite nous demander pour chaque propriété
  - Le nom de la propriété
  - Le type de propriété créée
  - D'éventuelles précisions

Base de données avec Doctrine

Création d'une entité avec MakerBundle

Une fois la commande exécutée avec succès, deux choses se sont produites :

- Maker a créé une classe PHP dans le dossier "/src/Entity"
  - C'est notre **entité**
- Maker a créé une classe PHP dans le dossier "/src/Repository"
  - C'est le **Repository** de notre entité (nous y reviendrons)

Base de données avec Doctrine

Création d'une entité avec MakerBundle

Regardons l'entité générée, c'est une classe PHP, MakerBundle a néanmoins ajouté des annotations.

- Annotation de création de table
  - **@ORM\Entity**
- Annotation de création de colonne
  - **@ORM\Column**
    - type : précise le type de la colonne

Base de données avec Doctrine

Création d'une entité avec MakerBundle

Désormais Doctrine a les informations nécessaires pour créer la table en base de données.

Cependant la création de la table n'est pas automatique, il faut lancer l'exécution du SQL.

- Pour cela nous allons utiliser les migrations

“

Qu’est-ce qu’une migration de base de données ?

Base de données avec Doctrine

Présentation des migrations

La construction de la base de données est un ensemble de commandes SQL.

Chaque “étape” de création de la base de données sera une migration.

- Les migrations sont des sortes de “version” de la base de données. Elles permettent de :
  - De ne pas bidouiller avec PHPMyAdmin
  - Partager les nouvelles versions du SQL avec les autres développeurs

Base de données avec Doctrine

Présentation des migrations

La migration s'effectue en 2 étapes :

- Création de la migration
  - Création du SQL
- Exécution de la migration
  - Exécution du SQL

Base de données avec Doctrine

Création d'une migration avec Doctrine

Création de la migration :

- Doctrine regarde la différence entre le dossier Entity et l'état de la base de données
- Les différences sont exprimées par du SQL (création/modification/suppression de tables)

Base de données avec Doctrine

Création d'une migration avec Doctrine

Commande de création d'une migration

php bin/console doctrine:migrations:diff

ou

php bin/console make:migration

Les 2 commandes sont équivalentes.

Base de données avec Doctrine

Execution d'une migration avec Doctrine

Exécution de la migration :

- Doctrine exécute toutes les migrations en attente

Base de données avec Doctrine

Execution d'une migration avec Doctrine

Commande d'exécution d'une migration

```
php bin/console doctrine:migrations:migrate
```

“

Comment gérer les clefs étrangères ?

Base de données avec Doctrine

Gestion des clefs étrangères

Base de données relationnelle :

Une clef étrangère est une colonne dans la table.

Base de données objet :

Une entité étrangère est une propriété dans une entité.

Base de données avec Doctrine

Gestion des clefs étrangères

MakerBundle peut nous aider à créer ses associations entre entités.

Pour cela MakerBundle a besoin de savoir quelle type de relation lie les 2 entités :

- OneToOne
- ManyToOne (inversé par rapport au MCD)
- ManyToMany

Le type de relation sera le type que nous allons fournir à MakerBundle dans la commande “make:entity”.

Base de données avec Doctrine

Gestion des clefs étrangères

Avec Doctrine, il n'y a pas de clef étrangère. Peu importe la relation, une propriété est créée dans l'entité appelée "maître".

- Dans le cas d'un OneToOne et d'un ManyToOne
  - Cette propriété n'est pas une clef étrangère
  - Cette propriété est une entité "inversée"
- Dans le cas d'un ManyToMany (et d'un OneToMany)
  - Cette propriété n'est pas une clef étrangère
  - Cette propriété est un tableau d'entités "inversées"

Base de données avec Doctrine

Gestion des clefs étrangères

Doctrine va alors se charger de créer les clefs étrangères et éventuellement des tables intermédiaires, nous n'avons pas à le faire.

Nous pouvons désormais créer l'ensemble de notre base de données.

Base de données avec Doctrine

Gestion des clefs étrangères

Nous allons mettre en place une base de données illustrant les **3 types de relations**.

Quel exemple voulez-vous prendre ?

- Le début d'un ecommerce ?
- Un portfolio ?
- Un blog ?
- D'autres idées ?

Merci de votre attention !

Pour la suite : Les données avec Doctrine, utilisation de bundles tiers