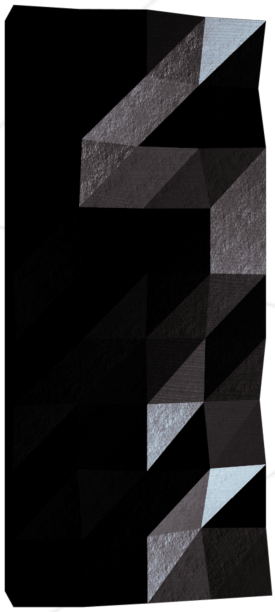

Formation PHP -Symfony

D07 - Concepts Symfony Avancés

Résumé: Cette journée est consacrée à l'apprentissage de concepts avancés de Symfony tels que les traducitons, les configurations personnalisées de bundles et les tests unitaires.

Table des matières

I	**Préambule**	**2**
II	**Consignes**	**3**
III	**Règles spécifiques de la journée**	**4**
IV	**Exercice 00**	**5**
V	**Exercice 01**	**6**
VI	**Exercice 02**	**7**
VII	**Exercice 03**	**9**
VIII	**Exercice 04**	**10**

Chapitre I

Préambule

Jusqu'à présent, vous avez été initiés à Symfony et son fonctionnement de base, SQL et ORM, vous avez appris à propos de l'authentification mais beaucoup reste à découvrir. Aujourd'hui nous allons découvrir de nouveaux concepts avancés utilisés au quotidien par les développeurs Symphony.

Chapitre II

Consignes

Sauf contradiction explicite, les consignes suivantes seront valables pour tous les jours de cette Piscine.

- Seul ce sujet sert de référence : ne vous fiez pas aux bruits de couloir.
- Le sujet peut changer jusqu'à une heure avant le rendu.
- Les exercices sont très précisément ordonnés du plus simple au plus complexe. En aucun cas nous ne porterons attention ni ne prendrons en compte un exercice complexe si un exercice plus simple n'est pas parfaitement réussi.
- Attention aux droits de vos fichiers et de vos répertoires.
- Vous devez suivre la procédure de rendu pour tous vos exercices. L'url de votre dépot `GIT` pour cette journée est disponible sur votre intranet.
- Vos exercices seront évalués par vos camarades de Piscine.
- En plus de vos camarades, vous pouvez être évalués par un programme appelé la Moulinette. La Moulinette est très stricte dans sa notation car elle est totalement automatisée. Il est donc impossible de discuter de sa note avec elle. Soyez d'une rigueur irréprochable pour éviter les mauvaises surprises.
- Les exercices shell doivent s'éxcuter avec `/bin/sh`.
- Vous ne devez laisser aucun autre fichier que ceux explicitement specifiés par les énoncés des exercices dans votre dépot de rendu.
- Vous avez une question ? Demandez à votre voisin de droite. Sinon, essayez avec votre voisin de gauche.
- Toutes les réponses à vos questions techniques se trouvent dans les `man` ou sur Internet.
- Pensez à discuter sur le forum Piscine de votre Intra et sur Slack !
- Lisez attentivement les exemples car ils peuvent vous permettre d'identifier un travail à réaliser qui n'est pas précisé dans le sujet à première vue.
- Réfléchissez. Par pitié, par Thor, par Odin !

Chapitre III

Règles spécifiques de la journée

- Pour cette journée, votre repository doit contenir seulement une application Symfony.
- Vous devrez respecter les bonne pratiques du framework Symfony.

Chapitre IV

Exercice 00

	Exercice :
Exercice 00 : Bundle personnalisé basique	
Dossier de rendu : *ex/*	
Fichiers à rendre : `Fichiers et répertoires de votre application`	
Fonctions Autorisées : `Toutes méthodes`	

Pour cet exercice vous devez mettre en place une application Symfony et nu bundle personnalisé qui sera étendu au fur et à mesure par les exercices suivants. Le bundle s'appelera **D07Bundle** et c'est le seul bundle qui se trouvera dans votre application. Le répertoire du bundle de devra pas contenir d'autres répertoires à par un répertoire vide pour le controleur pour le moment. Aussi, le répertoire **app/Resources/views** devra être vide pour le moment.
Vous devrez obtenir une erreur 404 si vous essayez d'accéder à votre site sur le chemin par défaut.

Chapitre V

Exercice 01

	Exercice :
Exercice 01 : Configuration du Bundle	
Dossier de rendu : *ex/*	
Fichiers à rendre : `Fichiers et répertoires de votre application`	
Fonctions Autorisées : `Toutes méthodes`	

Pour cet exercice vous ajouterez des fonctionnalités avancées à votre bundle. Vous êtes probablement déjà familiers avec les configurations système de Symfony et ce qu'un bundle. Chaque bundle peut avoir sa propre configuration. Premièrement vous devez créer un fichier de configuration pour votre bundle. La configuration de votre bundle devrait avoir la clé racine (root key) **d07** et les 2 sous-clés suivantes :

- **number** - obligatoire, devrait être un entier
- **enable** - facultatif, devrait être un boolean, ayant par défaut la valeur true

Apres avoir mis en place le fichier de configuration pour le bundle, ajouter les clés obligatoires de votre configuration bundle à votre fichier **app/config/config.yml**.

Afin de tester que votre configuration fonctionne correctement, créer un nouvelle classe controleur nommée **Ex01Controller** avec une action nommée **ex01Action** avec le route **/ex01**. Cette route devra simplement retourner un texte plein contenant la valeur de la sous-clé **number** de votre configuration du bundle.

Astuce : *Le controleur dispose d'une fonction* ***getParameter*** *afin de retourner n'importe quel paramètre du container. Cependant, assurez-vous que la configuration de votre bundle est disponible dans le container. Peut-être qu'une classe Extension peut aider ?*

Chapitre VI

Exercice 02

	Exercice :
Exercice 02 : Traductions	
Dossier de rendu : *ex/*	
Fichiers à rendre : `Fichiers et répertoires de votre application`	
Fonctions Autorisées : `Toutes méthodes`	

Il est temps de rendre votre application capable d'afficher des contenus dans plusieurs langues, **en** et **fr**. Parametrer le *locale* par défaut et activer les traductions. Créer 2 nouveaux fichiers dans le répertoire **app/Resources/translations**. Ils s'appelleront **messages.en.yml** et **messages.fr.yml**.

Créer un controleur personnalisé **Ex02Controller** avec une action personnalisée **translationsAction** qui prendra un paramètre facultatif **count**, entier, ayant comme valeur par défaut 0 et de valeurs possibles dans l'intervalle *0 à 9* La route pour cette action sera **/{_locale}/ex02/{count}**. Selon la valeur du paramètre **_locale**, le site sera affiché soit en anglaise soit en français. Maintenant créer un template appelé **ex02.html.twig** que sera retourné par votre action. Le template aura comme paramètres le nombre trouvé dans la configuration de votre bundle décrit dans l'exercice précédent ainsi que le paramètre **count**.

Le template devra contenir le texte suivant selon la langue et le texte devrait être récupéré à partir des fichiers de traduction .

Anglais

- *The config number is %number%* - le paramètre correct devra être passé à cette traduction
- *none/one/number %count%* - selon la valeur du paramètre une traduction différente sera utilisée

Français

- *Le numéro de configuration est %number%* - le paramètre correct devra être passé à cette traduction

- *aucun/un/nombre %count%* - selon la valeur du paramètre une traduction différente sera utilisée

Astuce : *Utilisez les filtres Twig pour traductions et l'annotation* ***@Route*** *pour la validation des paramètres.*

Chapitre VII

Exercice 03

	Exercice :
Exercice 03 : Extension Twig & Injection de dépendance	
Dossier de rendu : *ex/*	
Fichiers à rendre : `Fichiers et répertoires de votre application`	
Fonctions Autorisées : `Toutes méthodes`	

Je parie que vous vous demandez comment Twig fonctionne et comment ajouter vos propres fonctions et filtres à utiliser dans vos templates. Pour cela vous pouvez créer un extension Twig. La classe pour cette extension devra être **src/D07Bundle/Twig/Ex03Extension.php**.
Créer un nouveau filtre Twig et une fonction Twig. Le filtre s'appellera **uppercaseWords** et lorsqu'il est appliqué à une chaine de caractères il devra transformer en majuscule la première lettre de chaque mot. La fonction s'appellera **countNumbers** et retournera le nombre de chiffres dans une chaine de caractères.
Ces fonctions ne devraient pas être créées dans la classe extension mais dans une classe séparée de service **src/D07Bundle/Service/Ex03Service**. Cette classe de service sera ensuite injectée dans l'extension Twig.
Ensuite créer un controleur appelé **Ex03Controller** ayant une action **extensionAction**, la route **/ex03** et un template **ex03.html.twig**. Le template devra utiliser le filtre et la fonction sur la chaine de caractère que vous voulez en provenance des traductions et afficher les résultats.

Astuce : *Les services peuvent être définis dans le fichier* ***app/config/services.yml****.*

Chapitre VIII

Exercice 04

	Exercice :
Exercice 04 : Tests unitaires	
Dossier de rendu : *ex/*	
Fichiers à rendre : `Fichiers et répertoires de votre application`	
Fonctions Autorisées : `Toutes méthodes`	

Il est temps de tester le service créé dans l'exercice précédent à l'aide de **PHPUnit** et des tests unitaires. Créer une classe pour votre service qui devra suivre la convention de nommage de tests Symfony 2.8 et ensuite tester les 2 fonctions , **uppercaseWords** et **countNumbers**, dans ce service.
Ecrire au moins 3 tests différents pour chaque fonction et assurez vous de respecter les conseils de l'exercice précédent.
Tous les tests doivent passer !