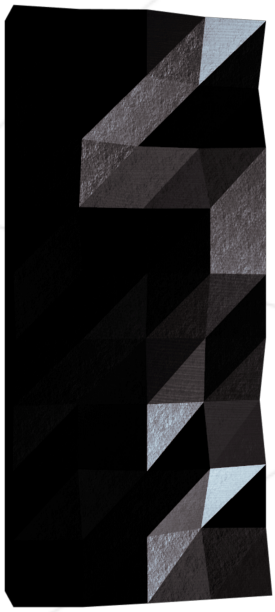

# Formation PHP Symfony - 1

## La base de Symfony

*Résumé: En suivant ce projet, vous apprendrez à connaître les concepts de base du framework Symfony.*

*Version: 1*

# Table des matières

# Chapitre I

# Préambule

The void type, in several programming languages derived from C and Algol68, is the type for the result of a function that returns normally, but does not provide a result value to its caller. Usually such functions are called for their side effects, such as performing some task or writing to their output parameters. The usage of the void type in such context is comparable to procedures in Pascal and syntactic constructs which define subroutines in Visual Basic. It is also similar to the unit type used in functional programming languages and type theory.

# Chapitre II

# Règles communes

- Votre projet doit être réalisé dans une machine virtuelle.
- Votre machine virtuelle doit avoir tout les logiciels necessaire pour réaliser votre projet. Ces logiciels doivent être configurés et installés.
- Vous êtes libre sur le choix du systmème d'exploitation à utiliser pour votre machine virtuelle.
- Vous devez pouvoir utiliser votre machine virtuelle depuis un ordinateur en cluster.
- Vous devez utiliser un dossier partagé entre votre machine virtuelle et votre machine hote.
- Lors de vos évaluations vous allez utiliser ce dossier partager avec votre dépot de rendu.
- Vos fonctions de doivent pas s'arrêter de manière inattendue (segmentation fault, bus error, double free, etc) mis à part dans le cas d'un comportement indéfini. Si cela arrive, votre projet sera considéré non fonctionnel et vous aurez 0 au projet.
- Nous vous recommandons de créer des programmes de test pour votre projet, bien que ce travail **ne sera pas rendu ni noté**. Cela vous donnera une chance de tester facilement votre travail ainsi que celui de vos pairs.
- Vous devez rendre votre travail sur le git qui vous est assigné. Seul le travail déposé sur git sera évalué. Si Deepthought doit corriger votre travail, cela sera fait à la fin des peer-evaluations. Si une erreur se produit pendant l'évaluation Deepthought, celle-ci s'arrête.

# Chapitre III

# Consignes spécifiques à cette journée

- Chaque exercice doit être résolu dans un bundle différent.
- Chaque controller doit aller dans le répertoire Controller à l'intérieur du bundle.
- Les noms de fichiers contenant les classes controller devraient contenir le suffixe `Controller` et devraient contenir une classe du même nom.
- Le contenu de chaque page devrait se trouver dans des fichiers twig et chaque fichier twig devrait avoir l'extension `.html.twig`
- Le serveur utilisé aujourd'hui est celui intégré dans Symfony. Il devrait être démarré et arrêté à l'aide des commandes console Symfony.
- Seulement les URLs explicitement demandés devraient afficher une pages sans erreur. Les URLs non configurées devrait générer une erreur 404.
- Les URLs demandés devraient fonctionner avec ou sans le slash final. Ex : `/ex00` et `/ex00/` doivent fonctionner de manière identique.
- Si aucune information contraire n'est explicitement présente, vous devez assumer les versions de langages suivantes :
  - `PHP - Symfony version LTS`
  - `HTML 5`
  - `CSS 3`

# Chapitre IV

# Exercice 00

|  | Exercice : 00 |
|---|---|
| Exercice 00 : Première page. | |
| Dossier de rendu : $ex00/$ | |
| Fichiers à rendre : `Tous les fichiers de l'application.` | |
| Fonctions Autorisées : `Toute fonctionnalité de Symfony.` | |

Dans cet exercice vous devez créer une simplemage dans une application Symfony.

Premièrement, créez un projet Symfony à l'aide du composer :

```
composer create-project symfony/skeleton d04 "^versionLTS"
```

`Vous devez changer la partie versionLTS par la vra version LTS de symfony.`

Ou à l'aide du binaire symfony :

```
symfony new d04 --version=lts
```

Créer un bundle `E00Bundle` et référencez le dans le fichier Bundle.

`Vous pouvez créer le bundle à l'aide des commandes natives de symfony.`

Pour définir les routes vous devrez utiliser les annotations dans le fichier controleur (controller) et vous devriez référencer les routes dans le fichier `config/routing.yml`.

Pour cet exercice vous devez créer une page qui est visible à l'URL suivante : `/e00/firstpage`. Dans la page vous devez afficher la chaine suivante : `"Hello world!"`.

Vous ne devez utiliser aucun template, vous devez avoir seulement le controller et utiliser l'objet `Response` du composant HttpFoundation.

# Chapitre V

# Exercice 01

|  | Exercice : 01 |
|---|---|
| Exercice 01 : Pages multiples. | |
| Dossier de rendu : $ex01/$ | |
| Fichiers à rendre : `Tous les fichiers de l'application.` | |
| Fonctions Autorisées : `Toute functionnalité de Symfony.` | |

A l'aide du projet créé dans l'exercice précédent, créer un nouveau bundle `E01Bundle` et référencez le dans le fichier Bundle.

`Vous pouvez créer le bundle à l'aide des commandes natives de symfony.`

Pour définir les routes vous devrez utiliser les annotations dans le fichier controleur (controller) et vous devriez référencer les routes dans le fichier `config/routing.yml`.

Vous devez créer une application qui affichera une page simple découpé en 3 partie avec trois articles. Seulement les fichiers de type `.html.twig` seront intégrés dans l'application.

L'application devrait contenir une page principale à l'adresse `/e01` avec au minimum un header un footer et la page de base. Vous devez lister tous les liens vers les articles que vous devez mettre en place. Ces articles seront accessibles à travers une URL structuré comme suit : `/e01/[articles]`. Par exemple : `/e01/goéland` devra afficher le contenu d'un article sur les goélands ainsi que le header et le footer de votre page principale.

Vous devez avoir au moins 3 articles disponible, vous pouvez choisir les sujets que vous souhaitez.

Dans le cas où une URL avec un article inexistante est appelée, l'application devrait afficher la page principale. Ainsi, si vous accédez à `/e01/wrongurl` vous devrez voir la page principale avec le liste des liens vers les pages des articles.

Vous pouvez définir la liste des catégories dans un tableau/array à l'intérieur du controller ou bien utiliser les paramètres de réglage dans le fichier : `Resources/config/services.yml` du Bundle.

Afin de permettre à vous pages de produire du HTML correct vous devez créer un fichier `base.html.twig` qui sera votre template de base et qui contiendra les balises `<html>` et `<body>` de la page. Toutes les pages des articles devraient utiliser ce template en surchargeant le block `content` du template de base.

# Chapitre VI

# Exercice 02

|  | Exercice : 02 |
|---|---|
| Exercice 02 : Premier formulaire. | |
| Dossier de rendu : $ex02/$ | |
| Fichiers à rendre : `Tous les fichiers de l'application.` | |
| Fonctions Autorisées : `Toute functionnalité de Symfony.` | |

En utilisant le projet créé dans le premier exercice, créer un nouveau bundle `E02Bundle` et référencez le dans le fichier Bundle.
Vous savez maintenant déjà comment définir les routes.

Dans cet exercice vous aurez à l'URL `/e03` un formulaire qui aura deux inputs : une zone de texte libellé "Message" contenant un message et une liste déroulante libellée "Include timestamp" avec deux valeurs sélectionnables : *Yes* et *No*. Les informations envoyés par le formulaire devront être écrites dans un fichier selon la logique suivante : si l'option "Yes" est sélectionnée l'application écrira le message et le timestamp sur une ligne du fichier ; si l'option "No" est sélectionnée seulement le message sera écrit sur une ligne dans le fichier.

Vous devrez ajouter une validation du message côté serveur afin de s'assurer que le message n'est pas vide. Attention, vous ne devrez pas utiliser la validation HTML5 côté client.

Le nom du fichier dans lequel seront stockés les messages sera défini dans :
`app/config/services.yml` et devra être créé dans le répertoire racine du projet aux côtés des fichiers composer.lock et composer.json. Si le fichier n'existe pas l'application ne devra pas planter mais elle devra créer le fichier à l'endroit indiqué avec le nom spécifié dans `config/services.yml`.

Le formulaire pourra être envoyé plusieurs fois, le contenu du fichier devra alors intégrer les nouveaux éléments envoyés par le formulaire sur une nouvelle ligne.

Après l'envoi du formulaire la page résultante sera celle du formulaire et le formulaire

devrait rappeler les informations envoyés (le message et le choix dans la liste déroulante devront être les mêmes que ceux envoyés). La dernière ligne du fichier texte (contenant les messages successivement envoyés par le formulaire) devra également être affichée sous le formulaire.

Vous ne devez pas hardcoder/écrire le formulaire directement en HTML. Vous devez utiliser le composant Form de Symfony ainsi que les types natifs de Symfony.

# Chapitre VII

# Exercice 03

|  | Exercice : 03 |
|---|---|
| Exercice 03 : Cinquante nuances de couleurs. | |
| Dossier de rendu : $ex03/$ | |
| Fichiers à rendre : `Tous les fichiers de l'application.` | |
| Fonctions Autorisées : `Toute functionnalité de Symfony.` | |

Créer un nouveau bundle `E03Bundle` et référencez le dans le fichier Bundle. Oubliez pas les routes.

Dans cet exercice vous devez créer une page que affiche un nombre de nuances pour les couleurs suivantes : *black*, *red*, *blue*, *green*. Le nombre de nuances devra être configurable dans le fichier `config/services.yml` dans le paramètre `e03.number_of_colors`.

La page devra contenir un header pour la tableau contenant les couleurs pour lesquelles les nuances sont affichées.

Les cases du tableau doivent avoir pour caractéristiques :

- Hauteur : 40 pixels.
- Largeur : 80 pixels.
- Couleur de fond : une nuance de la couleur à laquelle correspond la colonne.

Le tableau devra afficher les nuances des 4 couleurs de telle façon à ce que l'on observe un dégradé sur autant de lignes que spécifiées dans le paramètre `e03.number_of_colors`.

Il est interdit de hardcoder les valeurs des couleurs en HTML, vous devez les créer dynamiquement et les envoyer au template Twig en tant que paramètre à partir du controller.

# Chapitre VIII

# Rendu et peer-évaluation

Rendez votre travail dans votre dépôt `Git` comme d'habitude. Seul le travail présent dans votre dépôt sera évalué en soutenance. Vérifiez bien les noms de vos dossiers et de vos fichiers afin que ces derniers soient conformes aux demandes du sujet.

`L'évaluation se déroulera sur l'ordinateur du groupe évalué.`