**Symfony**

Table des matières

[Genéralité et installation 2](#_Toc77417575)

[Composer 2](#_Toc77417576)

[Initialisation d’un dossier symfony 2](#_Toc77417577)

[Php storm 2](#_Toc77417578)

[Hello world 3](#_Toc77417579)

[Création du routeur 3](#_Toc77417580)

[Affichage avec twig 3](#_Toc77417581)

[Twig 4](#_Toc77417582)

[Les balises twig 4](#_Toc77417583)

[Les filtres twig 4](#_Toc77417584)

[Bloc et héritage 5](#_Toc77417585)

[Gestion des assets et des URLs 5](#_Toc77417586)

[Passer des variables à la vue 6](#_Toc77417587)

[Attaque XSS sous twig 6](#_Toc77417588)

# Genéralité et installation

Symfony est un Framework qui permet d’avoir une structure spécifique afin d’avoir une organisation cohérente et qui permet un travail en équipe, évite de « réinventer la roue » et évite quelques problèmes de sécurité.

Il existe 2 versions possible de symfony sur laquel travaillé :

* le skeletons (version minimaliste et personnalisable)
* le website-skeleton (version complète et personnalisable)

## Composer

Composer permet en ligne de commande d’initialiser un dossier et d’installer des dépendances. Il faut installer commander sur le site officiel mais aussi cmder pour avoir un interpréteur de ligne de commande mieux que celui de windows.

## Initialisation d’un dossier symfony

Une image contenant texte

Description générée automatiquement

Puis dans le dossier que l’on viens de créer :



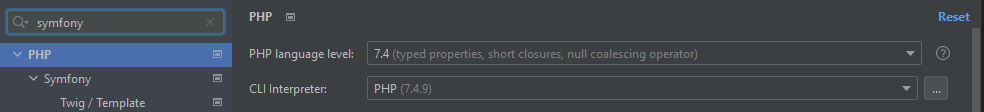
*Pour vérifier sa bonne implémentation, présence d’un fichier .htaccess dans /public.*

## Php storm

Une image contenant texte

Description générée automatiquementSur **PHPStorm** il faut installer le plugin symfony, l’activer quand on lance un projet symfony (une pop-up en bas a droite) et configurer dans les paramètre tel que

* enable plugin doit être coché
* Décoché les code folding (qui raccourcis le code)
* Ajouter le dossier /.idea/ dans gitignore



# Hello world

Si problème de drivers commenter postgres et décommenter mysql avec les bon id et données un nom de bdd



Puis taper la commande dans le terminal

Une image contenant texte, orange

Description générée automatiquement

## Création du routeur

Dans src>controller créer un fichier nommé MainController

Une image contenant texte

Description générée automatiquement

## Affichage avec twig

Dans templates créer un dossier main>home.html.twig

Une image contenant texte

Description générée automatiquement

# Twig

Twig permet en plus d’écrire du html classique d’y ajouter du code PHP. Twig est minimaliste et performant

Il existe 3 délimiteurs en twig

* **{# … #}** : pour écrire des commentaires
* **{{ … }}** : appeler double moustache permet de faire un echo
* **{% … %}** : pour exécuter du code (ce n’est pas du code php mais des balises twig)

## Les balises twig

Une image contenant texte

Description générée automatiquementUne image contenant texte

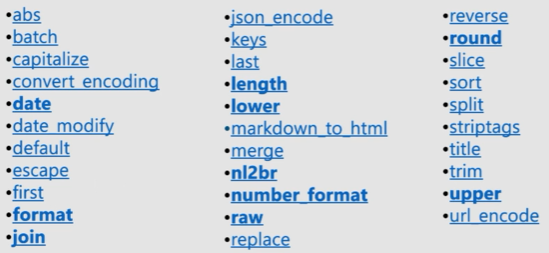
Description générée automatiquement



## Les filtres twig

Une image contenant texte

Description générée automatiquementLes filtres permettent de modifier des variables, pour les utiliser on ajoute un pipe | après la variable



## Bloc et héritage

Avec twig nous pouvons créer un « gabarit » de page avec des bloc qu’ont laisse vide (ou une valeur par défaut).

Une image contenant texte

Description générée automatiquement

Et maintenant notre page home ressemble à ça

Une image contenant texte

Description générée automatiquement

## Gestion des assets et des URLs

Pour avoir une gestion des url (afin d’éviter les problème avec les chemins relatif qui ne trouverons pas toujours le bonne emplacement) on utilise **asset(‘folder/rss’)** qui permet de rediriger vers le dossier /public



Et pour les url de page on utilise **path(‘name\_route’)**

Une image contenant texte

Description générée automatiquement

## Passer des variables à la vue

Pour passer des variable à la vue, il faut fournir en 2em arguments de la méthode render un tableau

Une image contenant texte

Description générée automatiquement

On peut aussi utiliser la méthode compact, qui ira chercher les variable qui porte le nom entre « » tel que :

Une image contenant texte

Description générée automatiquement

Il ne reste plus qu’a l’utiliser dans notre twig



## Attaque XSS sous twig

Par défaut twig « protège » des faille XSS les echo que l’ont fait avec **{{ … }}** par défaut. Dans certain cas on pourrait avoir envie de garder la lecture des balises en utilisant **raw**, **stripstags** si on fait confiance au data que l’on affiche, sinon si on est entre les deux on pourrait utiliser une bibliothèque tel que **HTMLPurifier** qui va limite l’autorisation à certaine balise.

Exemple :

# Routage et controleurs

Le routage symfony permet entre autres de centralisé les accès au URL avec le MainController, de pouvoir avoir des url définies plus simplement.

Pour effectuer le routage il faut faire un commentaire avant notre fonction qui commence toujours par **/\*\*** et dans le commentaire **@Route** (penser à importer la class correspondante). Le premier paramètre est le pattern URL.

Une image contenant texte

Description générée automatiquement

## Le parametre URL

Une image contenant texte

Description générée automatiquementIl doit être unique et en général définie des le début de l’application. Dans une URL nous pouvons avoir une partie variable (notamment avec les **GET**) c’est ce qu’ont appel des **jokers**.

Une image contenant texte

Description générée automatiquementPour éviter certains conflits il est une bonne pratique de définie via une expression rationnelle le résultat attendus, par exemple ici nous acceptons 1 ou plusieurs chiffres

Une image contenant texte

Description générée automatiquementOn peut également définir une valeur par défaut si l’URL n’en as pas

## Les routes : requirements et autres conditions

Nous pouvons avoir des adresse identique en URL qui rendront un résultat différent suivant le type de requête (**GET,POST**) ce qui sera très utile pour construire une API REST.

Une image contenant texte

Description générée automatiquementLe nom des routes est très fortement recommandé, elles permettent de générer les **URL** correspondant aux routes, par convention on les nomme avec le nom du contrôleur (ici main) et le nom de la méthode en **snake\_case**.

## Quelques commandes de débogage pour les routes

Une image contenant texte

Description générée automatiquement

## Les controleurs

Quand on créer un contrôleur il faut qu’il hérite de **AbstractControlle**r, ce qui permettra d’accéder a un tas de fonctionnalité. Il faut essayer de limiter nos contrôleurs à une dizaine d’URL maximum, chaque méthode du contrôleur représente une page. Grace au contrôleur nous avons un accès facile à **HttpFoundation\Request** et nous devons obligatoirement retourner un **HttpFoundation\Response.**

**Exemple de retour possible :**



## Préfixe sur les controleurs

Une image contenant texte

Description générée automatiquement

## créer un nouveau controlleur en commande

Une image contenant texte

Description générée automatiquement

## Débogage

Une image contenant texte

Description générée automatiquement

Avec **dump** on verra dans la barre du bas afin de voir le détail de la variable.

**dd()** permet de le voir directement dans la page

Une image contenant texte

Description générée automatiquementEn bas du site est disponible une barre pour avoir des infos et lors du click sur la barre on as accès au profiler qui servira pour le débogage, les formulaire, la sécurité et les requête en base de données.