|  |
| --- |
|  |
| Symfony 3.4 |
| **BEST PRACTICE** |

Version de la documentation

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Auteur | Descriptif |
| 2.0 | 02/10/2019 | Manitra B. | Architecture&Configuration |

Table des matières

[I. Installation de symfony 4](#_Toc21522018)

[II. Bundles utiles 5](#_Toc21522019)

[III. Installation des bundles 5](#_Toc21522020)

[1. Installation de FOSUserBundle 5](#_Toc21522021)

[2. Génération de nouvelle bundle 5](#_Toc21522022)

[3. Génération entity 6](#_Toc21522023)

[4. Cryptage des données 6](#_Toc21522024)

[IV. Différence entre serialize et json\_encode 7](#_Toc21522025)

## Installation de symfony

* La bonne pratique recommande l’utilisation d’une version stable, dont Symfony 3.4 jusqu’en 2020. Ci-après la feuille de route listant la maintenabilité dans le temps des différentes versions :

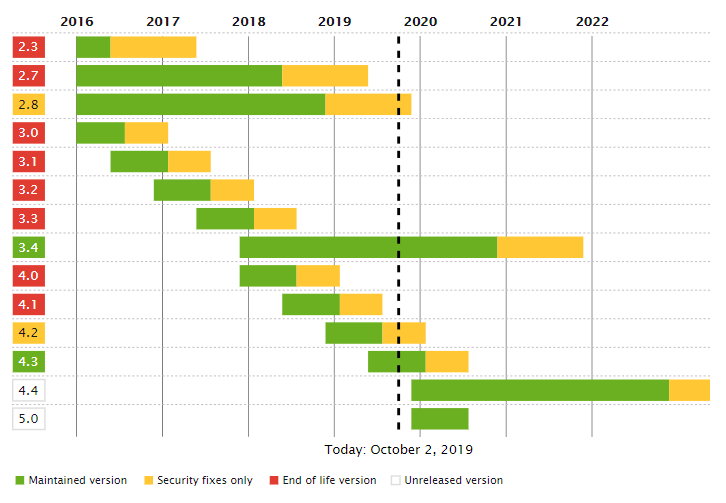


Figure 1: Feuille de route de maintenabilité de symfony

Réf : <https://symfony.com/roadmap>

* Pour l’installation, la liste complète des différentes possibilités de commandes se trouve dans la documentation officielle : <https://symfony.com/doc/3.4/setup.html>

Ci-présent l’installation le plus courant, à l’aide de la commande composer :

composer create-project symfony/framework-standard-edition my\_project\_name "3.4.\*"

php -r "echo ini\_get('memory\_limit').PHP\_EOL;"

* Installation du vendor

Si un problème de taille de mémoire autorisé se présente, il faut télécharger composer.phar puis exécuter la commande php suivante :

php –d memory\_limit=-1 composer.phar install

## Bundles utiles

* FOSUserBundle: Gestion CRUD utilisateur et gestion des rôles
* Translator

## Installation des bundles

Rechercher le nom du bundle dans <https://packagist.org/> pour trouver les commandes d’installation, les dépendances ainsi que la version minimale de PHP requise

### Installation de FOSUserBundle

La documentation accompagnée du guide d’installation se trouve sur le lien ci-après :

<https://github.com/FriendsOfSymfony/FOSUserBundle/blob/master/Resources/doc/index.rst>

composer require friendsofsymfony/user-bundle "~2.0"

Attention, la documentation présente une erreur dans security.yml

Remplacer la valeur de l’user\_cheker par :

user\_checker: security.user\_checker

Si la dépendance *templating* est manquante, il faut l’installer en executant la commande suivante : composer require symfony/templating

Puis, dans config.yml sous framework, ajouter :

templating:

{ engines: ['twig'] }

Liste de commandes utiles d’utilisation de Fos :

<https://symfony.com/doc/current/bundles/FOSUserBundle/command_line_tools.html>

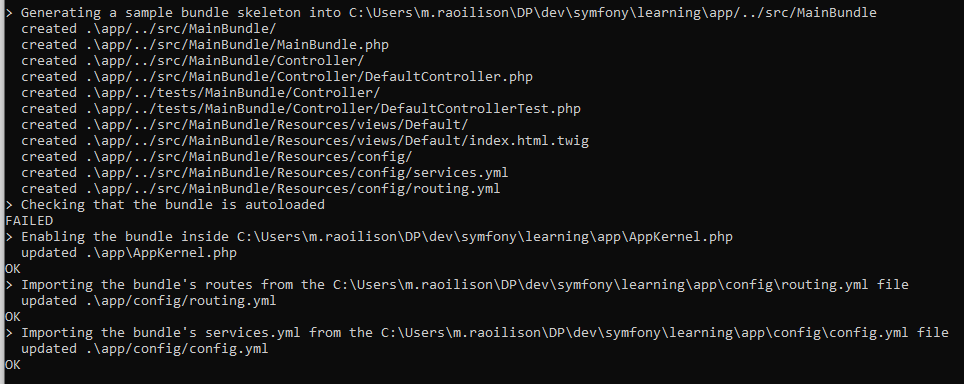
Ci-après un exemple de commande de création d’utilisateur :

php bin/console fos:user:create testuser test@example.com p@ssword

### Génération de nouvelle bundle

Entrer la commande suivante

php bin/console generate :bundle



Editer composer.json en :

"autoload": {

"psr-4": {

"": "src/"

Puis lancer la commande : composer dump-autoload pour prendre en compte la modification dans le fichier composer.json

### Effacer cache

php –d memory\_limit=-1 bin/console cache:clear –env=prod

Attention, la valeur -1 indique une infinité de mémoire allouée, ceci pourrait nuire au serveur. Définir une limite à 1024M par exemple est plus sécurisée.

### Génération entity

Entrer la commande suivante pour generer une nouvelle entite. Choisir annotation

php bin/console generate:doctrine:entity

php bin/console generate:bundle --dir=/var/www/myproject/src

### Enlever avertissement deprecated

Commenter dans php.ini error\_reporting = E\_ALL

### Cryptage des données

* OPENSSL

Référence : <https://www.php.net/manual/en/function.openssl-encrypt.php>

Cryptage avec openssl

**openssl\_encrypt** ( string $data , string $method , string $key [, int $options = 0 [, string $iv = "" [, string &$tag = NULL [, string $aad = "" [, int $tag\_length = 16 ]]]]] ) : string

Retourne une chaine de caractères brute ou encodé en base 64 en cas de réussie ou false en cas d’échec

## Différence entre serialize et json\_encode

Les deux méthodes permettent de formater un tableau de données sous une présentation textuelle pour le stockage dans une base de donnes (utilisation courante).

serialise($aData) ou json\_encode($aData)//unserialise($data) ou json\_decode($data, true)

La grande différence réside dans la présentation des formats de données.

* Fonction serialize :

{i:0;s:3:"PHP";i:1;s:4:"JAVA";i:2;s:1:"C";i:3;s:3:"C++";i:4;s:5:"MYSQL";i:5;s:6:"ORACLE";i:6;s:3:"VUE";s:3:"BOL";b:1;s:3:"Int";i:100;}}

* Fonction json\_encode:

{"data":{"0":"PHP","1":"JAVA","2":"C","3":"C++","4":"MYSQL","5":"ORACLE","6":"VUE","BOL":true,"Int":100}}

Le format json est plus universelle, ce qui n’est pas le cas de l’autre spécifique à PHP. Par ailleurs, le temps d’exécution de json\_encode est plus rapide, cette différence est devenue flagrante pour des données massives.

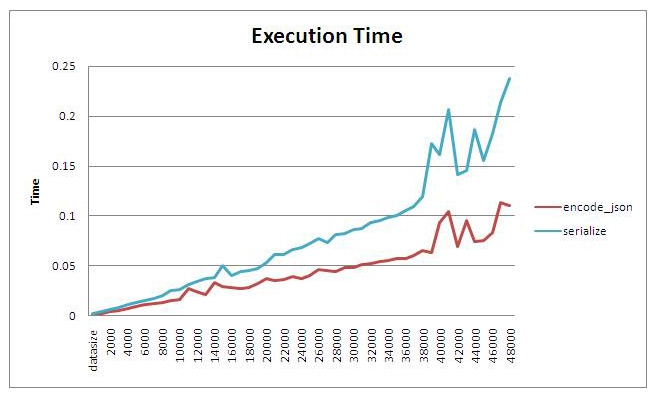


Figure 2 : Comparaison du temps d’exécution entre serialize et json\_encode

Référence : <https://medium.com/@moinuddinchowdhury/serialize-vs-json-67fe872a7755>