

# SERVICES

# 1. CONTENEUR DE SERVICES

Le concept



## PRINCIPE

Un service est une instance d'un objet qu'on veut partager dans toute l'application:

- EntityManager
- FormFactory
- Validator
- Mailer

Les librairies installées ajoutent automatiquement leurs services dans le conteneur

Le conteneur permet ensuite d'accéder aux services déclarés



## CONTENEUR

Voir la liste des services disponibles, et le détail de la configuration d'un service en particulier:

```
bin/console debug:container  
bin/console debug:container App\Repository\ArticleRepository
```

## 2. INJECTION DE DÉPENDANCES

Utiliser des services dans nos classes



## CRÉER UN SERVICE

Analyser config/services.yaml

Un service est automatiquement créé pour toute nouvelle classe déclarée dans le projet

Possibilité de faire de la configuration avancée depuis services.yaml:

- passer une variable d'environnement en argument
- passer des arguments hors services (nom de classe, ...)
- utiliser des factories
- décorer un service
- ...

# INJECTER UN SERVICE

```
● ● ●  
class ArticleManager  
{  
    public function __construct(  
        private EntityManagerInterface $entityManager,  
    ) {  
    }  
}
```

# POURQUOI ?



Symfony

Avantages:

- séparation des responsabilités (SoC)
- ré-utilisabilité
- facile à tester (unitaires & intégration)
- évolutivité

# 3. ATELIER

Organiser son code en services



# ATELIER

Créer un nouveau service ArticleManager

Injecter le service EntityManager

Déplacer le code permettant de créer et modifier des articles dans ce ArticleManager

Injecter ArticleManager dans les Controllers et l'utiliser pour enregistrer les articles en base

En option: ajouter l'auteur à l'article lors de la création

En option: créer un slug lors de la création avec symfony/string

# 4. LES COMMANDES

Créer des tâches pour la console



# COMMANDE

Une commande est une action exécutable en ligne de commande avec bin/console

Il est possible de créer ses commandes personnalisées

Maker permet de générer un squelette

Convention: situées dans App\Command, et nom suffixé par Command

# COMMAND

```
#[AsCommand(  
    name: 'app:articles:create',  
    description: '',  
)]  
class CreateArticleCommand extends Command  
{  
    protected function configure(): void  
    {  
        $this  
            ->addArgument('title', InputArgument::REQUIRED, 'Title, required')  
            ->addArgument('content', InputArgument::REQUIRED, 'Content, required')  
    }  
  
    protected function execute(InputInterface $input, OutputInterface $output): int  
    {  
    }  
}
```



Symfony

# ATELIER

Générer une nouvelle commande avec Maker

Créer une nouvelle commande pour créer un article, en utilisant ArticleManager. Cette article prend en paramètre un titre, un texte, et l'email de l'auteur

En option: afficher la liste des articles en utilisant une table