# Créer un service

## Le fichier `config/services.yaml`

Le fichier `config/services.yaml` permet de configurer les services :

```
parameters:

services:
  _defaults:
    autowire: true
    autoconfigure: true

  App\:
    resource: '../src/'
    exclude:
      - '../src/DependencyInjection/'
      - '../src/Entity/'
      - '../src/Kernel.php'
      - '../src/Tests/'
```

`_defaults` permet de définir les valeur par défaut de tous les services de l'application.

`autowire: true` permet d'activer l'`autowire` pour tous les services par défaut. Cela signifie qu'ils seront enregistrés dans le conteneur de service et que si vous les utilisez comme type ils seront automatiquement injectés par `Symfony`.

`autoconfigure: true` permet d'activer l'autoconfiguration pour tous les services par défaut. Cela permet d'activer automatiquement les commandes, les gestionnaires d'événement etc éventuellement définis dans le service.

`resource: '../src/'` permet d'indiquer que toutes les classes et toutes les interfaces dans le dossier `src` seront automatiquement ajoutées comme identifiants dans le conteneur de service. C'est extrêmement appréciable ! Lorsque nous créons des services ceux-ci deviennent automatiquement disponibles pour l'`autowiring`.

Cela ne compromet pas les performances car si un service n'est pas utilisé et donc injecté, il ne sera pas instancié par le conteneur de service : il ne prend donc pas de ressources systèmes.

`exclude:` permet à l'inverse d'exclure certains identifiants du conteneur de service car ils ne sont pas utilisables comme service.

# Créer un service

**Un service est simplement une classe qui a des méthodes qui réalisent des actions.**

Les services permettent d'éviter de dupliquer du code dans plusieurs contrôleurs pour réaliser les mêmes fonctionnalités.

Au lieu de copier / coller beaucoup de code, il vaut mieux l'extraire et en faire un service qui sera ensuite injecté dans les contrôleurs qui en auront besoin.

Pour rappel : un service n'est instancié qu'une seule fois par le conteneur de service. L'objet est ensuite envoyé à tous les contrôleurs et tous les services qui en ont besoin.

Voici un exemple de service qui ne fait rien :

```
<?php

namespace App\Service;

class MonSuperService
{
  public function neFaitRien(): void
  {
  }
}
```

# Injecter un service dans un autre service

**Pour injecter un service dans un autre service, il suffit de le déclarer comme dépendance sur le constructeur de la classe** :

```
<?php

namespace App\Service;

use Psr\Log\LoggerInterface;

class MonSuperService
{
```

```
  public function __construct(private LoggerInterface $logger)
  {
  }

  public function neFaitRien(): string
  {
    $this->logger->info('Je ne fais rien');
  }
}
```

Le service `LoggerInterface` est automatiquement injecté grâce à l'`autowiring` dans notre service `MonSuperService`.

# Exemple de la vidéo

Créez un dossier `Service` dans le dossier `src`.

Dans ce dossier, créez un fichier `MyLog.php` :

```
<?php

namespace App\Service;

use Symfony\Component\Filesystem\Filesystem;

class MyLog
{
  function __construct(private Filesystem $fs)
  {
  }

  public function writeLog(string $message)
  {
    $this->fs->appendToFile('logs/logs.txt', $message . PHP_EOL);
  }
}
```

*Si vous n'avez pas le `Generate namespace file for this file` il faut installer l'extension `PHP Namespace Resolver` dans `VS Code`.*

Le service est automatiquement disponible dans le conteneur de service car il est situé dans `src`, et comme nous l'avons vu, l'`autowiring` fonctionne pour le dossier `src` automatiquement.

Dans le `DefaultController`, nous pouvons l'injecter et l'utiliser :

```
<?php

namespace App\Controller;

use App\Service\MyLog;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Filesystem\Filesystem;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class DefaultController extends AbstractController
{
  function __construct(private Filesystem $fs)
  {
  }

  #[Route('/', name: 'index')]
  public function index(MyLog $log): Response
  {

    $log->writeLog("Première utilisation de votre service");
    return $this->json([]);
  }
}
```