Mémento Symfony 6/7

1. Structure de base d'un projet Symfony

Organisation des dossiers

```
project/
├── bin/            # Fichiers exécutables
├── config/         # Configuration
├── public/         # Point d'entrée web
├── src/            # Code source
│   ├── Controller/
│   ├── Entity/
│   ├── Repository/
│   └── Service/
├── templates/      # Templates Twig
├── tests/          # Tests
└── var/            # Fichiers temporaires
```

Fichiers de configuration principaux

- `config/packages/` : Configuration des bundles
- `config/routes.yaml` : Configuration des routes
- `config/services.yaml` : Configuration des services
- `.env` : Variables d'environnement

Gestion des environnements

- `.env` : Variables par défaut
- `.env.local` : Variables locales (non versionné)
- `.env.test` : Variables pour les tests
- `.env.prod` : Variables pour la production

2. Controllers

Structure de base

```php
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Attribute\Route;

class ExampleController extends AbstractController
{
    #[Route('/example', name: 'app_example')]
    public function index(): Response
    {
        return $this->render('example/index.html.twig', [
            'controller_name' => 'ExampleController',
        ]);
    }
}
```

Annotations et attributs de routage

```php
#[Route('/blog/{slug}', name: 'blog_show', methods: ['GET'])]
#[Route('/blog/{page<\d+>}', name: 'blog_list')]
```

Gestion des requêtes

```
use Symfony\Component\HttpFoundation\Request;

public function create(Request $request): Response
{
    $data = json_decode($request->getContent(), true);
    // ou
    $formData = $request->request->all();
}
```

Types de réponses

```
// Réponse HTML
return $this->render('template.html.twig', ['data' => $data]);

// Réponse JSON
return $this->json(['message' => 'Success']);

// Redirection
return $this->redirectToRoute('route_name');
```

3. Entités

Création d'une entité

```
<?php

namespace App\Entity;

use App\Repository\ProductRepository;
use Doctrine\ORM\Mapping as ORM;

#[ORM\Entity(repositoryClass: ProductRepository::class)]
class Product
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    #[ORM\Column(length: 255)]
    private ?string $name = null;
}
```

Types de données Doctrine

- `string`
- `integer`
- `float`
- `boolean`
- `datetime`
- `text`
- `json`
- `array`

Relations

```
// OneToMany
#[ORM\OneToMany(mappedBy: 'category', targetEntity: Product::class)]
private Collection $products;

// ManyToOne
#[ORM\ManyToOne(inversedBy: 'products')]
#[ORM\JoinColumn(nullable: false)]
private ?Category $category = null;

// ManyToMany
#[ORM\ManyToMany(targetEntity: Tag::class, inversedBy: 'products')]
private Collection $tags;
```

4. Templates Twig

Syntaxe de base

```
{# Commentaire #}
{{ variable }}
{% if condition %}
    {# code #}
{% endif %}

{% for item in items %}
    {{ item.name }}
{% endfor %}
```

Héritage de templates

```
{# base.html.twig #}
<!DOCTYPE html>
<html>
    <head>
        <title>{% block title %}{% endblock %}</title>
    </head>
    <body>
        {% block body %}{% endblock %}
    </body>
</html>

{# child.html.twig #}
{% extends 'base.html.twig' %}

{% block title %}Mon titre{% endblock %}
```

Filtres utiles

- `{{ variable|default('valeur par défaut') }}`
- `{{ variable|length }}`
- `{{ variable|date('Y-m-d') }}`
- `{{ variable|upper }}`
- `{{ variable|lower }}`

5. Formulaires

Création de formulaire

```
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\Form\FormBuilderInterface;

class ProductType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('name', TextType::class)
            ->add('price', NumberType::class);
    }
}
```

Types de champs

- `TextType`
- `EmailType`
- `PasswordType`
- `NumberType`
- `DateType`
- `ChoiceType`
- `EntityType`

6. Base de données

Configuration

```
# .env
DATABASE_URL="mysql://user:password@127.0.0.1:3306/db_name"
```

Migrations

```
# Créer une migration
php bin/console make:migration

# Exécuter les migrations
php bin/console doctrine:migrations:migrate
```

Requêtes DQL

```
$qb = $this->createQueryBuilder('p')
    ->where('p.price > :price')
    ->setParameter('price', 100)
    ->orderBy('p.name', 'ASC')
    ->getQuery();
```

7. Sécurité

Configuration

```
# config/packages/security.yaml
security:
    providers:
        app_user_provider:
            entity:
                class: App\Entity\User
                property: email
```

Authentification

```
use Symfony\Component\Security\Http\Attribute\IsGranted;

#[IsGranted('ROLE_USER')]
public function admin(): Response
{
    // ...
}
```

8. Services

Configuration

```
# config/services.yaml
services:
    App\Service\:
        resource: '../src/Service/'
        autowire: true
```

Injection de dépendances

```
class ProductService
{
    public function __construct(
        private EntityManagerInterface $entityManager,
        private LoggerInterface $logger
    ) {}
}
```

9. Commandes Doctrine utiles

Gestion des entités

```
# Créer un controller
php bin/console make:controller

# Créer un formulaire
php bin/console make:form

# Vider le cache
php bin/console cache:clear

# Créer une entité
php bin/console make:entity

# Mettre à jour le schéma de la base de données
php bin/console doctrine:schema:update --force

# Valider le mapping
php bin/console doctrine:schema:validate
```

Gestion des migrations

```
# Créer une migration
php bin/console make:migration

# Exécuter les migrations
php bin/console doctrine:migrations:migrate

# Voir le statut des migrations
php bin/console doctrine:migrations:status

# Annuler la dernière migration
php bin/console doctrine:migrations:migrate prev
```

Gestion de la base de données

```
# Créer la base de données
php bin/console doctrine:database:create

# Supprimer la base de données
php bin/console doctrine:database:drop --force

# Importer un schéma SQL
php bin/console doctrine:database:import file.sql
```