Fiche procédure Symfony

Table des matières

Installation de Symfony

1. Prérequis
    - PHP 8.2 ou supérieur, avec les extensions recommandées (Ctype, iconv, PCRE, Session, SimpleXML, Tokenizer).
    - Composer installé pour la gestion des dépendances.
    - (Optionnel) Symfony CLI pour simplifier la gestion des projets Symfony et vérifier la configuration de ton environnement.
2. Vérification de la configuration
    - Vérifie les prérequis avec la CLI Symfony :
    - `bash`

```
symfony check:requirements
```

    -
3. Installation de la CLI Symfony (optionnel mais recommandé)
    - Télécharge et installe la CLI depuis le site officiel ou via le terminal.

Création d’un projet Symfony

- Avec la CLI Symfony :
- `bash`

```
symfony new mon_projet --webapp
```

-
- Cette commande crée un dossier `mon_projet` avec l’architecture Symfony prête à l’emploi.
- Avec Composer :
- `bash`

```
composer create-project symfony/skeleton mon_projet
```

- Ou pour une application web complète :
- bash

```
composer create-project symfony/website-skeleton mon_projet
```

- Démarrer le serveur de développement :
- bash

```
symfony serve
```

- Ou :
- bash

```
php -S localhost:8000 -t public
```

Utilisation de Symfony

- Architecture MVC : Symfony utilise le modèle MVC (Modèle-Vue-Contrôleur).
- Routes : définies dans `config/routes.yaml` ou via annotations dans les contrôleurs.
- Contrôleurs : placés dans `src/Controller/`.
- Vues : créées en Twig dans `templates/`.
- Gestion des dépendances : avec Composer, via le fichier `composer.json`.
- Base de données : configuration dans `.env` et gestion via Doctrine ORM.

Bonnes pratiques

- Respecte les conventions Symfony pour la structure des dossiers et le nommage.
- Utilise Git pour la gestion des versions et travaille avec des branches pour chaque fonctionnalité ou correction.
- Sécurise ton application : protège les routes sensibles, utilise les outils de sécurité intégrés, mets à jour régulièrement les dépendances.
- Documente ton code et commente les parties complexes pour faciliter la maintenance.
- Automatise les tests et le déploiement avec des outils CI/CD.
- Exploite les outils de debug (Barre de debug Symfony, Profiler, etc.) pour améliorer la qualité du code.

- Reste à jour avec les dernières versions du framework pour bénéficier des nouveautés et correctifs.

Exemple de code Symfony

Contrôleur simple :

```php
// src/Controller/AccueilController.php
namespace App\Controller;

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

class AccueilController extends AbstractController
{
    #[Route('/', name: 'accueil')]
    public function index(): Response
    {
        return $this->render('accueil/index.html.twig', [
            'message' => 'Bienvenue sur Symfony !',
        ]);
    }
}
```

Vue Twig associée :

```text
{# templates/accueil/index.html.twig #}
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Accueil Symfony</title>
</head>
<body>
    <h1>{{ message }}</h1>
</body>
</html>
```