1. Installation du bundle

Dans votre projet Symfony 6.4, commencez par installer EasyAdmin :

```
composer require easycorp/easyadmin-bundle
```

Cela va ajouter le bundle à votre projet et, grâce à la flexibilité de Symfony Flex, l'enregistrer automatiquement dans le fichier `config/bundles.php`.

---

2. Configuration des routes

Si vous utilisez un **Dashboard** (la manière recommandée d'utiliser EasyAdmin 4), vous aurez un `DashboardController` (ex. `App\Controller\Admin\DashboardController`).
Pour que Symfony puisse acheminer correctement les requêtes vers l'interface d'administration, vous devez définir la route de votre tableau de bord (dashboard) dans un fichier de routes.

Par exemple, dans `config/routes/admin.yaml` :

```yaml
admin:
  path: /admin
  controller: App\Controller\Admin\DashboardController::index
```

> **Remarque** : Vous pouvez nommer et structurer votre fichier de routes comme vous le souhaitez, tant que vous déclarez la route pointant vers votre dashboard EasyAdmin.

---

3. Création du DashboardController

Créez un contrôleur qui va servir de tableau de bord principal. Vous pouvez le générer avec la commande :

```
php bin/console make:controller --no-template
```

Puis éditez ce fichier pour qu'il ressemble à quelque chose comme :

```php
<?php

namespace App\Controller\Admin;

use EasyCorp\Bundle\EasyAdminBundle\Controller\AbstractDashboardController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class DashboardController extends AbstractDashboardController
{
    #[Route('/admin', name: 'admin')]
    public function index(): Response
    {
        // Redirigez vers une page ou un CRUD particulier, ou bien affichez un template
        // De manière classique, on redirige vers un CRUD d'entité
        return $this->redirect($this->generateUrl('...'));
    }

    public function configureDashboard(): \EasyCorp\Bundle\EasyAdminBundle\Config\Dashboard
    {
        return \EasyCorp\Bundle\EasyAdminBundle\Config\Dashboard::new()
            ->setTitle('Mon Administration')
        ;
    }

    // Vous pouvez ajouter d'autres méthodes de configuration pour les menus, etc.
}
```

> **Note** : Il existe d'autres méthodes pour configurer le Dashboard (menus, thèmes, etc.). Pour plus de détails, consultez la documentation officielle EasyAdmin.

---

4. Création des CRUD Controllers

EasyAdmin utilise des contrôleurs spécifiques (ex. `ProductCrudController`, `UserCrudController`, etc.) pour gérer le CRUD de vos entités.
Vous pouvez les créer de façon manuelle ou vous appuyer sur les commandes `make:admin:crud` (si vous avez installé le **Maker Bundle** spécifique d'EasyAdmin, lorsqu'il est disponible). Sinon, vous pouvez créer un contrôleur CRUD à la main :

```php
<?php

namespace App\Controller\Admin;

use App\Entity\Product;
use EasyCorp\Bundle\EasyAdminBundle\Controller\AbstractCrudController;
use EasyCorp\Bundle\EasyAdminBundle\Config\Fields\TextField; // Exemple
use EasyCorp\Bundle\EasyAdminBundle\Config\Fields\IdField;   // Exemple

class ProductCrudController extends AbstractCrudController
{
    public static function getEntityFqcn(): string
    {
        return Product::class;
    }

    public function configureFields(string $pageName): iterable
    {
        return [
            IdField::new('id')
                ->hideOnForm(),         // Masque l'ID dans le formulaire
            TextField::new('name', 'Nom du produit'),
            // Ajoutez les champs selon vos propriétés d'entité
        ];
    }
}
```

Ensuite, n'oubliez pas de déclarer ce CRUD Controller dans votre `DashboardController` pour qu'il soit accessible depuis le menu EasyAdmin :

```php
use App\Controller\Admin\ProductCrudController;
use EasyCorp\Bundle\EasyAdminBundle\Config\MenuItem;

// ...

public function configureMenuItems(): iterable
{
    return [
        MenuItem::linkToDashboard('Dashboard', 'fa fa-home'),
        MenuItem::linkToCrud('Produits', 'fas fa-box', Product::class)
            ->setController(ProductCrudController::class),
        // ...
    ];
}
```

5. Vérifier le bon fonctionnement

- Lancez votre serveur Symfony :

  ```
  symfony serve
  ```

  (ou `php -S 127.0.0.1:8000 -t public` si vous utilisez le serveur PHP interne)

- Accédez à l'URL **`/admin`** (ou l'URL définie dans vos routes) dans votre navigateur.
- Vous devriez voir le dashboard EasyAdmin et pouvoir naviguer jusqu'à vos entités pour les gérer (list, edit, create, etc.).

---

6. Aller plus loin : personnalisation

EasyAdmin permet de nombreuses personnalisations :

1. **Configuration des champs** : vous pouvez définir les champs, leur label, leur type, l'ordre, les conditions d'affichage, etc.
2. **Actions personnalisées** : vous pouvez ajouter ou modifier des actions (boutons, actions massives, etc.).
3. **Role-based access** (sécurité) : vous pouvez conditionner l'accès à certaines sections de l'administration en fonction des rôles.
4. **Théming** : vous pouvez personnaliser (dans une certaine limite) l'apparence en adaptant vos assets ou en surchargeant les templates.

Toutes ces personnalisations sont documentées sur le site officiel :
Documentation EasyAdmin

Récapitulatif

1. **Installer** : `composer require easycorp/easyadmin-bundle`
2. **Configurer la route** vers votre Dashboard.
3. **Créer le DashboardController** (héritant de `AbstractDashboardController`).
4. **Créer les CRUD controllers** (héritant de `AbstractCrudController`) pour chacune de vos entités à gérer.

5. **Déclarer vos CRUD controllers** dans votre `DashboardController` pour qu'ils apparaissent dans les menus (ou y accéder par URL directement).
6. **Lancer votre serveur** et accéder à votre interface d'administration via `/admin`.