esprit Se former autrement HONORIS UNITED UNIVERSITIES	Année Universitaire : 2025-2026 **Workshops Framework Symfony 6** **UP-Web**

Workshop : Intégration Template

Objectif

L'objectif de cet atelier est de mettre en œuvre l'intégration d'un (Template) au sein d'une application Symfony 6.

I- Pourquoi utiliser un Template ?

Une application web n'est pas un bloc statique ; c'est un ensemble de diverses pages et interfaces (tableaux de bord, formulaires, profils, catalogues) qui sont en interaction constante avec l'utilisateur. Chaque clic déclenche une réponse visuelle et logique.Structuration optimale du contenu. Utiliser un template pré-construit offre trois avantages majeurs :

- Gain de temps massif : Vous ne réinventez pas la roue pour le design. Les composants (boutons, formulaires, barres de navigation) sont déjà prêts.
- Uniformité : Cela garantit une cohérence visuelle sur toutes vos pages sans effort supplémentaire.
- Réactivité (Responsive) : votre site s'adapte automatiquement aux mobiles, tablettes et ordinateurs.
- L'Efficacité du Développement : Au lieu de coder le CSS de chaque interface, vous utilisez des composants pré-faits (boutons, alertes, cartes)

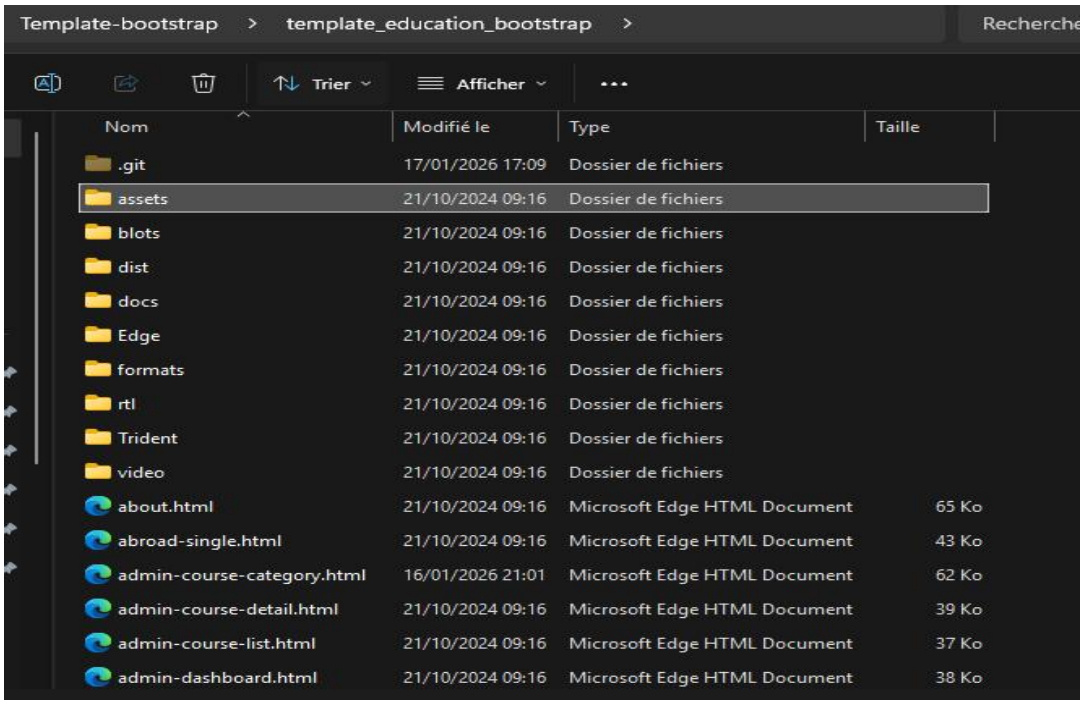
II- Structure générale d'un template web

Après avoir choisi un template depuis Internet (Bootstrap, Tailwind, etc.), on constate que la plupart des templates partagent une structure similaire, même si chaque template possède ses propres dépendances. On y retrouve généralement des fichiers CSS, des fichiers JavaScript, ainsi que des images, icônes et polices, qui sont le plus souvent regroupés dans un dossier assets. Ces fichiers sont responsables du design, de l'ergonomie et du comportement visuel du template.

Étapes clés

III- Étapes clés de l'intégration du template dans un projet Symfony

Le template en annexe contient un dossier assets, qui contient lui-même des répertoires css, img, js et vendor, ainsi que des fichiers HTML représentant des interfaces prêtes à être adaptées selon nos besoins, comme illustré dans la figure suivante.



Afin d'intégrer le Template dans notre projet, nous avons besoin de :

1. Copier les dossiers «**css**», «**img**», «**js**» et «**vendor**» sous le répertoire « **assets** ».
2. Choisir une page du template, par exemple index-11.html, et copier son contenu dans le fichier base.html.twig.
3. La page index-11.html étant un document HTML, elle contient des liens vers les fichiers CSS, JavaScript, fonts et images nécessaires au bon fonctionnement du template.
4. Pour permettre à Symfony d'accéder à ces dépendances, nous avons centralisé leur inclusion dans le fichier base.html.twig :
   - ✓ Les liens CSS et fonts présents dans la balise <head> sont placés dans le bloc {% block stylesheets %}.
   - ✓ Les scripts JavaScript sont copiés dans le bloc {% block javascripts %}.
5. Enfin, l'accès aux ressources est assuré grâce à la fonction asset() de Twig, qui permet de gérer correctement les chemins des fichiers, comme illustré dans la figure suivante.

```
    {# <link rel= icon  href= data:image/svg+xml,<svg xmlns=%22http://www.w3.org/2000/svg%22 viewBox=%220 0 128 128%22><text y=%221
    {% block stylesheets %}

<script>···
</script>

<!-- Favicon -->
<link rel="shortcut icon" href="assets/images/favicon.ico">

<!-- Google Font -->
<link rel="preconnect" href="https://fonts.googleapis.com/">
<link rel="preconnect" href="https://fonts.gstatic.com/" crossorigin>
<link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Heebo:wght@400;500;700&amp;family=Roboto:wght@400;500;700&amp

<!-- Plugins CSS -->
<link rel="stylesheet" type="text/css" href="{{asset('assets/vendor/font-awesome/css/all.min.css')}}">
<link rel="stylesheet" type="text/css" href="{{asset('assets/vendor/bootstrap-icons/bootstrap-icons.css')}}">
<link rel="stylesheet" type="text/css" href="{{asset('assets/vendor/tiny-slider/tiny-slider.css')}}">
<link rel="stylesheet" type="text/css" href="{{asset('assets/vendor/aos/aos.css')}}">

<!-- Theme CSS -->
<link rel="stylesheet" type="text/css" href="{{asset('assets/css/style.css')}}">

    {% endblock %}

    {% block javascripts %}
    <script src="{{asset('assets/vendor/bootstrap/dist/js/bootstrap.bundle.min.js')}}"></script>
    <script src="{{asset('assets/vendor/tiny-slider/tiny-slider.js')}}"></script>
    <script src="{{asset('assets/vendor/aos/aos.js')}}"></script>
```

6. Pour concevoir la page Home, nous avons créé un controller HomeController. Le dossier home ainsi que le fichier index.html.twig sont générés automatiquement par Symfony. Le fichier index.html.twig hérite ensuite du template principal base.html.twig afin de profiter de la structure et des dépendances du template

7. Enfin, le bloc body est redéfini en y copiant le contenu de la page index-11.html du template en annexe, permettant ainsi l'affichage de la page d'accueil dans le respect de l'architecture Symfony.

base.html.twig U | index.html.twig U

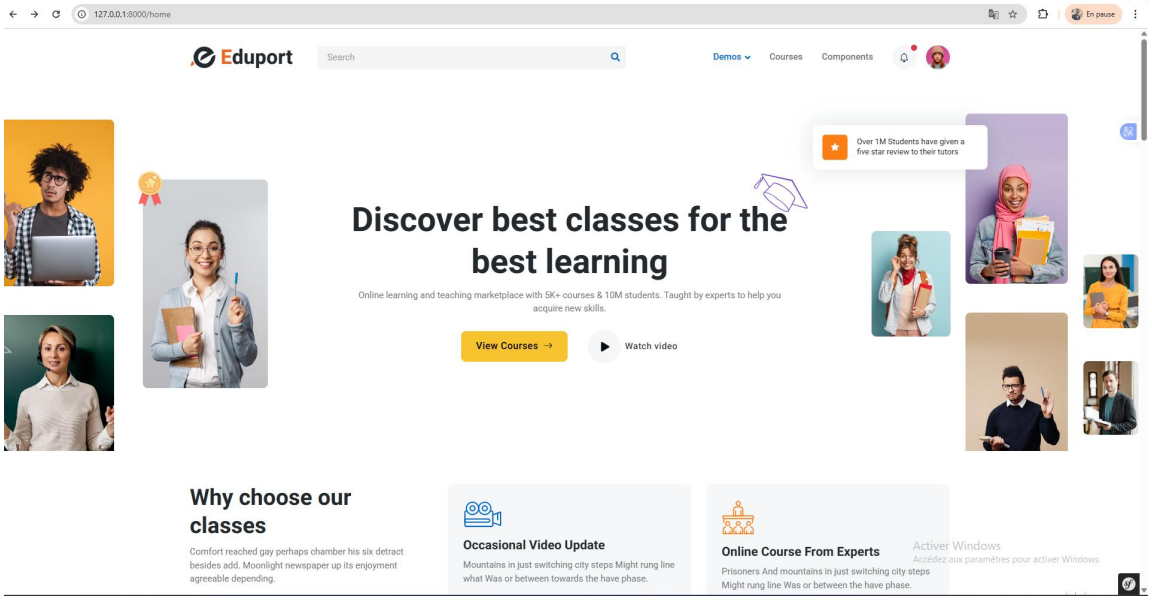
templates > home > index.html.twig

```
1   {% extends 'base.html.twig' %}
2
3   {% block body %}
4
5   <main>
6
7   <section class="overflow-hidden pt-6 pt-xxl-0">···
107 </section>
108 <!-- ========================
109 Main Banner END -->
110
111 <!-- ========================
112 Feature START -->
113 <section class="position-relative">
114
115     <!-- Svg image decoration -->
116     <div class="position-absolute bottom-0 end-0">
117         <img src="{{asset('assets/images/element/30.svg')}}" class="h-300px" alt="">
118     </div>
119
120     <div class="container">
121         <div class="row g-4">
122
123             <!-- Title -->
124             <div class="col-sm-6 col-lg-4">
125                 <h2>Why choose our classes</h2>
126                 <p class="mb-0">Comfort reached gay perhaps chamber his six detract besides add. Moonligl
127             </div>
128
129             <!-- Feature item -->
130             <div class="col-sm-6 col-lg-4">
131                 <div class="card card-body bg-light p-4 h-100">
```

Pour faire appel à nos images ou à nos fichiers CSS et JS on utilise {{asset(‘’)}}.
Exemple:

```
<div class="position-absolute bottom-0 end-0">
    <img src="{{asset('assets/images/element/30.svg')}}" class="h-300px" alt="">
</div>
```

8. Après avoir configuré la route dans le controller, il suffit de taper /home dans le navigateur.
   Selon la route définie, la page correspondante s’affiche et nous obtenons le résultat suivant, comme illustré ci-après.

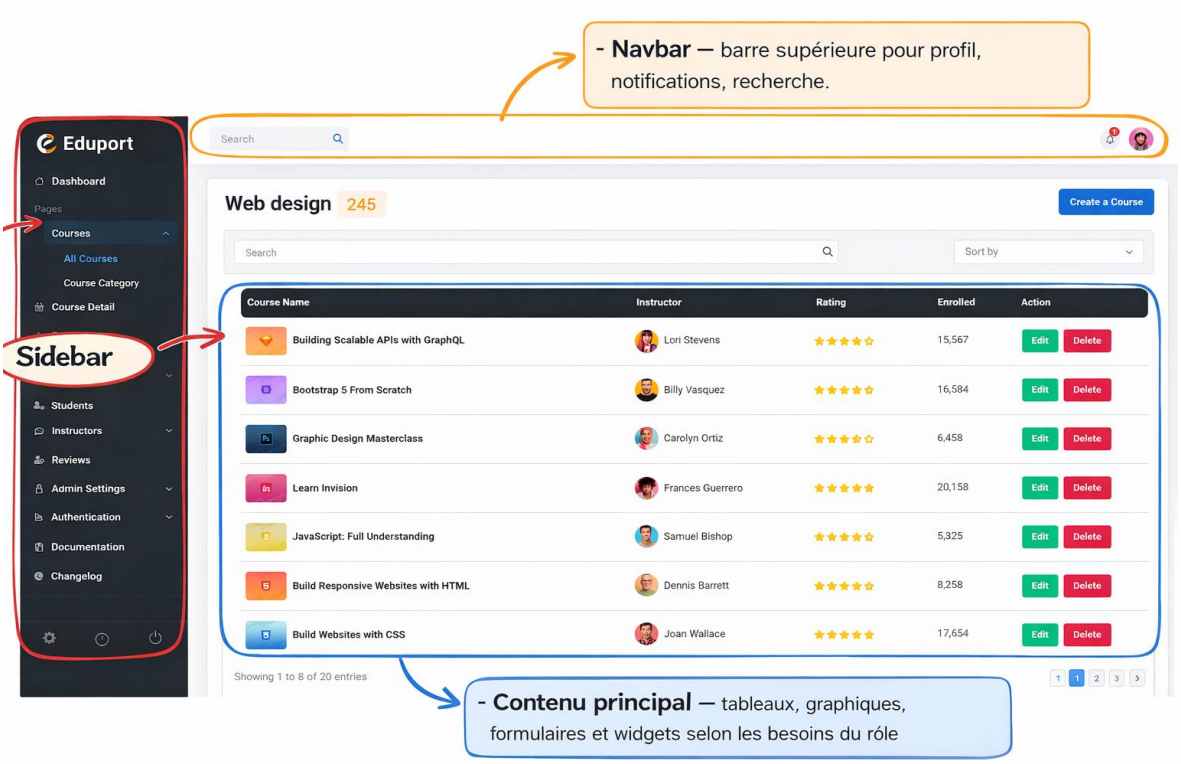


Remarque : à partir d’une nouvelle page, on peut redéfinir n’importe quel bloc défini dans le template principal pour personnaliser son contenu.
Exemple avec le bloc `title` :

```
{% block title %} esprit : meilleur école d'ingénieur{% endblock %}
```

IV- Réutilisabilité des composants : exemple d’intégration d’un dashboard

- Un dashboard est la page principale d’un espace personnel ou d’administration. Il présente les informations et actions importantes pour l’utilisateur connecté. En termes simples, c’est un tableau de bord personnalisé, similaire à celui d’une voiture, qui affiche ce qui est pertinent selon le rôle de l’utilisateur.
- Éléments communs d’un dashboard (voir image suivante) :
  - ✓ – Sidebar — menu latéral pour naviguer entre sections.
  - ✓ – Navbar — barre supérieure pour profil, notifications, recherche.
  - ✓ – Contenu principal — tableaux, graphiques, formulaires et widgets selon les be
  - ✓ soins du rôle.



➢ Étapes de création d'un template admin réutilisable

1. Générer un controller AdminController : Symfony crée automatiquement un dossier admin dans templates avec le fichier index.html.twig
2. Renommer ce fichier en base_admin.html.twig et mettre à jour la fonction index du controller pour utiliser ce nouveau nom.
3. Créer les composants partiels Dans templates/admin, créer un dossier Partials : Ajouter deux fichiers :
   - ✓ sidebar.html.twig
   - ✓ navbar.html.twig.
4. Intégrer les composants du template téléchargé : Copier uniquement le sidebar de votre template téléchargé et le coller dans sidebar.html.twig.
5. Copier et coller le navbar dans navbar.html.twig.
6. Hériter du template principal : Dans base_admin.html.twig, hériter de base.html.twig pour profiter de la structure HTML et des dépendances du template :
7. Redéfinir le bloc body en deux volets :
   A. Injecter les composants partiels :

```
{% include 'admin/partials/navbar-admin.html.twig' %}
{% include 'admin/partials/sidebar_admin.html.twig' %}
```

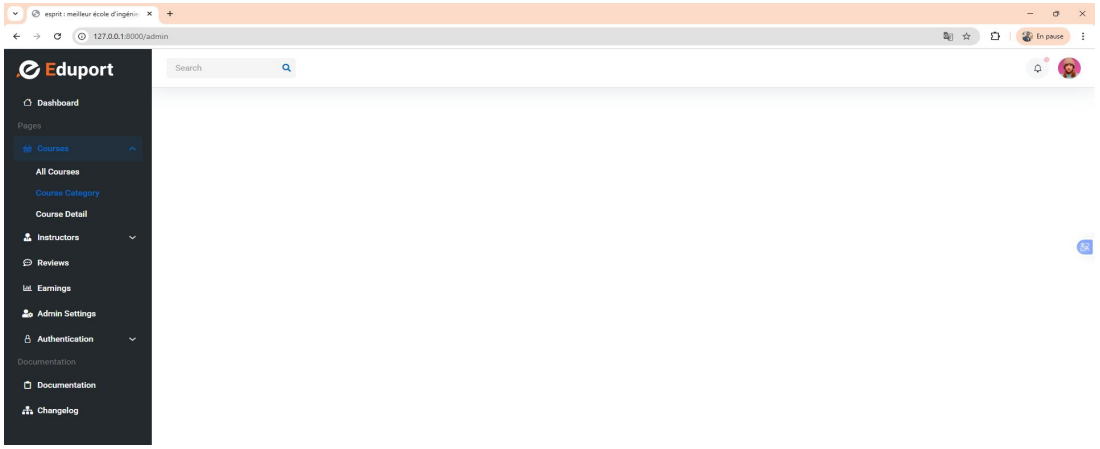
   B. Créer un bloc réutilisable pour le contenu admin : Ajouter un bloc admin_content dans base_admin.html.twig ,Ce bloc pourra être redéfini dans toutes les pages admin selon vos besoins, permettant une grande réutilisabilité et une structure cohérente pour tout l'espace d'administration.

base.html.twig U | base_admin.html.twig U | navbar-admin.html.twig U

templates > admin > base_admin.html.twig

```
{% extends 'base.html.twig' %}

{% block body %}
<div class="page-content">


    {% include 'admin/partials/navbar-admin.html.twig' %}
    {% include 'admin/partials/sidebar_admin.html.twig' %}

    {% block admin_content %}{% endblock %}
</div>
{% endblock %}

```

8. taper /admin dans votre navigateur.
   La page d'administration s'affiche avec le navbar et le sidebar, et le résultat obtenu est illustré dans la figure suivante.



9. Dans AdminController, créer une nouvelle fonction addUser avec la route

```
#[Route('/admin/add_user', name: 'app_admin_add_user')]
public function aaddUser(): Response
{
    return $this->render('admin/add_user.html.twig');
}
```

10. Créer le template add-user.html.twig : Dans le dossier templates/admin, créer add-user.html.twig et Hériter de base_admin.html.twig
11. Redéfinir le bloc admin_content et y écrire votre contenu (formulaire, liste, etc.)

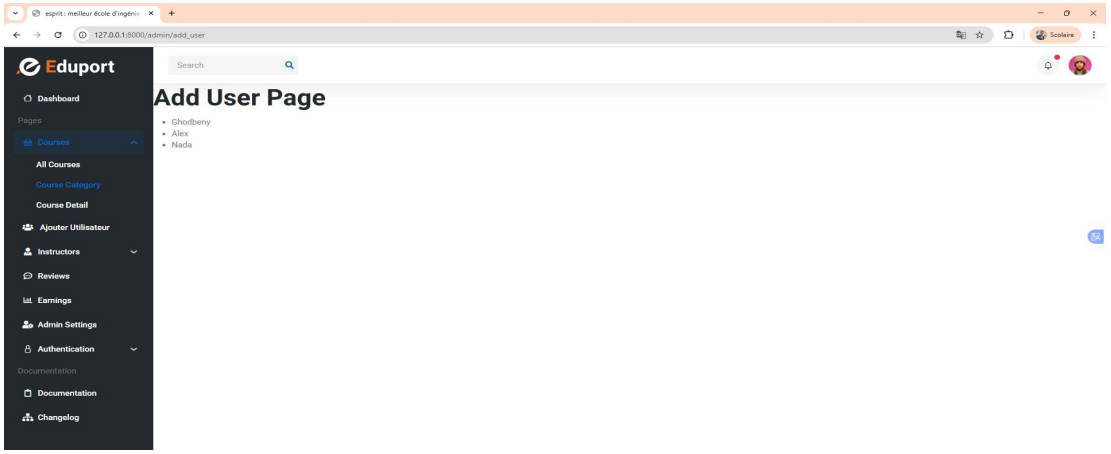
templates > admin > add_user.html.twig

```twig
{% extends 'admin/base_admin.html.twig' %}


{% block admin_content %}

<h1>Add User Page</h1>
<ul>
    <li>Ghodbeny </li>
    <li>Alex</li>
    <li>Nada</li>
</ul>


{% endblock %}
```

12. Mettre à jour le sidebar : Dans sidebar.html.twig, choisir un lien existant ou non configuré et le remplacer par :

```twig
<a class="nav-link" href="{{ path('app_admin_add_user') }}">
   <i class="fas fa-users fa-fw me-2"></i>
  Ajouter Utilisateur
</a>
```

13. Tester dans le navigateur
    - ✓ Aller sur /admin.
    - ✓ Cliquer sur le lien “Ajouter utilisateur” dans le sidebar.
      Le contenu du dashboard se met à jour dynamiquement et affiche la page add-user (formulaire, liste ou contenu que vous avez défini).



Remarque importante : on peut intégrer plusieurs templates en répétant les mêmes étapes, en plaçant leurs dépendances dans des dossiers séparés (ex. dependance_template2 sous assets) et en créant un fichier de base pour centraliser l’accès aux ressources.

Bon courage !