# Routage, Contrôleurs & Twig

# Mini-Exo 10 — Routage avancé

MODULE 2

### Objectif

Créer des routes propres et organisées pour votre application Symfony

## Routes à créer

| URL | Nom route | Fonction |
|---|---|---|
| /home | home | Accueil |
| /blog | blog_list | Liste articles |
| /blog/{id} | blog_show | Détail |
| /contact | contact | Contact |

### Routes avec attributs

Utilisez les annotations ou attributs PHP 8 pour définir vos routes

### Nom obligatoire

Chaque route doit avoir un nom unique pour faciliter la génération d'URL

### ID numérique

Contrainte : pour accepter uniquement des nombres

# Mini-Exo 11 — Contrôleurs structurés

MODULE 2

### Objectif

Organiser la logique métier de manière claire et maintenable

## Travail demandé

Vous allez créer trois contrôleurs distincts pour séparer les responsabilités de votre application :

- **HomeController** — Gestion de la page d'accueil
- **BlogController** — Logique du blog et des articles
- **ContactController** — Traitement du formulaire de contact

### 1 contrôleur = 1 domaine

Chaque contrôleur gère une zone fonctionnelle spécifique

### Méthodes courtes

Limitez la complexité : une action = une responsabilité

### Code lisible

Nommage explicite et commentaires si nécessaire

**Bonne pratique :** Placez vos contrôleurs dans `src/Controller/` et utilisez le suffixe `Controller` pour faciliter l'auto-wiring de Symfony.

# Mini-Exo 12 — Génération d'URL

MODULE 2

### Objectif

Éviter les liens en dur pour garantir la maintenabilité de votre application

## Étapes à suivre

01

### Utiliser path() dans Twig

Remplacez tous les liens HTML statiques par la fonction `path()`

02

### Utiliser generateUrl() en PHP

Dans vos contrôleurs, générez des URL dynamiques avec cette méthode

03

### Créer un menu dynamique

Construisez un menu de navigation réutilisable basé sur les noms de routes

> **Avantage clé :** Si vous modifiez l'URL d'une route, tous les liens se mettent à jour automatiquement sans toucher aux templates !

# Mini-Exo 13 — Layout global

MODULE 3

### Objectif

Centraliser l'interface pour éviter la duplication de code et faciliter la maintenance

### Header

Logo, navigation principale, zone utilisateur

### Menu

Navigation avec liens générés dynamiquement

### Zone contenu

Block Twig pour le contenu spécifique de chaque page

### Footer

Informations légales, liens secondaires, copyright

**Principe d'héritage :** Toutes vos pages doivent étendre `base.html.twig` avec `{% extends 'base.html.twig' %}` pour bénéficier automatiquement du layout global.

# Mini-Exo 14 — Liste dynamique

MODULE 3

### Objectif

Afficher des données depuis le contrôleur en utilisant Twig et PHP

**Point clé :** La séparation des responsabilités est respectée — le contrôleur prépare les données, le template les affiche.

# Mini-Exo 15 — Page détail article

MODULE 3

### Objectif

Relier une route dynamique à une vue détaillée pour afficher un article complet

## Configuration de la route

### Route dynamique

/blog/{id} avec contrainte {id<\d+>}

## Éléments à afficher

### Titre

Le titre complet de l'article avec une typographie mise en valeur

### Auteur

Nom de l'auteur avec avatar optionnel

### Contenu

Corps de l'article formaté et structuré

### Date

Date de publication formatée (ex: 15 mars 2024)

**Gestion d'erreur :** Si l'ID demandé n'existe pas dans votre tableau, lancez une exception `createNotFoundException()` pour renvoyer une erreur 404 propre.

# Mini-Exo 16 — Conditions Twig

MODULE 3

### Objectif

Utiliser les conditions `{% if %}` pour afficher du contenu dynamique selon le contexte

## Fonctionnalités à implémenter

- **Badge "Nouveau"**

  Afficher un badge si l'article date de moins de 7 jours

- **Message "Aucun article"**

  Gérer le cas où la liste d'articles est vide

- **Bouton admin**

  Afficher des actions d'édition si l'utilisateur est admin (simulation)

> **Astuce :** Combinez les conditions avec `{% elseif %}` et `{% else %}` pour créer des logiques d'affichage plus complexes.

# Mini-Exo 17 — Filtres Twig

MODULE 3

### Objectif

Formater les données directement dans les templates avec les filtres Twig

## Filtres à maîtriser

**upper**

Convertit le texte en majuscules

**date**

Formate les dates selon un pattern

**length**

Retourne la longueur d'une chaîne ou tableau

**slice**

Extrait une portion de texte

**Chaînage de filtres :** Vous pouvez combiner plusieurs filtres : `{{ title|lower|capitalize }}` applique d'abord `lower`, puis `capitalize`.

# Mini-Exo 18 — Composants réutilisables

MODULE 3

### Objectif

Découper l'interface en composants modulaires et réutilisables

## Composants à créer

### post_card.html.twig

Carte d'article avec titre, extrait et lien

### alert.html.twig

Message d'alerte configurable (succès, erreur, info)

---

**Principe DRY :** Don't Repeat Yourself — créez un composant une fois, réutilisez-le partout. Cela facilite la maintenance et garantit la cohérence visuelle.

> Bonus : Explorez aussi les `{% embed %}` pour des composants encore plus flexibles avec des blocks personnalisables.

# Mini-Exo 19 — Mini API interne

MODULE 4

### Objectif

Produire des réponses JSON pour créer une API REST simple

## Endpoints à créer

### /api/posts

Retourne la liste complète des articles au format JSON

### /api/posts/{id}

Retourne les détails d'un article spécifique par son ID

**Méthode recommandée :** Utilisez `$this->json()` dans vos contrôleurs pour créer automatiquement des `JsonResponse` avec les bons en-têtes HTTP.