DÉVELOPPEMENT BACKEND

Symfony 7 : initiation au framework PHP professionnel

Prendre en main Symfony 7 pour construire des applications web robustes et maintenables. Architecture MVC, Doctrine ORM, Twig, formulaires et sécurité : les fondamentaux du standard professionnel PHP.

DURÉE	TARIF HT	NIVEAU	LANGUE	GROUPE	FORMAT
4j (28h)	**1200.00 €**	**Intermédiaire**	**Français**	**3-12**	**Formation**

1 PUBLIC VISÉ

- Développeurs PHP maîtrisant les bases du langage et la programmation orientée objet souhaitant adopter un framework professionnel.
- Personnes ayant suivi les formations PHP moderne et PHP orienté objet ou disposant d'une expérience équivalente.
- Développeurs souhaitant structurer leurs projets PHP avec un framework enterprise utilisé par de nombreuses entreprises françaises.
- Personnes en reconversion visant un poste de développeur PHP back-end ou full-stack.

2 PRÉREQUIS

Maîtrise de PHP orienté objet : classes, héritage, interfaces, namespaces, Composer. Les formations PHP moderne et PHP orienté objet sont recommandées. Notions de SQL et de base de données relationnelle. Savoir utiliser le terminal et VS Code.

3 OBJECTIFS PÉDAGOGIQUES

- Comprendre l'architecture de Symfony 7 : composants, Flex, bundles et conventions
- Créer des routes et des contrôleurs pour gérer les requêtes HTTP
- Maîtriser Twig pour construire des templates HTML dynamiques et réutilisables
- Modéliser une base de données avec Doctrine ORM : entités, relations et migrations
- Construire des formulaires Symfony avec validation des données
- Mettre en place une authentification complète avec le Security Bundle
- Gérer les rôles et les permissions avec les voters
- Structurer et tester une application Symfony complète en conditions réelles

4 PROGRAMME DÉTAILLÉ

Jour 1 (7h) – Architecture Symfony et premiers composants

Module 1 – Découverte de Symfony 7 (2h)

Pourquoi Symfony ? (45min)

- Symfony dans l'écosystème PHP : histoire, adoption en entreprise, communauté
- Les composants Symfony : des briques réutilisables indépendantes du framework
- Symfony Flex : recettes, plugins et configuration automatique
- Symfony vs Laravel : positionnement et cas d'usage
- Ce que Symfony apporte concrètement par rapport à un MVC maison

Installation et structure du projet (1h15)

- Installer Symfony CLI et créer un projet avec symfony new
- Structure du projet : src/, config/, templates/, var/, public/
- Le kernel Symfony : point d'entrée et cycle de vie d'une requête
- Le fichier .env : variables d'environnement et configuration locale
- La console Symfony : les commandes essentielles bin/console
- Le Profiler et la Web Debug Toolbar : déboguer en développement
- Cas pratique : installer Symfony 7 et explorer la structure avec le Profiler

Module 2 – Routing et contrôleurs (2h30)

Le système de routing (1h)

- Déclarer une route avec les attributs PHP #[Route]
- Routes statiques, dynamiques et avec contraintes de type
- Routes nommées et génération d'URL avec le composant Router
- Routes avec méthodes HTTP : GET, POST, PUT, DELETE
- Lister et déboguer les routes avec bin/console debug:router
- Cas pratique : créer un ensemble de routes pour un blog

Les contrôleurs (1h30)

- Créer un contrôleur avec le Maker Bundle : make:controller
- La classe AbstractController et ses méthodes utilitaires
- L'objet Request : récupérer les données GET, POST et les headers
- L'objet Response : retourner du HTML, du JSON ou une redirection
- Les messages Flash : transmettre des notifications entre requêtes
- L'injection de services dans le contrôleur
- Cas pratique : contrôleur de blog avec liste, détail et redirection

Module 3 – Twig : moteur de templates (2h30)

Les bases de Twig (1h)

- La syntaxe Twig : {{ }}, {% %}, {# #}
- Variables, filtres et fonctions Twig
- Les structures de contrôle : if, for, set
- Générer des URLs depuis Twig avec path() et url()
- Inclure un template avec include

Héritage de templates (1h30)

- Le template de base : block et extends
- Surcharger un bloc dans un template enfant
- Les macros Twig : factoriser des fragments de template réutilisables
- Passer des variables globales à tous les templates avec Twig globals
- Cas pratique : layout principal avec header, footer et blocs surchargeables

Jour 2 (7h) – Doctrine ORM et base de données

Module 4 – Doctrine ORM : entités et migrations (3h)

Comprendre Doctrine ORM (1h)

- Le pattern Active Record vs Data Mapper : pourquoi Doctrine choisit Data Mapper
- L'Entity Manager : le chef d'orchestre de Doctrine
- Configurer la connexion à la base de données dans .env
- Créer une entité avec make:entity : propriétés et annotations
- Les types Doctrine : string, integer, text, boolean, datetime

Migrations et cycle de vie des entités (1h)

- Générer une migration avec make:migration
- Exécuter et annuler des migrations avec doctrine:migrations:migrate
- Les états d'une entité : new, managed, detached, removed
- persist(), flush() et remove() : écrire dans la base de données
- Cas pratique : entité Article avec migration et premiers enregistrements

Repositories et requêtes (1h)

- Le repository : l'objet dédié à la lecture des données
- Les méthodes magiques : findAll, findBy, findOneBy
- Le QueryBuilder Doctrine : construire des requêtes dynamiques
- DQL : le langage de requête orienté objet de Doctrine
- Cas pratique : repository Article avec filtres et tri personnalisés

Module 5 – Relations entre entités (2h)

Les types de relations Doctrine (1h30)

- ManyToOne et OneToMany : relation article/catégorie
- ManyToMany : relation article/tag avec table de jointure
- OneToOne : relation utilisateur/profil
- Le côté propriétaire et le côté inverse d'une relation
- Cascade : persist et remove sur les relations
- Cas pratique : entités Article, Categorie et Tag avec leurs relations

Chargement des relations (30min)

- Lazy loading vs eager loading : le problème des N+1 requêtes
- Optimiser avec les jointures dans le QueryBuilder
- Cas pratique : afficher les articles avec leurs catégories en une seule requête

Module 6 – Formulaires Symfony (2h)

Créer et afficher un formulaire (1h)

Créer un FormType avec make:form

- Les champs Symfony : TextType, EmailType, ChoiceType, EntityType, DateType
- Options des champs : label, required, attr, mapped
- Afficher un formulaire dans Twig : form_start, form_row, form_end
- Personnaliser le rendu d'un formulaire avec form_widget et form_errors

Traitement et validation (1h)

- Traiter la soumission avec handleRequest et isSubmitted/isValid
- Les contraintes de validation : NotBlank, Email, Length, Regex, UniqueEntity
- Afficher les erreurs de validation dans le template
- Cas pratique : formulaire de création et d'édition d'article avec validation

Jour 3 (7h) – Sécurité et services

Module 7 – Security Bundle : authentification et autorisation (4h)

Mise en place de l'authentification (2h)

- Le Security Bundle : vue d'ensemble et configuration dans security.yaml
- Créer l'entité User avec make:user : UserInterface et PasswordAuthenticatedUserInterface
- Générer le système de connexion avec make:security:form-login
- Hacher les mots de passe avec PasswordHasher
- La page de connexion : LoginFormAuthenticator et le formulaire
- La déconnexion et la redirection après connexion
- Cas pratique : système de connexion complet avec redirection selon le rôle

Inscription et gestion des rôles (1h)

- Créer un formulaire d'inscription avec make:registration-form
- Les rôles Symfony : ROLE_USER, ROLE_ADMIN et hiérarchie des rôles
- Restreindre l'accès par route dans security.yaml avec access_control
- Restreindre l'accès dans un contrôleur avec #[IsGranted] et denyAccessUnlessGranted
- Afficher du contenu conditionnel dans Twig avec is_granted

Voters : contrôle d'accès avancé (1h)

- Pourquoi les voters : gérer des permissions métier complexes
- Créer un voter avec make:voter
- Appliquer un voter depuis un contrôleur ou un template Twig
- Cas pratique : voter pour autoriser uniquement l'auteur à modifier son article

Module 8 – Services et injection de dépendances (3h)

Le conteneur de services Symfony (1h30)

- Qu'est-ce qu'un service ? Le principe de l'inversion de contrôle
- L'autowiring Symfony : injection automatique par type-hinting
- Créer un service métier et l'injecter dans un contrôleur
- Les services scalaires : injecter des paramètres de configuration
- Lister les services disponibles avec bin/console debug:autowiring
- Cas pratique : service de gestion des slugs pour les articles

Les événements et listeners Symfony (1h30)

Le composant EventDispatcher : émettre et écouter des événements

- Créer un EventListener et un EventSubscriber
- Les événements du kernel : KernelEvents::REQUEST, RESPONSE, EXCEPTION
- Les événements Doctrine : PrePersist, PostUpdate pour automatiser des traitements
- Cas pratique : listener qui génère automatiquement le slug avant la persistance

Jour 4 (7h) – Fonctionnalités avancées et projet de synthèse

Module 9 – Fonctionnalités avancées (3h)

Envoi d'emails avec Mailer (1h)

- Configurer le composant Mailer dans .env
- Créer et envoyer un email avec TemplatedEmail et un template Twig
- Tester les emails en développement avec Mailtrap ou le transport null
- Cas pratique : email de confirmation d'inscription

Commandes console personnalisées (1h)

- Créer une commande avec make:command
- Définir des arguments et des options
- Afficher de la progression avec ProgressBar
- Cas pratique : commande de purge des articles expirés

Pagination et upload de fichiers (1h)

- Paginer des résultats avec KnpPaginatorBundle
- Gérer l'upload d'un fichier dans un formulaire Symfony
- Valider et déplacer le fichier uploadé
- Cas pratique : liste d'articles paginée et upload d'image de couverture

Module 10 – Projet de synthèse (4h)

Réalisation d'une application Symfony 7 complète

- Cahier des charges : plateforme de blog avec espace d'administration
- Fonctionnalités : authentification complète avec inscription et connexion, gestion des rôles ROLE_USER et ROLE_ADMIN, CRUD articles avec catégories et tags, voter pour les permissions d'édition, formulaires avec validation, upload d'image de couverture, pagination de la liste des articles, email de confirmation d'inscription, interface d'administration réservée aux admins
- Étape 1 : modélisation des entités et migrations
- Étape 2 : authentification et gestion des rôles
- Étape 3 : CRUD articles avec formulaires et upload
- Étape 4 : voter, pagination et emails
- Revue de code collective : architecture, sécurité, conventions Symfony
- Retour formateur individualisé sur le projet rendu

5 COMPÉTENCES VISÉES

- Développer une application Symfony 7 complète en respectant les conventions du framework
- Modéliser et interroger une base de données relationnelle avec Doctrine ORM
- Construire des interfaces HTML dynamiques avec Twig
- Mettre en place un système d'authentification et de gestion des rôles
- Valider et traiter des formulaires complexes côté serveur
- Utiliser la console Symfony et les outils de débogage (Profiler, Maker Bundle)

6 MODALITÉS PÉDAGOGIQUES

Formation délivrée en présentiel ou distanciel (visioconférence). Le formateur alterne entre méthode démonstrative (live coding commenté et progressif sur une application construite tout au long de la formation), méthode interrogative (analyse des conventions Symfony et discussion des choix d'architecture) et méthode active (exercices pratiques et mini-projets par module). L'accent est mis sur la compréhension des mécanismes internes de Symfony plutôt que sur la mémorisation des commandes.

7 MOYENS PÉDAGOGIQUES ET TECHNIQUES

- Support de cours numérique mis à disposition des apprenants
- Dépôt GitHub de démonstration avec exercices et corrections par module
- Environnement de développement : VS Code + Symfony CLI + Docker ou Laragon
- Pour le distanciel : visioconférence (Zoom ou équivalent), partage d'écran, chat en direct
- Accès à la plateforme pédagogique LaPolaris (supports, ressources, émargement)

8 MODALITÉS D'ÉVALUATION

- En cours de formation : exercices pratiques et mini-projets corrigés à chaque module
- En fin de formation : réalisation d'une application Symfony 7 complète avec authentification, Doctrine ORM et gestion des rôles
- Questionnaire d'auto-évaluation des acquis en fin de parcours

9 CRITÈRES D'ÉVALUATION

- Respect des conventions Symfony : structure du projet, nommage, utilisation du Maker Bundle
- Modélisation correcte des entités Doctrine avec leurs relations et migrations
- Implémentation fonctionnelle de l'authentification et de la gestion des rôles
- Formulaires valides avec contraintes de validation adaptées aux données métier
- Qualité et lisibilité du code : séparation des responsabilités, injection de dépendances, absence de logique métier dans les contrôleurs

10 MODALITÉS DE VALIDATION

Attestation de fin de formation délivrée à l'issue du parcours, conditionnée à une assiduité d'au moins 80 % et à la réalisation du projet de synthèse. L'attestation précise les objectifs atteints et les compétences acquises.

11 SUIVI ET ACCOMPAGNEMENT

- Feuilles d'émargement signées par demi-journée (présentiel) ou émargement numérique (distanciel)
- Traçabilité des activités pédagogiques réalisées
- Attestation d'assiduité délivrée en fin de formation
- Suivi individuel via les exercices corrigés et le projet de synthèse

12 CONDITIONS D'ACCÈS

Formation accessible sur inscription directe, sans prérequis administratif particulier. Le financement peut être pris en charge par l'employeur dans le cadre d'un plan de développement des compétences, ou en autofinancement avec possibilité de paiement en plusieurs fois.

13 DÉLAIS D'ACCÈS

Inscription possible jusqu'à **5 jours ouvrés** avant le début de la session. Pour toute demande urgente, nous contacter directement.

ACCESSIBILITÉ · HANDICAP

Nos formations sont accessibles aux personnes en situation de handicap. Pour toute situation nécessitant un aménagement (matériel, temporel ou pédagogique), nous vous invitons à nous contacter avant l'inscription afin d'étudier les adaptations possibles. Référent handicap : contact@lapolaris.fr

LaPolaris

TÉL. +33762584798

EMAIL contact@lapolaris.fr

WEB lapolaris.fr