Évaluation en cours de formation

Développeur Web et Web Mobile

Prénom : Michel
Nom : Almont

ATTENTION ! PENSEZ À RENSEIGNER VOS NOM ET PRÉNOM DANS LE TITRE DE VOS FICHIERS / PROJETS !

Nom du projet : SPORT-TRAINING

Lien Github du projet https://github.com/NOA-FASHION/sport_training
Documentation technique : https://github.com/NOA-FASHION/sport_training/tree/main/Documentation
Lien du site : https://backend-strapi.online/sport-training/
Attention ! Merci de bien classer vos documents dans votre Github ou votre drive.
URL du site (si vous avez mis votre projet en ligne) : ……

Description du projet

1. Liste des compétences du référentiel qui sont couvertes par le projet

- Maquetter une application.
- Développer une interface utilisateur web dynamique.
- Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce.
- Créer une base de données.
- Développer les composants d'accès aux données.
- Développer la partie back-end d'une application web ou web mobile.
- Élaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce.

2. Résumé du projet en français d'une longueur d'environ 20 lignes soit 200 à 250 mots, ou environ 1200 caractères espaces non compris

PROJET SPORT-TRAINING

Ce projet a été réalisé Dans le cadre de ma formation à l'école STUDI, Développeur Web – Web Mobile qui a débuté le 1 février 2022.
L'application à développer est destinée à une grande marque de salle sport, qui propose déjà à ces partenaires franchisés, plusieurs services ayant pour but de faciliter la gestion d'une salle de sport au quotidien.
Ces services sont disponibles sur une plateforme déjà existante.
Voici quelques exemples de prestations disponibles :

- Faire son mailing
- Gérer le planning équipe
- Promotion de la salle
- Vendre des boissons

Chacune de ces activités peuvent être considérées comme des modules activables ou désactivables à volonté.
Ma mission est de créer un gestionnaire d'accès à ces modules disponibles sur une application tierce.
Un backend qui permet d'activer ou désactiver les services selon le package que le partenaire aura contracté.
L'application devra être intuitive, et aller l'essentiel.
Elle donnera la possibilité au staff de désactiver un partenaire par exemple qui n'aura pas renouveler son abonnement annuel ou encore de modifier son niveau de service.
Le développement de cette application m'a pris environ un mois.
C'est le duo Symfony/Postgress qui j'ai choisi pour la partie backend et Bootstrap et Twig pour la partie frontend.
Ce projet a été réaliser en total autonomie, J'ai pu acquérir durant cette expérience de nouvelle compétence, notamment sur la partie backend.

Ayant débuter mon apprentissage autodidacte dans le monde du développement mobile, J'ai été également agréablement surpris de la modernité du Framework Symfony.

3. Cahier des charges, expression des besoins, ou spécifications fonctionnelles du projet

Fonctionnalités de l'application

Gestion des partenaires
Utilisateurs concernés : Équipe Tech
Ajout de partenaire franchisé.
Association et création d'un compte partenaire par partenaire
Activation/désactivation des partenaires

Gestion des structures
Utilisateurs concernés : Équipe Tech
Ajout de structures par partenaire
Association et création d'un compte structure par structure
Activation/ désactivation des structures

Gestion des accès
Utilisateurs concernés : Équipe Tech le partenaire, la structure
Accès en lecture/écriture permission : Équipe tech
Accès en lecture permission : Partenaire et structure

Se connecter
Utilisateurs concernés : Équipe Tech le partenaire, la structure
Email envoyer lors de la création de structures ou partenaires
Information d'authentification envoyer par email aux gérants de structures et aux partenaires.
Email envoyer lors des modifications de permissions.
Accès des gérants de structures et des partenaires à la plateforme après authentification en lecture seul.
Accès après authentification en lecture/écriture aux admins à la plateforme.

Confirmation de sécurité
Utilisateurs concernés : Équipe Tech
A chaque modification ou suppression, affichage d'un message de confirmation pour valider l'action.

Recherche dynamique
Utilisateurs concernés : Équipe Tech le partenaire, la structure
L'application proposera une barre de recherche pour filtrer les recherches.
Le premier filtre sera disponible en tapant les premières lettres du partenaire ou de la structure.
Le deuxième sera paramétré par rapport aux éléments actifs ou non.

4. Spécifications techniques du projet, élaborées par le candidat, y compris pour la sécurité et le web mobile

Spécifications techniques du projet :

Maquette/Illustration/Logo/Image

Pour la partie maquettage j'ai utilisé **Figma**.

Diagrams.net est l'outil que j'ai utilisé pour réaliser le diagramme de classe ou, diagramme de cas d'utilisation et le diagramme de séquence

Canva.com est l'outils que j'ai utilisé pour la documentation.

Environnement de travail

L'environnement de développement tournait sous **MacOs**

Pour consulter, modifier et tester la base de données, j'ai utilisé l'IDE **Datagrip.**

Vscode est mon éditeur de texte préféré.

Technologie front-end

Pour coder rapidement une interface responsive et efficace j'ai choisi l'outils **Bootstrap** couplé au thème **Bootswatch**.

Le Template **Twig** est le moteur de gabarit que j'ai utilisé pour la gestion des vues.
Le moteur de Template open source **Twig** facilite le développement, la sécurisation et la maintenance d'applications web PHP.
JavaScript est un élément indispensable pour le bon fonctionnement de Bootstrap.

Technologie Back-end

Base de données

Le système de gestion de base de données est **POSTGRES.**
Ce système multi-plateforme est largement connu et réputé à travers le monde, notamment pour son comportement stable et pour être très respectueux des normes **ANSI SQL.**

Symfony

Le Framework PHP Symfony est le partenaire que j'ai choisi pour interfacer le SGBD POSTGRESS.

Pourquoi Symfony ?
Il contient tout ce dont a besoin pour créer un site web professionnel et sécurisé :

- Un moteur de gabarit
- Un ORM
- Un client de test

Symfony est un Framework qu'on pourrait qualifier de modulaire.
Pour utiliser toute la puissance de ce Framework il faut installer et activer plusieurs modules indispensables pour réaliser un site web sécurisé et professionnel.
Les composants qui ont été installés et activer dans ce projet sont décrit succinctement dans les prochaines lignes.

Doctrine

Doctrine est un ORM de Symfony
Un ORM est un ensemble de classes permettant de manipuler les tables d'une base de données relationnelle comme s'il s'agissait d'objets.

En base de données chaque élément ou objet d'une application est représenté par une table.
En programmation ces éléments sont des objets.

Un ORM est une couche d'abstraction d'accès à la base de données qui donne l'illusion de ne plus travailler avec des requêtes mais de manipuler des objets.

Mail

Symfony possède son propre composant nous permettant d'envoyer des e-mails depuis la version 4.3. C'est le composant Mailer.

L'installation du composant Mailer se fait via composer.

Forms

Les formulaires sont des éléments indispensables pour récupérer les informations de nos objets.
Symfony propose un système puissant et flexible qui permet d'unifier et de simplifier la génération et le traitement de formulaires.
A lui seul, il gère les éléments suivants :

- Rendu du formulaire dans la page
- Gestion de l'envoi du formulaire
- Validation des données
- Normalisation des données
- Protection CSRF

L'installation de ce module se fait par l'intermédiaire de composer.

Github

Pour la gestion du travail collaboratif, du versionning et de l'hébergement des dépôts du projet le tandem GIT/GITHUB a été utilisé.

Déploiement

Le déploiement a été effectué sur un serveur VPS Ubuntu.
Le serveur web utilisé est l'outils NGINX

Sécurité

SecurityBundle

Authentification avec le SecurityBundle de Symfony

Le contrôle de la sécurité sous Symfony est très avancé mais également très simple. Pour cela Symfony distingue :

- L'authentification,
- L'autorisation.

L'authentification, c'est le procédé qui permet de déterminer qui est votre visiteur. Il y a deux cas possibles :

- Le visiteur est anonyme car il ne s'est pas identifié,
- Le visiteur est membre de votre site car s'est identifié

Sous Symfony, c'est le firewall qui prend en charge l'authentification.

Régler les paramètres du firewall va vous permettre de sécuriser le site. En effet, vous pouvez restreindre l'accès à certaines parties du site uniquement aux visiteurs qui sont membres. Autrement dit, il faudra que le visiteur soit authentifié pour que le firewall l'autorise à passer.

L'autorisation intervient après l'authentification. Comme son nom l'indique, c'est la procédure qui va accorder les droits d'accès à un contenu. Sous Symfony, c'est l'Access control qui prend en charge l'autorisation.

Prenons l'exemple de différentes catégories de membres. Tous les visiteurs authentifiés ont le droit de poster des messages sur le forum mais uniquement les membres administrateurs ont des droits de modération et peuvent les supprimer. C'est l'accès control qui permet de faire cela.

Sécurisation de l'environnement

Concernant la sécurité du serveur, j'ai installé le tandem Fail2ban le firewall UFW pour filtrer les paquets et limiter l'ouverture des ports du serveur.

Un pare-feu est essentiel lors de la configuration du VPS pour limiter le trafic indésirable sortant ou entrant dans votre VPS.

L'outil fail2ban permet de surveiller l'activité des logs de certains services, tel que SSH ou Apache. Lors d'un trop grand nombre d'authentifications ratées fail2ban va générer une règle IPTables, cette règle aura pour but d'interdire pendant une durée déterminée les connexions depuis l'adresse IP susceptible d'être un attaquant.

5. Description de la veille, effectuée par le candidat durant le projet, sur les vulnérabilités de sécurité

La sécurité d'un site Web

La sécurité d'un site web exige de la vigilance dans tous les aspects de sa conception et de son utilisation. Cet article d'introduction ne fera pas de vous un gourou de la sécurité des sites web, mais il vous aidera à comprendre d'où viennent les menaces et ce que vous pouvez faire pour renforcer votre application web contre les attaques les plus courantes.

Pré-requis :
Connaissances de base en informatique.

Objectif :
Comprendre les menaces les plus courantes à la sécurité des applications web et ce que vous pouvez faire pour réduire le risque de piratage de votre site.

Qu'est-ce que la sécurité d'un site web?

Internet est un endroit dangereux ! Fréquemment, nous entendons parler de sites web devenus indisponibles en raison d'attaques par déni de service, ou affichant des informations modifiées (et souvent préjudiciables) sur leurs pages d'accueil. Dans d'autres cas, très médiatisés, des millions de mots de passe, d'adresses électroniques et de détails sur des cartes de crédit sont divulgués au public, exposant les utilisateurs du site web à la fois à l'embarras personnel et au risque de pertes financières.

L'objectif de la sécurité des sites web est de prévenir ces types d'attaques. Plus formellement, la sécurité des sites web est l'acte de protéger les sites web contre l'accès, l'utilisation, la modification, la destruction ou la perturbation non autorisées.

La sécurité efficace d'un site web nécessite un effort de conception sur l'ensemble du site : dans votre application web, dans la configuration du serveur web, dans vos politiques de création et de renouvellement des mots de passe et dans le code côté-client. Bien que tout cela semble très inquiétant, la bonne nouvelle est que si vous utilisez un framework web côté serveur, il incluera certainement par défaut des mécanismes de défense solides et bien pensés contre un certain nombre des attaques les plus courantes. D'autres attaques peuvent être atténuées grâce à la configuration de votre serveur web, par exemple en activant HTTPS. Enfin, les outils d'analyse de vulnérabilité accessibles au public peuvent vous aider à découvrir d'éventuelles erreurs dans votre conception.

Le reste de cet article détaille les menaces les plus courantes qui pèsent sur les sites web et quelques étapes simples pour protéger votre site.

Menaces visant la sécurité des sites web

Cette section n'énumère que quelques-unes des menaces les plus courantes visant les sites web et la façon dont elles sont tenues à distance. À mesure que vous lisez, notez comment les menaces apparaissent lorsque l'application web fait confiance ou n'est pas assez méfiante vis-à-vis des données provenant du navigateur !

Cross-Site Scripting (XSS)

XSS est un terme utilisé pour décrire une classe d'attaque qui permet à l'attaquant d'injecter des scripts, exécutés côté-client, au travers du site web pour viser le navigateur web des autres utilisateurs. Comme le code injecté provient du site web le navigateur web le considère comme sûr, il peut de ce fait faire des choses comme transmettre le cookie d'authentification de l'utilisateur à l'attaquant. Une fois que l'attaquant obtient ce cookie il peut se connecter sur le site comme si il était l'utilisateur attaqué et peut faire tout ce que l'utilisateur pourrait faire. En fonction du site sur lequel l'attaque se produit, cela peut inclure l'accès aux détails de carte bancaire, les informations des contacts, la modification du mot de passe, etc.
La meilleur défense contre les vulnérabilités XSS est de supprimer ou désactiver toutes les balises qui peuvent potentiellement contenir des instructions pour exécuter du code. Pour HTML cela inclut les tags comme <script>, <object>, <embed>, et <link>.

Il est nécessaire de traiter les données saisies par l'utilisateur pour être sûr qu'il ne puisse ni exécuter de scripts ni pertuber le fonctionnement normal du site (ce procédé est appelé input sanitization en anglais). De nombreux frameworks proposent par défaut cette vérification sur les entrées des formulaires.

Injection SQL

L'injection SQL est une vulnérabilité qui permet à un attaquant d'exécuter du code SQL frauduleux sur une base de données, permettant l'accès, la modification ou la suppression des données quelque soit le droit de l'utilisateur. Une attaque par injection réussie peut permettre l'usurpation d'un compte, la création d'un compte avec les droits administrateur, l'accès à toutes les données du serveur, ou la modification / destruction des données pour le rendre inutilisable.

Cette vulnérabilité est présente quand la saisie de l'utilisateur est transmise à une requête SQL sous-jacente qui peut modifier le sens de la requête. Par exemple, dans le code ci-dessous qui devrait lister l'ensemble des utilisateurs avec un nom particulier (userName) et qui provient d'un formulaire HTML:

Le moyen pour éviter ce type d'attaque est de s'assurer que toute saisie de l'utilisateur transmise à une requête SQL ne peut pas changer la nature de cette requête. Un moyen de faire cela est d'échapper tous les caractères de la saisie utilisateur quand ils ont un sens particulier en SQL.

Falsification de requête inter-sites (CSRF)

Les attaques CSRF permettent à un utilisateur malveillant d'éxécuter des actions à l'aide des identifiants d'un autre utilisateur sans que cet utilisateur ne soit informé ou consentant.

Ce type d'attaque s'explique mieux avec un exemple. John est l'utilisateur malveillant qui sait qu'un site particulier permet à des utilisateurs authentifiés d'envoyer de l'argent vers un compte particulier en utilisant des requêtes HTTP POST qui incluent le numéro de compte et le montant. John construit un formulaire qui inclut son numéro de compte et un montant dans des champs cachés (invisibles) et le transmet à un autre utilisateur du site (avec le bouton de validation déguisé en un lien vers un site "pour devenir riche".

Si un utilisateur clique sur le bouton de validation, une requête HTTP POST, contenant les informations de transaction, va être transmise au serveur ainsi que le cookie que le navigateur web associe au site (l'ajout à la requête du cookie associé au site est le comportement normal du navigateur). Le serveur va vérifier le cookie d'authentification, et l'utiliser pour déterminer si l'utilisateur est ou n'est pas connecté et donc permet ou non la transaction.

Au final tout utilisateur qui va cliquer sur le bouton de validation, alors qu'il sera connecté sur le site d'échange d'argent, va autoriser la transaction. John va devenir riche !

Un moyen de prévenir ce type d'attaque est que le serveur demande que chaque requête POST possède un secret généré par le serveur et spécifique à l'utilisateur (le secret serait transmis par le serveur lors de l'envoi du formulaire de transaction). Cette approche empêche John de créer son propre formulaire car il n'est pas capable de connaitre le secret que le serveur founit à l'utilisateur. Même si il venait à trouver ce secret et créer un formulaire pour un utilisateur particulier, il ne pourrait pas utiliser ce formulaire pour attaquer d'autres utilisateurs .

Autre menaces
Les autres attaques et vulnérabilités courantes incluent:

- Clickjacking. Dans cette attaque un utilisateur malveillant détourne les clics destinés à un site visible et les envoie dans une page cachée en dessous. Cette technique peut être utilisée, par exemple, pour afficher le site légitime d'une banque mais capturer les identifiants d'authentification dans une <iframe> invisible controlée par l'attaquant. Sinon cela peut être utilisé pour obtenir de l'utilisateur qu'il clique sur le bouton visible d'un site, mais en le faisant il va en fait cliquer involontairement sur un bouton différent. Comme défense, votre site peut se prévenir d'être inclut dans une iframe d'un autre site en configurant les bonnes en-têtes HTTP.

- Déni de Service (DoS). Le déni de service est souvent pratiqué en surchargeant de fausses requêtes un site cible avec afin que l'accès au site soit perturbé pour les usagers légitimes. Les requêtes peuvent simplement être nombreuses, ou elles peuvent individuellement nécessiter une grande

quantité de ressource (ex: chargement de fichiers lourds, etc). La défense contre cette attaque se base souvent sur l'identification et le blocage du mauvais trafic tout en autorisant l'arrivée des messages légitimes. Ces défenses sont généralement intégrées ou en amont du serveur web (elle ne font pas partie de l'application web).

- Découverte via la navigation dans les répertoires et fichiers. Dans cette famille d'attaque un utilisateur malveillant va tenter d'accèder aux fichiers du serveur web auxquels il ne devrait pas avoir accès. Cette vulnérabilité se produit quand l'utilisateur peut transmettre le nom d'un fichier qui comprend les caractères de navigation dans le système de fichier (par exemple: ../../). La solution est de désinfecter la saisie avant de l'utiliser.

- Inclusion de fichiers. Dans cette attaque un utilisateur est capable de spécifier un fichier "involontaire" pour être affiché ou exécuté dans les données transmises au serveur. Une fois chargé ce fichier peut être exécuté sur le serveur web ou du côté client (menant à une attaque XSS). La solution est de vérifier les saisies avant de les utiliser.

- Injection de commandes. Les attaques par injection de commande permettent à un utilisateur malveillant d'exécuter des commandes systèmes frauduleuses sur le système hôte. La solution est de vérifier chaque saisie des utilisateurs avant de les utiliser dans les appels systèmes.

- Il y en a beaucoup d'autres. Pour une liste plus exhaustive vous pouvez consulter la catégorie failles de sécurité Web (Wikipedia) et la catégorie Attaques (du projet OWASP).

Quelques messages clés

La majorité des attaques citées précédement réusissent lorsque l'application web fait confiance aux données provenant du navigateur web. Quoique vous fassiez d'autre pour améliorer la sécurité de votre site web, vous devez désinfecter toutes les saisies des utilisateurs avant de les afficher, de les utiliser dans les requêtes SQL ou de les transmettre dans les appels du système ou du système de fichier.

Quelques autres points que vous pouvez mettre en place :

- Utilisez une politique de gestion des mots de passe efficace. Encouragez les mots de passe solides avec leur renouvellement fréquent. Prenez en compte une authentification à deux facteurs sur votre site (l'utilisateur, en plus du mot de passe, devra fournir un autre code d'authentification généralement fourni par un matériel physique, que seul l'utilisateur possède, comme un code envoyé sur le téléphone par SMS).

- Configurez vos serveurs web pour utiliser HTTPS et HTTP Strict Transport Security (en-US) (HSTS). HTTPS chiffre les données transmises entre le client et le serveur. Cela garantit que les données d'authentification, les cookies, les données transistant avec POST et les informations d'en-têtes deviennent moins disponibles pour l'attaquant.

- Tenez vous informé des dernières menaces (la liste actuelle d'OWASP est ici) et préoccupez vous toujours des vulnerabilités les plus courantes en premier.

- Utilisez des outils de recherche de vulnérabilités pour faire automatiquement des recherches de bug sur votre site (votre site pourra également proposer un programme de buf bounty pour déceler des bugs, comme le fait Mozilla ici).

- Ne stockez et n'affichez que les données nécessaires. Par exemple, si votre utilisateur doit stocker des données sensibles comme des informations bancaires, affichez seulement ce qui sera suffisant pour être identifié par l'utilisateur, mais pas suffisament pour être copié par un attaquant et être utilisé sur un autre site. La méthode la plus courante en ce moment est de n'afficher que les quatre derniers chiffres du numéro de carte bancaire.
- Les frameworks web peuvent aider à atténuer beaucoup des vulnérabilités les plus courantes.

6. Description d'une situation de travail ayant nécessité une recherche, effectuée par le candidat durant le projet, à partir de site anglophone

Pour développer la fonctionnalité de l'envoi de mail, J'ai eu recours à la documentation officiel de Symfony qui se trouve à l'adresse suivante :
https://symfony.com/doc/current/mailer.html

7. Extrait du site anglophone, utilisé dans le cadre de la recherche décrite précédemment, accompagné de la traduction en français effectuée par le candidat sans traducteur automatique (environ 750 signes).

Installation
Symfony's Mailer & Mime components form a powerful system for creating and sending emails - complete with support for multipart messages, Twig integration, CSS inlining, file attachments and a lot more. Get them installed with:

```
$ composer require symfony/mailer
```

Traduction

Installation
Les composants Mailer & Mime de Symfony forment un système puissant pour créer et envoyer des e-mails - avec prise en charge des messages en plusieurs parties, intégration Twig, inlining CSS, pièces jointes et bien plus encore. Faites-les installer avec :

```
$ composer require symfony/mailer
```

Transport Setup

Emails are delivered via a "transport". Out of the box, you can deliver emails over SMTP by configuring the DSN in your .env file (the user, pass and port parameters are optional):

```
# .env
MAILER_DSN=smtp://user:pass@smtp.example.com:port
```

YAML

```
# config/packages/mailer.yaml
framework:
   mailer:
      dsn: '%env(MAILER_DSN)%'
```

Using a 3rd Party Transport

Instead of using your own SMTP server or sendmail binary, you can send emails via a 3rd party provider. Mailer supports several - install whichever you want:

8. Autres ressources

Traduction

Configuration des transports
Les e-mails sont livrés via un "transport". Par défaut, vous pouvez envoyer des e-mails via SMTP en configurant le DSN dans votre fichier .env (les paramètres user, pass et port sont facultatifs) :

Utilisation d'un moyen de transport tiers

Au lieu d'utiliser votre propre serveur SMTP ou votre binaire sendmail, vous pouvez envoyer des e-mails via un fournisseur tiers. Mailer en prend en charge plusieurs - installez celui que vous voul

Each library includes a Symfony Flex recipe that will add a configuration example to your .env file. For example, suppose you want to use SendGrid. First, install it:

```
composer require symfony/sendgrid-mailer
```

You'll now have a new line in your .env file that you can uncomment:

```
 # .env
MAILER_DSN=sendgrid://KEY@default
```

The MAILER_DSN isn't a real address: it's a convenient format that offloads most of the configuration work to mailer. The sendgrid scheme activates the SendGrid provider that you just installed, which knows all about how to deliver messages via SendGrid. The only part you need to change is the KEY placeholder.

Each provider has different environment variables that the Mailer uses to configure the actual protocol, address and authentication for delivery. Some also have options that can be configured with query parameters at the end of the MAILER_DSN - like ?region= for Amazon SES or Mailgun. Some providers support sending via http, api or smtp. Symfony chooses the best available transport, but you can force to use one:

Traduction

Chaque bibliothèque comprend une recette Symfony Flex qui ajoutera un exemple de configuration à votre fichier .env. Par exemple, supposons que vous souhaitiez utiliser SendGrid. Tout d'abord, installez-le :
Le MAILER_DSN n'est pas une vraie adresse : c'est un format pratique qui décharge la plupart du travail de configuration sur le mailer. Le schéma sendgrid active le fournisseur SendGrid que vous venez d'installer, qui sait tout sur la façon de livrer des messages via SendGrid. La seule partie que vous devez modifier est l'espace réservé KEY.

Chaque fournisseur a différentes variables d'environnement que l'expéditeur utilise pour configurer le protocole, l'adresse et l'authentification réels pour la livraison. Certains ont également des options qui peuvent être configurées avec des paramètres de requête à la fin du MAILER_DSN - comme ?region= pour Amazon SES ou Mailgun. Certains fournisseurs prennent en charge l'envoi via http, api ou smtp. Symfony choisit le meilleur transport disponible, mais vous pouvez en forcer un :

9. Informations complémentaires

Creating & Sending Messages

To send an email, get a Mailer instance by type-hinting MailerInterface and create an Email object:

```
# // src/Controller/MailerController.php
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Mailer\MailerInterface;
use Symfony\Component\Mime\Email;
use Symfony\Component\Routing\Annotation\Route;

class MailerController extends AbstractController
{
   #[Route('/email')]
   public function sendEmail(MailerInterface $mailer): Response
   {
      $email = (new Email())
         ->from('hello@example.com')
         ->to('you@example.com')
         //->cc('cc@example.com')
         //->bcc('bcc@example.com')
```

```
        //->replyTo('fabien@example.com')
        //->priority(Email::PRIORITY_HIGH)
        ->subject('Time for Symfony Mailer!')
        ->text('Sending emails is fun again!')
        ->html('<p>See Twig integration for better HTML integration!</p>');

    $mailer->send($email);

    // ...
  }
}
```

Traduction

Création et envoi de messages
Pour envoyer un e-mail, obtenez une instance Mailer en saisissant MailerInterface et créez un objet Email :