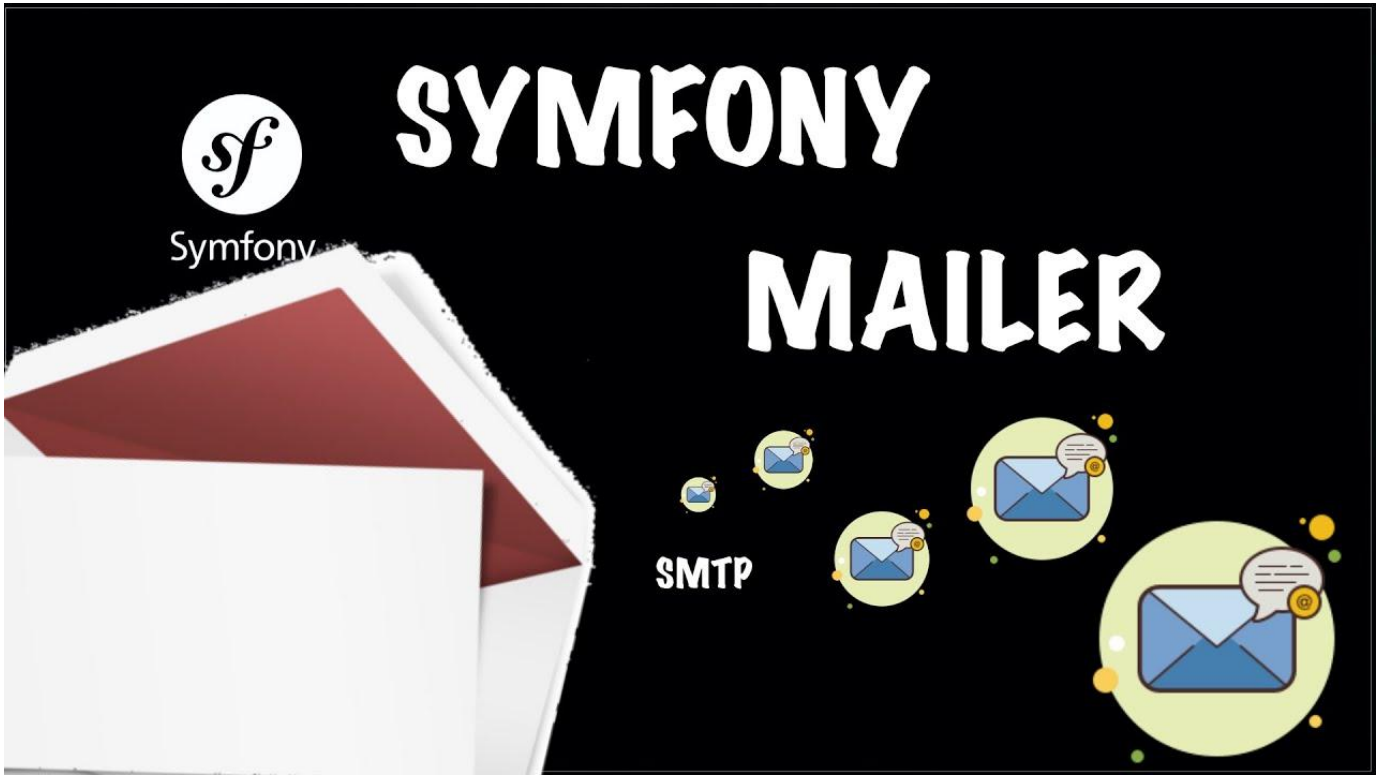
Symfony
SYMFONY
MAILER
SMTP

Mailer SYMFONY

Envoie un mail via mailer symfony

Créer un service pour l'envoi d'un mail

- Créer une classe src/Service/EmailService.php

```php
<?php
// src/Service/EmailService.php
namespace App\Service;

use Symfony\Component\Mailer\MailerInterface;
use Symfony\Component\Mime\Email;

class EmailService
{
    private $mailer;

    public function __construct(MailerInterface $mailer)
    {
        $this->mailer = $mailer;
    }

    public function sendEmail(string $from, string $to, string $subject, string $content): void
    {
        $from = $from ?? $_ENV['EMAIL_SENDER'];
        $email = (new Email())
            ->from($from)
            ->to($to)
            ->subject($subject)
            ->text($content)
            ->html('<p>' . $content . '</p>');

        $this->mailer->send($email);
    }
}
```

MailerInterface
Permet l'envoi des mails en utilisant un service de transport (SMTP, Sendmail, etc.).

Email
Permet de créer et de configurer un mail avec ses destinataires, sujet, contenu, et pièces jointes.

Configurer les paramètres de Gmail dans .env

- Modifier la variable: **MAILER_DSN** pour utiliser le service SMTP de Gmail.

```
###> symfony/mailer ###
MAILER_DSN=smtp://email:password@smtp.gmail.com:587?encryption=tls&auth_mode=login
###< symfony/mailer ###
```

adresse mail **mot de passe**

- smtp.gmail.com: Le serveur SMTP de Gmail.
- 587?encryption=tls: Le port avec le chiffrement tls pour la sécurité.
- auth_mode=login: L'authentification est effectuée avec l'email et le mot de passe.

Envoi de mail après inscription

- **Modifier l'entité User:**
  - Ajouter un champ token de type string et un champ isVerified de type booléen dans l'entité User
  - Effectuer une migration pour les nouveaux champs:

    php bin/console make:migration php bin/console doctrine:migrations:migrate
- **Générer un token de confirmation lors de l'inscription:**
  - Installer le composant symfony/uid:

    composer require symfony/uid

    Un UUID est une classe conçu pour générer un identifiant unique

Envoi de mail après l'inscription

- **Modifier le contrôleur InscriptionControllor:**
  - Injecter le service EmailService:

constructeur

```php
private $emailService;
public function __construct(EmailService $emailService)
{
    $this->emailService = $emailService;
}
```

ou

méthode

```php
#[Route('/register', name: 'app_register')]
public function register(
    Request $request,
    UserPasswordHasherInterface $userPasswordHasher,
    EntityManagerInterface $entityManager,
    EmailService $emailService
    ): Response
```

Envoi de mail après l'inscription

- **Modifier le contrôleur InscriptionControllor:**
    a. Importer le composant Uuid pour générer et manipuler des UUID (Universally Unique Identifiers).
    b. Générer le token unique:

```php
// Générer un token de confirmation unique
$token = Uuid::v4()->toRfc4122();
$user->setToken($token);

$entityManager->persist($user);
$entityManager->flush();
```

Envoi de mail après l'inscription

c. Générer un lien absolu

```php
// générer un lien absolu
$confirmationLink = $this->generateUrl('app_confirm_email', [
    'token' => $token,
], UrlGeneratorInterface::ABSOLUTE_URL);
```

La méthode generateUrl hérité de AbstractController prend 3 paramètres:

- **app_confirm_email**: le nom d'une route définie dans un contrôleur
- **'token'=>$token**: un paramètre nommé token
- **UrlGeneratorInterface::ABSOLUTE_URL**: constante indique que le url est absolu (contenant le nom de domaine)

Envoi de mail après l'inscription

d. Envoyer l'email avec le lien de confirmation

```
// Envoyer l'email avec le lien de confirmation
$this->emailService->sendEmail(
    $user->getEmail(),
    'Veuillez confirmer votre inscription',
    'Merci de vous être inscrit.
    Veuillez confirmer votre compte en cliquant sur ce lien :
    <a href="' . $confirmationLink . '">Confirmer mon compte</a>'
);
```

$this->emailService->sendEmail( ) : Cette ligne appelle la méthode sendEmail( ) d'un service nommé emailService.

Envoi de mail après l'inscription

- **Créer la route pour la confirmation d'email:**
    a. Créer le contrôleur EmailConfirmationController avec le même nom de la route envoyé dans le lien:

```php
#[Route('/confirm-email/{token}', name: 'app_confirm_email')]
public function index(
    string $token,
    UserRepository $userRepository,
    EntityManagerInterface $em): Response
{
    // Rechercher l'utilisateur avec le token
    $user = $userRepository->findOneBy(['confirmationToken' => $token]);

    if (!$user) {
        throw $this->createNotFoundException('Ce token de confirmation est invalide.');
    }

    // Activer le compte de l'utilisateur
    $user->setConfirmationToken(null);  // On supprime le token après confirmation
    $user->setVerified(true);  // Activer le compte
    $em->flush();

    return $this->render('email_confirmation/index.html.twig', [
        'user' => $user,
    ]);
}
```

Envoi de mail après l'inscription

b. Créer la vue templates/email_confirmation/index.html.twig:

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

{% block title %}Email Confirmé{% endblock %}

{% block body %}
    <h1>Merci {{ user.email }}, votre email a été confirmé !</h1>
    <p>Vous pouvez maintenant vous connecter avec votre compte.</p>
{% endblock %}
```

Empêcher la connexion avant la confirmation de l'inscription

- **Créer un service UserChecker dans src/Security/**

```php
<?php
// src/Security/UserChecker.php

namespace App\Security;

use Symfony\Component\Security\Core\Exception\CustomUserMessageAccountStatusException;
use Symfony\Component\Security\Core\User\UserCheckerInterface;
use Symfony\Component\Security\Core\User\UserInterface;
use App\Entity\User;

class UserChecker implements UserCheckerInterface
{
    public function checkPreAuth(UserInterface $user)
    {
        // Vérifie si l'utilisateur passé à cette méthode est bien une instance de la classe "User"
        // Si ce n'est pas une instance de "User", on quitte simplement la méthode.
        if (!$user instanceof User) {
            return;
        }

        // Si l'utilisateur n'a pas confirmé son compte, on l'empêche de se connecter
        if (!$user->isVerified()) {
            throw new CustomUserMessageAccountStatusException("Votre compte n'a pas encore été confirmé.");
        }
    }

    public function checkPostAuth(UserInterface $user)
    {
        // Ici, tu peux ajouter d'autres vérifications après l'authentification si nécessaire.
    }
}
```

L'interface `UserCheckerInterface` impose la présence de `checkPreAuth()` et `checkPostAuth()` dans la classe.

L'exception CustomUserMessageAccountStatusException arrête le processus de connexion

Empêcher la connexion avant la confirmation de l'inscription

- **Modifier le fichier config/packages/security.yaml**

```yaml
firewalls:
    dev:
        pattern: ^/(_(profiler|wdt)|css|images|js)/
        security: false
    main:
        user_checker: App\Security\UserChecker
        lazy: true
        provider: app_user_provider
        form_login:
            login_path: app_login
```

Ajouter la ligne suivante

Cela indique que la class **UserChecker** sera utilisée comme **vérificateur** d'utilisateur

Références:

https://symfony.com/doc/current/mailer.html#installation