SWIFT MAILER SYMFONY

Mise en place de la bibliothèque **SWIFT MAILER** (bundle) permettant la gestion et l'envoi de courrier électronique (mail) dans SYMFONY. Cette bibliothèque est héberger sur GITHUB et est accessible et téléchargeable avec **COMPOSER**.

Avant d'installer cette bibliothèque, nous allons mettre en place une boite de messagerie virtuelle utilisée en phase de développement de site web.

Nous allons donc installer **MAILDEV** ! C'est une boite de messagerie virtuelle qui permet de tester et d'envoyer des mails à partir d'un site web. Voici les étapes d'installation :

1ère étape Maildev :

Il faut tout d'abord installer NODE.JS, pour cela rendez-vous à l'adresse suivante :

Installer node.js : https://nodejs.org/en/

L'installation est facile et rapide.

2ème étape Maildev :

Modifier le fichier **php.ini** de votre serveur local (MAMP, XAMPP, WAMPP etc…).

Faites une recherche dans le fichier (ctrl + f) et lancer la recherche « **smtp** » (system mail transfert protocol) et modifier le port par :

smtp_port=1025

3ème étape Maildev :

Dans l'exécutable de Windows (powershell ADMIN) ou un terminal MAC, lancer la commande suivante afin d'installer MAILDEV :

npm install -g maildev

Voilà ! L'installation est faite !!

Lancer ensuite dans le terminal afin de lancer et d'ouvrir MAILDEV la commande suivante :

maildev

S'il y a une erreur, Pour activer l'exécution des scripts PowerShell, il faut saisir la ligne suivante. Syntaxe pour retirer la restriction:

set-executionpolicy unrestricted

MAILDEV est maintenant actif !!

Pour ouvrir la boite de messagerie virtuelle MAILDEV sur le navigateur, envoyer l'adresse suivante

dans google :

---

localhost:1080

TRAITEMENT DE L'ENVOI DE MAIL

1ère étape SWIFT MAILER :

Dans Symfony modifier le fichier .env :

MAILER_URL=smtp://localhost:1025

2ème étape SWIFT MAILER :

Nous allons maintenant demander à COMPOSER (gestionnaire de dépendances) d'installer la bibliothèque (bundle) **SWIFT MAILER**, pour cela lancer la commande suivante dans un terminal :

composer require symfony/swiftmailer-bundle

3ème étape SWIFT MAILER :

Nous allons maintenant créer une nouvelle entité **Contact :**

php bin/console make :entity

firstname -> string -> 255 -> not null

lastname -> string -> 255 -> not null

phone -> string -> 255 -> not null

email -> string -> 255 -> not null

message -> text -> not null

Ajouter des contraintes à souhait.

use Symfony\Component\Validator\Constraints as Assert;

Il faut ensuite créer le fichier de migration :

php bin/console make :migration

Demander ensuite à DOCTRINE d'exécuter le fichier de migration en BDD :

php bin/console doctrine:migrations:migrate

4ème étape SWIFT MAILER :

Créer un formulaire de contact à partir de l'entité Contact :

php bin/console make :form ContactType

Créer une nouvelle méthode dans votre controller principal :

```
/**
  * @Route("/blog/contact", name="blog_contact")
 */
 public function contact(Request $request, EntityManagerInterface $manager)
{
      $contact = new Contact();
      $form = $this->createForm(ContactType::class, $contact);

      $form->handleRequest($request);

      if ($form->isSubmitted() && $form->isValid()) {

         $manager->persist($contact); // on prépare l'insertion
         $manager->flush(); // on execute l'insertion

      }
    return $this->render("blog/contact.html.twig", [
         'formContact' => $form->createView()
      ]);
 }
```

5ème étape SWIFT MAILER :

Créer une nouvelle vue **contact.html.twig** et afficher le formulaire

Nous allons maintenant créer un nouveau controller spécialement pour l'envoi de mail.

Créer un dossier **src/Notification** et dans ce dossier créer un fichier :

src/Notification/ContactNotification.php

Ajouter le code suivant :

```
namespace App\Notification;
use App\Entity\Contact;
use Twig\Environment;
class ContactNotification
{
    /**
     * @var \Swift_Mailer
     */
    private $mailer;

    /**
     * @var Environment
     */
    private $renderer;
    public function __construct(\Swift_Mailer $mailer, Environment $renderer)
    {
        $this->mailer = $mailer;
        $this->renderer = $renderer;
    }
    public function notify(Contact $contact)
    {
        $message = (new \Swift_Message('Message : ' . $contact->getMessage()))
            ->setFrom($contact->getEmail())
            ->setTo('gregorylacroix78@gmail.com')
            ->setReplyTo($contact->getEmail())
            ->setBody($this->renderer->render('emails/contact.html.twig', [
                'contact' => $contact
```

```
        ]), 'text/html');
    $this->mailer->send($message);
  }
}
```

$message = (new \Swift_Message('Message : ' . $contact->getMessage())) :

Permet d'afficher le sujet du message à la reception du mail.

->setFrom($contact->getEmail()) :

Expéditeur de l'Email.

->setTo('gregorylacroix78@gmail.com')

Destinataire de l'Email

->setReplyTo($contact->getEmail())

Adresse de réponse de l'Email

->setBody()

Contenu de l'Email, le corps du message déclarée dans le template : **contact.html.twig.**

Modifier le code en fonction de votre traitement (Email d'envoi, contenu etc…).

6ème étape :

Nous allons maintenant créer un dossier template spécialement pour des templates de mails

Vous pouvez consulter cette adresse pour des avoir des structures déjà faites:

https://mjml.io/

Dans le dossier templates créer un nouveau dossier :

templates/emails

Puis créer un nouveau Template pour l'envoi de mail :

templates/emails/contact.html.twig

Vous êtes libre pour la mise en forme du mail :

```
<table class="table table-bordered">
  <tr>
```

```
    <td>Prénom</td>
    <td>{{ contact.firstname }}</td>
  </tr>

  <tr>
    <td>Nom</td>
    <td>{{ contact.lastname }}</td>
  </tr>
  <tr>
    <td>Téléphone</td>
    <td>{{ contact.phone }}</td>
  </tr>
  <tr>
    <td>Email</td>
    <td>{{ contact.email }}</td>
  </tr>
  <tr>
    <td>Message</td>
    <td>{{ contact.message }}</td>
  </tr>
</table>
```

7ème étape :

Enfin modifier la méthode contact() dans votre controller afin d'appeler la classe permettant l'envoi de mail :

```
/**
   * @Route("/blog/contact", name="blog_contact")
 */
public function contact(Request $request, EntityManagerInterface $manager, ContactNotification $notification)
{
      $contact = new Contact();
      $form = $this->createForm(ContactType::class, $contact);
      $form->handleRequest($request);

      if ($form->isSubmitted() && $form->isValid()) {
         $notification->notify($contact);
         $this->addFlash('success', 'Votre Email a bien été envoyé');
         $manager->persist($contact); // on prépare l'insertion
         $manager->flush(); // on execute l'insertion
      }

      return $this->render("blog/contact.html.twig", [
         'formContact' => $form->createView()
      ]);
}
```

addFlash() est une méthode permettant d'afficher des notifications à l'utilisateur, dans le formulaire de contact (contact.html.twig) ajouter le code suivant pour afficher le message :

```
{% for message in app.flashes('success') %}
<div class="col-md-5 mx-auto alert alert-success text-center">
   {{ message }}
</div>
```

{% endfor %}