フレームワーク Symfony 4.0 を使ってメールフォームを作ります。新規プロジェクトとして作っていくのですでに作成済みの箇所は読み飛ばして下さい。

Symfony のインストール

本体のインストールは Composer で行います。「symfony-contact」の部分はプロジェクト名なので好きな名前に書き換えて下さい。

```
composer create-project symfony/skeleton symfony-contact
```

コントローラー

Symfony のコントローラーは src/Controller フォルダにあります。「ContactController.php」ファイルを作成して下さい。
今回必要なページはフォーム画面と送信完了画面です。Symfony ではルーティングの方法がいくつかありますが、今回はコントローラーのアノテーション（コメント）を使います。

メールフォームを作成する上でいくつかのコンポーネントをインストールする必要があります。

アノテーションによるルーティングを行うためのコンポーネント

```
composer require annotations
```

フォームを表示するためのコンポーネント

```
composer require form
```

CSRF(クロスサイトリクエストフォージェリ)対策のためのコンポーネント

```
composer require symfony/security-csrf
```

バリデーション（入力値の検証）を行うためのコンポーネント

```
composer require validator
```

メールを送信するための Swift Mailer

```
composer require mailer
```

コントローラーの全体像は次のようになりました。

```
1  <?php
2  namespace App\Controller;
3
4  use Symfony\Bundle\FrameworkBundle\Controller\Controller;
```

```php
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Validator\Constraints as Assert;
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\Form\Extension\Core\Type\EmailType;
use Symfony\Component\Form\Extension\Core\Type\TextareaType;
use Symfony\Component\Form\Extension\Core\Type\SubmitType;

class ContactController extends Controller
{
    /**
     * @Route("/contact", methods={"GET", "POST"}, name="contact")
     */
    public function form(Request $request)
    {
        $form = $this->createFormBuilder()
            ->add('name', TextType::class, [
                'constraints' =>[
                    new Assert\NotBlank(),
                    new Assert\Length(['max' => 100])
                ]
            ])
            ->add('email', EmailType::class, [
                'constraints' => [
                    new Assert\NotBlank(),
                    new Assert\Email()
                ]
            ])
            ->add('message', TextareaType::class, [
                'constraints' => [
                    new Assert\NotBlank()
                ]
            ])
            ->add('submit', SubmitType::class, ['label' => 'Submit'])
            ->getForm();

        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid()) {
            $data = $form->getData();
            $this->send($data);
            return $this->redirectToRoute('contact_result');
        }

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

    public function send($data)
    {
        $subject = 'Test Mail';
        $message = (new \Swift_Message($subject))
            ->setFrom('from@example.com')
            ->setTo('to@example.com')
            ->setBody(
                $this->renderView(
                    'emails/contact.html.twig',
                    [
                        'name' => $data['name'],
                        'email' => $data['email'],
                        'message' => $data['message']
                    ]
                ),
                'text/html'
```

```
            );

        $this->get('mailer')->send($message);
    }

    /**
     * @Route("/contact/result", methods="GET", name="contact_result")
     */
    public function result()
    {
        return $this->render('contact/result.html.twig');
    }
}
```

createFormBuilder() でフォームを作っています。今回の入力項目は名前、メールアドレス、本文の３つで、それぞれを add() で追加しています。constraints がバリデーションルールで、必要に応じてルールを追加します。利用可能なルールについてはこちらを見て下さい。

ビュー

ビューを表示するためには Twig テンプレートエンジンをインストールします。

```
composer require twig
```

ビューファイルは templates フォルダに作成していきます。今回は Bootstrap 4 を利用するのでひとまず base.html.twig の <head> に次の行を追記します。

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="s
```

次に contact フォルダを作り、その中に contact.html.twig ファイルと result.html.twig ファイルを作ります。

contact.html.twig

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

{% block body %}
<div class="container">
    {{ form_start(form) }}
        {{ form_errors(form) }}

        <div class="form-group">
            {{ form_label(form.name) }}
            {{ form_widget(form.name, {'attr': {'class': 'form-control'}}) }}
            {{ form_errors(form.name) }}
        </div>

        <div class="form-group">
            {{ form_label(form.email) }}
            {{ form_widget(form.email, {'attr': {'class': 'form-control'}}) }}
            {{ form_errors(form.email) }}
        </div>

```

```
        <div class="form-group">
            {{ form_label(form.message) }}
            {{ form_widget(form.message, {'attr': {'class': 'form-control'}}) }}
            {{ form_errors(form.message) }}
        </div>

        <div class="form-group">
            {{ form_widget(form.submit, {'attr': {'class': 'btn btn-primary'}}) }}
        </div>
    {{ form_end(form) }}
</div>
{% endblock %}
```

result.html.twig

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

{% block body %}
<div class="container">
    <div class="card">
        <div class="card-body">
            <h3>Thanks.</h3>
            <p>Your email has been successfully sent.</p>
        </div>
    </div>
</div>
{% endblock %}
```

メールの送信

メールを送信するには送信サーバーの設定が必要です。先程の mailer コンポーネントをインストールすると .env ファイルに設定箇所が追加されるので、MAILER_URL の部分をお使いのメールアカウントに合わせて書き換えて下さい。
詳細についてはこちらをお読み下さい。

次にメールの文面を用意します。場所は templates/emails です。

contact.html.twig

```
<p>Name: {{ name }}</p>
<p>Email: {{ email }}</p>
<p>{{ message|e|nl2br }}</p>
```

これでメールフォームは完成です