

Symfony

Découverte du framework

Samih Habbani

Installations

- Installer chocolatey (<https://chocolatey.org/install>)
- Installer php (choco install php)
- Installer scoop (iex "& {\$(irm get.scoop.sh)} -RunAsAdmin »)
- Installer Symfony CLI (scoop install symfony-cli)
- symfony new blog_afpa --version="6.4.*" —webapp
- Symfony serve

Présentation

Du framework

- Français
- Cadre de travail
- Permet de créer des application web complètes (front - back)
- CRUD
- MVC
- SPA
- REST
- Twig (moteur de template qui permet de gérer le front)

Installation

Les pré-requis

- Php 8.2
- Composer (gestionnaire de paquets en PHP -> ça permet d'installer les dépendances d'un projet PHP)
- Symfony CLI (command line de symfony)
- Serveur web (WAMP)
- Node JS

Modèle REST

Explication sur les routes

articles/id => méthode GET

GET

articles/ => méthode POST
avec un paramètre post id

POST

articles/id => méthode PUT

PUT

articles/id => méthode DELETE

DELETE

Créer un projet Symfony

Symfony CLI

- symfony check:requirements
- symfony new my_project_directory --version="6.2.*" --webapp
- cd my_project_directory
- Symfony serve

Utiliser le protocole https

Installation

- `symfony server:ca:install`

Créer un projet Symfony

Commandes

- `symfony new my_project_directory --version="6.3.*" --webapp`

Se connecter sur GIT

A faire quand tu veux te co sur une autre machine Avec un autre user sur un autre projet

- `git config user.name 'name'`
- `git config user.userName 'userName'`
- `git config user.email 'email'`
- `git config user.password 'pwd'`

- Générer un token sur votre profil Git dans settings-> developer settings -> generate new token
- `git remote set-url origin https://<TOKEN>@github.com/<userName>/<Project>.git`

Initialiser un répertoire Git

Git init

- git init
- git add -A : ajout des fichiers au staging area
- git commit -m 'architecture du projet' : ajouter les fichiers au local directory pour futur envoi
- git branch -M main : positionner sur la branche Main (branche principale, dernière version stable de votre projet, accessible par tous)
- git remote add origin <https://github.com/webstartMaterial/blog-afpa.git> (connecter mon repo en local à mon repo en ligne)
- git push -u origin main (j'ai envoyé sur mon repo en ligne toutes mes modifs en local, pour que tout le monde puisse les récupérer sur la branche Main)
- Dans la réalité, ne jamais envoyer directement ses modifs sur Main (car si vos modifs créent des bugs, ça va créer des bugs pour tout le monde => toujours créer une nouvelle branche pour fonctionnalité développée)

MakerBundle

Créer notre première page

- `composer require --dev symfony/maker-bundle`
- Permet de créer des classes, des controllers, des entités via des lignes de commandes

Création de notre premier contrôleur

Maker bundle

- `composer require --dev symfony/maker-bundle`
- Permet de créer des classes, des contrôleurs, des entités via des lignes de commandes
- `php bin/console make:controller HomeController`

Architecture REST

Explications

- Architecture spécifique pour la gestion des urls
- Un url par entité
- Pour gérer l'entité Article
- /article/1 => get
- /article/1 => post (ajout)
- /article/1 => put (modification)
- /article/all => get All
- /article/1 => delete

Routes

Explications

- Nommée
- Liée à une méthode
- Capte une requête
- Renvoie une réponse
- Render()
- Variables à ma vue
- Routes multiples
- Paramètres dans l'url
- Valider les données en entrée d'un paramètre

ORM Doctrine

Explications

- C'est une interface qui se place entre votre application web et votre bdd et qui représente les tables en base par des objets (classes)
- Son rôle est de mapper une classe php (ou autres) à une table (entité) en base
- => représenter une table via une classe PHP
- Je vais avoir autant de classe Entité que de tables en BDD
- Doctrine va nous permettre de créer nos classes et de générer automatiquement les tables en BDD en fonction des classes générées (que nous n'allons créer que des classes, les tables et la base sera automatiquement générée)

ORM Doctrine

Installation

- `composer require symfony/orm-pack`
- `composer require --dev symfony/maker-bundle`
-

ORM Doctrine

Utilisation

- `.env.local => setter la connexion à la BDD`
- `php bin/console doctrine:database:create`
- `php bin/console make:entity`
- `php bin/console make:migration`
- `php bin/console doctrine:migrations:migrate`

Installer le driver pdo_mysql

Comment?

- Where php => indiquer ou se trouve la version de php que vous utilisez pour faire tourner votre projet symfony
- tools/php.ini => ouvrez avec Visual studio code
- Déocher l'extension extension=pdo_mysql (faut enlever le ;)
- Redémarrer votre serveur
- La commande create:database devrait fonctionner

EntityManagerInterface

Utilisation

- Injection de dépendance au niveau de nos méthodes de controller
- `showArticleByCategory(EntityManagerInterface $entityManager)`
- Permet de récupérer le Repository d'une classe
- Chaque repository hérite de `ServiceEntityRepository` qui nous donne accès à un CRUD de base (`find()`, `findAll()`, `findBy()`...)

Rappel des commandes

GIT

- Git status (qu'est-ce qui a été modifié dans tous le projet)
- Git diff chemin_fichier (qu'est-ce qui a été modifié dans un fichier)
- Pour sortir du git diff :q + entrer
- git checkout src/Entity/Articles.php => annuler les modifs du fichier
- git reset —hard => ça va annuler toutes les modifications que vous avez fait dans votre projet par rapport à ce qu'il y a en ligne
- git clean -fd => ça annuler les fichiers créés
- Git reset —hard => pour annuler les fichiers supprimés

Gérer les assets avec symfony

Installation

- `composer require symfony/webpack-encore-bundle`
- `Npm install`
- `npm install jquery --save-dev`
- `npm install bootstrap --save-dev`
- `npm install sass-loader@^13.0.0 sass --save-dev`
- `npm run watch` (lancer cette commande quand on fait du css ou js, car elle récupère votre css et js, le compile et l'inject dans le dossier public qui est le dossier lu par votre navigateur)
- Renommer `app.css` en `app.scss`
- `npm install jquery @popperjs/core --save-dev`

Customiser l'affichage pour la gestion des erreurs

Explication


- composer require symfony/twig-pack

```
1 templates/  
2 └─ bundles/  
3     └─ TwigBundle/  
4         └─ Exception/  
5             └─ error404.html.twig  
6             └─ error403.html.twig  
7             └─ error.html.twig      # All other HTML errors (including 500)
```

Example 404 Error Template

To override the 404 error template for HTML pages, create a new `error404.html.twig` template located at `templates/bundles/TwigBundle/Exception/`:

```
1 {# templates/bundles/TwigBundle/Exception/error404.html.twig #}  
2 {% extends 'base.html.twig' %}  
3  
4 {% block body %}  
5     <h1>Page not found</h1>  
6  
7     <p>  
8         The requested page couldn't be located. Checkout for any URL  
9         misspelling or <a href="{{ path('homepage') }}">return to the homepage</a>.  
10    </p>  
11 {% endblock %}
```



Commandes pratiques

Liste

- `php bin/console cache:clear`
- Dans le terminal : `ctr + c` => arrête le serveur

TWIG

Fonctionnalités

- `{% %}` => block de code
- `{{ }}` => interpréter une variable de vue
- `{% for %} {% endfor %}`
- `path('mon_chemin', { id : val })` => url dynamique
- `{{ article.date|date("d-m-Y") }}` => filtre
- `{% block body %} {% endblock %}`
- `{% extends 'base.html.twig' %}` => permet d'hériter de la structure de base du projet
- `{% include 'nav.html.twig' %}` => permet d'inclure un template html dans la page
- ``

Gestion des images sur le serveur

Copier les images depuis le dossier assets dans public

- `npm install file-loader@^6.0.0 --save-dev`
- `Npm run dev => copier vos assets dans public`

```
1 // webpack.config.js
2
3 Encore
4   // ...
5   .setOutputPath('public/build/')
6
7   + .copyFiles({
8     +   from: './assets/images',
9     +
10    +   // optional target path, relative to the output dir
11    +   to: 'images/[path][name].[ext]',
12    +
13    +   // if versioning is enabled, add the file hash too
14    +   //to: 'images/[path][name].[hash:8].[ext]',
15    +
16    +   // only copy files matching this pattern
17    +   //pattern: /\. (png|jpg|jpeg)$/
18   + })
```

Mettre en place une pagination

KnpPaginatorBundle

- composer require knplabs/knp-paginator-bundle
- Créer un fichier config/package/knp_paginator.yaml

```
config > packages > ! knp_paginator.yaml
1 knp_paginator:
2     page_range: 5                # number of links showed in the pagination m
3     default_options:
4         page_name: page          # page query parameter name
5         sort_field_name: sort     # sort field query parameter name
6         sort_direction_name: direction # sort direction query parameter name
7         distinct: true           # ensure distinct results, useful when ORM q
8         filter_field_name: filterField # filter field query parameter name
9         filter_value_name: filterValue # filter value query parameter name
10    template:
11        pagination: '@KnpPaginator/Pagination/bootstrap_v5_pagination.html.twig'
12        sortable: '@KnpPaginator/Pagination/sortable_link.html.twig' # sort link tem
13        filtration: '@KnpPaginator/Pagination/filtration.html.twig' # filters templ
```

```
// App\Controller\ArticleController.php

public function listAction(EntityManagerInterface $em, PaginatorInterface $paginator, Request $request)
{
    $dql = "SELECT a FROM AcmeMainBundle:Article a";
    $query = $em->createQuery($dql);

    $pagination = $paginator->paginate(
        $query, /* query NOT result */
        $request->query->getInt('page', 1), /*page number*/
        10 /*limit per page*/
    );

    // parameters to template
    return $this->render('article/list.html.twig', ['pagination' => $pagination]);
}
```

```
<div class="navigation">
|     {{ knp_pagination_render(listArticles) }}
</div>
```

Pour avoir la pagination en Fr

Configuration projet

```
KnnpaginatorBundle.fr.yml — blog-atpa

! KnnpaginatorBundle.fr.yml U × ! translation.yml M ! services.yml M

translations > ! KnnpaginatorBundle.fr.yml
1 | label_next: Suivant
2 + label_previous: Précédent
```

```
translation.yml — blog-atpa

! KnnpaginatorBundle.fr.yml U ! translation.yml M × ! services.yml M

config > packages > ! translation.yml
1 | framework:
2 + | default_locale: '%locale%'
3 | translator:
4 | | default_path: '%kernel.project_dir%/translations'
5 | | fallbacks:
6 | | | - '%locale%'
7 | # providers:
```

```
! KnnpaginatorBundle.fr.yml U ! translation.yml M ! services.yml M ×

config > ! services.yml
1 + # This file is the entry point to configure your own services.
2 | # Files in the packages/ subdirectory configure your dependencies.
3 |
4 | # Put parameters here that don't need to change on each machine where the app is
5 | # https://symfony.com/doc/current/best\_practices.html#use-parameters-for-application-configuration
6 | parameters:
7 | | locale: 'fr'
8 |
```

Créer un formulaire

MakerBundle - formbuilder

- composer require symfony/form
- Php bin/console make:form
- ContactType
- Contact (entité)

```
$contact = new Contact();  
$form = $this->createForm(ContactType::class, $contact);  
  
return $this->render('contact/index.html.twig', [  
    'contact_form' => $form,  
]);
```

```
$builder  
    ->add('firstName', TextType::class, [  
        'label' => 'Prénom',  
        'attr' => array(  
            'placeholder' => 'Entrer votre prénom'  
        )  
    ] )
```

Validation des champs Symfony

La procédure

- composer require symfony/validator

```
#[ORM\Column(type: Types::TEXT)]  
#[Assert\Length(  
    min: 2,  
    max: 50,  
    minMessage: 'Your first name must be at least {{ limit }} characters long',  
    maxMessage: 'Your first name cannot be longer than {{ limit }} characters',  
)]
```


Gérer les images dans le formulaire

Ajout - Modification

```
#[Assert\Image(  
    maxSize: "1024k"  
)]  
2 references  
private $posterFile;
```

```
->add('posterFile', FileType::class, array(  
    'required' => false,  
))  
add('relatedSubjects', CollectionType::class,
```

```
if($form->isSubmitted() && $form->isValid()) {  
  
    if($file = $article->getPosterFile()) {  
        $fileName = md5(uniqid()) . '.' . $file->guessExtension();  
        $file->move('./images/articles/', $fileName);  
  
        $article->setPicture($fileName);  
    }  
  
    $entityManager->persist($article);  
    $entityManager->flush();  
  
    $this->addFlash('confirmation', 'Votre article a bien été modifié en BDD');  
}
```

Supprimer une photo sur le serveur

Kezako

```
public function deleteArticle(EntityManagerInterface $entityManager, string $id, Request $request): Response {  
  
    // si j'ai un post  
    // je récupère le paramètre POST ID  
    $id = $request->get('id');  
    $article = $entityManager->getRepository(Articles::class)->find($id);  
  
    if($picture = $article->getPicture()) { // si j'ai une image, je la supprimer lors de la suppression de l'article  
        @unlink('./images/articles/' . $picture);  
    }  
  
    $entityManager->remove($article);  
    $entityManager->flush();  
  
    // rediriger vers la page d'accueil avec un msg de confirmation  
  
    $this->addFlash('confirmation', 'L'article a bien été supprimé !');  
    return $this->redirectToRoute('app_home');  
}
```

Messages Flash

Notification en session utilisable une fois liées au User

```
if ($form->isSubmitted() && $form->isValid()) {  
  
    $contact = $form->getData();  
    $entityManager->persist($contact);  
    $entityManager->flush();  
  
    $this->addFlash('confirmation', 'Votre email a bien été envoyé !');  
  
    // return $this->redirectToRoute('task_success');  
}
```


Fixtures

Jeu de données

- `composer require --dev orm-fixtures`
- `composer require fakerphp/faker`
- `php bin/console doctrine:fixtures:load --append`
- `composer require fakerphp/faker`

Fixtures

Jeu de données

```
$faker = Factory::create('fr_FR'); // créer mon faker

// $product = new Product();
// $manager->persist($product);

for($i = 0; $i <= 5; $i++) {

    // Création des catégories
    $category = new Category();
    $category->setName($faker->word());
    $category->setDescription($faker->text());

    $manager->persist($category); // insérer en BDD la catégorie

    // Création de nos articles
    for($j = 0; $j <= 3; $j++) {
        $article = new Articles();
        $article->setAuthor($faker->name());
        $article->setAuthorWebsite($faker->name());
        $article->setCatchPhrase($faker->text());
        $article->setCategory($category);
        $article-> setDate($faker->dateTime());
        $article->setPicture($faker->imageUrl());
        $article->setRelatedCourse($faker->numberBetween(1, 99));
        $article->setTitle($faker->title());
        $article->setRelatedSubjects([$faker->word()]);
        $article->setLegendMainPicture($faker->text());
        $article->setChapo($faker->text());
        $article->setDescription($faker->text());

        $manager->persist($article); // insère en bdd l'article
    }
}
```

Récupérer un projet existant sur ordi sans config

La procédure

- Installer chocolatey
- Installer scoop
- Installer php
- Installer composer
- Installer symfony CLI
- Git clone + lien répo GitHub => copie le projet sur GitHub sur votre ordi
- Composer update => réinstaller les dépendances php
- Npm i => réinstalle les dépendances npm (webpack encore)
- Créer le fichier .env.local
- php bin/console doctrine:database:create => créer la BDD
- php bin/console doctrine:migrations:migrate => créer les tables
- Npm run watch => compile les assets dans public
- Symfony serve => démarre le serveur

Récupérer un projet existant

La procédure

- `Git clone + lien répo GitHub =>` copie le projet sur GitHub sur votre ordi
- `Composer update =>` réinstaller les dépendances php
- `Npm i =>` réinstalle les dépendances npm (webpack encore)
- Créer le fichier `.env.local`
- `php bin/console doctrine:database:create =>` créer la BDD
- `php bin/console doctrine:migrations:migrate =>` créer les tables
- `Npm run watch =>` compile les assets dans public
- `Symfony serve =>` démarre le serveur

Récupérer les modifs sur GitHub sur votre ordi

La procédure

- Git pull depuis la branche Main

INSERT EN SYMFONY

Kezako

```
if ($form->isSubmitted() && $form->isValid()) {  
  
    $contact = $form->getData();  
    $entityManager->persist($contact);  
    $entityManager->flush();  
  
    $this->addFlash('confirmation', 'Votre email a bien été envoyé !');  
  
    // return $this->redirectToRoute('task_success');  
}
```

DELETE EN SYMFONY

Kezako

```
// si j'ai un post
if ($request->isMethod('post')) {
    // je récupère le paramètre POST ID
    $id = $request->get('id');
    $article = $entityManager->getRepository(Articles::class)->find($id);
    $entityManager->remove($article);
    $entityManager->flush();

    // rediriger vers la page d'accueil avec un msg de confirmation

    $this->addFlash('confirmation', 'L'article a bien été supprimé !');
    return $this->redirectToRoute('app_home');
}
```


UPDATE EN SYMFONY

Kezako

```
#[Route('/article/{id}/modify', name: 'modify_article', requirements: ['id' => '\d+'])]
4 references | 0 overrides
public function modifyArticle(EntityManagerInterface $entityManager, string $id, Request $request) {

    // faut récupérer l'article en BDD qui à l'id $id
    // ensuite créer le formulaire via ArticleType
    // render la page articles/article-modify.html.twig
    // faut render le formulaire dans cette page

    $article = $entityManager->getRepository(Articles::class)->find($id); // récupère l'article en BDD
    $form = $this->createForm(ArticlesType::class, $article);

    $form->handleRequest($request);

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

        if($file = $article->getPosterFile()) {
            $fileName = md5(uniqid()) . '.' . $file->guessExtension();
            $file->move('.', $fileName);

            $article->setPicture($fileName);
        }

        $entityManager->persist($article);
        $entityManager->flush();

        $this->addFlash('confirmation', 'Votre article a bien été modifié en BDD');
    }
}
```


FILTRE REQÊTE AVEC QUERY BUILDER

Afficher seulement les catégories qui ont des articles

```
public function findCategoriesWithArticles(): array
{
    // SELECT * FROM category c
    // WHERE c.id IN (SELECT DISTINCT(a.category_id) from articles a);

    $entityManager = $this->getEntityManager();

    $query = $entityManager->createQuery(
        "SELECT c FROM App\Entity\Category c
        WHERE c.id IN (SELECT DISTINCT(a.category) FROM App\Entity\Articles a)"
    );

    return $query->execute();

    // $query = $entityManager->createQueryBuilder()
    // ->select('c')
    // ->from('App\Entity\Category', 'c')
    // ->where('c.id IN (SELECT DISTINCT a.category FROM App\Entity\Articles a)')
    // ->getQuery()->getResult();
}
```

ALLER PLUS LOIN AVEC LE FORMULAIRE

Kezako

- CHECKBOX
- INPUT TYPE FILE
- UPLOADER UNE IMAGE ET L’AFFICHER
- GÉRER AJOUT/SUPPRESSION IMAGE

Personnaliser l'apparence d'un formulaire

Kezako

```
<div class="mt-5 pt-5 px-5 d-flex flex-column align-items-center">

  {{ form_start(add_article_form) }}

    {{ form_row(add_article_form.title) }}
    {{ form_row(add_article_form.catchPhrase) }}
    {{ form_row(add_article_form.date) }}
    {{ form_row(add_article_form.author) }}
    {{ form_row(add_article_form.description) }}
    {{ form_row(add_article_form.posterFile) }}
    {{ form_row(add_article_form.relatedSubjects) }}
    {{ form_row(add_article_form.chapo) }}
    {{ form_row(add_article_form.legendMainPicture) }}
    {{ form_row(add_article_form.authorWebsite) }}
    {{ form_row(add_article_form.relatedCourse) }}
    {{ form_row(add_article_form.category) }}
    {{ form_row(add_article_form.Modifier, { 'label': 'Ajouter' }) }}

  {{ form_end(add_article_form) }}

</div>
```

Créer un backOffice

EasyAdmin

- `composer require easycorp/easyadmin-bundle`
- `php bin/console make:admin:dashboard`
- `php bin/console make:admin:crud`

LA SÉCURITÉ AVEC SYMFONY

Kezako

- CRÉER UN USER
- AUTHENTIFICATION
- INSCRIPTION
- PROTÉGER NOS ROUTES
- CONFIRMER LA CRÉATION DU COMPTE PAR MAIL
- RÉINITIALISER UN MOT DE PASSE
- Mot de passe oublié

LA SÉCURITÉ AVEC SYMFONY

CRÉATION LOGIN-FORM ET REGISTER-FORM

- `composer require symfony/security-bundle`
- `php bin/console make:user`
- `php bin/console make:auth`
- `php bin/console make:migration`
- `php bin/console doctrine:migrations:migrate`
- `composer require symfonycasts/verify-email-bundle`
- `php bin/console make:registration-form`
- `php bin/console make:migration`
- `php bin/console doctrine:migrations:migrate`
- `composer require symfony/google-mailer`

Se connecter avec symfony

Make:auth

- `php bin/console make:auth`

Envoyer un mail en local avec Symfony

Maildev

- `npm install -g maildev`
- `maildev`
- Dans `.env.local` : `MAILER_DSN=smtp://localhost:1025?verify_peer=0`
- `http://0.0.0.0:1080`
- (`maildev -v --ip 127.0.0.1` pour les pc)
- Désactiver l'envoi de mail asynchrone avec Symfony `config/package/messenger.yaml`

```
routing:
    # Symfony\Component\Mailer\Messenger\SendEmailMessage: async
    Symfony\Component\Notifier\Message\ChatMessage: async
    Symfony\Component\Notifier\Message\SmsMessage: async
```


Mot de passe oublié

Réinitialiser un mdp oublié

- `composer require symfonycasts/reset-password-bundle`
- `php bin/console make:reset-password`
- `php bin/console make:migration`
- `php bin/console doctrine:migrations:migrate`

Envoyer un mail en local avec Symfony

MAIL + TEMPLATE

- composer require symfony/mailer

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

YAML XML PHP

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

```
$ composer require symfony/twig-bundle

# or if you're using the component in a non-Symfony app:
# composer require symfony/twig-bridge
```

HTML Content

To define the contents of your email with Twig, use the [TemplatedEmail](#) class. This class extends the normal [Email](#) class but adds some new methods for Twig templates:

```
1 use Symfony\Bridge\Twig\Mime\TemplatedEmail;
2
3 $email = (new TemplatedEmail())
4     ->from('fabien@example.com')
5     ->to(new Address('ryan@example.com'))
6     ->subject('Thanks for signing up!')
7
8     // path of the Twig template to render
9     ->htmlTemplate('emails/signup.html.twig')
10
11     // pass variables (name => value) to the template
12     ->context([
13         'expiration_date' => new \DateTime('+7 days'),
14         'username' => 'foo',
15     ])
16 ;
```

SESSION SYMFONY

Kezako

- MESSAGE FLASH
- USER

SESSION SYMFONY

App

```
{% block body %}

    <div class="mt-5 p-5">
        <ul>
            {# <li> {{ dump(app.user) }} </li> #}
            <li> id : {{ app.user.id }} </li>
            <li> adresse email : {{ app.user.email }} </li>
        </ul>
    </div>

{% endblock %}
```

```
{% if app.user %}
    <li class="nav-item">
        <a class="nav-link text-light" href="{{path('app_profile')}}">Profile</a>
    </li>
    <li class="nav-item">
        <a class="nav-link text-light" href="{{path('app_logout')}}">Déconnexion</a>
    </li>
{% else %}
    <li class="nav-item">
        <a class="nav-link text-light" href="{{path('app_login')}}">Connexion</a>
    </li>
    <li class="nav-item">
        <a class="nav-link text-light" href="{{path('app_register')}}">Inscription</a>
    </li>
{% endif %}
```

SÉCURITÉ ET ROLES

Sécuriser nos routes

```
security.yaml — blog-alpa

! security.yaml M ×
config > packages > ! security.yaml
31      # switch_user: true
32
33      # Easy way to control access for large sections of your site
34      # Note: Only the *first* access control that matches will be used
35      access_control:
36      - { path: ^/admin, roles: ROLE_ADMIN }
37      - { path: ^/profile, roles: ROLE_USER }
38
```

Customiser un accès refusé

AccessDeniedHandler

YAML XML PHP

```
1 # config/packages/security.yaml
2 firewalls:
3     # ...
4
5     main:
6         # ...
7         entry_point: App\Security\AuthenticationEntryPoint
```

Copy

Customize the Forbidden Response

Create a class that implements [AccessDeniedHandlerInterface](#). This interface defines one method called `handle()` where you can implement whatever logic that should execute when access is denied for the current user (e.g. send a mail, log a message, or generally return a custom response):

```
1 // src/Security/AccessDeniedHandler.php
2 namespace App\Security;
3
4 use Symfony\Component\HttpFoundation\Request;
5 use Symfony\Component\HttpFoundation\Response;
6 use Symfony\Component\Security\Core\Exception\AccessDeniedException;
7 use Symfony\Component\Security\Http\Authorization\AccessDeniedHandlerInterface;
8
9 class AccessDeniedHandler implements AccessDeniedHandlerInterface
10 {
11     public function handle(Request $request, AccessDeniedException $accessDeniedExcept
12     {
13         // ...
14
15         return new Response($content, 403);
16     }
17 }
```

AFFICHER DES ÉLÉMENTS SELON LES ROLES

Dans TWIG

```
! security.yaml M X
config > packages > ! security.yaml
28      # https://symfony.com/doc/current/security.html#the-firewall
29
30      # https://symfony.com/doc/current/security/impersonating\_user.html
31      # switch_user: true
32
33      # Easy way to control access for large sections of your site
34      # Note: Only the *first* access control that matches will be used
35      access_control:
36          - { path: ^/admin, roles: ROLE_ADMIN }
37          - { path: ^/profile, roles: ROLE_USER }
38          - { path: ^/article/[0-9]+/modify, roles: ROLE_ADMIN }
39          - { path: ^/article/[0-9]+/delete, roles: ROLE_ADMIN }
40
```

```
{% if is_granted('ROLE_ADMIN') %}
    <a href="{{path('delete_article', { id : article.id } }}" class="btn btn-danger"> SUPPRIMER </a>
    <a href="{{path('modify_article', { id : article.id } }}" class="btn btn-warning"> MODIFIER </a>
{% endif %}
```


CRÉER UN SERVICE

Kezako

Symfony Flex

Kezako

RAPPEL DES FICHIERS DE CONFIGURATION

Kezako

- security.yaml : sécuriser vos routes

Référencer un site

Le faire apparaître dans Google

- Générer un sitemap.xml à la racine de mon projet
- Ajouter une propriété sur Google Search Console
- Ajouter l'entrée TXT Dans la zone DNS de votre nom de domaine
- Créer un robots.txt à la racine de mon projet
- Optionnel (obligatoire pour les projets symfony) : .htaccess

Param converter Symphony

A chercher

Rappel du Jargon

Jargon

- Maker bundle
- Doctrine
 - Webpack
- ORM
- CRUD
- MVC
- ENTITY
- MODEL
- REPOSITORY
- CONTROLLER
- VUE
- MOTEUR DE TEMPLATE HTML
- TWIG
- RESPONSE
- REQUEST
- COMPOSER

Rappel du Jargon

Jargon

- Injection de dépendance

Icons

Font Awesome

- `npm install --save-dev @fortawesome/fontawesome-free`

Sitemaps

Architecture du projet pour Google Search Console

- Plan de votre site à déposer sur Google Search Console pour lancer l'indexation

Robots.txt

Bot

- Aider le robot google à indexer votre site

HTTPS - modifier le fichier .htaccess

Comment rediriger les connexions HTTP en HTTPS

- RewriteEngine on
- RewriteCond %{HTTPS} !=on
- RewriteRule .* https://%{HTTP_HOST}%{REQUEST_URI} [R=301,L]

Accessibilité numérique

Bonnes pratiques

- W3c
- Alt
- Aria-label
- Title
-

SEO

Bonnes pratiques

- Meta description
- Title
- Aria-label
- Alt
- Mot clés
- Netlinking

Responsive Web Design

Bonnes pratiques

- Meta viewport

Créer un crud autour d'une entité

symfony console make:crud

Comment protéger vos controllers Symfony?

CSRF Cross-Site Request Forgery

- `isCsrfTokenValid()`

Login

Pour se connecter et créer le formulaire de connexion

- `php bin/console make:security:form-login`

maildev n'est pas reconnue

Solution

- Npm list -g maildev
- Créer une variable d'environnement avec le dossier où se trouve le .cmd