

Mon PROJET APPLICATION WEB en Symfony



***TABLIEH Walid***

15 rue Wattu - 6700 STRASBOURG

 07 69 34 40 70  waledo.1997@hotmail.com

www.elan-formation.fr

**SOMMAIRE**

[I. Introduction de projet 3](#_Toc80698694)

[II. Les Application&Langues Utiliser 3](#_Toc80698695)

[III. Maquettage 7](#_Toc80698696)

[IV. MCD 9](#_Toc80698697)

[V. Modèle Vue Contrôleur (MVC) 9](#_Toc80698698)

[VI. Mise en place des entités (tables de la base de données) 12](#_Toc80698699)

[1- Relation (ASSO) de table 13](#_Toc80698700)

[2- Détail d'une entité Symfony 14](#_Toc80698701)

[3- Controller 15](#_Toc80698702)

[3- Template code 20](#_Toc80698703)

[5- Base de donnes 27](#_Toc80698704)

[VII. Security Par Symfony 28](#_Toc80698705)

[VIII. Le type de Security différent 32](#_Toc80698706)

[Authentication & Firewalls 32](#_Toc80698707)

[Denying Access, Roles and other Authorization 33](#_Toc80698708)

[IX. Partie Front (HTML/CSS) 35](#_Toc80698709)

[X. L’amélioration à faire 36](#_Toc80698710)

[XI. Conclusion 36](#_Toc80698711)

# Introduction de projet

Ce projet permet de insère la voiture, le marque et le modèle par le client qu’il va inscrite déjà pour être connecté sur le site

À la suite,

Le site se permette de mettre l’information des voitures avec une avis,

On peut voir les autres voitures en ligne avec son avis aussi.

L’utilisateur peut faire :

1. **Registration**
2. **Login**
3. **Allez ou page activité pour ajouter :**
4. **Marque**
5. **Modèle**
6. **Voiteur**
7. **Allez ou page Avis pour :**
   1. **Ajouter des avis**
   2. **Modifie des avis**
   3. **Supprimer des avis**
   4. **Allez voir un avis**
8. **Allez ou page galerie pour :**
   1. **Voir table de voitures**
   2. **Voir un modèle information**
9. **Logout.**

# Les Application&Langues Utiliser

* HTML : HTML (HyperTextMarkupLanguage) est le format de données conçu pour représenter les pages web donc C'est un langage de balisage permettant d'écrire de l'hypertexte, HTML vous permet de formater le contenu des pages, y compris les ressources multimédias telles que les images, les sons, les vidéos, les formulaires de saisie et les programmes informatiques.



* CSS : Les feuilles de styles (enanglais"CascadingStyleSheets",abrégéCSS) sont un langage qui permet de gérer la présentation d'une page Web. Le langage va pointer une balise HTML est ainsi lui donner du style comme mettre la couleur rouge sur du texte ou positionner une balise image dans un coin.



* PHP : PHP (HyperTextPreprocessor) est un langage de programmation gratuit, principalement utilisé pour produire des pages Web dynamiques.



* Symfony : Symfony est un ensemble de composants PHP ainsi qu'un framework MVC gratuit écrit en PHP. Il fournit des fonctionnalités flexibles et adaptables qui vous permettent de rendre le développement de sites Web plus facile et plus rapide.

Une image contenant texte, clipart

Description générée automatiquement

* Twig : Twig est un moteur de template pour le langage de programmation PHP, utilisé par défaut par le framework Symfony.



* UIkit : est à l'écoute des manipulations du DOM et initialise, connecte et déconnecte automatiquement les composants lorsqu'ils sont insérés ou supprimés du DOM. De cette façon, il peut facilement être utilisé avec des frameworks JavaScript comme Vue



* Visual Studio Code est un éditeur de code source créé par Microsoft pour Windows, Linux et macOS. Les fonctionnalités incluent la prise en charge du débogage, de la mise en évidence de la syntaxe, de la complétion intelligente du code, des extraits de code, de la refactorisation du code et de Git intégrer.



* **Looping**est un logiciel de modélisation conceptuelle de données qui vous permet d'organiser et de structurer vos données en diagrammes et représentations graphiques ordonnées.

Une image contenant texte, clipart

Description générée automatiquement

* Laragon est un environnement de développement universel portable, isolé, rapide et puissant pour PHP, Node.js, Python, Java, Go, Ruby. Il est rapide, léger, facile à utiliser et facile à étendre.

- Laragon est idéal pour créer et gérer des applications Web modernes.

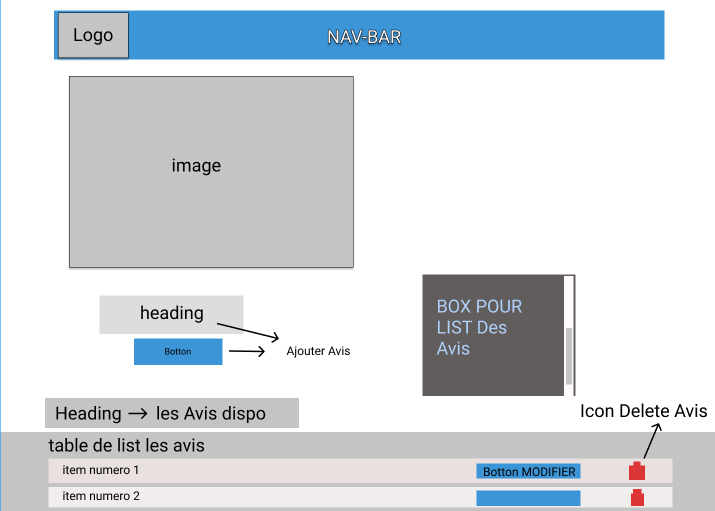


* HeidiSQL : est un outil d'administration gratuit et open source pour MySQL et ses forks, ainsi que Microsoft SQL Server, PostgreSQL et SQLite. Sa base de code a été extraite à l'origine du logiciel MySQL-Front 2.5 d'Ansgar Becker.

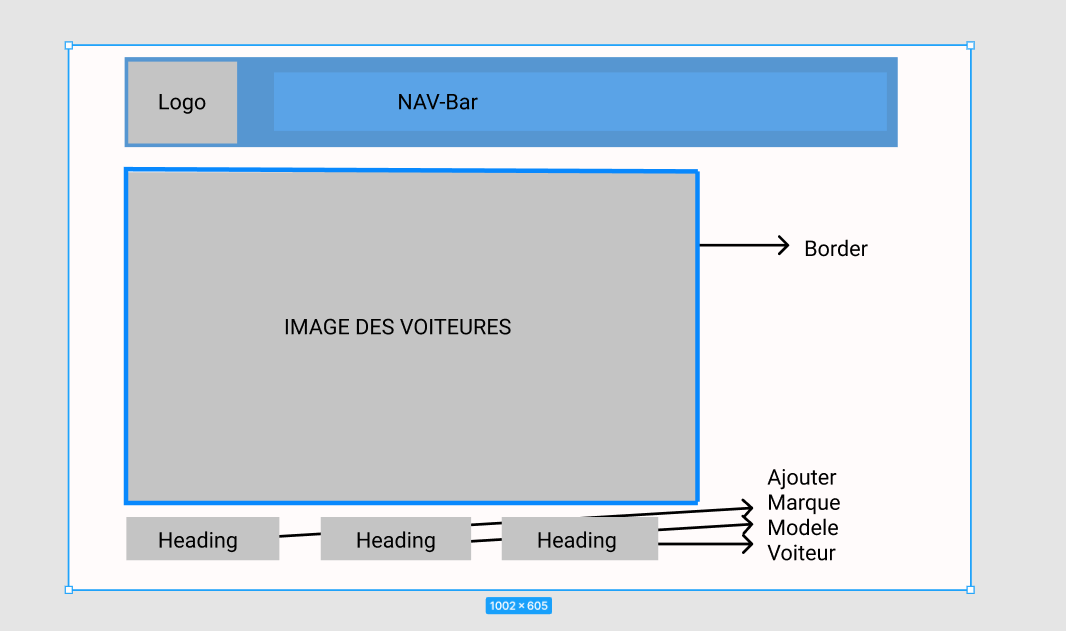


# Maquettage

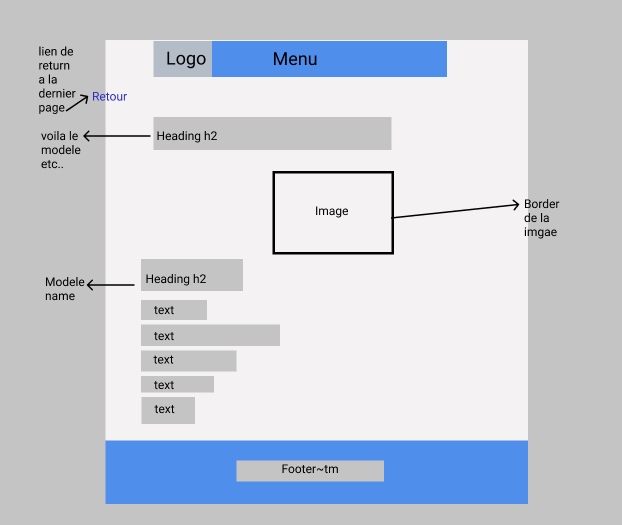
* **Page Avis**



* **Page Activity**

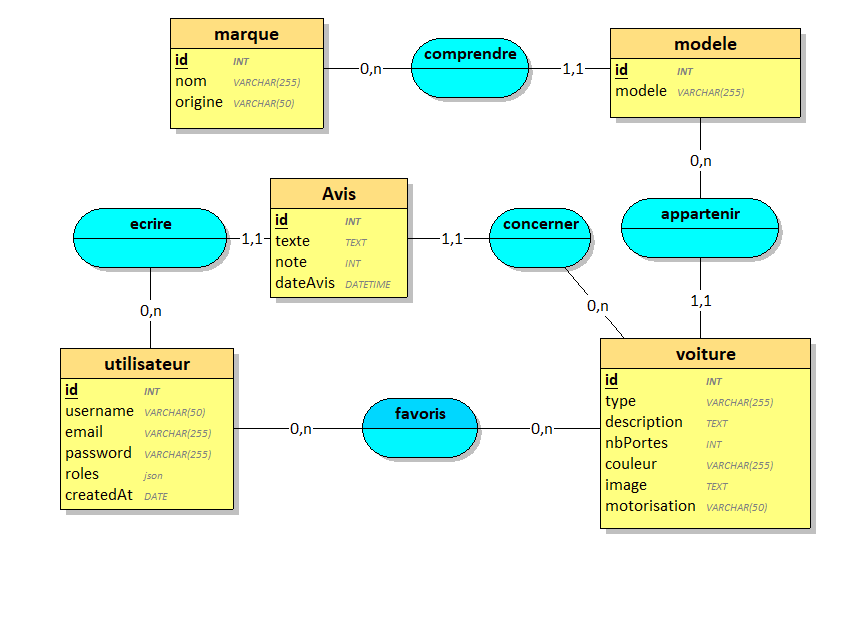
****

* **Page Show Voitures**

****

# MCD

**Ça présent les type des communications entre des valeurs identifient et les déférent valeurs des tableaux et ça clé propre de ASSO qui permet faire des connexions entre eux**.

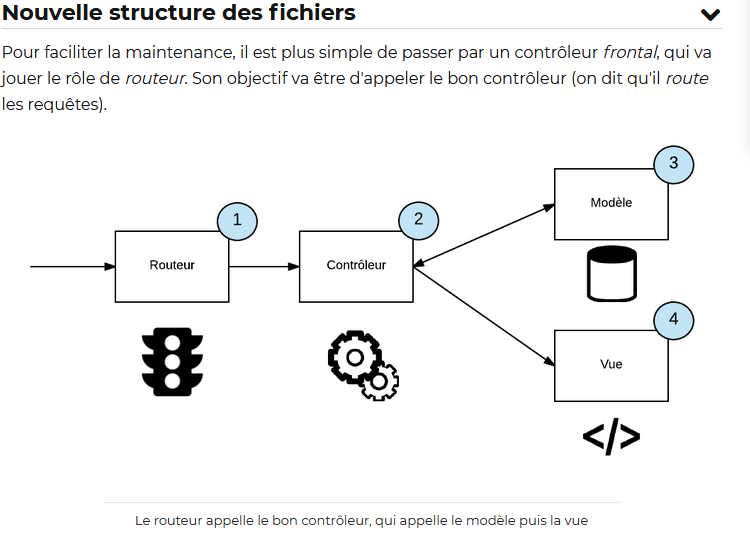


# Modèle Vue Contrôleur (MVC)

**Modèle-vue-contrôleur** ou **MVC** est un motif d'architecture logicielle destiné aux interfaces graphiques lancé en 1978 et très populaire pour les applications web. Le motif est composé de trois types de modules ayant trois responsabilités différentes : les modèles, les vues et les contrôleurs.

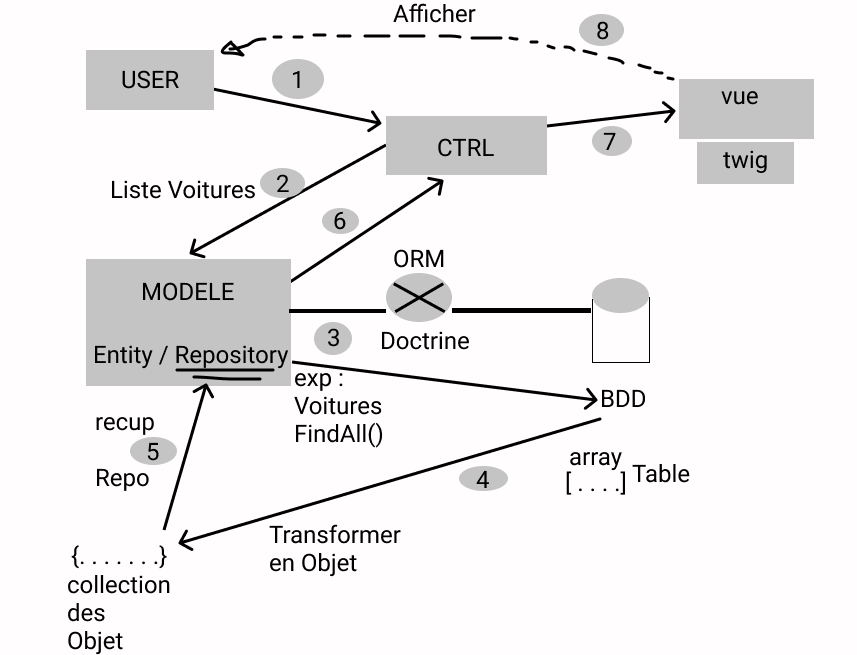
* Un modèle (Model) contient les données à afficher.
* Une vue (View) contient la présentation de l'interface graphique.
* Un contrôleur (Controller) contient la logique concernant les actions effectuées par l'utilisateur.

En utilisant des fonctions de classe, Chaque action des Controller d’une table ou (plusieurs tables) avait un Controller avec des index de chaque un.



Symfony utilise la méthode de MVP (MVC mais pas d’échange entre le modelé et vu) pour crée le projet alors

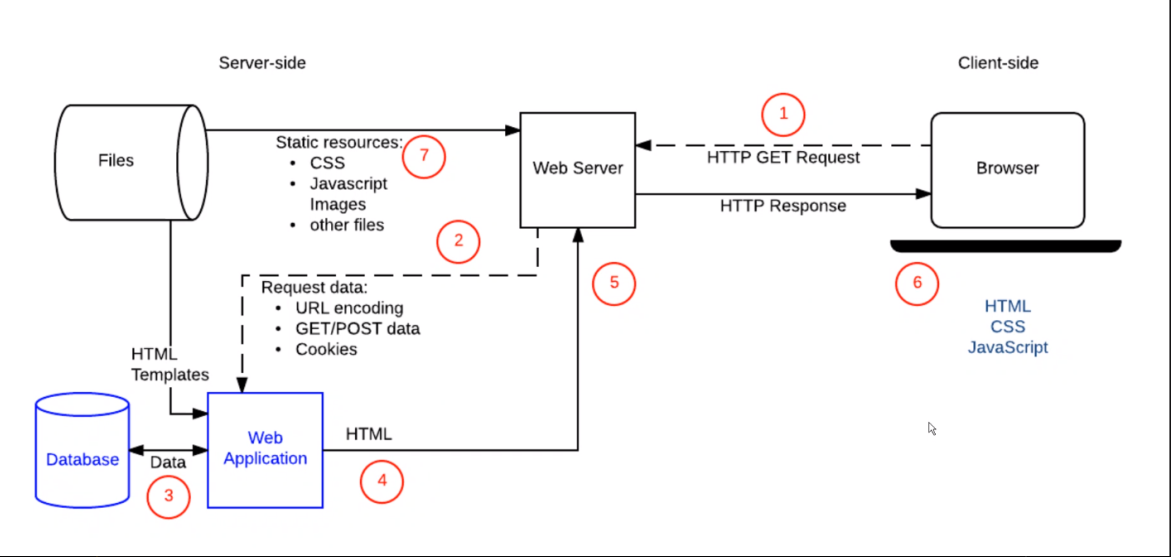
Voici un Autre Example qui est plus détaillé :



L’User demande une marque de voiture pour Example

1. L’user demande requête http à la Controller
2. Controller demande une liste de voitures à Modèle
3. Modèle demande aller voir toute la voiture pour Example en BDD Table de base de donnes
4. Array de Table transforme en collection des objets
5. Le Repository récup la collection
6. Modèle renvoie le à Controller
7. Le Controller envoie le à vue [Twig] en fait un foreach pour voitures pour recouper voiture.marque ( le marque demandée )
8. La vue affiche le résultat à USER.

Comment ça se passe en url voici un Example pour le schéma :



1. Envoi de la requête http
2. Le Web server demande un request data
3. Echange entre l’entité et le base de donnes
4. Envoie le résultat de la collection récupéré par Repository au Controller
5. Resu le résultat a la web server
6. Retourner un résponse http
7. Charger le style-Sheets de la page web et envoie le au web server
8. Afficher le style avec le reponse http

Donc on va utiliser ce type de schéma par default de Symfony

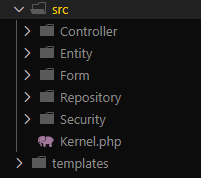
Ou moment de faire d’action le Controller viens de faire le routage au chemin d’accès de chaque fonction qui va présenter avec cette manier en haut dans le navigateur (URL).

Et voilà c’est le mvc par default :

Entity, repository remplace le (MODELE)

Controller c’est Le Controller

Templates remplace le (VEU)



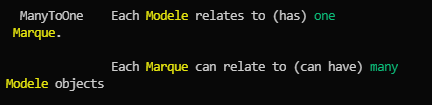
# Mise en place des entités (tables de la base de données)

**CONSOLE INTEGREE A SYMFONY:**



## 1- Relation (ASSO) de table

**Dans la classe Marque sont reliés OneToMany $modeles, et dans classe Modele est relié ManyToOne $marque =>**



**Dans la classe Modele sont reliés OneToMany $voiteures, et dans classe Voiteur est relié ManyToOne $modele =>**



**Dans la classe Voiture sont reliés OneToMany $Avis, et dans class Avis est relié ManyToOne $voiture =>**



**Dans la classe User sont reliés OneToMany $AvisDeUtilisatures, et dans class Avis est relié ManyToOne $user =>**



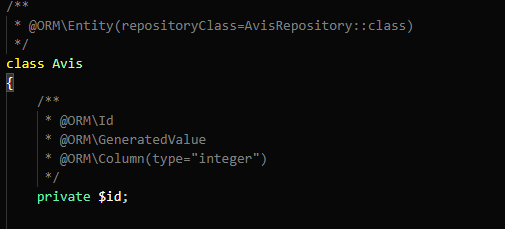
**Dans la classe User sont reliés ManyToMany $voitures, et dans class Voiture est relié ManyToMany $users**

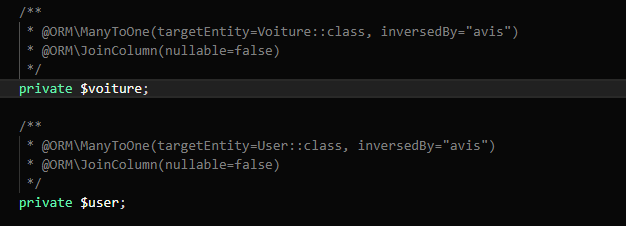
**Je vais appeler le Relation $listFavoris =>**



## Détail d'une entité Symfony

Example Class avis :





Avec la commande php/bin make:entity : il crée le entité avec sa repository

Comme est-il présenté en haut chaque classe avec sa Respositry,

Et sa id par default crée par Symfony.

Chaque classe avait plusieurs entités qu’ont créé par nous comme on veut.

Avant passe au Controller ici en petite exemple (( private $valeur ))

Les propriétés sont privées pour qu'il soit impossible d'y accéder directement en dehors de la classe correspondante.

Et avec la valeur de la classe il n’y a pas besoin d’être public alors c’est créé par default en mode service private par le command de Symfony,

Private $user c’est une valeur ManyToOne de la classe avis comme on a expliqué avant et cette valeur ne peut pas être Nullable (null = vide).

## 3- Controller

C’est importé que le $valeur est écrite en Minuscule

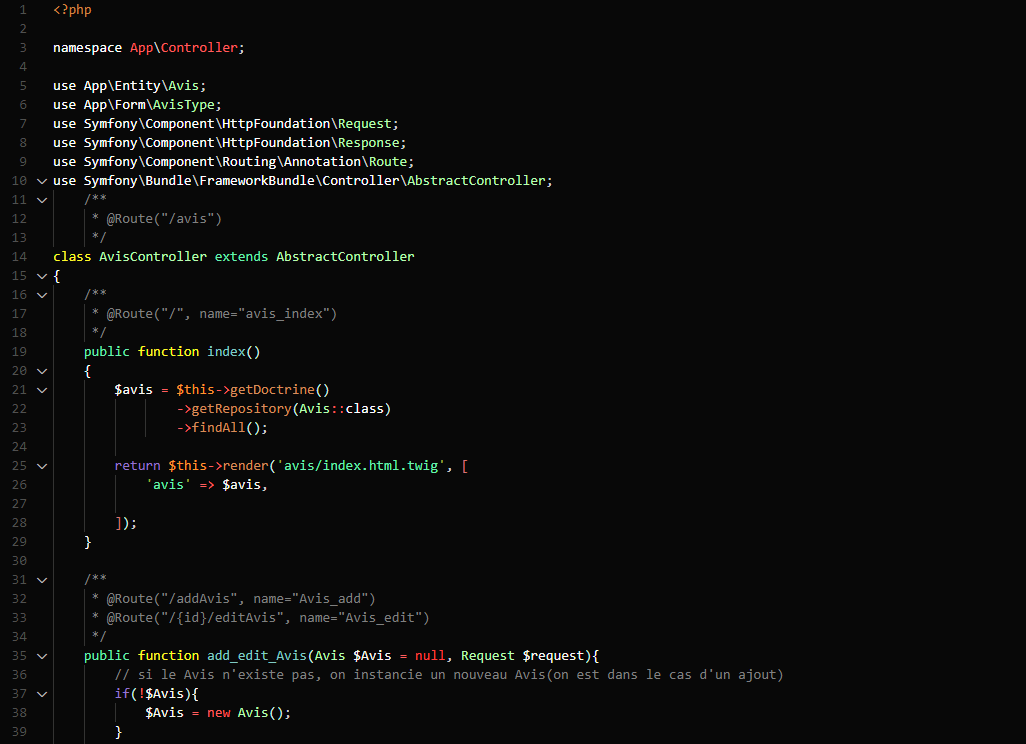
Et le Nom de Controller avait la première lettre en Majuscule ,

Et la première lettre du deuxième mot c’est aussi en Majuscule

Example AvisController .

Même le Nom de Controller avait le 1er lettre Majuscule

Avec le reste de les lettres minuscule



*  Ligne 1 c’est le commande PHP pour dire: de ici la code est-il écrite en PHP.
*  Ligne 3 c’est le NameSpace: dans la définition la plus large, les espaces de noms sont un moyen d'encapsuler des éléments. Cela peut être considéré comme un concept abstrait dans de nombreux endroits.

Par exemple avec n'importe quel système d'exploitation, les répertoires servent à regrouper les fichiers liés et servent d'espace de noms pour les fichiers qu'ils contiennent. A titre d'exemple concret, le fichier foo.txt peut exister à la fois dans le répertoire /home/greg et dans /home/other,

Et ici le NameSpace est-il App/Controller.

*  Ligne 5,6 use est un fonction native de Symfony qui dit utilise: le entité avis et forme de AvisType par utilisent App/Entity/Avis, App/Form/AvisType.
*  Ligne 7,8,9 use est un fonction native de Symfony qui dit utilise: composent de Symfony HttpFondation: Response, Request, Route.
*  Ligne 10 use est un fonction native de Symfony qui dit utilise: AbstactController qui permet faire le sécurité avec les plusieurs fonction dedans.
* De  Ligne 16 au 18 : Routage, Alors c’est un router au name qui est le chemin d’accès (/avis).
* De  Ligne 19 : Fonction index de AvisController.
* De  Ligne 21 au 26 : un variable avis qu’est égal à getdoctrine() donc aller utiliser le service Symfony de AbstractController qui permet de manipuler le doctrine avec les fonction getdoctrine() et donne-moi tout le doctrine en suite on passe d’une autre méthode de service AbstractController qui avait une méthode getRespotory() d’après donne-moi le respotory de la classe avis et trouver tous les entité dedans, ensuit en utilisent la Route au Template (/avis/) qui avait un (( indix.html.twig )) donc le route sera (( avis/index.html.twig )),

Et après en déclare la route avis qui est égal à avis de l’entité de la classe avis que on a déjà déclaré.

* De  Ligne 35 : Fonction add\_edit\_avis qui addition les avis et le modifie.

Un autre example des fonctions utiliser pour crée le site :



*  Ligne 42 au 43: (name= /addMarque) c’est le Manier de trouver le fonction par le navigateur au URL donc c’est fait une partie de routage,
* Marque\_add c’est le Manier de trouver quelle fonction est-il demande par le navigateur et ici il existe deux :

1. C’est Marque\_add
2. Est Marque\_edit

Alors c’est au moment un table va au formulaire qui permet savoir :

* Si on ajoute le marque Marque\_add va être appelée
* Si on modifie la marque on utilise Marque\_edit,

Et bien sûr au moment on utilise la modification on a besoin de connaitre le quelle marque aux bases de donne qu’on parle

Ici c’est le job de les ID qui sont créé par default en utilisent comme on a le code en haut au routage pour savoir le quelle entité on parle.

Donc (( /{id}/editMarque )) qui est-il affiché au URL.

* De  Ligne 46 : Fonction add\_edit\_Marque c’est le nom de fonction qui addition les marque et le modifie,
* On utilise le classe Marque et en le mettre une valeur $marque au cas de mettre la donnée dedans et est-il en Null donc est-il vide
* Et on utilise ici le méthode Request qui vient du service de AbstractController qui est créé par Symfony donc on a créé la valeur $request Null (vide).
* De  Ligne 48 au 50 : au langue PHP c’est une méthode (( if )) qui permet dire si et seulement si la condition est (( vrai )) ou (( false )),
* Ça débande l’utilisation on faire une action
* Ici au cette example si le Maque n’existe pas il faut crée un ,

En utilise la classe marque pour les donnes d’entité marque et le mettre aux valeurs.

* De  Ligne 52 au 54 : il faut créer un Marque au préalable avec la fonction de service Symfony en utilisant le commande : (( php bin/console make:form )) qui permet faire cette forme , $form : est-il une valeur crée par nos pours mettre les donnes dedans $this et $this est une référence de l’objet courant $this objet, depuis on le prends et créions-nous le form avec creat forme qui avait deux paramètres :

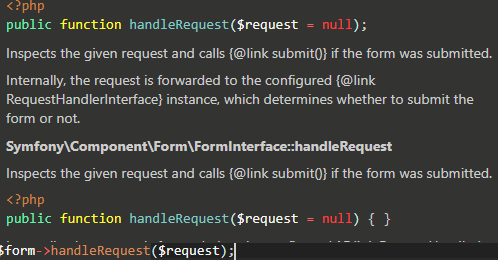
1- une classe de MaqueType en utilisant en commande de Symfony

* + php bin/console make:form .

Ma forme de Marque qui J’ai déjà crée est appelle MarqueType et avait-il des donnes avec de chaque type le donne par default.

2- le valeur de marque et est-il appelé $Marque

Et line 54 :



En crée un valeur $form qui est Null (vide) , et mitons le $request dedans

en utilisant des donnes de la fonction handleRequest ,

qui avait un paramètre $request dedans ,

Alors handleRequest permit voir et inspecter le lien de submit en vois si la form est-il submit et mettre le réponse ( ((true)) ou ((false)) => vrai ou faux ) au paramètre de handleRequest.

Depuis on prend la donne de ce paramètre et mitons le dans notre valeur $request ,

comme ça on voit si notre donne $form est-il submit au pas ,

Bien sûr le handleRequest est une fonction écrite déjà par le service Symfony dans AbstractController et utilise ici.

* De  Ligne 56 au 66 : un condition simple If qui permit de dire si on soumet le formulaire et que le forme est-il valide (( isSubmitted ))
* isSubmitted : est une fonction de Symfony qui permit voir si est submit alors le résultat est true (vrai) et avec la donne de forme $form qui on avait déjà ,
* juste avant le résultat de (( HandleSubmit )) il compare et si la condition est vrai ((si est-il submit et valide )) il passe à l’action
* comme ici : ( true &&true ) : && permit dire ( And )

alors c’est ( et ) en français ,

* en suit on a déjà en condition vrai , Il nos manque le résultat de la deuxième condition isValid et isValid est une fonction de Symfony de AbstactController qui vérifie si le forme est valide avec le même type entre l’entité et SQL (base de donne) ,
* Depuis lui va donner une réponse vraie ou false , valide ou non pas valide
* et bien sûr ici c’est true alors ça va nous donner deux condition ( vrai && vrai )

D’après il y a un crocher qui est le dépôt de notre if

donc si le deux sont vrai en applique la condition .

* De  Ligne 58 : on récupère les données du formulaire avec la fonction getData() et on le mettre à la variable $Marque

//on ajoute le nouveau Marque

            $entityManager = $this->getDoctrine()->getManager();

            $entityManager->persist($Marque);

            $entityManager->flush();

Une image contenant texte

Description générée automatiquement

* De  Ligne 60 au 62 : on ajoute le nouveau Marque en manipulation de doctrine :

1. getDoctrine() une fonction qui permet faire un shortcuts (chemin d’accès court) au service de régénération doctrine
2. et getManager() permet de prends une doctrine
3. Et après persist permet crée un objet
4. Depuis flush permet envoie l’objet en base de donnes.

Donc si on veut modifie qqe chose seulement en utilisent un formulaire (une fonction pour sauf modifie) on pas besoin de faire un deuxième persist.

* Comme ça notre nouvelle marque est bien ajoutée aux bases de donnes
* après que on click valider dans le formulaire tous ces opérations sont appliquées depuis bien sûr on a besoin de faire redirect le client dans un page home

ou un page accueil etc...

* Donc le  Ligne 64 : $this->redirectToRoute() avec cette fonction qui permet faire le redirect de client au nom de la Route existé entre (. /...route…/.) et bien sûr on mit le nom entre gimer comme ("home")

voilà la fin de notre if .

* De  Ligne 66 au 70 : Notre fonction doit return qqc

car toute la fonction doit retourner qqc au moment d’appeler la fonction ,

Et ici on rendre le client:

* + soit  ajouter le nouveau marque
  + soit  on le modifie.

Donc ce lien de redirection au Template qui est-il la vue de formulaire

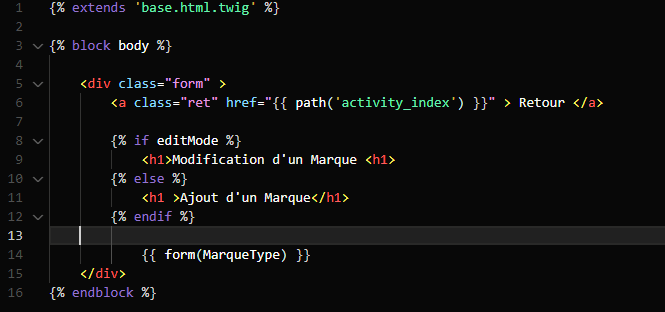
et on prend les marque concerner avec ,



Ça ID en avance est est-il Null (vide -> existe) ou pas.

* Et voilà dans ce cas c’est au Template qui existe au activity.
* la page c’est (( add\_edit\_Marque.html.twig )) .

## Template code



*  Ligne 1: on extends : (mis un page dans un autre page donc le page de mère avait un page dedans qui est le partie body)

 on extends  de page : (( base.html.twig ))

*  Ligne 2: le dépôt de la block
*  Ligne 5: un div -> div: c’est un manier de mettre un block vide (quarré)

et peut-il avait un id ou un classe

donc le défection sera un élément de [HTML](https://fr.wikipedia.org/wiki/Hypertext_Markup_Language) qui crée un niveau logique.

*  Ligne 6: le balise (L'élément) HTML <a> avec son attribut href, crée un lien hypertexte vers des pages Web, des fichiers, des adresses e-mail, des emplacements dans autre pages etc...

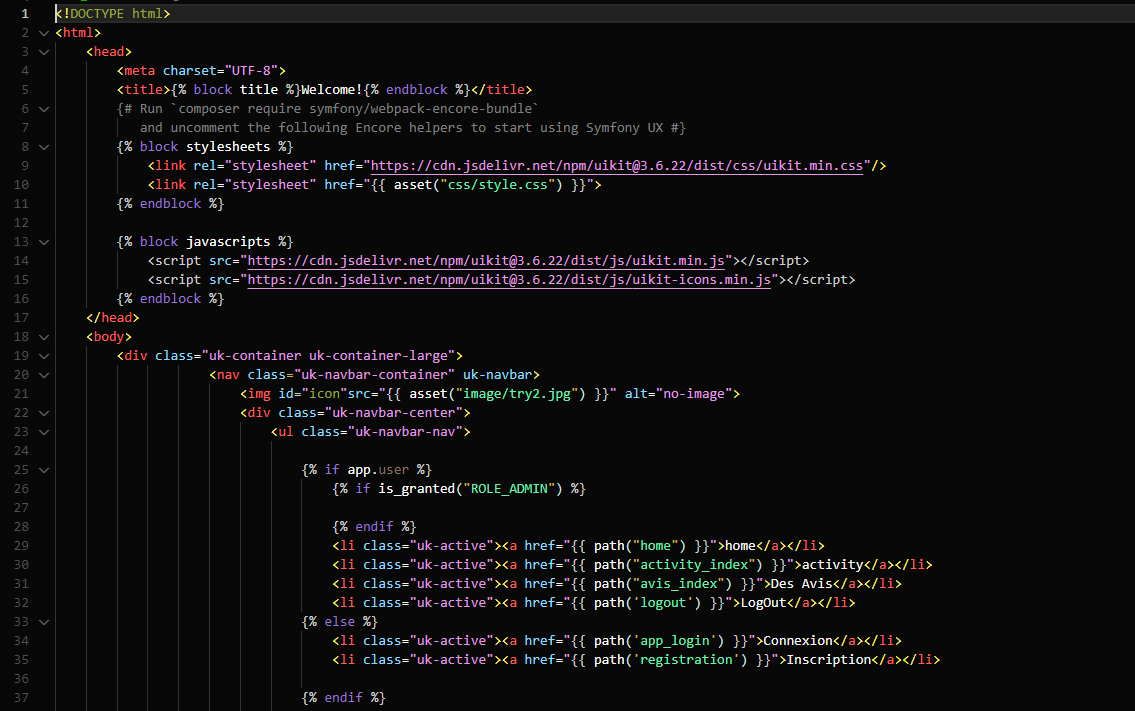
Path() est une méthode utilise pour le page de Twig ,

pour écrire le nom de chemin d’accès.

Class : est un manier de appelle des balises défièrent uniquement ou en plusieurs.

*  Ligne 8: le dépôt de if en langue ((Twig ))
*  Ligne 9: le balise <h1> c’est un header de premier niveau.
*  Ligne 10: un (( else )) de if : veut dire si non.
*  Ligne 12: le fin de if
*  Ligne 14: la place de forme
*  Ligne 15: le fin de la balise div
*  Ligne 16: le fin de la block

On va à la page de base :



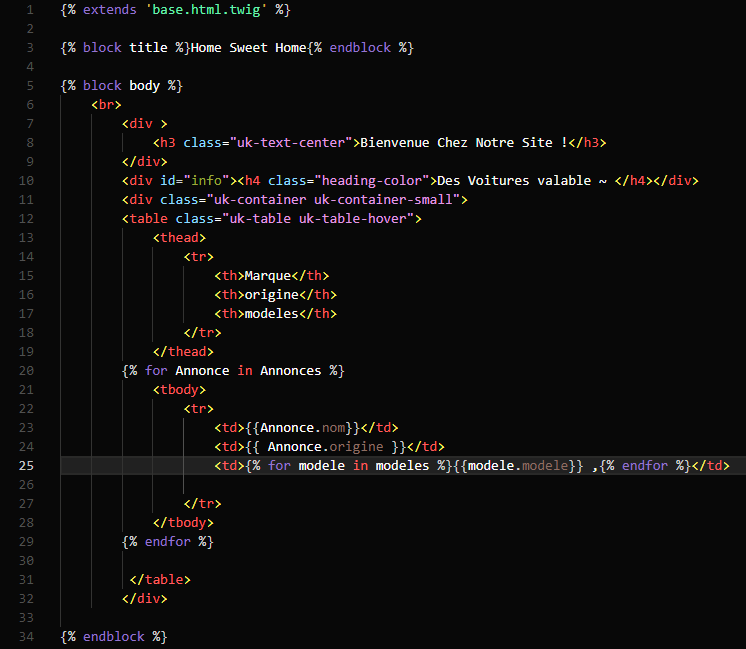
*  Ligne 1: la page est définit en html comme un langue utilisé.
*  Ligne 2: la balise html commence ici.
*  Ligne 3: L'élément <head> est un conteneur de métadonnées (données sur les données) et est placé entre la balise <html> et la balise <body>. ... Les métadonnées définissent généralement le titre du document, le jeu de caractères, les styles, les scripts et d'autres méta-informations.
*  Ligne 4: on utilise le meta UTF-8 ici
*  Ligne 5: définition de titre comme -> Welcome!
*  Ligne 8: le dépôt de block de css de style
*  Ligne 9: Uikit c’est un style utiliser par nous qui est écrite par le server Uikit
*  Ligne 10: notre ficher de style
*  Ligne 11: le fin de block de css de style
*  Ligne 13 et 16: le dépôt et la fin de block de Javascript
* Appliquée par nous écrit par Uikit.
*  Ligne 14 et 15: le liens vers le script de Uikit.
*  Ligne 17: la fin de balise head.
*  Ligne 18: le dépôt de balise body.
*  Ligne 19: le dépôt de balise div qui avait un style de Uikit
*  Ligne 20: le dépôt de balise de navigation donc un liste de navigation est commencer
* Styler par le Uikit
*  Ligne 21: un balise image qui est présenter dans le nav.
*  Ligne 22: deuxième balise de div qui est centre par le style de Uikit.
*  Ligne 23: le dépôt de balise UL <ul> permit faire un liste.
*  Ligne 25 et 26: condition de if qui dit si le user (utilisateur) est-il un admin ici le condition sera a rien

car dans mon site le admin et l’utilisateur normal, font qu’ils veulent car c’est un site de avis a la fin J’ai mis le condition pour savoir qu’il existe ce type de chose dans le codage on peut mettre le site sécuriser et garantie pour l’admin uniquement.

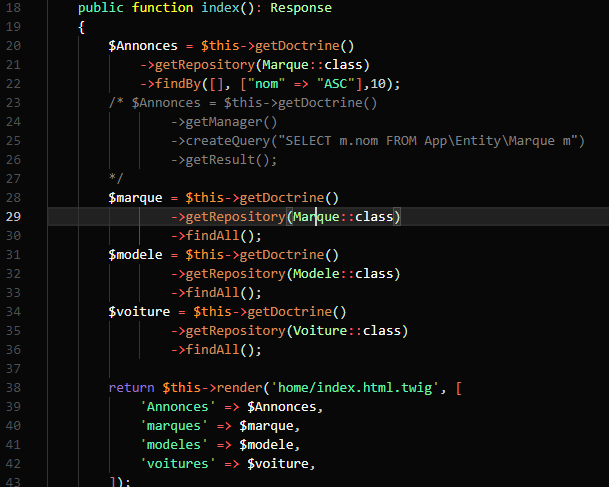
*  Ligne 29: le premier élément de la liste <ul>.
* Chaque élément va avoir un lien bien sur car c’est un navigateur.
*  Ligne 29 a 35: ce sont les élément de la liste.
*  Ligne 35 a 40: on ferme tous le balise utilise.

Et voilà après cette page je veux vous montrez le page de index c’est un Template aussi mais pour afficher les donnes de la base de donnes.

Ex :



*  ce page est appeler (index.html.twig ) est-il dans un dossier home
* Le dossier home avait sa propre Controller qui avait un fonction index qui affiche le 10 dernière voiture ajouter. En utilisant la valeur $Annonces

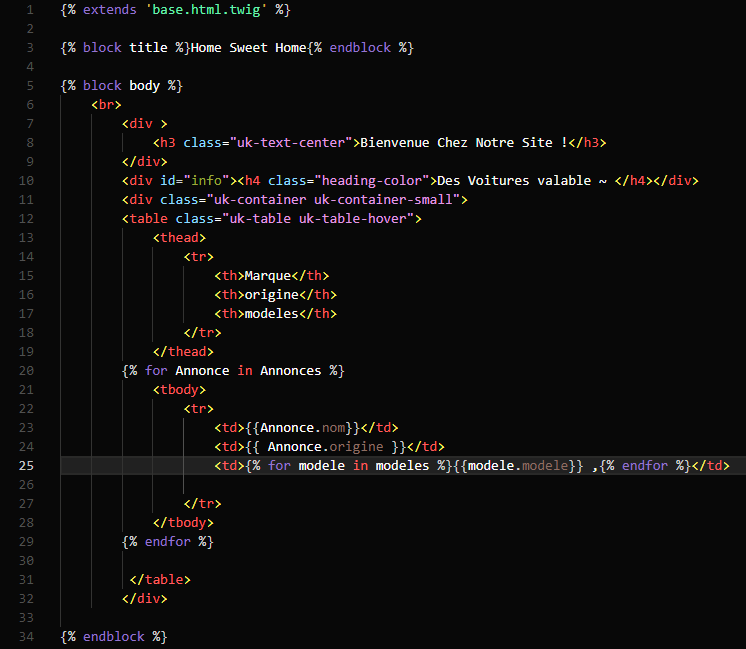


En utilisant les trois tables on peut récupère toutes les infos de base de donnes.

Alors je vais présenter le page index en détail,

Avec la connexion entre le Controller et de la vue

Allons-y : ~ ~



*  Ligne 1 on extends (mis un page dans un autre page donc le page de mère avait un page qui est dans le body)

on extends la page de: base.html.twig.

*  Ligne 3: titre.
*  Ligne 5: dépôt body.
*  Ligne 6: <br> c’est un balise de une line vide
*  Ligne 7 et 9: dépôt et fin la balise div.
*  Ligne 10: div qui contient un header niveau 4 ,

le div est style par un id appelle info

*  Ligne 11: div qui est style comme un container (div mère) Par Uikit.
*  Ligne 12: dépôt de balise table stylée par le Uikit.
*  de Ligne 13 au 19: dépôt balise thead qui est le tète de table.
*  Ligne 14 et 18: dépôt balise tr qui permet défini la colonne de table.
*  Ligne 20: un for (boucle) qui permet boucles sur les id de valeur $Annonces

Et la valeur vient de HomeController au page 16 en haut

qui est-il dans le index ,

et permet avoir un contenu ( des donnes-> dans mon code sont des marques) les 10 dernières voitures en ordre Alphapitique (ASC) et return ces donnes dans un valeur appelle Annonces été

on utilise Annonces comme un valeur max de la boucle et on boucle là-dessus pour obtenir une valeur que on a créé Annonce <- (( singulaire))

ici dans cet exemple : de numéro 1 de 10

*  Ligne 21 et 28: dépôt balise tbody qui permit dire que c’est le body de table
*  Ligne 23 et 24 et 25: dépôt balise td qui permit dire la colonne de body.

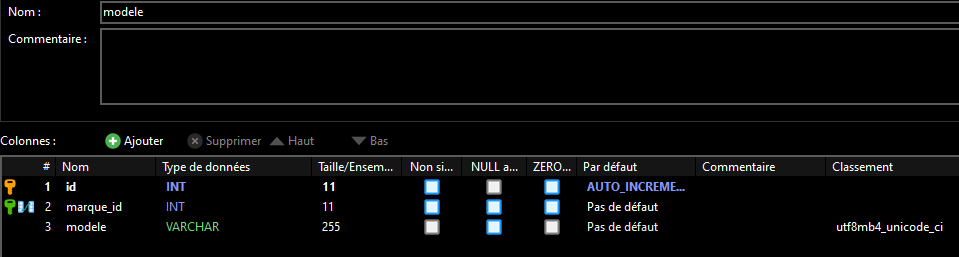
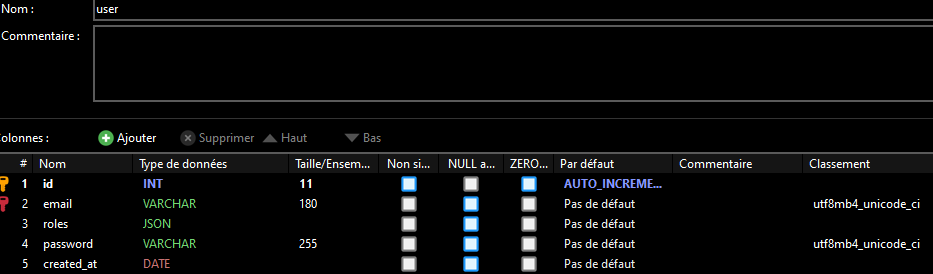
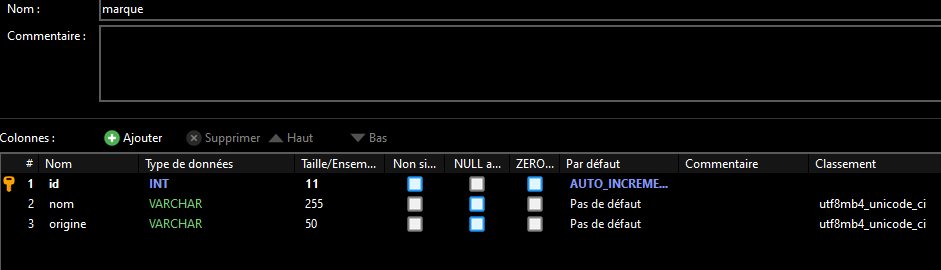
 exemple: Ligne 23 (( Annonce.nom)) -> nom c’est le nom qui est défini dans le entité de la classe marque.

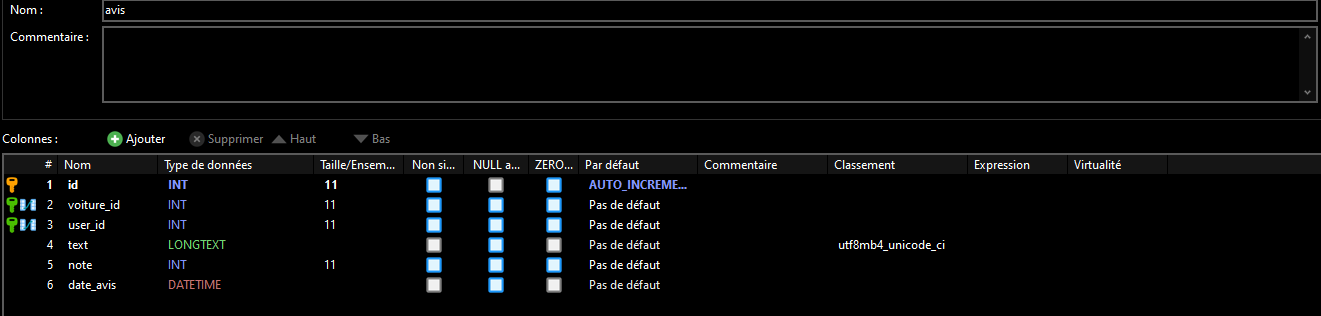
*  Ligne 25: deuxième boucle sur les modeles et voilà comme Annonces modèles vient la fin de la entité de la classe modèle et (( modele.modele )) permit dire le modèle de la voiture et ici le modèle des 10 dernier voiture ajouter dans les ordre Alphapitique (ASC).
* Ligne 29 et 34 : la fin de for, la fin de block body.

## 5- Base de donnes

La 2 capture suivante présente HeidiSQL qui est notre base des données de cote serveur :

* Les Paramètres

******Une image contenant texte, capture d’écran, écran

Description générée automatiquement**

# Security Par Symfony

Comme on a expliqué au parti de Controller , Symfony permet nous faire une sécurité par défaut avec l’utilisation de la classe UserInterface qui avait de fonction par exemple :

Une image contenant texte

Description générée automatiquement

par exemple :

La fonction GetPassword ()

Une image contenant texte

Description générée automatiquement

Cette fonction permet d’authenticité la utilisateur et la mot de passe dois être hachée et en authenticité la mot de passe, la mot de passe va être comparé entre le valeur qui est prise avant le hachage et le valeur d’après le hachage et ce méthode existe depuis Symfony 5,3 et on peut le implémenté de (( PasswordAuthenticatedUserInterface )) comme ici dans ce Example :

Une image contenant texte

Description générée automatiquement

Il y a d’autre type de Security comme getUsername()

Une image contenant texte

Description générée automatiquement

Qui permet vérifie le username est-il valide ou pas !

Et il y a aussi getRoles()

Une image contenant texte

Description générée automatiquement

Cette une fonction qui permet de vérifie si le rôle est le même rôle qui est stocke dans la base de donnes quand on a créé l’objet.

Et c’est utiliser ici :

Une image contenant texte

Description générée automatiquement

Ce fonction permet faire le registration les utilisateur et valide le objet dedans comme: username,password,rôle et comme on a vu Symfony utiliser la Security Controller qui fait par un commande php bin/console make:Auth qui permet fait le Altération et dans ce classe Auth il y a une implémente qui est fait par défaut et une classe de Controller qui est appeler SecurityController et dans ce classe on utilise UserPasswordEncoderInterface que on a parlé et bien sûr ce interface qui on a utilisée c’est dans le component 

# Le type de Security différent

## Authentication & Firewalls

Une image contenant texte

Description générée automatiquement

Un firewall : est votre système d’authentification : la configuration ci-dessous définit comment vos utilisateurs seront capable s’authentifier (par exemple, formulaire de connexion, jeton apl, etc…)

Un seul pare-feu est actif sur chaque requête : Symfony utilise la clé de motif pour trouver la première correspondance (vous peut également correspondre par hôte ou d’autres choses).

Le pare-feu de développement est vraiment un faux pare-feu : il s’assure que vous ne bloquez pas accidentellement les outils de développement de Symfony – qui vivent sous des Url comme / \_profiler et wdt

Toutes les URL réelles sont gérées par le pare-feu principal (la clé de modelé N0 signifie qu’elle correspond à toutes les URL). Un pare-feu peut avoir plusieurs modes d’authentification. C’est-à-dire non connecté) lorsqu’il visite votre site Web pour la première fois.

En fait, si vous allez à la page d’acceuil maintenant, vous y aures accès et vous verrez que vous êtes << authentifie >> comme anon. Le pare-feu a vérifié qu’il ne connait pas votre identité, et donc vous êtes anonyme

Cela signifie que toute peut avoir un jeton anonyme pour accéder à un formulaire de connexion.

### Limiting Login Attempts

Une image contenant texte

Description générée automatiquement

Permet de mettre un numéro fix pour le numéro de attempts(en minute ) que on peut rester connecter comme un anymous.

## Denying Access, Roles and other Authorization

Les utilisateurs peuvent désormais se connecter à votre application à l'aide de votre formulaire de connexion. Génial ! Maintenant, vous devez apprendre à refuser l'accès et à utiliser l'objet Utilisateur. C'est ce qu'on appelle l'autorisation, et son travail consiste à décider si un utilisateur peut accéder à une ressource (une URL, un objet modèle, un appel de méthode, …).

Le processus d'autorisation a deux aspects différents :

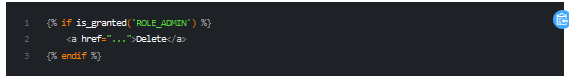
L'utilisateur reçoit un ensemble spécifique de rôles lors de la connexion (par exemple, ROLE\_ADMIN).

Vous ajoutez du code pour qu'une ressource (par exemple, une URL, un contrôleur) nécessite un "attribut" spécifique (le plus souvent un rôle comme ROLE\_ADMIN) pour être accessible.

#### Securing URL patterns (access\_control)

#### Le moyen le plus simple de sécuriser une partie de votre application consiste à sécuriser l'intégralité d'un modèle d'URL dans security.yaml. Par exemple, pour exiger ROLE\_ADMIN pour toutes les URL commençant par /admin, vous pouvez définir autant de modèles d'URL que nécessaire - chacun est une expression régulière

#### Access Control in Templates



J’ai utilisé ce type de sécurité dans mon Template base, Et sa permet dire que le rôle-admin

Et sauf le rôle-admin dans nous base de donne qui peut et d’après on mis de condition

Dans mon site le home on peut le voir en revanche tout l’user peut voir le reste ma site le reste de page dans le menu.

Et ici que le admin peut supprime qqe chose, et s’il me reste de temps je fais applique cette condition a supprimassions de avis .

# Partie Front (HTML/CSS)

Une image contenant texte

Description générée automatiquementUne image contenant texte

Description générée automatiquement

* Le Tradiction de Mot Englais de css avec sa fonction :

Html : c’est permet de de dire tout la tail de la page de site utiliser.

#icon : c’est un idefetant ( ID) unique qui permet spicifier un parti selctione ou html et appliquer avec le css et je le appller le #icon car sa definie un icon

Box-shadow : ca permet dire que le block ( box ) avait un border de shadow qui utilise un color , width : c’est le longeur de la block ici au code est ( longeur de icon )

Hight : hauteur de la block .

#info : id de la block info( block de l’information ) ,

Margin-left : le espace droit de l’extrieur de la balise ( extrieur droit de la balise en genral ), margin-bottom : le espace de bas de l’extrieur de la block.

.heading-color : c’est un classe qui permet spicifier un parti selctione de la classe

Color : le color de la text

Font-size : le font ( tail de carictaire ) ,

Large,medium,smal : grande,moyenne, petit .

font-family : le type de font utiliser serif , robot etc…

Background : le color de l’arrier plan de la blaise utiliseé.

Margin-top : le espace en haut de l’extrieur de la balise

Margin-right : le espace de gauch de l’extrieur de la balise

Display : le type de affichage ,

flex : mettre le page ( le conteneur ou le body ou le div de mere de la page ) en mood de flexer et mode flex permette dire rangment de prmier item jusqua le dernier item qui sont ranger( flexer) de un manier qui fait block par block par default ,

Et on peut le changer le type de flex avec le enfant de ce continer avec un autre fonction.

Par-exp :

.continer : c’est un classe qui avait le display flex donc

.flex : classe appller flex qui avait un margin

margin: le espace de l’extrieur de la balise et un display flex de parant

flex-direction : qui spisife le type de flex qu’est ici en cloumn .

Border-radius : le coin de la border ( carrie ) en transform le coin de carrie en reqtangle

Position : c’est le place de la balise que on utilise et le fonction postion utilise le

fonctionment de balise mere et enfant comme flex mais ici au code cest le postion

ca faut dire si le balise-mere a changer le place le enfant pouge avec le mere

et voila le expliqation de postion : relativ .

# L’amélioration à faire

Une page de list-favoris de voiture

Utiliser les addFlash(), qui sont des méthodes qui permet de créer un message flash qui informera l'utilisateur de la réussite ou non d'une action côté Controller.

Liste d’achat des voitures.

# Conclusion

Ces sept mois de cette formation et le moment de mon étudie ici chez Elan était passée très vite vraiment et c’est amusant, Je suis passé par des moments de doutes et des moments d’euphorie quand une fonctionnalité s’affichait comme prévu. Il reste beaucoup de choses à faire et à améliorer. Ce projet et cette formation ont validés que je veux continuer dans ce domaine. Et bien sûr découvrir en plus et aussi je veux apprendre plus en plus tous les jours, en mettant en application ses connaissances et les confronter aux exigences de la réalité du terrain. Se tromper et se relever, mais ne pas se décourager... Je suis heureux d’avoir eu la possibilité d‘avoir accès à cette formation, qui m’aura fait grandir professionnellement. Je mesure ma chance d’avoir pu faire le choix de la reconversion vers un métier que j’aime et je souhaite démontrer chaque jour. PHP et Symfony ont été très agréable à apprendre mais ma curiosité a été attisé en formation et je suis ravi de voir que les possibilités sont innombrables dans le développement web.