Framework :

* Cadre de travail
* Contient outils => rapidité
* Fonctionnalités testées et approuvées
* Grande Communauté
* Documentation riche

Symfony :

* Ensemble composants
* Facilite la vie
* Plus utilisés
* Adapté à toutes applications

Composant :

* Implémentent fonctionnalités communes

Bundle : fonctionnalité partagée par des devs de symfony

MVC :

Model : accès aux données (BDD) = classe, library etc

Vue : affichage (html, json etc)

Controller : écouter la requête http, de traiter et renvoyer une réponse

Twig :

* Simple apprentissage
* Extensible (filtre, fonction etc)
* Plus concis que PHP natifs
* A peine plus lent
* Syntaxe orientée Templating (structures de contrôles optimisés)

Commentaire Twig :

{# #}

Les filtres :

Formatez la façon dont une variable va s’afficher

Gabarit : {% extends 'base.html.twig' %}

BASE DE DONNEES:

ORM = Doctrine (object relational mapping)

* Classe PHP (=tables)
* Entity = classe = table
* Manager = d’insérer, mettre à jour, manipuler enregistrement
* Repository = sélection de données

qu’est ce qu’un bon commentaire ?

Repository :

$repo = $this->getDoctrine()->getRepository(Ad::class);

        $ads = $repo->findAll();

* Récupérer les données dans l’entity Ad

Condition ternaire :

($genre == 'male' ? 'men/' : 'women/') . $pictureId;

Encoder PASSWORD :

Security.yaml :

encoders:

        App\Entity\User:

            algorithm: bcrypt

private $encoder;

    public function \_\_construct(UserPasswordEncoderInterface $encoder){

        $this->encoder = $encoder;

    }

$hash =$this->encoder->encodePassword($user, 'password');

IMPLEMENTATION DE CLASSE :

* Obligé d’ajouter les functions de la classe implémentée pour pouvoir l’utiliser

PROVIDER : security.yaml

* Récupérer les données des utilisateurs
* providers:
* users\_in\_memory: { memory: ~ }
* in\_database:
* entity:
* class: App\Entity\User
* property: email

COURS bdd : propriétaire