Développement Web
Symfony 7

- Introduction à cette formation
  - Votre formateur … Et Vous
  - Le matériel et logiciels
    - L'environnement de développement VisualStudio Code orienté développement Web.
    - Navigateurs Chrome et Firefox
    - Emploi de WAMP
  - L'organisation – horaires
    - Formation de 4 jours
  - La forme :
    - Un cours très succinct
    - La fabrication d'un site web

- Les liens utiles

- https://symfony.com/doc

- https://grafikart.fr/formations/apprendre-symfony-7

- https://www.youtube.com/watch?v=4t3fNkGwRWo

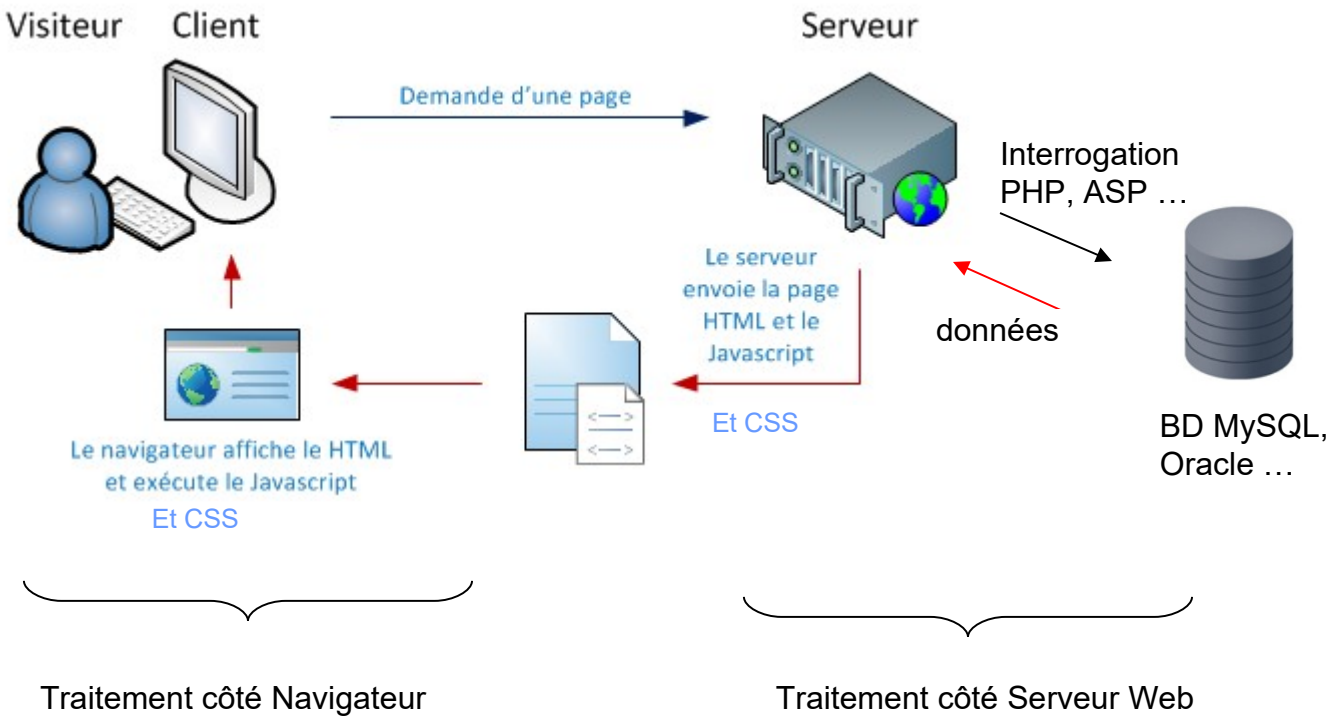
- https://openclassrooms.com/fr/courses/8264046-construisez-un-site-web-a-laide-du-framework-symfony-7/8399947-installez-symfony

- Rappels sur la programmation PHP et Objet
- Introduction au framework Symfony
- Environnement de travail
- Un premier projet

- La communication client/serveur (navigateur/serveur du site Web) se fait en TCP/IP
- Protocoles utilisés
  - http, HyperText Transfert Protocol
  - https, crypté
  - ftp pour télécharger de gros fichiers
  - Imap, smtp pour les e-mails
  - …
- Pour l'affichage d'une page web, deux acteurs principaux
  - Le serveur Web qui fournit les données
  - Le navigateur qui sait les interpréter et afficher




- Traitements serveur et client



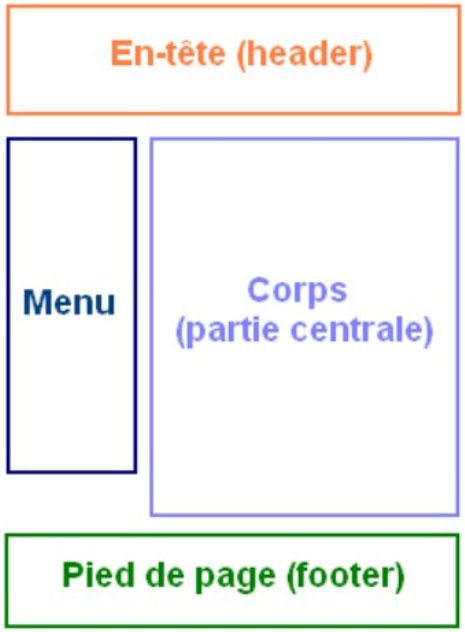
Rappels sur la programmation PHP et Objet

- Insérer du code php dans du code html :
  <?php
  …. Code php
  ?>
  Les balises <? .. ?> et <% .. %> sont équivalentes
- Le code php peut être mis partout dans le html

```
<p>
    Cette page contient du PHP.<br />
    Voici quelques petits tests :
    <?PHP echo "texte issu de PHP"; ?>
</p>
```

- Tout fichier contenant du html et du PHP doit avoir le suffixe .php

- Inclusion de pages
  La problématique à résoudre est un site contenant plusieurs pages dont l'en-tête, menu, pied de pages sont identiques. Seul le corps change.



- La bonne démarche est de définir les parties communes dans des fichiers php, et de mettre des inclusions dans le fichier principal par l'instruction
  <?php include("xxx.php"); ?>

- Chaque instruction se termine par ;
- Les commentaires sont // ou # ou /* .. */
- Affichage pour le navigateur : echo, print, printf
- Les types de variables :
  Booléens, entiers, flottants, chaînes, tableaux, objets
- Toute variable commence par $. Le nom est sensible à la casse.
- Les types de variables sont implicites (pas de déclaration)
  $aa=12.5; $aa est float
  $bb="abcd"; $bb est une string idem $cc='abcd';
- Booléens :
  - Valeur true ou false
  - Un entier = 0 est considéré comme false
  - Une chaîne vide ou "0" est considérée comme false

- Guillemets et apostrophes

```
$mot="Bonjour";
echo ‘valeur : $mot’;                  // affiche  valeur : $mot
echo "valeur : $mot";                  // affiche  valeur : Bonjour
echo "valeur : "  . $mot;  // affiche  valeur : Bonjour
```

- Caractères spéciaux
  \n nelle ligne, \t tabulation, \$ dollar
- Une variable peut être déclarée (apparaître) n’importe où dans le script, mais a une portée de type
  - local
  - global
  - static
- var_dump($var) montre le type et contenu d’une variable

- Une var déclarée hors d'une fonction a une portée GLOBAL et est accessible uniquement en dehors de la fonction

```php
<?php
$x = 5; // global scope

function myTest() {
    // using x inside this function will generate an error
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();

echo "<p>Variable x outside function is: $x</p>";
?>
```

Variable x inside function is:

Variable x outside function is: 5

- A l'inverse, une var déclarée dans une fonction a une portée LOCAL et est accessible uniquement dans la fonction

```php
<?php
function myTest() {
    $x = 5; // local scope
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();

// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
```

Variable x inside function is: 5

Variable x outside function is:

- Le mot clé global rend une var accessible à l'intérieur d'une fonction

```php
<?php
$x = 5;
$y = 10;

function myTest() {
    global $x, $y;
    $y = $x + $y;
}

myTest();  // run function
echo $y; // output the new value for variable $y
?>
```

Result:

15

- PHP stocke les var globales dans un tableau $GLOBALS[]
  Le nom de la var sert d'index. L'accès au tableau est permis

```php
<?php
$x = 5;
$y = 10;

function myTest() {
    $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}

myTest();
echo $y;
?>
```

Result:

15

- Le mot clé static rend une var rémanente (on garde sa place mémoire) dans une fonction

```php
<?php
function myTest() {
    static $x = 0;
    echo $x;
    $x++;
}

myTest();
echo "<br>";
myTest();
echo "<br>";
myTest();
?>
```

Result:

0
1
2

Dans cet exemple la ligne static $x = 0; n'est exécutée qu'une fois
$x reste locale à la fonction

- Déclaration par le mot array
- Peut contenir des éléments de type différents
- La taille varie au cours de son utilisation
- Ex

  $tabcolors = array('red','green','blue');
  $tabcolors[] = 'green'; // ajout d'un élément
- Parcours d'un tableau

```
$i = 0
while(i<count($tab))
        echo $tab[$i++] . "<br>";
```

```
foreach($tab as $elem)
        echo $elem
```

- Gère la notion de tableau associatif : associer un mot clé à chaque valeur

```php
<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

```
Peter is 35 years old.
```

- Boucle sur les valeurs d'un tableau associatif :

```php
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
```

```
Key=Peter, Value=35
Key=Ben, Value=37
Key=Joe, Value=43
```

- print_r($tab) est un moyen rapide d'afficher un tableau (debug)
- list($a,$b,$c) = array("Dog","Cat","Horse");

- Constantes
  Définition : define(nom, valeur);
  Leur portée est globale

```php
<?php
define("GREETING", "Welcome to W3Schools.com!");
echo GREETING;
?>
```

Opérateurs
Regarder le site https://www.w3schools.com/php/

- If .. else

```php
<?php
$t = date("H");

if ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

- Switch

```php
<?php
$favcolor = "red";

switch ($favcolor) {
    case "red":
        echo "Your favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
    case "green":
        echo "Your favorite color is green!";
        break;
    default:
        echo "Your favorite color is neither red, blue, nor green!";
}
?>
```

- Boucles

```php
<?php
$x = 1;

while($x <= 5) {
  echo "The number is: $x <br>";
  $x++;
}
?>
```

```php
<?php
$x = 1;

do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

```php
<?php
for ($x = 0; $x <= 10; $x++) {
  echo "The number is: $x <br>";
}
?>
```

```php
<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
  echo "$value <br>";
}
?>
```

- Les fonctions peuvent prendre des arguments dont il n'est pas besoin de spécifier le type
- Elles peuvent retourner une valeur
- Tout type peut être renvoyé : tableau, entier ...
- Les identificateurs de fonctions sont insensibles à la casse

```php
<?php
function familyName($fname, $year) {
    echo "$fname Refsnes. Born in $year <br>";
}

familyName("Hege","1975");
familyName("Stale","1978");
familyName("Kai Jim","1983");
?>
```

```
Hege Refsnes. Born in 1975
Stale Refsnes. Born in 1978
Kai Jim Refsnes. Born in 1983
```

- Il existe la notion d'argument par défaut

```php
<?php
function setHeight($minheight = 50) {
    echo "The height is : $minheight <br>";
}

setHeight(350);
setHeight();
setHeight(135);
setHeight(80);
?>
```

```
The height is : 350
The height is : 50
The height is : 135
The height is : 80
```

- Valeur de retour : le mot return suffit

```php
<?php
function sum($x, $y) {
    $z = $x + $y;
    return $z;
}

echo "5 + 10 = " . sum(5,10) . "<br>";
echo "7 + 13 = " . sum(7,13) . "<br>";
echo "2 + 4 = " . sum(2,4);
?>
```

```
5 + 10 = 15
7 + 13 = 20
2 + 4 = 6
```

- Depuis PHP 5, PHP est un langage réellement OBJET
- Définition d'une classe

```php
class NomDeLaClasse {
    // Propriétés

    // Métodes
}

$unObjet = new NomDeLaClasse();
```

- Les propriétés
  Doivent commencer par la visibilité public, private, protected

```php
class NomDeLaClasse {
    // Propriétés
    public $nom="default";
    private $id=1;      // accessible uniquement dans la classe
    protected $adresse; // accessible a l'interieur et classes filles
```

- Appel de propriété ou méthode
  - A l'intérieur de la classe $this->

```php
class NomDeLaClasse {
    // Propriétés
    public $nom="default";
    private $id=1;      // accessible uniquement dans la classe
    protected $adresse; // accessible a l'interieur et classes filles

    // Métodes
    public function xx()
    {
        $val = $this->id + 1;
        return $val;
    }
}
```

  - Sur un objet $obj->prop; $obj->xx();

```php
$unObjet = new NomDeLaClasse();
echo $unObjet->nom;
```

- Constructeur : public function __construct()

```php
public function __construct($id){
    $this->id = $id;
}
```

- Destructeur : public function __destruct()
- Constante de classe const nom = valeur;
  Utilisation dans la classe self::nom

```php
class Ex0 {
    const DEFAULT_DATE = '2016-12-01';

    public function getDefaultDate(){
        return self::DEFAULT_DATE;
    }
}
```

  Utilisation en dehors nomClasse::nom

```php
$date = Ex0::DEFAULT_DATE;
```

- Définition d'une méthode

```php
class Telephone {
    public function quiSuisJe(){
        echo "<br/>Je suis un telephone";
    }
}
```

- Définir des accesseurs (getters et setters) pour accéder et protéger les données

```php
class Membre
{
    private $pseudo;
    private $email;
    private $signature;
    private $actif;

    public function getPseudo()
    {
        return $this->pseudo;
    }


    public function setPseudo($nouveauPseudo)
    {
        // Vérifier si le nouveau pseudo n'est ni vide ni trop long
        if (!empty($nouveauPseudo) AND strlen($nouveauPseudo) < 15)
        {
            // Ok, on change son pseudo
            $this->pseudo = $nouveauPseudo;
        }
    }
}
```

- Attributs et méthodes statiques
  L'effet 'static' : l'information ou le comportement appartiennent à la classe et sont indépendants des objets

```php
class Ex1 {
    static private $count;

    public function __construct(){
        $this::$count ++;
    }
    public function __destruct(){
        $this::$count --;
    }
    static public function getCount(){
        return self::$count;
    }
}
```

```php
$ob1Ex1 = new Ex1();
$ob2Ex1 = new Ex1();
echo "<br/>getCount = " . Ex1::getCount();
```

Etre vigilant sur $this::$val self::$val

- Héritage

```php
class Telephone {
    public function quiSuisJe(){
        echo "<br/>Je suis un telephone";
    }
}
class Smartphone extends Telephone {
    public function quiSuisJe(){
        echo "<br/>Je suis un telephone intelligent";
    }
}
class SmartphonePlus extends Telephone {
    public function quiSuisJe(){
        echo parent::quiSuisJe() . " plus intelligent";
    }
}
```

```php
$phone = new Smartphone();
$phone->quiSuisJe();
$phone = new SmartphonePlus();
$phone->quiSuisJe();
```

Je suis un telephone intelligent
Je suis un telephone plus intelligent

Exercice 11

Symfony 7

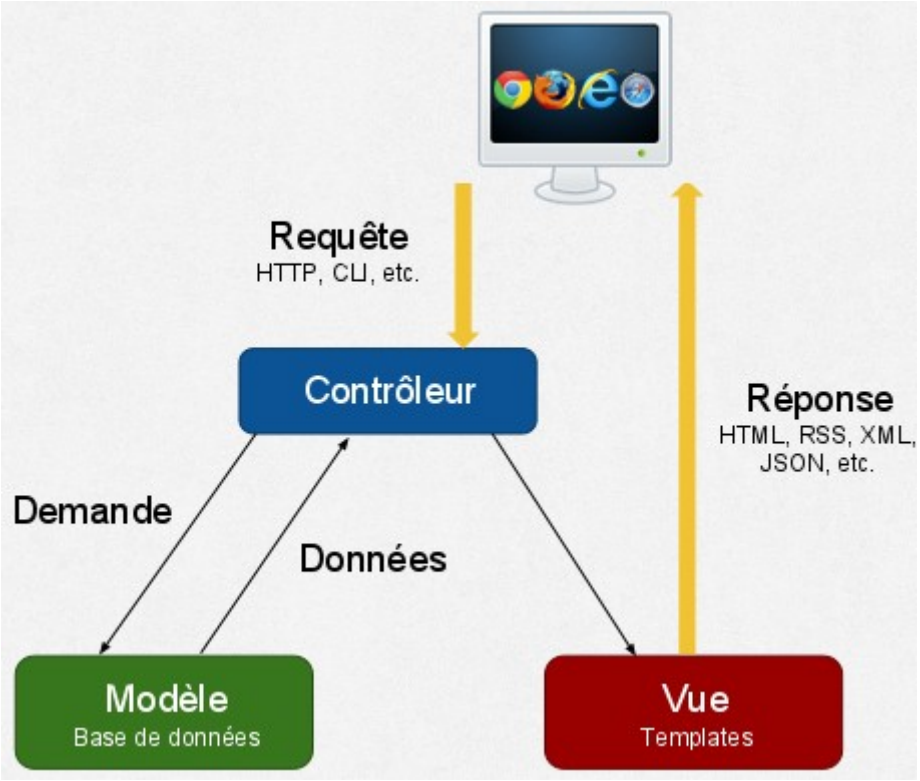
- Utiliser le fichier fourni InstallWampVSCode pour installer Wamp et l'éditeur VSCode

- Utilisation de Composer : gestionnaire de dépendance. Permet de chercher et installer des modules PHP

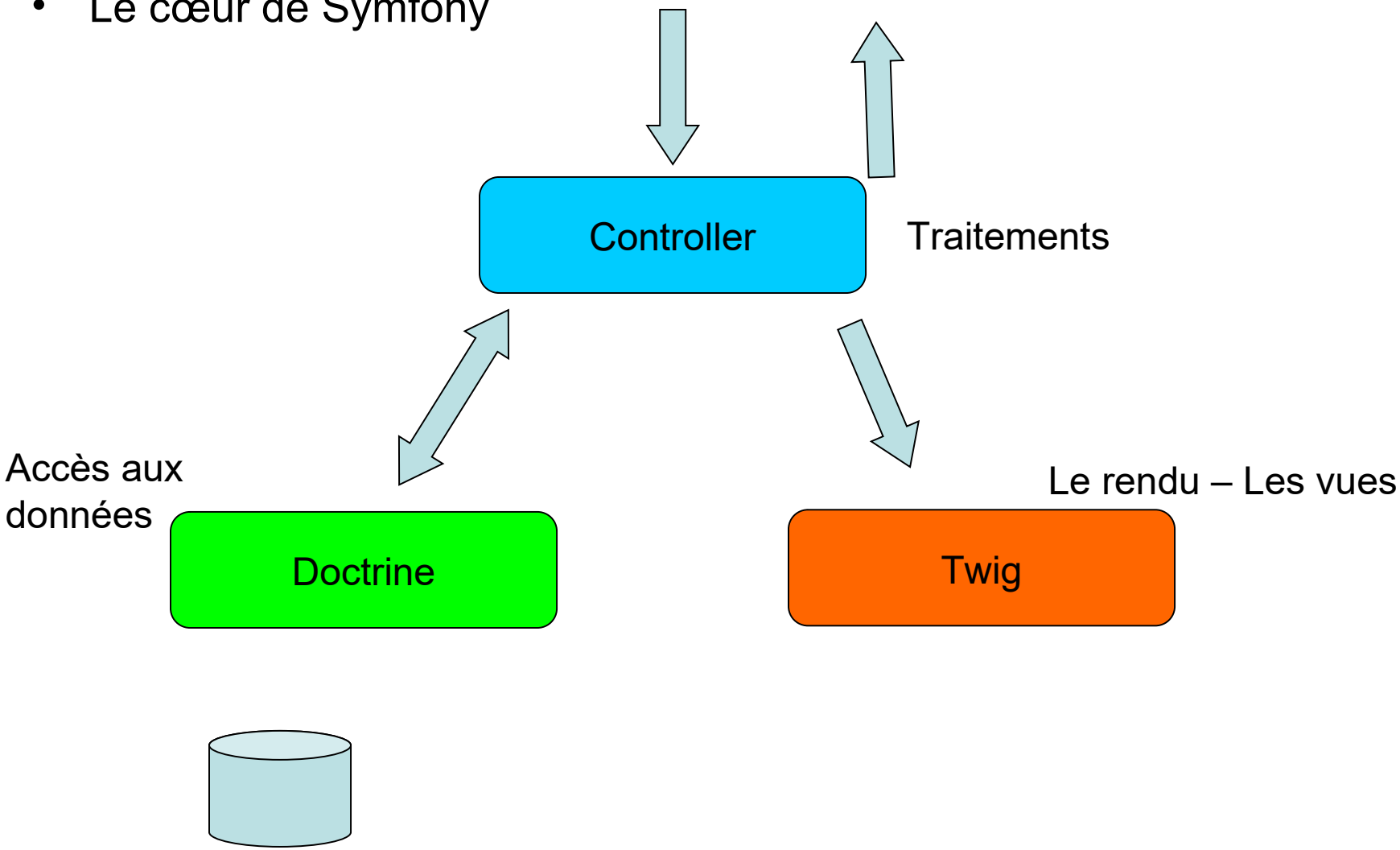
Symfony 7

- Framework PHP créé par la Sté SensioLabs pour offrir un support de dev Web efficace et performant
- Utilisé par les framework Drupal, Laravel, Joomla!, Composer, Prestashop …
- Le milliard de téléchargements en 09/2017
- Compatibilité Symfony et PHP :
  - Symfony 1.x , PHP >= 5.2.4
  - Symfony 2.x , PHP >= 5.3.3
  - Symfony 3.x , PHP >= 5.5.9
  - Symfony 4.x , PHP >= 7.1.3
  - Symfony 5.x , PHP >= 7.2.5
  - Symfony 6,x , PHP ≥ 8.0
  - Symfony 7,x , PHP ≥ 8.2
- Implémente le modèle MVC pour une maintenabilité et évolutivité
- Système de configuration par fichiers YAML, XML, PHP, Annotations
- Internationalisation native
- Architecture extensible

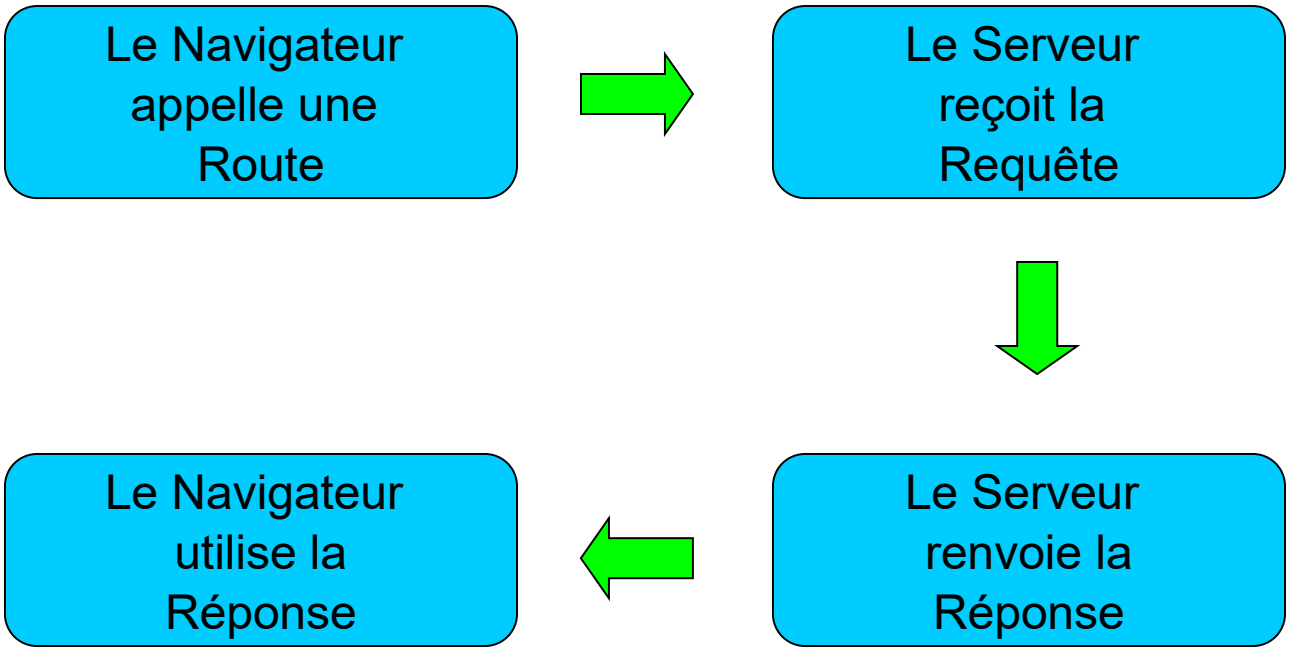
- Le but du framework est de formuler côté serveur une réponse consécutive à une Demande (Request) côté navigateur.
- Symfony utilise le modèle MVC



- Le cœur de Symfony



- La logique du Controller :
  - Écouter la requête http et extraire la Route
  - Fabriquer une réponse
  - Renvoyer la réponse au navigateur



- Le code des Controllers se trouvent dans src/Controller

- Création d'un Controller, en cmde :
  - php bin/console make:controller
  - Fournir un nom de classe <XX>Controller (ex BlogController)

- Simplicité
  - Facilité d'écriture (boucle, condition, filtres …)

- Absence de PHP
  - Clarifie le code
  - Partage de travail dans une équipe (développeurs / désigners)

- Documentation claire sur https://twig.symfony.com

- Les vues twig sont regroupées dans templates

Commençons le projet !

- Quelques éléments :
  - {# …..#} commentaire multi lignes
  - {{ var.b }} récupération variable passée par le controller
  - {% set x =‘abcd’ %} affecter une variable
  - {% fonction %} appelle une fonction

```
{% apply upper %}
    This text becomes uppercase
{% endapply %}
```

  - {% structure de contrôle %}

```
{% if users|length > 0 %}
    <ul>
        {% for user in users %}
            <li>{{ user.username|e }}</li>
        {% endfor %}
    </ul>
{% endif %}
```

```
{% for box in boxes %}
    {{ include('render_box.html') }}
{% endfor %}
```

- Le mécanisme d'héritage de template

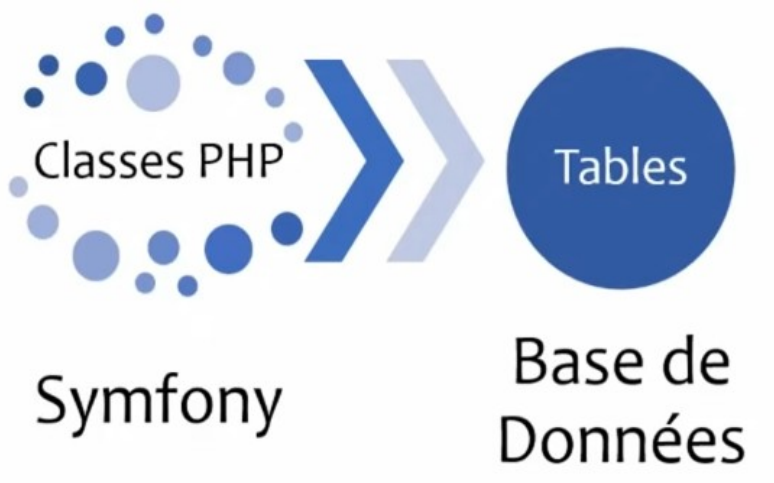
Base.html

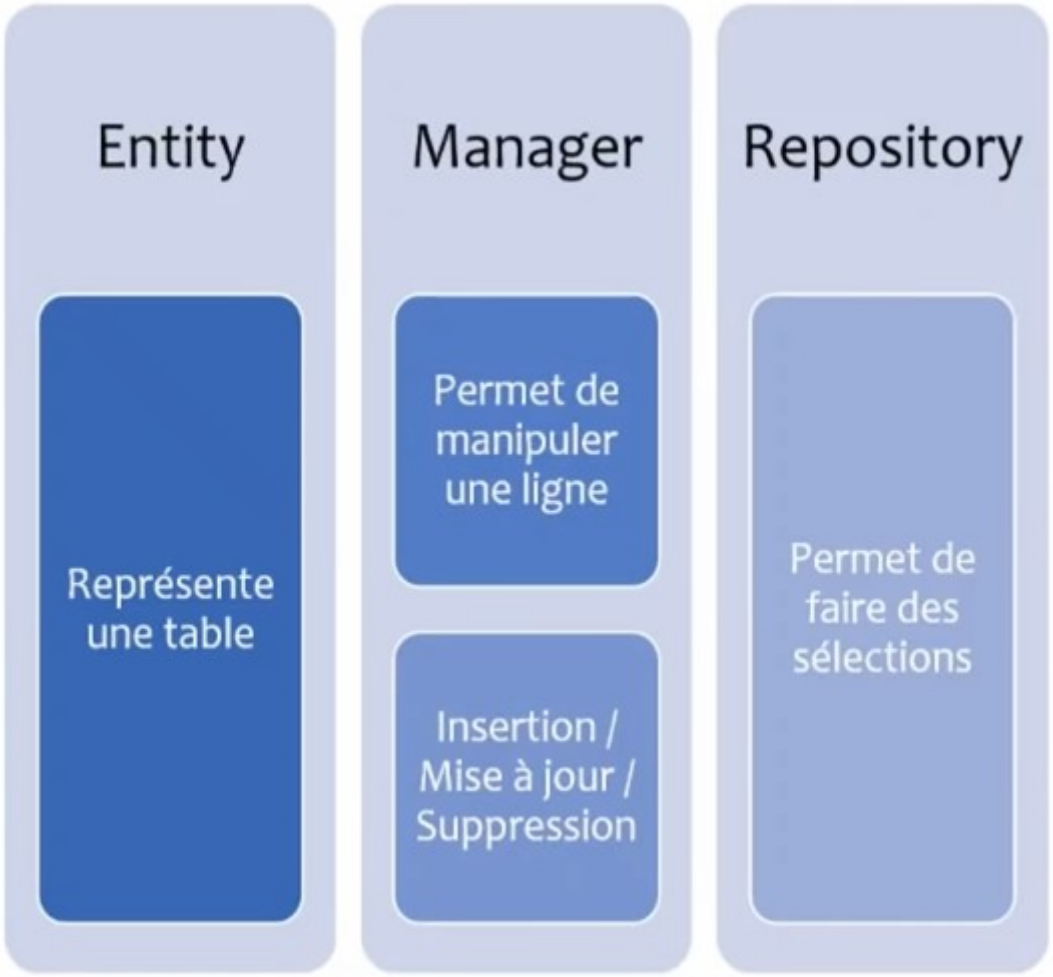
```
<!DOCTYPE html>
<html>
    <head>
        {% block head %}
            <link rel="stylesheet" href="style.css" />
            <title>{% block title %}{% endblock %} - My Webpage</title>
        {% endblock %}
    </head>
    <body>
        <div id="content">{% block content %}{% endblock %}</div>
        <div id="footer">
            {% block footer %}
                &copy; Copyright 2011 by <a href="http://domain.invalid/">you</a>.
            {% endblock %}
        </div>
    </body>
</html>
```

```
{% extends "base.html" %}

{% block title %}Index{% endblock %}
{% block head %}
    {{ parent() }}
    <style type="text/css">
        .important { color: #336699; }
    </style>
{% endblock %}
{% block content %}
    <h1>Index</h1>
    <p class="important">
        Welcome to my awesome homepage.
    </p>
{% endblock %}
```

- C’est un ORM : Object Relational Mapping
- Fait le lien entre une application et une base de données

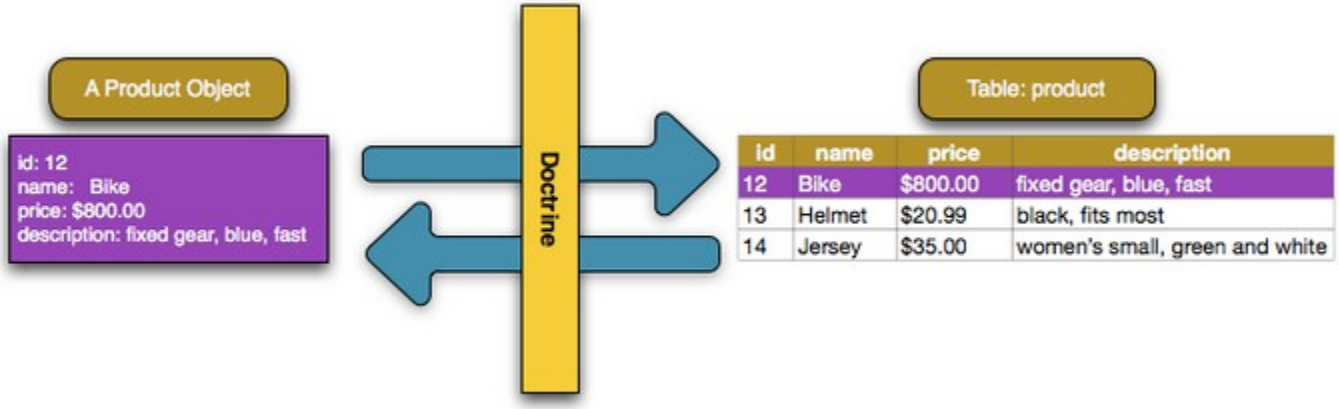



Entity
Représente une table
Manager
Permet de manipuler une ligne
Insertion / Mise à jour / Suppression
Repository
Permet de faire des sélections

- Entity

https://symfony.com/doc/current/doctrine.html#migrations-adding-more-fields

  - Une Entity est une classe qui représente tous les champs (colonnes) d'une table en base de données
  - Définie dans src/Entity avec l'espace de nom App\Entity



  - La commande

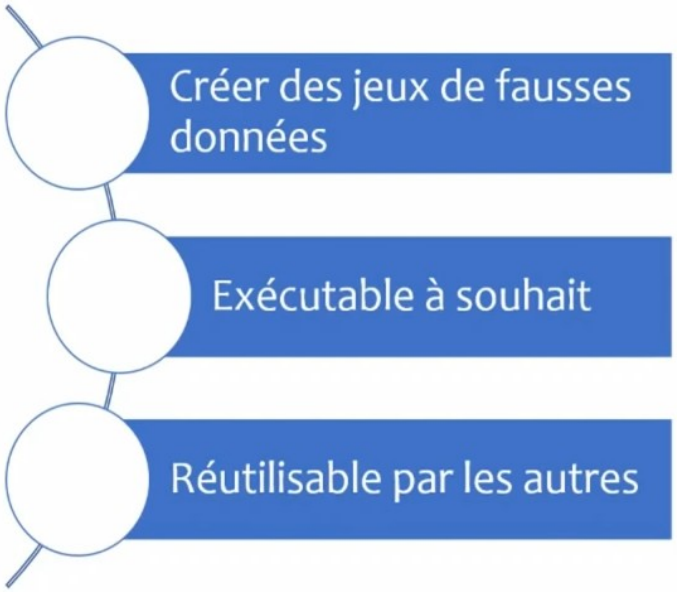
    php bin/console make:entity

    permet de créer automatiquement le code de cette classe et le code du repository (la classe qui permet d'obtenir un 'miroir' en BD)

- Les migrations dans Symfony
  https://symfony.com/doc/current/bundles/DoctrineMigrationsBundle/index.html
  - Ensemble de fichiers exécutés dans un ordre précis

  - Permet à une copie du projet à retrouver un même contenu de base de données
  - La cmd : **php bin/console make:migration** compare le code et la BD et écrit un fichier de migration dans src/Migrations
  - La cmd : **php bin/console doctrine:migrations:migrate** exécute les fichiers de migration et met à jour la BD

- Les Fixtures : création de jeux de données factices dans la base de données



  - **composer require orm-fixtures --dev** pour importer le module
  - **php bin/console make:fixtures** pour créer une classe et fichier fixture
  - **php bin/console doctrine:fixtures:load** pour exécuter le fichier

- Symfony dispose d'un Service Container
  https://symfony.com/doc/current/service_container.html

  php bin/console debug:container pour la liste des services

- Symfony est un ensemble de services administrés par ce Service Container

- Il permet d'allouer automatiquement des objets et de fournir les bons paramètres. Ce mécanisme s'appelle l'injection de dépendances.

- L'injection de dépendances va faire de l'introspection de code et permet ici :
  - De disposer d'un objet implémentant list()
  - De trouver et passer en paramètre un objet de classe LoggerInterface

```php
// src/Controller/ProductController.php
// ...

use Psr\Log\LoggerInterface;

/**
 * @Route("/products")
 */
public function list(LoggerInterface $logger)
{
    $logger->info('Look! I just used a service');

    // ...
}
```

- Symfony 7 implémente les mécanismes de sécurité sous la forme suivante :